Functional Dependencies for Graphs 图函数依赖关系

简介：

1. 背景：对于常见的数据关系，已经有了很好的数据依赖关系，并且可以扩展为XML，经过修订之后可以有效的捕获关系当中的不一致性
2. 问题：在图表中也明显需要FD（依赖关系）和关系数据库不同的是，在现实生活中图形经常不会拥有架构，FD指定了数据语义中的基本部分，因此对于图来说格外重要，此外
    1. FD可以检查知识库中的不一致性例如：

        （一）航班号为A123的飞机有两个入口但是都是从14:50出发到22:35抵达目的地，但是一架航班是巴黎到纽约，另一架是巴黎到新加坡

        （二）堪培拉和墨尔本都被标记为澳大利亚首都

        （三）鸟类被归类为可以飞行，但是尽管企鹅也是鸟但是不能飞

        宗上我们可以看到在图的依赖关系中的FD可以轻松捕获这不一致的地方
    2. 社会图，当一个博客Z发布了一张照片Y，一个社会公司定义一个状态X有一个附属状态Y，并且要求X中的X.text必须和Y中Y.desc匹配

        （一）Blog:if Z has statues X,Z has photo Y, and if X has attachment Y then X.text = Y.desc

        这本质是图结构数据的函数依赖关系

        （二）在捕获垃圾邮件也很有用

        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

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。

3. 解决问题：对于一组GFD ，我们研究它的可满足性，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
4. 研究了一个可行的问题：以GFD作为数据质量规则来检测图形中的错误。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，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。这些使检测大型图形中的错误变得可行。
5. 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。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。

改进

1. （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。
2. 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 [¯ X]中，X → Y），其中 ¯ X是变量列表，以及X和Y与在 ¯ 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]。
3. 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。

具体算法

1. 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]。

基本定义

1. 常见图标的标签表示：

symbols	notations
G	graph (V, E, L, F_A)
$Q[\bar{x}]$	graph pattern (V_Q, E_Q, L_Q, μ)
φ, Σ	GFD $\varphi = (Q[\bar{x}], X \to Y)$, Σ is a set of GFDs
$h(\bar{x}) \models X \to Y$	a match $h(\bar{x})$ of Q satisfies $X \to Y$
Σ_Q	a set of GFDs of Σ embedded in pattern Q
$\mathsf{Vio}(\Sigma, G)$	all the violations of GFDs Σ in graph G
$t(\lvert\Sigma\rvert, \lvert G\rvert)$	sequential time for computing $\mathsf{Vio}(\Sigma, G)$
$T(\lvert\Sigma\rvert, \lvert G\rvert, n)$	parallel time for $\mathsf{Vio}(\Sigma, G)$, using n processors
$W(\Sigma, G)$	workload for computing $\mathsf{Vio}(\Sigma, G)$
$\mathsf{PV}(\varphi)$	a pivot vector $(\bar{z}, \bar{c}_Q)$ of GFD φ
$w = \langle \bar{v}_z, G_{\bar{z}} \rangle$	work unit ($\bar{v}_z$: candidate; $G_{\bar{z}}$: neighbors of $\bar{v}_z$)

Table 1: Notations

2. 图G = (V, E, L, FA)中，V表示有限的节点集合；E表示有限的边集合；L表示节点和边的标签可分为L(v)和L(e)；FA(v)表示节点的属性与属性值的对应关系；

   图形模式，VQ是模式结点的集合，EQ是模式边的集合；LQ是给节点还有边分配标签的函数；是标量的列表；是参数数量，其等于节点的数量；μ是一个双射函数，将映射到VQ中，或找到节点对应在中对应的变量；使用通配符'_'作为特殊标签

图形模式匹配：在图G上找到与模型Q同构的子图，这个子图G'= (V', E', L', F'A)；h是一个双射函数用于从节点VQ到节点VQ'的映射；而对于标签有LQ(u) = L'(h(u))，LQ(e) = L'(e') 其中e = (u, u')，e'= (h(u), h(u'))。当然若 LQ(u)是通配符"_"，则LQ(u) = L'(h(u))总是成立。

