# 摘 要

基于博弈的独立集和顶点覆盖问题研究

现实世界中的许多复杂系统都可以抽象为复杂网络的表达形式，复杂网络可以用于描述系统内个体与个体之间、个体与群体之间和群体与群体之间的相互关系。复杂网络规模庞大、结构错综复杂，很难用肉眼直接观察到有规律的信息，但是，不同的网络又具有一定的共性，例如人类社会中的人际关系网络、电力网络、科学家著作合作网络等，都使用网络的节点来抽象地表示系统中的每一个个体，以节点之间的连边来表达不同个体之间的联系，通过运用复杂网络的有关分析方法，可以发现整个系统所具有的规律，有助于学者们对相关问题的研究。现今社会，随着互联网的飞速发展，事物之间的联系越来越密切，复杂网络作为真实社会中不同事务关系的载体，正需要更深入的研究与探索。所以，研究复杂网络，对于促进人类社会和科学的进步具有深远的意义。

复杂网络上的最大独立集问题(MISP)和最小顶点覆盖问题(MVCP)是复杂网络研究领域的两个典型的集合优化问题，这两个问题之间具有密切联系，同时，又有助于研究复杂网络领域的其他问题。最大独立集问题指的是求解一个（或若干个）网络节点集合，所求得的节点集合规模最大、且集合内的节点相互之间不存在连边。该优化问题得到的是整个网络中个体之间互不影响的最大的个体集合。最小顶点覆盖问题的目的在于找出给定复杂网络的规模最小的节点集合，并保证网络中任意一条边至少有一个顶点在所选的节点集合中。现实生活中的许多问题可以归结为这两种问题，比如高校教师排课系统、监控设备安置问题、线路规划问题、网络优化问题等。作为NP难问题，虽然对这两个问题的研究历史已久，但已存在的解决这两个问题的算法仍有不足之处，我们创新性地结合复杂网络的演化博弈动力学，分别对局部搜索和全局搜索方法进行了研究、改进与实验，提出了基于囚徒困境博弈求解MISP的IGLS算法和基于雪堆博弈求解MVCP的GMA-MVC算法。通过大量的实验，验证算法在解决这两个问题上的表现，验证算法的有效性。本文的主要工作如下：

1．用于求解MISP的基于囚徒困境博弈的迭代局部搜索算法(IGLS)。该算法是在传统的迭代局部搜索算法的基础上，基于复杂网络的博弈演化规律，创新地提出基于复杂网络博弈演化与局部搜索机制，可以准确而快速地搜索到局部最优解并对该局部最优解进行一定程度的改进。借鉴传统的迭代局部搜索思路，引入扰动机制，以一定的概率跳出局部最优状态，扩大搜索范围，尽可能地搜索到全局最优解。该算法可以同时实现(1,k)-swap的局部搜索，改进了已经存在的步骤繁琐的局部搜索方式，实验表明，该算法在大多数网络上可以搜索到全局最优解。

2.基于雪堆博弈求解MVC的自然进化算法。我们借鉴我们所研究的网络的囚徒困境博弈与MISP的联系，建立了雪堆博弈与最小顶点覆盖问题的关系。将网络的节点状态（被覆盖或未被覆盖）集合看作进化算法的个体（染色体），不同个体代表不同的局部最优解，与传统的进化算法不同，在该算法中，群体中每一个个体都要通过进行基于囚徒困境博弈的个体进化而收敛到局部最优解，并对当前局部最优进行一定程度的改进。当所有个体完成个体进化后，再进行传统的自然进化过程，完成对全局最优解的搜索。通过实验对比，该算法的性能优于当前存在的经典算法。

**关键词：**复杂网络，独立集，节点覆盖，囚徒困境博弈，雪堆博弈，自然进化

# ABSTRACT

Keywords:

# 目 录

[摘 要 1](#_Toc503384465)

[ABSTRACT 3](#_Toc503384466)

[目 录 i](#_Toc503384467)

[第一章 绪 论 1](#_Toc503384468)

[1.1 复杂网络概述 1](#_Toc503384469)

[1.1.1 复杂网络的基本概念 2](#_Toc503384470)

[1.2 复杂网络研究现状 2](#_Toc503384471)

[1.3 复杂网络研究意义 4](#_Toc503384472)

[1.4 复杂网络的集合优化问题 5](#_Toc503384473)

[1.5 本文结构安排 6](#_Toc503384474)

[第二章 问题描述及相关算法简介 9](#_Toc503384475)

[2.1 最大独立集问题 9](#_Toc503384476)

[2.2 最小顶点覆盖问题 10](#_Toc503384477)

[2.3 求解最大独立集的局部搜索算法简介 10](#_Toc503384478)

[2.3.1 基于2-improvement的迭代局部搜索算法ILS34 10](#_Toc503384479)

[2.3.2 基于交换的禁忌局部搜索算法SBTS35 12](#_Toc503384480)

[2.3.3 求解最大团的局部搜索算法BLS36 12](#_Toc503384481)

[2.4 最小顶点覆盖问题算法简介 13](#_Toc503384482)

[2.4.1 求解最小顶点覆盖的遗传算法GA33 13](#_Toc503384483)

[2.4.2 混合遗传算法HGA38 14](#_Toc503384484)

[2.4.3 基于粗糙集的算法VCAR39 15](#_Toc503384485)

[2.4.4 基于边权值的局部搜索算法NuMVC37 17](#_Toc503384486)

[第三章 复杂网络上的演化博弈 19](#_Toc503384487)

[3.1 演化博弈 19](#_Toc503384488)

[3.1.1 囚徒困境博弈 19](#_Toc503384489)

[3.1.2 雪堆博弈 21](#_Toc503384490)

[3.1.3 纳什均衡 21](#_Toc503384491)

[3.1.4 博弈策略更新方法 22](#_Toc503384492)

[3.2 复杂网络的同步演化博弈 23](#_Toc503384493)

[3.2.1 同步演化博弈 23](#_Toc503384494)

[3.2.2 基于同步雪堆博弈的MBR43算法 23](#_Toc503384495)

[3.3 复杂网络的异步演化博弈 28](#_Toc503384496)

[3.3.1 异步演化博弈 29](#_Toc503384497)

[3.3.2 异步囚徒困境博弈与最大独立集 30](#_Toc503384498)

[3.3.3 异步雪堆博弈与最小顶点覆盖 32](#_Toc503384499)

[3.4实验结果与分析 33](#_Toc503384500)

[第四章 基于PDG求解最大独立集的局部搜索算法 35](#_Toc503384501)

[4.1 基于囚徒困境博弈的局部搜索机制GLS 35](#_Toc503384502)

[4.1.1 基于囚徒困境博弈求得极大独立集 35](#_Toc503384503)

[4.1.2 基于囚徒困境博弈的局部搜索 35](#_Toc503384504)

[4.1.3 GLS的博弈顺序 39](#_Toc503384505)

[4.1.4 基于博弈的扰动方法 40](#_Toc503384506)

[4.2 基于囚徒困境博弈的迭代局部搜索算法IGLS 41](#_Toc503384507)

[4.2.1 弱扰动机制 41](#_Toc503384508)

[4.2.2 基于模拟退火思想的解集更新方法 42](#_Toc503384509)

[4.2.3 算法整体思路 42](#_Toc503384510)

[4.3 实验结果与分析 43](#_Toc503384511)

[4.3.1 实验网络简介 43](#_Toc503384512)

[4.3.2 不同算法搜索合法解的性能对比实验 45](#_Toc503384513)

[4.3.3 局部搜索性能对比 47](#_Toc503384514)

[4.3.4 综合对比实验 49](#_Toc503384515)

[4.3.5 实验结果分析与统计检验 55](#_Toc503384516)

[第五章 基于雪堆博弈求解最小顶点覆盖的自然进化算法 57](#_Toc503384517)

[5.1 个体进化 57](#_Toc503384518)

[5.1.1 基于雪堆博弈求解合法解 57](#_Toc503384519)

[5.1.2 基于雪堆博弈的局部搜索 57](#_Toc503384520)

[5.1.3 个体进化总体步骤 59](#_Toc503384521)

[5.2 带有个体进化的自然进化算法GMA-MVC 59](#_Toc503384522)

[5.2.1 基于节点度的初始化方法 59](#_Toc503384523)

[5.2.2 GMA-MVC算法元素 59](#_Toc503384524)

[5.2.3 GMA-MVC算法总体步骤 61](#_Toc503384525)

[5.3 实验结果与分析 62](#_Toc503384526)

[5.3.1 实验网络简介 62](#_Toc503384527)

[5.3.2 初始化的作用 62](#_Toc503384528)

[5.3.3 个体进化的作用 63](#_Toc503384529)

[5.3.4 基于演化博弈的不同算法对比 64](#_Toc503384530)

[5.3.5 综合对比实验 65](#_Toc503384531)

[5.3.6 实验结果分析与统计检验 67](#_Toc503384532)

[5.3.7 算法参数分析 68](#_Toc503384533)

[第六章 总结与展望 71](#_Toc503384534)

[6.1 总结 71](#_Toc503384535)

[6.2 展望 71](#_Toc503384536)

[致 谢 72](#_Toc503384537)

[参考文献 73](#_Toc503384538)

# 第一章 绪 论

## 1.1 复杂网络概述

复杂网络是一门多学科相互交叉、相互融合而形成的一门交叉学科，从上世纪九十年代起，已经逐渐发展成为一类研究热点。复杂网络的主要作用是用形象化的网络描述方式对现实社会中的复杂系统进行描述，以有助于直观地研究复杂系统的统计特性或演化规律。复杂网络广泛存在于我们的日常生活和工作中，例如，人类社会中的每一个个体都需要与外界进行沟通和交流，这就会构成属于每一个人的人际交往网络，整个社会中存在着庞大的人际网[[1]](#endnote-1)，再比如，不同生物群体构成的交互网络、互联网、计算机病毒传播网络、传染病传播网络、食物网[[2]](#endnote-2)、交通运输网络、生物机体代谢网络等，都属于复杂网络。

复杂网络问题起源于解决小规模网络问题，例如“七桥问题”[[3]](#endnote-3)[[4]](#endnote-4),欧拉为解决“七桥问题”，首次提出了网络拓扑结构，将每座桥抽象为边，将桥连接的地方抽象为顶点。后来，出现了ER随机网络模型，并最终确立了ER随机图理论[[5]](#endnote-5)[[6]](#endnote-6)。为了形象具体地描述网络的一些特性，将图论中的“图”用于描述网络的结构特性，将网络中的节点抽象化为图中的节点，并用图中一个节点与一个节点之间的联系抽象地表示网络中节点之间的联系。由此可见，为了更好地研究网络，需要借助于图的拓扑结构与性质，这样有助于我们更加直观形象地解释并理解网络的结构与性质。

到1998年，Watts及其老师Strogatz发表文章——《Collective dynamics of small-world networks》[[7]](#endnote-7)，该文章首次提出了WS模型，并提出了复杂网络具有小世界特性这一观点。现实中，复杂网络的存在主要是在信息流通方面，而网络又具有小世界特性。小世界网络的特点是平均路径长度较短，并具有较高的聚类系数，所以，这种模型代表了现实世界中复杂网络传播信息的一种非常有效的模型[[8]](#endnote-8)。目前还没有精确地描述小世界网络的定义，Watts和Strogatz提出的小世界网络通过调整相应的参数，可以实现从规则网络到随机网络的过度7，这种是WS小世界网络，另一种小世界网络由Newman和Watts提出，称为NW小世界网络，WS小世界网络采用随机化重连NW小世界网络与其不同，它采用随机化加边。总的来讲，这两类小世界网络模型都具有小世界特性，而网络的小世界特性用两个网络参数来描述，这两个网络参数是：平均路径长度和聚类系数。

1999年，Barabasi和Albert发表文章——《Emergence of scaling in random networks》[[9]](#endnote-9),该文章阐述了网络具有无标度特性，并建立了一个无标度网络，无标度网络的显著特点就是节点的度分布服从指数为3的幂律分布。

复杂网络的研究开始进入全新的研究领域就是由小世界网络模型和无标度网络模型的提出开始的，这两类复杂网络模型各具特点，极大地促进了复杂网络领域的进步与发展8，使得人们对复杂网络的研究进入一个新纪元。

近年来，随着计算机以及通信网络的飞速发展[[10]](#endnote-10)，网络规模越来越大，急需更多的关于大型网络统计特性的研究，这都推动了复杂网络领域的进步与发展。

### 1.1.1 复杂网络的基本概念

在复杂网络中，有一些特定的网络名词[[11]](#endnote-11)[[12]](#endnote-12)，这些名词在复杂网络的研究领域中，具有不同的含义，本小节将给出这些网络名词的详细解释。

（1）节点、边。一个最基本的复杂网络由节点和边这两个元素构成，将网络中的每一个个体看作一个独立的节点，个体之间的相互联系产生连边，即，如果两个个体之间具有某种联系，那么就说这两个节点之间具有连边。边可以携带不同的网络信息，比如，边可以携带权值信息，用于衡量这两个个体之间的关联关系的重要程度，边也可以是有方向的，进而构成有向网络。

（2）图。图是一种形象化的表示方式，通常情况下，对于一个具有N个节点，M条连边的网络，设网络节点集合为，网络边的集合为，其中，表示网络中第个节点，表示网络中第条边。那么可以用图来表示该网络。

（3）补图。如果用图的形式来形象地表达复杂网络，那么就可以用图的补图表达节点与原网络节点相同、节点之间有连边当前仅当他们在原网络中没有连边。

（4）路径。在网络中，连通任意两个节点、的边的集合，构成一条节点、之间的路径。两节点之间可能存在多种路径，经典的最短路径问题就是求解两点之间的最短路径，在现实生活中具有重要的实际意义。

（5）邻居。在一个网络中，如果两个节点之间存在连边，那么这两个节点互相成为对方的邻居节点。

（6）度。度是网络中每一个节点所具有的一种统计特性。在无向图中，节点的度定义为与该节点相连的边的个数。在有向图中，度分为入度和出度，节点的入度表示的是直接指向该节点的连边的各式各样，节点的出度表示的是从该节点出发、指向其他节点的边数。

（7）度分布。在一个复杂网络中，任意一个节点的度值为的概率为，衡量了一个网络中节点分布的规律，不同类型的网络，度分布规律不同。

## 1.2 复杂网络研究现状

目前，对于复杂网络的研究已经有较为成熟的统计特征参数和标准，研究复杂网络通常是将任意一个系统都简化并分解为节点和节点相互之间的连边，然后再研究其结构。有三个参数用来衡量一个网络的结构特点，这三种参数分别为平均路径长度、聚类系数和度与度的分布，这三类特征共同刻画一个特定类型的复杂网络。

平均路径长度描述了复杂网络中节点与节点之间的距离特征[[13]](#endnote-13)，通过这个参数，我们就可以知道系统中各节点之间联系的紧密程度，从而获得系统的整体结构特征。

网络中的节点聚集情况用聚类系数来描述，网络中任意一个节点的聚类系数是节点的所有邻接节点之间实际存在的边数与这些邻接节点全连通时的总边数之间的比值，而整个网络的聚类系数是将网络中所有节点的聚类系数求平均得到的。举个例子来说明一下，我们在社会交往中，我们每个人都与自己的朋友有着直接的联系，但还存在一种情况，我们的朋友之间也可能是朋友，朋友之间真实存在的联系关系数与可能存在的最大联系关系数的比值，就表示了这个朋友圈网络的聚类系数，以此可以衡量这个朋友圈联系的紧密程度。通过聚类系数的定义我们可以看出，聚类系数可以用来描述复杂网络的节点间关系的紧密程度，用聚类系数0来表示孤立节点的聚类系数，用聚类系数1表示全连通的节点的聚类系数13。在复杂网络中，一个节点有n个邻居节点，那么我们就称这个节点的度为n,而度分布描述的则是这个节点有k个邻居节点的概率[[14]](#endnote-14)。由上可得，这三个参数能够全面地刻画一个复杂网络的结构特性。以下两个方面是研究复杂网络的重点：

（1）理解并掌握复杂网络的拓扑结构和性质。

（2）分析理解网络上的动力学行为和网络上的演化过程。

对于复杂网络的研究，理解并掌握网络的拓扑结构及不同拓扑结构的网络性质，这有助于我们进一步理解复杂网络的动力学行为，并解释这些动力学行为，网络上的系统动态性质指的就是复杂网络的动力学行为[[15]](#endnote-15)。在复杂网络上的动力学系统同步方面，目前也已经有成熟的研究成果，主要包括复杂网络同步方面的稳定性分析、具有复杂网络性质的系统在同步方面的特点、和复杂网络动力学同步的稳定性的影响因素这几个方面[[16]](#endnote-16)。同时，在复杂网络研究方面，研究热点突出，近几年里在复杂网络研究方面比较热门的是复杂网络社区检测，随着互联网的发展，我们的生活依赖于网络，我们每一个人都离不开社交网络，社区检测涉及到的数据量大，结构复杂，借助于“图”的拓扑结构，将社区检测转化为研究节点与节点之间的边的关系，简化社区检测问题，在发现社区结构特点和性质方面取得了很大的进展，并由此提出了许多有效的社区检测算法。

另一方面，针对复杂网络本身，网络的鲁棒性研究也取得很大成果，鲁棒性用于衡量一个网络结构在遭到破坏时，网络仍能继续正常运行的能力，所以，一个稳定性好的复杂网络应当具有较高的鲁棒性[[17]](#endnote-17)。在国内，对于复杂网络的研究逐步成熟，2003年朱涵17介绍了国内复杂网络的研究成果，主要从网络的小世界性、集团性和无标度性等方面阐述复杂网络；随后，对无向网络、加权网络和有向网络等不同网络结构的网络进行了总结概括，这主要是吴金闪[[18]](#endnote-18)等学者从新的角度研究复杂网络，应用统计物理学，在复杂网络研究方面取得新突破，对规则网络、小世界网络、无标度网络、完全随机网络等不同网络模型进行了概括性的总结；目前，网络统计特性方面的研究也取得进展，刘涛13等学者分别阐述了网络的平均路径长度、聚集系数和度与度的分布这三类描述网络特征的参数。

另一种应用广泛的网络模型是无标度网络模型（BA网络模型），这种网络模型的演化模型由Barabasi和Albert进行了详细阐述9，在现实中的真实系统中，可以通过自组织生长生成这种无标度网络，在这方面，增长性和择优连接性14主要会影响网络的生成。李翔[[19]](#endnote-19)提出的观点是择优连接性并不是对于整个网络都适用的,作者进一步提出了“局域世界演化模型”，该模型不仅保持了无标度网络的较强的鲁棒性，而且还改进了存在恶意攻击时，无标度网络所固有的脆弱性。陈庆华[[20]](#endnote-20)等人提出的观点主要是重新连接网络中的各个节点，进而实现网络，这被称为BA网络模型的拓展模型。陈禹[[21]](#endnote-21)等人研究BA网络模型，提出更具创新的观点，他认为，虽然BA网络模型可以很好的反应显示世界中网络的基本性质，但是现实情况远远比假设的情况要复杂很多，所以BA网络模型还过于简化，在此基础上，可以得到基于BA网络模型的三种扩展的网络模型。前人的这些研究都对以后研究网络结构模型提供了丰富的经验。

当然，对于复杂网络还要进行更深更远的研究，目前，在研究复杂网络方面也遇到了许多困难。在复杂网络的理论发展方面，关于复杂网络的一些基本理论还不成熟，在以后的研究中，还需要更加严格更加系统地给出网络科学中的概念及相关规律。比如说，当前在复杂网络理论研究方面，幂律分布[[22]](#endnote-22)的定义还没有统一确定地给出，现在如果要判断一个分布是否服从幂律分布，大多数是观察双对数坐标下的分布图像是否是一条直线，而这并不是经过严格数学论证得到的，所以，在这方面还需要得到严格的科学定义，这样才能保证所进行的研究是严格而科学的。

## 1.3 复杂网络研究意义

我们身边的交通网络、人际交往网络都是与我们紧密联系的，比较显而易见的是，对于交通网络的研究，有助于帮助人们了解在哪个时刻车流量大，以有效控制交通，实现城市交通的正常化运行；也可以帮助人们得到某些城市之间的最佳路径，进而节约人工成本。还有一些网络是在现实生活中不易观察到，但是却也非常重要的，例如，蛋白质结构网络、人的大脑神经网络、传染病传播网络、电力网络、移动通信网络、计算机病毒传播网络等8，研究这些网络有助于我们通过结构预测网络的发展或者遏制网络中不良因素的扩展，防控传染病和病毒的传播，从而避免一些不必要的人力、物力损失。复杂网络分析会在社会科学领域和生命科学领域大有作为，对于具体的实际问题和实际系统，例如优化交通系统网络、防控传染病和病毒等问题，通过网络分析，提出具体的解决方案，进而，在社会经济中，实现复杂网络的重要作用8。

