清 华 大 学

综 合 论 文 训 练

题目：动态社交网络图数据生成研究

系　　别：软件学院

专　　业：软件工程

姓　　名：冯　昊

指导教师：王朝坤副教授

2020 年 5 月 22 日

关于学位论文使用授权的说明

本人完全了解清华大学有关保留、使用学位论文的规定，即：学校有权保留学位论文的复印件，允许该论文被查阅和借阅；学校可以公布该论文的全部或部分内容，可以采用影印、缩印或其他复制手段保存该论文。

(涉密的学位论文在解密后应遵守此规定)

签　名：_____ 导师签名：_____ 日　期：_____

中文摘要

本论文对动态社交网络图的生成进行了一系列研究，将图结构的时序变化抽象化，使用多种事件进行建模。在此基础上，文中设计实现了一个高效的算法，在给定分布特征、社区结构、事件要求之后可以进行动态图的生成操作，所得结果可以以多种格式进行存储，兼容常用的一些图数据处理与可视化库。文中还设计实现了一个完整的网页端动态图生成管理系统，包括前端后端两部分，将动态图生成工具集成在后端服务中，可以在网页上进行动态图的配置工作，并在网页端完成图生成与结果的可视化任务。

本文的创新点主要有：

- 使用“事件”这一概念进行动态图中时序变化的定义；
- 将一些典型的事件进行建模并抽象化为图结构上的变化；
- 设计并完成了一个完整的可配置动态图生成工具；
- 设计并完成了一个便于交互操作的动态图生成管理系统，可以便捷地完成动态图的配置、生成、管理与可视化。

关键词：动态图；事件；生成；管理系统

ABSTRACT

In this paper, I conduct a series of studies on the generation of dynamic social graphs, abstracting the temporal changes of the structure of dynamic graphs and modeling them with a variety of events.On this basis, an efficient algorithm is designed and implemented to generate dynamic graphs using given distribution characteristics, community structure, and event requirements. The results can be stored in a variety of formats, compatible with some commonly used graph data processing and visualization libraries. A complete web-side dynamic graph generation management system is implemented, including front-end and back-end. It can configure the dynamic graph on the web page, and complete the visualization task of graph generation and result on the web page.

The main innovations of this paper are:

- I use the concept of ”events” to define temporal changes in dynamic graphs.
- I model some typical events and abstract them into changes in the graph structure.
- A complete configurable dynamic graph generator is designed and completed.
- A dynamic diagram generation management system is designed and completed, which is easy for interactive operation. It is easy to configure, generate, manage and visualize dynamic diagrams.

Keywords: dynamic graph; events; generation; management system

目 录

主要符号对照表

静态图	我们日常提到的图论中的图，由节点和边组成的图结构
动态图	动态社交网络图，指社交网络图的时间序列
$G = (V, E)$	由节点 V 和边 E 组成的图
V	图中的节点
E	图中的边
C	图中的社区
$P\left(E_{A,B}\right)$	图中的节点 A 和 B 有边相连的概率
rho	表征社区紧密度的参数，表示社区间形成边和社区内形成边的概率之比
$dist_{out}$	出度分布
$dist_{in}$	入度分布
d_{min}	给定的最小度数
d_{max}	给定的最大度数
$outd_i$	节点 i 的出度
ind_i	节点 i 的入度
ind_{edge_j}	边 j 对应目标节点的入度

第 1 章　引言

1.1　研究背景

1.1.1　图的概念与应用

图（Graph）是我们经常会见到的一种数据结构，它由若干给定的节点（Node、Vertex）和节点之间的边（Edge）组成。其中节点可以表示真实世界中的一些事物，而边则可以描述它们之间的一些联系（Relationship）。图的形式化定义为：

$$G = (V, E) \tag{1-1}$$

其中 V 表示节点的集合，而 $E \subseteq \{(x, y)|(x, y) \in V^2, x \neq y\}$ 表示图中的边（即节点二元组）的集合。

本质上来说，图是一种更优雅地表示多对多关系的一种数据结构，用图的方式进行描述，可以更直观、更有效地展示事物之间的关系，并且进行抽象与建模之后可以用图上的很多高效的算法来解决实际生活中的问题。

图的历史，可以追溯到欧拉的柯尼斯堡七桥问题[1]，他将河岸和岛抽象成节点，桥抽象成边，利用度数的奇偶性巧妙地解决了这一问题，展示了图论的优越性。

我们的生活中很多的关系、事件都可以用图进行建模。如：

- 在社会科学的研究过程中，每一个人都可以看作一个节点，而不同人之间的朋友关系、师生关系、都可以看作是节点之间的边。比如在微博上，不同的用户可以看作是不同节点，他们之间的好友关系可以看作节点之间的边；
- 在计算机科学中，通信网络、数据流等许多问题都可以被建模成图论中的问题，一台设备、一个网页都可以被定义为一个节点，从而应用成熟的图论算法使问题得以解决。

那么，都有哪些问题是用图进行分析得到很好的成果的呢？我们耳熟能详的一笔画问题、最短路径问题等都是可以用图论进行建模、研究的问题，这些问题都有了比较优雅的解决方法。还有四色问题、最大流最小割问题、哈密顿回路问题、社区划分问题等都是在图论的解决范围内的。

1.1.2 合成图的必要性

现在随着互联网的发展，人与人之间的边界被打破，人们之间沟通交流的关系越来越紧密，传递的数据量也越来越大。同时，物联网的蓬勃发展也带来了更多新的需求，随着万物互联时代的到来，越来越多的图数据在生活中生成、刻画着我们愈发丰富的生活。随之而来的是一系列有关信息传播、节点重要性计算的问题，许多学者对此进行了深入的研究。如：

- 在谣言的传播过程中[2][3]、在文章或广告的转发宣传过程中，究竟是哪一些节点起了更大的作用？
- 在互联网上究竟哪一些网站更加权威、更应该在用户搜索结果中被赋予更高的权重？
- 根据用户对于信息的访问记录，如何能够更好地匹配用户的需求，将更合适的消息推广他/她？
- 如何通过用户之间的好友关系、用户发布的微博等信息来判断出一个用户可能认识的人？
- 如何通过银行账户之间的转账关系，分析得到涉嫌洗钱的可疑账户？

解决这些问题的算法无论是基于传统方法还是神经网络的方法，都是要使用大量数据进行分析、验证和评估的。最好的解决方法是使用真实世界的数据集进行分析和研究，但是只依赖真实数据的话会有许多问题存在：

- 数据难以收集，或收集到的数据会人为加入噪声：
  - 由于一些数据的敏感性，比如银行的转账数据、推特账户的好友关系，可能不会公开或者在公开前会进行一系列混淆、脱敏的操作；
  - 由于数据属于公司内部资产，对数据的分析可以带来潜在的经济效益，所以很多公司不会选择公开自己内部的数据，并且会对爬取数据的行为进行各种限制。如淘宝的页面就采取了多种反爬虫措施以免自己的数据泄漏。
- 数据已经固定，通过修改数据的方式得到的新数据不能保证与原数据同样的特征：
  - 公开的真实数据有限，特别是超大规模的社交数据，在这种情况下如果想要找到类似的数据进行对照就不是那么容易。比如，在已经得到推特的数据之后，由于社交网络领域体量较大的公司只有几个，很难找到类似的数据与之进行对比。

– 若对原始数据进行修改（删除或者扩展），使得其中一些特征发生变化，可以解决数据来源少的问题，但是修改的过程本身可能会破坏数据内在的结构，结果可能并不具有参考意义。

因此，合成数据集对于图论相关算法的研究具有很重要的意义，合成数据集的广泛采用可以一定程度上解决数据来源不足的问题。

1.2 课题意义

目前，已经有许多学者对合成数据集的方法进行了探究，其中包括用基于概率模型的传统方法与基于神经网络的方法，两者都各有特色，可以基于已有的真实世界中的图特征先验知识得到更真实的图结构。

但是目前已经有的工作基本上都是基于静态图进行生成的，也就是生成的结果只是某个时刻的图切片数据，并不具有随时间变化的特征。与之相对应的一个概念就是动态图（动态社交网络图，Dynamic network，即随时间有所变化的社交网络图[4]），动态图相当于是若干个图的时间序列，将一个图的演化过程用动态的方式表现出来。与静态图相比，动态图相关的研究并不是特别充分。

那么，为什么要考虑图的时序变化呢？因为真实世界中的图往往都是动态图，是随着时间流逝逐渐建立起来的。随时间的演化信息中蕴涵着某些节点的特征信息以及这些特征的变化，并且可以反映某些事件的影响：

- 有影响力节点的形成与衰退[5]：在微博的用户关系图中，可以通过某个节点度数增长速度的变化来判断某个大 V 的形成与过气的过程；
- 热点事件的发生：在疫情期间，一些与医疗相关的用户与话题可能会收到更大的关注度，相比平常会更加活跃，具体表现在图结构中可能就是度数增长速度的阶段性提升；
- 周期性事件的发生：冰雕手艺人可能在冬天的关注度会更多，冰激淋在夏天往往受到更多的关注。

如果将动态特性加入到图的生成过程中，让图可以反映类似真实网络中的时序变化，就可以更好地体现社交网络图的演化特征，并且可以让用户更自由地定义符合自己要求的图结构。用控制变量的方法，将某种时序演化结果的图与不同演化条件下的模拟结果相对比，可以更好地为相关算法服务。特别是对于一些动态图的算法而言，可以更自由地生成需要的数据意味着这些算法可以更好地进行设计与验证。

1.3 已有研究情况

在对已有算法的调研与分析基础上，本节对几种比较典型、有效的合成图数据方法进行介绍。

1.3.1 基于传统方法生成

下面两个方法可以基于概率模型进行生成：

- 一个名为 S3G2[6] 的研究将节点之间边的形成概率与节点本身的属性建立起了联系，用基于标签相似度的概率算法来进行边的生成，得到了较好的结果。这里提到的属性是指节点本身的一些信息，如人物的年龄、所在城市等，作者观察到相同属性标签的节点之间产生关联的概率相对更大，例如相同时间在同一所大学上学的人之间有好友关系的概率更大。在 S3G2 的工作中，使用了一个相似度函数将节点转换为一个数字，从而通过排序函数得到不同节点的位置，位置相近的节点有相对较高的相似程度。根据相似程度不同，会给节点分配不同的邻接概率，在此基础上通过随机算法来进行边的生成操作。
- 为了能够更高效地进行边的生成操作，同时能够兼顾生成结果的度数分布特征与社区特征，FastSGG[7] 这篇工作中提出了一个通用性的生成算法，并且给出了一个高效的度数分布生成工具模型，通过用户指定的生成图的一些特征，来进行针对性的生成。这个算法只需要提供概率密度函数就可以进行计算，对于累计分布函数较难求出的情况也可以很好地解决。而社区结构体现在社区内和社区间生成边的概率不同，通过一个参数可以控制社区结构的明显与否，从而能够得到具有社区结构结果的图结构。这篇工作中使用离散化的方法，将求解累计分布函数逆函数的过程转化为分位点确定的问题，极大地提高了效率，使得 FastSGG 这篇工作可以胜任超大规模图的生成任务。

