计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
2025,34(11):42−55 [doi: 10.15888/j.cnki.csa.009997] [CSTR: 32024.14.csa.009997]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

时序图模式匹配的查询优化①

覃紫云[1], 郭青松[2], 马国帅[1], 张凯涵[1], 石　琼[1], 蔡江辉[1]

[1](中北大学 计算机科学与技术学院, 太原 030051)
[2](湖南工业大学 计算机与人工智能学院, 株洲 412007)
通信作者: 郭青松, E-mail: qingsongg@gmail.com

摘　要: 图模拟被广泛应用于近似回答复杂的图模式匹配问题. 针对时序图模式匹配的查询效率问题, 本文提出一种基于联合图的 SimAssign 图模拟算法及动态规划的优化框架. 其核心思想在于: (1) 将时序图的快照序列合并为联合图, 首先利用静态图模拟算法在联合图上计算初始查询结果, 然后根据时间区间将结果分发至每个快照, 从而生成相应的匹配结果; (2) 为图模拟策略构建统一的代价模型, 并基于动态规划优化图模式匹配的执行策略. 本文在真实数据集和合成数据集上进行了全面实验, 实验结果表明代价模型能够准确估计静态图模拟、增量图模拟以及 SimAssign 等图模拟策略的执行代价, 并且优化方法能够选择出最优的执行方案. 在绝大多数情况下, SimAssign 的性能显著优于静态图模拟和增量式图模拟算法, 在部分实验中 SimAssign 的性能达到增量法的 4 倍.

关键词: 时序图; 图模式匹配; 图模拟; 代价模型; 动态规划

引用格式: 覃紫云,郭青松,马国帅,张凯涵,石琼,蔡江辉.时序图模式匹配的查询优化.计算机系统应用,2025,34(11):42–55. http://www.c-s-a.org.cn/1003-3254/9997.html

Query Optimization of Temporal Graph Pattern Matching

TAN Zi-Yun[1], GUO Qing-Song[2], MA Guo-Shuai[1], ZHANG Kai-Han[1], SHI Qiong[1], CAI Jiang-Hui[1]

[1](School of Computer Science & Technology, North University of China, Taiyuan 030051, China)
[2](School of Computer Science and Artificial Intelligence, Hunan University of Technology, Zhuzhou 412007, China)

Abstract: Graph simulation is widely adopted to approximately solve complex graph pattern matching problems. To enhance the query efficiency for matching temporal graph patterns, this study proposes a simulation algorithm, SimAssign, which is based on a union graph model, along with a dynamic programming based query optimization framework. The proposed approach consists of two main components: (1) a sequence of temporal graph snapshots is merged into a union graph, on which initial query results are computed using a static graph simulation algorithm; these results are then partitioned according to temporal intervals to generate matching results for each snapshot; (2) a unified cost model is developed to support various graph simulation strategies, and a dynamic programming framework is introduced to identify optimal execution plans based on this model. Extensive experiments on both real and synthetic datasets demonstrate that the cost model accurately estimates the execution costs of static graph simulation, incremental graph simulation, and SimAssign. Moreover, the proposed optimization framework consistently selects optimal strategies. In most cases, SimAssign substantially outperforms both static and incremental simulation methods, with performance improvements reaching up to four times that of the incremental approach on certain datasets.

Key words: temporal graph; graph pattern matching; graph simulation; cost model; dynamic programing

① 基金项目: 山西省科技合作交流专项 (202204041101033); 山西省基础研究计划 (202203021222075)
收稿时间: 2025-03-26; 修改时间: 2025-04-29; 采用时间: 2025-05-26; csa 在线出版时间: 2025-09-30
CNKI 网络首发时间: 2025-10-09

时序图 (temporal graph) 通过引入时间维度扩展了传统图模型, 能够更加精确地刻画现实世界中动态变化的网络结构, 如社交网络中的用户交互、交通网络的实时状态变化以及生物分子间的相互作用等[1,2]. 根据报道, Facebook 月活跃用户预计 2024 年达 30.7 亿[3], 随着数据规模的快速增长, 时序图的建模与分析已成为图数据库、网络科学和图神经网络等领域的研究热点. 本文聚焦时序图模式匹配 (temporal graph pattern matching, TGPM), 深入研究使用图模拟近似回答时序图匹配时的查询优化问题. 通过时序图模式匹配能够完成检索时序模体 (temporal motifs)[4,5]、时序旅行 (temporal journey)[6]以及时序路径 (temporal path)[7,8]等任务.

图模式匹配是图数据库的核心问题, 其目标是从数据图中检索出与给定模式图结构相匹配的子图. 该问题在众多领域都有着极为广泛的应用, 例如社交网络、金融、生物信息学等. 在过去近半个世纪里, 它一直是学术界的研究热点. 自 20 世纪 70 年代 Ullmann[9] 提出首个子图同构算法后, 学界相继发展了 VF2[10]、QuickSI[11]、GraphQL[12]、GADDI[13]、SPath[14] 等一系列经典算法. 这些算法主要是针对静态图设计的, 即假定图的拓扑结构保持固定, 边和顶点的属性不会随着时间发生改变[15]. 然而, 在现实世界中, 图数据本质上是动态演化的. 以社交网络为例, 用户兴趣的漂移以及话题的传播都具有明显的时序特征. 再如交通网络, 路况拥堵程度会随着不同时段而动态变化. 若直接将静态图模式匹配算法应用于时序图分析, 往往会引发一些问题, 例如在航空网络行程规划场景中, 静态算法可能会返回“经停时间不足”的无效路径.

在时序图上进行图模式匹配是一个极具挑战性的问题. 一种较为直观的解决方法是按照时间顺序将时序图划分为一系列快照图$(G_1,G_2,\cdots,G_l)$, 随后在每个快照图G_i上直接运用静态图模式匹配方法来处理时态查询. 这种方法存在明显缺陷, 它未能充分利用相邻快照之间重叠的计算结果, 导致大量的重复匹配. 为了提升时序图上图模式查询的效率, 许多研究工作将时序图视为图流, 并采用增量查询处理技术, 尽量避免在更新过程中重复计算相邻快照间重叠部分的匹配结果, 以此提升图模式匹配的查询效率. 增量法能大幅提升单个图模式匹配的响应时间, 在特定应用场景下具有一定优势, 但是不能从整体上优化图模式匹配的执行效率. 当模式图包含环且操作为边插入时, 使用增量图模式匹配需要进行全局检查以重新识别匹配结果, 同时维护的辅助信息也会带来大量额外开销, 反而会大幅降低图模式匹配的执行效率.

基于上述分析, 将静态图匹配算法与增量法相结合是一种有效的解决方案. 具体而言, 可以先将时序图的快照序列整合为一个联合图, 以该联合图的匹配结果为基础, 通过计算仅包含边删除操作的增量图, 并根据边的有效时间区间将初始的匹配结果分发到每个快照, 生成相应的匹配结果. 通过上述操作, 既可以避免重复检测每条边的匹配, 又可以避免在新插入边造成环时引发的全局检查, 可以极大地提高图模式匹配的整体效率.

图模式匹配的复杂度很高, 是经典的 NP-hard 问题. 近年来, 图模拟 (graph simulation) 技术被广泛应用于近似回答复杂的图模式匹配问题. 针对时序图模式匹配已有方法的缺陷, 本文提出了一种基于联合图的 SimAssign 图模拟算法及基于动态规划的优化框架. 其核心思想如下: (1) SimAssign 算法首先将时序图的快照序列合并为一个联合图, 在联合图上应用静态图模拟算法生成初始的匹配结果, 然后依据边的有效时间区间, 将初始匹配结果中的边分配给各个快照; (2) 还设计了一个图模式匹配的代价模型, 并基于动态规划为 SimAssign 算法生成最优的执行计划.

本文针对时序图上的图模式匹配问题, 提出一种高效的图模拟算法及优化框架. 本文主要贡献如下: (1) 提出一种用于高效计算时序图模拟的算法 SimAssign. 该算法有效克服了静态图模式匹配无法重用跨快照重叠匹配结果的局限, 以及增量法在处理模式图中含环且图更新操作为插入边时的不足. (2) 设计了一个通用的代价模型, 可以准确估计静态图模式匹配、增量法和 SimAssign 的执行代价. (3) 基于动态规划策略设计了一个针对图模式匹配的优化框架, 能生成 SimAssign 策略的最优执行计划. (4) 通过在真实数据集和合成数据集上的实验, 验证了上述方法的有效性.

1 相关工作

1.1 时序图数据管理

时态数据管理可以追溯到 20 世纪 80 年代甚至更早, 而时序图数据管理却在近 10 年才受到广泛关注. 当前时序图数据管理研究主要围绕紧凑存储、分布式

查询, 及如何高效地实现时序查询等问题展开. Neo4j[16]等一大批图数据库系统被设计出来用于加强图数据的管理. 这些系统大多采用属性图模型 (property graph model), 将数据表示为有向的、带属性的多重图, 以在数据表达能力和建模复杂性之间取得平衡.