是本文常使用的匹配向量，由h(x)组成。x是变量，是变量的列表。

图函数依赖：GFDs ϕ = ,其中Q是图形模式，称为ϕ的模型；X和Y是两个的两个字段集合。

GFD ϕ指定了两个约束条件：1、由模式Q施加的拓扑约束。2、由X → Y指定的属性依赖。 即GFD既要被Q约束也要被函数依赖约束，Q规定了GFD的范围，使得依赖关系X → Y只施加在由Q标识的每个子图中的顶点的属性上

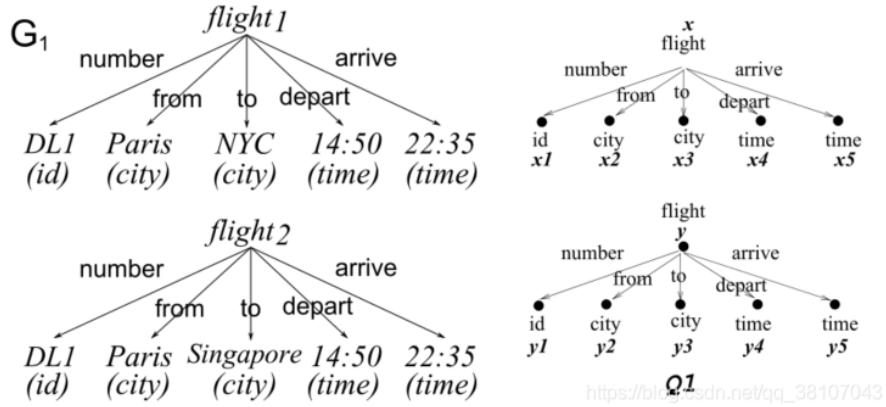
3. 举例说明：

满足GFD $\varphi = (Q[\bar{x}], X \to Y)$，首先将h(μ(x))表示为h(x)，以后就用h(μ(x))表示为h(x)了，因为x本是一个字段，μ是根据这个字段找到其节点。若存在v=h(x)且v.A=c则 $h(\bar{x})$满足c，若存在v=h(x)且v.A=y.B则 $h(\bar{x})$满足y.B。而若X内的所有字段，$h(\bar{x})$都存在一个节点有满足字段的属性，则 $h(\bar{x})$满足X，可表示为 $h(\bar{x}) \models X$。若X为空，则 $h(\bar{x}) \models X$

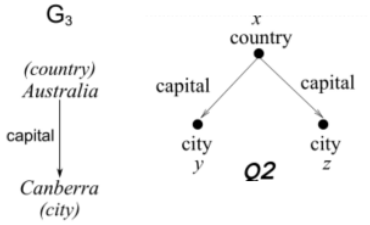
对所有的 $h(\bar{x})$，若 $h(\bar{x}) \models X$ 则 $h(\bar{x}) \models Y$，我们说图G满足GFD φ 记作G $\models \varphi$。每当 $h(\bar{x}) \models X$ 就有 $h(\bar{x}) \models Y$ 则说明 $h(\bar{x}) \models X \to Y$。若 $h(\bar{x})$不满足X则 $h(\bar{x})$无论满不满足Y都 $h(\bar{x}) \models X \to Y$ 都成立，但是若 $h(\bar{x})$满足X，则 $h(\bar{x})$必须满足Y，$h(\bar{x}) \models X \to Y$ 才能成立，这就跟普通的蕴含关系一样，当X真时，Y必须为真，不然X->Y就为假了。

例1

例1、GFD $\varphi_1 = (Q_1[x, x_1\text{-}x_5, y, y_1\text{-}y_5],\ X_1 \to Y_1)$，其中X1是 $x_1.\text{val} = y_1.\text{val}$，Y1由 $x_2.\text{val} = y_2.\text{val}$ 和 $x_3.\text{val} = y_3.\text{val}$ 组成。由Q1所示，x1为flight id，x2为flight出发的city，x3为flight到达的城市等。y1，y2,y3，y4，y5也是如此。因此GFD φ_1 表示若实体x和y有相同的flight id 则它们的离开城市以及目的地一定一样。但是从图中可以看出 $G_1 \not\models \varphi_1$，因为 $h(\bar{x}) \models X$ 但是 $h(\bar{x})$ 不满足Y：$h(\bar{x}) \models X$ 是因为h(x1).val=DL1,h(y1).val=DL1,则h(x1)=h(y1)，X1是 $x_1.\text{val} = y_1.\text{val}$。但是h(x3).val=NYC,h(y3).val=Singapore，二者不相等，所以 $h(\bar{x})$不满足Y。所以 $G_1 \not\models \varphi_1$