复杂网络的研究绝不是仅仅让我们每个人在生活中获益，有关复杂网络的研究，更能够对其他领域各方面的研究提供新的研究思路和科研方法，从而促进整个科研领域的进步。智能电网、物联网等的发展离不开研究复杂网络作为技术支持，真实的网络是在自然、社会和工程中存在的，通过研究网络结构，可以对现实中存在的网络的网络结构及演化规律做出科学的分析与理解，并根据分析结果建立起来基于网络的不同类型的网络模型，逐渐在这一领域构建完善的网络模型，为以后的研究积累知识经验。有了网络模型，我们对其进行数学分析，进一步进行计算和仿真，从而获得网络的本质、性质、和行为规律，进而改造网络，让其朝着对人类与自然有益的方向发展。

## 1.4 复杂网络的集合优化问题

在复杂网络研究领域，有三个比较经典的集合优化问题，分别是最大独立集问题、最小顶点覆盖问题和最大团问题，这三个问题都属于NP-hard[[23]](#endnote-23)组合优化问题[[24]](#endnote-24)，且相互联系。给定一个无权无向图，代表网络节点集合，代表网络边集合，假设表示网络的最大独立集，表示网络的最小顶点覆盖集，表示网络的最大团，那么它们之间的关系[[25]](#endnote-25)[[26]](#endnote-26)是：, 。

最大独立集的研究对许多问题的求解具有重要作用，比如在计算机视觉[[27]](#endnote-27)、图标注[[28]](#endnote-28)、路线规划[[29]](#endnote-29)和社会网络分析等。同时，需要优化问题最终都可以描述为最小顶点覆盖问题，例如，某一地区监控设备的安置问题，在保证完全覆盖该地区所有道路的基础上，以最小化花费为目标来寻求最优的监控安置方案，这一问题就可以通过最小顶点覆盖问题来求解。另外，最小顶点覆盖问题也广泛应用与无线传感网络[[30]](#endnote-30)、电力系统网络[[31]](#endnote-31)、网络鲁棒性评价[[32]](#endnote-32)和各类匹配问题[[33]](#endnote-33)等。

对于最大独立集问题和最小顶点覆盖问题的求解，常见一类算法是分布式算法，首先运用启发式的方法获得局部最优解，然后运用局部搜索方法不断优化局部最优解，比如求解最大独立集问题的ILS算法[[34]](#endnote-34)、SBTS算法[[35]](#endnote-35)、BLS算法[[36]](#endnote-36)等，求解最小顶点覆盖问题的NuMVC算法[[37]](#endnote-37)、HGA算法[[38]](#endnote-38)、VCAR算法[[39]](#endnote-39)等。第二类是精确算法，精确算法一般在迭代过程中不断运用“修剪”操作来搜索最优解[[40]](#endnote-40)，比如求解最大独立集的算法[[41]](#endnote-41)。第三类是集中优化方法[[42]](#endnote-42)，主要使用整个网络的全局信息，比如经典的自然进化算法，通过全局信息的不断迭代使用来搜索最优解，比如传统的GA算法33。近几年，随着复杂网络演化博弈的发展，出现了基于演化博弈思想来求解最小顶点覆盖问题的新方法，典型的代表是MBR算法[[43]](#endnote-43)，该算法创新性地将网络节点看作完全理性的博弈参与者，将博弈的严格纳什均衡状态中策略为合作的节点组成的集合看作顶点覆盖集，通过引入博弈记忆机制克服演化博弈不收敛的问题，以一种新的视角来研究该类问题。

对于这类NP难问题，解决问题的关键是得到局部最优解，并不断优化局部最优解，尽力搜索全局最优解。因此，求解最大独立集或最小顶点覆盖问题的局部搜索算法或集中式算法都在使用这种思路，我们通过对复杂网络的博弈演化规律的研究，克服了博弈演化在求解这两个问题时的缺陷，运用博弈机制对局部最优解进行优化，以简单的思路可以达到复杂的局部搜索的目的，在这个基础上，提出了IGLS算法和GMA-MVC算法。

针对最大独立集问题，我们首先建立了复杂网络的囚徒困境博弈与网络极大独立集之间的等价关系，然后，在这个理论基础上，提出了基于博弈的局部搜索机制，可以实现一般性的，其中，表示从当前局部最优解中移出个节点，当重新恢复到局部最优解状态时，有个节点加入解集，从而使解集规模变大。而已有的局部搜索机制只能实现、34等，并不能给出一种普遍的优化方法。另外，基于博弈的局部搜索方法本身带有扰动因素，所以，我们提出的局部搜索方法优于现有的局部搜索方法。最后，借鉴迭代局部搜索机制，将我们提出的局部搜索方法作为迭代过程中的一个步骤，依靠简单的扰动操作和基于模拟退火思想的解集更新方式，得到完整的IGLS算法。

对于最小顶点覆盖问题，我们改进传统的遗传算法，首先根据雪堆博弈的严格纳什均衡状态与顶点覆盖集之间的关系，可以得到满足约束条件的合法解，克服了传统遗传算法需要满足问题的约束条件这一缺陷。然后，对种群中的个体进行改进，在传统的遗传算法中，每一个个体都会携带全局信息中的一部分，我们基于博弈思想，提出了个体进化思路，让每一个个体都收敛到合法解范围内，将全局信息的规模缩小为所有合法解构成的集合，同时，每一个个体进行一定程度的局部搜索，达到自我优化的目的。最后，将个体进化加入遗传框架，我们只需要使用最基础的交叉、变异和种群更新操作，就可以得到优于其他算法的GMA-MVC算法。

## 1.5 本文结构安排

研究复杂网络上的演化博弈具有重要的实际意义，在人类社会和自然生活中，博弈广泛存在，而将博弈规律与网络结构相结合又可以促进对于网络问题的理解。

复杂网络的最大独立集问题和最小顶点覆盖问题相互联系，又具有不同的处理方法，这类问题的研究对研究其他复杂网络组合优化问题具有很大的帮助。

本文首先研究了囚徒困境博弈和雪堆博弈在复杂网络上的演化规律，提出了异步博弈演化方法，建立了网络博弈与最大独立集和最小顶点覆盖问题之间的联系。然后，运用博弈机制改进传统的求解这两个问题的方法，提出了基于囚徒困境博弈求解最大独立集的迭代局部搜索算法IGLS和基于雪堆博弈求解最小顶点覆盖问题的自然进化算法GMA-MVC。本文共分为六章，各章节主要内容分别为：

第一章为绪论部分，概括性地介绍了复杂网络这一研究领域的基本情况、复杂网络的基础知识以及研究现状，并总结性地介绍文章的主要研究成果与论文的组织结构。

第二章详细介绍了论文中主要求解的问题，以及现有的解决该问题的算法。介绍了已有的求解最大独立集问题的ILS算法、SBTS算法、BLS算法，求解最小顶点覆盖问题的GA算法、HGA算法、VCAR算法和MBR算法。

第三章详细介绍了囚徒困境博弈与雪堆博弈在复杂网络上的演化规律，提出了异步博弈演化方法，并证明了这两类博弈在复杂网络上的博弈规律分别与最大独立集问题和最小顶点覆盖问题之间的联系。通过实验结果，验证我们提出的异步博弈方式在复杂网络演化的优势。

第四章针对已有的求解最大独立集的局部搜索算法，改进其不足之处，基于囚徒困境博弈的演化规律，提出了基于囚徒困境博弈的迭代局部搜索算法IGLS，详细阐述了算法原理及理论证明，最后，将所提出的算法与其他算法进行实验结果对比，验证了所提出的算法的优势。

第五章改进了传统的遗传算法，基于复杂网络的博弈演化规律，提出了基于雪堆博弈的个体进化机制，将个体进化与传统遗传算法相结合进而提出求解最小顶点覆盖问题的GMA-MVC算法，同时，详细介绍了算法原理，通过大量对比试验，验证了所提出的算法的优势。

第六章为本论文工作的总结与展望，概括性地总结了所有工作，并对下一步的研究方向进行展望。

# 第二章 问题描述及相关算法简介

## 2.1 最大独立集问题

已知无权无向网络，代表网络节点集合，代表网络边集合。网络的独立集(Independent set )指的是，该图中任意几个互相不相邻的节点组成的集合。独立集的形式多种多样，例如，单独一个节点就可以构成一个独立集（如图2.1.(a)所示），此时，还可以有其他与该节点不相邻的节点加入该独立集。如果没有任何一个节点可以加入该独立集，那么这个独立集就是极大独立集(the Maximal Independent Set )，极大独立集不是其他任何独立集的子集。在网络的所有极大独立集中，节点个数最多的独立集就是该网络的最大独立集(the Maximum Independent Set )。独立集、极大独立集和最大独立集都可能不是唯一的，可以看到他们之间的包含关系是：。图2.1表示了一个6节点的简单网络上的、和，深颜色节点表示该节点在选中的节点集合中。

 

(a) 单一节点4构成独立集 (b) 节点2、6构成独立集

 

(c) 节点1、4构成极大独立集 (d) 网络的最大独立集

图2.1 网络的、和

## 2.2 最小顶点覆盖问题

已知无权无向网络，代表网络节点集合，代表网络边集合，其中，，表示节点与节点之间的连边。网络的顶点覆盖集(the Vertex Cover Set )是顶点集合的子集，即，且满足对于任意一条边，至少有一个顶点（或）在集合中。一个网络的顶点覆盖集会有多种形式，特殊情况下，网络的节点全集就是规模最大的顶点覆盖集。在所有顶点覆盖集中，集合中节点数最小的集合就是网络的最小顶点覆盖集(the Minimum Vertex Cover Set )，即。图2.2表示了网络的一个顶点覆盖集和最小顶点覆盖集，深颜色节点表示该节点在选中的节点集合中。

 

(a) 顶点覆盖集 (b) 最小顶点覆盖集

图2.2 简单网络的和

从图2.1(d)和图2.2(a)，我们可以看出对于同一个无权无向网络，其最大独立集与最小顶点覆盖集满足。

## 2.3 求解最大独立集的局部搜索算法简介

目前已有的求解最大独立集问题的算法中，表现较好的一类算法是局部搜索算法343536，这类算法在获得初始局部最优解的基础上，基于节点交换的思想，实现或其他交换方式，意思是从当前解集中移出1个节点，同时，该节点的移出会导致其他两个节点进入解集，从而使求得的独立集规模增加1。本小节将简要介绍3种使用局部搜索思路求解最大独立集的算法。

### 2.3.1 基于2-improvement的迭代局部搜索算法ILS34

给定无权无向网络，代表网络节点集合，代表网络边集合，表示节点总数，表示网络的总边数。参考文献34提出了基于的迭代局部搜索（Iterated Local Search based on  ILS）算法。假设当前得到的极大独立集为，这是目前的局部最优解，对于网络中任意不属于的节点(且)，该算法定义了节点的紧密度：等于节点在当前局部最优解中的邻居节点的个数。例如，如果某个节点有3个邻居在解集中，那么此时该节点的紧密度就是3。如果某节点的紧密度为，则称该节点为节点。如果某节点的紧密度为0，则称该节点为自由节点。

在ILS算法中，实现了和操作，该算法分别将和操作记为和，是指从当前局部最优解中移出1个节点，并将2个符合要求的节点移入解集，是指从当前局部最优解中移出2个节点，并将3个符合要求的节点移入解集。可以看出，该算法仅仅可以实现通过一次交换操作，将当前解集规模增加1。

ILS算法中的的具体思路为，根据当前的局部最优解，期望找到该解集以外的两个节点和，使这两个节点代替解集中的节点，要想实现这一操作，那么在当前局部最优解为的情况下，节点、、之间需要满足以下三个条件：

1. 节点和都是节点的邻居节点；
2. 节点和都是节点；
3. 节点和之间没有连边。

条件1）保证了当前解集为极大独立集，即局部最优解，条件2）和条件3）共同保证了使用节点和代替节点后，新得到的集合符合独立集的要求。

ILS算法的重点是快速地寻找可以进行的节点，传统方法是依次遍历解集中的节点，共分为两步：

1. 将节点从解集中临时移出，得到新解集；
2. 如果在新解集的情况下，网络中自有节点的个数小于2，则说明节点肯定不能实现，停止；否则，设节点的邻居节点是解集下的自由节点，将节点 插入解集，得到产生的新解集，检查下是否有一个自有节点，如果有，则将插入，完成一次操作；否则，将节点从解集中移出，处理节点的下一个邻居节点，如果节点仍不能完成操作，则将节点重新移入解集，即恢复最初的解集，然后遍历中的其他节点。

快速实现操作的方法是：对于当前的局部最优解，定义一个执行操作的候选解集，使用中的所有节点产生初始候选解集，如果某个节点的邻居节点的个数小于2，则该节点肯定不能实现操作，所以直接将这种节点从集合中剔除，完成集合的化简，然后，从集合中随机选择一个节点，运用传统方法为该节点执行操作，当成功完成一轮操作后，再更新解集。将这一过程循环执行，知道解集为空时，局部搜索终止。

可以看出，ILS算法实现简单的操作就需要非常繁杂的操作，同时，该算法提出的快速实现操作的方法，也离不开传统方法的支持，所以，不仅算法过程复杂，而且局部搜索能力较弱，一次局部搜索操作只能讲解集规模增大1。这是该算法明显具有的缺陷。

### 2.3.2 基于交换的禁忌局部搜索算法SBTS35

基于交换的禁忌局部搜索（Swap-based Tabu Search SBTS）算法的局部搜索机制是，其中，或大于2的整数，表示的是在当前局部最优解的基础上，将一个不属于该解集的节点强制性地移入解集，然后，根据独立集的定义，在解集中删除这个节点的邻居节点，保证解集是一个合法的独立集，因此，1个节点的移入，可能会导致个节点移出解集。如果，那么解集得到改进；如果，那么解集规模不变，结构发生变化；否则，解集规模都在缩小。SBTS算法将的情况看作对解集的扰动，这样，SBTS算法的多次局部搜索过程本质上等价于带有扰动的迭代局部搜索。SBTS算法为防止节点被循环选中，在局部搜索的同时引入禁忌，当一个节点从当前解集中移出时，给该节点赋予一个禁忌时间，使得该节点在接下来的轮迭代中，状态不再发生变化。SBTS算法的整体过程如下：

1. 随机产生初始局部最优解，初始化禁忌表，迭代次数，最大迭代次数；表示目前搜索到的最优解集，表示中节点的个数，初始时，；
2. 如果存在或，则执行该操作得到新解集，如果，则更新和，即，；否则，执行，；迭代次数加1；
3. 更新禁忌表；
4. 循环执行第2)、3)步，直到。

从总体来看，SBTS算法的操作，只有在的情况下，才能够实现对解集的改进，同时，为了实现操作，该算法同样给出了一些复杂的定义，步骤复杂，本论文这里并未给出该算法实现的具体过程。

### 2.3.3 求解最大团的局部搜索算法BLS36

BLS（Breakout Local Search）算法是一种用于求解网络最大团的局部搜索算法，因为网络最大独立集与最大团之间的关系，求解一个网络的最大独立集，等价于求解该网络的补图的最大团，因此该算法也可以直接用于最大独立集的求解。

BLS算法与SBTS算法35思路相近，都是用禁忌局部搜索的机制。首先使用随机的方法，产生初始合法解，然后，运用局部搜索方法，从一个局部最优解搜索到另一个局部最优解。在进行局部搜索之前，需要遍历整个当前解集中的节点的所有邻居节点，判断是否可以将该节点移入解集并且达到改进解集的目的，如果无法找到满足要求的节点，那么执行扰动操作，BLS算法提出了直接扰动、随机扰动和强制扰动三种扰动方式。BLS算法的禁忌机制与SBTS算法类似，BLS算法为了防止某个移入解集的节点在下一轮局部搜索过程中被移出，因此针对所有移入解集的节点设置禁忌表，使得所有刚刚被移入解集的节点在轮迭代过程中不能再次移出解集。虽然BLS算法和SBTS算法在局部搜索过程中禁忌的对象不同，但其本质都实现了同一个目的，即保证某个节点不被连续执行局部搜索操作。

## 2.4 最小顶点覆盖问题算法简介

最小顶点覆盖问题要求所求出的解集必须覆盖网络中所有的边，因此，求解该问题的算法必须在满足这一约束条件的基础上，搜索最优解。

### 2.4.1 求解最小顶点覆盖的遗传算法GA33

遗传算法（Genetic Algorithm GA）[[44]](#endnote-44)是基于自然进化理论的优化方法。遗传算法通过种群的交叉、变异，利用全局信息，使用适应度函数评价种群中每一个个体的性能，借鉴自然界中的优胜劣汰思想，保留适应度函数较大的个体，因此，当种群经过多次迭代以后，种群不断收敛到每一个个体的适应度函数值都较高的状态，此时，将当前种群中表现最优的个体作为全局最优解。

文献33提出了求解复杂网络最小顶点覆盖问题的遗传算法（GA），种群中的每一个个体都由个元素构成，对应着网络中的个节点，每个元素的取值为0或者1，第个位置取值为1代表网络中节点在顶点覆盖集中，取值为0代表该节点不在顶点覆盖集中。例如，对于图2.2(a)所示的顶点覆盖集，其遗传编码为，表示节点2、3、5、6在顶点覆盖集中。

该算法在种群迭代过程中引入指标，用于评价是否满足最小顶点覆盖问题的约束条件，即判断染色体所代表的顶点覆盖集合是否可以覆盖网络的所有边，指标的计算公式为：

 (2-1)

其中，表示第个染色体的指标值，表示第个染色体所代表的解集未覆盖的网络的边数，是一个不小于网络总边数的正整数。如果染色体代表的解集能够完全覆盖所有边时，那么该染色体的值为最大值。在GA中，每一轮迭代，个体之间两两进行交叉、变异，产生两个新个体，然后在所有个体中，保留指标较大的一半个体，构成下一轮迭代的新种群。如果存在指标相同的个体，则使用轮盘赌选择方法，根据适应度函数，选择适应度值较大的个体，在这里，GA使用的适应度函数为：

 (2-2)

其中，表示第个染色体的适应度函数值，表示第个染色体所表示的顶点覆盖集的规模。

对于给定的无权无向网络，代表网络节点集合，代表网络边集合，表示节点总数，表示网络的总边数。该算法的执行过程如下：

1. 随机初始化种群，种群规模设置为；
2. 种群中个体直接进行交叉、变异；
3. 根据公式(2-1)为种群中每一个染色体赋予值；
4. 使用公式(2-2)计算种群中每个个体的适应度值；
5. 根据个体值的取值范围，确定阈值，阈值等于取值范围的一半，根据阈值，选择值超过阈值的个体；
6. 根据轮盘赌选择方法，从种群中选择较优地个体进入下一代新种群；
7. 重复步骤2)-步骤6)，直到达到最大迭代次数。

该算法使用遗传算法求解最小顶点覆盖问题，存在两个明显的缺陷。首先，该算法无法保证一定可以完全覆盖网络的所有边，初始种群是随机产生的，无法保证对每一条边的覆盖，公式(2-1)只能衡量不同个体之间，哪个个体可以覆盖更多的边，也无法保证被保留在新种群中的个体一定可以覆盖所有边。其次，该算法使用的适应度函数为，这个函数的缺点是，当的取值较大时，该函数无法给出有效的评价值，例如，当时，函数值为0.999955，当时，函数值为1.000000，而在实际实验过程中，所进行实验的网络的顶点覆盖集规模往往较大，因此，该函数无法起到较好的筛选作用。基于这两点缺陷，该算法实际上并不能得到较好顶点覆盖集。

### 2.4.2 混合遗传算法HGA38

混合遗传算法（Hybrid Genetic Algorithm HGA）是将局部搜索方法与遗传算法相结合，HGA算法使用与遗传算法中相同的编码方式，即每一个染色体代表一个解集，染色体由个元素组成，每个元素取值为0或者1，表示与之对应的节点是否在解集中。

HGA算法使用简单的局部搜索机制，对于每一个染色体表示的解集，如果将染色体的某一位编码由1变成0，即将这个位置对应的节点从解集中移出，构成的新解集仍然是一个顶点覆盖集，那么保留该操作，将解集规模减小，完成一次局部搜索。这种简单的局部搜索有可能会产生无用搜索，如果解集恰好是局部最优解，那么任何节点的移出都会导致剩余节点无法构成可行解，所以，针对局部最优解，HGA的局部搜索机制失效。

为了保证种群中的个体是一个合法的顶点覆盖集，HGA算法采用启发式的个体交叉方式，变异操作与传统遗传算法变异操作相同，适应度函数如公式(2-3)所示：

 (2-3)

