**图的功能依赖性**

文飞凡1 *，* 2 映辉吴3 颈脖旭1 *，* 2

1爱丁堡大学 2北京航空航天大学RCBD和NLSDE实验室 3华盛顿州立大学

{wenfei @ inf，jingbo.xu @}。ed.ac.uk，yinghui @ eecs.wsu.edu

# 摘要

我们为图提出了一类函数依赖关系，称为GFD 。GFD捕获实体的属性值依赖关系和拓扑结构，并作为特殊情况包含条件功能依赖关系（CFD ）。我们表明，GFD的可满足性和含意性问题分别是coNP -complete和 NP -complete ，并不比其CFD 同行差。我们还表明，GFD的验证问题是coNP -complete 。尽管存在棘手的问题，我们还是开发了并行可扩展算法来捕获违反GFD的行为在大型图中。使用现实生活和综合数据，我们通过实验验证了GFD提供了一种有效的方法来检测知识和社会图谱中的不一致之处。

# 关键字词

功能依赖性；图; 含义; 可满足性 验证

# 引言

对于关系数据，已经很好地研究了数据依存关系。特别是，在每本数据库教科书中都可以找到我们熟悉的功能依赖关系（FD ），并且已经将其扩展为XML [9]。他们的修订，例如条件功能依赖项（CFD ）[16]，已证明可以有效地捕获关系中的语义不一致[15]。

在图表（一种常见的数据源）中也很明显需要FD 。与关系数据库不同，现实生活中的图形通常不带有架构。FD指定数据语义的基本部分，因此对于图形尤为重要。此外，（1）FD帮助我们检测知识库中的不一致性[45]，需要将其识别为对依赖性的违反[15]。（2）对于社交网络，FD可以帮助我们捕获垃圾邮件和管理博客[11]。

**示例1：**考虑以下示例，这些示例来自现实生活中的知识库和社会图表。

只要不为牟利或商业利益而制作或分发副本，且副本带有本声明和全文引用，则允许免费或免费制作全部或部分本作品的数字或纸质副本供个人或教室使用。第一页。必须尊重非ACM拥有的本作品组件的版权。允许使用信用摘要。要以其他方式复制或重新发布以发布在服务器上或重新分发到列表，需要事先获得特定的许可和/或费用。从Permissions@acm.org请求权限。

*SIGMOD'16，2016年6月26日至7月1日，美国加利福尼亚州旧金山*

⃝c 2016 ACM。国际标准书号（ISBN）978-1-4503-3531-7 / 16/06 ... 15.00美元

DOI：http ： //dx.doi.org/10.1145/2882903.2915232

1. *知识库*，其中不一致之处很普遍[45]：

◦飞行A123有两个入口使用同一个出发时间14:50和到达时间22:35，但一个是从巴黎到纽约，而其他从巴黎到新加坡[40]。

◦堪培拉和墨尔本都被标记为澳大利亚的首都[13]。

◦标记表明，尽管它们的机翼结构已进化，但所有鸟类都可以飞翔，而企鹅是鸟类[26]。

我们将看到，在具有图形结构的实体上定义的FD可以轻松捕获所有这些不一致之处。

1. *社会图*。当博客*ž*有照片 *Ÿ*被提出，社交网络公司定义了一个状态*X*与附件*ÿ*。它要求注释*X.*文本的*X*必须描述不符*Y.*递减的*ÿ*。那是，

◦博客：如果*Z的*状态为*X*，*Z的*照片为*Y*，如果*X的*附件为*Y*，则*X.* text = *Y.* desc 。

这本质上是图结构数据的FD 。

功能依赖性在捕获垃圾邮件方面也很有用。

◦假帐户[11]：如果确认帐户*x* '为假，则帐户*x*和*x* '都类似于博客*P* 1 *，...，P k*，*x*发布博客*y*，*x* '发布*y* '以及是否*y*和*y* '拥有特定的关键字*c*，则*x*也被标识为假帐户。

识别伪造帐户的此规则是图形上的FD 。✷

但是，无论多么重要，从公式到经典问题再到应用，对图的FD 的研究仍处于起步阶段。为图定义FD而不是为关系定义FD更具挑战性，因为现实生活中的图是半结构化的，通常没有架构。此外，对于图形中由顶点表示的实体，FD不仅必须指定实体的属性值之间的规律性，而且还必须指定实体的拓扑结构。

**贡献**。我们研究图的功能依赖性，从它们的基本问题到应用程序。

1. 我们提出了图的一类功能依赖关系，称为GFD （第3节）。与关系FD相对，GFD指定两个约束：（a）根据图形模式的拓扑约束（第2节），以标识定义了依赖的实体，以及（b）CFD 的扩展以指定实体属性值的依赖性。我们发现，GFDs归入文件描述符和差价合约同一实体的属性之间在不同的实体作为特殊情况，并捕获不一致。
2. 我们解决了GFD的两个经典问题（第4节）。对于一组GFD ，我们研究（a）它的可满足性，以确定是否存在一个满足Σ中所有GFD的非空图，以及（b）它的含义，以确定Σ是否需要GFD 。我们表明，GFD的可满足性和隐含问题分别是coNP -complete和 NP complete。结果告诉我们，关于GFD的推理并不比它们的关系对应物（如CFD）难，[ CFD ]也很棘手[16]。
3. 作为GFD的应用之一，我们研究了验证问题，以GFD作为数据质量规则来检测图形中的错误（第5节）。我们表明它是CONP -complete决定图表中是否包含没有违反一组的GFDs 。尽管难以处理，我们仍开发了可*并行扩展的*算法，*即，*当使用更多处理器时*，*它们可以保证花费更少的时间。它们是用于双准则优化问题的2近似算法，以平衡工作量并最小化通信成本（第6节）。这些使检测大型图形中的错误变得可行。
4. 使用现实生活中的合成图，我们通过实验验证了我们的GFD技术的有效性和效率（第7节）。我们发现以下内容。（a）在现实生活中，使用GFD进行不一致检测是可行的。对于一组50个GFD ，使用20个处理器的YAGO [44]复制（重新分区）需要156（326次）秒。（b）我们的算法是并行可扩展的：平均为2 *。*4和3 *。*当处理器从4增加到20时，在片段化和复制的现实图中分别快7倍。（c）我们的优化技术是有效的：它们将性能提高了1.9倍。（d）GFD在现实生活中捕获各种不一致的图形，从而验证了将拓扑约束和值依赖相结合的必要性。

我们主张GFDs是传统的自然延伸文件描述符，通过将图形拓扑结构。GFD为我们提供了图的原始依赖关系，以指定基本语义并检测不一致之处。这项工作还提供了有关GFD推理的第一个复杂性界限。此外，我们开发了第一个并行可伸缩算法以实际使用GFD 。

**相关工作**。我们将相关工作分类如下。

函数依赖*于图表*。FD和CFD的扩展已被研究-

请参考RDF [8、10、12、23、24、49]。[ 8、12、24 ]中FD的定义基于RDF三重嵌入和变量评估的巧合。FD被扩展[49]以通过*例如*路径模式来指定对聚集值的值依赖性。对于差价合约的扩展也是如此[23]。在[10]中提出了一个模式匹配框架，用于RDF和关系之间的转换。它将FD定义为树，其中每个节点表示对应关系中的属性。

我们的工作在以下方面不同于先前的工作。（1）我们定义了具有图模式的GFD ，以表达超出RDF的（属性）图的拓扑约束。（2）GFD捕获由模式识别的图结构实体中的不一致。相反，[ 8、12、24 ]的FD都是基于值的，而不管哪个实体携带值，并且[24]的推理技术是基于RDF数据的关系编码的。此外，这些FD不能像CFD中那样用常量（语义值绑定）来表示等式，*例如x。*city =“ Edi”，而GFD包含CFD 。该函数依赖[10]中的定义为树，并采用关系模式。它们不支持一般的拓扑约束。类似的[23,49]。（3）我们为GFD分析提供了复杂性界限，并为图形中的错误检测提供了并行可扩展算法，而先前的工作并未对此进行研究。

更加接近这项工作的是图形[14]的键上的[14]，在以下方面与GFD不同。（1）键被简单地定义为图形模式*Q* [ *x* ]，其中指定的变量*x*表示实体。与此相反，GFDs具有形式（*Q* [ˉ *X* ]*中，X* → *ÿ*），其中ˉ *X*是变量列表，以及*X*和*ÿ*与在ˉ常量和变量平等原子的连词*X*。GFDs不能表示为键，就这样的关系文件描述符 不能表示为键。此外，[14]的键被递归地定义，以确定实体，而GFDs是常规的扩展的FD ，并且不递归定义。（2）键在RDF三元组（*s，p，o*）上定义，而GFD在属性图（*例如*社交网络）上定义。（3）键是根据三个同构映射来解释的：两个从子图到*Q*，一个在两个子图之间。相反，GFD 需要从子图到*Q*的单个同构映射。根据不同的语义，GFD的算法和键完全不同。（4）研究了GFD的可满足性及其含义。这些经典问题并未针对密钥进行研究[14]。

已经研究了*不一致检测*的关系（请参阅

[15]进行调查），以及最近的知识库（链接数据）[23,32,35,37,42,45]。知识库的方法采用规则[23,32,35,42,45]或概率推论[37]。（1）使用数据记录规则[42]提取实体并检测不一致的“事实”。SOFIE [45]通过使用表示为一阶逻辑（FO ）公式的规则以及文本模式，现有本体和语义约束来保持提取事实的一致性。Pellet [35]通过在描述逻辑中使用推理规则来检查不一致之处（*例如，*OWL-DL）。依赖规则用于检测语义Web [32]和RDF [23]中属性值的不一致。BigDansing [28]支持用于修复关系数据的用户定义规则。要清理图结构的实体，它需要将图表示为表，并编码关系查询语言以外的同构函数。（2）[37]的推理方法使用马尔可夫逻辑将FO和概率图形模型相结合，并通过学习和计算结构上的联合概率来检测错误。

我们的工作与以前的工作有所不同，如下所示。（1）通过支持图形模式的拓扑约束，GFD成为（属性）图上的第一批数据质量规则之一，而不仅限于RDF。（2）GFD旨在在复杂性和表现力之间取得平衡。关于GFD的推理比分析FO公式便宜得多。（3）我们提供了GFD的可满足性和含义的复杂性和特征; 这些是通常关于图依赖关系，尤其是关于图的数据质量规则的推理的第一批结果。（4）我们开发了用于错误检测的并行可扩展算法和用于工作负载分配的新策略，而不是使用昂贵的大规模推理和逻辑编程。这些使错误检测在具有可证明的性能保证的大型图形中变得可行，而这是以前的工作无法提供的。

与GFD验证算法相关的*并行*算法是

（1）用于检测分布式数据中的错误的算法[17，18]，以及（2）用于子图枚举，子图同构和SPARQL的算法[5,20,22,25,30,31,39,41,46]。

1. [17，18]的算法（递增）检测错误

（水平或垂直）基于CFD的分区关系。该方法适用于关系，但不能帮助需要子图同构计算的GFD 。确实，我们的算法与[17,18]的算法根本不同。

1. 与这项工作更接近的是用于子图枚举的并行算法[5,30,36,41]。（a）MapReduce算法是通过连接多向联接操作[5]和分解的边联接[36]提出的。该策略对于三角计数是有效的[47]。（b）为减少一般模式的过度局部答案，[30]中的MapReduce解决方案将模式分解为双枝（单边或两个入射边），并采用左深连接策略将多条边连接为星形。为了应对倾斜的节点，需要对高级节点的邻域进行分区，复制和分布。分解策略用于减少MapReduce运算和I / O成本。（c）在[41]中，通过以顶点为中心的编程开发了BSP框架。它采用在线贪婪策略，将部分子图分配给总工作量最少的工作人员，

减少子图实例的混合策略。

1. 针对子图同构[39,46]和SPARQL查询[20,22,25，31]，开发了许多并行算法。Twig分解用于修剪中间结果并减少Trinity内存云中的等待时间[46]。[39]的内存中算法通过（a）在线程之间平均分配部分答案以进行局部扩展，以及（b）将部分答案复制到全局存储以在下一轮中进行平衡分配，从而使回溯过程并行化。针对RDF上的并行SPARQL引入了基于哈希的分区，查询分解和负载平衡策略[20,25]。在[22]中研究了查询分解和计划生成技术，该技术通过复制图形来避免通信成本。在[31]中，通过提取公共子模式，提供了多模式匹配的优化技术。GFD和一般属性图。

这项工作在以下方面不同于先前的工作。（a）分布式图形中的GFD 验证是一个双准则优化问题，其目的是平衡工作量并最大程度地降低通信成本，同时还要结合*断开*图形的子图枚举和碎片图中的依赖性检查而带来的复杂性。它比先前工作中研究的图形查询更具挑战性。（b）我们针对难解决的优化问题引入工作量分配策略，并采用近似界限，而不是分别处理工作量平衡和通信成本最小化[30,41]。（c）我们保证并行可伸缩性，这是现有算法无法保证的。

另一方面，这项工作可以受益于用于快速并行子图匹配和列出的现有技术，*例如*查询分解策略[22,30,46]和多线程内存算法[39]，用于每个工人的本地错误检测。我们采用了[31]的优化技术，并将其他技术整合到GFD 工具中。