1.3.2 基于神经网络方法生成

也有很多学者研究用神经网络的方法进行图的生成，其中包括以下几种比较典型的方法：

- 使用 GAN 或者 VAE 进行图的邻接矩阵特征的学习，将图的生成问题转化成为一个邻接矩阵的生成问题，而邻接矩阵可以变换为一个大小为 n^2 的向量，从而可以套用生成向量的相关方案。但是这样生成的结果大小固定为

n^2，比较难进行扩展。同时，由于一个图的邻接矩阵表示与节点排列有关，造成图的向量表示的不唯一性，因此可能需要对所有可能的节点排列都进行分析，或者在所有排列中指定一个具有代表性的排列，这样的操作时间复杂度较高。

- 使用图神经网络的方法，进行节点的向量嵌入（Node Embedding），得到每个节点的向量表示后用这些向量来计算节点之间的边生成概率，从而进行后续的生成操作。但是这样的做法也有一定的缺陷，因为节点的向量嵌入是基于某些给定的图计算出来的，因此这个方法只对给定的节点集合有效，并且只能够在给定一个示例图的基础上进行生成，难以在给定的多个图基础上进行训练。
- 使用 RNN 的方法进行生成，将之前的节点和边的相关信息放在隐藏状态中，从而达到能够基于之前的信息指导后续生成的目的。在 GraphRNN[8] 的工作中，就使用了这种方法，将图生成的过程分解为一系列节点和边的生成，用递归的方式将图建模成为新节点和边的添加序列，以获取邻接关系的概率，从而得到生成图。GraphRNN 可以学习生成与目标集的结构特征相匹配的各种图，整体的方法可以看作是一个层次模型，存在图和边这两个层次。
  - 图层次的 RNN 可以维护图的状态并且产生新的节点。因为节点没有属性信息，所以生成新的节点并不需要特别复杂的操作；
  - 而在边层次的 RNN 上，负责进行新产生节点上的边生成操作。

1.3.3 动态图的生成

据笔者所知，目前并没有能够模拟真实世界某些时序变化特征的高效生成动态图框架。但基本上现有的静态图生成方法都可以进行或者经过修改后进行图生成过程的分步化，也就是将一整个图的生成过程分为几个子过程，每一个过程中只生成部分的节点或者部分的边。本质上来说这样只是将一个图的生成分为几步，生成的图遵循同样的分布，不能体现总体分布特征的变化与一些节点的变化。

目前有一些基于行为定义的动态图生成方法，如：

- Bob De Caux 等人在 2013 年发表的一篇论文[9] 就是基于代理交互行为的。具体来说，这个方法将网络的生成过程与粒子随机运动相互碰撞的过程相类比，让每一个节点都有对应的移动行为，随机选择其方向与距离以在这一过程中随机遇到其余节点生成对应的边。

由于动态性也可以用于进行其他类型图的研究，也有一些特定领域上进行图动态生成的研究：

- 在调度算法（如列车时刻表的安排）的研究中，每个时刻的调度都要基于上一时刻的状态，因此可以用动态的方法逐步构建，如 Fischer 等人的研究[10]就是在考虑图上最短路径的基础上进行每一时刻的生成任务的；
- 知识图谱构建过程中，新的知识在引入时往往与已有内容具有关联，因此知识体系图构建时可以采用动态生成的形式。如图表信息提取、构建知识图的过程中就可以用逐步生成的方式[11]，从而建立起来不同概念之间的关联关系。

上述研究中涉及到的图并非社交网络图，其中的生成过程也是人为设计、根据一定规则确定性进行的，与本课题中涉及到的受一些事件影响、由人们的行为生成的动态社交网络图有很大的区别。

1.4 本课题要解决的问题

目前已有的图生成工具大部分关注于静态图的生成而非动态图的生成，无法模拟真实世界中一些事件导致的图结构动态性变化。同时，一些已有的用物理模拟的方法进行动态图生成的工作并不高效。

本课题旨在对一些动态图行为建模并且用高效的方法（基于给定分布的邻接关系随机生成）进行动态图的配置与生成。同时，为了能够便于用户使用，本课题中进行了网页形式配置、管理的实现，从而避免手写 JSON 内容的繁杂与困难。

本课题主要包含以下几个方面：

- 分析真实世界动态社交网络图的特征，定义几种典型的动态社交网络图上的事件，模拟真实社交网络图中的动态结构变化；
- 基于 FastSGG 这篇工作实现的一个可配置静态图生成工具，进行时间序列相关配置的扩展，实现一个动态图的配置与生成工具；
- 完成了一个完整的网页端动态图生成管理系统，包括前端后端两部分，可以用网页的形式进行动态图的配置、生成与结果的可视化。

第 2 章　动态图特性分析与生成器配置方式研究

本章前两节将从图的节点和边、图的结构特征入手，对这两方面的动态性进行分析，进而提出用“事件”进行图结构特征动态性表现的方法。在第三节中，将提出一个包含动态特征的图生成器配置方式。

本章中提到的“动态图”是指具有时间序列结构的社交网络图，是普通图在时间轴上的扩展，可以展示图的动态变化过程。因此，动态社交网络图相比于静态的社交网络图，蕴含着更多的信息，能够更好地体现社交网络中的一些结构特征、演化特点，能够让数据更好地为人们所用。

例如：

- 在电商平台的推荐算法中，可以用一个二分图来表示用户和商品之间的关注、购买关系，从而利用这些数据进行推荐[12]。由于用户的需求往往具有时效性，可能在某一段时间内关注一类商品（如在新房装修时会搜索有关家具、电器的内容）但是这段时间之后就不会有类似的需求。如果在这样的数据图中加入时间信息，就可以更好地利用在某一个时间窗口中用户与商品之间、商品与商品之间的相关性，做出更好的推荐，如一篇对推荐系统进行研究的文章[13] 中就考虑了消费在时间上的重叠性。
- 在反洗钱的操作中，可以将不同账户之间的转账关系建模成一个图结构，由于洗钱操作需要防侦查操作，往往是在很短的时间内进行了大笔金钱交易。在银行账户转账记录的图中加入时间信息可以更好地进行此类的分析操作[14]。

和静态图相比，动态图最重要的特征就是其中的动态性。对动态图进行分析，首先需要对图的动态性进行建模，用一个更为具体的方式对这种性质进行描述。因此本章接下来将从动态性的分析、建模等角度进行介绍。

2.1　节点与边上的动态性

所谓动态性，就是在时间序列上的变化。要探究图上动态性的体现方式，就需要先了解一个静态图的基本组成要件：

- 节点：某个实体/事物的抽象化表示，可以具有一些属性值，属性可以用

key-value 对的字典形式表示；

- 边：节点之间的连接方式，可以有多个属性值。边可以按照以下几种不同的方式进行分类：
  - 有向边与无向边
  - 单边与多重边
  - 带权重的边与无权重的边

在图上还可以有路径、环、社区等结构的定义，这些都是由节点、边组成的更高层次的结构。从最底层、最基本的角度来看，图的基本单元就是节点与边。

要给图赋予动态性，本质上来说就是对图结构的修改。而修改一词具体的含义就包括增加、删除、修改三个方面，具体来看也就是包含以下几种：

- 节点的增加、删除
- 边的增加、删除
- 节点和边属性的变化（增加、删除、修改）

其中最关键的在于节点和边的增加部分，对于一部分图而言可能不会有节点与边的删除，但这种删除操作对于某些特定类型的图而言也是经常见到的一种操作。

例如电商平台中用户收藏商品的关系可以看成一个图，上述几种动态性在这个例子中具体的体现形式如下：

- 节点的增加——新用户的进入、新商品的上线
- 节点的删除——用户的注销、商品的下线（商家主动下线或遭投诉被处理下线）
- 边的增加——用户新收藏一个商品
- 边的删除——用户对某个商品取消收藏、商品下线导致对该商品的所有收藏关系失效
- 节点和边属性的变化——用户修改个人信息、商品修改详情信息、用户修改商品收藏所在的收藏夹

以上定义的确可以表述一个图的动态性包含的内容，以节点和边为粒度进行的动态性分析对于基于行为模拟的生成方法比较有效，可以考虑每一种行为对应的概率进行模拟生成。但是这样的方法具有一定的局限性，需要大量的数据对每个节点的行为进行建模，可能会造成效率方面的不足。

2.2 图的结构特征与动态性

上一节从微观的层面进行了图动态性的讨论，完备地说明了图的动态性对于节点和边这些基本结构的影响。但仅止于此并不足以指导生成过程，正如1.4节所说，本文旨在将动态特性体现于基于概率的邻接关系生成过程中，因此需要对动态图中更高层次、更抽象化的特征进行抽取。

2.2.1 图的结构特征

对于真实社交网络图而言，目前已经有许多对其特征进行抽取、总结的工作，学术界也已经有许多公认的社交网络图特征。如：无尺度网络[15]、小世界原理[16]、社区结构[17] 等，这些特征的解释如下：

- 无尺度网络（Scale-Free Network）：大部分节点的度数很小，只有小部分节点拥有极大的度数；这一特点在很多情况下可以用幂律分布进行建模：

$$P(k) \sim k^{-\gamma} \tag{2-1}$$

  这里的 k 表示节点度数，一般情况下 $2 < \gamma < 3$。目前有已经发现了满足无尺度网络特性的很多真实网络，如银行间支付网络[18]、语义网络[19] 等。
- 小世界原理：真实社交网络中，往往两个节点之间可以通过少数节点联通。这一原理曾经用六度分离理论（Six Degrees of Separation）来表示，也就是说“任意两个人都可以通过最多五个人相认识”。这一概念充分地说明了现实世界社交网络结构的联通性，而一个纯随机的网络是不具有这种特征的。在 1961 年就已经有学者对此进行了实证研究[20]，哥伦比亚大学也曾经使用电子邮件记录对这一原理进行验证[21]。
- 社区结构：在真实社交网络中，某些节点之间的连接会相对更加紧密，如在同一个小区、同一所学校、具有相同兴趣爱好的人往往有更多的交互。基于此，整个社交网络可以被划分为几个部分，每一部分就是一个社区。社区本质上来说就是一些节点组成的集合（可能会重叠[22]），其内部的联系比较紧密，社区间的联系更加疏松。具体到图的结构上，也就是同一社区内节点之间连接概率更高、不同社区节点之间连接概率更低。例如：引文网络会按照研究主题形成社区[23]。

