GPU 加速的贝叶斯网络精确推理方法研究

肖 旭,慕德俊,张慧翔,陈春雷
(西北工业大学 自动化学院,陕西 西安 710072)

摘 要:对于复杂输入的贝叶斯网络,精确推理时间较长。文中针对贝叶斯网络精确推理中的团树传播算法,提出了一种基于 CPU-GPU 异构计算平台的并行化方法。首先研究团节点间信念势更新方式,提出了节点级并行化方法加速更新过程;其次,提出了利用计算复杂度的优先级队列方法,通过拓扑级并行化加速全局推理过程;最后,通过输入不同团树结构—线性结构、两分支二叉树结构和完全二叉树结构验证算法加速效果。实验结果表明,节点级并行化方法对线性结构有明显加速效果,拓扑级并行化对两分支二叉树和满二叉树结构有明显加速效果。

关键词:贝叶斯网络;团树传播算法;GPU 加速;并行化信念传播
中图分类号:TP391 **文献标识码**:A **文章编号**:1673-629X(2014)10-0001-05
doi:10.3969/j.issn.1673-629X.2014.10.001

Research on Bayesian Network's Exact Inference Using GPU Acceleration

XIAO Xu, MU De-jun, ZHANG Hui-xiang, CHEN Chun-lei
(College of Automation, Northwestern Polytechnical University, Xi'an 710072, China)

Abstract: For Bayesian network with complex input, the inferring is time-consuming. Aiming at the junction tree propagation algorithm of Bayesian network' exact inference, a parallel method is proposed based on the CPU-GPU heterogeneous computing platform. Firstly, the updating method between junction tree nodes is investigated, and node-level parallelism method is proposed to accelerate the updating. Secondly, the priority queue method is presented according to the computational complexity to achieve topological-level parallelism to accelerate the global inference process. Finally, speedup of the proposed method is verified on various input junction trees including linear structure, two branches of binary tree structure and complete binary tree structure. The experimental results indicate that the node-level parallelism method can significantly accelerate the linear structure, and topological-level parallelism is effective for the two branches of binary tree and complete binary tree.

Key words: Bayesian network; junction tree propagation algorithm; GPU acceleration; parallel belief propagation

0 引 言

贝叶斯网络(Bayesian Network)是不确定性知识表达和推理领域的一种有效的理论模型,其中网络推理是贝叶斯网络的核心环节,推理速度与精度决定了网络应用的实时性与准确性。目前基于贝叶斯网络的推理算法主要有两种,即精确推理算法与近似推理算法。近似推理力求在较短的时间内给出满足精度要求的结果,但由于贝叶斯网络推理结果本身的不确定性以及近似推理产生的计算误差,使得该方法在推理精度要求较高的情况下难以达到预期要求。精确推理旨在将输入证据与网络信息传播至网络中的全部节点,信息传播的完整性使得推理结果更加准确,对节点信息的查询更加全面。但是,随着网络中节点数目与节点信息的增加,精确推理的计算复杂度呈指数增加,研究证明复杂贝叶斯网络的精确推理是 NP-Hard[1]。

精确推理中,团树传播算法应用最为广泛[2]。该算法的优势在于:可缩小贝叶斯网络规模,同时团节点的信念势传播运算简单。Huang Jinbo[3]等将贝叶斯网络编译在逻辑电路中,实现了基本推理方法中最大后验概率查询(MAP)方法的加速;Wu Dan[4]等改进了团树传播算法,建立了基于团树的二次结构实现算

收稿日期:2013-11-05 **修回日期**:2014-02-11 **网络出版时间**:2014-07-17
基金项目:教育部高校博士点基金(20126102110036);西北工业大学基础科研基金(JC20110264)
作者简介:肖 旭(1988-),男,硕士研究生,研究方向为 GPU 加速方法策略;慕德俊,教授,博士生导师,研究方向为网络信息安全、控制理论应用。
网络出版地址:http://www.cnki.net/kcms/detail/61.1450.TP.20140717.1231.031.html