其中，表示一个染色体。染色体适应度函数值等于其所表示的顶点覆盖集中的节点数。

这里详细介绍HGA的启发式交叉操作，给定无权无向网络，代表网络节点集合，代表网络边集合，表示节点总数，表示网络的总边数。设节点表为，边表为，、分别表示两个父代染色体，表示顶点覆盖集，由染色体可以得到。启发式交叉操作的具体过程为：

1. 初始为空，创建空表、；
2. ，其中且，表示节点在、中出现的频率，表示节点的度。
3. ；
4. 从节点表中选择使得 ，如果存在多个度相同的节点，则选择频率最高的节点，选择节点后，更新顶点覆盖集，更新边表；
5. 重复步骤4)知道边表为空，此时是合法的顶点覆盖集。

HGA算法的总体思想为，设种群规模为，将种群中每两个个体相互进行交叉、变异，产生两个新个体，得到的新个体种群规模为，然后在2个个体中选择较优地个个体，执行局部搜索，然后进入下一轮进化过程。

### 2.4.3 基于粗糙集的算法VCAR39

基于粗糙集求解最小顶点覆盖问题的VCAR算法（the Minimum Vertex Cover algorithm based on Rough Set VCAR）建立在粗糙集理论的基础上，在文献39中，证明了对复杂网络的最小顶点覆盖问题的求解等价于求解粗糙集的属性约减问题[[45]](#endnote-45)[[46]](#endnote-46)，因此，将最小顶点覆盖问题的求解转化为粗糙集的属性约减问题。



图2.3 4节点网络示意图

如图2.3所示，网络，，，该网络的布尔方程表示为：



化简后，，因此可以得到，该网络有两个顶点覆盖集和，是最小顶点覆盖集。

给定一个决策表，如果属性集合满足，且，则集合是决策表的一个约减。

对于给定的无权无向网络，代表网络节点集合，代表网络边集合，表示节点总数，表示网络的总边数。表示网络的关联矩阵，基于粗糙集求解最小顶点覆盖问题的步骤为：

1. 设，，构成关于网络的决策表，信息函数的求解公式如下：

 (2-3)

 (2-4)

1. 初始化图的节点覆盖集，如果图中存在环状路径，则环状路径上的所有节点构成集合；否则，初始化为空集；
2. 分别求决策属性关于集合、集合的正域和
3. 如果，则更新集合、集合，其中，，其中满足且最大；如果，则集合即为顶点覆盖集。

在该算法中，决策属性关于属性集合的正域表示为：

 (2-5)

其中，是由决策属性导出的集合的等价类。表示个体子集关于属性子集的正域。

VCAR算法求解正域和更新集合、的本质是不断将度最大的节点移入解集，直到所有边被覆盖。

### 2.4.4 基于边权值的局部搜索算法NuMVC37

NuMVC算法使用局部搜索方法求解最小顶点覆盖集，在求解过程中，确保满足最小顶点覆盖问题的约束条件。

对于给定的无权无向网络，代表网络节点集合，代表网络边集合，表示节点总数，表示网络的总边数。为阐述NuMVC算法，给出一些需要的名词定义：

1. 解集：表示一个顶点覆盖的候选解，不一定可以覆盖网络的所有边；如果解集可以覆盖网络中所有边，则它是一个可行解。
2. 节点的：从该节点最近一次状态被改变开始，到当前为止，总的局部搜索次数。
3. 边的权值：为无向无权图的每一条边赋予一个正权值，每条边的权值都初始化为1，当完成一轮迭代后，网络中每一条仍未被覆盖的边的权值加1；当网络的平均边权值超过阈值时，采用公式(2-2)更新每一条边上的权值。

 (2-6)

其中，满足。

1. 解集的值：没有被覆盖的边的权重之和。
2. 节点的值：如果节点在候选解中，那么其值表示将节点从候选解中移出后，的减小值，此时；如果节点不在候选解中，那么其值表示将节点移入候选解后，的减小值，此时。其计算公式为：

 (2-7)

NuMVC算法本质上是一种基于边权重的迭代局部搜索算法，边的权重保证算法朝着满足约束条件的方向搜索解集，首先，使用随机方法得到一个满足约束条件的可行解集，设解集规模为，然后从该解集开始，使用局部搜索方法，从解集中移出1个节点，搜索一个规模为的新集合。局部搜索方法为：

1. 从可行解中选择值最大的节点，将从该可行解中移出，得到；
2. 从所有未被覆盖的边中随机选择一条未被覆盖的边，设边的两个顶点分别为节点和节点，从节点和节点中选择值最大的节点，将其加入。

NuMVC算法的整体过程为：

1. 随机产生初始可行解 ，所有边权值初始化为1，将全局最优解初始化为；
2. 计算所有节点的值；
3. 如果解集是可行解，则更新为该解集，然后将值最高的节点从解集中移出，得到；如果解集不是可行解，则将更新为当前解集；
4. 从中移出值最高的节点；
5. 随机选择一条未被覆盖的边，并在的两个顶点中选择值最高的顶点，将该节点移入解集；
6. 所有未覆盖边的权值增加1；如果网络平均边权值超过阈值，则按公式(2-3)更新权值；
7. 重复步骤3)-步骤6)，直到达到最大迭代次数，返回全局最优解。

NuMVC算法最终输出的结果是全局最优解，因为一直保持是可行解，所有算法最终得到的结果满足约束条件，在该算法中，权值更新方法保证了算法在搜索解集的过程中满足约束条件，一次局部搜索过程等价于一次，即从可行解中移出一个节点，同时将另一个节点移入解集，通过局部搜索过程与边权值更新过程的循环迭代，不断搜索较优解集。

# 第三章 复杂网络上的演化博弈

博弈是指在做出决策的时候，参考对方选择的策略，从而选择出对自己而言的最佳策略，以达到最大收益。博弈，简单来说，可以理解为两个人对弈，在对弈的每一步，对手走一步棋，自己就需要参考对手的策略，选择自己下一步的策略，期望达到自己能够最终赢得棋局的目的。Neumann和Morgenstern[[47]](#endnote-47)发表了著名的详细阐述博弈论的论著——《Theory of Games and Economic Behavior 》，标志着博弈论的诞生。在文章《Proceedings of the national academy of sciences 》[[48]](#endnote-48)中，Nash提出了纳什均衡定理，这是对于系统博弈达到稳定状态的描述，系统中每一个参与博弈的个体要想自己得到最大收益，就必须综合考虑博弈对手的策略而调整自己的策略。这对于博弈论的研究来说，纳什均衡的提出具有非常重大意义，是里程碑式的研究成果。

复杂网络进行演化博弈时，采用随机初始化的方式产生初始策略状态，具体过程为：对于网络的个节点，产生个0到1之间的随机数，当时，设置节点初始策略为；当时，设置节点初始策略为。我们对这一初始化方式进行了改进，详细介绍在第五章。

## 3.1 演化博弈

一个完整的博弈模型由四部分元素组成：（1）博弈参与者；（2）参与者可以选择的博弈策略；（3）参与者获得的收益；（4）博弈策略的演化[[49]](#endnote-49)。复杂网络上的演化博弈，是指在复杂网络的网络结构上建立真实情景下描述的博弈模型，将网络中的节点作为博弈参与者，每一个节点代表一个参与者，每个节点与自己所有的邻居节点进行博弈并产生一定的收益，综合与所有邻居的博弈收益，根据博弈规则，更新所选择的策略49。

### 3.1.1 囚徒困境博弈

囚徒困境博弈(prisoner's dilemma game，简称PDG)是一类两策略博弈模型，每一个参与者都有可以选择两种策略：合作（Cooperation C）或背叛（Defection D）。该博弈模型讲述的具体情景是警察和罪犯的故事，假设小偷与都是犯罪嫌疑人，被警察所抓，但是警察并没有确凿的证据证明他俩或者他们其中的一个是罪犯，于是对小偷与小偷进行隔离审讯，使他们之间不能进行沟通，在该模型中，合作策略意味着不向警察说出实情，背叛策略意味着背叛对方，警察给出的审讯政策为：（1）如果两个人都主动坦白了对方的罪行，即都选择背叛策略，那么就有确凿的证据证明了他俩都有罪，但是因为主动坦白罪行，所以可以减轻刑罚，两人都坐牢5年；（2）如果一个人选择坦白对方的罪行（背叛），而另一个人一直抵赖（合作），那么，选择坦白的一方会被立即释放，同时有充分的证据证明另一方有罪，抵赖者坐牢10年；（3）如果两个人都选择抵赖（合作），即都不说出对方的罪行，那么警察没有充分的证据，所以两个人都因为证据不足而减轻刑罚，各坐牢1年49[[50]](#endnote-50)。从以上三种情况分析可以看到，当两个人都选择合作策略时，他们都只需要坐牢1年，对于整个系统来说，这是最佳结果。但是，当每一个个体自私地为自己考虑的时候，就会发现，如果对方选择背叛策略，那么，选择背叛策略会导致自己坐牢5年，选择合作策略会导致自己坐牢10年，很明显，背叛策略才是对自己最佳的策略；而当对方选择合作策略时，如果自己选择背叛策略，那么会被立即释放，如果自己选择合作策略，则会坐牢1年，所以，背叛策略显然也是对自己最有益的策略。综上所述，不论对方选择什么策略，出于自身利益，背叛都会是对自己最有益的策略。所以，随着博弈的演化，博弈参与者都会趋向于选择使自己收益最大的策略，即所有节点最终都会选择背叛策略，以得到个体的最大收益，然而，系统整体得到的收益却是最小的。

囚徒困境博弈的一般性收益[[51]](#endnote-51)如表3-1所示：

表3-1 PDG博弈收益表

|  |  |  |
| --- | --- | --- |
|  | C | D |
| C |  |  |
| D |  |  |

在表3-1中，满足，当参与者都选择合作策略时，系统得到最大收益，即成立。在表3-1中，表示：当一个参与者选择合作策略另一个参与者选择背叛策略时，合作者的收益为，背叛者的收益为。收益矩阵为51：

 (3-1)

将收益参数归一化为收益参数[[52]](#endnote-52)，重新得到囚徒困境博弈的收益矩阵为：

 (3-2)

### 3.1.2 雪堆博弈

雪堆博弈（Snowdrift Game SG）是一种两策略博弈模型，每个参与者的可选策略为合作（Cooperation）和背叛（Defection），描述了博弈双方有利益冲突时的决策情况[[53]](#endnote-53)。雪堆博弈描述的具体情景是：两个司机在一条道路上相向而行，两车之间有一个雪堆阻碍了车辆通行，每位司机都有两种选择，铲雪（合作策略）或者不铲雪（背叛），假设每个人可以顺利通行的收益是，铲除雪堆需要的劳动为，两位司机的选择可能有三种情况：（1）一个人选择合作策略，另一个人选择背叛策略，两人都可以通行，但是合作者需要付出劳动，所以合作者收益为，背叛者收益为；（2）两人都选择背叛策略，则两人均无法通行，收益均为0；（3）两人都选择合作策略，则每个人都可以直接通行，同时，每个人付出的劳动均为，则获得的收益均为。将归一化，引入博弈收益比53，得到使用博弈收益比表示的收益表3-2

表3-2 SG博弈收益表

|  |  |  |
| --- | --- | --- |
|  | C | D |
| C |  |  |
| D |  |  |

其中，，满足。表示：当一个参与者选择合作策略另一个参与者选择背叛策略时，合作者的收益为，背叛者的收益为。

收益矩阵为：

 (3-3)

### 3.1.3 纳什均衡

设有个参与者的博弈，博弈空间为，其中，表示第个参与者的可选的策略集合，例如，在囚徒困境博弈或雪堆博弈中，，其中表示合作策略，表示背叛策略。设表示第个参与者的策略，则。

个参与者选择的策略构成的集合为：

 (3-4)

除第个参与者外其余参与者选择的策略构成的集合为：

 (3-5)

第个参与者选择策略，其余参与者的策略集合为时，第个参与者的收益表示为：

 (3-6)

如果每一个参与者都不能通过改变当前策略使自身收益增加，即所有参与者的策略集合满足公式(3-7)时，博弈达到纳什均衡（Nash Equilibrium NE）状态48。

 (3-7)

其中，，。

如果对于任意一个参与者，在满足时，如果公式(3-7)严格成立，即，则称当前博弈的纳什均衡状态为严格纳什均衡（Strict Nash Equilibrium SNE）状态48记作。

### 3.1.4 博弈策略更新方法

在博弈演化过程中，每一个博弈参与者在完成一轮博弈后，都需要更新自己所选择的策略，进入下一轮博弈。在博弈论中，有多种常用的博弈策略更新准则。以下是三种博弈策略更新方法：

（1）最优响应策略更新规则[[54]](#endnote-54)[[55]](#endnote-55)

这种策略更新方法以找到使自身收益最大的策略为目的，每一轮博弈结束后，比较自身获得的总收益与每位对手总收益，将策略更新为使自己收益最大的策略。

图3-1以网络中的囚徒困境博弈为例，演示网络中节点6的策略更新过程，采用公式(3-2)所示的囚徒困境博弈的收益矩阵，取参数，网络初始策略为，节点6的邻居节点集合为，如果节点6选择合作策略，其收益为，如果节点6选择背叛策略，其收益为，因为，根据最优响应策略更新规则，节点6在下一轮选择背叛策略，如图3-1(c)所示。

  

1. 网络原图 (b) PDG策略初始状态 (c) 节点6更新策略



图3-1 PDG策略更新示意图。(a) 网络原图；(b) 囚徒困境博弈的初始状态为；(c) 节点6按照最优响应策略更新规则，策略由更新为。

（2）模仿最优者的策略更新方法[[56]](#endnote-56)

每一轮博弈结束后，每位参与者会观察所有博弈对手的当前收益及其所选择的策略，选出所有收益高于自己的对手，并以一定的概率将某一个收益较高的对手的策略作为自己下一轮的博弈策略。同样以图3-1所示的囚徒困境博弈为例，取，博弈初始策略如图3-1(b)所示，考察节点6的策略更新过程，根据公式(3-2)，节点6所有的邻居节点以当前策略进行博弈时，获得的总收益分别为：，，，，。则节点6的邻居节点中，节点2、4、5的收益最高，按照模仿最优者的策略更新规则，节点6将选择节点2或节点4或节点5的策略作为自己下一轮的策略，在这里，这三个节点的策略均为，所以节点6会将自己的策略由更新为。

（3）费米公式更新方法54

每一轮博弈结束后，每位参与者随机地从所有博弈对手中选择一位，计算该博弈对手的总收益，然后将自己的总收益与之比较，按照公式(3-8)计算出概率值，并以这个概率将自身策略转化为所选中的博弈对手的策略。

 (3-8)

公式(3-8)表示了参与者将策略更新为参与者的策略的概率，其中，和分别表示参与者与参与者的博弈总收益，是博弈动力学中的不确定性参数，当时，表示选择过程完全确定，此时，如果，则概率值为0，即参与者不会接受收益比自己小的对手的策略，如果，则概率值为1，即参与者一定接受收益比自己大的对手的策略；当时，表示不考虑博弈收益值的影响，以完全随机的方式采取博弈对手的策略。

## 3.2 复杂网络的同步演化博弈

### 3.2.1 同步演化博弈

复杂网络上的同步演化博弈是指，当网络节点进行第轮博弈时，所有节点只能利用历史信息更新自己的状态，即只能利用第轮或第轮以前的博弈状态。采用种博弈方式时，即使某一节点在第轮改变了自己的策略，其邻居节点依然无法利用这个节点在第轮的博弈状态信息。

### 3.2.2 基于同步雪堆博弈的MBR43算法

MBR(the Memory-based best response)是基于带记忆的雪堆博弈求解复杂网络最小顶点覆盖问题的算法，该算法使用最优响应博弈策略更新规则，使用同步演化博弈方式。

对于给定的网络，网络中每一个节点对应着一个完全理性的博弈参与者，完全理性是指完全根据收益来决定自己的策略，即采用最优响应策略更新准则。每个节点与所有的邻居节点进行博弈，得到与所有节点进行博弈产生的总收益。

对于给定的无权无向网络进行同步雪堆博弈，代表网络节点集合，代表网络边集合，表示节点总数，表示网络的总边数。个节点选择的策略构成的集合用公式(3-4)表示，将节点所选择的策略与顶点覆盖集相对应，每个节点可选的策略为或，如果节点选择策略，则该节点在顶点覆盖集中，否则，该节点不在顶点覆盖集（这种对应关系不是唯一的，也可以使用策略对应于节点在顶点覆盖集中，不同的对应关系只会影响参数要满足的条件）。设第个节点选择的策略为，除第个节点外，其余节点选择的策略构成的集合用公式(3-5)表示，表示严格纳什均衡状态，表示网络的一种最小顶点覆盖集，表示网络的所有最小顶点覆盖集组成的集合，表示网络的一种顶点覆盖集，表示网络的所有顶点覆盖集组成的集合。

**引理1**43：给定无权无向网络进行同步雪堆博弈，定义，当且(表示网络节点的最大度值)时，。

引理1建立了网络进行同步雪堆博弈时，博弈纳什均衡状态与网络顶点覆盖集之间的关系。当雪堆博弈参数满足时，使网络的最小顶点覆盖集中的节点选择合作策略，其余节点选择背叛策略，则此时网络博弈一定是严格纳什均衡状态，当已知网络的严格纳什均衡状态时，则所有选择合作策略的节点构成的集合一定是一个合法的顶点覆盖集。

**引理2**43：当且时，网络博弈达到严格纳什均衡状态的充分必要条件是：对于任意节点，（1）如果，那么节点的所有邻居节点都选择策略；（2）如果，那么节点至少有一个选择的邻居节点。

MBR算法采用同步演化博弈的优点是所有节点进行博弈策略更新所需要的信息是完全已知的，所有节点可以并行地计算收益并更新各自的策略。但是，当博弈参数满足且使引理1成立时，同步雪堆博弈存在一个明显的缺陷——无法保证网络的博弈过程趋于纳什均衡状态。如果无法保证一定收敛到纳什均衡，那么就无法利用引理1求解最小顶点覆盖问题。MBR算法提出了博弈记忆机制，克服了同步演化博弈无法收敛到纳什均衡的缺陷。





(a) 初始状态 (b) 状态1



(c) 状态2



图3-2 网络上的同步雪堆博弈演化过程。(a)网络节点初始博弈策略；(b) 由初始状态，经一轮同步雪堆博弈而得到的状态1，；(c) 由状态1，经一轮同步雪堆博弈而得到的状态2，

图3-2演示了简单网络上进行无记忆的同步雪堆博弈的过程，雪堆博弈的收益矩阵如公式(3-3)所示，在满足且的条件下，当没有博弈记忆机制时，博弈状态会一直震荡，无法收敛。从初始状态更新到状态1的博弈收益计算结果如表3-3所示，博弈更新过程的具体分析如下。

表3-3 雪堆博弈收益计算结果

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 节点  总收益  策略 | 1 | 2 | 3 | 4 | 5 | 6 |
| C |  |  |  |  |  |  |
| D | 0 |  |  |  |  | 0 |

雪堆博弈参数满足 且，在该网络中，。

对于节点1、节点6，均有两个邻居节点，且邻居节点所选的策略均为，以节点1为例，如果节点1选择策略，其收益为 ，如果节点1选择策略，收益为，因为，所以节点1将策略更新为，同理，节点6也会将策略更新为。

对于节点2、节点3和节点5，分别有一个选择的邻居和一个选择的邻居，以节点2为例，如果节点2选择策略，则收益为，如果选择策略，收益为，因为且，所以，所以节点2会将策略更新为，同理，节点3和节点5更新后的策略也为。

对于节点4，有一个选择的邻居和三个选择的邻居，如果节点4选择策略，其收益为，如果节点4选择策略，收益为，因为且，所以，所以节点4会将策略更新为。

由以上分析得到，网络的博弈状态从初始的更新到状态1。

由状态1开始更新时，每个节点的所有邻居节点都选择策略，对某个节点，假设共有个选择的邻居节点，则该节点选择的收益为，选择的收益为，根据最优响应策略更新规则，显然该节点会将策略更新为。所以，从状态1开始，博弈得到的状态2为。

由状态2开始更新时，每个节点的所有邻居节点都选择策略，对某个节点，假设共有个选择的邻居节点，则该节点选择的收益为，选择的收益为0，根据最优响应策略更新规则，显然该节点会将策略更新为。所以，从状态2开始，博弈得到的状态3为，状态3与状态1相同，网络上的同步雪堆博弈最终会陷入状态1与状态2之间的震荡，无法收敛到纳什均衡状态。