例2、GFD $\varphi_2 = (Q_2[x, y, z], \emptyset \to y.\text{val} = z.\text{val})$，次GFD表示没有实体能够满足y.val和z.val，因为X为空，所以有GFD满足X，可以推出GFD必须要满足Y，则这个GFD才成立。如下图所示通过图形模式匹配有h(μ(y))和h(μ(z))都是表示Canberra，所以二者值相等，所以GFD满足X->Y，所以 $G_3 \models \varphi_2$。



例3、GFD $(Q[x], \emptyset \to x.A = x.A)$可以表示为x必须有一个属性A，不然这个图就不满足GFD了。

推论

2、关于GFDS的推理

GFDs的可满足性问题

定理1：GFDs的可满足性问题是coNP-complete的

推论2: 定义于有向无环图的常数GFDs，其可满足性问题是coNP-complete的

引理3：当且仅当Σ不冲突时，GFDs集Σ是可满足的。

推论4:如果满足下列条件之一，GFDs的集合Σ总是可满足的:

1、Σ仅由变量GFDs组成

2、Σ不包含形式为 $(Q[\bar{x}], \emptyset \rightarrow Y)$的GFDs

GFDs的蕴含问题
定理5、GFDs的蕴含问题是NP-complete的

推论6、当所有的GFDs都被定义为DAG模式，并且它们都没有形式时，仅对于常数GFDs和变量CFDs，蕴含问题是NP-complete。

引理7、对于 $\varphi = (Q[\bar{x}], X \rightarrow Y)$ 和一个GFDs集合Σ，有Σ|=ϕ，当且仅当Y可以从Σ和X中扣除.

推论8、用树结构模式定义的GFDs的蕴含问题是PTIME的

不一致性检测

命题9、GFDs的验证问题是coNP-complete。

定理10、存在并行可测量的算法，给一个GFDs集合Σ和一个复制到n个处理器的图G，在 $O(\frac{t(|\Sigma|,|G|)}{n} + |W(\Sigma,G)|(n + \log|W(\Sigma,G)|))$ 并行时间内计算Vio(Σ,G)

定理11、存在并行可测量的算法，给定一个GFDs集合Σ，一个分好了的图G和n个处理器在并行时间内。Vio(Σ, G)在$O(\frac{t(|\Sigma|,|G|)}{n} + n|W(\Sigma,G)|^2 \log|W(\Sigma,G)| + |\Sigma||W(\Sigma,G)|)$

若 $G_h \not\models \varphi$ 则我们说h(x)违反了C，其中GFD $\varphi = (Q[\bar{x}], X \rightarrow Y)$ 是G根据图模式Q使用h(x)匹配到的一个子图。我们将所有违反的h(x)添加到一个违规集合Vio(Σ, G)里。

而我们所谓的错误检测就是输入一个GFDs集合 Σ和一个图G，会有输出一个违反集合Vio(Σ, G)。我们称这个决策问题为GFDs的验证问题，用于判断是否G |= Σ，当然只有违反集合Vio(Σ, G)为空时，G |= Σ。