利用上述特征而非直接着眼于每一个节点、边的行为模拟，可以更高效地进行图结构的生成。

2.2.2 基于结构的图生成

图的生成过程其实就是节点和边的生成，总体上可以划分为结构和属性两部分。本节中主要着眼于图的结构信息。

基于之前对于节点度数分布、社区特征的研究，在忽略属性信息的情况下，可以提出一个简单的有向图生成模型，其中包括如下信息[7]：

- 节点的数量
- 边的数量
- 边的选择（起点和目标节点分别是哪个已有节点），其中的影响因素包括：
  - 每个节点度数分布的拟合函数（包括入度分布与出度分布）
  - 由于社区结构导致的节点之间连接概率不同

其中，社区结构对边生成的影响简单地可以用一个参数 ρ 表示，其含义为社区间生成边概率与社区内部生成边的概率的比值，形式化定义如公式2-2所示。

$$[H]\rho = \frac{P\left(E_{A,B} \middle| \exists C_i, C_j, i \neq j, A \in C_i, B \in C_j\right)}{P\left(E_{A,B} \middle| \exists C_i, A \in C_i, B \in C_i\right)} \tag{2-2}$$

其中 $P\left(E_{A,B}\right)$ 表示节点 A 和 B 之间有边相连的概率，C_i, C_j 表示两个不重叠的社区。

将上述生成模型中的参数全部进行定义后，即可进行生成工作。

模型中节点数量、边数量的定义较为简单，下面着重说明模型中节点度数分布、社区分布特征的定义。

基于前人的研究，Power-Law 分布（公式2-1）可以用来进行节点度数的拟合。同时每一个社区的构成、社区对边生成影响程度这些社区相关信息也会影响边的生成过程。

除了度数的分布特征的约束，每个节点对应的度数、每个节点度数的相对次序也需要进行约束。这是因为度数分布特征只是能基于概率给出度数为 m 的节点个数，但并没有指定具体度数为 m 的哪些节点是哪些。为了解决这个问题，可以将所有节点进行排序，进而将这些节点分别对应到所有的度数之中。

总体的入度分布与出度分布不一定一致，同一个节点的入度与出度也并不一定相同，甚至每个节点的入度与出度不一定相关。如微博上的网红可能有几百万的关注量，但是他们关注的其他用户个数可能很少；反过来，关注几百万个其他用户的人也并不多。因此，需要将入度分布与出度分布分离，分别进行定义，并

且需要分别分析入度分布与出度分布的节点排序特征。

基于对入度分布、出度分布、社区结构的约束，可以生成一个较符合真实网络分布特征的合成图。

2.2.3 结构上的动态特征

上一节中已经抽取出了图结构中的一些特征信息，可以在生成过程中使用。这些特征在图的动态性中会有以下几方面的体现：

- 节点的数量变化：新增与删除
- 边的数量变化：新增与删除
- 边生成过程的变化[①]：
  - 分布函数本身的变化（分布函数参数变化、分布函数类型的变化）
  - 分布中每个节点地位的变化[②]
  - 社区分布的变化（社区本身结构的变化、社区参数 ρ 的变化）

从真实世界社交网络图的研究来看，分布函数本身的类型发生变化并不常见。幂律分布相对而言能更好拟合真实世界的情况，但是幂律分布中参数 λ 在不同网络中有所区别。

另外需要注意的是，上述这些变化并不是在所有的社交网络中都会出现。例如，引文网络中一般情况下不会出现边的删除操作。

基于上面的分析，可以将上述各种变化的形式提取出来作为生成时的依据。本节将这些变化的过程抽象为事件，每一种事件都可以描述某一种特定的图结构随时间的变化，如表2.1所示。

事件的定义中需要指定影响的目标、事件发生的时刻。其中一些事件可以具有周期性，如：若一位冰雕手艺人在每个冬季都会发布更多的冰雕相关作品，则会导致其在冬季的关注量增长较快，这就可以认为是季节变化带来的周期性影响。因此，可以将上述各种事件发生的时刻用周期的形式进行定义。

2.3 包含动态特征的图生成工具配置

前文中讨论了如何利用更高层、抽象的结构信息对图的生成过程进行分析，提出了一个简单的有向图生成模型，并且提出了用事件的方法进行动态性定义的

① 因为生成模型中是基于度数分布和社区分布，因此需要关注与此相关的变化

② 节点地位指在所有节点中此节点度数的相对大小，如：原来是 A 节点度数最大，现在变成 B 节点度数最大

表 2.1　各种事件的定义、示例与其所对应的结构动态性

事件	示例（微博中的用户关注关系）	结构动态性
节点增长	微博的 APP 在推广后一段时间内有大量用户注册	节点增加
节点删除	用户注销账户或被封号	节点删除
边的删除	用户取消关注别人	边删除
边的生成	用户在浏览过程中关注喜爱的博主	边增加
突发事件 *	新冠肺炎导致医生的关注度增加，疫情之后可能关注度降低	分布中节点地位临时变化
节点重要度变化 **	某大 V 由于作品受欢迎而逐渐成为网红	分布中节点地位变化
社区参数变化 ***	一些有相同爱好（比如都喜欢游戏）的用户互相交流，成为好友，从而自发形成了一个小圈子	社区参数 ρ 的变化

*：特指突发事件中对图中节点在所有节点中地位排序的临时性变化，这会导致其度数、度数增长速度受到影响

**：特指某些节点在所有节点中地位排序逐渐变化

***：社区参数指公式2-2中的 ρ

思路。本节将对2.2.2节提出的模型进行扩展，加入动态特征，并且用形式化的方式定义其配置方式。

包含动态特征的图生成工具配置需要包括节点、边、社区、事件四大模块，其中动态性主要在事件模块中体现，具体见表2.2。

表 2.2　包含动态特征的图生成工具配置

模块	名称	说明
节点配置	节点标签	此类节点的唯一标识符
	节点个数	此类节点在起始时刻的个数，后续节点的增加与删除可以在事件中定义
	节点属性	属性信息配置时须指明属性名、属性类型与取值范围/生成方式
边配置	边标签	此类边的唯一标志符
	边属性	属性信息配置时须指明属性名、属性类型与取值范围/生成方式
	出度分布	包含： - 分布类型：幂律分布、对数正态分布、均匀分布等 * - 分布参数：分布函数中需要的参数，如幂律分布中的 λ - 分布的度数范围：最小度数与最大度数
	入度分布	形式上同出度分布，但入度分布只影响每个节点被选择的相对概率大小，不能严格约束节点入度的范围 **
社区配置	边标签	边的唯一标志符 ***
	比例	各个社区的节点数量比例
	参数	表征社区结构紧密度的参数 ρ
事件配置	事件类型	表2.1中定义的事件之一 ****
	相应标签	受事件影响的节点标签/边标签
	参数	事件中需要的参数信息，如：影响范围、影响程度等

*：为了能够增加用户配置过程的灵活性，在此不约束必须使用幂律分布

**：给定的入度分布会在3.1.2节中的算法中计算各个节点被选为边的目标节点的概率，但度数范围不能保证（如：共有 100 个节点、5000 条边，但要求入度最大值为 3，则此要求不会被满足）

***：由于每种关系（每类边）对应的社区特征可能不同，在此需要说明这个社区特征是在哪类边上的。如：微博上相互关注的人不一定有对方的手机号，因此微博组成的社区与通话记录组成的社区并不相同

****：边的生成无需在事件中进行定义

第 3 章　可配置动态图生成工具

此章中将介绍可配置动态图生成工具搭建的相关内容。在第3.1节与第3.1.3节中将进行生成算法的详细介绍，在第3.3节中将介绍工具的整体架构与存储方式。

此生成工具的搭建基于第2.1节与第2.2节中对图动态性的分析、第2.3节展示的图生成工具配置，用户定义的配置信息使用 JSON 格式传入生成工具中。此工具使用 Python 实现，其中使用了 numpy 库进行相关的生成操作。

3.1　静态图基本生成算法

静态图的生成是动态图生成的基础。在给定节点数量、边的分布特征（入度分布与出度分布）之后，就可以进行静态图的生成操作了。

生成算法包括以下两部分：节点度数确定与边的目标节点确定[7]，分别利用了出度分布与入度分布的信息。在确定每一个节点的出度并且确定每一条边的目标节点后，即可得到一个完整的图。

3.1.1　节点度数确定算法

节点度数确定相对比较简单。

在给定出度分布（包含相关参数的概率密度函数 $dist_{out}$）与最小度数 d_{min}、最大度数 d_{max} 之后，可以用公式3-1计算得到每一个可能的度数对应的概率。

$$P\left(outd_i = m\right) = \frac{dist_{out}(m)}{\sum_{j=d_{min}}^{d_{max}} dist_{out}(j)} \tag{3-1}$$

接下来，即可使用得到的概率信息为每个节点随机分配出度。在 numpy 中提供了一个随机函数 np.random.choice 可以按照给定的概率随机取值，只需将可用度数列表与计算出的概率信息传入即可随机生成。

用公式3-1确定每个度数对应概率、选择每个节点的出度的具体过程如算法1所示，其中包含每一个度数对应概率的计算、归一化、度数选择几个部分。

算法 1 确定每个节点出度的算法

输入: 出度分布的概率密度函数 $dist_{out}$，出度分布最小度数 d_{min}，出度分布最大度数 d_{max}

输出: 某个节点的出度

function DegreeProb($dist_{out}, d_{min}, d_{max}$)
    $P \leftarrow []$
    $sum \leftarrow 0$
    for each $i \in [d_{min}, d_{max}]$ **do**
        $P[i - d_{min}] \leftarrow dist_{out}(i)$
        $sum \leftarrow sum + P[i - d_{min}]$
    end for
    for each $i \in [d_{min}, d_{max}]$ **do**
        $P[i - d_{min}] \leftarrow P[i - d_{min}]/sum$
    end for
    return P // 每一个度数对应的概率 P（列表形式，第一个元素对应于 d_{min}）
end function
$P \leftarrow$ DegreeProb($dist_{out}, d_{min}, d_{max}$)
function GetOutDegree(d_{min}, d_{max})
    return $choice([d_{min}, d_{max}], P)$
end function

3.1.2 边的目标节点确定算法

上一节中可以通过得到的概率信息用随机方法给每一个节点分配其出度，也就是确定了以该节点为起点的边的数量。在此基础上，需要确定每一条边对应的目标节点。

先看一个例子，通过例3.1可以直观感受到目标节点确定的流程。

例 3.1： 参数为 1.12 的 power-law 分布，节点总数 100，度数范围 $[1, 10]$

将所有节点按照入度从小到大排序，只需选择最后的节点排序序号即可进行目标节点的选择。

对度数为 i 的节点进行分析，存在归一化参数 α, β 使得：

- 入度为 i 的节点占所有节点的比例为：$\alpha \times i^{-1.12}$
- 目标节点的入度为 i 的边占所有边的比例为：$\beta \times i \times \alpha \times i^{-1.12}$