MBR算法提出的博弈记忆机制为：

1. 设置记忆长度为，每个节点进行博弈时，会保留对最近次博弈所选策略的记忆；
2. 网络中每一个节点与所有邻居节点进行同步雪堆博弈，根据最优响应准则，得到自己的最优策略；
3. 每个节点更新记忆：如果记忆长度不足，则直接将最近一次的策略存入记忆；如果记忆长度已经达到，则删除最早的记忆，并经最近一次的策略存入记忆，保记忆长度始终为；
4. 每个节点从自己的记忆中随机选择一个策略进入下一轮博弈；
5. 重复步骤2)至步骤4)，直到所有节点的个记忆完全一致。

因为每个节点都会将最新的最优策略放入记忆，那么，当所有节点的最新策略和记忆中的所有策略相同时，更新记忆时，节点随机地从记忆中选择策略，将一定会选择这个最优策略，所以，博弈记忆迭代过程的终止条件保证了所有节点都不会再改变策略，即博弈最终收敛到稳定状态。

表3-4给出了MBR算法在图3-2所示的网络上进行博弈演化的全过程。图3-2所示的网络共有6个节点，，参数满足 且，设记忆长度，分别为。初始时，所有记忆为空，随机产生初始状态，（1）由经一轮同步雪堆博弈后，得到状态，将存入记忆，然后每个节点从记忆中随机选择一个策略，构成，此时；（2）由经一轮同步雪堆博弈后，得到状态，成为较早一次的记忆，将存入最新记忆，然后每个节点从记忆中随机选择一个策略，构成，此时；（3）由经一轮同步雪堆博弈后，得到状态，同理，将存入记忆，和分别成为记忆和，然后每个节点从记忆中随机选择一个策略，构成，此时；（4）当记忆长度达到时，每次博弈产生的新状态作为存入记忆，原来的和分别成为和，然后从记忆中随机选择策略，进入下一轮博弈。由表3-4结果可知，迭代至第10轮时，所有个记忆均为，从记忆中选择的策略一定是，由该状态进行博弈，得到，即得到与记忆完全相同的博弈结果，所以，网络博弈达到稳定的纳什均衡状态，最终得到网络的顶点覆盖集为。

表3-4 MBR算法在如图3-2所示的网络上的迭代过程

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 节点的最优策略 | | | | | | | 节点记忆的状态(,表示记忆为空) | | | | | | |
| 节点  代数 |  |  |  |  |  |  | 节点  记忆 | 1 | 2 | 3 | 4 | 5 | 6 |
| 初始  策略 | C | D | D | D | D | C |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
| 1 | C | C | C | C | C | C |  | C | C | C | C | C | C |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
| *X*(1): 从记忆中随机选择策略 | | | | | | | C | C | C | C | C | C |
| 2 | D | D | D | D | D | D |  | D | D | D | D | D | D |
|  | C | C | C | C | C | C |
|  |  |  |  |  |  |  |
| *X*(2): 从记忆中随机选择策略 | | | | | | | C | D | D | D | D | D |
| 3 | C | C | C | C | C | C |  | C | C | C | C | C | C |
|  | D | D | D | D | D | D |
|  | C | C | C | C | C | C |
| *X*(3): 从记忆中随机选择策略 | | | | | | | C | C | C | C | C | D |
| 4 | D | D | D | C | C | D |  | D | D | D | C | C | D |
|  | C | C | C | C | C | C |
|  | D | D | D | D | D | D |
| *X*(4): 从记忆中随机选择策略 | | | | | | | D | D | D | D | D | C |
| 5 | C | C | C | C | C | C |  | C | C | C | C | C | C |
|  | D | D | D | C | C | D |
|  | C | C | C | C | C | C |
| *X*(5): 从记忆中随机选择策略 | | | | | | | C | C | C | C | C | D |
| 6 | D | D | D | C | C | D |  | D | D | D | C | C | D |
|  | C | C | C | C | C | C |
|  | D | D | D | C | C | D |
| *X*(6): 从记忆中随机选择策略 | | | | | | | D | C | D | C | C | D |
| 7 | C | C | C | C | C | D |  | C | C | C | C | C | D |
|  | D | D | D | C | C | D |
|  | C | C | C | C | C | C |
| *X*(7): 从记忆中随机选择策略 | | | | | | | C | D | C | C | C | D |
| 8 | C | D | D | C | C | D |  | C | D | D | C | C | D |
|  | C | C | C | C | C | D |
|  | D | D | D | C | C | D |
| *X*(8): 从记忆中随机选择策略 | | | | | | | C | C | D | C | C | D |
| 9 | C | D | D | C | C | D |  | C | D | D | C | C | D |
|  | C | D | D | C | C | D |
|  | C | C | C | C | C | D |
| *X*(9): 从记忆中随机选择策略 | | | | | | | C | D | D | C | C | D |
| 10 | C | D | D | C | C | D |  | C | D | D | C | C | D |
|  | C | D | D | C | C | D |
|  | C | D | D | C | C | D |
| *X*(10): 从记忆中随机选择策略 | | | | | | | C | D | D | C | C | D |

MBR算法的博弈记忆机制保证了复杂网络的雪堆博弈在满足引理1的条件下，一定会收敛到稳定状态，但是仍存在不足之处，在MBR算法中，记忆长度是非常关键的参数，如果较小，例如或，博弈过程较容易收敛，但是得到的结果较差；如果较大，例如，可以得到很好的解，但是对有些网络，博弈过程无法在合理的时间内收敛。具体的实验结果请看。

## 3.3 复杂网络的异步演化博弈

### 3.3.1 异步演化博弈

与同步演化博弈相对应的是异步演化博弈方式[[57]](#endnote-57)。设无权无向复杂网络，代表网络节点集合，代表网络边集合，表示节点总数，表示网络的总边数。网络中个节点分别对应着博弈的个完全理性的参与者，运用异步演化博弈方式，所有节点依次进行博弈并更新策略57，例如，对于某一网络，设节点和节点互为对方的邻居节点，当网络节点进行第轮异步博弈时，设节点和节点在第轮博弈结束后的策略分别为和，设节点的博弈顺序在节点之前，那么，节点与节点进行第轮博弈时，节点与节点的策略进行博弈，得到，当节点与节点与进行第轮博弈时，节点与节点节点的策略进行博弈，而非，即每个节点在与邻居进行博弈时，是与邻居所选择的最新状态进行博弈。所以，异步博弈方式是有顺序的博弈，节点博弈顺序不同，则节点的策略选择不同。

图3-3展示了网络节点进行异步囚徒困境博弈的状态更新过程，对任意节点，如果，则该节点选择合作策略，否则选择背叛策略，设节点博弈顺序为：1、2、3、4、5、6。图3-2(a)表示了每个节点选择的初始策略，图3-2(a)至图3-2(g)分别表示了每个节点的博弈结果，异步博弈方式下，某个节点进行博弈后，其他节点会与该节点更新后的策略进行博弈，例如，节点1、2、3、4、5完成博弈后，策略均变为，则当节点6与其邻居节点进行博弈时，节点与策略集进行博弈，而在同步博弈方式下，节点6会与进行博弈。



(a) 初始状态

  

1. 节点1博弈 (c) 节点2博弈 (d) 节点3博弈

  

(e) 节点4博弈 (f) 节点5博弈 (g) 节点6博弈

图3-3 异步博弈策略更新示意图。(a)网络节点选择的初始策略，节点博弈顺序为1、2、3、4、5、6；(b) 节点1与邻居节点6博弈，，策略更新为；(c) 节点2与邻居节点6博弈，，策略更新为；(d) 节点3与邻居节点6博弈，，策略更新为；(e) 节点4与邻居节点6博弈，，策略更新为；(f) 节点5与邻居节点6博弈，，策略更新为；(g) 节点6与邻居节点1、2、3、4、5博弈，节点6与策略进行博弈，，策略更新为。

### 3.3.2 异步囚徒困境博弈与最大独立集

给定的无权无向网络进行同步雪堆博弈，代表网络节点集合，代表网络边集合，表示节点总数，表示网络的总边数，对于任意节点，设节点的度为，节点的邻居节点组成的集合为。复杂网络节点进行囚徒困境博弈，每一个节点作为一个完全理性的博弈参与者，与所有邻居进行异步博弈，根据最优响应策略跟新规则更新策略，假设选择策略的节点在独立集中，选择策略的节点不在独立集中。设表示网络博弈达到纳什均衡状态时，所有选择策略的节点构成的集合，表示所有构成的集合。表示网络的极大独立集，表示网络所有可能的极大独立集构成的集合。

**定理1：**网络进行囚徒困境博弈达到纳什均衡状态，则恒成立。

**证明：**要证明，则需要证明和。

1、，如果一直网络的一种极大独立集，设极大独立集中的节点选合作策略，其余节点选背叛策略，则此时网络博弈处于纳什均衡状态。

1. ，则根据极大独立集的定义，节点的所有邻居节点一定选择策略，根据囚徒困境博弈的收益矩阵，计算得到节点选择不同策略时的收益分别为：，。满足，根据最优响应策略更新准则，节点会选择策略。即原本选择合作策略的节点会继续选择合作策略。
2. ，根据极大独立集的定义可以得到，，假设节点有个选择合作策略的邻居节点，有个选择背叛策略的邻居节点，则且，根据囚徒困境博弈的收益矩阵，计算得到节点选择不同策略时的收益分别为：，。因为参数满足且，所以，根据最优响应策略更新准则，节点会选择策略。即原本选择背叛策略的节点会继续选择背叛策略。

综合情况1)和情况2)可以得到，极大独立集所对应的网络博弈状态为纳什均衡状态。

2、，当网络囚徒困境博弈处于纳什均衡状态时，所有选择合作策略的节点构成网络的一种极大独立集。

1) ，假设节点有个选择合作策略的邻居节点，有个选择背叛策略的邻居节点，，则节点选择不同策略时的收益分别为：，。因为博弈处于纳什均衡状态，所以节点不会因为改变策略而使自身收益增加，所以，即且，所以，即节点的所有邻居节点都选择策略，所以纳什均衡状态下选择合作策略的节点构成独立集。

2) ，同样的分析可以得到，，，因为博弈处于纳什均衡状态，所以节点不会因为改变策略而使自身收益增加，所以，即且且是自然数，所以。即对于任意一个选择背叛策略的节点，其至少有一个选择合作策略的邻居节点。

综合情况1)和情况2)可以得到，当网络博弈处于纳什均衡状态时，所有选择合作策略的节点构成极大独立集。

综合考虑情况1、2可以得到。定理1证明完毕。

**推论1：**网络进行异步囚徒困境博弈，达到纳什均衡状态的充分必要条件是：（1）任意一个选择合作策略的节点都没有选择合作策略的邻居节点；（2）任意一个选择背叛策略的节点至少有一个邻居节点选择合作策略。

**证明：**1) 充分性。对于任意一个选择合作策略的节点，且节点没有选择的邻居节点，则有和，则，所以节点博弈后仍选择策略，不会改变自己的策略。对于任意一个选择背叛策略的节点，且节点至少有一个选择的邻居节点，设节点的选择的邻居节点数为()，则有，，则，所以节点博弈后仍选择策略，不会改变自己的策略。由此可知，所有节点都不会改变所选择的策略，所以博弈达到纳什均衡状态。

2) 必要性。由定理1可得，所以纳什均衡状态时所有选择合作策略的节点构成网络的极大独立集，由网络极大独立集的定义可知必要性成立。

定理1给出了网络进行异步囚徒困境博弈时，博弈的纳什均衡状态与网络极大独立集之间的等价关系，通过这一等价关系，可以使用博弈方法求得网络的极大独立集，即求得一个局部最优解。接下来需要证明网络进行异步囚徒困境博弈时，一定能收敛到纳什均衡。

**定理2：**从任意初始博弈状态开始，异步囚徒困境博弈一定会收敛到纳什均衡状态。

**证明：**设有个博弈参与者，初始状态为，初始状态不是纳什均衡状态，所以，由推论1可知，网络节点的策略选择可能会存在两种情况：（1）网络中存在两个顶点都选择的连边，记作“边”，即选合作策略的节点有选择合作策略的邻居节点；（2）存在选择的节点的所有邻居节点都选择，即选择背叛策略的节点的所有邻居节点都选择背叛策略。如果情况（1）存在，假设节点与节点相邻，且都选择策略，节点的博弈顺序在节点之前，那么，当节点进行博弈时，节点至少有1个选择的邻居节点，设节点有()个选择的邻居节点，个选择的邻居节点，得到和，因此，所以节点博弈后会选择策略，当节点开始博弈时，因为采用异步博弈方式，节点会根据节点更新后的策略计算与之进行博弈的收益值，所以博弈后“边”消失。如果情况（2）存在，对于满足情况（2）的节点，得到，，根据最优响应策略更新规则，节点会将策略更新为，所以情况（2）也会消失。综上所述，如果第次博弈得到的状态不是纳什均衡状态，那么第次博弈的结果是纳什均衡状态。所以网络进行异步囚徒困境博弈时，一定可以收敛到纳什均衡状态。

由定理1和定理2，建立了网络异步囚徒困境与网络极大独立集直接的关系，并保证了关系稳定存在，因此可以基于博弈的思想求解网络最大独立集的局部最优解，进优化得到全局最优解，定理1和定理2是基于博弈求解最大独立集方法的理论基础。

### 3.3.3 异步雪堆博弈与最小顶点覆盖

引理1给出了网络进行同步雪堆博弈时，严格纳什均衡状态与最小顶点覆盖集、顶点覆盖集之间的关系，但是，要想保证博弈状态收敛到纳什均衡，还需要加入博弈记忆，这是MBR算法的理论基础。这里，我们将证明网络进行异步雪堆博弈且不使用博弈记忆机制时，引理1的结论依然成立。

给定的无权无向网络进行同步雪堆博弈，代表网络节点集合，代表网络边集合，表示节点总数，表示网络的总边数。假设选择合作策略的节点在顶点覆盖集中，相反，选择背叛策略的节点不在顶点覆盖集中。表示严格纳什均衡状态，表示网络的一种最小顶点覆盖集，表示网络的所有最小顶点覆盖集组成的集合，表示网络的一种顶点覆盖集，表示网络的所有顶点覆盖集组成的集合，表示严格纳什均衡状态中所有选择合作策略的节点构成的集合，表示网络的所有组成的集合。

**定理3：**无权无向网络的节点从任意状态开始进行异步雪堆博弈，当且(表示网络节点的最大度值)时，博弈一定会收敛到严格纳什均衡。

**证明：**设第轮博弈的结果为，之后进行第轮博弈。如果不是严格纳什均衡状态，那么由引理2可得，可能存在以下两种情况：（1）至少存在一条边，其两个顶点都选择策略，将这种边记作“边”；（2）存在选择的节点，其所有邻居节点都选择策略。在异步博弈方式下，如果存在情况（1），假设节点和节点相邻且都选择策略，假设节点的博弈顺序在节点之前，那么，当节点进行博弈时，节点至少有1个选择策略的邻居节点，设节点有()个选择的邻居节点，个选择策略的邻居节点，得到和，可以得到，因为且、，所以，所以节点博弈后会选择策略，当节点开始博弈时，因为采用异步博弈方式，节点会根据节点更新后的策略计算与之进行博弈的收益值，所以博弈后“边”消失。如果存在情况（2），假设节点选择策略，且节点的所有邻居节点都选择策略，即节点与个选择的邻居节点进行博弈，那么，，显然，所以节点会将策略更新为，则第2种情况经博弈后会消失。综上所述，如果不是严格纳什均衡状态，那么经过一轮异步雪堆博弈，如果将符合严格纳什均衡状态的充分必要条件，即是严格纳什均衡状态。定理2证明完毕。

定理3证明了复杂网络进行无记忆的异步雪堆博弈一定会收敛到严格纳什均衡，再结合引理1，可以得到复杂网络异步无记忆雪堆博弈与最小顶点覆盖问题之间的联系，这是我们所提出的基于博弈求解最小顶点覆盖问题的理论基础。

## 3.4实验结果与分析

以图3-2所示的网络为例，分别进行比同步、异步雪堆博弈和囚徒困境博弈，初始状态为，博弈过程中每一轮博弈的状态如表3-5所示：

表3-5 同步、异步雪堆博弈在图3-2所示网络上的博弈状态表

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 同步雪堆博弈 | | | | | | 异步雪堆博弈 | | | | | |
| 节点代数 | 1 | 2 | 3 | 4 | 5 | 6 | 1 | 2 | 3 | 4 | 5 | 6 |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |

图3-2中展示了采用同步博弈方式的演化过程，每个节点同时根据上一轮的博弈结果更新自己的策略，得到，然后，由得到，再由得到……，表3-5给出了每一轮博弈的具体结果，可以看出，，即博弈状态变化过程为，无法收敛到稳定状态。

当采用异步演化方式时，节点按照1、2、3、4、5、6的顺序依次进行博弈，每个节点都根据上一轮博弈结果和当前轮较早进行博弈的节点的博弈结果更新自己的状态，由表3-5所示的博弈结果可得，4轮博弈的过程为，即博弈收敛到严格纳什均衡状态。

# 第四章 基于PDG求解最大独立集的局部搜索算法

## 4.1 基于囚徒困境博弈的局部搜索机制GLS

### 4.1.1 基于囚徒困境博弈求得极大独立集

给定的无权无向网络进行同步雪堆博弈，代表网络节点集合，代表网络边集合，表示节点总数，表示网络的总边数。表示网络的极大独立集，表示网络的最大独立集。网络进行异步囚徒困境博弈，选择策略的节点对应于独立集中的节点，表示网络博弈纳什均衡状态时所有选择策略的节点构成的集合。

由定理1可知，当网络进行异步囚徒困境博弈时，成立。由定理2可知，网络的异步囚徒困境博弈一定会收敛到纳什均衡状态，所以，定理1和定理2共同奠定了基于囚徒困境博弈求解合法独立集的理论基础。网络的极大独立集属于独立集，同时，任何非独立集中的节点都不可再加入该独立集，所以，极大独立集肯定是一种合法的独立集，同时是一种局部最优解。

### 4.1.2 基于囚徒困境博弈的局部搜索

在的基础上，我们提出了基于囚徒困境博弈的局部搜索机制GLS(Game-based Local Search)，该局部搜索机制借鉴ILS算法34的操作，在实现的同时，可以实现多种版本的局部搜索操作。

**定理3：**对无权无向网络，节点集，由纳什均衡状态开始，随机选择节点，将其策略由变为，得到，表示节点的所有邻居及节点构成的集合，使中的节点按照节点最后进行博弈的顺序进行异步囚徒困境博弈至纳什均衡状态，则整个网络也处于纳什均衡状态。

**证明：**设表示进行博弈后，整个网络中所有选择策略的节点构成的集合。

1) 对于中的任意一个节点，如果节点的邻居节点中有选择策略的节点，那么，根据推论1，节点经博弈后会选择策略，即这样的节点不在集合中，所以，对于，，使得节点与节点相邻。即中的节点的策略选择不会因为中的节点的策略变化而改变。

2) 对于原纳什均衡状态下选择背叛策略，且不属于的任意一个节点，有。根据推论1，节点至少有一个选择策略的邻居节点，又因为，所以这个选择策略的邻居节点一定不是节点，所以对于节点来说，当中的节点的策略变化时，仍然至少有一个选择策略的邻居节点，所以始终可以维持，所以节点始终选择策略，不会因为中的节点的策略变化而改变。

3) 中的节点进行博弈后，中至少存在一个选择策略的节点。根据推论1，中所有邻居都选择策略的节点博弈后，会选择策略。节点最后进行博弈，特殊情况下，当节点的所有邻居节点都没有在博弈过程中改变策略时，节点博弈后将重新选择策略，即网络状态又恢复到最初状态。

综上所述，定理3所描述的内容成立。

**定理4：**按照定理3的描述，由得到新纳什均衡状态，并得到，则有(即，表示新得到的极大独立集)。

**证明：**按照定理3的描述得到，可以分为4种情况讨论：

1. ，节点博弈后选择策略，此时有，但是，即两次极大独立集中的节点不同。
2. ，节点和节点博弈后选择策略，此时有，即完成操作；
3. ，集合中的节点博弈后选择策略，此时有，即完成操作；
4. 节点重新选择策略，解集未发生改变。

 

1.  (b) 



1. ，即

图4-1 基于囚徒困境博弈的过程图。(a) 网络的一种纳什均衡状态，对应的极大独立集为；(b) 从当前极大独立集中选择节点5，将其移出独立集，即策略由改为，得到节点5的；(c) 按照的顺序进行异步囚徒困境博弈至纳什均衡，对应的，此时得到网络的最大独立集。

  

(a)  (b)  (c) 

图4-2 基于囚徒困境博弈的过程图。(a) 网络的一种极大独立集；(b) 从当前极大独立集中选择节点7，将其策略由改为，得到节点7的；(c) 按照的顺序进行异步囚徒困境博弈至纳什均衡，得到。