1. 在MapReduce算法的通信成本[6]，MapReduce计算/通信成本的限制（MRC [27]，MMC [48]和SGC [38] 方面，还存在表征并行算法有效性的工作。 ），以及递归程序的多项式边缘属性[4]。我们采用并行可扩展性的概念[29]，它在计算和通信成本方面通过在多个处理器上并行化来衡量加速。它不是针对通用并行算法的

ģ 1号 flight1到达ģ 2 “免费奖品”，“免费礼品卡及奖”自由奖“”赢抽奖“”免费ģ 3从到depar吨p 5（博客） p 6（博客）p 7（博客） p 8（博客）

（国家）

DL1 （ID） （Pariscity） （NYCcity） （14:50 22：35time）（时间） 后 澳大利亚

机票网 is\_fake = '真' ACCT 1 is\_fake = '真' ACCT 2 is\_fake = '真' ACCT 3 is\_fake = '假' ACCT 4 资本数 到达

从 到 出发 喜欢 堪培拉（市）

DL1 （id） （巴黎）新加坡（城市） （14:50 22:35时间）（时间） p 1 （博客） p 2 （博客） p 3 （博客）p 4 （博客）

## 图1：图形

仅限于MapReduce。并行可扩展算法可通过添加处理器来保证对大型图形进行缩放。但是，并行可伸缩性对于某些图形计算来说是无法实现的[19]。通过提供这样的算法，我们证明了GFD验证是并行可扩展的。

*静态分析*。在关系上，可满足性和隐含性

阳离子问题被称为是在*Ö*（1）和线性时间的FD ，NP -complete和CONP -complete用于 差价，*Ô*（1）的时间和PSPACE列入依赖（-complete的IND ），分别。验证问题是在PTIME为的FD ，差价和的IND （参见[3,15]）。我们表明，GFDs图形上，验证，可满足和寓意为GFDs是CONP -complete，CONP -complete和 NP-分别完成。如第4节所述，GFD的复杂性来自图形模式之间的相互作用（子图形同构）。它不是从CFD继承的。

# 前提条件

我们首先回顾基本符号。

**图**。我们考虑有向*图G* =（*V，E，L，F A*），带有标记的节点和边，并在其节点上具有属性。这里（1）*V*是节点的有限集合；（2）*ê* ⊆ *V* × *V*是一组边缘; （3）每个节点*v*在*V*（分别为边缘*ê*在*ë*）承载标签*大号*（*v*）（相应地，*大号*（*ê*）），和（4）为每个节点*v*，*˚F甲*（*v*）是一个元组

（*A* 1 = *a* 1 *，...，A n* = *a n*），其中*a i*是常数，*A i*是*v*的*属性*，写为 *vA i* = *a i*，带有*v*的内容，例如属性，关键字，博客和评分，如在社交网络，知识库和属性图中所发现的。

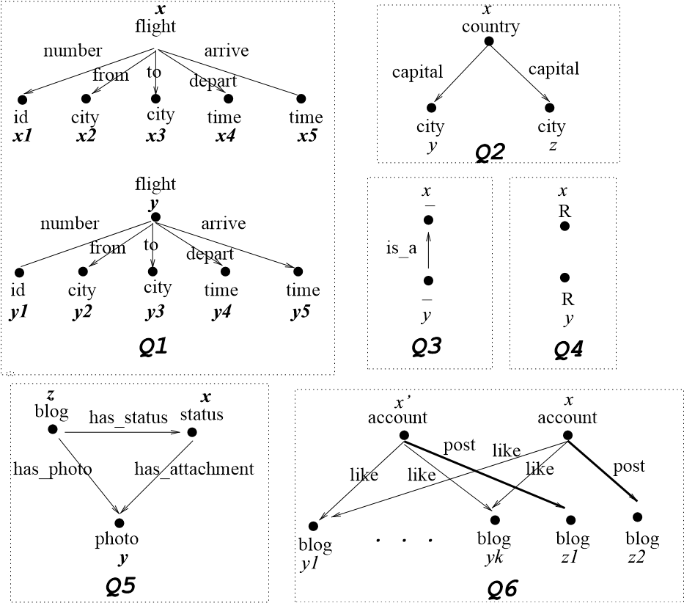
**示例2：**图**2**中描绘了三个图：（a）*G* 1是知识图的片段，其中每个航班实体（*例如，*航班1）具有ID （值val = DL1 ），出发城市（巴黎） ，目的地（NYC ）以及出发和到达时间；每个节点的值都具有属性val （未显示）；（b）*G* 2记录假账户；每个帐户的属性均为假，如果该帐户为假，则为“ true”，否则为“ false”；一个帐户可以发布包含关键字的博客（*例如，*博客*p* 5的属性关键字=“免费奖赏”），并且可能喜欢其他博客；（c）*G* 3描绘了一个国家实体及其首都，其价值带有属性val （未显示）。✷

我们回顾了两个子图的概念。

 *的G子*=（*V，E，L，F甲*），记为*G ^* ' ⊆ *ģ*，如果*V* ' ⊆ *V*，*ê* ' ⊆ *é*，并为每个节点*v* ∈ *V* '，*大号*'（*v*）=*大号*（*v*）和*F A* '（*v*）= *F A*（*v*）; 类似地，对于每个边缘*Ë* ∈ *ë* '， *大号*'（*e*）= *L*（*e*）。

◦我们说*摹*“是*由引起的子*集*V* '的节点，如果*摹*' ⊆ *摹*和*ē* '包括所有的边缘*摹*其终点都在*V* '。

**图形模式**。甲*图形图案*被定义为一个有向图*Q* [ˉ *X* ] =（*V Q*，*ê Q*， *大号Q*，*μ*），其中（1）*V Q*（相应的 *ê Q*）是



## 图2：图形模式

集模式的节点的（相应的边缘），（2）*大号Q*是一个函数，分配标签*大号Q*（*Û*）（分别为*大号Q*（*ë*））给每个图案节点*ü* ∈ *V Q*（相应的边缘*Ë* ∈ *ë Q*），（3）＆OverBar; *X*是的变量的列表，使得其元数|| *X* ˉ|| 等于数字| *V Q* | （4）*μ*是从*x*到*V Q*的双射映射， *即，*它为每个节点分配了一个不同的变量*V*在*V Q中*。对于*X* ∈ *X*＆OverBar;，我们使用*μ*（*X*）和*X*可互换当它是在上下文中明确。特别是，我们允许使用通配符''作为特殊标签。

**示例3：**图2描述了六个图形模式*Q* 1 – *Q* 6：（1）*Q* 1 指定两个航班实体，其中*μ*将*x*映射到航班，将*x* 1 – *x* 4映射到其id ，出发城市，目的地，出发时间 和到达时间，分别; 同样对于*y* 和*y* 1 - *y* 5；（2）*问题*2描绘了一个国家 具有两个不同资本的实体；（3）*Q* 3示出了一个通用的是一种关系，其中两个节点被标记为通配符'“; （4）*Q* 4描绘了关系*R的*两个元组，在图中以顶点表示，标记为*R*；（5）*Q* 5示出了包括 照片*y*的博客 实体*z*，并且*z*由状态*x*描述；（6）*Q* 6指定帐户s *x*，*x*之间的关系'和博客s *y* 1 *，...，y k*和*z* 1 *，z* 2，其中*x*和 *x* '都像*k个*博客，*x* '发布博客*z* 1 和*x*发表*z* 2。✷

**图形模式匹配**。我们采用通过子图同构进行匹配的常规语义。甲*匹配*图案的*Q*在曲线图*G ^*是一个子图的 *ģ*是同构的*Q*，*即，* 存在一个*双射函数h*从 *V Q*到*V* '，使得（1）为每个节点*ü* ∈ *V Q*，*大号Q*（*Û*）= *L* '（*h*（*u*））; 和（2）*e* =（*u，u*'）是在边缘*Q* 当且仅当*Ë* ' =（*H ^*（*Û*）*中，h*（*ú* '））是在边缘*ģ* '和*大号Q*（*ë*）=*大号*'（*ê* '）。特别是，如果*L Q*（*u*）为''，则*L Q*（*u*）= *L* '（*h*（*u*））始终成立，*即*通配符匹配任何标签以指示通用实体，

*例如，*是一个在*Q* 3实施例3的; 边缘标签也是如此。

我们还表示匹配为向量*ħ*（ˉ *X*），由*ħ*（*X*）（*即，H*（*μ*（*X*）））对于所有*X* ∈ *X*＆OverBar;，以相同的顺序作为ˉ *X*。直观地说，ˉ *X*是由被识别的实体的列表*Q*，和*^ h*（ˉ *X*）是这样的实例中*ģ*，对于每个实体的一个节点。

**实施例4：**的匹配*Q* 1 实施例3的在*G ^* 1，图2是*ħ* 1：*X*＆→飞行1，*y*＆→飞行2，*X* 3及→NYC，*ÿ* 3＆→新加坡，并且类似地对于*Q* 1中的其他变量。

当*ķ* = 2，一个匹配的*Q* 6 在*ģ* 2是*ħ* 2：（*X* ' ＆→ ACCT 3*中，x*＆→ ACCT 4，*ÿ* 1＆→ p 3，*ÿ* 2＆→ p 4，*ž* 1＆→ p 7 *，z* 2＆→ p 8）。✷

表1总结了本文的符号。

|  |  |
| --- | --- |
| 符号 | 记号 |
| *G* | 图（*V，E，L，F A*） |
| *Q* [ˉ *X* ] | 图形模式（*V Q*，*E Q*，*L Q*，*μ*） |
| *ϕ，* Σ | GFD *φ* =（*Q* [ˉ *X* ]*中，X* → *ÿ*），Σ是一组GFDs |
| *ħ*（ˉ *X*）| = *X* → *ÿ* | 匹配*ħ*（ˉ *X*）的*Q*满足*X* → *ÿ* |
| Σ *Q* | 嵌入在图案*Q中*的Σ的GFD集合 |
| 威欧（Σ *，G*） | 图 *G中*所有违反GFDΣ的行为 |
| *t*（|Σ| *，* | | *G* |） | 计算Vio的顺序时间（Σ *，G*） |
| *T*（|Σ| *，* | *G* | *，n*） | 使用*n个*处理器的Vio并行时间（Σ *，G*） |
| *宽*（Σ *，G*） | 计算Vio的工作量（Σ *，G*） |
| PV （*ϕ*） | 一个枢轴矢量（ˉ *Z，C*＆OverBar; *Q*的）GFD *φ* |
| *瓦特*=⟨ *v*＆OverBar; *Ž，G ž* ˉ ⟩ | 工作单元（ˉ *v ž*：候选; *G ^ ž* ˉ ：邻居ˉ *v Ž*） |

**表1：符号**

# GFDS：语法和语义

现在，我们*为图*（GFD ）定义*功能依赖性*。

**GFD**。甲GFD *φ*是一对（*Q* [ˉ *X* ]*中，X* → *ÿ* ），其中◦ *Q* [ˉ *X* ]是一个曲线图模式，被称为*图案*的*φ* ; 和◦ *X*和*ÿ*是两个（可能是空的）套ˉ文字的*X*。

这里一个*字面*ˉ的*X*具有任一形式*XA* = *C ^*或*XA* = *YB*，其中*的x，y* ∈ *X*＆OverBar;，*阿*和*乙*分别表示属性（在没有指定*Q*），和*Ç*是常数。我们将*xA* = *c*称为*常量常量*，将*xA* = *yB*称为*变量常量*。

直观地，GFD *ϕ*指定了两个约束：

◦由模式*Q*施加的*拓扑约束*，以及◦由*X* → *Y*指定的*属性依赖关系* 。

回想一下，关系FD *R*（*X* → *Y*）的“范围”是由关系模式*R*指定的：FD仅适用于*R的*实例。与关系数据库不同，图没有模式。在此*Q*指定了GFD的范围，使得依存关系*X* → *Y*仅作用于*Q所*标识的每个子图中顶点的属性。常量字面量*xA* = *c*强制执行与语义相关的常量的绑定，就像CFD一样[16]。

**示例5：**为了捕获示例1中描述的不一致之处，我们定义了图2的*Q* 1 – *Q* 6模式的GFD 。

1. *飞行*：GFD *ϕ* 1 =（ *Q* 1 [ *x，x* 1 - *x* 5 *，y，y* 1 - *y* 5 ]， *X* 1 → *Y* 1），

其中*X* 1是*x* 1 *。*val = *y* 1 *。*val和*Y* 1由*x* 2组成*。*val ＝*y* 2 *。*val和*x* 3 *。*val ＝*y* 3 *。*VAL 。这里VAL为节点的内容的属性。通过*Q* 1， *x* 1，*x* 2和*x* 3 分别表示航班ID，航班*x的*出发城市和目的地；类似地，对于实体*y的y* 1，*y* 2和*y* 3。因此，GFD *ϕ* 1指出，对于所有飞行实体*x*和*y*，如果它们共享相同的飞行ID，则它们必须具有相同的出发城市和目的地。