当前的研究趋势之一是将图更新与图计算解耦, 通过快照 (snapshots) 捕捉图的演化行为, 并利用相邻快照之间的图结构具有较大重叠的特点减少存储空间, 即通过增量 (delta) 实现紧凑存储. Khurana 等人[17]提出了一种索引结构 DeltaGraph 来保存动态图的历史变化信息, 并支持高效地快照构建和扩展查询. Macko 等人[18]提出的 LLAMA 采用写时复制 (copy-on-write) 机制管理快照, 并利用局部性优化实现多核并行计算. Labouseur 等人[19]提出 G*来将动态图存储为紧凑的快照. ImmortalGraph[20]将数据存储在内存中, 并在重叠快照上共享存储和计算. Then 等人[21]提出了 SAMS, 通过对多个快照同时执行算法来加速图分析, 但在进行小子图模式查询时, 代价较基于模型的方法较高, 因为其需要完整的物化快照. Raphtory[22]允许通过数据流进行更新, 在内存中维护完整的图历史记录, 但该方法会对全局查询性能产生负面影响.

在时序图查询的相关研究中, 路径及可达查询得到广泛关注. 时序图上路径的相关定义不同于静态图, 主要包括最短长度路径、最小代价路径、最早到达路径、最晚出发路径及最快到达路径[23]. Byun 等人[24]提出的 ChronoGraph 系统实现了时序广度优先搜索、时序深度优先搜索及时序单源最短路径查询. Johnson 等人[25]提出了一种类似 SQL 的查询语言 Nepal, 用于支持多层通信网络上的时序路径查询. Debrouvier 等人[26]提出了一种类似 Cypher 的图查询语言 T-GQL. Areans 等人[27]扩展了流行图查询语言 MATCH, 定义了时序正则化路径查询 (Temporal regular path query, TRPQ) 的语义.

1.2 图模式匹配和图模拟

图模式匹配是指从数据图中找到所有与模式图匹配的子图, 匹配的语义通常由子图同构定义, 是经典的 NP-完全问题[9]. 现有图模式匹配算法主要分为静态[9–15]和增量[28–39]两类.

静态图模式匹配算法假设数据是静态的, 不考虑数据随时间的变化. 自 1970 年起, 静态图模式匹配算法得到了广泛的研究, 包括 Ullmann[9]算法、VF2[10]、QuickSI[11]、GraphQL[12]、GADDI[13]、SPath[14]等子图同构算法. Henzinger 等人[15]提出了一种时间复杂度为$O((|V|+|V_P|)(|E|+|E_P|))$的静态图模拟算法 (StaticSim), 其中$|V|$、$|E|$和$|V_P|$、$|E_P|$是数据图 (模式图) 的顶点数和边数. 静态方法的吞吐率很高, 但是对内存和计算资源的需求很高, 无法很好地应用到动态图上.

近年来一些学者提出了使用图模拟来近似回答时序图上的图模式匹配[28–35], 通过对匹配条件进行适当放松, 以应用于更多应用场景, 如结构索引、过程演算、网站分类等. Gao 等人[28]研究了时序图上的时间相关流图模式匹配, 提出了一个基准算法, 包括图模拟、时间过滤及寻找源匹配图这 3 个步骤, 但存在大量冗余匹配, 为此又提出了一种优化算法, 使用拓扑排序执行模式匹配来进一步减少执行时间. Zhang 等人[29]研究了对偶模拟问题, 综合考虑了多种更新操作, 并采用节点匹配状态检测序列的优化方法, 提出了 一种更高效的增量匹配算法. Fan 等人[30,32]提出了有界模拟将模式边映射到数据图中限定长度的路径, 提出通过增量法来回答时序图上的图模拟查询. Zhang 等人[33]研究了基于模拟的约束时间图模式匹配, 提出了一种称为约束时间对偶模拟的放松匹配规则, 该方法首先将数据图分解为源时间连通分量, 然后对分解后的子图进行匹配. 该方法保留了祖先和后代的时间连通性, 实现了边到时间的路径映射. Ma 等人[34]研究了时序图上的有界模拟问题, 为此提出了一个模拟匹配框架. 该框架包含 3 个阶段, 即模式分割、模式段的有界模拟及结果集成. Bouhenni 等人[35]提出了有界对偶模拟 BDSim 来捕捉更多的语义相似性, 通过消除生成的匹配图中无界长度的环路来保持顶点的邻近性.

增量法将时序图看成图流的形式, 通过索引结构存储快照之间公共的匹配结果来避免重复计算, 在图更新的同时更新匹配结果. 但是, 当模式图包含环时, 增量法的效率变得很差. 本文将对静态图模拟方法 (StaticSim) 和增量图模拟方法 (IncSim) 进行改进, 优化时序图上图模拟查询的效率.

2 数据模型和问题定义

本文采用时序属性图 (temporal property graph, TPG) 作为数据模型. TPG 扩展了带标签属性图 LPG[26]模型的定义, 为数据图中的顶点和边引入有效时间区间的概念.

定义 1. 时序属性图 (TPG). TPG 可以表示为一个带标签的有向图$G=(V,E,L)$, 其中V是图G中的有限顶点集, $E\in(V\times V)$是时序边的集合, L为图G的标签集. 图G的任意顶点$v\in V$表示为一个五元组$v=(vid,l,p,ts,te)$, 任意边$e\in E$表示为一个七元组$e=(eid,u,v,l,p,ts,te)$. 其中, vid和eid分别为顶点和边的标识符; l为顶点或边的标签, 由函数$f:V\cup E\to L$给定; p为顶点或边的属性; u和v分别为边e的源顶点和终止顶点; $[ts,te)$为顶点或边的有效时间区间, 其中$ts<te$.

如图 1 所示, 时序图的更新可以按时间顺序表示为一个无限长度的操作序列$S=(u_1,u_2,\cdots)$. 更新u_i表示为一个三元组$u_i=(t,id,op)$, 其中, t、id、op分别为更新u_i的提交时间、标识符、操作类型.

更新操作有 3 种类型, 分别为插入 (Insert)、删除 (Delete) 顶点或边、更新顶点属性 (Update). 顶点的时间区间 (生命周期) 不能短于其关联的边的时间区间, 所以执行更新操作时需要进行约束条件检测和修改时间戳.

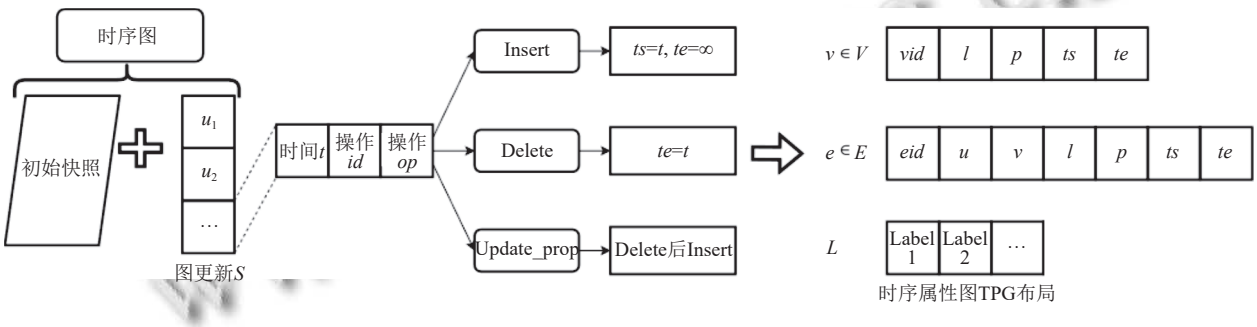

图 1 时序图模型的更新

(1) 插入操作: 执行边插入操作前, 要求与之关联的顶点在图中已经存在. 插入新的顶点或边时, 其开始时间戳ts修改为操作提交时间t, 终止时间戳te修改为∞.

(2) 删除操作: 删除一个顶点前, 要求没有与该顶点关联的边存在. 删除顶点或边时需要将其终止时间戳te修改为操作提交的时间t.

(3) 更新操作: 更新操作要求对应的顶点和边存在, 且更新操作不影响其有效时间区间.

如图 2 所示, 时序图G中包含 7 个顶点和 9 条时序边. 其中, 顶点v_2的有效时间区间为$[0,100)$, 这表示v_2在$t=0$时被插入, 在$t=100$时被删除; v_3的有效时间区间为$[10,\infty)$, 这表示v_3在$t=10$时被插入, 到目前为止还没被删除.

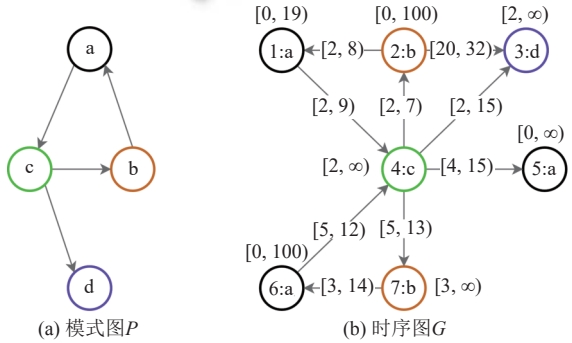

图 2 时序图 G 和模式图 P 实例

基于子图同构的图模式匹配条件十分严格, 子图同构的复杂度已被证明是 NP-完全[10]. 基于子图同构的图模式匹配的详细定义如下.

定义 2. 图模式匹配 (GPM). GPM 是指在数据图$G=(V,E,L)$中查找与模式图$P=(V_P,E_P,L_P)$同构的子图$G_S=(V_S,E_S,L_S)$. 即寻找满足如下条件的双射函数$f:V_P\to V_S$: (1) 对于任意模式图中的顶点$u\in V_P$, 都存在顶点$f(u)\in V_S$使得$L(u)=L(f(u))$; (2) 对于任意模式图中的边$(u,v)\in E_P$, 存在$(f(u),f(v))\in E_S$使得$L(u,v)=L(f(u),f(v))$.