图4-1和图4-2给出了简单网络上基于囚徒困境博弈的局部搜索过程。图4-1所示的过程完成了一次基于囚徒困境博弈的。图4-2所示的过程完成了一次基于囚徒困境博弈的。可以看出，基于囚徒困境博弈的局部搜索过程不局限于实现或，而是可以根据具体的网络结构，实现任意数目的局部搜索，而且仅仅依靠异步囚徒困境博弈来完成这一操作，不需要像其他局部搜索算法那样定义复杂的节点关系，思路简单，效果较好。

由定理4的证明过程的情况1可知，该局部搜索方法可能存在中的一个节点(记作节点)与另一个非中的节点(记作节点)的交换操作，局部搜索后产生新极大独立集 。如果在下一轮局部搜索过程中恰好选择将节点移出独立集，那么博弈后将导致节点重新选择策略，即重新进入独立集，产生的无效操作，这是我们所提出的的局部搜索思路的一个缺点，为防止这种情况的发生，我们引入博弈记忆机制，设记忆长度为。我们的博弈记忆的处理过程与MBR算法的不同，避免了MBR算法中记忆长度较长时博弈很难收敛的问题，我们的基于博弈记忆的局部搜索过程如下：

1. 初始时，每个节点的个记忆均为空；中所有可以执行局部搜索的节点构成集合，初始时；
2. 初始化记忆：网络博弈第一次达到纳什均衡状态后，每个节点将纳什均衡状态下自己所选的策略存入记忆；
3. 局部搜索：从中选择随机选择节点，同时，将节点的策略由改为，中的节点进行异步雪堆博弈至纳什均衡状态，并得到网络的()；
4. 更新记忆和：在新得到的纳什均衡状态下，每个节点将自己选择的策略存入记忆；对中任意节点，如果节点的记忆长度达到且记忆中有策略，那么并将节点的记忆清空，使节点重新累积记忆；如果节点的记忆长度未达到，则只增加当前记忆，不更新；
5. 循环：重复步骤3)和步骤4)，直到达到最大迭代次数或集合变为空集。

以图4-2所示的网络来说明基于博弈记忆的局部搜索过程。（1）如图图4-2(a)所示，初始，则，设记忆长度，分别表示为，和，表示最新的记忆，此时节点集中节点的记忆分别初始化为；（2）如图4-2(b)所示，从中选择节点7，更新为；中的节点进行异步雪堆博弈得到图4-2(c)所示的新纳什均衡，节点记忆更新为，，因为所有节点的记忆都未达到设定的长度，所以不更新；（3）从中选择节点2，更新为空集，博弈至纳什均衡状态后，节点2重新选择策略，此时博弈记忆更新为，，，节点4、5、6的记忆长度为且记忆中带有策略，则将这三个节点的记忆清空，并更新；（4）从中选择节点4进行局部搜索，此时，记忆更新为，，；（5）从中选择节点5进行局部搜索，此时，记忆更新为，，；（6）从中选择节点6进行局部搜索，此时为空，记忆更新为，，；此时，对于中的节点，节点记忆长度都达到且记忆中都没有策略，所以为空集，算法终止。在编程实现过程中，使用“为空集”这一终止指标往往导师迭代冗余，所以设置最大局部搜索迭代次数，当最大迭代次数或集合变为空集时，迭代终止。

基于囚徒困境博弈的局部搜索算法(GLS)的算法流程图如图4-3所示。表示输入网络；表示囚徒困境博弈参数，满足；表示最大局部搜索次数；表示博弈记忆长度，建议长度为，其中是0到1之间的小数，表示由初始状态产生的第一个极大独立集中节点的个数；表示迭代次数累加值。我们所使用的博弈记忆机制，是为了防止局部搜索过程中循环出现两节点之间的互相交换，尽可能保证连续次进行局部搜索节点不同，因此将的长度设置为与初始局部最优解的规模相当的值，保证局部最优解中的节点都有可能进行局部搜索。



图4-3 GLS流程图

### 4.1.3 GLS的博弈顺序

由异步博弈的过程可知，异步博弈结果与节点的博弈顺序密切相关，我们分别采用随机(Random)方式和基于节点度(Degree-based)的方式执行GLS算法，并将随机GLS算法记作RGLS，将基于节点度的GLS算法记作DGLS，这两者的不同之处仅在于从集合中选择节点的方式不同。

RGLS算法是指在局部搜索过程中，从集合中随机选择节点进行局部搜索，我们对GLS算法的描述都是基于RGLS进行描述的。如果从集合中优先选择度值最大的节点进行局部搜索，并且规定网络节点按照度值从大到小的顺序进行博弈，则RGLS算法变为DGLS算法。DGLS算法是一种贪婪的处理方式，优先选择度值最大的节点进行局部搜索时，该节点有选择策略的邻居节点的概率较大，则该节点会以较大的概率选择策略，即DGLS倾向于使度较大的节点选择策略，因此让更多的节点以更大的可能性选择策略。

### 4.1.4 基于博弈的扰动方法

扰动操作是局部搜索过程中的非常重要的步骤，当搜索达到局部最优解时，往往需要对局部最优解进行人为干扰，达到跳出局部最优的目的。

定理4的证明过程中所列举的第一种情况为，且，这种只改变极大独立集中的节点构成，而不改进极大独立集规模的局部搜索结果，本身是对于当前局部最优解的扰动。即我们所提出的基于囚徒困境博弈的局部搜索方法在局部搜索的同时，可以实现对局部最优解的扰动。另外我们引入博弈记忆机制，避免了这一扰动过程在一对节点上重复出现多次。

 

(a)  (b) 节点7移出，

 

(c) 节点1移出， (d) 节点3移出，

图4-4 基于囚徒困境博弈的扰动示意图。(a) 初始极大独立集；(b) 从中移出节点7进行局部搜索，节点5进入，实现了扰动操作；(c) 从中移出节点1进行局部搜索，节点6进入，实现了扰动操作；(d) 从中移出节点3进行局部搜索，节点2和节点4进入，即得到最优解。

图4-4展示了扰动操作在局部搜索过程中的作用，4-4(a)所示的局部最优解，在没有扰动操作的情况下，将无法继续进行局部搜索操作，而节点7与节点5的交换（如图4-4(b)），节点1与节点6的交换（如图4-4(c)），打破了陷入局部最优解的僵局，使局部搜索过程最终搜索到最优解。在节点7与节点5完成交换后，我们的博弈记忆机制将防止下一轮局部搜索时产生 “节点5与节点7交换”这一循环过程，因此保证算法在发生扰动之后的正常运行。

## 4.2 基于囚徒困境博弈的迭代局部搜索算法IGLS

我们提出的GLS方法可以实现基于节点交换的扰动操作，可以实现对局部最优解的改进。但是，因为扰动操作形式单一，所以这种单纯的局部搜索依然无法得到全局最优解，因此我们借鉴传统的迭代局部搜索[[58]](#endnote-58)思路，将我们提出的GLS作为迭代局部搜索过程中的局部搜索算法，在迭代局部搜索过程中，需要人为干预局部最优解作为扰动操作，并在一轮局部搜索结束后，更新解集。经典的迭代局部搜索思路58为：

1、；

2、；

3、.

在迭代局部搜索算法中，第一步是对当前局部最优解的扰动操作，即强制打破局部最优的状态；第二步是对扰动后的解集进行局部搜索操作，直到搜索到另一个局部最优解；第三步是判断是否接受新得到的解集，如果得到的解集更优，则接受，否则，以一定的概率接受更较差的；整个迭代局部搜索过程就是步骤一到步骤三的循环，直到循环次数达到设定的阈值。

我们以GLS作为局部搜索操作，采用简单的弱扰动机制，并基于模拟退火思想以一定的概率接受较差的解，综合提出基于囚徒困境博弈的迭代局部搜索算法(Iterated Game-based Local Search Algorithm，简称IGLS)

### 4.2.1 弱扰动机制

因为GLS自带扰动操作，所以，在IGLS中，我们使用较为简单的弱扰动机制。具体操作过程为：对于极大独立集，随机从中选择一个节点，将这个节点强制移入独立集，并按照满足独立集的要求，从中删除与该节点相邻的节点。

### 4.2.2 基于模拟退火思想的解集更新方法

在迭代局部搜索中，当对解集执行扰动和局部搜索操作后，得到新解集，当比更优时，我们将更新为，即；当比差时，我们需要按照一定的概率接受，这也是一种扰动操作，有助于搜索到全局最优解。在IGLS算法中，根据固体退火过程中的Metropolis准则[[59]](#endnote-59)计算概率，假设固体初始状态的能量为，随后状态发生变化，状态能量下降为，表示能量变化值，则有计算公式如下：

 (4-1)

其中，表示初始温度，是玻尔兹曼(Boltzmann)常数，在固体退火系统中，公式(4-1)表示固体处于状态和状态的概率的比值。因为，所以，且，所以是一个0到1之间的小数。

在IGLS中，我们使用固体退火过程中的公式(4-1)计算概率，当时，以概率接受。此时取，在IGLS编程实现中，取，。

综上可得，IGLS中的解集更新过程为：设进行某一次迭代时，初始解集为，经过扰动和局部搜索操作后得到解集，当时，；当时，随机产生0到1之间的小数，按照公式(4-1)计算概率，当时，，当时，保持不变。

### 4.2.3 算法整体思路

IGLS算法流程图如图4-5所示：



图4-5 IGLS算法流程图

图4-5中，表示初始温度，是玻尔兹曼(Boltzmann)常数；表示迭代局部搜索的最大次数；用于记录迭代次数，当是，迭代终止，将当前极大独立集作为全局最优解。在编程实现中，参数，， 的取值在不同网络中保持一致。

## 4.3 实验结果与分析

为验证GLS和IGLS的性能，我们与算法ILS34、SBTS35和BLS36在不同规模、不同种类的网络进行实验对比，并给出详细的实验结果。实验环境为：Windows7， 2GB内存。

### 4.3.1 实验网络简介

为了验证算法的适应性，我们在不同种类和规模的网络上进行实验，所进行实验的网络分为两类，第一类是标准测试集网络，如DIMACS网络[[60]](#endnote-60)、CODE网络[[61]](#endnote-61)和BHOSLIB网络[[62]](#endnote-62)，这类网络是具有最优解的标准网络，可以测试算法得到的结果是否是最优解；第二类是现实世界中常见的网络类型，如BA无标度网络[[63]](#endnote-63)、ER随机网络[[64]](#endnote-64)、WS小世界网络[[65]](#endnote-65)和LFR社区结构网络[[66]](#endnote-66)，这类网络符合现实中的真实网络结构，因此，在这类网络上的结果具有实际意义和实际价值。

第一类网络的网络名、节点数、平均度、最优解的个数（best列）如表4-1所示。

表4-1 BHOSLIB、CODE、DIMACS网络信息表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 网络种类 | 网络名 |  | <k> | best |
| The BHOSLIB family | Frb30-15-1 | 450 | 79 | 30 |
| Frb35-17-1 | 595 | 94 | 35 |
| Frb40-19-1 | 760 | 109 | 40 |
| Frb45-21-1 | 945 | 125 | 45 |
| Frb50-23-1 | 1150 | 139 | 50 |
| Frb100-40 | 4000 | 286 | 100 |
| CODE family | 1dc.1024 | 1024 | 47 | 94 |
| 1tc.1024 | 1024 | 16 | 196 |
| 1et.2048 | 2048 | 22 | 316 |
| 2dc.2048 | 2048 | 493 | 24 |
| The DIMACS  family | brock200\_2 | 200 | 99 | 12 |
| brock200\_4 | 200 | 131 | 17 |
| C1000.9 | 1000 | 900 | 68 |
| C125.9 | 125 | 112 | 34 |
| C250.9 | 250 | 224 | 44 |
| C500.9 | 500 | 450 | 57 |
| DSIC1000-5 | 1000 | 1000 | 15 |
| MANN a\_27 | 378 | 374 | 126 |
| hamming8-4 | 256 | 163 | 16 |
| hamming10\_4 | 1024 | 848 | 40 |
| keller4 171 | 171 | 110 | 11 |
| keller5 776 | 776 | 582 | 27 |
| gen200\_p0.9\_44 | 200 | 179 | 44 |
| gen200\_p0.9\_55 | 200 | 179 | 55 |
| gen400\_p0.9\_55 | 400 | 359 | 55 |
| gen400\_p0.9\_65 | 400 | 359 | 65 |
| gen400\_p0.9\_75 | 400 | 359 | 75 |
| p-hat300-1 | 300 | 73 | 8 |
| p-hat300-2 | 300 | 146 | 25 |
| p-hat300-3 | 300 | 223 | 36 |
| p-hat700-2 | 700 | 348 | 44 |
| p-hat700-3 | 700 | 523 | 62 |
| p-hat1500-1 | 1500 | 380 | 12 |
| p\_hat1500-2 | 1500 | 759 | 65 |
| p\_hat1500-3 | 1500 | 1130 | 94 |

第二类网络的命名设定如下：（1）BA无标度网络的度分布服从幂律分布，生成这类网络需要两个参数：网络节点个数和网络总边数(或网络节点平均度)，在本文中所使用的BA网络被命名为“BA ”，表示节点数为，平均度为的BA网络。（2）对于一个个节点的ER网络，其条边随机均匀地分布在个节点之间，我们用“ER ” 表示节点数为，平均度为的BA网络。（3）构造WS小世界网络需要有三个参数：节点个数、节点平均度和重连概率，重连概率表示将任意两个不相邻的节点加上连边的概率，我们将WS网络记作“WS ”，表示节点数为，平均度为、重连概率为的WS网络。（4）生成LFR网络需要6个参数：节点个数、节点平均度、节点最大度、度分布的指数、社区分布指数和混合参数，我们所使用的LFR网络采用、和，“LFR ”表示节点数为，平均度为、节点最大度为的LFR网络。

### 4.3.2 不同算法搜索合法解的性能对比实验

局部搜索算法的前提是对局部最优解进行局部搜索，在最大独立集问题上，合法的局部最优解是极大独立集，定理1证明了网络进行异步囚徒困境博弈时，成立，所以可以基于博弈得到初始的局部最优解，然后对这个解进行局部搜索和优化。由4.1.3节可知，不同的节点博弈顺序会对博弈结果产生影响，我们将节点以随机顺序进行博弈得到初始的方法记作Rgame，如果节点按照度值由大到小的顺序进行博弈，则记为Dgame。

Rgame的执行过程为：1) 随机产生初始策略；2) 打乱节点顺序，节点按照打乱的顺序进行异步囚徒困境博弈直至纳什均衡，得到初始的。

Dgame的执行过程为：1) 随机产生初始策略；2) 节点按照度值有大到小的顺序进行排序，然后按照这个顺序进行异步囚徒困境博弈直至纳什均衡，得到初始的。

在ILS、SBTS和BLS中，采用的都是随机产生初始的方法（记作Random）343536，随机产生初始的过程为：1) 初始化为空集，初始化，表示所有可能加入独立集的节点构成的集合；2) 从随机选择一个节点移入；3) 将节点的邻居节点从中移出；4) 重复第2、3步直至为空，此时是一个极大独立集。

在ILS中，还采用另外一种贪婪方法（记作Greedy）34，这种方法与Random不同之处在于，每次从中优先选择自由邻居（第二章2.3.1节）数目最少的节点进入独立集。这种贪婪的方式是尽可能保留自由节点，从而使解集被扩充的可能性最大。

表4-2给出了这三种算法在不同网络上搜索初始的实验结果。每个算法在每个网络上都独立运行15次，统计15次独立运行的初始的平均值、标准差和平均运行时间（单位：秒）。Rgame和Dgame的博弈参数均为。

表4-2 Rgame、Dgame、Random、Greedy搜索初始极大集的实验结果

解集规模的平均值/标准差/算法平均运行时间

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 网络名 | best | Rgame | Dgame | Random | Greedy |
| Frb30-15-1 | 30 | 21.33  /0.52/ 0.004 | 21.33  /1.18/0.013 | 20.00  /1.21/ 0.016 | 24.47  /1.03/ 0.029 |
| Frb100-40 | 100 | 68.40  /1.37/ 0.258 | 67.20  /3.29/1.068 | 66.27  /1.93/ 1.626 | 80.40  /1.24/ 3.73 |
| 1dc.1024 | 94 | 70.67  /0.61/ 0.016 | 66.00  /0.82/0.070 | 59.73  /1.98/ 0.065 | 74.67  /1.19 /0.221 |
| 2dc.2048 | 24 | 19.93  /0.25/ 0.080 | 18.40  /0.21/0.325 | 15.67  /1.14/ 0.338 | 21.13  /0.49 /0.639 |
| Brock200\_2 | 12 | 7.27  /0.68/ 0.001 | 7.27  /0.52/0.002 | 7.20  /0.75/ 0.003 | 9.2  /0.40 /0.006 |
| C1000.9 | 68 | 45.87  /1.75/ 0.019 | 47.06  /2.43/0.070 | 44.53  /2.25/ 0.093 | 58.40  /1.25 /0.194 |
| Hamming10\_4 | 40 | 26.40  /1.96/ 0.019 | 26.40  /1.79/0.075 | 20.00  /2.68/ 0.076 | 35.73  /1.00 /0.185 |
| p-hat1500-1 | 12 | 7.13  /0.81/ 0.067 | 7.00  /0.63/0.183 | 6.73  /0.93/ 0.235 | 9.07/  /0.69/0.360 |
| BA500<4> | - | 273.47  /1.36/ 0.003 | 274.93  /2.94/0.016 | 253.4  /7.79/ 0.015 | 289.27  /1.18 /0.158 |
| BA1000<8> | - | 415.67  /5.64/ 0.013 | 416.20  /6.09/0.062 | 373.87  /8.47/ 0.076 | 456.00  /4.13/ 0.905 |
| BA2000<10> | - | 737.87  /6.51/ 0.052 | 737.80  /7.15/0.252 | 644.80  /9.64/ 0.311 | 817.73  /5.37/ 5.96 |
| ER500<4> | - | 210.73  /3.73/ 0.003 | 221.60  /3.43/0.017 | 206.33  /4.60/ 0.012 | 243.27  /1.18/ 0.134 |
| ER1000<8> | - | 282.27  /6.82/ 0.013 | 303.20  /5.23/0.063 | 275.47  /7.70/ 0.056 | 336.67  /2.15 /0.659 |
| ER2000<10> | - | 486.07  /7.10/ 0.051 | 526.33  /5.59/0.250 | 477.47  /7.44/ 0.288 | 595.20  /3.17/ 4.195 |
| WS500<10>  P=0.5 | - | 96.13  /1.93/ 0.003 | 96.80  /2.11/0.017 | 83.00  /3.24/ 0.010 | 100.67  /0.79/ 0.060 |
| WS1000<10>  P=0.5- | - | 329.27  /4.27/ 0.013 | 329.33  /5.88/0.062 | 308.47  /5.92/ 0.076 | 368.00  /0.52/ 0.792 |
| WS2000<10>  P=0.5 | - | 670.93  /7.79/ 0.050 | 671.33  /7.29/0.250 | 622.73  /9.04/ 0.319 | 744.80  /0.40/ 5.952 |
| LFR500  <15>(50) | - | 125.20  /4.38/ 0.004 | 150.33  /3.39/0.017 | 117.20  /7.62/ 0.021 | 157.67  /3.42 /0.109 |
| LFR1000  <20>(60) | - | 226.27  /11.08/ 0.013 | 276.80  /4.00/0.062 | 215.73  /12.32/ 0.093 | 294.07  /2.98/ 0.696 |
| LFR2000  <25>(70) | - | 358.00  /8.93/ 0.052 | 449.00  /5.05/0.253 | 347.00  /12.62/ 0.379 | 485.87  /6.34/ 3.826 |

由表4-2可知，Rgame与Dgame相比，Rgame的标准差小、平均运行时间短，得到的初始独立集的规模在有些网络上较大、在有些网络上较小，这两个方法没有明显的较优算法。Rgame与Random相比，Rgame一直搜索到规模较大的初始解，且标准差和运行时间较短，所以Rgame产生初始解的性能较好。Rgame与Greedy相比，Greedy得到较优解，但运行时间往往是Rgame的几十倍。

在编程实现中，我们提出的GLS和IGLS算法使用Rgame的方式产生初始解，ILS算法、SBTS和BLS算法采用Greedy的方式产生初始解。

### 4.3.3 局部搜索性能对比

我们提出了GLS的局部搜索机制，将其与ILS算法中的局部搜索机制在不同网络上进行实验对比，验证算法性能。在实验中，分别采用了RGLS和DGLS这两种实现方式。所有实验都独立运行30次，并统计平均值、标准差和平均运行时间。