1. *资本*：GFD *φ* 2 =（ *Q* 2 [ *X，Y，Z* ]，∅→*收率*VAL= *Ž。* VAL）。这是

确保对于*x的*所有国家实体，如果*x*具有两个资本实体*y*和 *z*，则*y*和*z*共享相同的名称。

1. *通用*为：GFD *ϕ* 3 =（ *Q* 3 [ *x，y* ]，∅→ *xA* = *yA*）。它

强制实施的一般性质是一个关系：如果实体*y*是*X*，则对于任何财产*甲*的*X* （由属性表示*甲*），*XA* = *YA*。观察到*X*和*ÿ*在*Q* 3 被标以通配符“”，以匹配任意实体。同样，GFD可以强制执行继承关系子类。

特别是，如果*X*标记有鸟，*ÿ*与企鹅，和*阿*是能飞，则*φ* 3个渔获的不一致性例1中描述：企鹅不能飞而是被归类为鸟。

1. FD*和*CFD。考虑一个 FD *R*（ *X* → *Y*）

关系图*R* [3]。当*R的*一个实例表示为一个图，其中每个元组都由一个标记为*R*的节点表示时，我们写*ϕ* 4 =（*Q* 4 [ *x，y* ]，*X* ' → *Y* '）。这里*Q* 4由两个顶点的*X*和*ÿ*表示的两个元组*- [R* ， *X* '由

*XA* = *YA*为所有*甲*∈ *X*，和*ÿ* '包括*的xB* = *YB*为所有*乙*∈ *ÿ*。注意，*ϕ* 4仅用变量文字定义。

使用常量字面量，GFD可以表达CFD [16]。例如，*R*（国家= 44 *，*邮政编码→街道）是在关系*R*上定义的CFD ，表明在英国，邮政编码唯一确定了街道[16]。它可以表示为GFD ]，*X* ' → *Y* '），其中*X* '由*x*组成*。*国家= 44，*y。*国家/地区= 44，*x。*邮编= *y。*zip和*Y*′是*x。*街道= *y。*街道。再举一个例子，CFD *R*（国家= 44 *，*区号= 131→ 城市= Edi）指出在英国，如果城市的区号为131，则城市为Edi [16]。它可以表示为GFD ，其中 由标记为*R*的单个节点*x*组成，并且*X*包括*x。*国家/地区= 44和*x。*区号= 131，而*Y* ''是*x。*城市= Edi。

1. *博客*： *φ* 5 =（ *Q* 5 [ *X，Y，Z* ]，∅→ *X。*文本=*收率*降序）。它指出

如果实体*x，y*和*z*满足图3所示的*Q* 5的拓扑约束，则博客*z*的状态*x* 的注释必须与*z中*包含的照片*y*的描述匹配。

1. *假账户*： *ϕ* 6 =（ *Q* 6 [ *x，x* ' *，y* 1 *，...，y k，z* 1 *，z* 2 ]， *X* 6 →

*Y* 6），其中*X* 6包括*x* ' *。*是假的= true ，*z* 1 *。*关键字= *c*，*z* 2 *。*关键字= *c*，且*Y* 6为*x。*是假= true ; 这里*c*是一个常数，表示一个特殊的关键字。它声明对于帐户*x*和 *x* '，如果满足*X* 6中的条件，包括*x* '被确认为伪造的，则*x*也是伪造的帐户。✷

**语义学**。为了解释GFD ，我们使用以下符号。考虑一个GFD *φ* =（*Q* [ˉ *X* ]*中，X* → *ÿ*）。考虑一个匹配*ħ*（ˉ *X*）的*Q*中的曲线图*G ^*，和一个文字*XA* = *Ç*。我们说*ħ*（ˉ *X*） *满足*字面如果*存在*属性*甲*在节点*v* = *ħ*（*X*）*和VA* = *Ç*; 类似地，对于*xA* = *yB*。我们用*ħ*（ˉ *X*）| = *X*如果*ħ*（ˉ *X*）满足*所有*在文字*X* ; 类似地，对于*ħ*（ˉ *X*）| = *ÿ*。在这里，我们写*ħ*（*μ*（*X*））为*H ^*（*X*），其中，*μ*是在映射*Q*从ˉ *X*到节点*Q*。

的曲线图*ģ满足*GFD *φ*，记为*G ^* | = *φ*，如果*对所有*的匹配*ħ*（ˉ *X*）的*Q*中*ģ*，如果*ħ*（＆OverBar; *X*）| = *X*然后*ħ*（＆OverBar; *X*）| = *ÿ*。我们写*^ h*（ˉ *X*）| = *X* → *Y ^* 如果*^ h*（ˉ *X*）| = *Ÿ*每当*^ h*（ˉ *X*）| = *X*。

请注意以下几点。（1）对于文字*XA* = *Ç*在*X*，节点*ħ*（*X*）不一定必须属性*甲*。如果*ħ*（*X*）具有*无*属性*甲*，*ħ*（ˉ *X*）满足平凡*X* → *ÿ*。这使我们能够适应图的半结构化性质。（2）相反，当*XA* = *C ^*是在*ÿ*和*ħ*（ˉ *X*）| = *Y*，然后*ħ*（*x*）*根据*上述满意度的定义，必须具有属性*A*；类似地对于*xA* = *yB*。（3）当*X*是∅，*ħ*（ˉ *X*）| = *X*为任何匹配*ħ*（ˉ *X*）的*Q*中*ģ* ; 对于*Y* =∅同样。

**例6：**考虑**例5的**GFD *ϕ* 1 *，ϕ* 2和*ϕ* 6以及图2的*G* 1 *，G* 2 *，G* 3。一个人可以验证以下内容。

1. *G ^* 1 ̸| = *φ* 1。实际上，由于*h* 1（*x* 1），因此示例4中给出的匹配*h* 1满足*X* 1 *。*val = *h* 1（*y* 1）*。*val ，但是因为*h* 1（*x* 3）不满足*Y* 1 *。*VAL ̸= *ħ* 1（*y* 3）*。*VAL 。同样，*G*2 ̸| = *φ* 6，通过匹配作为目睹*ħ* 2的实施例4。注意，也有其它的匹配*Q* 6中*ģ* 2满足*X* 6 → *ÿ* 6，*例如，*当我们映射*X* ' ＆→ ACCT 1和*X*＆→ ACCT 2，但是，*G ^* 2 | = *φ* 6仅当*所有*的比赛*Q* 6在*ģ* 2满足*X*6 → *Y* 6。
2. *g ^* 3 | = *φ* 2由于存在*不*匹配的*Q* 2中*g ^* 3：在国内*ģ* 3具有独特的资本，平凡满足*φ* 2。

请注意以下几点：（a）处于*Q*相同匹配中的实体可能相距很远；*例如，*航班1和航班2彼此*断开连接*；（b）*X* → *Y*仅对*Q的*匹配（满足其拓扑约束）施加，*例如ϕ* 2。✷我们说一个图*摹满足*的一组Σ GFDs如果对所有*φ*＆Element;Σ，*g ^* | = *φ*，*即，G*满足每GFD在Σ。

**特殊情况**。GFD 包含以下特殊情况。

1. 如通过 和 实施例5中，关系的FD和差价是特殊情况GFDs ，当在元组关系被表示为图中的节点。实际上，GFD能够表达产生相等性的依存关系（EGD ）[3]。
2. 甲GFD （*Q* [ˉ *X* ]*中，X* → *ÿ*）被称为*恒定*GFD如果*X*和*ÿ*包括常量文字的ˉ *X*只。如果*X*和*Y*仅包含变量文字，则称为*变量* GFD 。直观地说，恒定GFDs归入恒定差价[16]，并且变量GFDs类似于传统的FD [3]。

在示例5中，*ϕ* 1 - *ϕ* 5是变量GFD ，是常数 GFD ，而 既不是常数也不是变量。

1. GFD可以指定某些类型信息。对于*τ*类型的实体*x*，GFD （*Q* [ *x* ] *，* →→ *xA* = *xA*）强制*x*必须具有*A*属性，其中*Q*由标记为*τ*的单个顶点组成，并由变量*x*表示。但是，GFD*无法*强制*x*具有有限域，*例如*布尔值。

# 关于GFDS的原因

接下来，我们研究GFD的可满足性和隐含性问题。这些是与任何类型的数据依赖项相关的经典问题。我们的主要结论是，对于GFD而言，这些问题并不比CFD难。

## GFD的满足能力问题

如果Σ具有*模型*，则GFD的集合Σ是*可满足*的; 即，曲线图*ģ*使得（a）*G ^* |每个=Σ，和（b）GFD （*Q* [ˉ *X* ]*中，X* → *ÿ*）在Σ中，存在一个匹配的*Q*在*ģ*。

在*满足性问题*的GFDs是确定，给出了一组Σ GFDs ，Σ是满足的。

直观地，当用作数据质量规则时，检查GFD是否本身“脏”了。Σ的模型*G*要求Σ的GFD中的所有模式都可以在*G中*找到匹配项，以确保GFD不相互冲突。

对于关系数据，CFD的集合Σ可能无法满足[16]。图形上的GFD也会发生同样的情况。

**示例7：**考虑两个GFD，它们定义为图3中所示的相同模式*Q* 7：*ϕ* 7 =（*Q* 7 [ *x* ] *，* ∅→ *xA* = *c*）和），其中*c*和*d*是不同的常数。然后不存在图表*ģ*包括*τ*实体*v*和同时满足*φ* 7和。因为如果存在这样的节点*v*，那么由*ϕ* 7，*v*具有一个属性*阿*与值*Ç*，同时通过， *vA型*必须采取不同的值*d*，这是不可能的。

作为另一个例子，考虑GFDs *φ* 8 =（*Q* 8 [ *X，Y，Z* ] *，* ∅→ *XA* = *C ^*）和*φ* 9 =（*Q* 9 [ *X，Y，Z，W* ] *，* ∅→ *XA* = *d*），用于分别表示*c*和*d*，其中*Q* 8和*Q* 9如图3所示。一个人可以验证*ϕ* 8和*ϕ* 9中的每一个有一个模型，当单独使用时。但是，将它们放在一起时是不能令人满意的。事实上，如果他们有一个模型*ģ*，那么就必须存在同构映射*ħ*和*ħ* '从*Q* 8 和*Q* 9至*G ^*，分别使得*ħ*（*X*）= *^ h* '（*X*）= *v*对于一些节点*v*在*ģ*。再一次，要求*v*具有具有不同值的属性*A。*✷

如示例7所示，用不同图形模式定义的GFD可能会相互影响。事实上，*Q* 8和*Q* 9是不同的，但是*φ* 8和*φ* 9 可以在同一节点上被执行，因为*Q* 8是同构的一个子图*Q* 9。这告诉我们，可满足性分析必须检查子图

0

0

0

0

0

0

0

1

1

1

1

1

1

1

0

0

0

1

1

1

0

0

0

1

1

1

0

0

0

1

1

1

0

0

0

1

1

1

0

0

1

1

0

0

1

1

0

0

1

1

0

0

0

1

1

1

y

z

x

τ

τ

τ

l

l

l

τ

x

Q7

Q8

y

z

x

τ

τ

τ

l

l

l

l

l

Q9

w

τ



### 图3：GFD中的图形模式

GFDs模式之间的同构是NP完全的（参见[34]）。鉴于此，我们有以下内容。

**定理1：***对于*GFD *，可满足性问题是*coNP完全的*。*✷

人们可能会认为，如果Σ仅由常数GFD组成（请参见第3节），或者Σ中的所有模式都是非循环有向图（DAG ），则问题会变得更加简单。但是，复杂性范围相当强大。

**推论2：***对于用DAG模式定义的恒定*GFD *，可满足性问题是*coNP-完全的*。*✷

GFD的复杂性不是从CFD继承的。实际上，CFD的可满足性分析仅在强制属性具有*有限域*[16]（*例如*布尔值）的方案的情况下才是NP困难的， *即，*将CFD和有限域放在一起时。相反，图不带有模式。而GFDs归入差价合约，他们不能指定有限域。也就是说，可满足性问题GFDs已经CONP -hard在没有一个模式的。

上限证明是不平凡的。它需要以下符号和引理。

1. 一种图案*Q* ' =（*V Q* ' ，*ê Q* '，*大号*' *Q*，*μ* '）是*嵌入*在*Q* =（*V Q*，*ê Q*，*大号Q*，*μ*），如果存在一个同构映射*˚F*从（*V Q* ' *，E Q* '）到（*V Q，E Q*）的子图，并保留节点和边缘标签。如果*Q*' 是可嵌入在*Q*经由*˚F*，那么对于任意GFD *φ* ' =（*Q* ' [ˉ *X* ' ] *，X* ' → *y* '与所定义的）]， *˚F*（*X* '）→ *˚F*（*Y* '））是一个*嵌入式* GFD的*φ* '在*Q*，其中*˚F*（*X*）可替代*˚F*（*X* '），用于每个 *X* '在*X* ' ; 对于*f*（*Y* '）同样。在这里，我们再次交替使用变量*x*和节点*μ*（*x*）。
2. 对于一个图案*Q*和的一组Σ GFDs ，一组Σ *Q*的GFDs被说成是*嵌入在Q和衍生自*Σ如果对于每个*φ*＆Element;Σ *Q*，的图案*φ* 是*Q*，此外，还有存在*φ*＆Element;＆Sigma;使得*φ*是一个嵌入式GFD的*φ*在*Q*。
3. 对于一组Σ *Q*的GFDs嵌入在相同的模式*Q*中，我们定义一组执行（Σ *Q*文字的）感应如下：