近年来基于图模拟近似回答的图模式匹配得到了广泛关注. 图模拟对子图同构的匹配条件进行了适当放松, 其复杂度为多项式级别[30]. 图模拟详细定义如下.

定义 3. 图模拟 (graph simulation). 给定模式图$P=(V_P,E_P,L_P)$和数据图$G=(V,E,L)$. P和G之间的图模拟由二元关系$R\subset V_P\times V$所确定, 并且R满足如下条件: (1) 对于任意模式图中的顶点$u\in V_P$, 都存在对应数据图中的顶点$v\in V$使得$(u,v)\in R$; (2) 对于任意$(u,v)\in R$满足: 1) u和v有相同的标签, 即$L(u)=L(v)$; 2) 对于任意模式图中的边$(u,u')\in E_P$, 都存在数据图中的边(v,v')使得$(u',v')\in R$.

在时序图查询中, 用户通常只对部分历史数据感兴趣, 本文引入滑动窗口 (sliding window) 来精确描述用户查询的时间区间. 滑动窗口可以表示为一个四元组$SW=(ts,te,len,slide)$, 其中, ts和te $(ts<te)$ 分别为滑动窗口的起止时间, len和$slide$分别为窗口的长度和滑动步长. 滑动窗口SW包含一系列窗口实例, 即$SW=(w_1,w_2,\cdots w_i,\cdots)$. 其中$w_i=[ts_i,te_i)$为第$i$个窗口实例, 并且$ts_i=ts+(i-1)\times slide$, $te_i=ts_i+len$.

将滑动窗口SW作用于时序图G后, 可生成与该窗口对应的快照图序列$(G_1,G_2,\cdots,G_i,\cdots,G_n)$. 对于时序图$G$, 时间区间被窗口$w_i$完全覆盖的顶点和边, 均划入相应的快照图$G_i$. 假设$G_i$上图模拟的匹配结果为$S_i$, 则整个滑动窗口对应快照序列上的图模拟匹配结果为$(S_1,S_2,\cdots,S_i,\cdots,S_n)$, 如图 3 所示.

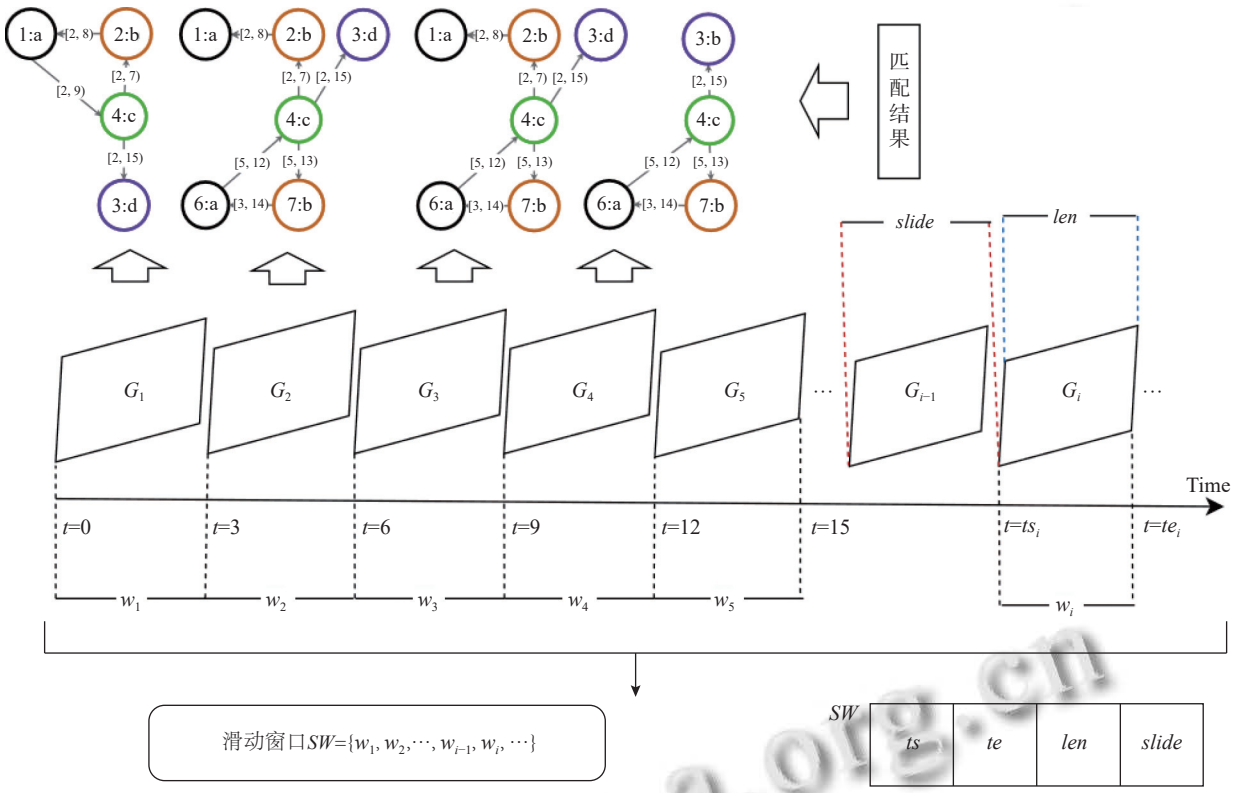

图 3 滑动窗口上的图模拟匹配

基于图模拟和滑动窗口, 可以严格定义时序图上的图模式匹配.

定义 4. 时序图模式匹配 (temporal GPM). 给定时序图$G=(V,E,L)$、模式图$P=(V_P,E_P,L_P)$和滑动窗口$SW=(ts,te,len,slide)$, 时序图模式匹配是指在时序图G上应用滑动窗口SW得到快照图序列$(G_1,G_2,\cdots,G_i,\cdots,G_n)$, 并在快照图序列上执行模式图$P$的图模拟得到匹配结果$(S_1,S_2,\cdots,S_i,\cdots,S_n)$.

如图 3 所示, 给定滑动窗口$SW=(0,15,3,3)$, 可生成 5 个窗口实例w_i, 即$w_1=[0,3)$、$w_2=[3,6)$、$w_3=[6,9)$、$w_4=[9,12)$、$w_5=[12,15)$. 通过将滑动窗口SW应用于数据图G上, 可以得到 5 个快照图(G_1,G_2,G_3,G_4,G_5). 其中, 每个快照对应的时序图模式匹配结果S_i分别为:

$$\begin{cases} S_1=\{(a,v_1),(b,v_2),(c,v_4),(d,v_3)\} \\ S_2=\{(a,v_1),(a,v_6),(b,v_2),(b,v_7),(c,v_4),(d,v_3)\} \\ S_3=\{(a,v_1),(a,v_6),(b,v_2),(b,v_7),(c,v_4),(d,v_3)\} \\ S_4=\{(a,v_6),(b,v_7),(c,v_4),(d,v_3)\} \end{cases} \tag{1}$$

3 时序图模式匹配

为解决时序图上的图模式匹配问题, 本文设计了图模拟匹配算法 SimAssign, 并为该算法基于代价模型设计了优化算法 MCDP. 这两个算法分别承担匹配和优化的关键功能, 共同构成了本文提出的图模式匹配系统, 整体设计如图 4 所示. SimAssign 作为核心匹配

模块, 直接对时序图数据进行结构和时间双重约束的子图匹配操作; 而动态规划算法 MCDP 则作为前置模块, 依据预设成本为 SimAssign 提供最优的快照粒度划分方案, 具有良好的实用性和扩展性.

3.1 时序图模拟算法

SimAssign 利用滑动窗口将静态结构匹配与时间区间验证相结合, 通过传播更新机制实现时序图上精确、高效的图模拟查询. 如图 4 中的 SimAssign 部分所示, SimAssign 算法包含 Sim 和 Assign 两部分. 算法的基本思想是将合并快照图 (快照段) 上的图模拟结果 (Sim), 通过检测图模拟条件和数据图中相关边的有效时间区间来更新顶点对的有效匹配时间, 通过迭代计算传播顶点对时间区间更新后的影响. 最后将匹配的数据边分发到对应的窗口完成匹配 (Assign). 时序图模拟算法计算过程中会涉及顶点对有效时间区间信息的更新传播, 特别是当模式图有环时可能需要进行多次时间信息更新. 为此传播过程引入堆栈数据结构来临时保存需要进行更新的候选匹配顶点对.