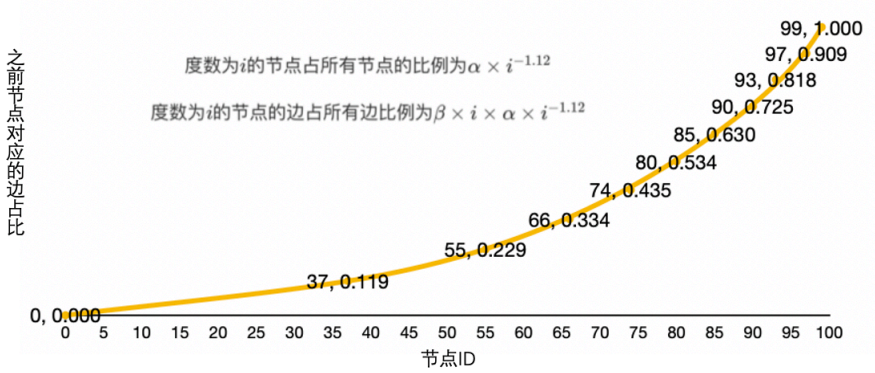

图 3.1　示例-一个 Power-Law 分布节点选择过程 1

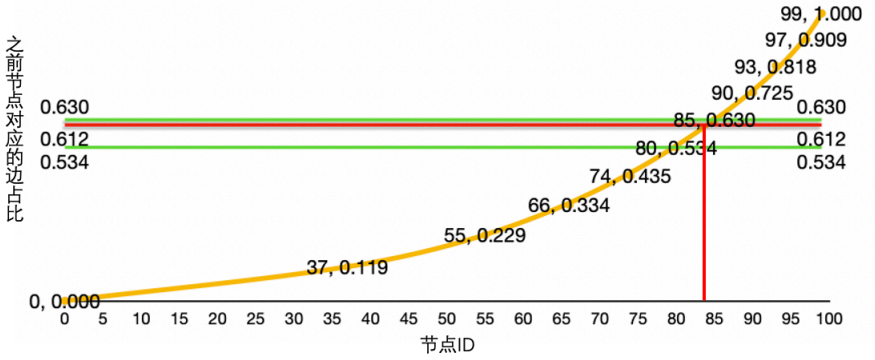

图 3.2　示例-一个 Power-Law 分布节点选择过程 2

如图3.1所示，图中标出的节点横纵坐标其实是累积量，如 $55, 0.229$ 点表示入度不超过 2 的节点一共有 55 个，并且以这些节点为目标节点的边占比为 22.9%：

- 横坐标：入度不超过 m 的节点个数：$100 \times \sum_{i=1}^{i<=m} \alpha \times i^{-1.12}$
- 纵坐标：目标节点入度不超过 m 的边占比：$\sum_{i=1}^{i<=m} \beta \times i \times \alpha \times i^{-1.12}$

由于图中纵坐标表示的是边的累计分布。因此在选择目标节点时，可以在 $[0, 1)$ 区间内得到一个随机数 r，找到纵坐标为 r 对应点的横坐标，即为需要的节点序号。这一过程可以分为两步：确认入度范围、在范围中取值找到目标节点序号。

如图3.2所示，当随机数选择为 0.612 时，确定横坐标的过程可以为：

- 确认 0.612 对应的点所在区间：从图3.2上可以看到 $0.534 < 0.612 < 0.630$，两个定位点分别是 $80, 0.534$ 和 $85, 0.630$，所以应当在入度为 6 的节点中选择
- 目标节点序号取为：$80 + \frac{0.612-0.534}{0.630-0.534} \times (85 - 80) \approx 84$

通过对例3.1的讨论，可以得到目标节点选择算法的一个直观理解。接下来，将对例子中涉及到的公式进行重新定义，并且给出一个完整的选择算法。

首先为所有节点分配一个重要性排序，排序在前的节点重要性较低即入度较小。

用公式3-2即可确定每一个入度的节点在所有节点中占的比例，进而用累积的方法即可得到在节点排序中的区间。

$$P\left(ind_i = m\right) = \frac{dist_{in}(m)}{\sum_{j=d_{min}}^{d_{max}} dist_{in}(j)} \tag{3-2}$$

确定一条边的目标节点时，需要按照公式3-3计算目标节点的入度为 m 的概率。

$$P\left(ind_{edge_i} = m\right) = \frac{dist_{in}(m) \times m}{\sum_{j=d_{min}}^{d_{max}} dist_{in}(j) \times j} \tag{3-3}$$

这里需要特别注意，在公式3-2和公式3-3中，分别计算的是**所有节点中入度为 i 的概率**和**所有边中目标节点中入度为 i 的概率**，因此其表示形式不同。

基于前述讨论，边的目标节点选择过程包含以下几步：

1. 确定每个节点的排序信息，可以使用一个随机打乱函数实现；
2. 依据公式3-2，按照排序信息给确定每个度数对应的节点范围，排序在前的节点入度较低；
3. 依据公式3-3，计算该边选择目标节点度数为 i 的概率 P；
4. 按照得到的概率 P，随机选择每一条边目标节点对应的入度；
5. 得到入度后，在这个入度对应的节点范围中进行随机选择。

在例3.2中给出了一个计算公式3-2和公式3-3两个概率的示例。

例 3.2： 两个概率的计算

条件：节点个数为 2，入度范围为 $[1,2]$，对应的概率密度函数为：$P(ind_i = 1) = 0.4, P(ind_i = 2) = 0.6$，节点排序为 $[1,0]$（即：下标为 0 的节点代表 2 号节点）

计算过程：

- 所有节点中入度为 1 的概率为 0.4，入度为 2 的概率为 0.6
- 节点 $[0, 0.8)$ 对应的入度分配为 1，节点 $[0.8, 2]$ 的入度分配为 2[①]
- 选择的目标节点入度为 1 的节点的概率为 $\frac{0.4\times1}{0.4\times1+0.6\times2} = 0.25$
- 选择的目标节点下标为 0(2 号) 的概率为 $0.25 + (1 - 0.25) \times \frac{1-0.8}{2-0.8} = 0.375$

进行目标节点选择的完整算法如算法2所示。

算法 2 确定一条边的目标节点

输入: 入度分布的概率密度函数 $dist_{in}$，入度分布最小度数 d_{min}，入度分布最大度数 d_{max}，节点排序数组 $Nodes$

输出: 某条边的目标节点

```
function EdgeDegreeProb(dist_in, d_min, d_max)
    P ← []
    sum ← 0
    for each i ∈ [d_min, d_max] do
        P[i − d_min] ← dist_in(i) × i
        sum ← sum + P[i − d_min]
    end for
    for each i ∈ [d_min, d_max] do
        P[i − d_min] ← P[i − d_min]/sum
    end for
    return P // 每一个度数对应的概率 P（列表形式，第一个元素对应于 d_min）
end function
P ← DegreeProb(dist_in, d_min, d_max) // 可以直接调用确定出度概率的函数
P_edge ← EdgeDegreeProb(dist_in, d_min, d_max)
function GetTargetNode(d_min, d_max)
    degree ← choice([d_min, d_max], P_edge)
    node ← choice(P[degree])
    return Nodes(int(node))
end function
```

① 注意这里的节点范围是浮点数而非整数。如果后续确定应该在入度为 2 的节点中选择，则会在 $[0.8, 2]$ 范围内均匀取随机数取整得到结果，即所得结果在 $[0.8, 1)$ 范围内则目标节点为排序中下标为 0 的节点。同时，也可以直接对每个入度对应的节点范围取整处理而非使用浮点数。

3.1.3 社区处理

社区的划分其实就是将节点对应的 ID 进行分类，可以用一个字典结构对社区信息进行存储。实现时可以根据配置信息中指定的每个社区对应的规模，进行节点所属社区的随机分配。

在2.2.2节中曾给出了社区参数 ρ 的定义，表示社区间生成边的概率与社区内部生成边的概率的比值。为了生成能够满足社区结构的图，可以对于生成的每一条边进行检测，得到的结果如果不在同一个社区内则按照概率 $1-\rho$ 进行舍弃。这样得到的图即可满足社区参数 ρ 的要求。

3.2 事件处理

第3.1节对生成静态图的方法进行了探讨，给出了对应的伪代码。而为了生成动态图，需要基于上述过程进行扩展，加入在第2章讨论的动态特性。

在表2.1中已经定义了多种动态事件，此处需要将这些事件进行具体的实现，分别是：

- 节点个数的变化
- 边个数的变化
- 节点在分布中的重要度变化
- 社区参数的变化

在进行事件具体的实现之前，首先对需要记录的数据类型进行分析。除了节点、边、社区信息需要存储之外，节点在分布中的重要度信息也是必不可少的，3.2.1节中将对节点重要性的表征方式进行介绍。

3.2.1 节点映射

在出度确定与边的目标节点确定过程中，都会体现出节点度数的不同，而这也就体现了不同节点的重要程度，并且在事件中也会有与节点重要度变化相关的内容。为了解决这一问题，本节提出了一种叫做“节点映射”的结构，用此结构进行节点重要度信息的存储。

节点映射包括两部分：

- 出度分布：进行出度的分配后，得到的结果代表了节点的重要程度。因此，对于出度分布而言，可以直接使用初次分配得到的节点出度信息。

- 入度分布：第3.1.2节中曾经说明节点排序信息的重要性，入度分布节点映射即为对节点重要程度的表征。

形式上来说，节点映射的形式为：

- 出度分布：节点 ID 到节点出度的字典。
- 入度分布：排序[①]后，节点 ID 组成的数组。

出度分布节点映射中的信息可以指导后续的生成过程。也就是说，对于初始重要度比较高（选定的出度更大）的节点，后续生成过程中的出度往往也会更大。入度分布节点映射则可以直接影响边目标节点的选择，在入度分布的节点映射不变的情况下，节点相对的重要程度在后续生成过程中也不会改变。

3.2.2 事件的实现方式

接下来，对各种事件的具体实现方式进行简要介绍。

由于事件中需要确定每个事件发生、结束的时刻，因此本节引入了“帧”的概念。一帧代表一个时刻，对应于动态图中保存的一个快照。同时，在生成工具配置时需要记录动态图中涉及到的总帧数（动态图存储的快照数目），以便在后续的生成过程中使用。

3.2.2.1 边的生成

在动态图的生成过程中，边的生成是里面非常重要的一部分。一种比较简单的生成方式是：

1. 根据给定的节点配置、边的出度分布信息，计算出每一个节点的出度；
2. 根据给定的帧数进行分配，计算得到该帧需要生成的边个数。这一过程中可以在平均分配的基础上加入随机因素；
3. 根据得到的该帧边数目进行生成操作。

3.2.2.2 节点新增

在新增节点后需要重新调整节点映射，其中的关键在于原有节点重要性信息的保持。

对于出度分布而言，在不改变原有节点出度的基础上计算新的节点出度即可。但是对于入度分布，需要考虑原有节点在所有节点中相对位置，可以用等比例缩放的方法。具体可以用式3-4表示，其中 N_{old}, N_{new} 分别表示新增节点前后的