GLS的实验参数为：，，，表示初始解集，表示网络节点个数。

表4-3 GLS和实验结果

解集规模的平均值/标准差/算法平均运行时间

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| network | best | RGLS | DGLS |  |
| Frb30-15-1 | 30 | 23.80  /0.98/0.010 | **24.73**  /1.15/0.010 | 23.07  /0.77/ 0.023 |
| Frb50-23-1 | 50 | 41.20  /1.19/0.053 | **41.23**  /1.31/0.051 | 39.33  /1.34/ 0.176 |
| 1dc.1024 | 94 | **72.53**  /0.99/0.024 | 72.00  /1.00/0.026 | 72.00  /0/ 0.032 |
| C1000.9 | 68 | 57.47  /1.52/0.073 | 57.66  /1.46/0.070 | **59.93**  /1.67/ 0.198 |
| DSIC1000-5 | 15 | 12.27  /0.49/0.068 | **12.43**  /0.66/0.079 | 11.33  /0.58/ 0.205 |
| MANNa\_27 | 126 | 117.60  /0.75/0.004 | 117.90  /0.87/0.006 | **119.20**  /1.00/ 0.002 |
| p-hat1500-1 | 12 | 9.63  /0.55/0.123 | **9.93**  /0.25/0.191 | 8.20  /0.73/ 0.630 |
| BA1000<10> | - | 382.63  /4.95/0.018 | 393.00  /4.62/0.058 | **401.47**  /2.50/ 0.081 |
| ER500<8> | - | 145.27  /3.60/0.007 | **159.47**  /3.53/0.015 | 153.27  /2.62/ 0.020 |
| ER1000<8> | - | 316.00  /5.15/0.063 | **322.63**  /4.83/0.050 | 321.53  /3.79/ 0.094 |
| ER2000<8> | - | 610.33  /0.38/0.228 | **634.37**  /5.11/0.221 | 624.07  /9.48/ 0.400 |
| WS500  p=0.1 | - | 100.90  /2.62/0.009 | **102.90**  /2.93/0.012 | 100.33  /2.11/ 0.009 |
| WS1000  p=0.1 | - | 205.70  /3.72/0.047 | **206.13**  /2.91/0.045 | 204.80  /2.29/ 0.039 |
| WS2000  P=0.1 | - | 405.67  /4.45/0.075 | **412.90**  /4.66/0.167 | 410.67  /4.22/ 0.138 |
| LFR500  <15>(50) | - | 145.17  /5.00/0.020 | **151.40**  /2.60/0.017 | 150.33  /1.73/ 0.048 |
| LFR1000  <25>(70) | - | 248.00  /5.87/0.085 | **260.40**  /3.97/0.082 | 254.40  /4.90/ 0.362 |
| LFR2000  <25>(70) | - | 474.13  /8.78/0.332 | **489.13**  /5.01/0.280 | 485.33  /6.21/ 1.500 |

GLS算法迭代终止条件是集合为空或达到最大搜索次数，越大，理论上进行局部搜索的次数越多，可以得到越好的解，但是运行时间增加。在GLS编程实现中，取，由表4-3可知，此时DGLS的平均运行时间与的运行时间相当，而且，除C1000.9、MANNa\_27和BA1000<10>这三个网络以外，DGLS都可以得到比更好的结果。同时，还可以看出，DGLS的局部搜索性能比RGLS好，因为DGLS在局部搜索过程中以有序的方式进行博弈，度越大的节点越有可能选择策略。

### 4.3.4 综合对比实验

将IGLS算法与求解最大独立集问题的ILS算法、SBTS算法和BLS算法进行对比，验证这四种算法在求解最大独立集的性能。

IGLS算法的参数为：；；，；对于较复杂的网络，取，对于简单网络（名字后带\*标记），取；对于第一类网络，取，对于第二类网络，。

ILS算法的参数为：最大迭代次数，ILS算法中的操作没有设定迭代次数，会一直迭代到候选集为空。

SBTS算法的参数为：对于第一类网络，当搜索到最优解或者达到最大交换次数时，算法终止，最大交换次数为10万；对于第二类网络，统一设置最大交换次数为10万。

BLS算法的参数为：对所有网络，，， ，；如果网络节点数不超过1000，则，否则，。

在所有网络上，每个算法都独立运行30次，并统计得到的最优解及其出现次数、平均值、标准差和平均运行时间，表4-3至表4-9表示了在不同网络族上的实验结果。

表4-4 DIMACS网络族上的综合对比实验结果

最优解(出现次数)/平均值/标准差/算法平均运行时间

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| network(best) | IGLS | ILS | SBTS | BLS |
| brock200\_4  (17) | 17(30)/17.00  /0/0.672 | 17(30)/17.00  /0/4.000 | 17(15)/16.50  /0.50/28.258 | 17(3)/16.07  /0.37/0.471 |
| C125.9  (34) | 34(30)/34.00  /0/0.003 | 34(30)/34.00  /0/0.287 | 34(30)/34.00  /0/0.002 | 34(30)/34.00  /0/0.162 |
| C250.9  (44) | 44(30)/44.00  /0/0.054 | 44(12)/43.27  /0.73/1.381 | 44(30)/44.00  /0/0.093 | 44(29)/43.93  /0.37/0.555 |
| C500.9  (57) | 57(8)/56.03  /0.71/2.086 | 57(5)/54.40  /0.95/5.690 | 57(4)/56.13  /0.34/79.706 | 57(2)/54.00  /0.67/2.341 |
| DSJC1000\_5  (15) | 15(2)/14.07  /0.25/123.769 | 15(30)/15.00  /0/103.00 | 15(27)/14.90  /0.30/559.221 | 15(2)/14.07  /0.25/6.216 |
| gen200\_p0.9\_44  (44) | 44(30)/44.00  /0/0.465 | 44(9)/40.93  /2.06/0.827 | 44(30)/44.00  /0/0.088 | 44(28)/43.70  /1.15/0.342 |
| gen200\_p0.9\_55  (55) | 55(30)/55.00  /0/0.263 | 55(16)/48.63  /6.81/0.864 | 55(30)/55.00  /0/0.027 | 55(30)/55.00  /0/0.331 |
| gen400\_p0.9\_55  (55) | 55(3)/52.97  /0.80/1.509 | 52(13)/51.13  /0.92/3.404 | 55(30)/55.00  /0/1.231 | 55(6)/51.80  /1.69/1.466 |
| gen400\_p0.9\_65  (65) | 65(30)/65.00  /0/0.763 | 65(15)/57.70  /7.33/3.474 | 65(30)/65.00  /0/0.102 | 65(30)/65.00  /0/1.296 |
| gen400\_p0.9\_75  (75) | 75(30)/75.00  /0/0.318 | 75(30)/75.00  /0/3.489 | 75(30)/75.00  /0/0.263 | 75(30)/75.00  /0/1.230 |
| hamming8-4  (16) | 16(30)/16.00  /0/0.133 | 16(30)/16.00  /0/3.970 | 16(30)/16.00  /0/0.008 | 16(30)/16.00  /0/0.388 |
| hamming10-4  (40) | 40(30)/40.00  /0 /4.969 | 40(30)/40.00  /0/39.00 | 40(40)/40.00  /0/0.111 | 40(30)/40.00  /0/6.471 |
| keller4 171  (11) | 11(30)/11.00  /0/0.067 | 11(30)11.00  /0/1.655 | 11(30)/11.00  /0/0.003 | 11(30)/11.00  /0/0.145 |
| keller5 776  (27) | 27(29)/26.97  /0.18/19.687 | 27(6)/25.27  /1.18/33.18 | 27(29)/26.97  /0.18/50.019 | 27(2)/25.17  /0.65/3.270 |
| MANN\_a27  126 | 126(28)/125.93  /0.25/2.198 | 126(30)/126.00  /0/1.002 | 122(2)/120.17  /0.90/17.189 | 125(9)/124.07  /0.74/2.255 |
| p-hat300-1  (8) | 8(30)/8.00  /0/1.654 | 8(9)/7.20  /0.60/8.920 | 8(30)/8.00  /0/0.127 | 8(27)/7.90  /0.31/0.386 |
| p-hat300-2  (25) | 25(30)/25.00  /0/0.189 | 25(30)25.00  /0/16.487 | 25(30)/25.00  /0/0.107 | 25(30)/25.00  /0/5.314 |
| p-hat300-3  (36) | 36(30)/36.00  /0/0.339 | 36(27)/35.83  /0.52/6.241 | 36(30)/36.00  /0/0.443 | 36(29)/35.97  /0.18/0.615 |
| p-hat700-2  (44) | 44(30)/44.00  /0/1.254 | 44(25)43.83  /0.37/134.72 | 44(30)/44.00  /0/2.887 | 44(30)/44.00  /0/3.475 |
| p-hat700-3  (62) | 62(30)/62.00  /00.426 | 62(21)/61.67  /0.54/45.143 | 62(30)/62.00  /0/4.557 | 62(29)/61.97  /0.18/4.401 |
| p\_hat1500-2  (65) | 65(30)/65.00  /0/19.074 | 65(22)/64.70  /0.53/150.00 | 65(30)/65.00  /0/31.349 | 65(29)/64.97  /0.18/18.622 |
| p\_hat1500-3  (94) | 94(30)/94.00  /0/13.041 | 94(30)/94.00  /0/88.00 | 94(30)/94.00  /0/35.245 | 94(16)/93.53  /0.51/21.571 |

表4-5 CODE网络族上的综合对比实验结果

最优解(出现次数)/平均值/标准差/算法平均运行时间

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| network(best) | IGLS | ILS | SBTS | BLS |
| 1dc.1024  (94) | 94(10)/93.33  /0.47/23.576 | 94(4)/80.60  /3.70/14.00 | 94(11)/93.37  /0.48/109.31 | 94(16)/93.53  /0.51/13.222 |
| 1et.2048  (316) | 316(1)/314.33  /0.65/26.265 | 316(30)/316.00  /0/16.00 | 315(4)/308.73  /4.85/346.134 | 316(9)/315.07  /0.74/20.019 |
| 1tc.1024  (196) | 196(30)/196.00  /0/1.581 | 196(30)/193.00  /0/8.323 | 196(28)/195.93  /0.25/36.276 | 196(29)/195.97  /0.18/10.022 |
| 2dc.2048  (24) | 24(20)/23.67  /0.47/174.320 | 24(30)/24.00  /0/165.00 | 24(8)/23.27  /0.44/679.91 | 24(18)/23.60  /0.55/21.929 |

表4-6 BHOSLIB网络族上的综合对比实验结果

最优解(出现次数)/平均值/标准差/算法平均运行时间

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| network(best) | IGLS | ILS | SBTS | BLS |
| frb30-15-1  (30) | 30(21)/29.70  /0.46/42.383 | 30(30)/30.00  /0/9.00 | 30(29)/29.97  /0.18/36.529 | 30(30)/30.00  /0/1.338 |
| frb35-17-1  (35) | 35(2)/33.80  /0.50/60.174 | 35(28)/34.93  /0.18/13.00 | 35(1)/33.53  /0.56/182.555 | 35(15)/34.40  /0.70/2.564 |
| frb40-19-1  (40) | 40(13)/39.27  /0.73/75.368 | 40(30)/40.00  /0/19.00 | 40(6) /38.80  /0.75/225.593 | 40(30)/40.00  /0/4.018 |
| frb45-21-1  (45) | 45(1)/43.30  /0.53/89.761 | 45(22)/44.47  /0.89/27.00 | 45(1)/43.10  /0.47/366.299 | 45(3)/43.60  /0.70/6.694 |

表4-7 BA网络族上的综合对比实验结果

最优解(出现次数)/平均值/标准差/算法平均运行时间

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| network | IGLS | ILS | SBTS | BLS |
| BA100  <4>\* | 59(30)/59.00  /0/0.038 | 59(30)/59.00  /0/0.823 | 59(30)/59.00  /0/0.003 | 59(30)/59.00  /0/0.104 |
| BA100  <8>\* | 43(30)/43.00  /0/0.057 | 43(30)/43.00  /0/0.136 | 43(30)/43.00  /0/0.005 | 43(30)/43.00  /0/0.106 |
| BA100  <10>\* | 41(30)/41.00  /0/0.061 | 41(30)/41.00  /0/0.185 | 41(30)/41.00  /0/0.006 | 41(30)/41.00  /0/0.104 |
| BA500  <4>\* | 290(30)/290.00  /0/0.066 | 290(30)/290.00  /0/0.874 | 290(30)/290.00  /0/18.145 | 290(30)/290.00  /0/3.843 |
| BA500  <8> | 222(30)/222.00  /0/0.983 | 222(24)/221.80  /0.40/1.114 | 222(30)/222.00  /0/25.001 | 222(18)/221.60  /0.50/3.957 |
| BA500  <10> | 205(29)/204.97  /0.18/1.059 | 205(14)/204.43  /0.56/1.345 | 205(20)/205.00  /0/27.339 | 205(1)/203.03  /0.89/5.791 |
| BA1000  <4> | 574(30)/574.00  /0/1.229 | 574(30)/574.00  /0/2.716 | 574(4)/573.00  /0.52/46.308 | 574(24)/573.80  /0.45/5.926 |
| BA1000  <8> | 461(30)/461.00  /0/1.432 | 462(9)/461.10  /0.70/3.494 | 461(6)/459.80  /0.88/59.582 | 461(9)/459.80  /1.23/5.932 |
| BA1000  <10> | 420(30)/420.00  /0/1.520 | 420(11)/418.93  /0.93/3.293 | 419(3)/417.23  /0.99/54.095 | 420(12)/419.00  /0.94/5.974 |
| BA2000  <4> | 1165(30)/1165.00  /0/2.172 | 1165(30)/1165.00  /0/9.532 | 1165(2)/1162.77  /1.36/71.361 | 1165(30)/1165.00  /0/17.216 |
| BA2000  <8> | 902(9)/901.20  /0.65/2.475 | 903(1)/899.87  /1.45/8.254 | 899(2)/895.83  /1.69/111.177 | 900(6)/899.00  /0.71/17.659 |
| BA2000  <10> | 837(1)/834.03  /0.98/2.467 | 835(1)/832.17  /1.95/8.551 | 834(1)/827.87  /2.63/106.638 | 834(18)/833.60  /0.55/17.826 |

表4-8 ER网络族上的综合对比实验结果

最优解(出现次数)/平均值/标准差/算法平均运行时间

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| network | IGLS | ILS | SBTS | BLS |
| ER100  <4>\* | 47(30)/47.00  /0/0.041 | 47(30)/47.00  /0/0.135 | 47(30)/47.00  /0/0.004 | 47(30)/47.00  /0/0.090 |
| ER100  <8>\* | 35(30)/35.00  /0/0.062 | 35(27)/34.90  /0.30/0.139 | 34(30)/34.00  /0/0.005 | 35(29)/34.97  /0.18/0.087 |
| ER100  <10>\* | 31(30)/31.00  /0/0.076 | 31(14)/30.47  /0.50/0.179 | 30(30)/30.00  /0/0.006 | 31(12)/30.40  /0.50/0.092 |
| ER500  <4>\* | 245(30)/245.00  /0/0.145 | 245(30)/245.00  /0/1.570 | 245(30)/245.00  /0/20.069 | 245(3)/241.73  /1.89/3.372 |
| ER500  <8> | 177(29)/176.97  /0.18/4.970 | 177(3)/174.80  /1.47/1.024 | 177(30)/177.00  /0/26.204 | 177(3)/174.77  /1.38/2.630 |
| ER500  <10> | 155(23)/154.73  /0.51/0.450 | 155(4)/153.40  /1.08/1.186 | 155(30)/155.00  /028.271/ | 155(2)/153.60  /0.81/4.069 |
| ER1000  <4> | 479(12)/478.40  /0.49/6.087 | 479(1)/476.20  /1.62/2.210 | 478(3)/474.70  /2.04/49.699 | 479(1)/476.25  /1.37/6.682 |
| ER1000  <8> | 355(4)/353.20  /1.14/7.055 | 352(3)/349.43  /1.71/2.443 | 355(2)/351.43  /2.09/49.358 | 353(8)/351.13  /1.46/8.002 |
| ER1000  <10> | 313(14)/312.03  /1.05/7.505 | 313(1)/308.43  /1.93/2.693 | 313(8)/309.97  /3.07/56.752 | 312(6)/308.50  /2.27/8.286 |
| ER2000  <4> | 953(23)/952.70  /0.59/10.940 | 951(1)/948.23  /1.36/7.046 | 950(1)/945.93  /2.53/81.049 | 952(6)/949.80  /1.48/17.288 |
| ER2000  <8> | 696(2)/692.03  /2.07/11.633 | 691(2)/684.30  /3.03/6.372 | 688(1)/679.73  /4.73/102.082 | 691(18)/690.00  /1.41/18.104 |
| ER2000  <10> | 629(2)/625.33  /2.15/12.103 | 624(1)/616.90  /2.95/6.544 | 623(2)/616.37  /3.59/118.952 | 620(24)/619.20  /1.79/14.018 |

表4-9 WS网络族上的综合对比实验结果

最优解(出现次数)/平均值/标准差/算法平均运行时间

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| network | IGLS | ILS | SBTS | BLS |
| WS100<10>\*  p=0.1 | 21(30)/21.00  /0/0.071 | 21(26)/20.87  /0.34 /0.167 | 21(30)/21.00  /0/0.009 | 21(28)/20.93  /0.25/0.092 |
| WS100<10>\*  p=0.5 | 38(30)/38.00  /0/0.087 | 38(29)/37.97  /0.18/0.248 | 38(30)/38.00  /0/0.101 | 38(30)/38.00  /0/0.095 |
| WS500<10>  p=0.1 | 112(30)/112.00  /0/6.198 | 112(4)/112.10  /1.16/0.946 | 112(1)/110.07  /1.15/30.242 | 112(25)/111.83  /0.38/3.448 |
| WS500<10>  p=0.5 | 187(30)/187.00  /0/13.559 | 187(17)/186.30  /0.97/1.828 | 187(3)/184.83  /1.49/32.046 | 187(23)/186.77  /0.43/3.913 |
| WS1000<10>  p=0.1 | 224(23)/223.77  /0.42/9.546 | 224(23)/222.47  /1.36/2.080 | 222(4)/218.57  /2.0154.635/ | 224(3)/222.57  /0.73/6.689 |
| WS1000<10>  p=0.5 | 379(22)/378.73  /0.44/25.243 | 379(21)/377.47  /1.18/7.156 | 377(5)/374.43  /1.71/65.256 | 379(6)/377.63  /0.96/7.424 |
| WS2000<10>  p=0.1 | 453(3)/451.20  /1.11/15.500 | 452(1)/447.63  /7.40/4.560 | 447(1)/436.50  /4.66/107.519 | 451(6)/448.60  /1.95/16.689 |
| WS2000<10>  p=0.5 | 763(10)/762.10  /0.79/48.756 | 763(2)/758.37  /6.08/53.524 | 754(1)/747.40  /3.16/116.397 | 760(18)/759.20  /1.30/15.945 |

表4-10 LFR网络族上的综合对比实验结果

最优解(出现次数)/平均值/标准差/算法平均运行时间

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| network | IGLS | ILS | SBTS | BLS |
| LFR100  <15> 50\* | 32(30)/32.00  /0/0.098 | 32(27)/31.90  /0.30/0.412 | 32(30)/32.00  /0/0.007 | 32(30)/32.00  /0/0.974 |
| LFR100  <20> 60\* | 35(30)/35.00  /0/0.118 | 35(30)/35.00  /0/0.616 | 35(30)/35.00  /0/0.008 | 35(30)/35.00  /0/0.126 |
| LFR100  <25> 70\* | 30(30)/30.00  /0/0.145 | 30(30)/30.00  /0/0.848 | 30(30)/30.00  /0/0.010 | 30(30)/30.00  /0/0.092 |
| LFR500  <15> 50 | 159(30)/159.00  /0/6.626 | 159(1)/158.73  /0.73/2.061 | 159(17)/158.57  /0.50/36.568 | 159(24)/158.80  /0.41/3.536 |
| LFR500  <20> 60 | 162(30)/162.00  /0/1.601 | 162(1)/161.93  /0.36/2.707 | 162(30)/162.00  /0/42.003 | 162(30)/162.00  /0/3.562 |
| LFR500  <25>70 | 129(29)/128.97  /0.18/1.824 | 130(1)/128.93  /0.36/3.460 | 129(23)/128.77  /0.42/50.274 | 129(23)/128.77  /0.43/3.326 |
| LFR1000  <15>50 | 354(30)/354.00  /0/8.810 | 354(8)/352.87  /0.92/4.826 | 354(3)/351.90  /1.58/81.474 | 354(23)/353.77  /0.43/7.841 |
| LFR1000  <20>60 | 289(27)/288.90  /0.30/9.528 | 289 (21)/289.13  /0.81/4.996 | 289(5)/287.13  /1.61/100.532 | 288(15)/287.50  /0.53/7.666 |
| LFR1000  <25>70 | 272(30)/272.00  /0/11.534 | 273(2)/271.37  /0.84/6.709 | 272(16)/271.27  /0.89/102.483 | 272(18)/271.60  /0.52/7.414 |
| LFR2000  <15>50 | 670(26)/669.87  /0.34/13.740 | 670(20)/668.67  /1.27/10.045 | 668(6)/665.57  /1.82/159.678 | 670(3)/668.40  /0.97/14.946 |
| LFR2000  <20>60 | 600(11)/599.27  /0.68/14.375 | 600(1)/597.47  /1.335/11.760 | 598(3)/593.93  /2.50/164.633 | 600(15)/599.20  /1.03/16.678 |
| LFR2000  <25>70 | 520(23)/519.77  /0.42/15.882 | 520(4)/518.00  /1.34/14.171 | 518(6)/514.10  /3.37/198.214 | 520(23)/519.77  /0.43/16.763 |

### 4.3.5 实验结果分析与统计检验

由表4-4至表4-10所示的实验结果可以得到，我们提出的IGLS算法在BA网络、ER网络、WS网络和LFR网络上一直优于其他算法。在DIMACS网络族和CODE网络族的大多数网络，也可以得到最优解，但是，在BHOSLIB网络族上，IGLS无法得到最优解。

为了从统计学角度验证IGLS的性能，我们使用表4-4至表4-10所示的74个不同网络上的结果，通过统计检验[[67]](#endnote-67)的方式验证IGLS是否优于其他算法，使用实验结果中的平均值进行t检验。

1、IGLS算法与ILS算法的t检验

在这74个网络中，IGLS算法所得的平均值与ILS算法所得平均值之间的差记为，={0,0,0.73,1.63, ……,-0.23,0.63,1.20,1.80,1.77}，表示IGLS算法在某个网络上独立运行30次所得最大独立集规模的平均值。因为这些网络相互独立，所以中的元素相互独立且服从相同的分布，不失一般性，我们假设中的元素服从正态分布，即。

在t检验中，我们的假设为：

：IGLS算法得到的独立集的大小比ILS算法得到的独立集大；

：IGLS算法得到的独立集的大小不比ILS算法得到的独立集大.