三个算法

顺序算法

此算法使用了一个GFDs集合Σ，一个图G，一个单处理器来计算违反集合Vio(Σ, G)。我们称这个算法为detVio

执行如下：

刚开始时Vio(Σ, G) = ∅。枚举G中的模式Q中的所有h(x)，检查是否存在即h(x)!=x->y是否存在，若违反了就添加到Vio(Σ, G)。

这种算法非常简单，执行起来的时间复杂度非常非常得高，所以对于大图来说是禁止的。

并行算法的介绍

以上算法复杂度这么高，因此我们提出了一个用于解决此问题的并行算法，也就是复制图的并行算法和碎片图的并行算法

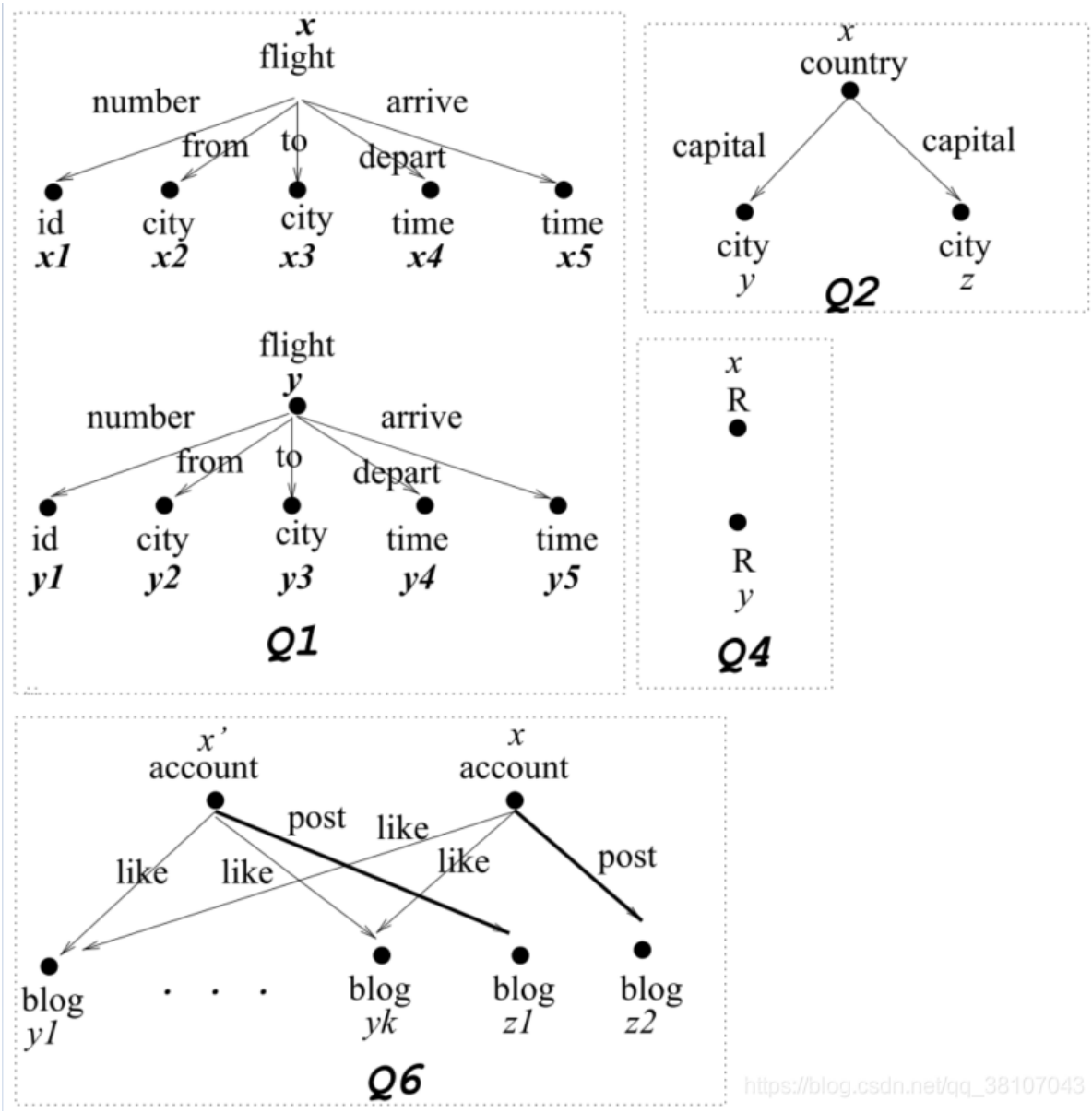
首先介绍一下新的符号和概念：

W(Σ, G)表示工作量，是算法计算所有违反集合Vio(Σ, G)所需要的消耗

t(|Σ|,|G|)表示最好的顺序算法在计算Vio(Σ, G)所消耗的时间

T(|Σ|,|G|, n)表示使用n个处理器通过并行算法来计算Vio(Σ, G)所使用的时间

$(Q_1,\ldots,Q_k)$表示Q的最大连通分量；$\bar{z}=(z_1,\ldots,z_k)$称为GFD的轴(pivot)；z_i 是连通分量 Q_i 的具有最小半径节点的标签，可通过 $\mu(z_i)$ 获得 z_i 对应的节点；使用 c_Q^i 表示 z_i 半径的大小；使用 $\mathrm{PV}(\varphi)$ 表示 $((z_1,c_Q^1),\ldots,(z_k,c_Q^k))$ 称之为GFD的轴向量(pivot vector)；如下图所示 $\mathrm{PV}(\varphi_1)$，$\mathrm{PV}(\varphi_2)$,$\mathrm{PV}(\varphi_4)$和$\mathrm{PV}(\varphi_6)$的轴向量分别是((x,1),(y,1)), ((x,1)), ((x,0),(y,0))和((x,3))



σ 是一个一对一的映射，从Z映射到G中的节点，Z就是上面介绍的轴向量；O(zi)和U(zi)是不一样的，前者表示G中的节点，后者表示Q中的节点，O(zi)是Q(zi)的候选，是对应的分享相同的标签

$w=\langle\bar{v}_z,G_{\bar{z}}\rangle$ 其中(a) $\bar{v}_z=\sigma(\bar{z})$ (b) $G_{\bar{z}}$ 是G中的片段，子图 $G_{\bar{z}}$ 是对所有 $z_i\in\bar{z}$ 的 c_Q^i 邻居 $\sigma(z_i)$，它大于模型Q对应的子图 G_h。我们称 $\bar{v}_z=\sigma(\bar{z})$ 为图G中 φ 的轴候选(pivot candidate)