① 排序在前的节点 ID 重要性较低，入度较小

节点总数，pos, pos_{new} 则分别表示新增节点前后的，某原始节点的位置（数组下标）。

$$pos_{new} = \text{int}\left(pos \times \frac{N_{new}}{N_{old}}\right) \tag{3-4}$$

一种可行的实现方式为：

- 出度分布节点映射：重新生成，为新节点分配出度。
- 入度分布节点映射：将原有节点的位置等比例缩放，剩余的空间随机填入新增的节点。

  例如，原有节点一共有 3 个，入度分布节点映射为 $[1, 2, 0]$。在新增 7 个节点之后，原有节点 ID 调整到了如下位置：$[1, \_, \_, 2, \_, \_, 0, \_, \_, \_]$，空余位置会将新节点调整顺序之后排入。

除此之外，还需要考虑新节点所属社区。在此可以采用随机的方式，根据不同社区的规模按照对应比例进行社区分配。

3.2.2.3 边与节点删除

边的删除较为简单，直接从记录中按照要求去掉对应比例边即可。节点删除会导致其关联的边删除，并且节点映射等结构也会随之受到影响。

对于节点的删除事件，在一些真实社交网络上并不多（相对于节点新增而言），如在微博上关注关系图中用户被官方封号、引文网络某篇文章被撤稿，在这两个社交网络中节点删除的操作并不多。

为了尽量减少总的运算量，可以采用集合的方式记录被删除的节点，在新的生成过程中遇到这个节点后跳过即可。

在实现过程中可以设置一个阈值 α，若删除节点数与总节点数的比值低于 α 则使用前述集合记录被删除节点的方式，反之则进行节点映射的调整，具体操作同节点新增操作。

3.2.2.4 社区结构变化

在日常生活中经常可以看到不同社区之间融合、小团体的形成等现象，这些对应于社区结构的变化。这些变化特征包括社区参数 ρ 的变化与社区的融合、分裂操作。

在3.1.3节中介绍了社区的应用方式，社区结构可以用一个字典形式进行存储，

对社区结构的改变与参数 ρ 的改变只作用于后续新生成的边，新生成的边按照新的社区划分与参数 ρ 进行一定概率的舍弃操作即可。

3.2.2.5 节点重要度改变

在表2.1中，定义了两个与节点重要度改变有关的事件，分别是“突发事件”与“节点重要度变化”，前者是聚焦于突发事件导致的节点重要度的临时性改变，而后者则着重强调节点重要度的永久性改变。临时性改变是可逆的，因此需要保存改变的记录，以便后续恢复。

在节点重要度变化的过程中，修改的其实就是分布对应的节点映射。因此，需要分出度分布和入度分布分别讨论：

- 出度分布节点映射：可以将其中某个节点的出度提高，或者将两个节点的出度互换；
- 入度分布节点映射：将两个节点的位置互换即可。

3.3 工具实现

3.3.1 整体架构

可配置动态图生成工具的整体架构如图3.3所示，主要包括以下几个部分：

- 解析部分：相关配置的输入与解析，进行调度与生成工具的配置与初始化；
- 调度与生成工具部分：接收解析部分得到的配置信息，进行任务安排、相关内容的存储，包含以下部分：
  - 任务列表生成：将图的生成与各种事件抽象为“任务”这一概念，每一个任务可能由几个步骤组成；
  - 行为列表生成：将之前得到的任务进行后续处理，其中对结构影响的基本步骤被称为“行为”（如：修改某个边对应分布的节点映射，修改某个边对应社区的参数）；
  - 调度器：维护一个时钟，每一帧都检查该帧需要执行的基本步骤（即上文的“行为”），进行对应函数的调用；
  - 结构信息存储：存储在生成过程中需要的各种信息，包括：
    * 节点标签、数量、属性，事件中被删除、禁用的节点列表等；
    * 存储相关的文件句柄、目录管理、已生成边和节点的记录等；
    * 边的标签、属性、社区配置、分布配置、节点映射等。

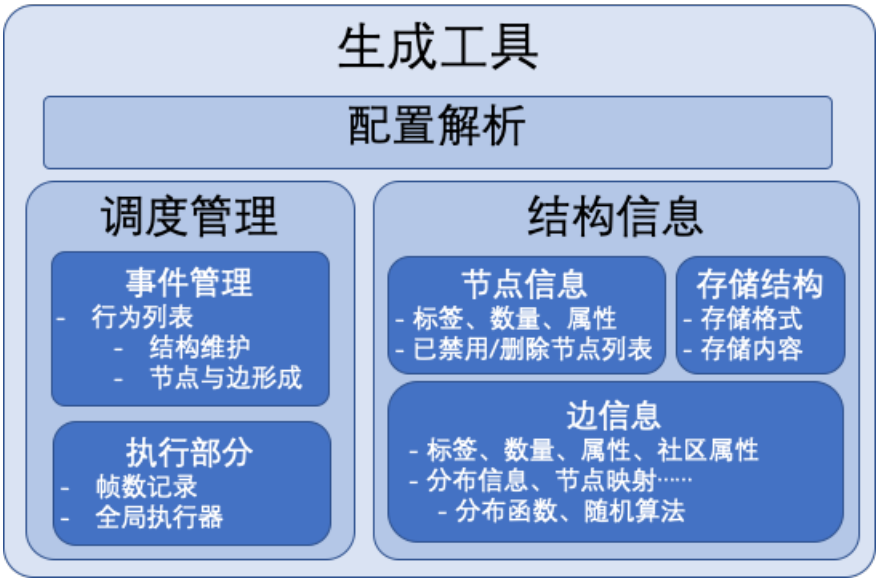

图 3.3　可配置动态图生成工具整体架构

3.3.2　结果保存

本节将介绍在可配置动态图生成工具中实现的一部分存储结构，并且用示例加以说明，具体如表3.1所示。

表3.1中所有列出的格式可以分成快照形式（每一帧的数据单独存储）与增量形式（记录每一帧与之前一帧的区别，以达到节省存储空间的目的）两类，其中增量形式所记录的新增的边、删除的边等内容也可以用与快照形式中相同的方法存储。

除了表3.1中呈现的格式，生成工具中还实现了 networkx 中支持的 GraphML、ADJACENCY、CYTOSCAPE、JIT 等格式，并且可以很方便地修改支持其他格式。

表 3.1　生成的动态图存储结构

分类	名称	说明	示例（之前已有 $0 \to 1$，此帧新增 $0 \to 2, 1 \to 2, 2 \to 0$）
快照形式 *	TSV	文本格式，每行记录一条边	0 1 0 2 1 2 2 0
	ADJ	文本格式，每行记录起始节点与对应的边	0 1 2 1 2 2 0
	CSR	正向表，第一个文件记录节点后继的位置，第二个文件记录所有后继节点	第一个文件：0 2 3 第二个文件：1 2 2 0
	node_link	JSON 格式存储、记录每一个节点、每一条边的信息	{“directed”: True, “multigraph”: False, “graph”: {}, “nodes”: [{“id”: 0}, {“id”: 1}, {“id”: 2}], “links”: [{“source”: 0, “target”: 1}, {“source”: 0, “target”: 2}, {“source”: 1, “target”: 2}, {“source”: 2, “target”: 0}]}
增量形式	DIFF	JSON 格式存储增加与删除的边、增加与删除的节点	{“nodes_added”: [2], “nodes_deleted”: [], “edges_added”: [(0, 2), (1, 2), (2, 0)], “edges_deleted”: []}

*：直接将每一帧的数据保存到一个文件中，以文件命名区分帧数

第 4 章　网页端动态图生成管理系统

在第3章中，动态图生成工具中的动态图配置使用了 JSON 格式，用户手动编写与修改较为困难。本章使用网页配置的形式对配置过程进行优化，可以让用户更方便地进行配置与修改。在此基础上，本章中设计并构建了一个完整的网页端动态图生成管理系统，使用 Vue.js 进行前端整个体系的搭建，其中用 ElementUI 进行美化，用 Echarts 进行相关可视化的操作；并且使用 Django 进行后端 Restful API 的搭建，将第3章中实现的可配置动态图生成工具作为其中的一个组件接入其中，以便用户方便、直观地进行系统的使用。

4.1　系统架构

如图4.1所示，系统使用前后端分离的 Restful 架构设计，在后端 Django 服务中嵌入第3章中实现的可配置动态图生成工具，便于进行处理。

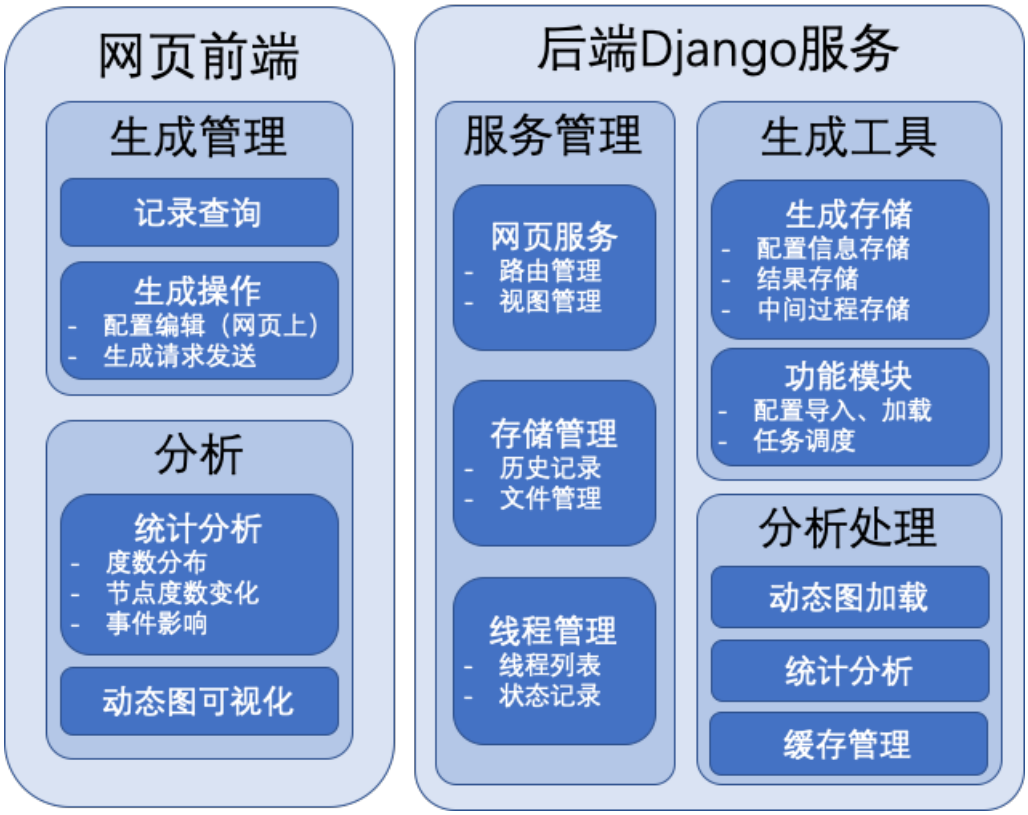

图 4.1　网页端动态图生成管理系统整体架构

前端部分主要的工作集中于图的配置、生成与后续的分析、可视化操作。其中图的生成命令、生成结果与历史记录、分析数据的获取都用网络请求 API 与后端进行交互。

后端部分相对较为复杂，关键点在于将生成工具与网页服务结合在一起。

由于生成过程时间较长会阻塞网页请求，因此在系统中使用了多线程的方法，配置了一个专门的线程管理器，每次有生成请求时便分配给一个新的线程，结果查询时会检测线程是否成功结束。这样的设计也可以很好地避免生成工具部分出现问题导致进程挂起，提高了整个系统的可靠性。