◦如果（*Q* [ˉ *X* ] *，* ∅→ *ÿ* ）是Σ *Q*，然后*ÿ* ⊆执行（Σ *Q*），*即*所有的文字*y*都包括在执行（Σ *Q*）; 和

◦如果（*Q* [ˉ *X* ]*中，X* → *ÿ* ）是Σ *Q*，并且如果所有的文字*X*可以衍生自执行（Σ *Q*）通过平等原子的传递性，则*ÿ* ⊆执行（Σ *Q*）。

作为传递性的一个例子，如果*XA* = *C ^*和*YB* = *C ^*是在执行（Σ *Q*），然后*XA* = *YB* ∈强迫（Σ *Q*）。直观地，执行（Σ *Q*）是一组平等原子具有上的曲线图，以强制执行*ģ*其满足Σ（并因此Σ *Q*）。

一个可以验证给定的Σ *Q*，执行（Σ *Q*）可以在多项式时间（计算PTIME ）沿着相同的路线如何传统封闭的FD被计算（参见，*例如，* [3]）。我们说Σ *Q*是*相互矛盾的*，如果存在（*XA，一*）和（*XA，B*）在执行（Σ *Q*），使得*一个*̸= *b*。

（4）一组Σ GFDs是*相互矛盾的*，如果*存在*的图案*Q*和一组Σ *Q*的GFDs嵌入在*Q*和从Σ导出，使得Σ *Q*是*相互矛盾的*。

冲突的GFDs特征的可满足GFDs 。

**引理3：***当且仅当*Σ*不冲突时，* GFD*的集合*Σ才可*满足。*✷

**定理证明1**。基于引理，我们开发了一种算法，在给定GFD集合Σ的情况下，如果Σ*不满足*（ *即*GFD可满足性的补充），则返回“是” 。的（a）猜想（I）的一组Σ ' ⊆Σ，（ⅱ）一个图案*Q*，使得*Q*携带出现在Σ和标签| *问*| 最大为Σ中最大模式的大小，并且（iii）从Σ '中每个GFD的模式到*Q*的映射。（b）检查映射与*Q的*子图是否同构。（c）如果是这样，导出组Σ *Q* 的从Σ ' 和猜测的映射嵌入*Q中的*GFD 。（d）检查∑ *Q*是否冲突；如果是这样，则返回“是”。该算法在引理3中是正确的。在步骤NP中，步骤（b），（c）和（d）在PTIME中。因此，GFD的可满足性在coNP中。

下限通过从子图同构减少到可满足性问题的补集来验证。归约仅使用由DAG 模式定义的恒定GFD ，因此也证明了推论2。✷

*可处理的案件*。接下来，我们将确定可以有效执行可满足性分析的特殊情况。

**推论四：***一组*Σ*的*GFDs*始终是满足的，如果满足下列条件之一满足：*

◦Σ*由可变的*GFDs *，或者仅*◦Σ*不包括*GFDs*形式的*（*Q* [ˉ *X* ] *，* ∅→ *ÿ* ）*。*

*它是在*PTIME*检查是否*Σ*是可满足的，如果*Σ*由*GFDs*仅树形结构的图案，定义*即，*如果对于每个*GFD （*Q* [ˉ *X* ]*中，X* → *ÿ* ）*在*Σ*中，Q是一个树。*✷

## GFD的隐含问题

我们说的一组Σ GFDs*意味着*另一个GFD *φ*，由Σ表示| = *φ*，如果对所有的图表*摹*，使得*G ^* | =Σ，我们有*g ^* | = *φ*，*即φ*是Σ的一个合乎逻辑的结果。

我们假设*wlog*如下：（a）Σ是可满足的，因为否则考虑Σ| = *ϕ*是没有道理的；和（b）*X*是令人满意的组文本，其中的*φ* =（*Q* [ˉ *X* ]，*X* → *ÿ*），因为否则*φ*平凡成立。我们将看到，这些不会增加暗示问题的复杂性。

的*含义问题*的GFDs 是确定，给出了一组Σ GFDs和另一GFD *φ*，是否Σ| = *φ*。

实际上，隐含分析可帮助我们消除定义为GFD的冗余数据质量规则 ，从而通过最小化规则来优化错误检测过程。

**示例8：**考虑两个GFD的集合Σ （*Q* 8 [ *x，y，z* ]，*xA* = *yA* → *xB* = *yB*）和（*Q* 9 [ *x，y，z，w* ]，*xB* = *yB* →

*zC* = *wC*）。考虑GFD *ϕ* 11 =（*Q* 9 [ *x，y，z，w* ]，*xA* =

*yA* → *zC* = *wC*），其中模式*Q* 8和*Q* 9给出为

图3.可以验证Σ| = *ϕ* 11。 ✷

GFD的含义分析是NP-完全的。相反，对于差价合约，问题是coNP完全的[16]。

**定理5：** GFD*的蕴涵问题是*NP*完全。*✷

如通过实施例8中建议，以决定是否Σ| = *φ*，我们必须考虑它们的图形图案之间的相互作用即使在*φ*和所有GFDs在Σ是可变GFDs ，而当它们都没有的形式（*Q* [ˉ *X* ] *，* ∅→ *Y*）。因此，与推论2相比，GFD的含义分析比其可满足性分析更有趣。

**推论6：***其含义的问题是*NP *-complete为恒定*GFDs*单独，和用于可变*差价*单独，即使当所有的*GFDs*与定义DAG 图案和当它们都不具有形式*（*Q* [ˉ *X* ] *，* ∅→ *ÿ*）*。* ✷

为了证明这些，考虑的一组Σ GFDs和GFD *φ* =（*Q* [ˉ *X* ]*中，X* → *ÿ*）。我们定义以下符号。

1. 我们假设*ϕ*是*正常形式*，*即，*当 *Y*由单个字面量*xA* = *yB*或*yB* = *c组成时*，它不是重言式*xA* = *xA*。这不会失去普遍性。事实上，如果*ÿ*由多个文字，则*φ*相当于一组GFDs （*Q* [ˉ *X* ]*中，X* →*升*），一个用于每个文字*升*∈ *ÿ*。如果*是*是∅或重言式，则Σ| = *ϕ*成立。
2. 对于一组Σ *Q*的GFDs嵌入*Q*，我们定义一组闭合（Σ *Q，X*）文字的感应如下：

◦ *X* ⊆闭合（Σ *Q，X*），*即，*所有的文字*X*都在上面; 和◦如果（*Q* [ˉ *X* ' ] *，X* ' → *y* '）是Σ *Q* ，并且如果所有文字*X* ' 可以衍生自闭合（Σ *Q，X*）经由平等原子，那么及物*ÿ* ' ⊆闭合（Σ *Q，X*）。

需要注意的是封闭件（Σ *Q，X*）不同于执行（Σ *Q*）仅在基础案例：与一组给定前开始*X*文字的，而后者使用*X*从 GFDs与∅→ *X*。

沿着相同的线作为关系的封闭件的FD [3]，可以验证封闭件（Σ *Q，X*）可以被计算PTIME 。

1. 回想一下，*Y*是上面定义的正常形式的文字。我们说，*Ÿ*是*可推断*从Σ和*X*，如果*存在*一组Σ *Q*的GFDs嵌入在*Q*和Σ导出，使得*ÿ* ∈关闭（Σ *Q，X*）。

我们将隐含分析的特征描述如下。

**引理7：***对于φ=* （*Q* [ˉ *X* ]*中，X* → *ÿ*）*和一组*Σ*的*GFDs *，*

Σ| = *φ当且仅当Y是推断出来*Σ*和X*  ✷

引理的证明是其关系的延伸FD对应物（参见[3]的关系的FD ）。

**定理证明5**。对于上限，我们给出一种确定Σ| = *ϕ*的算法，如下所示。（a）中猜测一组Σ ' ⊆Σ，并且从每个图案的映射GFD在Σ '中的图案*Q* 的*φ*。（b）检查映射与*Q的*子图是否同构。（c）如果是这样，导出组Σ *Q*的 GFDs嵌入*Q*从Σ '和猜测的映射。（d）检查是否*Y* ∈闭合（Σ *Q，X*）; 如果是这样，则返回“是”。该算法在NP中因为步骤（b），（c）和（d）在PTIME中。它的正确性来自引理7。

当解除关于*ϕ中*的Σ和*X*的可满足性的假设时，该算法可以通过两个初始步骤扩展：（i）检查Σ是否在NP中不能满足；如果是，则返回“无效”，否则继续；（ii）检查是否

*X*是可满足的，在PTIME中；如果是这样，继续；否则返回

“是的”。扩展算法仍在NP中。即，该假设不会增加复杂度界限。

下限通过从子图同构的变体还原而得到验证，该变体显示为NP-完全。减少仅使用常量GFD或仅使用变量CFD ，所有这些均由DAG定义。因此，它也证明推论2 ✷ *容易处理的情况*。一个有效的特殊情况如下。

**推论8：***隐含的问题在*PTIME*中*

*用树结构模式定义的* GFD *。*✷

# 确定性检测

作为GFD的应用，我们根据GFD的验证分析来检测图中的不一致。我们的主要结论是，尽管GFD的验证问题棘手，但通过并行可扩展算法有效地检测实际图形中的错误是可行的。

## GFD验证和错误检测

给定一个GFD *φ* =（*Q* [ˉ *X* ]*中，X* → *ÿ*）和曲线图*G ^*，我们说一个匹配*ħ*（ˉ *X*的）*Q*在*ģ*是*违反*的*φ*如果*ģ ħ* ̸| = *φ*，其中*G ^ ħ*是通过诱导的子图 *ħ*（ˉ *X*）。对于一组Σ GFDs ，我们表示由比奥（Σ *，G*）该组所有侵犯的GFDs在*ģ*，*即，H*（＆OverBar; *X*）∈比奥（Σ *，G*），当且仅当存在一个GFD *φ*在Σ使得*ħ*（ˉ *X*）是违反*φ*在*ģ*。也就是说，当将GFD的集合Σ用作数据质量规则时，Vio （Σ *，G*）会收集所有不一致的*G* 实体。该*错误检测问题*表述如下：

◦*输入*：一组Σ GFDs和的曲线图*G ^*。

◦*输出*：违规的集合Vio （Σ *，G*）。

其决策问题，被称为*验证问题*为GFDs ，是决定是否*ģ* | =Σ，*即，*是否比奥（Σ *，G*）是空的。这个问题很重要。

**命题9：***验证*GFDs*是*CONP *-complete。*✷

**证明：** 我们证明它是NP难的检查，因为*摹*和Σ，是否*摹*̸| =Σ，由子图同构减少。对于上限，我们给出一个返回的算法

“是”如果*G ^* ̸| =Σ：（一）猜GFD （*Q* [ˉ *X* ]*中，X* → *ÿ*）从Σ和映射*ħ*从*Q*到的子图*ģ* ; （b）检查*h*是否同构；（c）中如果是，则检查是否*ħ*（＆OverBar; *X*）| = *X*但*ħ*（ˉ *X*）̸| = *ÿ* ; 如果是这样，则返回“是”。这是在NP中。✷

相反，对于FD和CFD来说，验证是在PTIME中进行的，并且可以通过两个SQL查询来检测关系中的错误，这两个查询可以从FD 和CFD中自动生成[16]。也就是说，错误检测在图形中更具挑战性。

**顺序算法**。我们给出一种算法，给定一组GFD的Σ和一个图形*G*，用一个处理器即可计算Vio （Σ *，G*）。它表示为detVio ，其工作原理如下。（1）以Vio （Σ *，G*）= starts开始。（2）对于每个（*Q* [ˉ *X* ]*中，X* → *ÿ*）在Σ，它枚举所有比赛*ħ*（ˉ *X*）的*Q*中*ģ*，并检查是否*ħ*（ˉ *X*）̸| = *X* → *ÿ* ; 如果是这样，它会添加*ħ*（ˉ *X*）至比奥（Σ *，G*）。

的成本detVio由枚举匹配为主*ħ*（ˉ *X*）的*Q* [ˉ *X* ]在Σ。对于大*G来说，*它是指数级的，而禁止的。

## 并行可伸缩性

错误检测在大规模图形中可行吗？通过使用并行算法计算Vio （Σ *，G*），我们的答案是肯定的。为了表征并行化的有效性，我们采用了*并行可伸缩性*的概念[29]。表示为

◦ *w ^*（Σ *，G*）的*工作量*，*即，*以计算所需的工作的必要量比奥（Σ *，G*）是否有任何算法;

◦*吨*（|Σ| *，* | *g ^* |）“最佳”的运行时间*顺序算法*来计算比奥（Σ *，G*），*即，* 所有这样的算法中，它具有至少最坏情况的复杂性; 和

◦ *Ť*（|Σ| *，* | *g ^* | *，N*）通过平行算法来计算所花费的时间比奥（Σ *，G*通过使用）*Ñ*处理器。

错误检测算法是*并行可伸缩的，*如果

*，*

这样当*ñ* ≤| *G* | 在实践中发现，其中 *c*和*l*是常数。当*n*变大时，它将减少运行时间。直观地，这种算法保证了对于（可能很大）图*G*，使用的处理器越多，计算 Vio （Σ *，G*）所需的时间就越少。因此，它使错误检测变得可行。