W(Σ, G)是所有 $W(\varphi,G)$ 的集合符号表示为 $\bigcup_{\varphi\in\Sigma}W(\varphi,G)$；$W(\varphi,G)$ **是 φ 中所有的连通分量Qi的标签 z_i 对应的轴候选 $\bar{v}_z=\sigma(\bar{z})$ 与其对应的G的子图组成的 $\langle\bar{v}_z,G_{\bar{z}}\rangle$ 集合，一般只有1-2个，毕竟Q中的连通分量一般都是1-2个，可以看出W(Σ, G)包含 $W(\varphi,G)$，$W(\varphi,G)$ 包含 $\langle\bar{v}_z,G_{\bar{z}}\rangle$，三者是包含关系。**

上面我们通过Q进行匹配得到的子图 G_h 小于通过 (z_k,c_Q^k) 得到的 $G_{\bar{z}}$ 小于图G，如果使用 $G_{\bar{z}}$ 来进行枚举，那么复杂度将会远小于使用G来进行枚举。

若 $T(|\Sigma|,|G|,n)=\frac{c*t(|\Sigma|,|G|)}{n}+(n|\Sigma||W(\Sigma,G)|)^l$ 我们说错误测量算法是并行可测量的，当 $n\le|G|$ 时 $\frac{c*t(|\Sigma|,|G|)}{n}\ge(n(|\Sigma||W(\Sigma,G)|)^l$ 成立，其中c和l是常数。直觉上使用的处理器数量越多则使用的时间越少，当然在下面的章节会论述这个直觉的错误的。

复制图的并行算法

命题12: 1、负载平衡问题是NP-complete的。2、给定Σ, n和W(Σ, G)，在O(n|W(Σ, G)| + |W(Σ, G)| log|W(Σ, G)|) 时间内有一个2-近似算法来寻找平衡的工作负载分区

此算法先将图G复制到每一个处理器上，我们所要做的就是将工作量均匀的分配到每一个处理器上。

算法名称为repVal，由一个协调器和n个处理器组成。协调器上运行了一个程序bPar，处理器上并行地运行了程序localVio，运行过程为：

协调器的程序bPar先将工作量W(Σ, G)均匀地分为$W_i(\Sigma, G)$，然后将n个$W_i(\Sigma, G)$分别发送到n个处理器上。处理器上的程序localVio收到工作量之后，开始检测违规，并将违规放入$\mathsf{Vio}_i(\Sigma, G)$，处理完之后localVio将$\mathsf{Vio}_i(\Sigma, G)$返回给bPar，bPar再将所有的$\mathsf{Vio}_i(\Sigma, G)$合并成一个Vio(Σ, G)。过程如下图所示：

Algorithm repVal

Input: A set Σ of GFDs, coordinator S_c, n processors $S_1, \ldots, S_n$, a graph G replicated at each processor
Output: Violation set $\mathsf{Vio}(\Sigma, G)$.