由于需要进行大量的生成与结果查询的工作，在后端服务中设计实现了用于结果存储的模块。使用数据库进行配置信息、生成记录、文件存放位置、线程状态等内容的存储。

对于前端的统计分析、可视化需求，为了节省带宽、加快访问速度，避免不必要的前端计算，采用了带有缓存的动态图后端分析处理体系，在后端完成节点度数统计后直接将结果返回，缓存机制可以避免重复计算，可以更好地减少用户等待时间、提升用户体验。

主页设计如图4.2所示，展示了所有的生成记录，并且使用不同颜色代表不同生成状态。在此页面中用户可以选择某一个记录进行后续的分析与可视化操作。

图 4.2　管理系统-主页设计

在配置选择页面，用户可以用交互式操作进行每一个选项的详细配置。如图4.3、4.4所示。

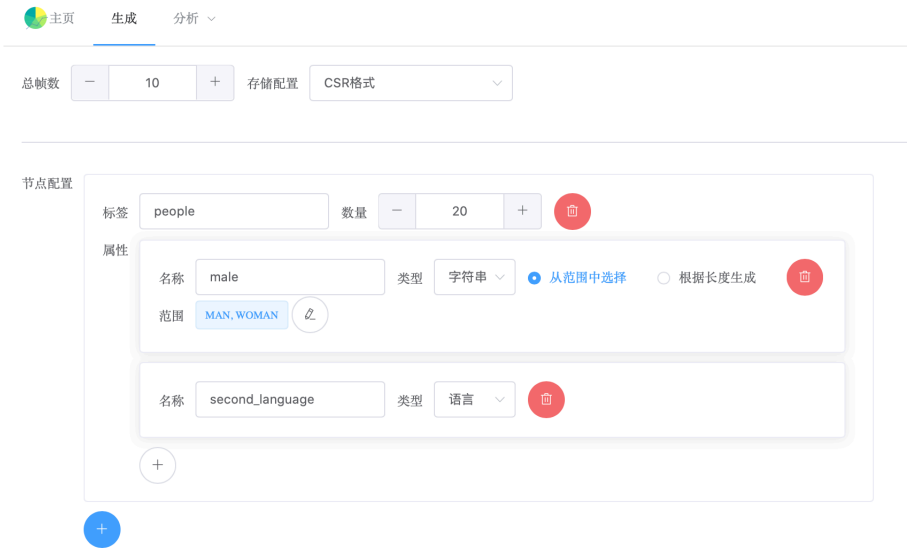

图 4.3　管理系统-生成配置选项 1

图 4.4　管理系统-生成配置选项 2

如图4.5所示，系统中实现了如下几种分析与可视化的显示：度数分布①、度数变化②、节点变化③、图可视化④。

图 4.5　管理系统-分析选项

以度数分布为例，其显示结果如图4.6所示。选择边的标签、时刻信息、出度/入度分布之后，即可看到在这一个时刻的分布特征。

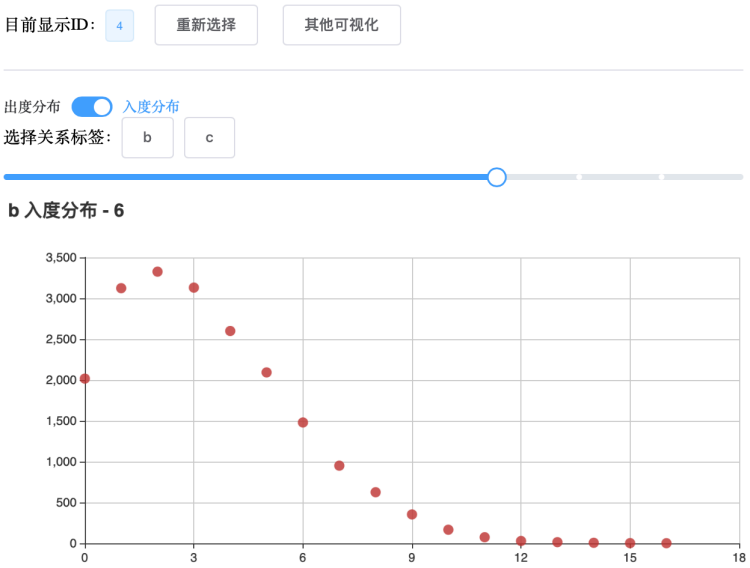

图 4.6　管理系统-度数分布分析

① 度数分布部分展示每一个时刻对应的度数分布图
② 度数变化部分在选定一个节点，进行该节点度数变化分析
③ 节点变化部分展示在各个事件中影响了哪些节点
④ 图可视化部分会逐帧展示每一个时刻的图

4.2 执行流程

系统的具体执行流程如如图4.7与图4.8所示。

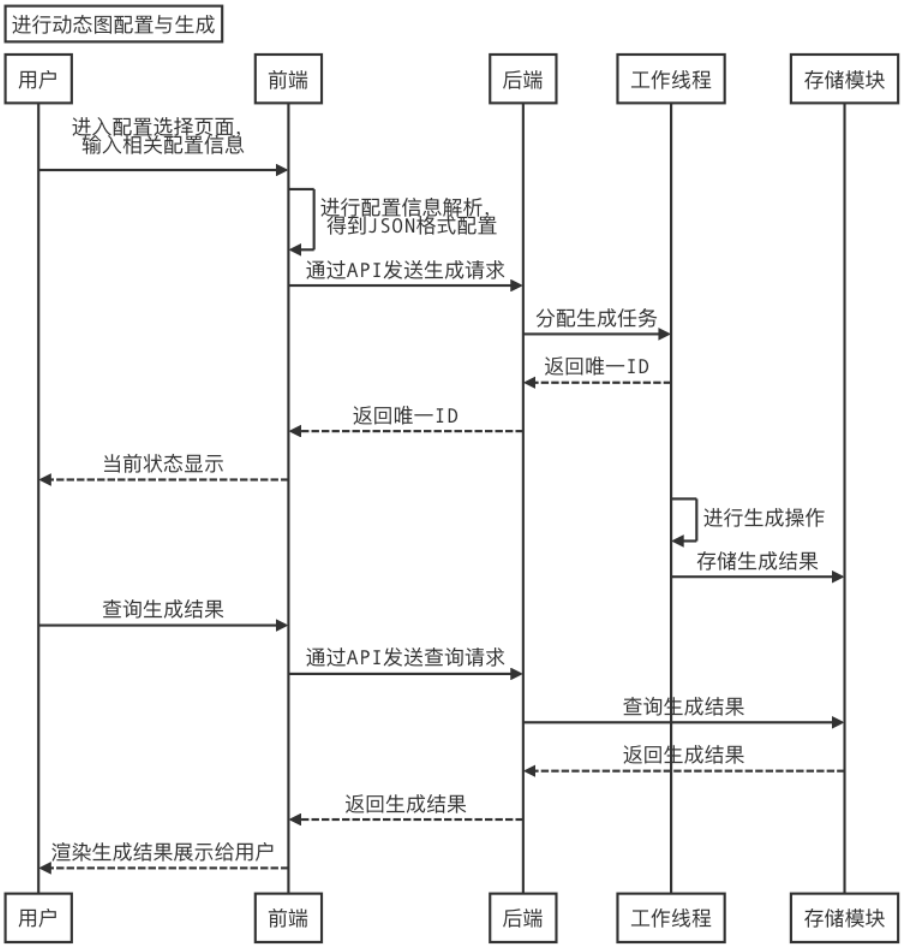

图 4.7　动态图生成过程流程图

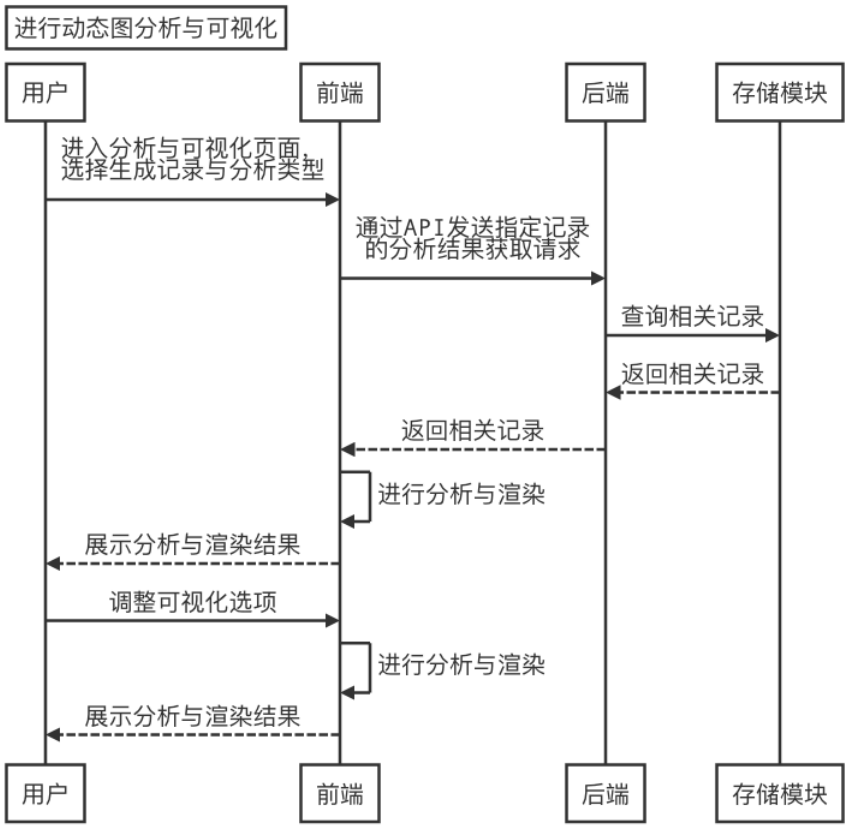

图 4.8　动态图分析与可视化流程图

第 5 章　实验

本章将使用前文设计并实现的动态图生成工具进行实验，检验图生成结果的各项性质并展示生成过程的复杂度。

实验过程中使用的机器配置如下：

- 处理器：2.7 GHz 双核 Intel Core i5
- 内存：8 GB 1867 MHz DDR3
- 系统：macOS 10.15.3

5.1　动态图特征分析

本节将使用指定的配置信息进行动态图的生成，并且检测生成的图是否满足要求。

本节使用如下配置方式，检验入度分布与出度分布是否符合预期：

- 总帧数为 10
- 定义了标签为 student 的节点、标签为 friend 的边，节点总数为 2000，不存在多重边
- 入度分布与出度分布为 $\lambda = 2$ 的 Power-Law 分布
- 社区按照 8：2 的比例进行划分，社区参数 $\rho = 0.5$
- 定义以下事件：
  - 第 3 帧：节点重要度变化，影响 1% 的节点
  - 第 5-7 帧：突发事件导致 1% 节点重要度上升
  - 第 6 帧：社区更为明显，社区参数变化为 $\rho = 0.3$
  - 第 5 帧：节点增加，新增 50 个节点
  - 第 9 帧：节点删除，原有的 90 个节点被删除