**工作量模型**。为了表征错误检测的成本，我们首先引入一个模型来量化其工作量。

我们从概念开始。考虑一个GFD *φ* =（*Q* [ˉ *X* ]*中，X* → *ÿ*），其中（*Q* 1 *，...，Q ķ*）是（最大）连接的部件*Q*。考虑ˉ *ž* =（*Ž* 1 *，...，Z ķ*），其中对于*我*∈[1 *，K* ]，*ž我*在可变ˉ *X*，使得*μ*（*Ž我*）处于节点*Q我*，其中*μ*是从变量到节点中的映射*问*（请参阅第2节）。我们修复ˉ *Ž*，简称为*枢轴*的*φ*，通过拾取*ž我*与最小半径*Q我*，其中所述*半径*是之间的最长最短距离*μ*（*Ž我*），并在任何节点*Q我*。我们使用PV （*φ*）来表示（（*Ž* 1 *，C* 1个*Q*）*，...，* （*Ž ķ，C ķ Q*）），被称为*枢轴矢量*的*ϕ*，其中*c i Q*是在*μ*（*z i*）处的*Q i*的半径。

请注意以下几点。（a）在子图同构的局部性，对任意一个图*ģ*，匹配*ħ*（ˉ *X*）的*Q*中*ģ*，任何节点*v* = *ħ*（*X*为）*X* ∈ *X*＆OverBar;，*v*是内*Ç我Q*啤酒花的一些*h*（*z i*）。（b）向量PV （*ϕ*）可以在*O*（| *Q* | 2）时间内计算，其中*Q*远小于*G*在真实生活中。（c）模式*Q*通常具有1个或2个连接的组件，并且99％的组件的半径最大为2 [21]。因此，在PV （*ϕ*）中，ar ||| *ž* ˉ|| 每个半径*c i Q* 通常为1或2。

**示例9：**对于示例5的GFD ，PV （*ϕ* 1），PV （*ϕ* 2），

PV （*ϕ* 4）和PV （*ϕ* 6）是（（*x，* 1）*，*（*y，* 1）），（（*x，* 1）），（（*x，* 0）*，*（*y，* 0））和（（*x，* 3））（分别见图2）; 特别是，我们采取帐户*X*为一个枢轴*Q* 6 ; 类似地，对于*φ* 3为*φ* 5。✷

甲*工作单位瓦特*用于检查*φ*在图*ģ*的特征在于一一对一映射*σ*从ˉ *Ž*到在节点*G ^*，其中ˉ *ž*是在枢轴PV （*φ*），使得对于每*ž我*∈ *ž* ˉ，*σ*（*ž我*）和*μ*（*ž我*）共享相同的标签，*即，σ*（*ž我*）是*候选*的*μ*（*ž我*）。更具体地，*瓦特*=⟨ *v*＆OverBar; *Ž，G Ž* ˉ ⟩，其中的（a）＆OverBar; *v Ž* = *σ*（ˉ *Ž*）; 和（b）*ģ Ž* ˉ是的片段*ģ*包括，对于每个*ž我*∈ *Ž* ˉ，所述*Ç我Q -neighbor*的*σ*（*Ž我*），*即，*的子图*G ^*由内所有节点诱导*Ç我Q*的啤酒花*σ*（*z i*）。直观地，*ģ ž* ˉ是在一个数据块*ģ*必须被检查，以验证*φ*。我们指的ˉ *v ž*作为*支点候选人*为*φ*在*摹*。

的*工作量W ^*（*φ，G*），用于检查*φ*在*ģ*，记为*W ^*（*φ，G*），是设定的作业单位⟨的*v*＆OverBar; *Ž，G ž* ˉ ⟩当ˉ *v ž*范围以上的所有枢轴候选*φ*在*摹*。的*工作量W ^*（Σ *，G*的一组Σ的）GFDs在*ģ*是！*φ* ∈Σ *w ^*（*φ，G*）。

请注意以下几点。（a）中，以验证GFD *φ*中的曲线图*G ^*，它足以枚举匹配*ħ*（ˉ *X*）的*Q*中的数据块*ģ ž* ˉ 的每个工作单元的*φ*，由子图同构的局部性。也就是说，*我们枚举小G ž* ˉ *，而不是在大G*。（b）顺序成本*t*（|Σ| *，* | *G* |）是

| *摹ž* ˉ | |Σ| 对于所有*ģ Ž* ˉ的出现在*w ^*（Σ *，G*）。（c）大小| *W*（Σ *，G*）| 最多| *G* | *ķ*，其中*ķ*是的最大元数*Ž* ˉ在所有PV （*φ*）的*φ*＆Sigma;＆Element;。正如前面所指出的，通常*ķ* ≤2。因此| *W*（Σ *，G*）| 是*呈指数较小*比*吨*（|Σ| *，* | *G ^*|）。（d）对于匹配*ħ*（ˉ *X*），检查是否*ħ*（＆OverBar; *X*）| = *X* → *Y*需要*ø*（（| *X* | + | *ÿ* |）日志（| *X* | + | *ÿ* |））的时间，和| *X* | + | *Y* | ≤| *ϕ* |。自大小 *ϕ* | 的*φ*远小于| *ģ* |， *w ^*（*φ，G*）足以评估用于检查工作的量*φ*在*ģ*。

**挑战**。计算Vio （Σ *，G*）是一个双准则优化问题。（a）*工作负载均衡*，以将*W*（Σ *，G*）均匀分配到*n个*处理器上；这样做是为了避免“偏斜”的分区，*即*当处理器获得的工作单元比其他处理器多得多时，从而使并行度最大化。（b）*最小化数据传输*，以降低通信成本，这通常是瓶颈[4]。当一个图形*ģ*被分段和跨处理器而分布，以处理单位*瓦特*=⟨ *v*＆OverBar; *Ž，G ž* ˉ⟩，我们需要从一个处理器到另一艘船数据组装*ģ ž* ˉ。成本用CC （*w*）表示，用*c s* ∗ |来衡量。*M* |，其中*c s*是一个常数，*M*是出厂的数据。

**并行可扩展错误检测**。我们在以下两种情况下解决这些挑战，这是[22]演示的实际并行范例。我们证明了在这些设置中并行可伸缩性是可以实现的。

*复制摹*。在每个处理器[22]处复制图形*G。*

我们研究了*具有复制G的错误检测*（第6.1节），以平衡*n个*处理器上的工作负载*W*（Σ *，G*），从而使计算Vio （Σ *，G*）的总体并行时间最小化。

**定理10：***存在一个可扩展的并行算法，给定一组*Σ*的*GFDs*和图G复制在n个处理器，计算*比奥*的Tg*记录| *W*（Σ *，G*）|））*并行时间。*✷

*分区摹*。当*G*在处理器之间分区时，不可避免地要进行数据传送。我们研究了*使用分区G*（第6.2节）的*错误检测*，其*双重标准*目标是（a）最小化数据传输，并（2）平衡工作量。

**定理11：***存在一个可扩展的并行算法，给定一组*Σ*的*GFDs *，分区图G和n亲*

*继承人，计算 t G*

日志| *W*（Σ *，G*）| + |Σ|| *W*（Σ *，G*）|）*并行时间。* ✷

# 并行算法

接下来，我们将在上面给出的设置中开发用于错误检测的并行可伸缩算法，分别作为第6.1节和第6.2节中定理10和11的证明。这样的算法使得检测大型图形中的错误变得可行。我们应该指出，还有其他衡量并行算法有效性的标准（请参阅第1节）。

## 复制图的并行算法

我们从在每个处理器上复制*G*时的设置算法开始。在这种情况下，主要挑战是平衡每个处理器的工作量。这个想法是对工作负载*W*（Σ *，G*）进行并行划分，并为*n个*处理器分配（大约）相等数量的工作单元。

**算法**。该算法表示为repVal ，如图4所示。在使用协调器*S c*和 *n个*处理器*S* 1 *，...，S n的情况下*，需要执行以下步骤。（1）第一估计工作量*w ^*（Σ *，G*），并创建一个平衡分区*Wˉˉ我*（Σ *，G*的）*w ^*（Σ *，G*），用于*我*∈[1 *，N* ]，通过调用一个并行程序BPAR （线1）。然后将*W i*（Σ *，G*）发送到处理器*S我*（第2行）。（2）每个处理器*S i*通过并行的过程localVio（第3行）来检测由Vio *i*（Σ*，G*）表示的*一*组局部违规，该过程仅访问*W i*（Σ*，G*）中指定的数据块。。（3）当所有的处理器*š我* 返回比奥*我*（Σ*，G*），*小号Ç*单位计算比奥（Σ*，G*）通过取所有的联合比奥*我*（Σ*，G*）（行4-5）。然后返回Vio（Σ*，G*）（第6行）。

接下来，我们介绍程序bPar和localVio 。

**工作负载平衡**。过程bPar在两个阶段平衡工作负载：并行进行估计和分区。

*工作量估算*。过程bPar首先按照以下三个步骤并行估计工作负载*W*（Σ*，G*）。

（1）在协调器*小号Ç*，对于每个GFD *φ*＆Element;＆Sigma;，BPAR构造一个枢轴矢量PV （*φ*）=（＆OverBar; *Z，C*＆OverBar; *Q*）。然后，它按如下所述在*n个*处理器上平衡计算以进行工作量估计。

**算法**repVal

输入：GFD的集合Σ ，协调器*S c*，*n个*处理器*S* 1 *，...，S n*，在每个处理器处复制的图形*G*输出：违规集合Vio （Σ *，G*）。

1. bPar （Σ *，G*）; / \*并行平衡工作量\* /

/ \*在协调器*S c上*执行\* /

1. 将*W i*（∑ *，G*）发送到处理器*S i*；
2. 调用localVio （Σ *，W我*（Σ *，G*））在每个处理器*š我* 为*我*∈[1 *，N* ]。
3. **如果** 每个处理器*S i* 返回答案Vio *i*（Σ *，G*），**则**
4. Vio （Σ *，G*）：= ！*我*∈[1 *，N* ] 比奥*我*（Σ *，G*））;
5. **返回** Vio （Σ *，G*）;

**过程**localVio （Σ *，W i*（Σ *，G*））

/ \*在每个处理器*S i*上并行执行\* /

1. 设置Vio *i*（Σ *，G*）：=∅;
2. **对于每个***瓦特*=⟨ *v Ž* ˉ *，* | *ģ ž* ˉ |⟩∈ *Wˉˉ我*（Σ *，G*），用于GFD *φ*＆Element;Σ**待办事项**
3. 枚举匹配*ħ*（ˉ *X*）通过访问*ģ ž* ˉ ;
4. **为每个***ħ*（ˉ *X*），使得*ħ*（ˉ *X*）̸| = *X* → *ÿ* **做**
5. 比奥*我*（Σ *，G*）：=比奥*我*（Σ *，G*）∪{ *ħ*（ˉ *X*）};
6. **返回** Vio *i*（Σ *，G*）;

**图4：** repVal**算法**

1. 对于每个变量*Ž*在枢轴ˉ *Ž*，它提取的频率分布*Ç候选*（*μ*（*Ž*）），*即，* 在这些节点*ģ*具有相同标号作为*μ*（*Ž*）。这可以通过本地存储在*S c*的*G*的统计数据来支持。
2. 对于每个PV （*φ*）=（（*Ž* 1 *，C* 1个*Q*）*，...，* （*Ž ķ，C ķ Q*）），并且每*Ž我*，它均匀地分区候选*Ç*（*μ*（*Ž我*））转换成*米*套，对于预定义的数*m*。更具体地说，它导出了*m个*平衡分区*R μ*（*z i*）= { *r* 1 *，...，r m*}选定属性*C*（*μ*（*z i*））的值范围的} ，使得属性值落在每个范围*r j* 中的*C*（*μ*（*z i*））的候选数是偶数。这可以通过使用*例如* 预先计算的等深度直方图来完成（*例如* [33]）。然后它构造一组*中号*的形式的消息的⟨ PV （*φ*）*中，r Ž* ⟩，其中*φ*是一个GFD ，*- [R ž* =⟨ *ř ž* 1*... [R Ž ķ* ⟩，并且每个*ř ž我*∈ *[R μ*（*Ž我*）是一个*范围*的*Ç*（*μ*（*Ž我*对））*Ž我*。除去重复项，*中号*含有至多*米ķ*消息*φ*，其中*ķ* ≤2在实践中（见第5章）。
3. 将集合*M*平均分配给*n个*处理器；每个处理器*S i*接收关于消息的子集*M i*。

**示例10：**考虑示例5的GFD *ϕ* 1，其中PV （*ϕ* 1）=（（*x，* 1）*，*（*y，* 1））（*即k* = 2）。考虑图*摹* 包括9个航班飞行1 -飞行9。对于*n* = 3 = *m*，过程bPar平衡估计*W*（*ϕ* 1 *，G*），如下所示。

1. 对于两个*μ*（*x*），确定飞行实体的三个范围的*R*飞行，*例如*{{飞行1 *，*飞行3 ]，[飞行4 *，*飞行6 ]，[飞行7 *，*飞行9 ]}。和*μ*（*y*），基于属性*μ*（*x*）*。*val和*μ*（*y*）*。*VAL 。
2. 删除重复项后，它产生6条消息的集合*M*（因为*ϕ* 1的 *Q* 1（图2）中的两个连接分量是同构的，（PV （*ϕ* 1）*，r i，r j*）和（PV （*ϕ* 1）*，r j，r i*）是范围*r i*和*r j*）的重复项。