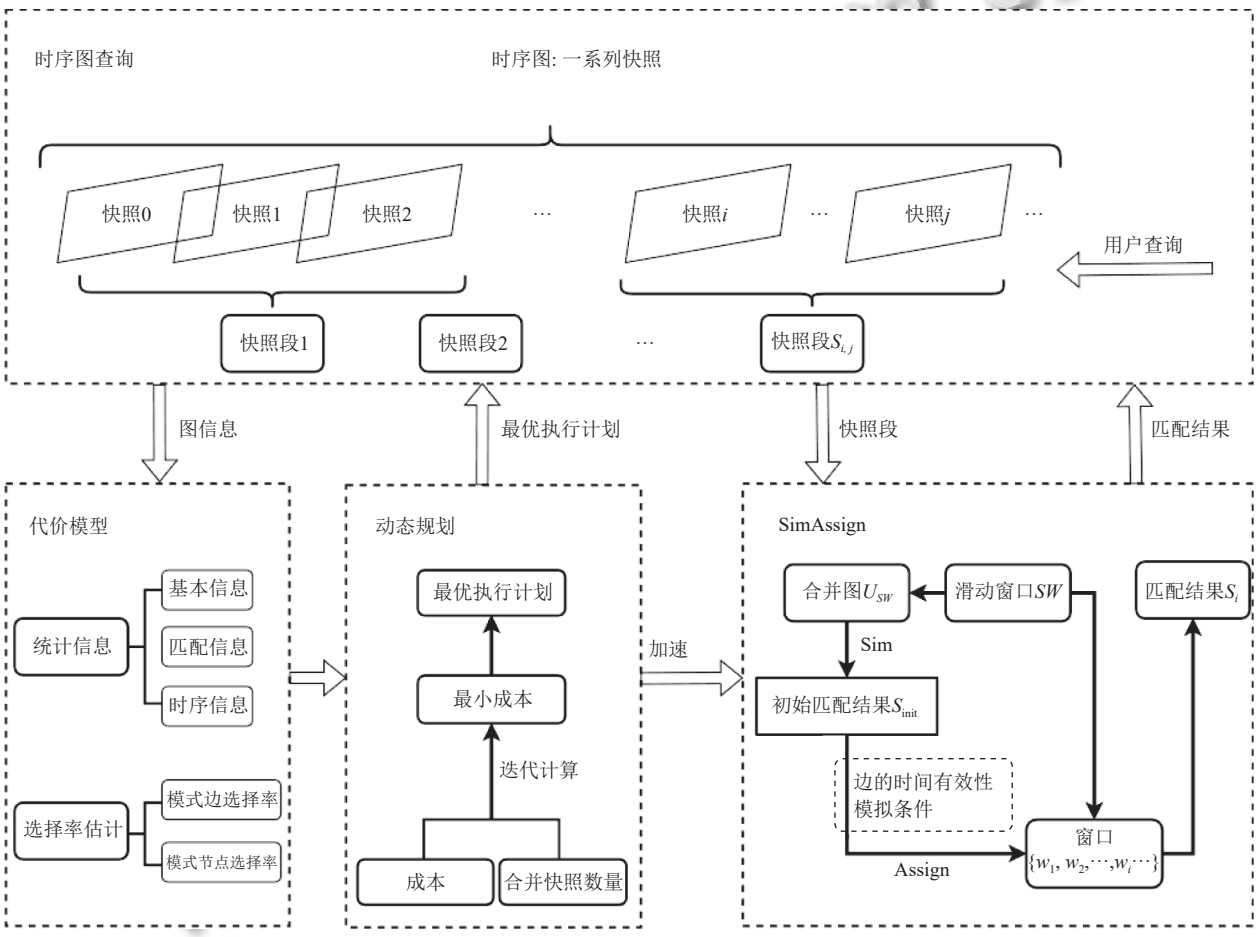

图 4 基于联合图的 SimAssign 图模拟算法及动态规划的优化框架

如图 4 中的 SimAssign 部分所示, 时序图模拟算法首先在合并图 $U_{SW}=G_1\cup G_2\cup\cdots\cup G_n$ 上执行静态图模拟算法 (StaticSim), 得到初始的匹配结果 S_{init}. 由于时序图中顶点和边的删除操作只反映在有效时间区间的变化, 所以根据初始匹配 S_{init} 中的顶点对 (u,v) 与数据图中的顶点 v 关联的数据边的时间有效性和图模拟条件, 将匹配的时序边 (v,v') 分发到窗口实例 w_i, 即可以生成快照图 G_i 上的图模拟匹配 S_i.

Sim 部分的核心任务是依据定义 3, 对数据图中所有与模式图顶点 $u\in V_P$ 具有相同标签的顶点 $v\in V$ 进行匹配, 并进一步检测其间的模拟关系, 以确保模式约束得到满足. 假设, $sim(u)$ 是模式图顶点 u 的候选匹配集, 包含数据图中所有与模式图顶点 u 标签相同的数据顶点 v, 即 $sim(u)=\{v|L(v)=L(u),v\in V\}$. $post(u)$ 和 $post(v)$

分别为模式图顶点u和数据图顶点v的后继顶点集合, 即$post(u)=\{u'|(u,u')\in E_P\}$, $post(v)=\{v'|(v,v')\in E\}$.

对于模式图顶点u的任意候选顶点$v\in sim(u)$, 顶点u的任意后继顶点u', 如果顶点v不存在后继顶点$v'\in post(v)$可以使得$(u',v')\in R$, 则数据顶点v应该从模式顶点u的候选匹配集$sim(u)$中被删掉.

Assign 部分的核心是根据顶点对(u,v)的模拟条件和时序边(u,v')的有效时间区间来更新顶点对(u,v)的有效匹配时间区间$(u,v).TI$, 并将其更新的有效模拟时间传播到父节点. 如果顶点对(u,v)在窗口w_i中图模拟有效, 则该区间对应的时序边(u,v')将被分发到对应的匹配集S_i.

SimAssign 的详细过程如算法 1 所示. 主要包括以下 4 个部分.

(1) 初始化 (第 1–4 行): SimAssign 算法初始化匹配结果集S和栈$pset$为空; 构建滑动窗口集合U_{SW}, 并计算在窗口SW上的初始匹配结果S_{init}; 将所有初始匹配结果压入用于传播更新匹配对的栈$pset$中.

(2) 初始化状态并加入栈 (第 5 行): 对于每一个模式图中的顶点u, 在初始匹配集中找出可能与之匹配的数据图顶点v. 为每一对顶点对(u,v)初始化状态, 默认时间区间是$[1,n)$. 将这些顶点对压入栈$pset$中, 准备进行传播更新.

(3) 迭代传播匹配状态 (第 6–17 行): 栈$pset$依次弹出每对顶点对(u,v), 准备对顶点对(u,v)的有效模拟时间进行更新. 更新过程应当根据图模拟定义, 遍历模式图顶点u的后继顶点u', 验证与相关数据顶点v的后继v'的有效匹配时间, 基于此得到时序边(v,v')在图模拟查询期间的有效时间区间. 最后根据时序边的有效时间区间将顶点对(u,v)的匹配时间完成更新, 即更新当前的$newTI$. 最后, 如果新时间区间和原来的时间区间不同, 说明匹配时间区间发生变化, 需要更新当前(u,v)的区间 (第 9–15 行). 为了保持图结构的一致性, 向前传播变化, 找出影响的顶点对(u',v')并将其压入栈中进行更新 (第 16–17 行). 通过迭代以上过程, 可完成匹配顶点对(u,v)的有效时间区间更新.

(4) 输出结果 (第 18–20 行): 遍历每个状态, 根据顶点对(u,v)的有效时间区间将匹配的时序边(v,v')分发到对应的窗口实例w_i中, 以保证匹配结果拓扑结构的完整性. 该过程要求数据边(v,v')能在窗口w_i中存在 (第 18, 19 行). 最后返回在滑动窗口SW上的所有快照匹配结果S (第 20 行).

算法 1. 时序图模式匹配算法 SimAssign

输入: 模式图P, 时序图G, 滑动窗口SW.
输出: 匹配结果$S=(S_1,S_2,\cdots,S_n)$.

```
1. 初始化快照匹配结果S=(S1,S2,…,Sn):=∅;
2. 初始化栈pset:=∅;
3. 初始化合并图USW=G1∪G2∪…∪Gn;
4. 初始匹配结果Sinit=静态图模拟(P,USW);
5. for 每对顶点对(u,v)∈Sinit do 推入栈pset中;
6. while 栈pset不为空 do
7.    从栈pset弹出顶点对(u,v);
8.    有效区间TI=(u,v).TI;
9.    for 模式顶点u的每个后继顶点u' do
10.      有效区间newTI=∅;
11.      for 满足(u',v')∈Sinit的每条时序边(v,v') do
12.         newTI=newTI∪((v,v').TI∩(u',v').TI);
13.      更新TI=newTI∩TI;
14.   if 更新后的TI不等于(u,v).TI do
15.      更新(u,v).TI为有效区间TI;
16.      for 数据顶点v的每个前驱w, 并且顶点w与模式
           顶点u的前驱w'满足(w',w)∈Sinit do
17.         if (w',w)不存在栈pset中 do
              将(w',w)加入到栈pset中;
18. for 顶点对(v,v')满足: 时间区间(u,v).TI∩(v,v').TI 与窗口wi交集不
为空 do
19.   将顶点对(v,v')加入到快照Gi的结果集Si中;
20. return 匹配结果集合 S;
```

如图 5 所示, 如果给定滑动窗口SW的一系列实例$w_1=[0,3)$、$w_2=[3,6)$、$w_3=[6,9)$、$w_4=[9,12)$、$w_5=[12,15)$. 时序图G的初始匹配结果为$S_{\text{init}}=\{(a,v_1),(a,v_6),(b,v_2),(b,v_7),(c,v_4),(d,v_3)\}$. 将初始匹配结果$S_{\text{init}}$推入栈$pset$中依次进行访问. 第 1 对为$(a,v_1)$. 根据图 5 中的匹配信息可知, 模式图顶点$a$的后继顶点为$c$. 通过检查数据图顶点$v_1$的相关时序边$(v_1,v_4)$时间区间, 及顶点对$(c,v_4)$的有效时间区间可知, 时序图顶点$v_1$可通过时序边$(v_1,v_4)$在时间区间$[2,9)$模拟模式图顶点$a$, 故将顶点对$(a,v_1)$的有效时间区间更新为$[2,9)$ (第 9–15 行). 此时, (a,v_1)的时间区间发生了变化, 但由于其父节点(b,v_2)已在栈$pset$中, 所以结束本次计算. 同理可将顶点对(a,v_6)、(b,v_2)、(b,v_7)的有效时间区间依次更新为$[5,12)$、$[2,8)$、$[5,12)$.