这种假设等价于如下形式：

：；

：.

拒绝域为：

 (4-1)

其中，和分别表示中元素的均值和方差，表示样本数目，即中元素的个数，表示置信度为自由度为的t分布的值。

根据，可得，，因此。给定置信度，查表得到。因此恒成立，所以根据实验结果计算得到的值不在拒绝域内，所以接受假设，即IGLS算法在这些实验网络上得到的效果比ILS算法的效果好。

2、IGLS算法与其他算法的t检验

对IGLS算法和SBTS算法进行t检验，同样计算，作出假设：

：IGLS算法得到的独立集的大小比SBTS算法得到的独立集大；

：IGLS算法得到的独立集的大小不比SBTS算法得到的独立集大.

计算中所有差值的平均值和方差，并计算，最终得到成立，所以在置信度时，接受假设，IGLS算法在这些实验网络上比SBTS算法更优。

同样进行IGLS算法与BLS算法的t检验，可知IGLS算法比BLS算法更优。

# 第五章 基于雪堆博弈求解最小顶点覆盖的自然进化算法

我们提出的基于雪堆博弈的自然进化算法（Game-based memetic algorithm for Minimum Vertex Cover，简称GMA-MVC），使用遗传算法框架，并让复杂网络进行异步雪堆博弈演化，每一个节点作为完全理性的博弈参与者，如果选择策略，则该节点在解集中，否则，选择策略的节点不在解集中。

在遗传算法框架中，每个染色体代表一个解集，在GMA-MVC中，对应于网络博弈的策略选择，将染色体进行二进制编码，对于个节点的网络，有个二进制位，如果第位编码为1，则对应于第个节点选择策略，即该节点在解集中，第位编码为0则表示该节点不在解集中。博弈中的策略对应于解集中的节点，设种群规模为。

## 5.1 个体进化

### 5.1.1 基于雪堆博弈求解合法解

对于最小顶点覆盖问题，其约束条件为任意边至少有一个顶点在顶点集中，为了满足这个条件，HGA算法设计了启发式交叉算子，GA算法为每一个个体赋予用公式(2-1)计算的指标。

我们提出的基于雪堆博弈的算法，可以通过异步博弈方式满足问题的约束条件。当雪堆博弈参数满足且时，定理3证明了博弈一定收敛到纳什均衡，引理1证明了网络雪堆博弈的严格纳什均衡状态与网络顶点覆盖集之间满足关系。所以，结合定理3和引理1，我们可以推出：网络进行异步雪堆博弈，且参数满足、时，成立。

在遗传算法中，种群中个体通过交叉、变异使用全局信息，但是，对于顶点覆盖问题，交叉、变异操作后，将不能保证每个个体仍满足约束条件。在GMA-MVC中，种群执行交叉、变异操作后，所有个体分别由当前状态进行异步雪堆博弈直至严格纳什均衡状态，因为，所以一定可以保证种群中每一个个体都是满足约束条件的合法解。

### 5.1.2 基于雪堆博弈的局部搜索

设无权无向网络处于雪堆博弈的严格纳什均衡状态，表示严格纳什均衡状态下所有选择策略的节点构成的集合，表示中与中的节点只有一条连边的节点构成的集合，，设中与中的节点至少有两条连边的阶段构成集合，即。我们提出的基于雪堆博弈的局部搜索过程为：

1. 随机选择，将由变为；
2. 节点的邻居节点及节点进行异步雪堆博弈，定理5将证明这一过程会使网络状态收敛到严格纳什均衡，得到新；
3. 更新和，重复执行步骤1)和步骤2)次。

**定理5：**设无权无向网络进行异步雪堆博弈，达到，随机选择节点并使其邻居节点及其自身进行异步雪堆博弈至严格纳什均衡，则网络也处于严格纳什均衡状态。

**证明：**（1）由引理2可得，在时，节点的所有邻居节点都选择策略，将其邻居节点划分为两个集合：，，则中的节点只有节点这一个选择策略的邻居节点，中的节点至少有2个选择策略的邻居节点（其中有一个是节点）；由定理5的前提条件可知节点至少有两个邻居节点在集合中，即。（2）当节点的策略由改为后，中的节点至少有1个选择的邻居节点，由引理2可知，这些节点不会更新策略。中的节点没有选择的邻居节点，满足这个条件的节点在博弈后会将策略由改为。如果中各节点之间互相不相邻，则中某个节点的策略改变不会影响中的其他节点，所以中所有节点都会选择；如果中的节点存在邻居关系，则至少也会有1个节点在博弈后将策略由改为。对于中的节点的邻居节点，这些节点在原时就至少有一个选择策略的邻居节点，所以中的节点策略更新为之后，其邻居节点会依旧选择策略。所以，中的节点的策略更新不会影响其邻居节点的策略选择。（3）综上所述，节点的邻居节点及重新博弈至严格纳什均衡状态时，整个网络也处于严格纳什均衡状态，且解集可以得到改进。

图5-1演示了基于雪堆博弈的局部搜索过程。

 

(a)  (b) 

 

(c) 博弈 (d) 

图5-1 基于雪堆博弈的局部搜索过程。(a) 初始严格纳什均衡状态，对应顶点覆盖集，，；(b) 根据定义得到；(c) 节点按照顺序进行异步雪堆博弈至严格纳什均衡状态；(d) 得到网络的新严格纳什均衡状态，对应顶点覆盖集，也是最小顶点覆盖集。

### 5.1.3 个体进化总体步骤

我们提出的个体进化过程共分为两步：

1. 网络节点进行异步雪堆博弈直至严格纳什均衡状态，得到合法的顶点覆盖集；
2. 对于状态，基于雪堆博弈进行局部搜索。

## 5.2 带有个体进化的自然进化算法GMA-MVC

### 5.2.1 基于节点度的初始化方法

在GA、HGA和MBR算法中，都采用随机初始化的方式，我们提出的GMA-MVC算法使用基于节点度的初始化方法，对于度值较大的节点，以较大的概率使该节点选择合作策略。对于节点，其选择策略的概率为：

 (5-1)

其中，表示网络中节点的度值，表示网络节点总度值。根据公式(5-1)，节点度越大，则在初始化策略时，该节点选择策略的概率越大。

### 5.2.2 GMA-MVC算法元素

1、个体

网络节点进行博弈，每个节点所选择的策略代表该节点是否在解集中，如果节点选择策略，则该节点在解集中。所有节点的策略选择组成网络的博弈状态，即网络的解集状态，以图5-1所示的网络为例，博弈状态对应于解集为。在GMA-MVC算法中，将每一个博弈状态作为一个个体，设种群规模为，则共有个个体，可以表示个博弈状态。

2、个体进化

在GMA-MVC算法中，每个个体都进行5.1节中介绍的个体进化，当个体经过交叉、变异操作后，通过执行个体进化，收敛到合法的顶点覆盖集，并通过局部搜索对当前解集进行一定程度的改进。

3、适应度函数

在进化算法中，个体按照适应度函数得到个体适应度值，适应度值用于评判个体状态与目标状态之间的差异，也可以对比个体之间的优劣。在GMA-MVC算法中，适应度函数如公式(5-2)[[68]](#endnote-68)所示：

 (5-2)

其中，表示当前个体的博弈状态，如果，即节点在解集中，则；如果，则。该公式的第一部分统计了解集中的节点数目。第二部分以系数（为网络节点总数）惩罚未被覆盖的边，当节点未被覆盖，节点的邻居节点也未被覆盖时，即边未被覆盖，此时结果为。当解集是一个合法的顶点覆盖集时，公式(5-2)计算的就是解集中节点的个数。

在进行对比实验时，为保证统一性，都采用公式(5-2)计算搜索到的解集的适应度值。

4、交叉变异

与HGA和GA不同，GMA-MVC算法通过博弈的方式将解集约束在合法解的范围内，因此，在进行交叉、变异时，不需要额外考虑约束条件的影响，所以在GMA-MVC算法中，采用简单的两点交叉、随机变异的方式。

两点交叉是指，从种群中随机选择两个个体，然后产生两个1到之间的随机整数，进行交叉的这两个个体将两个随机数之间的部分进行交换，产生两个子代个体。

个体进行单点变异是指，设定变异概率，产生0到1之间的随机小数，当这个随机数小于时，该个体进行变异，否则不进行变异。对发生变异的个体，产生一个1到之间的随机整数，将个体第位的状态反转。

5、种群进化

GMA-MVC算法使用遗传算法框架，因此要通过种群进化不断搜索并使用全局信息。种群进化过程为：

1. 种群中任意个体进行交叉、变异产生2个子代个体；
2. 子代个体进行个体进化；
3. 从父代个体和子代个体中选择两个较优的个体，进入下一轮种群进化，作为下一轮种群进化的父代个体。
4. 将种群中所有个体都执行一轮步骤1)至步骤3)，完成一轮种群进化，产生规模为的子代种群。

### 5.2.3 GMA-MVC算法总体步骤

GMA-MVC的算法流程图如图5-2所示：

图5-2 GMA-MVC算法流程图

图5-2中，表示种群规模；表示种群最大迭代次数；表示个体变异概率；表示雪堆博弈参数，取，恒满足；用于记录迭代次数。

## 5.3 实验结果与分析

我们分别在不同网络上做实验，验证基于度的初始化方法的性能、个体进化的性能，并将GMA-MVC算法与MBR算法43、GA算法33、HGA算法38、SBTS算法35进行对比。

### 5.3.1 实验网络简介

与第四章4.3.1节所介绍的网络类似，这里主要使用BHOSLIB网络62、BA网络63、ER网络64、WS网络65和LFR网络66进行实验。4.3.1节中有对这几类网络的详细介绍。

在实验中，我们还使用了一类PS网络[[69]](#endnote-69)，PS网络常用于评估求解最小顶点覆盖问题的算法性能，标准PS网络共有个节点，包含两行含有个节点的行，第三行含有个节点，第一行的每个节点连接到同一列上的第二行的各节点，第二和第三行各节点相互连接，总共有条边。图5-3表示了一个的PS网络及其顶点覆盖的情况。



(a) 网络结构示意图

 

(b) 顶点覆盖 (c) 最小顶点覆盖

图5-3 的PS网络。(a) PS网络结构示意图；(b) 顶点覆盖；(c) 最小顶点覆盖

我们使用了1000个节点的PS网络进行实验，通过得到，可知该网络的最小顶点覆盖集中共有334个节点。

### 5.3.2 初始化的作用

为验证GMA-MVC算法中基于节点度的初始化方法的作用，我们对GMA-MVC算法进行两类实验，一类使用均匀随机方式进行初始化，另一类采用基于节点度进行初始化。在每个网络上，GMA-MVC算法分别使用不同的方式产生初始种群，使用相同的参数：；；；；。对每个实验网络都独立运行30次，统计30次独立运行所得适应度值的平均值、标准差和平均运行时间。得到的实验结果如表5-1所示：

表5-1 GMA-MVC中均匀随机初始化与基于度初始化的对比

适应度平均值/标准差/平均运行时间

|  |  |  |
| --- | --- | --- |
| network | Uniform random | Degree-based |
| WS200p=0.1<4> | 128.00/0/2.34 | 127.00/0/2.12 |
| BA1024<8> | 553.10/0.70/13.59 | 551.60/0.49/8.12 |
| ER500<10> | 347.10/0.30/2.67 | 346.70/0.64/2.63 |
| ER2000<8> | 1318.50/2.37/41.64 | 1316.70/2.15/40.64 |

从表5-1可以看出，使用基于度值的初始化方式时，适应度值和运行时间都较好，基于度值进行初始化时，种群收敛到严格纳什均衡状态的时间缩短，因此算法整体的运行时间也较短。

对于200个节点、平均度为4的WS网络，均匀随机初始化时，GMA-MVC在30次运行中得到的结果都是128，然而基于度值初始化时，30次的结果都是127，也就是说，在这30次运行中，基于度值的方法严格比均匀随机方法好的概率为100%。对于1024个节点、平均度为8的BA网络，均匀随机方式所得的适应度值在30次运行中有9次554、15次553和6次552，基于度值的方式得到18次552和12次551，所以，在这30次独立运行中，基于度值的方法一定不比均匀随机的结果差，且以概率保证比均匀随机方式得到更优的解。

对于500个节点、平均度为10的ER网络，均匀随机方式在30次独立运行中共得到3次348、27次347，基于度值的方式在30次独立运行中得到3次348、15次347和12次346，由此可以看出，基于度值的方式得到的解集不会比均匀随机的方式得到的解集的适应度值差，但是，严格比均匀随机方式好的概率仅为。使用同样的分析方式，对于对于2000个节点、平均度为8的ER网络，基于度值的方式严格比均匀随机的方式好的概率为50%。

根据这几类网络的特性，ER网络的度分布是均匀随机的，所以基于度值的方式对于具有这种度分布的网络，其作用效果退化为与均匀随机方式的作用效果类似，这可以解释基于度值的初始化方式在ER网络上的改进效果不明显。

### 5.3.3 个体进化的作用

将GMA-MVC算法中的个体进化部分去除后进行实验，用于验证个体进化在GMA-MVC算法中的作用。此时GMA-MVC算法退化为普通的遗传算法，仅仅使用公式(5-2)所示的适应度函数来搜索满足约束条件的个体。GMA-MVC算法参数和实验网络与5.3.2节相同，实验结果如表5-2所示。

表5-2 GMA-MVC中个体进化的作用

适应度平均值/标准差/平均运行时间

|  |  |  |
| --- | --- | --- |
| network | 有个体进化 | 没有个体进化 |
| WS200p=0.1<4> | 127.00/0/2.12 | 130.20/0.60/0.12 |
| BA1024<8> | 551.60/0.49/8.12 | 733.53/7.43/0.58 |
| ER500<10> | 346.70/0.64/2.63 | 383.30/1.27/0.30 |
| ER2000<8> | 1316.70/2.15/40.64 | 1453.80/3.28/1.16 |

由表5-2可知，个体进化对于GMA-MVC算法起到非常重要的作用，在所有网络上，最终得到的顶点覆盖集的适应度值都明显变小。

### 5.3.4 基于演化博弈的不同算法对比

受MBR算法的启发，我们提出GMA-MVC算法。MBR算法在基于博弈的基础上，每个网络节点需要带有博弈记忆，记忆长度是MBR算法的重要参数。GMA-MVC算法借鉴遗传框架，重要参数是种群迭代次数。为更好地对比MBR算法与GMA-MVC算法，分别对它们设置不同的参数取值，在WS1000<4>、Dolphin、LFR500<15>和BA2000<4>网络上进行实验，对每个实验网络独立运行10次，并统计10次独立运行结果的平均值和平均运行时间，实验结果如表5-3所示。GMA-MVC的其他实验参数为；；；。

表5-3 GMA-MVC与MBR不同参数取值实验结果

适应度平均值/平均运行时间

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| results network  algorithm | | WS1000<4> | Dolphin | LFR500<15> | BA2000<4> |
| MBR |  | 654.20/0.11 | 34.70/0.06 | 333.17/4.31 | 847.30/1.12 |
|  | 643.50/59.85 | 34.50/19.19 | 329.00/117486.00 | 841.30/2.16 |
|  | N/A | 34.40/4657.29 | N/A | 841.00/6.54 |
| GMA-MVC |  | 640.20/5.41 | 34.00/0.13 | 329.60/1.96 | 841.80/9.84 |
|  | 636.30/19.32 | 34.00/0.54 | 328.20/8.42 | 840.70/22.73 |
|  | 635.90/34.71 | 34.00/1.00 | 328.00/16.05 | 840.30/38.22 |
|  | 635.60/71.90 | 34.00/2.08 | 328.00/31.92 | 840.30/72.18 |

注：划横线的结果表示时算法不收敛，故采用；N/A表示算法不收敛，无法得到有效值。

由表5-3中的实验结果可以看出，对于WS1000<4>网络，GMA-MVC算法不论种群迭代次数取值多少，总能得到比MBR更好的结果。对于Dolphin海豚网络，GMA-MVC算法都比MBR效果好，且总能搜索到最优解。对于LFR500<15>网络，MBR算法在、时无法收敛，当时，GMA-MVC算法在不同参数取值下得到的结果总比MBR的结果好。对于BA2000<4>网络，GMA-MVC算法也总能得到比MBR算法更好的结果。

GMA-MVC算法和MBR算法都基于雪堆博弈的思想求解最小顶点覆盖问题，由实验结果可以推导出，GMA-MVC算法效果比MBR算法效果好。MBR算法仅通过带记忆机制的博弈方式搜索解集，只使用了网络的局部信息，而GMA-MVC算法的个体进化机制基于博弈思想使用局部信息，同时在遗传算法框架下通过种群进化使用全局信息，所以GMA-MVC算法的效果较好。同时，还可以看出MBR存在局限性，该算法对网络拓扑结构有一定的依赖性，当记忆长度较大时，对于WS网络和LFR网络，MBR算法无法收敛，但是对于BA网络，却可以很快收敛。

### 5.3.5 综合对比实验

将GMA-MVC算法分别与MBR算法43、GA算法33、HGA算法38、SBTS算法35进行对比，在每个网络上，每个算法都独立运行10次，并统计适应度值的平均值、标准差和平均运行时间，实验结果如表5-4。表5-5记录了这几种算法在已知最优解的BHOSLIB网络上的实验结果，独立运行10次并统计10次运行中的解集的最优适应度值和平均值。