1. bPar(Σ, G); /*balance workload in parallel*/
/*executed at coordinator S_c*/
2. send $W_i(\Sigma, G)$ to processor S_i;
3. invoke localVio$(\Sigma, W_i(\Sigma, G))$ at each processor S_i for $i \in [1, n]$;
4. **if** every processor S_i returns answer $\mathsf{Vio}_i(\Sigma, G)$ **then**
5. $\mathsf{Vio}(\Sigma, G) := \bigcup_{i \in [1,n]} \mathsf{Vio}_i(\Sigma, G))$;
6. **return** $\mathsf{Vio}(\Sigma, G)$;

Procedure localVio$(\Sigma, W_i(\Sigma, G))$
/*executed at each processor S_i in parallel*/
1. set $\mathsf{Vio}_i(\Sigma, G) := \emptyset$;
2. **for each** $w = \langle v_{\bar{z}}, |G_{\bar{z}}| \rangle \in W_i(\Sigma, G)$ for GFD $\varphi \in \Sigma$ **do**
3. enumerate matches $h(\bar{x})$ by accessing $G_{\bar{z}}$;
4. **for each** $h(\bar{x})$ such that $h(\bar{x}) \not\models X \to Y$ **do**
5. $\mathsf{Vio}_i(\Sigma, G) := \mathsf{Vio}_i(\Sigma, G) \cup \{h(\bar{x})\}$;
6. **return** $\mathsf{Vio}_i(\Sigma, G)$;

Figure 4: Algorithm repVal

bPar是进行工作量均匀分配方式：

GFD $\varphi_1 = (Q_1[x, x_1\text{-}x_5, y, y_1\text{-}y_5],\ X_1 \to Y_1)$，其中X1是$x_1.\mathsf{val} = y_1.\mathsf{val}$，Y1由$x_2.\mathsf{val} = y_2.\mathsf{val}$和$x_3.\mathsf{val} = y_3.\mathsf{val}$组成。

举个例子：我们使用flight的图和图模式，假设G1有9架飞机flight1–flight9，使用范围为n=3的分区有：
$\{[\mathsf{flight}_1, \mathsf{flight}_3], [\mathsf{flight}_4, \mathsf{flight}_6], [\mathsf{flight}_7, \mathsf{flight}_9]\}$，然后根据分区再产生m=3个M,其中$\langle \mathsf{PV}(\varphi_1), (r_{\mathsf{flight}}, r'_{\mathsf{flight}}) \rangle$，rflight和rflight'分别对应Q1中的x x1 x2 x3 和y y1 y2 y3。可以算出$M_1 = \{\langle \mathsf{PV}(\varphi_1), ([\mathsf{flight}_1, \mathsf{flight}_3], [\mathsf{flight}_1, \mathsf{flight}_3]) \rangle, \langle \mathsf{PV}(\varphi_1), ([\mathsf{flight}_1, \mathsf{flight}_3], [\mathsf{flight}_4, \mathsf{flight}_6]) \rangle\}$. M2包含了4-6,4-6和4-6,7-9. M3包含了7-9,7-9和7-9,1-3. bPar将M1、M2、M3分别发送个3个处理器，交由3个处理器处理。

若S_c收到9个工作单元{w1, . . . , w9}且评估大小为{22,22,26,26,30,30,24,28,28}，则S_c的贪婪分配策略根据评估生成了一个3分区工作单位{{w1, w3, w9},{w2, w4, w5},{w6, w7, w8}}，之后再将之分到处理器S1, S2, S3。

localVio的工作方式：

localVio通过$\text{PV}(\varphi_1)=((z_1,c_Q^1),\ldots,(z_k,c_Q^k))$，这里有两个连通分量所以k等于2，$c_Q^k$等于1，$z_1$表示x对应的flight；通过PV可以得到G的子图$G_{\bar{z}}$