之后出栈的元素为(c,v_4), 根据上述计算过程, 可将(c,v_4)的有效时间区间更新为$[2,12)$. 但此时, (c,v_4)的父节点(a,v_1)、(a,v_6)不在栈$pset$中, 故推入栈中进行时间信息更新 (第 16, 17 行). 当栈$pset$为空时, 完成所有的匹配时间信息更新. 如图 5 分发过程所示, 数据边

(v_2,v_1)被分发到滑动窗口w_1、w_2、w_3; 边(v_7,v_6)被分发到w_2、w_3、w_4窗口; 边(v_4,v_2)被分发到窗口w_1、w_2; 边(v_4,v_7)被分发到w_2、w_3、w_4窗口; 边(v_4,v_3)被分发到w_1、w_2、w_3、w_4窗口; 边(v_1,v_4)被分发到窗口w_1、w_2、w_3; 边(v_6,v_4)被分发到w_2、w_3、w_4窗口. 至此完成所有匹配.

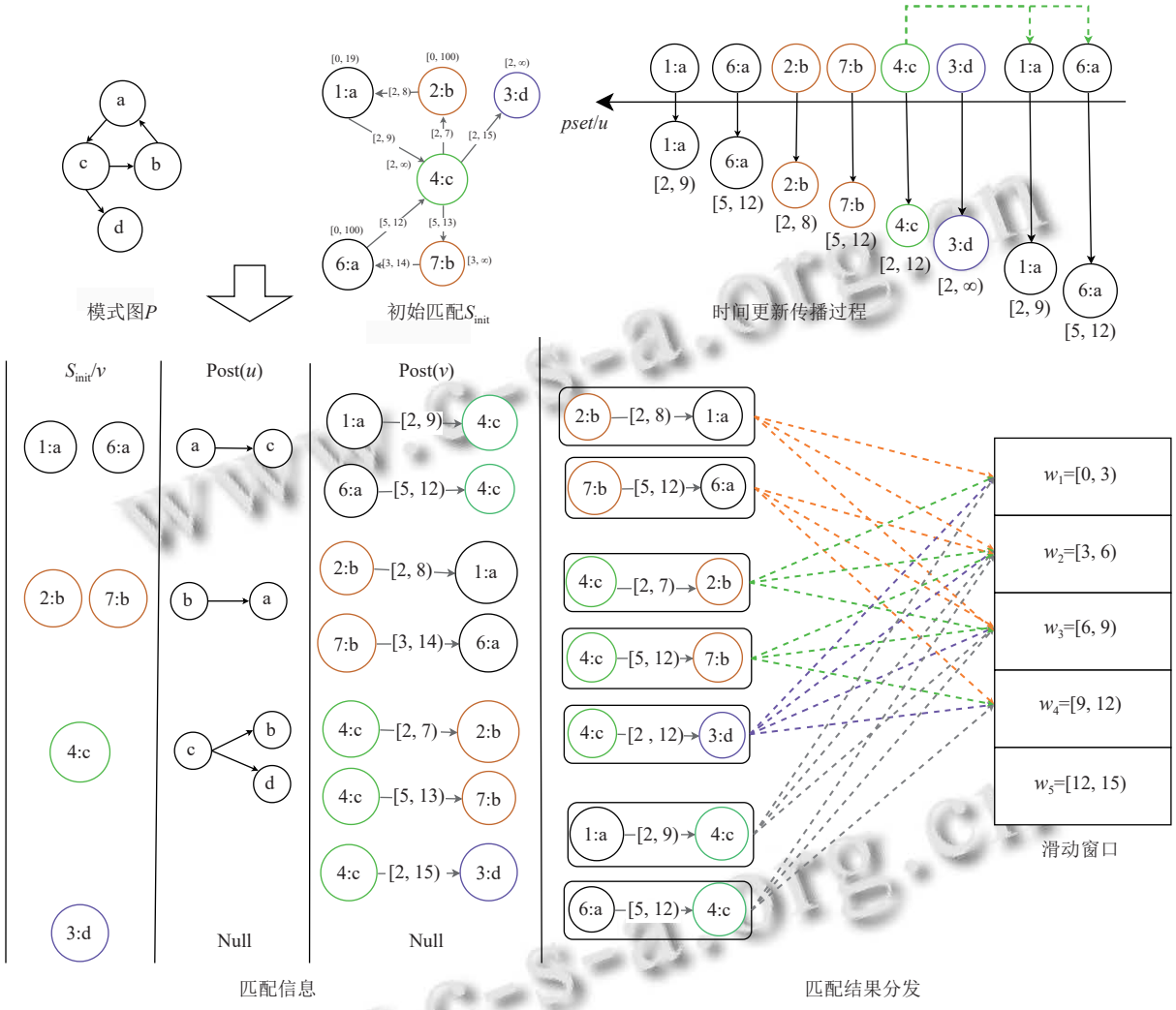

图 5 算法 SimAssign 实例

3.2 查询优化策略

SimAssign 算法将所有快照图$(G_1,G_2,\cdots,G_n)$分割成合适大小的快照图段, 并将快照图段合并为一个联合图进行图模拟, 段的数量将影响总体执行时间. 本节将通过寻找一个最优的合并计划将所有快照分割成合适大小的快照图段, 从而达到优化总体执行时间的目的. $\boldsymbol{P}$表示整个计划空间, 大小为2^n.

定义 5. 最小代价时序图匹配. 给定模式图P和时序图G的快照序列$(G_1,G_2,\cdots,G_n)$, 最小代价时序图匹配问题是指寻找最优的快照分割段$S_{i,j}$使得总的图模拟代价最小, 即计算$\min_{P\in\boldsymbol{P}}\sum_{S_{i,j}\in S} cost(P,S_{i,j})$. 其中, $S_{i,j}=G_i\cup\cdots\cup G_j$为快照$G_i$到快照$G_j$的合并图, $cost(P,S_{i,j})$指的是在段$S_{i,j}$上执行图模拟的代价.

3.2.1 代价模型

估计图模拟的代价$cost(P,S_{i,j})$, 需要统计每个阶段基本操作的执行次数, 然后通过曲线拟合来计算每个基本操作的对应系数. 使用代价系数需要评估每个基本操作的执行频率, 这取决于时序图的统计信息和查询选择率.

● 统计信息. (1) 基本信息: 时序图G中顶点和边的数量, 以及顶点的平均出度和入度等; (2) 匹配信息: 与模式图P中每个标签匹配的顶点的数量、平均出入度,

以及与模式图中每个标签对匹配的边的数量; (3) 时序信息: 所有边的时间区间长度.

● 选择率估计. 查询模式的选择性直接影响执行时间. 在进行 Assign 时, 选择率较高的顶点会导致更大的初始匹配, 即在时间有效性期间需要检查更多的顶点对. 对于模式图边(u,u'), 根据统计数据可得到满足顶点u谓词的顶点的基数N, 匹配(u,u')标签对的边的基数M. 对于匹配标签a的顶点u, 其匹配标签的出边顶点u'的数量服从 0~x 的均匀分布, 均值为$\frac{M}{N}$, $x=k\times\frac{M}{N}$, $k=2$. 模式图边的选择率$sel(u,u')=1-\frac{1}{x+1}$是指匹配模式图顶点$u$的数据顶点具有与$u'$匹配的子顶点的概率. 模式图顶点的选择率$sel(u)=prob(u)\times\prod_{(u,u')\in E_P}sel(u,u')$是指匹配顶点$u$的数据顶点的比例. 其中, $prob(u)$指的是数据图顶点v与模式图顶点u标签相同的概率.

3.2.2 基于代价的动态规划算法

动态规划算法 MCDP 目的是在给定一系列快照和它们代价的情况下, 找到一种总成本最小的快照分割方案. 其核心思想是使用一个一维数组$f(i)$来记录前i个快照的最小总成本, 依赖子问题的最优解来构建整体最优解. 使用反证法可以证明 MCDP 算法可以找出最优子结构证明. 由于篇幅有限, 略过具体证明过程.

MCDP 由算法 2 详细描述, 其输入是快照数量和一组成本, 输出是最优执行计划, 即当在时序图$G_1\cdots G_i$上处理查询的代价最小时, 快照被分割的段的数量. 该算法主要包括两部分.

(1) 状态定义: $f(i)$表示将前i个快照进行划分的最小总代价, 初始值设置为$f(0)=0$, 其他均为$+\infty$.

(2) 状态转移: 对于每个$i\in[1,n]$, 尝试所有可能的分段起点$k\in[1,i)$, 也就是最后一段中最后一个快照 ID, 更新最小代价$f(i)=\min_{0<k\leqslant i}\{f(k-1)+cost(k,i)\}$, 并记录此次选择用于回溯出最优执行方案$p$. 其中, $cost(k,i)$指的是在段$S_{k,i}$上图模拟的执行代价.