由于 Power-Law 分布的概率密度函数为指数形式，因此将入度分布与出度分布使用对数坐标轴呈现，度数范围为 $[1, 100]$ 的结果如图5.1与图5.2所示。

从图中可以看出，结果的出度分布范围与给定条件几乎完全吻合，从图上来看的确符合 Power-Law 分布。

总体来看，入度分布的确符合 Power-Law 分布。但入度分布与给定的条件并不完全相符，出现了许多度数超出配置时的度数范围（$[1, 100]$）的节点。这是因

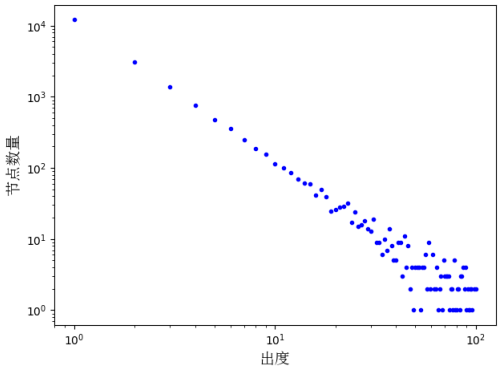

图 5.1　生成结果出度分布

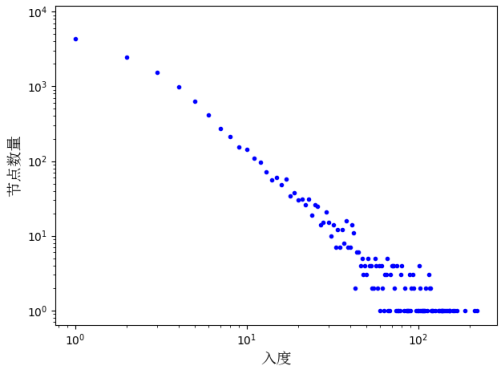

图 5.2　生成结果入度分布

为入度分布的信息只是提供目标节点选择概率的参考，由于入度分布于出度分布可能不相吻合，因此不会保证结果的入度分布完全符合给定的要求。

接下来检验单个节点的度数变化。为了避免度数极小的点对分析造成干扰，在此将度数范围设置为 [30, 100]。

在此选择一些在事件中重要度发生变化的节点。如图5.3与图5.4所示，节点 1514 和节点 326 是节点重要度变化事件中受到影响的节点，其重要性分别在第 3 帧时得到提升与降低。

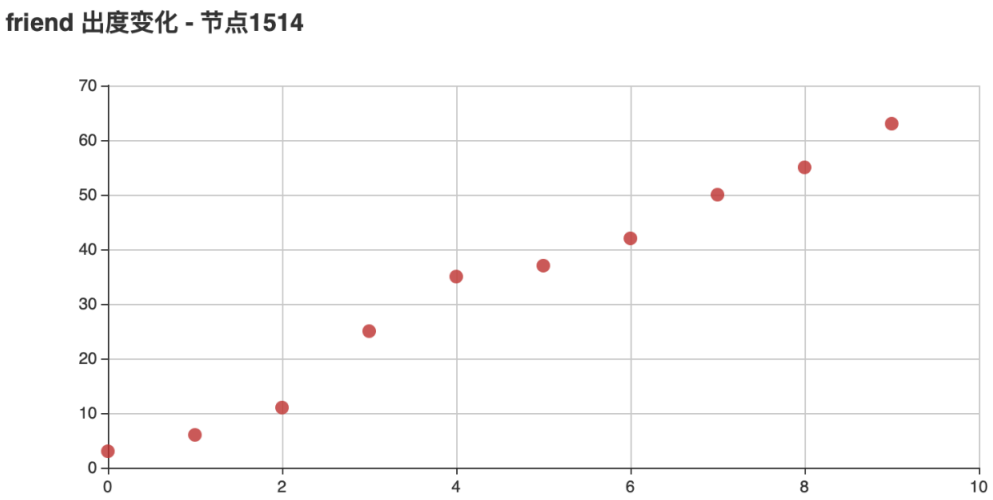

图 5.3　节点重要度提升—以节点 1514 为例

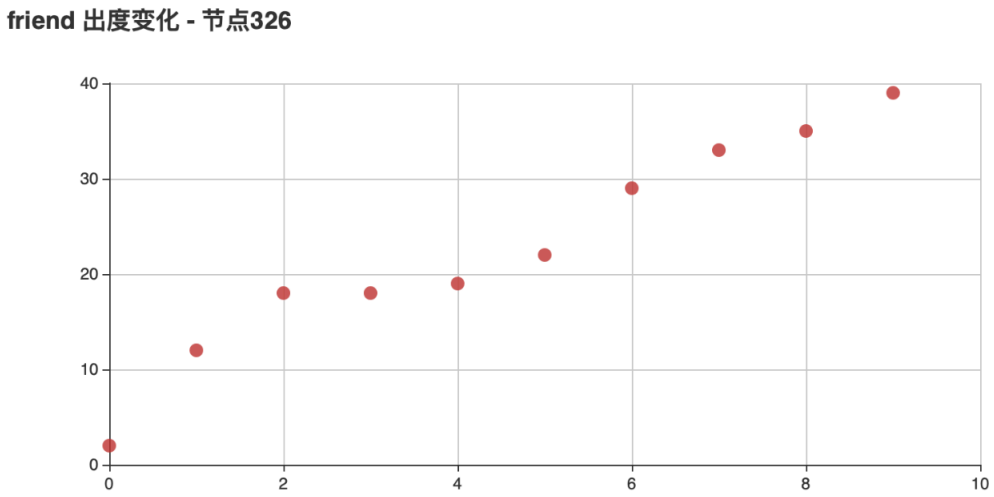

图 5.4　节点重要度降低—以节点 326 为例

第 5-7 帧中的突发事件模拟代表着某些节点度数增长速度的临时加快。如图5.5所示，节点 1866 在该事件中受到影响，第 5-7 帧中度数增长速度明显加快。

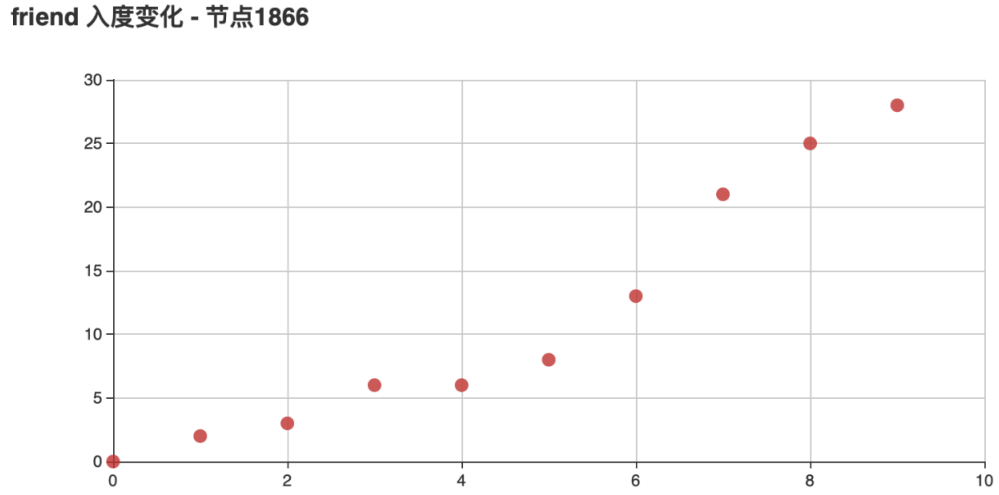

图 5.5　突发事件模拟—以节点 1866 为例

接下来对生成结果的社区特征进行分析。此处将社区划分比例调整为 2：3：5，并且将节点按照社区重排，结果如图5.6所示。

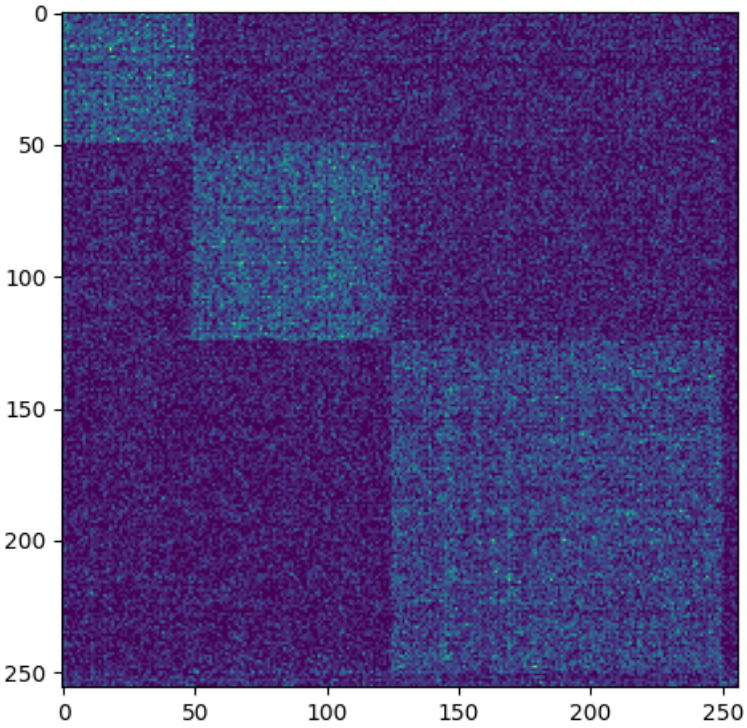

图 5.6　社区可视化

我们可以清楚地看出其中的社区，这可以说明此生成工具的结果符合给定的社区分布。

5.2　性能分析

此节旨在用不同的节点数、边数、帧数等配置信息进行生成，比较所用时间，用以验证生成工具的性能。

若无特殊说明，实验中共用的配置如下：

- 定义了标签为 student 的节点、标签为 friend 的边，不存在多重边
- 出度分布采用 $\lambda = 2$、最大度数为 100、最小度数可变的 Power-Law 分布
- 入度分布与出度分布定义方式相同
- 社区按照 8：2 的比例进行划分，社区参数 $\rho = 0.5$
- 不设置事件
- 使用 ADJ 格式进行存储
- 节点总数、分布的最小度数、边总数[①]、帧数信息会分别在每个实验中说明