然后，它均匀地分布*中号*至3级的处理器， *例如S* 1接收*中号*1 = {⟨ PV （*φ* 1）*，*（[飞行1 *，*飞行3 ] *，* [飞行1 *，*飞行，⟨ PV （*φ* 1），（[飞行1 *，*航班3 ] *，* [航班4 *，*航班6 ]）⟩}。

（2）然后，过程bPar并行地识别每个处理器*S i*处的工作单元。对于每个消息⟨ PV （*φ*）*中，r Ž*在⟩*中号我*，*š我*发现（a）所有枢轴候选*v ž* ˉ ˉ的*ž*使得对于每个*ž我*∈ *Ž* ˉ，*其候选v Ž* ˉ [ *ž我*]在*v ž* ˉ在范围具有属性值*[R ž我*∈ *řz* ; 和（b）的*C ^我Q* -neighbors *ģ ž* ˉ对于每个*v ž* ˉ。

然后，每个处理器*S i*向 协调器*S c*发送消息。下面 是一组⟨的*v Z ^* ˉ *，* | *ģ ž* ˉ |⟩，各自编码枢轴候选和*大小*为一个单位的数据块的。

注意 *摹ž* ˉ | 被发送，而不是*ģ ž* ˉ。此外，*š我*跟踪*ģ Ž* ˉ便于本地错误检测（稍后看到的）。

**实施例11：**对于⟨ PV （*φ* 1）*，*（[飞行1 *，*飞行3 ] *，* [飞行1 *，*飞行3 ]）⟩，处理器*š* 1点的发现3名候补{飞行1 *，*飞行2 *，*飞行3 }的范围在[航班1 *，*航班3 ]和其1跳邻居。这些产率*v Ž* ˉ [ *X* ]为（飞行*我，*飞行*Ĵ*）（*我*∈[1 *，* 3]，*Ĵ* ∈[1 *，* 3]，*我<Ĵ*删除重复）并且相应地，用编码3个工作单位| *ģ ž* ˉ |，其中| *摹ž* ˉ | 是的1跳邻居的总大小飞行*我*和飞行*Ĵ*在*v Ž* ˉ [ *X* ]。例如，单位*w* 1是⟨（航班1 *，*航班2）*，* 22⟩，其中*ģ ž* ˉ为*瓦特*1是曲线图*G ^* 1中

图2，总共有22个节点和边。 ✷

（3）步骤BPAR ，在协调器*小号Ç*，收集一组消息⟨的*v Ž* ˉ *，* | *ģ ž* ˉ |⟩从所有的处理器，记*w ^*（Σ *，G*）。它对要分区的工作单元集进行编码。

*工作负载分区*。这就产生了负载平衡的问题-

lem。*W*（Σ *，G*）的*n个*分区W是一组*n*对成对不相交的工作单元集{ *W* 1（Σ *，G*）*，...，W n*（Σ *，G*）}，使得*W*（ Σ *，G*）=！*我*∈[1 *，N* ] *Wˉˉ我*（Σ *，G*）。如果将成本*t*（|Σ| *，W i*（Σ *，G*））估计为|的和，则可以达到*平衡*。*摹ž* ˉ | |Σ| 对于所有*Gž* ˉ在 *Wˉˉ我*（Σ*，G*）），近似相等。的*负载平衡问题*是要找到一个平衡*Ñ* -partition W代表一个给定的*W ^*（Σ*，G*）。将处理器产生的最大成本称为并行处理的*周期*。通过将每个处理器的容量设置为，通过减少PTIME，负载平衡问题“等同于”使制​​造时间最小化[7]。这个问题是棘手的，但是是可以解决的。

**命题12：***（1）负载均衡问题已解决*NP*问题。（2）有一种*2*近似算法可在O*（*n* | *W*（Σ *，G*）| + | *W*（Σ *，G*）| log | *W*（Σ *，G*）|）*并行时间中找到平衡的工作负载分区给定*Σ *，n和W*（Σ *，G*）*。*✷

给定*W*（Σ *，G*），过程bPar遵循[7]的近似算法，采用贪婪策略计算出均衡的*n个*分区，以最大程度地缩短制造期。（1）首先关联一个权重| *ģ*（ˉ *Ž*）| 与每个工作单元*瓦特*=⟨ *v Ž* ˉ *，* | *摹ž* ˉ |⟩。然后按权重降序对所有工作单元进行排序。对于每个处理器，它将初始为0的负载与之相关联。（2）贪婪地选择*重量*最小的工作单元*w*和负载最小的处理器*S i*进行分配*瓦特*到*小号我*和更新的负载*小号我*通过添加的重量*瓦特*。（3）继续进行直到分配所有工作单元为止。通过保留与[7]相对应的近似因子，可以得出2-近似算法。

**实施例12：**假设协调*小号Ç*接收9个工作单元{*瓦特*1 *，...，W* 9 }中总，与估计的大小{22 *，* 22 *，* 26 *，* 26 *，* 30 *，* 30 *，* 24 *，* 28 *，* 28}，分别。bPar的贪婪分配策略生成工作单元的3个分区，分别为{{ *w* 1 *，w* 3 *，w* 9 } *，* { *w* 2 *，w* 4 *，w* 5 } *，* {*w* 6 *，w* 7 *，w* 8 }}，平衡块大小分别为76、78、82。然后，*S c*将这三个分区分别分配给处理器*S* 1， *S* 2，*S* 3。✷

**本地错误检测**。在接收到分配的*W i*（Σ）时，过程localVio并行计算每个处理器*S i上*的本地违规集Vio *i*（Σ *，G*）。对于每一个工作单位⟨ *v Z ^* ˉ *，* | *ģ ž* ˉ |⟩∈ *Wˉˉ我*（Σ *，G*），用于GFD *φ*，它（一个）枚举匹配*ħ*（ˉ *X*）在该图案的*φ*使得*ħ*（ˉ *X*）包括*v Ž*ˉ，通过只访问*ģ Ž* ˉ，和（b）检查是否*ħ*（＆OverBar; *X*）| = *X* → *y*的*φ*。它在Vio *i*（Σ *，G*）中收集所有检测到的违规，并在处理结束时将Vio *i*（Σ *，G*）发送给协调器*S c*。

**实施例13：**考虑GFD *φ* 1 =（*Q* 1 [＆OverBar; *X* ]*中，X* 1 → *Y* 1）（实施例5）和工作单位 *瓦特*1分配给处理器（实施例11）*š* 1。过程 localVio检查 *ģ* 1（图2），用于*瓦特*1，并找到了匹配*ħ* 1（＆OverBar; *X*）的*Q* 1在*ģ* 1，其中*ħ* 1实施例4中给出如图所示，*ħ*1（＆OverBar; *X*）̸| = *X* 1 → *ÿ* 1。因此localVio加*ħ* 1（＆OverBar; *X*）至比奥1（Σ *，G*）。类似地，*S* 1处理分配给它的*w* 3和*w* 9，最后将Vio 1（Σ *，G*）返回给*S c*。✷

**定理证明10**。通过子图同构的局部性，过程bPar标识了所有工作单元，而localVio计算了所有违规。根据这些，可以得出repVal的正确性。对于复杂性，可以验证以下内容：

通过使用平衡分区，过程bPar估计）并行时间；划分需要*O*（*n* | *W*（Σ *，G*）| + | *W*（Σ *，G*）| log | *W*（Σ *，G*）|）时间[7]；和（b）

过程localVio需要*ø*（*吨*（|Σ *Ñ* | *，* | *g ^* |） ）平行时，通过一个平衡的工作量分区。因此，repVal具有定理10中所述的复杂性，并且可以并行扩展。✷6.2**碎片图算法**

图*G*可能已经碎片化并分布在*n个*处理器上，特别是当在每个处理器上复制*G的*成本太高时。在这种情况下，我们有一个*双标准错误*检测问题。给定GFD的集合Σ和碎片图*G*，将并行计算Vio （Σ *，G*），以使（1）通讯成本最小化，以及（2）平衡*n个*处理器的工作量。

考虑*G*（*V，E，L，F A*）的碎片（*F* 1 *，...，F n*），这样（a）每个*F i*（*V i，E i，L，F A*）是的子图。*G*（b）！*E i* = *E*和！*V我*= *V*，和（c）*˚F我*驻留在处理器*š我*（*我*∈[1 *，N* ]）。假设*博客*中*F i*的大小大约相等。此外，*˚F我*保持（a）中的轨道*innodes˚F我.I*， *即，*在节点*V我*到其中存在来自另一个片段存在的边缘，和（b）*出器节点F我.O*，*即，*节点在另一个片段到其中存在从在一个节点的边*V我*。我们指的是在节点*˚F我。我*还是 *˚F我.O*为*边界节点*。

**算法**。我们为片段*G*提供了一个错误检测算法，称为disVal 。它在工作量估计和分配以及本地错误检测方面与repVal不同，以最大程度地减少通信和计算成本。算法disVal与协调器*S c*和*n个*处理器*S* 1 *，...，S n一起工作*。（1）首先通过过程disPar估计和划分工作负荷*W*（Σ *，G*）*，*以使每个*S i*处的工作负荷*W i*（Σ *，G*）平衡，以最低的通讯成本。（2）每个处理器*S i都*使用过程dlovalVio来并行检测本地违规Vio *i*（Σ *，G*），并进行数据交换。（3）最后，Vio （Σ *，G*）=！*我*∈[1 *，N* ]比奥（Σ*我，G*）。接下来，我们介绍程序disPar和dlovalVio 。

**双标准分配**。过程disPar通过支持（a）具有通信成本的工作量估计和（b）双标准分配来扩展其对应的bPar 。

*工作量估算*。过程disPar并行估计每个*S i*处的*W*（Σ*，G*）。对于每个枢轴矢量PV（ *ϕ*）=

（（*ž* 1 *，C* 1个*Q*）*，...，* （*Ž ķ，C ķ Q*）），并且每*Ž升*在ˉ *Ž*，它发现的（a）本地候选*Ç*（*μ*（*Ž升*的））*μ*（*Ž升*）在*˚F我*，（b）该*ç升Q* -neighbors *ģ ž* ˉ [ *ž升*]的每个候选*ç*（*μ*（*ž升*）），和（c）边界节点*乙Ž* ˉ [ *ž升*从] *ģ Ž* ˉ [ *ž升*中]到某些节点*ģ Ž* ˉ [ *Ž升*]。它编码*部分工作*

*单元W φ* 作为⟨ *v Ž* ˉ *，* | *摹ž* ˉ | *，B ž* ˉ ⟩，其中（i）*v Ž* ˉ是一个枢轴候选

在*F i中的z*；如果*Ç*（*μ*（*Ž升*））=∅，*v ž* ˉ [ *Ž Ĵ* ]需要的占位符⊥; （ii）| *摹ž* ˉ |

是|的列表 *ģ ž* ˉ [ *ž升*] |; 和（iii）*乙Ž* ˉ是边界节点的列表*乙Ž* ˉ [ *Ž升*]，对于所有*ž升*∈ *Ž* ˉ，指示“丢失数据”。然后，每个*S i*将消息*M i*连同所有单元一起发送给协调器*S c*，以及*F i .I*中边界节点的*c*邻居的大小，其中*c的*范围为Σ中的图案*Q*的半径。

收到*M i后*，disPar便建立*W*（*ϕ，G*），这是位于*S c*的完整工作单元的集合。在一个工作单元⟨ *v Z ^* ˉ *，* | *摹ž* ˉ | *，B ž* ˉ ⟩加入到*w ^*（*φ，G*）如果对于每个*ž升*∈ *Ž* ˉ，*v ž* ˉ [ *ž升*]是候选

]从单位*瓦特φ我* 的*中号我*使得 是的总和（从提取| *G ^我Ž* ˉ |），和*乙ž* ˉ是的并集（从萃取*乙Ž我*ˉ），对于所有*我*∈[ 1 *，正*]和*φ*＆Element;＆Sigma;。 

也就是说，disPar将]组装为工作单元。它还用来源*w ϕ i*标记和[ ] 。 

*工作量分配*。该*双标准分配问题*是

找到一个*Ñ*的-partition *w ^*（Σ *，G*）插入*Wˉˉ我*（Σ *，G*），用于*我*∈[1 *，N* ]，使得（a）*w ^我*（Σ *，G*）是平衡的，以及（b）其通信成本CC *i*被最小化，其中CC *i*表示如果将*W i*（Σ *，G*）分配给*S i*，则需要传送到处理器*S i*的数据量。它应确保各关键候选*v ž* ˉ，存在一个独特的单位⟨ *v Z ^* ˉ *，* | *摹ž* ˉ | *，B ž* ˉ ⟩在所有的*w ^我*（Σ *，G*），*即，*所述候选被检查一次。

成本CC *i*估算如下。对于每个⟨ *v Z ^* ˉ *，* | *摹ž* ˉ | *，B ž* ˉ ⟩在*Wˉˉ我*（Σ *，G*），并且每个*Ž升*∈ *Ž* ˉ，定义CC *v Ž* ˉ [ *Ž升*]为的总和（a）中， *即，* ]必须是从片段取出*Ĵ* ; （b）| *G* （*c* *lQ ，b* ） | 对于每个边界节点*v b* ∈*乙ž* ˉ [ *Ž升*]，它也要求数据提取。这些是通过使用源标识*瓦特φ我*上述记录。让CC *v ž* ··是的总和CC *v Ž* ˉ [ *ž升*]所有*ž升*∈ *ž* ˉ。然后CC*我*是所有的总和CC *v ž* ˉ候选人*v Ž* ˉ 在*Wˉˉ我*（Σ *，G*）。注意，CC *i*仅对每个数据块计数一次。