法加速。近些年来,基于GPU的通用并行计算发展迅速,目前已形成了CPU-GPU异构计算平台。GPU适用在高并行化及数据间依赖关系不强的情况下处理大容量数据,在具体应用中,GPU作为CPU的加速部分,提供了强大的并行处理能力[5]。传统精确推理算法只在CPU中进行,由于CPU本身计算能力的限制,推理时间不可估计。而CPU-GPU异构计算平台的并行计算能力使复杂贝叶斯网络的精确推理成为可能,该平台基于多核并行计算对复杂运算的快速处理能力,实现对精确推理的加速。GPU多线程并行计算方法适用于团节点间条件概率表各条目更新及团树结构中各子树的并行推理。

文中研究了贝叶斯网络精确推理算法在CPU-GPU异构通用计算平台下的并行化实现问题,首先介绍贝叶斯网络团树传播算法,分析算法结构;其次在CPU-GPU异构计算平台上实现并行的团树传播算法,包括节点级并行和拓扑级并行;最后通过不同输入网络,验证并行算法的加速性能。

1 贝叶斯网络团树传播算法

贝叶斯网络是包含一个条件概率表的有向无环图,主要由三部分组成:

(1)代表随机变量的节点;

(2)代表节点间依赖关系的有向边;

(3)代表节点间依赖强度的条件概率表。

其中,证据节点是已经被观测到或实例化的节点。网络推理是在给定网络结构和证据条件下,通过联合概率公式,快速计算出目标节点发生概率的过程[6]。团树传播算法是一种典型的精确推理方法,主要思想是将贝叶斯网络转换为一次团树结构,通过定义在团树上的消息传递过程进行概率计算。

团树定义为 $J=(T,P)$,T 代表树结构,P 代表树中的参数。每个顶点 C_i 即团节点 J 是一系列随机变量的集合。假定 C_i 与 C_j 是相邻的,它们之间的分离器定义为 $C_i \cap C_j$,P 是一个信念势集。C_i 的信念势,表示为 φ_{C_i} 是团 C_i 中随机变量的联合分布概率。对于一个具有 w 个变量的团节点,每个变量有 r 个状态,C_i 信念势的条目数为 r^w 。在团树结构中,推理过程如下:

首先,给定证据 $E=\{A_i=a\}$,$A_i \in C_y$,通过实例化 A_i ,E 被 C_y 吸收并且标准化团内的其他节点。然后,该证据由 C_y 传播至所有的邻接团节点 C_x ,令 φ_y^* 表示吸收证据 E 后的 C_y 的信念势,φ_x 是 C_x 的信念势。证据的传播可表示为[1]:

$$\varphi_S^* = \sum_{y \setminus S} \varphi_y^* ,\varphi_x^* = \varphi_x \frac{\varphi_S^*}{\varphi_S} \tag{1}$$

其中,S 是团 x,y 的分离器;$\varphi_S(\varphi_S^*)$ 表示原始(更新后)的 S 信念势;φ_x^* 是 C_x 更新后的信念势。

基于上述的证据传播方法,团树传播算法分为两个阶段:信念的收集阶段及信念的分发阶段[7-8]。在信念的收集阶段,证据由叶子节点传播至根节点,每个团节点 C 吸收该节点与孩子间的分离器的信念,更新自身的信念势。然后,C 用更新后的信念势 φ_y^* 更新自身与父节点间的分离器。信念的分发阶段除了证据的传播方向是从根节点到叶子节点之外,其余部分与收集阶段相同。