① 边的实际数量由边生成过程中通过给定的出度分布确定

5.2.1 所用时间随图规模的变化

本节通过对节点数目、分布的最小度数这两项配置信息的修改，观察所用时间与图的整体规模的关系。

本实验的配置与结果如表5.1所示。

表 5.1 所用时间随图规模的变化

节点总数	最小度数	边总数	帧数	时间 (s)
100	3	867	10	0.114
100	30	5251	10	0.226
1000	3	9581	10	0.640
1000	30	51554	10	1.650
5000	3	47861	10	3.355
5000	30	254777	10	9.619
5000	90	499767	10	11.648
10000	3	93037	10	6.425
10000	30	515136	10	16.006
10000	90	947884	10	21.712
50000	3	476961	10	32.299
50000	30	2555196	10	81.717
50000	90	4736425	10	118.890
100000	3	953490	10	80.333
100000	30	5106549	10	190.093
100000	90	9475146	10	240.551

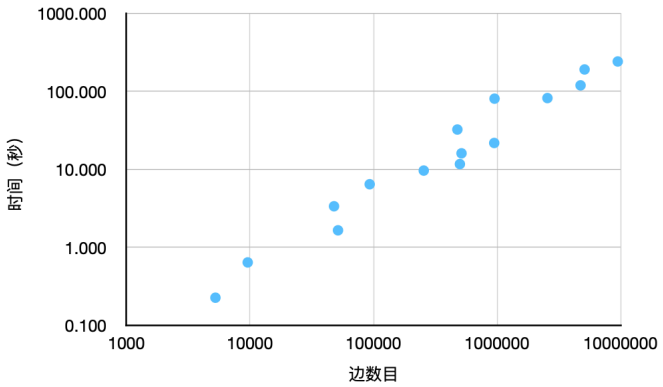

图 5.7 用时与边数目的关系

将表中边数目与所用时间两项抽取出来，如图5.7所示，可以看出总体时间的

增长随边数目呈线性关系。

5.2.2 所用时间随帧数的变化

本节将节点数目、分布的最小度数固定，观察所用时间与总帧数的关系。

本实验的配置与结果如表5.2所示。

表 5.2 所用时间随帧数的变化

节点总数	最小度数	边总数	帧数	时间 (s)
10000	30	509966	1	3.096
10000	30	508252	3	7.663
10000	30	512508	9	14.937
10000	30	512150	27	37.837

从表5.2中可以看出，所用时间与总帧数有着很大的关联，总耗时随着总帧数增加逐渐增长，速度略慢于线性关系。

更大的总帧数会导致每帧中的保存工作消耗更多的时间，并且总的循环次数增多会导致单次循环产生的边更少，生成效率更低。

第 6 章　结论

本文使用事件巧妙地定义了一些典型的社交网络图结构的时序动态性，用一些结构特征的变化来模拟真实社交网络中的变化特性，是一个新颖的动态社交网络图定义方式。这一定义方法高度抽象化，聚焦于图结构本身的特征，并非完全使用模拟的方式进行动态的定义，因此可以方便、快速地通过这种定义进行动态图的生成。

基于对动态图的认识与研究，本文在一个名为 FastSGG 的工作中提出的静态社交网络图快速生成方法基础上进行了改进和扩展，成功将这篇工作中提到的图生成方法扩展应用于动态图的生成过程，构建了一个动态图的配置与生成工具，并且构建了一个完整的网页端生成系统。

在本文设计与实现的动态图生成工具中，设计并应用了节点映射（NodeMap）的概念，将许多动态图的事件定义关联到这一结构中，从而顺利进行动态图的事件执行。

同时，本文中对动态图的生成配置过程进行封装，完成了一个完整的网页端动态图生成管理系统。系统中使用前后端架构，利用 Vue.js、Echarts、ElementUI 进行前端配置、可视化展示体系的搭建，利用 Django 进行后端资源、任务的管理，并使用多线程技术进行生成过程的管理，有效提升了系统的可用性、易用性。与原始的 JSON 形式配置方法相比，文中提供的动态图生成管理系统可以更高效、更方便地进行动态图的配置、运行、结果查看与可视化，并且可以方便用户进行远程生成与管理。

本次研究主要遇到的问题就是如何定义与实现动态社交网络图的动态性、如何在邻接表上用随机方式进行高效生成而非直接使用模拟的方式生成。本文关注社交网络本身的一些结构特征信息，包括图中节点度数的分布情况、节点的度数相对排序，因此使用了名为节点映射（NodeMap）的结构，将节点按照其度数大小进行排序组织，让节点蕴含的度数特性在节点映射中得以保存。在相关事件的处理过程中就可以通过对节点映射的修改来实现。

除此之外，在动态图管理生成系统的搭建过程中也曾经遇到一些问题，如怎样实现数据项的动态添加、如何更高效地进行图的可视化、如何避免耗时较长的生成过程阻塞后端服务进程等。为了解决前端配置与可视化展示的相关问

题，本文中使用了流行的 Vue.js 框架与 echarts 框架进行数据的绑定与可视化，用 ElementUI 进行界面的美化操作。为了解决后端阻塞的问题，文中使用多线程的方式进行处理，收到生成请求之后在后台开启一个新的线程，线程结束后将结果反馈给主线程以便用户进行结果的查看、分析。

由于时间与经验所限，本课题中提出的用事件进行定义、用节点映射（NodeMap）进行实现的方法并不是尽善尽美，这样的方式使用的是图结构的相关特征，关注点在于动态图的结构方面的变化，而一定程度上忽视了结构变化内在的联系与规律，如真实网络中一些节点的变化趋势并非相同，符合某些特征的节点更有可能在给定的事件中受到影响，实践中相关数据生成、事件处理方面使用的随机方式有很大的提升空间。并且本次研究中高度自由化的配置方法使得用户在所提供的可配置选项中进行的配置不一定可以符合真实网络中的相关特征，一些动态图的内部节点间关联等更多特征难以用这里提出的基于随机的动态图模拟方法进行模拟。

对于动态社交网络图的研究还有很大的空间，在对真实网络进行后续分析的基础上可能可以提出更合适的方法。结合神经网络的方式，用 RNN 提取之前的节点特征来进行后续事件配置，指导事件中影响到的节点的选择与具体的事件执行过程，可能会有更符合真实情况的结果。

插图索引

表格索引

公式索引

参考文献

[1] BIGGS N, LLOYD E K, WILSON R J. Graph theory, 1736-1936[M]. Oxford University Press, 1986.

[2] DALEY D J, KENDALL D G. Stochastic rumours[J]. IMA Journal of Applied Mathematics, 1965, 1(1):42-55.

[3] MAKI D P, THOMPSON M. Mathematical models and applications: with emphasis on the social life, and management sciences[R]. 1973.

[4] SARKAR P, MOORE A W. Dynamic social network analysis using latent space models[C]// Advances in Neural Information Processing Systems. 2006: 1145-1152.

[5] BRAHA D, BAR-YAM Y. From centrality to temporary fame: Dynamic centrality in complex networks[J]. Complexity, 2006, 12(2):59-63.

[6] PHAM M D, BONCZ P, ERLING O. S3g2: A scalable structure-correlated social graph generator[J]. 2012.

[7] WANG C, WANG B, HUANG B. Fastsgg: Efficient social graph generation using a degree distribution generation model[R]. 2019.

[8] YOU J, YING R, REN X, et al. Graphrnn: Generating realistic graphs with deep auto-regressive models[J]. 2018.

[9] DE CAUX R, SMITH C, KNIVETON D, et al. Dynamic, small-world social network generation through local agent interactions[J]. Complexity, 2014, 19(6):44-53.

[10] FISCHER F, HELMBERG C. Dynamic graph generation for large scale operational train timetabling[J]. 2011.

[11] KIM D, YOO Y, KIM J, et al. Dynamic graph generation network: Generating relational knowledge from diagrams[J]. 2017.

[12] Yu S, Chen D, Li B, et al. A personalized recommendation algorithm based on interest graph [C]//The 2014 2nd International Conference on Systems and Informatics (ICSAI 2014). 2014: 933-937.

[13] NAJAFABADI M K, MOHAMED A, ONN C W. An impact of time and item influencer in collaborative filtering recommendations using graph-based model[J/OL]. Information Processing & Management, 2019, 56(3):526 - 540. http://www.sciencedirect.com/science/article/pii/S0306457318303376. DOI: https://doi.org/10.1016/j.ipm.2018.12.007.

[14] PAREJA A, DOMENICONI G, CHEN J, et al. Evolvegcn: Evolving graph convolutional networks for dynamic graphs[J]. arXiv preprint arXiv:1902.10191, 2019.

[15] ONNELA J P, SARAMÄKI J, HYVÖNEN J, et al. Structure and tie strengths in mobile communication networks[J]. Proceedings of the national academy of sciences, 2007, 104(18):7332-7336.

[16] WATTS D J, STROGATZ S H. Collective dynamics of ‘small-world’ networks[J]. nature, 1998, 393(6684):440.

[17] GIRVAN M, NEWMAN M E. Community structure in social and biological networks[J]. Proceedings of the national academy of sciences, 2002, 99(12):7821-7826.

[18] SORAMAKI K, BECH M, ARNOLD J, et al. The topology of interbank payment flows [J/OL]. Physica A: Statistical Mechanics and its Applications, 2006, 379:317-333. DOI: 10.1016/j.physa.2006.11.093.

[19] STEYVERS M, TENENBAUM J B. The large-scale structure of semantic networks: Statistical analyses and a model of semantic growth[J]. Cogn Sci, 2010, 29(1):41-78.

[20] GUREVITCH M. The social structure of acquaintanceship networks[J]. Massachusetts Institute of Technology, 1961.

[21] DODDS P S, MUHAMAD R, WATTS D J. An experimental study of search in global social networks[J/OL]. Science, 2003, 301(5634):827-829. https://science.sciencemag.org/content/301/5634/827. DOI: 10.1126/science.1081058.

[22] PALLA G, DERÉNYI I, FARKAS I, et al. Uncovering the overlapping community structure of complex networks in nature and society[J]. nature, 2005, 435(7043):814-818.

[23] GIRVAN M, NEWMAN M E J. Community structure in social and biological networks[J]. Proceedings of the National Academy of ences, 2002, 99(12):7821-7826.