尽管双标准分配比负载平衡更吸引人，但实际上通过逼近可以做到。

**命题13：***（1）双准则分配问题是*NP-完全的*。（2）存在一种*2*近似算法，以找到在O*（*n* | *W*（Σ *，G*）| 2 log（| *W*（Σ *，G*）|）） *时间内通信成本最小的平衡工作负载分配。*✷

延伸的策略完工时间最小化[43]，程序毒蛾计算一个*Ñ*的-partition *Wˉˉ*（Σ）（后单元分组）转换成*Wˉˉ我*（Σ *，G*），发送到处理器*š我*为*我*∈[1 *，N* ]。

**本地错误检测**。在接收到*W i*（Σ *，G*）时，过程dlovalVio通过选择以下评估方案，在处理器*S i上*计算局部违规Vio *i*（Σ *，F i*）。

*预取*。 对于工作单位*w ^* =⟨ *v Z ^* ˉ *，* | *摹ž* ˉ | *，B ž* ˉ ⟩时，它首先

取*ģ ž* ˉ和*G ^* （*C，B*）为*˚F我.O*节点*乙Ž* ˉ 从其他片段。它确保每个节点（边缘）仅检索一次。数据放置到位后，它将像localVio一样在本地检测错误以计算Vio *i*（Σ *，F i*）。

*部分检测*。我们还可以提供部分匹配而不是数据块。我们的想法是通过估计部分匹配的尺寸*图形仿真*[19]从图案*Q* [ˉ *X* ]在GFD *φ*到*˚F我*。如果部分匹配的数目不大，则*S i*以流水线方式与其他处理器交换此类匹配，并在可以从部分匹配形成完整匹配后立即更新Vio *i*（Σ*，F i*）。

对于一个单元*瓦特*∈ *Wˉˉ我*（Σ *，G*），用于GFD *φ*在*小号我*，过程dlovalVio选择一种策略，即被更小（估计）通信成本CC （*瓦特*）（参见附录的估计CC （*瓦特*））。凭直觉，dlovalVio决定在本地或在远程处理器上处理每个单元，以较小的数据传送为准。

我们的算法还支持偏斜图和减少工作量的优化策略（请参阅附录）。

我们通过证明disVal是正确的并且具有所需的复杂度来验证定理11 ，类似于定理10。

|  |
| --- |
| **图5：绩效评估** |

# 实验研究

我们使用现实生活中的合成图通过实验评估了（1）并行可伸缩性，（2）工作负载分区，（3）通信成本，（4）我们算法的可伸缩性，

（5）GFD在错误检测中的有效性。

**实验设置。**我们使用了三个真实的图：（a）DBpedia ，一个知识图[1]，包含2千8百万个实体（200个类型）和33个*。*400万个边缘的160种类型，（b）YAGO2 ，YAGO的扩展知识库[44]，具有13种类型的350万个节点和36种类型的735万个边缘，（c）Pokec [2]，一个具有1的社交网络*。*269万种不同类型的6300万个节点，以及30个*。*600万种11种类型的边缘。我们删除了无意义的节点和标签，以实现紧凑的表示形式。然后，我们通过在一组采样实体上反复取消引用HTTP URI来插入新边缘，以进一步扩大DBpedia （分别为YAGO2）），1230万（相当于320万）实体和3270万（相当于710万）边缘。

我们还开发了一种生成器，可以根据幂律度分布生成合成图*G* =（*V，E，L，F A*）。它由节点数控制。*V* | （多达5000万）和优势| *E* | （最多1亿个），其中*L*由30个标签的字母L组成，*F A为*5个属性分配有效域中1000个值的值。

GFDs*生成器*。我们产生套Σ GFDs （*Q* [ˉ *X* ]*中，X* →

*Y*），由（a）||Σ||，GFD的数量和（b）| *Q* |，图形模式*Q*的平均大小，以Σ为单位，具有1个或2个连接的组件。对于每个真实的图，（1）我们首先挖掘频繁特征，包括长度最大为3的边和路径。我们选择了前5个最频繁特征作为“种子”，并将它们组合形成大小| | |的模式*Q。问*|。（2）对于每个 *Q*，我们使用由节点属性组成的文字构造依赖项*X* → *Y* 。通过这种方式，我们在每个现实图上生成了100个GFD 。对于合成图，我们生成了50个带有从L绘制的标签的GFD 。

*算法*。我们全部用Java实现了以下内容：（1）

顺序算法detVio （第5节），（2）并行算法repVal （图4），与它的两个变体（a）rep ran，随机地将工作单元分配给处理器，和（b）rep nop，不支持优化策略（多查询处理[31]和工作量减少;见附录），和（3）并行算法disVal （第6.2节），相对于它的两个变体DIS RAN和DIS NOP类似于对口（2）。

我们在Amazon EC2 c4.2xlarge实例上部署了算法，每个实例均由2.6GHz的Intel Xeon处理器提供支持。我们使用了多达20个实例。每个实验进行了5次，平均值在此处报告。

**实验结果**。接下来，我们报告我们的发现。

**Exp-1：并行可伸缩性**。我们首先评估了并行算法repVal和disVal以及它们的变体。固定| *Q* | = 5和||Σ|| = 50，我们将处理器的数量*n*从4更改为20。我们分别为repVal和disVal复制和分段*G。*图5（a），5（b）和5（c）分别报告了它们在现实生活中的DBpedia ，YAGO2和Pokec的性能。我们发现以下内容。（1）当*n为n*时，repVal和disVal都大大减少了并行时间增加：*n*从4到20分别平均快3.7和2.4倍。这些验证定理10和11（2）两个repVal和disVal胜过它们的变体：repVal （RESP disVal ）平均为1.9和1.4倍（分别为1.5和1.3倍。）的速度比代表NOP和代表RAN（相应的DIS nop和dis ran）。这些证明了我们优化和负载平衡技术的有效性。（3）算法repVal是

0

500

1000

1500

2000

2500

3000

M, 20M

(10

)

(20

)

M, 40M

(30

)

M, 60M

)

(40

M, 80M

(50

)

M, 100M

Time (seconds)

dis

nop

dis

ran

disVal

## 图6：可扩展性：变| *G* | （合成的）

比disVal更快，因为它不需要通过使用复制的*G*进行交易就可以进行数据交换。（4）repVal和disVal在大型现实图上均能很好地工作。例如，在具有20个处理器的YAGO2上，repVal （resp。disVal ）花费156（resp。326 ）秒。相反，顺序算法detVio不会在6000秒内终止于三个图上的任何一个。平均而言，平行图复制（未显示）需要21 *。*YAGO2 ，DBpedia和Pokec分别为3、89和75秒， 分别。复制仅执行一次，然后可用于所有查询。

**Exp-2：工作量复杂度**。接下来，我们评估的复杂性的影响GFDs通过改变||Σ||，数量上估算工作量和分区，GFDs和| *Q* |，平均图案尺寸。我们固定*n* = 16。

*变*||Σ||。 固定| *问*| = 5，我们||Σ|| 从50到

100.如分别在DBpedia ，YAGO2和Pokec上的图5（d），5（f）和5（h）所示，（a）如预期的那样，所有算法在较大的Σ上花费更长的时间，并且（b）repVal （RESP。disVal ）表现优于代表RAN 和代表NOP （分别为DIS RAN 和DIS NOP），通过平衡工作量和减少沟通。但是，当||Σ||时，detVio不会在三个图表中的任何一个图表上在120分钟内终止。≥80。

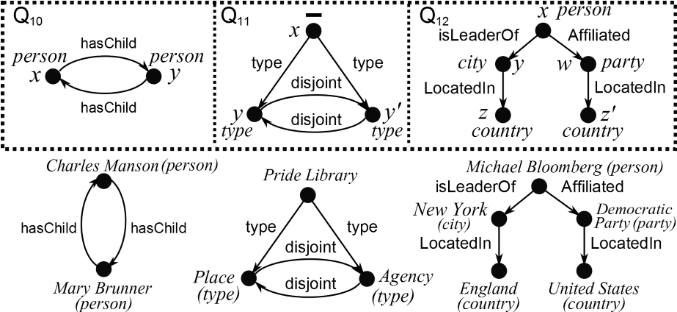
*变*| *问*|。 固定||Σ|| = 50，我们有所不同| *问*| 从2到

6.如图5（e），5（g）和5（i）所示，所有算法在| *Q* |，由于工作单位较大。然而，repVal （RESP。disVal ）优于代表NOP和代表RAN（分别为DIS NOP和DIS RAN）在所有情况下，由于上述同样的原因。同样，在以下情况下，detVio不会在120分钟内终止：*问*| 在所有三个图表上≥6。

**Exp-3：通讯费用**。在与Exp-1相同的设置中，我们评估了图5（j），5（k）和5中报告的三个数据集的disVal ，dis ran和dis nop的总通信成本（以并行数据传送时间衡量）。（l）。我们省略了repVal，因为它不需要数据交换。我们发现以下内容：（a）发送的数据总量（未显示）远小于基础图形的大小；这证实了我们对通讯费用的估计（第5和第6节）；（b）当*n为n*时，通信成本占总检错成本的12％至24％从4变为20；这就是为什么添加处理器并不能总是减少并行运行时间的原因之一[19]，因为使用更多的处理器会导致不同处理器之间进行更多的数据交换。（c）尽管随*n的*增加而运送了更多的数据，但由于并行运送，通信时间对*n*并不十分敏感。

**EXP-4：合成***ģ*。我们还评估了算法DisVal在50M节点和100M边的大型合成图上的性能。由于复制*G的*存储容量有限，因此我们仅在分区*G*时测试了该设置。

固定*n* = 16，我们变化| *G* | 从（10M，20M）到（50M，100M）。如图6所示，（1）所有算法在较大的| *G* |，如预期；（2）错误检测功能强大-



**图7：**大型图形中的**真实**GFDs ：当|时，disVal需要21分钟。*G* | =（50M，100M）; （3）disVal分别比dis ran 和dis nop平均快1.9和1.5倍；这与现实生活中的结果一致；和（4）顺序算法detVio 在以下情况下无法运行完毕：*G* | 使用一台处理器在120分钟内≥（30M，60M）。

**经验5：效果**。为了证明GFD在错误检测中的有效性，我们在图7中显示了三个真实的GFD和它们所捕获的错误。与其他方法相比，附录中报告了另一组实验。

GFD *1*是（*Q* 10 [ˉ *X* ] *，* ∅→ *X。* VAL = *Ç* ∧*收率*VAL = *d*为不同）*Ç*和*d*，（*即，x。* VAL = *Ç* ∧*收率*VAL = *d*是假，指出一个人*x*不能既有孩子又有父母，就没有*y*，这导致了图7所示的YAGO2中的不一致。

GFD *2*是（*Q* 11 [ˉ *X* ] *，* ∅→*收率*VAL = *y* ' *。* VAL ），指出一个实体不能有两个不相交的类型（不具有共同的实体）。它确定了DBpedia在“架构”级别上的矛盾，该矛盾与不相干的关系矛盾。

GFD *3*是（*Q* 12 [ˉ *X* ] *，* ∅→ *Ž。* VAL = *Ž* ' *。* VAL ）。它可以确保，如果某人是国家*z*所在城市的市长，并且隶属于国家*z* '的当事方，则*z*和*z* '必须是同一国家/地区。它检测到YAGO2中的错误，该错误使纽约市市长目睹了与纽约市（NYC）和民主党相关的不同国家。

**摘要**。从实验结果我们发现以下内容。（1）在实际图形中使用GFD进行错误检测是可行的，*例如，*在具有20个处理器的YAGO2上*，* repVal （resp。disVal ）花费156（resp。326 ）秒。（2）更妙的是，它们是并行可伸缩的，响应时间缩短了3 *。*7和2 *。*4次，分别当处理器的数量为4至20。增加（3）我们的优化技术提高的性能代表NOP和DIS NOP 1 *。*9和1 *。*5次；工作负载平衡得到改善代表跑和DIS跑出1 *。*4和1 *。*3次。（4）GFD能够捕获实际图形中的不一致之处。

# 结论

这项工作是迈向图依赖理论的第一步。我们提出了GFD ，为其经典问题建立了复杂性界限，并为其应用提供了并行可扩展算法。我们的实验结果证明了GFD技术的有效性。

未来工作的主题之一是为现实图形中的GFD发现开发有效的算法。另一个问题是，提供一个健全和完整的公理系统GFDs ，沿着相同的路线作为阿姆斯特朗的公理关系的FD [3]。第三个主题是在存在常见于知识库中的类型和其他语义约束的情况下，重新研究GFD的可满足性和隐含性问题。

**致谢**。范和徐得到了ERC 652976、973计划2014CB340302和2012CB316200，NSFC 61133002和61421003，EPSRC EP / J015377 / 1和EP / M025268 / 1，NSF III 1302212，深圳孔雀计划1105100030834361，广东省创新研究团队计划2011D005的部分支持，深圳市科学技术基金JCYJ20150529164656096，广东应用研发计划2015B010131006，北京大数据与脑计算高级创新中心以及华为技术有限公司的研究资助。Wu获得了Google教师研究奖的支持。