算法 2. 最小代价优化算法 MCDP

输入: 快照数量n, 一组代价$cost$.
输出: 最优执行计划p.

```
1. 给定一个数组f, 初始化f[0]=0;
2. for i∈1,⋯,n do f[i]均初始化为+∞;
3. for i∈1,⋯,n do
4.    for k∈1,⋯,i do
5.       if f[k−1]+cost(k,i)<f[i] do
6.          更新最小代价f[i]=f[k−1]+cost(k,i);
7.          记录f[i]的更新;
8. 通过从f(n)检索记录生成最佳执行计划p;
9. return p;
```

4 实验结果与分析

4.1 实验设置

本文算法均使用 Python 实现, 实验均在一台配置为 Intel Xeon X555 2.67 GHz CPU 和 24 GB RAM 的单机上运行, 操作系统为 Ubuntu 14.04. 实验数据分为合成数据集 (Synthetic) 和真实数据集 (WikiTalk) 两组, 实验中使用了无环图 (non-cyclic) 和有环图 (cyclic) 两种图模式查询. 合成数据集 (D1–D5) 是由 EGG[40]生成, 数据集 (D1–D5) 的演化速率从 1%–20% 不等. WikiTalk[41]包含有 7 833 140 个事件, 描述了 2 278 天内维基百科讨论页面上用户之间的交互 (修改历史). 实验过程基于 WikiTalk 原始数据集, 设计了 5 种滑动窗口配置, 得到了相应的 5 个实验数据视图 (W1–W5). 窗口的起止时间为整个数据集的有效时间, 窗口长度分别为 360、180、90、60 和 30 天, 步长为 1 天.

4.2 优化策略 MCDP

本组实验在真实数据集 WikiTalk 上对动态规划算法 MCDP 生成的计划和 SimAssign 算法的一组执行计划进行了比较. 执行计划将一系列快照均匀分割成不同大小 (记为 K) 的段. 图 6 展示了模式图在不含环和含环情况下, SimAssign 算法采用不同计划的执行时间, OPT 为 MCDP 算法给出的最优划分方案. 如图 6 所示, 随着 K 从 20 减至 1, 执行时间随 K 值减小急剧下降, 快速趋于平稳. OPT 方案的执行时间总是低于其他方案的执行时间, 这表明 MCDP 算法在绝大多数情况下总是能给出最优执行计划, 这一实验结果验证了查询优化策略的有效性.

本组还研究了分段数量与图的演化率之间的关系. 如图 7 所示, 最优执行计划 OPT 中的分段数随着窗口长度的减少而增加. 最优执行计划 OPT 中段的数量随着演化速率的增加而增加. 这是因为当图频繁演化时, SimAssign 算法需要更多的迭代才能在相同计划下收敛, 从而导致 MCDP 算法倾向于找到由更短的段组成的优化计划. 对比图 7(a) 和图 7(b), 模式图中环的存在会影响最优执行计划 OPT 中段的数量, 但是不会影响分段数量与时间窗口长度之间的增长趋势.

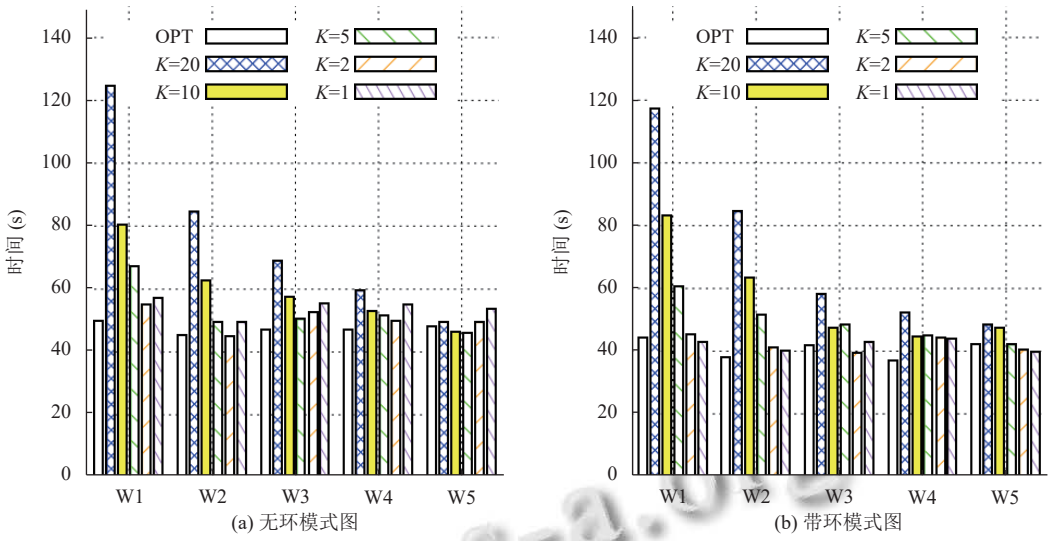

图 6 SimAssign 不同执行计划时间比较

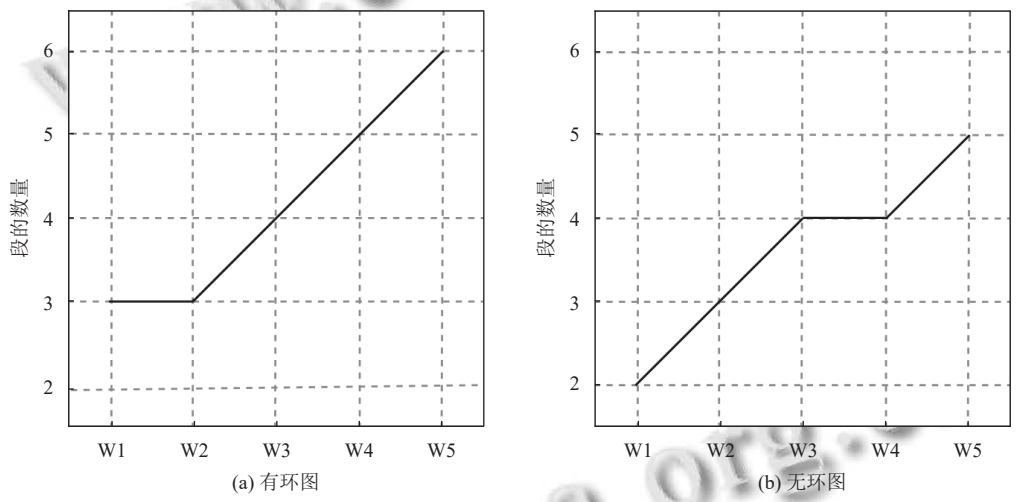

图 7 最优执行计划中的段的数量

4.3 SimAssign 算法的性能

本组实验在合成数据集和 WikiTalk 上比较了 SimAssign 算法、静态图模拟 (StaticSim) 和增量图模拟 (IncSim) 的性能. 图 8 展示了 3 种方法在不同模式图和数据集上的表现.

总的来说, 通过图 8 所有实验结果可以看到, SimAssign 算法无论在何种查询模式图的情况下, 几乎总是显著优于 IncSim 和 StaticSim, 在一些情况下甚至至少比增量法快 4 倍, 如图 8(d) 中的 W1 数据集所示. 而 IncSim 算法的性能介于 SimAssign 和 StaticSim 之间, 只在极少数情况下可能稍优于 SimAssign, 这一点在图 8(a) 的 D1 数据集中得以体现.

对于增量法 IncSim 和静态法 StaticSim 来说, 在无环模式图下, 增量法 IncSim 相较于静态法 StaticSim 具有明显优势; 而当模式图中存在环结构时 (如图 8(b) 和 (d)), IncSim 相对于 StaticSim 的优势并不明显. 这一结果证实了当模式图包含环时, 增量法的效率不够高这一现象.

在合成数据集上 (图 8(a) 和 (b)), 本组实验研究了图数据的演化率和查询模式图是否含环对算法性能的影响. 通过对比相同数据集下不同的查询模式图, 例如不同模式图在合成数据集 D1 进行图模拟查询时 (图 8(a) 和 (b) 的 D1 数据集), 可以看出查询模式图是否含环对增量法 IncSim 和算法 SimAssign 都有影响, 但显然对 SimAssign 的影响远小于增量法 IncSim. 通过对比同一查询模式下, 图在不同演化率下 3 种算法的性能表现,

如图 8(a) 或 (b), 可以看出随着图演化率的升高, 增量法 IncSim 的执行时间显著升高, 而对 SimAssign 的执行时间影响较小. 对于静态法 StaticSim 来说, 这两个因素对其几乎没有任何影响.