我们举例处理器s1收到的M1为$\langle \text{PV}(\varphi_1),([\text{flight}_1,\text{flight}_3],[\text{flight}_1,\text{flight}_3])\rangle$。有$\bar{v}_z = \sigma(\bar{z})$中z bar等于$(\text{flight}_i,\text{flight}_j)$其中$i\in[1,3],\ j\in[1,3],\text{ and } i<j$，可生成子图$G_{\bar{z}}$。则有小工作量$w=\langle \bar{v}_z, G_{\bar{z}}\rangle$。例有$w_1$ is $\langle(\text{flight}_1,\text{flight}_2),22\rangle$，22表示$G_{\bar{z}}$的节点和边数量总和为22. 然后localVio通过枚举的方法检测$G_{\bar{z}}$是否满足GFD，若不满足则将$G_{\bar{z}}$对应的那个h放入本地的 Vio(Σ, G)，直到所有的h都尝试之后，返回本地的 Vio(Σ, G)给bPar

碎片图的算法

此算法论文没有举例，以后遇到再补充对此算法的使用简介，此处只说了一些定义

命题13: (1)双标准分配问题是NP-complete (2)在$O(n|W(\Sigma,G)|^2\log(|W(\Sigma,G)|))$时间内，存在一种2-近似算法来寻找具有最小通信成本的平衡工作负荷分配

本算法名为disVal。使用了一个协调器S_c和n个处理器$S_1,\ldots,S_{n_o}$。disPar首先评估且分区工作量 W(Σ, G)，使得$W_i(\Sigma,G)$在每个处理器上都是平衡的，且通信成本最小化。之后每个处理器并行地使用程序dlocalVio来探测局部违规 $\text{Vio}_i(\Sigma,G)$。最终 $\text{Vio}(\Sigma,G)=\bigcup_{i\in[1,n]}\text{Vio}(\Sigma_i,G)$

首先先介绍一下**边界节点**再介绍算法的详细工作流程：$(F_1,\ldots,F_n)$是等于图G； $F_i(V_i,E_i,L,F_A)$是图G的子图；$\bigcup E_i = E \text{ and } \bigcup V_i = V$；每个子图都驻留在对应的处理器上；而边界节点由内节点和外节点组成，内节点是子图F_i中那些与其他碎片图有相连边的节点，因为F_i是直接从G上切割下来的图，那么就一定会有断掉的边，内节点就是F_i中那些有断边的节点；外节点是原本与F_i的内节点连接的其他碎片的节点。

disPar拓展了bPar通过支持1、考虑通信成本的工作量估算 2、双标准分配(Bi-criteria assignment)

dispart工作方式：

使用了轴向量$\text{PV}(\varphi)=((z_1,c_Q^1),\ldots,(z_k,c_Q^k))$，其中$z_l$是在$F_i$里$\bar{z}$中的；$\mu(z_l)$的局部候选$C(\mu(z_l))$是在$F_i$中的；有G中$z_l$的$c_Q^l$邻居图$G_{\bar{z}}[z_l]$；$G_{\bar{z}}[z_l]$中的边界节点$B_{\bar{z}}[z_l]$；部分工作单元$w_\varphi$（即分配到各个处理器的工作）使用$\langle v_{\bar{z}},\overline{|G_{\bar{z}}|},\overline{B_{\bar{z}}}\rangle$表示；若$C(\mu(z_l))=\emptyset$则$v_{\bar{z}}[z_j]$使用占位符⊥表示；$\overline{|G_{\bar{z}}|}$是$|G_{\bar{z}}[z_l]|$组成的列表; $\overline{B_{\bar{z}}}$是$B_{\bar{z}}[z_l]$组成的列表，表示丢失数据；

若对于每个$v_{\bar{z}}[z_l]$都一个来自M_i的单元w_φ^i的候选$v_{\bar{z}}^i[z_l]$则将$\langle v_{\bar{z}},|G_{\bar{z}}|,B_{\bar{z}}\rangle$添加到$W(\varphi,G)$中去；$|G_{\bar{z}}|$是$|G_{\bar{z}}^i[z_l]|$的总和；$B_{\bar{z}}$是$B_{\bar{z}}^i[z_l]$的联合。注i ∈ [1, n]且$\varphi\in\Sigma$.

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。