GMA-MVC算法的实验参数为：；；；；。

HGA算法的实验参数为：；；。

GA算法的实验参数为：；；；。

MBR算法的实验参数为：；。

SBTS算法的实验参数为：局部搜索交换次数为10万次。

表5-4 综合对比实验

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| network | GMA-MVC | MBR | HGA | GA | SBTS |
| ps1000 | **334.00**  /0  /2.09 | **334.00**  /0  /1.75 | **334.00**  /0  /962.22 | 666.00  /0  /1634.00 | **334.00**  /0  /51.67 |
| WS100p=0.1 | **64.00**  /0  1.12 | 65.40  /0.49  /4818.61 | **64.00**  /0  /23.44 | 94.80  /0.98  /25.30 | **64.00**  /0  /0.306 |
| WS100p=0.5 | **55.00**  /0  /0.19 | 57.20  /0.60  /440.99 | 57.10  /0.83  /20.83 | 89.00  /0  /25.30 | **55.00**  /0  /0.32 |
| WS500p=0.1 | **314.00**  /0  /9.15 | 317.90  /0.70  /27328.90 | 316.30  /0.46  /340.01 | 484.20  /5.81  /572.60 | 315.87  /1.31  /2.00 |
| WS1000p=0.1 | **635.90**  /0.83  /34.71 | N/A,655.50  /2.58  /0.52 | 638.90  /0.30  /1933.34 | 943.40  /3.53  /5109.00 | 640.43  /2086  /3.09 |
| LFR1000<15> | **668.50**  /0.50  /46.67 | N/A,673.80  /1.47  /1.55 | 675.00  /0  /1920.48 | 914.20  /1.33  /4982.00 | 670.40  /2.10  /7.61 |
| LFR1000<20> | **746.20**  /0.87  /25.46 | N/A,755.70  /2.10  /6.28 | 753.70  /0.46  /7221.46 | 937.40  /1.50  /2231.00 | 747.70  /2.12  /8.52 |
| LFR2000<20> | **1481.40**  /0.92  /246.79 | N/A,1496.80  /2.12  /6.28 | 1497.80  /1.25  /6912.42 | 1867.80  /3.82  /9700.00 | 1487.70  /3.28  /16.70 |
| LFR2000<25> | **1598.80**  /1.08  /135.18 | N/A,1614.50  /2.58  /8.00 | 1604.90  /1.50  /8297.97 | 1901.80  /3.25  /10267.00 | 1603.50  /3.45  /19.48 |
| BA1024<4> | **429.10**  /0.54  /8.93 | 429.20  /0.75  /3.62 | 430.21  /0.40  /1201.48 | 503.60  /0.80  /4813.00 | 429.30  /0.86  /3.31 |
| BA2000<4> | **840.30**  /0.64  /38.22 | 840.90  /0.83  /7.60 | 844.80  /0.87  /2331.83 | 1001.00  /0.49  /9556.00 | 842.43  /1.32  /6.63 |
| BA1024<8> | **551.60**  /0.49  /8.12 | 553.30  /1.10  /5.18 | 559.40  /0.92  /1499.40 | 678.30  /0.80  /4940.00 | 552.57  /1.67  /4.85 |
| ER100<8> | **65.00**  /0  /0.37 | 66.00  /0  /12.21 | **65.00**  /0  /29.37 | 83.00  /0  /6.83 | **65.00**  /0  /0.45 |
| ER2000<8> | **1317.70**  /2.45  /49.15 | 1332.00  /2.16  /8.10 | 1341.00  /1.00  /6081.48 | 1756.00  /0.18  /9063.12 | 1318.90  /2.91  /10.33 |
| ER2000<10> | **1385.50**  /2.64  /46.44 | 1395.33  /1.25  /6.43 | 1407.10  /1.04  /6477.56 | 1794.00  /0.60  /9122.00 | 1386.00  /2.72  /11.30 |

注：MBR中N/A,655.50/2.58/0.52表示当MBR算法取时无法收敛，当取时，结果为N/A,655.50/2.58/0.52。

表5-5 标准测试集网络上的对比试验

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| network | best | GMA-MVC | MBR | HGA | GA | SBTS |
| frb30-15-1 | 420 | **420**  **/420** | N/A,423  /423.80 | 424  /424.80 | 444.40  /445.20 | **420**  /420.90 |
| Frb40-19-1 | 720 | **721**  **/721** | N/A,726  /727.40 | 727  /727.10 | 753.00  /754.60 | **721**  /722.00 |
| Frb45-21-1 | 900 | **901**  **/901.60** | N/A,906  /908.30 | 907  /907.20 |  | 902  /902.90 |

由表5-4可以看出，GMA-MVC算法在大多数网络中都得到最好的结果。MBR算法对网络结构依赖性大，在BA网络上表现出的性能较好，但是对于WS网络和LFR网络，采用相同的记忆长度时，需要较长的时间才能收敛或无法收敛。SBTS算法性能较好，可以得到与GMA-MVC相差较小的结果，同时运行时间较短。由表5-5可以看出。GMA-MVC算法均可以搜索到最优解，且十次独立运行结果的平均值最小，由于其他几种算法，SBTS算法也可以搜索到与GMA-MVC算法一样好的最优解，但是平均性能较差。

### 5.3.6 实验结果分析与统计检验

使用表5-4所示的实验结果，对GMA-MVC算法和MBR算法、GMA-MVC算法和SBTS算法分别进行t检验。t检验的具体过程在第四章4.3.5节有详细介绍。

1. GMA-MVC算法和MBR算法的t检验

在表5-4所示的15个网络的实验结果中，计算GMA-MVC算法与MBR算法适应度值平均值之间的差值，中的元素互相独立且服从正态分布，做出假设如下：

：GMA-MVC得到的平均值比MBR的小；

：GMA-MVC得到的平均值不比MBR的小.

这种假设等价于如下形式：

：；

：.

拒绝域为：

 (5-3)

根据，计算得到，，，给定置信度，查表得到。因此恒成立，所以根据实验结果计算得到的值不在拒绝域内，所以接受假设，所以GMA-MVC得到的实验结果比MBR的结果好。

1. GMA-MVC算法和SBTS算法的t检验

首先计算得到，然后做出假设如下：

：GMA-MVC得到的平均值比SBTS的小；

：GMA-MVC得到的平均值不比SBTS的小.

根据，计算得到，给定置信度，查表得到。因此恒成立，所以接受假设，所以GMA-MVC得到的实验结果比SBTS的结果好。

将GMA-MVC算法与其他算法进行统计检验，同样可以验证GMA-MVC算法的优势。

### 5.3.7 算法参数分析

GMA-MVC算法的参数分别为种群规模、种群最大迭代次数、个体变异概率、雪堆博弈参数和个体进化中局部搜索次数。只要参数满足，就可以保证引理1、引理2和定理5的成立，取。个体变异概率是一个较小的小数，因此对算法性能影响较小。种群规模、种群最大迭代次数和个体进化中局部搜索次数是对算法性能影响最大的三个实验参数，因此分别针对他们进行实验并根据实验结果分析其影响。

HGA算法中的重要参数同样为种群规模、种群最大迭代次数，因此将GMA-MVC算法与HGA算法在不同参数取值下分别对LFR1000<15>网络进行10次独立实验，将实验结果作出图5-4。



(a) 平均适应度值 (b) 平均运行时间

图5-4 参数的影响。(a) 平均适应度值；(b) 平均运行时间

由图5-4(a)可知，（1）保持不变时，当时，随着的增加，GMA-MVC和HGA的平均适应度值的下降速度都较快，当时，的增加对实验结果的改进不明显。（2）保持不变时，当从25增大到100时，GMA-MVC算法和HGA算法的适应度值都得到明显改进。（3）对于相同的和，GMA-MVC总能得到比HGA好的结果。（4）保持不变时，当从100增大到300时，GMA-MVC和HGA的结果均能得到小幅度改进，同时，由图5-4(b)可以看出，运行时间急剧增加，因此在进行实验时，对于GMA-MVC，我们采用和，鉴于HGA效果较差，综合考虑实验效果和运行时间，对于HGA，我们采用和。

在GMA-MVC的个体进化过程中，需要进行次局部搜索。如果参数很大，那么会影响算法速度，同时，如果陷入了局部最优解，那么搜索次数的增加也不会产生积极作用；如果参数很小，那么对当前解集进行的改进程度弱，无法发挥个体进化的作用。因此，我们在LFR1000<15>和WS1000(p=0.1)这两个网络上分别进行实验，保证GMA-MVC的其他参数、、、 不变，将分别取1、2、4、6、8、10、12、14、16、18和20，在每个网络上独立运行10次，将10次运行结果的平均适应度值作出图5-5所示的折线图。

由图5-5可以看出，总体来讲，当增加时，解集规模不断变小。但是，当时，随着的增加，解集的改进并不明显，同时，由5-5(c)可以看出，运行时间与成正比。所以，综合考虑算法性能与运行速度，在实验中取。



(a) 对LFR1000<15>网络的影响 (b) 对WS1000(p=0.1)网络的影响



(c) 对运行时间的影响

图5-5 参数对GMA-MVC性能的影响。(a) 对LFR1000<15>网络的影响；(b) 对WS1000(p=0.1)网络的影响；(c) 对运行时间的影响。

# 第六章 总结与展望

## 6.1 总结

## 6.2 展望

# 致 谢

# 参考文献

1. Rapoport A, Horvath W J. A study of a large sociogram[J]. Behavoiral Science, 1961, 6(4):279-291. [↑](#endnote-ref-1)
2. Martinez N D. Artifacts or attributes Effects of resolution on the Little Rock Lake food web[J]. Ecological Monographs, 1991, 61(4):367-392. [↑](#endnote-ref-2)
3. B. Bollobás. Random Graphs[M]. London: Academic Press, 1985. [↑](#endnote-ref-3)
4. B. Bollobás. Modern Graph Theory[M]. New York: Springer-Verlag, 1998. [↑](#endnote-ref-4)
5. P. Erdös, A. Rényi. On random Graphs[M]. Publicationes Mathematicae, 1959, (6): 290-297. [↑](#endnote-ref-5)
6. P. Erdös, A. Rényi. On the evo1ution of random graphs[J]. Publications of the Mathematical Institute of Hungarian Academy of Sciences, 1960, (5): 17-61. [↑](#endnote-ref-6)
7. D.J.Watts, S.H.Strogatz. Collective dynamics of small-world networks[J]. nature, 1998, 393(6684): 440-442. [↑](#endnote-ref-7)
8. 杜海峰,李树茁, W.F.Marcus等. 小世界网络与无标度网络的社区结构研究[J]. 物理学报，2007, 56(12):6886-6893. [↑](#endnote-ref-8)
9. AL Barabasi, R Albert. Emergence of Scaling in Random Networks[J]. Science, 1999, 286(5439):509-512. [↑](#endnote-ref-9)
10. Liu J. Comparison and analysis of matrix multiplications on GPU and CPU[J]. 计算机工程与应用, 2011, 47(19): 9-10. [↑](#endnote-ref-10)
11. Dorogovtsev S N, Mendes J F F. Evolution of networks[J]. Advances in physics, 2002, 51(4): 1079-1187. [↑](#endnote-ref-11)
12. Kernighan B W, Lin S. An efficient heuristic procedure for partitioning graph[J]. Bell system tecical journal, 1970, 49(2): 291-307. [↑](#endnote-ref-12)
13. 刘涛，陈忠，陈晓荣. 复杂网络理论及其应用研究概述[J]. 系统工程，2005, 23(6): 01-07. [↑](#endnote-ref-13)
14. 黄欣荣. 复杂网络：认知世界的科学新方法[J]. 江西财经大学学报, 2012, (79): 0089-0095. [↑](#endnote-ref-14)
15. 刘建香. 复杂网络及其在国内研究进展综述[J]. 系统科学学报, 2009, 17(4): 0031-0037. [↑](#endnote-ref-15)
16. 赵明，汪秉宏，蒋品群等. 复杂网络上动力学系统同步的研究进展[J]. 物理学进展. 2005, 25(3): 273-295. [↑](#endnote-ref-16)
17. 朱涵，王欣然，朱建阳. 网络“建筑学”[J]. 物理学报. 2003, 32(6): 364-369. [↑](#endnote-ref-17)
18. 吴金闪，狄增如. 从统计物理学看复杂网络研究[J]. 物理学进展, 2004, 24(1): 0018-0046. [↑](#endnote-ref-18)
19. Xiang Li, Guanrong Chen. A local-world evolving network model[J]. Physic A, 2003, (328): 274-286. [↑](#endnote-ref-19)
20. Q H Chen, D H Shi. The modeling of scale-free networks[J]. Physic A, 2004, (335): 240-248. [↑](#endnote-ref-20)
21. 陈禹，宗骁，郝杰等. BA模型的三种扩展[J]. 系统工程学报, 2005, 20(2): 0120-0127. [↑](#endnote-ref-21)
22. 史定华. 网络度分布理论[M]. 北京, 高等教育出版社, 2011. [↑](#endnote-ref-22)
23. M Garey, DS Johnson. A Guide to the Theory of the NP-completeness[M]. A Series of Books in the Mathematical Sciences, 1979 [↑](#endnote-ref-23)
24. RM Karp. Reducibility Among Combinatorial Problems[J]. Journal of Symbolic Logical, 2010, 40(4): 618-619. [↑](#endnote-ref-24)
25. F Gavril. Algorithms for Minimum Coloring, Maximum Clique, Minimum Covering by Cliques and Maximum Independent Set of a Chordal Graph[J]. SIAM Journal on Computing, 1972, 1(2): 180-187. [↑](#endnote-ref-25)
26. G Valiente. Clique, Independent Set, and Vertex Cover[J]. Springer Berlin Heidelberg, 2002: 299-350. [↑](#endnote-ref-26)
27. P V Sanser, D Nehad, E Chlamtac et al. Efficient traversal of mesh edges using adjacency primitives[J]. ACM Trans, 2008, 27(5): 144-152. [↑](#endnote-ref-27)
28. A Gemsa, M Nollenburg, I Rutter. Evaluation of labeling strategies for rotating maps[J]. Journal of Experimental Algorithmics, 2014, 21(1): 235-246. [↑](#endnote-ref-28)
29. T kieritz, D Luxen, P Sanders et al. Distributed time-dependent contraction hierarchies[J]. Springer Heidelberg, 2010, 60(49): 83-93. [↑](#endnote-ref-29)
30. WC Ke, BH Liu, MJ Tsai. Efficient algorithm for Constructing Minimum Size Wireless Sensor Networkks to Fully Cover Critical Square Grids[J]. IEEE Trans. Wireless Comm, 2011, 10(4): 1154-1164. [↑](#endnote-ref-30)
31. J E Anderson, A Chakrabortty. Minimum Cover Algorithm for PMU Placement in Power System Network Under Line Observability Constraints[J]. IEEE Power and Energy Society General Meeting, 2012, 1(2)1-7. [↑](#endnote-ref-31)
32. E Saenz-de-Cabezon, H P Wynn. Measuring the robustness of a network using minimal vertex covers[J]. Mathematics and Computers in Simulation, 2014, 104(2): 82-94. [↑](#endnote-ref-32)
33. B Harsh, A Geetanjli. Harnessing Genetic Algorithm for Vertex Cover Problem[J]. International Journal on Computing, 2012, 4(2): 218-223.（GA参考文献） [↑](#endnote-ref-33)
34. Diogo V.Andrade, Mauricio G.C Resende, Renato F.Werneck. Fast Local Search for the Maximum Independent Set Problem [J]. Springer Science, 2012, 18: 525-547. [↑](#endnote-ref-34)
35. Y. Jin, J K Hao. General Swap-based multiple neighborhood tabu search for the Maximum Independent Set Problem[J]. Engineering Applications of Artificial Intelligence, 2015, 37: 20-33. [↑](#endnote-ref-35)
36. Una Benlic, Jin-Kao Hao. Breakout Local Search for maximum clique problems[J]. Computers&Operations Research, 2013, 10: 192-206. [↑](#endnote-ref-36)
37. Cai Shaowei, Luo Chuan, Sattar et al. NuMVC: An Efficient Local Search Algorithm for Minimum Vertex Cover[J]. Journal of Artificial Intelligence Research, 2013, 46: 687-716. [↑](#endnote-ref-37)
38. K Kotecha, N Gambhava. A Hybrid Genetic Algorithms and the Vertex Cover Problem[R]. In the First India International Conference on Artificial Intelligence, India, 2003. [↑](#endnote-ref-38)
39. J Chen, Y Lin, J Li et al. A rough set methods for the minimum vertex cover problem of graphs[J]. Applied Science and Engineering, 2016, 42(7): 360-367. [↑](#endnote-ref-39)
40. F V Fomin, D Kratsch. Exact Exponential Algorithms[J]. Springer, 2013, 56(3): 80-88. [↑](#endnote-ref-40)
41. Mingyu Xiao, Hiroshi Nagamochi. Exact Algorithms for Maximum Independent Set[J]. Springer Berlin Heidelberg, 2013, 8283: 328-338. [↑](#endnote-ref-41)
42. S Kratsch, F Neumann. Fixed-Parameter Evolutionary Algorithms and the Vertex Cover Problem[J]. Algorithmica, 2013, 65: 754-771. [↑](#endnote-ref-42)
43. Y Yang, X li. Towards a Snowdrift Game Optimization to Vertex Cover of Networks[J]. IEEE Transactions on Cybernetics, 2013, 43(3): 948-956. [↑](#endnote-ref-43)
44. D Goldberg. Genetic algorithms in search[J]. Optimization and Machine Learning. Massachusetts: Addison Wesley, 1989. [↑](#endnote-ref-44)
45. DY Deng, DX Yan, JY Wang. Parallel reduces based on attribute significance[J]. Rough Set and Knowledge Technology, 2010, 64(1): 336-343. [↑](#endnote-ref-45)
46. V Chvatal. A greedy heuristic for the set-covering problem[J]. Math, 1979, 4(3): 233-235. [↑](#endnote-ref-46)
47. J Von Neumann, O Morgenstern. Theory of games and economic behavior[M]. Princetion University Press,1953. [↑](#endnote-ref-47)
48. JF Nash. Equilibrium points in n-person games[J]. Proceedings of national academy of sciences, 1950, 36(1): 48-49. [↑](#endnote-ref-48)
49. 吴枝喜, 荣智海, 王文旭. 复杂网络上的博弈[J]. 力学进展，2008, 38(6): 794-804. [↑](#endnote-ref-49)
50. G Szabo, C Toke. Evolutionary prisoner’s dilemma game on square lattice[J]. Physical Review E, 1998, 58(1): 69-73. [↑](#endnote-ref-50)
51. M Perc, A Szolnoki. Co-evolutionary games A mini review[J]. BioSystems, 2010, 99(3):109-125. [↑](#endnote-ref-51)
52. WX Wang, YC Lai. Network Reconstruction Based on Evolutionary-Game via Compressive Sensing[J]. Physical Review X, 2011, 1(2): (021021-1)-(021021-7). [↑](#endnote-ref-52)
53. G Szabo, G Fath. Evolutionary games on graphs[J]. Physics Reports, 2007, 446(4-6): 97-216. [↑](#endnote-ref-53)
54. MA Nowak, RM Mary. Evolutionary Games and Spatial Chaos[J]. Nature, 1992, 359(6398): 826-829. [↑](#endnote-ref-54)
55. MA Nowak, RM Mary. The spatial dilemma of evolution[J]. International Journal of Bifurcation & Chaos, 1993, 3(1): 35-78. [↑](#endnote-ref-55)
56. C Hauert, M Doebeli. Spatial structure often inhibits the evolution of cooperation in the snowdrift game[J]. Nature, 2004, 428(6983): 643-649. [↑](#endnote-ref-56)
57. B Genest, H Gimbert, A Muscholl. Asynchronous Games over Tree Architectures[J]. International Colloquium on Automata, Languages and Programming, 2013, 275-286. [↑](#endnote-ref-57)
58. HR Lourenco, OC Martin, T Stutzle. Iterated Local Search[J]. Springer US, 2003, 32(3): 320-353. [↑](#endnote-ref-58)
59. A Askarzadeh, LDS Coelho, CE Klein et al. A population-based simulated annealing algorithm for global optimization[J]. IEEE International Conference on System, 2017, 4: 626-633. [↑](#endnote-ref-59)
60. http://cs.hbg.psu.edu/txn131/clique.html [↑](#endnote-ref-60)
61. http://neilsloane.com/doc/graphs.html [↑](#endnote-ref-61)
62. https://www.nlsde.buaa.edu.cn/\_kexu.benchmarks/graph-benchmarks.html [↑](#endnote-ref-62)
63. AL barabasi, R Albert. Emergence of scaling in random networks[J]. Science, 1999, 286(5439): 509-512. [↑](#endnote-ref-63)
64. P Erdos, A Renyi. On random graphs[J]. Publicationes Mathematicae, 1959, 6(290): 290-297. [↑](#endnote-ref-64)
65. DJ Watts, SH Strotagz. Collective dynamics of ‘small-word’ networks[J]. Nature, 1998, 393(6684): 440-442. [↑](#endnote-ref-65)
66. A Lancichinetti, A Fortynato, F Radicchi. Benchmark graphs foe testing community detection algorithms[J]. Physical Review E Statistical Nonlinear & Soft Matter Physics, 2008, 78(2): 046-110. [↑](#endnote-ref-66)
67. JS Wu, ZY Chang, L Yuan. A Memetic algorithm foe Resource Allocation Problem based on Node-Weighted Graphs[J]. IEEE Computational Intelligence Magazine, 2014,

    9(2): 58-69. [↑](#endnote-ref-67)
68. S Khui, T Back. An evolutionary heuristic for the minimum vertex cover problem[J]. Genetic Algorithms within the Framework of Evolutionary Computation, 1994, 86-90. [↑](#endnote-ref-68)
69. CH Papadimitriou, K Steiglitz, “Algorithms and Complexity” in Combinatorial Optimization[J], New York: Dover, 1998, 15(6): 360-363. [↑](#endnote-ref-69)