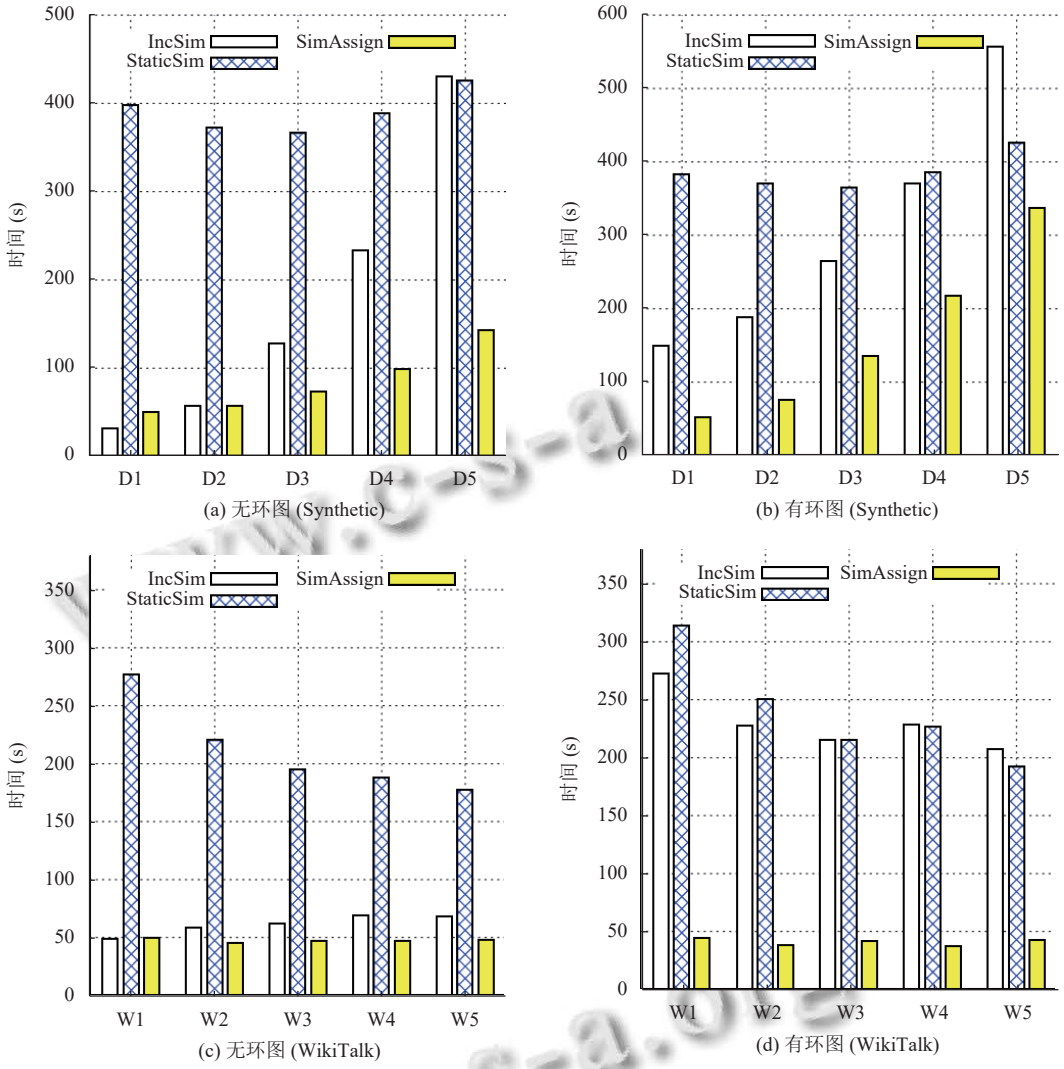

图 8 两种模式情况下 3 种算法在两种数据集上的性能对比

在数据集 WikiTalk 上 (图 8(c) 和 (d)), 本组实验比较了滑动窗口和查询模式图是否含环对 3 种算法的影响. 通过图 8(c) 或 (d) 可以观察到, 在统一查询模式下, 静态法 StaticSim、增量法 IncSim 及 SimAssign 分别与窗口长度成正相关、负相关及无明显相关性. 通过对比图 8(c) 和 (d) 可以看出, 在统一滑动窗口长度时, 模式图含环时对增量法 IncSim 的影响最为明显, 而对静态法 StaticSim 以及 SimAssign 的算法性能几乎没有影响. 这一结果再次说明了模式图含环时增量法的局限性.

4.4 成本模型

虽然图演化速度的范围难以精确界定, 也就是难以确定在什么情况下 SimAssign 算法的执行时间比增量法 IncSim 小, 但是通过借助成本模型可以预测在不同图演化率、不同滑动窗口长度、模式图是否含环等复杂情况下, 不同图模拟方法的执行时间, 帮助用户找到性能最优的解决方案.

图 9 展示了成本模型预测的 3 种图模拟方法的运行时间. 在图 9(a) 中, 代价模型预测增量法 IncSim 在 D1 上优于 SimAssign, 这与图 8(a) 中的实际实验结果一致. 在图 8(a) 中可见, 当数据图的演化速率高于 D1 数据集时, SimAssign 的性能将显著优于增量法 IncSim, 这一趋势也被成本模型准确地预测到了. 在其他情况下的 3 种算法的性能对比结果, 通过对比相应的实验结果, 可以验证成本模型准确预测了所有结果.

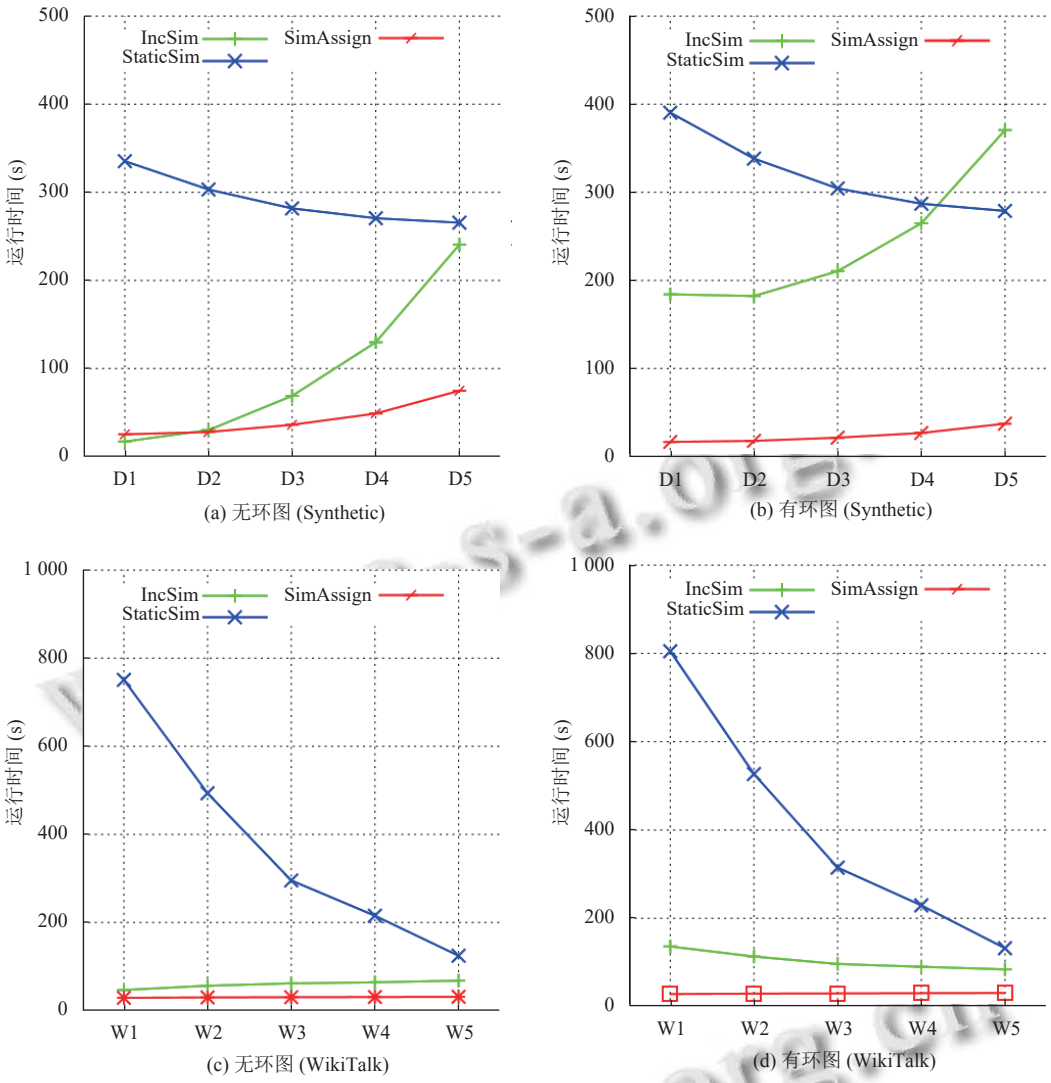

图 9 成本模型预测的 3 种图模拟方法的运行时间

5 结论

本文研究了时序图模拟查询优化问题, 提出了一种基于时间窗口的动态匹配方法 SimAssign. 该方法综合静态图模拟和增量式图模拟的优点, 通过分析时序边的有效时间区间, 将匹配结果动态分配到相应时间窗口, 显著提升了查询效率. 此外, 本文还在 SimAssign 算法的基础上构建了一个基于通用代价模型的动态规划优化框架. 实验结果表明, 与静态法和增量法相比, SimAssign 图模拟算法及动态规划 MCDP 优化框架在多数实验场景下均表现出更优的性能, 尤其在时序图演化速度较高或模式图存在环结构时, 其性能优势更为显著. 而本文提出的统一成本模型准确地预测了 3 种实验算法的性能表现.