# 参考资料

1. Dbpedia。http://wiki.dbpedia.org/Datasets 。
2. Pokec社交网络。http://snap.stanford.edu/data/soc-pokec.html 。
3. S. Abiteboul，R。Hull和V. Vianu。*数据库基础*。Addison-Wesley，1995年。
4. FN Afrati，V。Borkar，M。Carey，N。Polyzotis和JD Ullman。Map-reduce扩展和递归查询。在*EDBT中*，2011年。
5. FN Afrati，D。Fotakis和JD Ullman。使用map-reduce枚举子图实例。在*ICDE中*，2013年。
6. FN Afrati和JD Ullman。在map-reduce环境中优化联接。在*EDBT中*，第99-110页，2010年。
7. G. Aggarwal，R。Motwani和A. Zhu。负载平衡问题。在*SPAA中*，2003年。
8. W. Akhtar，A。Cort′es-Calabuig和J. Paredaens。RDF中的约束。在*SDKB中*，第23–39页，2010年。
9. M. Arenas和L. Libkin。XML文档的标准格式。在*PODS中*，第85-96页，2002年。
10. D. Calvanese，W。Fischl，R。Pichler，E。Sallinger和M. Simkus。在RDFS中捕获关系模式和功能依赖性。在*AAAI中*，2014年。
11. Ca. Q.，M。Sirivianos，X。Yang和T. Pregueiro。协助在大型社交在线服务中检测假帐户。在*NSDI中*，第197-210页，2012年。
12. A. Cort′es-Calabuig和J. Paredaens。RDFS中的约束语义。在*AMW中*，第75–90页，2012年。
13. O. Etzioni，M。Cafarella，D。Downey，S。Kok，A.-M。

Popescu，T。Shaked，S。Soderland，DS Weld和

答：耶茨。KnowItAll中的Web级信息提取。在2004年的*WWW中*。

1. 范W，范Z，田C和董LX。图的键。在*PVLDB中*，2015年。
2. 范·范和盖尔特斯。*数据质量管理的基础*。Morgan和Claypool出版社，2012年。
3. 范凡，F。盖特斯，贾X和A. Kementsietsidis。用于捕获数据不一致的条件功能依赖项。*TODS*，33（1），2008。
4. W. Fan，F。Geerts，S。Ma和H. Muller。检测¨分布式数据中的不一致。在*ICDE*，64-75页，2010。
5. 范伟，李健，唐北和余伟。增量检测分布式数据中的不一致性。*TKDE*，26（6），2014。
6. 范伟，X。Wang和Y. Wu。分布式图形模拟：不可能和可能性。*PVLDB*，2014年。
7. L. Gal′arraga，K。Hose和R. Schenkel。Partout：用于高效RDF处理的分布式引擎。在

*WWW*，2014年。

1. MA Gallego，JD Fern′andez，MA Mart′ınez-Prieto和P. de la Fuente。对现实世界中SPARQL查询的实证研究。在*USEWOD研讨会上*，2011年。
2. M. Hammoud，DA Rabbou，R。Nouri，S.-M.-R。Beheshti和S. Sakr。DREAM：具有自适应查询计划器和最少通信的分布式RDF引擎。*PVLDB*，2015年。
3. B. He，L。Zou和D. Zhao。使用条件函数依赖关系来发现RDF图中的异常数据。在*SWIM中*，第1-7页，2014年。
4. J. Hellings，M。Gyssens，J。Paredaens和Y. Wu。功能和恒定约束的含义和公理化。*安 数学。Artif。智力* ，第1–29页，2015年。
5. J. Huang，DJ Abadi和K. Ren。大型RDF图的可扩展SPARQL查询。*PVLDB*，4（11）：1123-1134，2011年。
6. S. Huang，Q。Li和P. Hitzler。混合MKNF知识库中的不一致推理。*IGPL逻辑杂志》*，2012年。
7. H. Karloff，S。Suri和S. Vassilvitskii。MapReduce的计算模型。在*SODA中*，第938–948页，2010年。
8. Z. Khayyat，IF Ilyas，​​A。Jindal，S。Madden，M。Ouzzani，P。Papotti，J.-A。Quian′e-Ruiz，N。Tang和S. Yin。Bigdansing：大数据清理系统。在*SIGMOD中*，2015年。
9. CP Kruskal，L。Rudolph和M. Snir。有效并行算法的复杂性理论。*TCS*，71（1）：95-132，1990。
10. 黎黎，秦黎，林X.和张昌。MapReduce中的可伸缩子图枚举。*PVLDB*，8（10）：974-985，2015年。
11. W. Le，A。Kementsietsidis，S。Duan和F. Li。SPARQL的可扩展多查询优化。在*ICDE中*，2012年。
12. 门多萨（L. Mendoza）。一种基于规则的方法来解决链接数据上的语义准确性问题。2004年。
13. H. Mousavi和C. Zaniolo。快速准确地计算数据流上的等深度直方图。在*EDBT中*，2011年。
14. CH Papadimitriou。*计算复杂性*。Addison-Wesley，1994年。
15. B. Parsia和E. Sirin。佩莱特：OWL DL推理机。在*ISWC海报*，第18卷，2004年。
16. 普兰滕加（T. Plantenga）。MapReduce中不精确的子图同构。*J.平行分布。计算* ，73（2）：164-175，2013年。
17. H. Poon和P. Domingos。信息提取中的联合推理。在*AAAI*，第7卷，第913-918页，2007年。
18. 秦L.余建雄，L。Chang，H。Cheng，C。Zhang和X. Lin。MapReduce中的可伸缩大图处理。在*SIGMOD中*，2014年。
19. R. Raman，O。van Rest，S。Hong，Z。Wu，H。Chafi和J. Banerjee。Pgx。iso：用于子图同构的并行高效内存引擎。在*GRADES中*，第1-6页，2014年。
20. A. Rula和A. Zaveri。评估链接数据质量的方法。在*LDQ @ SEMANTiCS中*，2014年。
21. 邵宇，崔斌，陈林，马林，姚洁和徐楠。

大规模图中的并行子图列表。在*SIGMOD中*，2014年。

1. 沉W. A. Doan，JF Naughton，和

拉玛克里希南（R. Ramakrishnan）。使用具有嵌入式提取谓词的数据日志进行声明式信息提取。在*VLDB中*，2007年。

1. DB Shmoys和E. Tardos。广义分配问题的一种近似算法。*数学编程*，62（1-3）：461-474，1993。
2. FM Suchanek，G。Kasneci和G. Weikum。Yago：语义知识的核心。在*WWW上*，第697-706页，2007年。
3. FM Suchanek，M。Sozio和G. Weikum。SOFIE：用于信息提取的自组织框架。在*WWW中*，第631–640页，2009年。
4. 孙中山，王洪波，王洪波，邵乙和李健。

在十亿个节点图上进行有效的子图匹配。*PVLDB*，2012年。

1. S. Suri和S. Vassilvitskii。计算三角形和最后一个减速器的诅咒。在*WWW上*，第607-614页，2011年。
2. 陶耀，林伟和肖小。最小的MapReduce算法。*SIGMOD*，第529–540页，2013年。
3. Yu。Yu和J. Heflin。扩展功能依赖性以检测RDF图中的异常数据。在*ISWC中*。

2011。

1. A. Zaveri，D。Kontokostas，MA Sherif，L。Buhmann，M。Morsey，S。Auer和J. Lehmann。用户驱动的dbpedia质量评估。在*ICSS中*，第97-104页，2013年。

# 附录优化策略

接下来，我们介绍算法所采用的优化策略。

**偏斜图**。现实生活中的图（*例如*社交网络）可能会出现偏斜的分布，*即，*大多数节点几乎没有邻居，而一小部分节点与大部分边缘相邻。这种偏态分布可导致少量的大数据块的*g ^ Ž* ˉ，因此偏斜工作量。

程序BPAR和毒蛾可以容易地适应于偏斜图形，通过施加*复制和分离*的策略大*ģ ž* ˉ。（a）设置|的阈值*θ*。*摹ž* ˉ |。（b）对于工作单位*瓦特*与*ģ Ž* ˉ 超过*θ*，我们复制*瓦特*具有相同ˉ *Ž*，但分割*ģ Ž* ˉ 至多与大小的子图*θ*。（c）原始的*w*被这些新的单元代替。对于这些单位，localVio 和dlovalVio通过传送部分匹配而不是数据块来检测*G z中*的错误。

我们通过偏斜图对dlovalVio进行了实验评估。我们生成合成图*ģ*通过控制歪斜测量的的“偏斜” *ģ*，量化为比，其中| *g dm* | （resp。| *G dm* ' |）是*G中*节点的前10％最小（resp。最大）*d*跳邻居的平均大小（我们将*d* = 3设置）。它估计工作单元的数据块大小的不平衡度。偏斜越小，越不平衡。

300

600

900

1200

1500

10

-1

20

-1

30

-1

40

-1

50

-1

Time (seconds)

dis

nop

dis

ran

disVal

## 图8：偏斜图的影响

固定| *G* | =（10M，20M）并且*n* = 16，我们从0改变了偏斜*。*1至0 *。*02.图8显示，由于工作负载不平衡，所有偏斜变小（“偏斜更大”）时，所有算法都会恶化。但是，disVal更重要

“稳健”地倾斜：其运行时间仅增加1 *。G*偏斜5倍时，*G*变7倍，而dis ran变为2.0倍，dis nop变为2.2倍。这验证了我们的“复制和拆分策略”的有效性。

**多查询处理**。已经通过提取公共子模式来优化多模式匹配[31]。算法repVal和disVal使用某些技术（*例如，*模式包含和子模式调度）来处理Σ中的常见子模式，并删除多余的工作单元（*例如，* 示例10）。

**减少工作量**。GFD的含义分析可以帮助我们进一步减少工作量。给定的一组Σ GFDs ，如果Σ\ { *φ* } | = *φ*，我们可以安全地删除*φ*从Σ而不影响比奥（Σ *，G*）。如定理5所示，蕴涵问题是棘手的。但是，如果Σ中的模式是树，则在PTIME中是正确的（推论8），此外，还有启发式算法可以有效地进行分析。

# 与其他方法比较

GFD*与其他模型*。我们评估的有效性GFDs用于误差检测与YAGO2 ，通过用（a）所述的扩展比较差价到RDF [23]中，被称为GCFDs ，和（b）BigDansing [28]。由于YAGO2中完整的“真实”错误集合是未知的，因此我们对一组实体进行了采样。对于每个采样的实体*x*，我们根据[50]的建议，将噪声（概率为2％，总共2690个错误）随机注入YAGO2中：（a）*属性不一致*，通过更改属性*xA*的值；（b）*类型不一致*，通过修改*x*的类型; 和（c）*的代表性的不一致*，通过修改任一值*XA*或*X* ' *.A*如果*XA* = *X* ' *.A*和*X*和*X* '是相同类型的。表示该组与噪声实体比奥，我们定义了*精度*（相应的*召回*的错误检测方法的） （相应的|比奥∩| VioVio |（A）| ，其中，比奥（A）表示不一致的实体集通过检测一种。

我们（1）在YAGO2上构造了一个由10个GFD组成的Σ，其频繁模式与一部分采样实体相匹配，并且具有噪声注入之前原始值的常数；（2）[23]之后的抽样实体上的一组7个GCFD ，包括Σ中具有连接路径的*所有*GFD （GCFD不允许使用一般的图形模式）。（3）由于BigDansing不支持子图同构，因此我们将GFD硬编码为Σ中每个GFD的用户定义函数。

我们在图9中报告了这些方法的运行时间和准确性，其中YAGO2上的*n* = 16扩展了噪声。我们发现（a）GFD比GCFD具有更高的准确性（91％），

|  |  |  |  |
| --- | --- | --- | --- |
| **模型** | **记起** | **精确的** | **时间** |
| **GFD** | **0.91** | **1.0** | **131秒** |
| **气相色谱** | 0.57 | 1.0 | 106秒 |
| **大丹星** | 0.91 | 1.0 | 609年代 |

## 图9：运行时间和准确性

因为它发现了与GCFD无法表示的一般模式的不一致; （b）GFD和GCFD花费的时间相当；（c）BigDansing慢4.6倍，因为它必须将子图同构测试作为关系联接进行转换。它报告的精度与我们的算法相同，因为它对GFD的相同集合Σ进行了硬编码。

*实际的*GFD。关于图7中所示的GFD，请注意以下几点。

GFD *1*不能表示为（a）GCFD，因为*Q* 10是循环模式，或者（b）BigDansing的CFD或拒绝约束（DC ），因为如果不强制执行子图同构，则GFD *1*将变为假负数。

出于与GFD 1相同的原因，无法将GFD *2*表示为GCFD ，CFD或DC 。

尽管*Q* 12是一棵树，但GFD *3*不能表示为GCFD ，因为GCFD无法执行测试*z。*id = *z* ' *。*id ; BigDansing的CFD和DC同样 如此。

**摘要**。从实验结果我们可以看到以下内容。（1）GFD在错误检测方面比GCFD更准确。（2）BigDansing要求用户编码GFD和子图同构，并且比我们的算法慢4.6倍。

1. **原文**

提供更好的翻译建议