参考文献

1 Khurana U, Deshpande A. Historical graph management. Encyclopedia of Big Data Technologies. Cham: Springer International Publishing, 2018: 1–7. [doi: 10.1007/978-3-319-63962-8_210-1]

2 Massri M, Miklos Z, Raipin P, *et al*. Clock-G: Temporal graph management system. Transactions on Large-scale Data- and Knowledge-centered Systems LIV: Special Issue on Data Management—Principles, Technologies, and Applications. Berlin, Heidelberg: Springer, 2023: 1–40. [doi: 10.1007/978-3-662-68014-8_1]

3 Facebook users, stats, data & trends for 2025. https://datareportal.com/essential-facebook-stats. [2025-03-12].

4 Lou YK, Wang CK, Gu TK, *et al*. Time-topology analysis on temporal graphs. The VLDB Journal, 2023, 32(4): 815–843.

[doi: 10.1007/s00778-022-00772-y]

5 Li JT, Qi JP, Huang YL, *et al*. MoTTo: Scalable motif counting with time-aware topology constraint for large-scale temporal graphs. Proceedings of the 33rd ACM International Conference on Information and Knowledge Management. Boise: ACM, 2024: 1195–1204. [doi: 10.1145/3627673.3679694]

6 Holme P, Saramäki J. Temporal networks. Physics Reports, 2012, 519(3): 97–125. [doi: 10.1016/j.physrep.2012.03.001]

7 Bollen E, Kuijpers B, Soliani V, *et al*. Temporal paths in real-world sensor networks. ISPRS International Journal of Geo-Information, 2024, 13(2): 36. [doi: 10.3390/ijgi13020036]

8 Martens W, Niewerth M, Popp T, *et al*. Compact path representations for graph database pattern matching. Proceedings of the 40th IEEE International Conference on Data Engineering Workshops (ICDEW). Utrecht: IEEE, 2024: 379–380. [doi: 10.1109/ICDEW61823.2024.00059]

9 Ullmann JR. An algorithm for subgraph isomorphism. Journal of the ACM, 1976, 23(1): 31–42. [doi: 10.1145/321921.321925]

10 Cordella LP, Foggia P, Sansone C, *et al*. A (sub)graph isomorphism algorithm for matching large graphs. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2004, 26(10): 1367–1372. [doi: 10.1109/TPAMI.2004.75]

11 Shang HC, Zhang Y, Lin XM, *et al*. Taming verification hardness: An efficient algorithm for testing subgraph isomorphism. Proceedings of the VLDB Endowment, 2008, 1(1): 364–375. [doi: 10.14778/1453856.1453899]

12 He HH, Singh AK. Graphs-at-a-time: Query language and access methods for graph databases. Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data. Vancouver: ACM, 2008: 405–418. [doi: 10.1145/1376616.1376660]

13 Zhang SJ, Li SR, Yang J. GADDI: Distance index based subgraph matching in biological networks. Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology. Saint Petersburg: ACM, 2009: 192–203. [doi: 10.1145/1516360.1516384]

14 Zhao PX, Han JW. On graph query optimization in large networks. Proceedings of the VLDB Endowment, 2010, 3(1–2): 340–351. [doi: 10.14778/1920841.1920887]

15 Henzinger MR, Henzinger TA, Kopke PW. Computing simulations on finite and infinite graphs. Proceedings of the 36th IEEE Annual Foundations of Computer Science. Milwaukee: IEEE, 1995: 453–462. [doi: 10.1109/SFCS.1995.492576]

16 Neo4j graph database & analytics graph database management system. https://neo4j.com/. [2025-03-12].

17 Khurana U, Deshpande A. Efficient snapshot retrieval over historical graph data. Proceedings of the 29th International Conference on Data Engineering (ICDE). Brisbane, QLD: IEEE, 2013: 997–1008. [doi: 10.1109/ICDE.2013.6544892]

18 Macko P, Marathe VJ, Margo DW, *et al*. LLAMA: Efficient graph analytics using large multiversioned arrays. Proceedings of the 31st IEEE International Conference on Data Engineering. Seoul: IEEE, 2015: 363–374. [doi: 10.1109/ICDE.2015.7113298]

19 Labouseur AG, Birnbaum J, Olsen PW, *et al*. The G* graph database: Efficiently managing large distributed dynamic graphs. Distributed and Parallel Databases, 2015, 33(4): 479–514. [doi: 10.1007/s10619-014-7140-3]

20 Miao YS, Han WT, Li KW, *et al*. ImmortalGraph: A system for storage and analysis of temporal graphs. ACM Transactions on Storage, 2015, 11(3): 14. [doi: 10.1145/2700302]

21 Then M, Kersten T, Günnemann S, *et al*. Automatic algorithm transformation for efficient multi-snapshot analytics on temporal graphs. Proceedings of the VLDB Endowment, 2017, 10(8): 877–888. [doi: 10.14778/3090163.3090166]

22 Steer B, Cuadrado F, Clegg R. Raphtory: Streaming analysis of distributed temporal graphs. Future Generation Computer Systems, 2020, 102: 453–464. [doi: 10.1016/j.future.2019.08.022]

23 张天明, 徐一恒, 蔡鑫伟, 等. 时态图最短路径查询方法.计算机研究与发展, 2022, 59 (2): 362–375. [doi: 10.7544/issn1000-1239.20210893]

24 Byun J, Woo S, Kim D. ChronoGraph: Enabling temporal graph traversals for efficient information diffusion analysis over time. IEEE Transactions on Knowledge and Data Engineering, 2020, 32(3): 424–437. [doi: 10.1109/TKDE.2019.2891565]

25 Johnson T, Kanza Y, Lakshmanan LVS, *et al*. Nepal: A path query language for communication networks. Proceedings of the 1st ACM SIGMOD Workshop on Network Data Analytics. San Francisco: ACM, 2016: 1–8. [doi: 10.1145/2980523.2980530]

26 Debrouvier A, Parodi E, Perazzo M, *et al*. A model and query language for temporal graph databases. The VLDB Journal, 2021, 30(5): 825–858. [doi: 10.1007/s00778-021-00675-4]

27 Arenas M, Bahamondes P, Aghasadeghi A, *et al*. Temporal regular path queries. Proceedings of the 38th IEEE International Conference on Data Engineering (ICDE). Kuala Lumpur: IEEE, 2022: 2412–2425. [doi: 10.1109/ICDE53745.2022.00226]

28 Gao YJ, Zhang TM, Qiu LS, *et al*. Time-respecting flow graph pattern matching on temporal graphs. IEEE Transactions on Knowledge and Data Engineering, 2021, 33(10): 3453–3467. [doi: 10.1109/TKDE.2020.2968901]

29 Zhang LX, Gao JL. Incremental graph pattern matching algorithm for big graph data. Scientific Programming, 2018, 2018(1): 6749561. [doi: 10.1155/2018/6749561]

30 Fan WF, Li JZ, Ma S, *et al*. Graph pattern matching: From intractable to polynomial time. Proceedings of the VLDB Endowment, 2010, 3(1-2): 264–275. [doi: 10.14778/1920841.1920878]

31 Chen L, Wang CL. Continuous subgraph pattern search over certain and uncertain graph streams. IEEE Transactions on Knowledge and Data Engineering, 2010, 22(8): 1093–1109. [doi: 10.1109/TKDE.2010.67]

32 Fan WF, Li JZ, Luo JZ, *et al*. Incremental graph pattern matching. Proceedings of the 2011 ACM SIGMOD International Conference on Management of data. Athens: ACM, 2011: 925–936. [doi: 10.1145/1989323.1989420]

33 Zhang TM, Cai XW, Chen L, *et al*. Towards efficient simulation-based constrained temporal graph pattern matching. World Wide Web, 2024, 27(3): 22. [doi: 10.1007/s11280-024-01259-2]

34 Ma YL, Yuan Y, Liu M, *et al*. Graph simulation on large scale temporal graphs. GeoInformatica, 2020, 24(1): 199–220. [doi: 10.1007/s10707-019-00381-y]

35 Bouhenni S, Yahiaoui S, Nouali-Taboudjemat N, *et al*. Distributed graph pattern matching via bounded dual simulation. Information Sciences, 2022, 610: 549–570. [doi: 10.1016/j.ins.2022.08.038]

36 Zhang TM, Gao YJ, Qiu LS, *et al*. Distributed time-respecting flow graph pattern matching on temporal graphs. World Wide Web, 2020, 23(1): 609–630. [doi: 10.1007/s11280-019-00674-0]

37 Du X, Tao ZC, Li L. Dynamic graph pattern matching in medical knowledge graphs. Proceedings of the 4th International Conference on Data Intelligence and Security (ICDIS). Shenzhen: IEEE, 2022: 306–313. [doi: 10.1109/ICDIS55630.2022.00054]

38 Jiang GX, Zhao YJ, Li YC, *et al*. Wings: Efficient online multiple graph pattern matching. Proceedings of the 40th International Conference on Data Engineering (ICDE). Utrecht: IEEE, 2024: 3013–3027. [doi: 10.1109/ICDE60146.2024.00260]

39 Sun GH, Liu GF, Wang Y, *et al*. Incremental graph pattern based node matching with multiple updates. IEEE Transactions on Knowledge and Data Engineering, 2021, 33(4): 1585–1600. [doi: 10.1109/TKDE.2019.2942294]

40 Alami K, Ciucanu R, Mephu Nguifo E. EGG: A framework for generating evolving RDF Graphs. Proceedings of the 16th International Semantic Web Conference (ISWC 2017). Vienna: ISWC, 2017: 425–440. [doi: 10.1007/978-3-319-68204-4_39]

41 SNAP: Network datasets: Wikipedia talk network. https://snap.stanford.edu/data/wiki-Talk.html. (2008-01-01) [2025-03-12].

(校对责编: 王欣欣)