2 CPU-GPU异构平台的并行团树传播算法

在CPU-GPU异构平台中,GPU作为协处理器提供大规模的并行计算能力,其程序设计基于NVIDIA CUDA编程模型[9-10]。CUDA(Compute Unified Device Architecture,统一计算设备架构)是一种将GPU作为数据并行计算设备的软硬件体系。CUDA执行模型中,程序首先由串行CPU部分开始执行,申请GPU资源(计算资源—线程块数,存储资源—全局内存、共享内存)后,CPU将待计算数据拷贝至GPU全局内存;然后,GPU执行核函数Kernel进入并行部分,此时CPU可同步等待,也可异步执行其他串行指令。核函数调用后回送数据至CPU;最后,CPU释放GPU资源。CUDA架构中,GPU端线程组织结构分为三层,最小单位为线程,一组线程构成线程块,一组线程块构成网格。除了编程和执行模型,CUDA同样有存储模型。同一线程块内的线程可访问共享内存,所有线程均可访问全局内存,由于线程访问共享内存速度优于全局内存,因此GPU核函数设计时,先将数据从全局内存读入共享内存,再将计算结果写回全局内存,达到访存延迟最低[11]。

团树传播算法最耗时部分是节点间信念传播,全局推理速度依赖于节点间信念传播。信念传播过程包括节点信念更新和信念边缘化,信念更新过程中信念势表各条目的概率计算适用于GPU并行化。这里基于CPU-GPU异构计算平台,加速团树传播算法中节点间信念传播过程,系统总体设计如图1所示。其中团节点队列由待计算节点组成;调度器的作用是从队列中取出合适的节点数发送给团节点发送器;团节点发送器将待处理节点相关数据发送至GPU。处理流程如下:

首先调度器收到队列非空信号后,获得当前待处理队列,根据当前GPU的处理能力,发送合适的团节点数至团节点发送器;然后,团节点发送器准备当前待处理节点需要的数据发送至核函数,核函数调用GPU计算资源更新节点信念势表,GPU计算完成后异步通

知 CPU 收集更新后的信念势表,之后 CPU 回调边缘化函数更新与父节点的分离器;最后发送父节点信息至信号处理单元,信号处理单元收到节点信息后令该节点入队列并通知调度器当前队列非空。

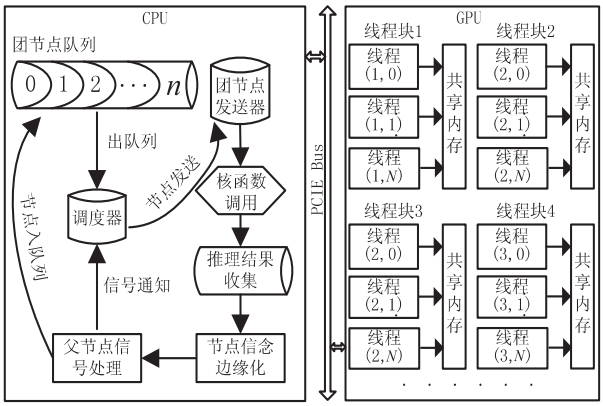

图 1　基于 CPU-GPU 异构计算平台并行推理

为了使原算法获得更高的并行性,适用于 GPU 计算,这里提出两点并行化方法:节点级并行化和拓扑级并行化。节点级并行化是信念在一对节点间传播时,利用 GPU 并行更新信念势表;拓扑级并行化基于全局团树结构,使团树各分支并发推理,达到全局推理时间最短。

2.1　节点级并行化

团树结构中,每个团节点的联合分布都由信念势表给出,信念传播过程依赖团节点信念势表的更新过程,每个团节点首先吸收子节点间分离器的信念势,更新自身信念势表;然后边缘化该节点与父节点的分离器。分离器是两个团节点间共享的随机变量集,分离器的信念势表可通过其连接的两个相邻节点获得[12]。例如,A,B 是两个相邻团节点,其分离器 $S=A\cap B$,分离器 S 的信念势表是通过边缘化 A,B 的信念势表取得。考虑 A 向 B 传播信念,过程描述为:首先,A 通过边缘化其信念势表得到分离器 S 的信念势 φ_S^*;然后,B 吸收 S 的信念势更新其自身的信念势表,即

$$\varphi_B^* = \varphi_B \frac{\varphi_S^*}{\varphi_S} \tag{2}$$

若一个团节点 C 包含 w 个随机变量,第 i 个随机变量有 r_i 个状态数,信念势表长度为 $L_c = \prod_{i=1}^{w} r_i$,若 C 与相连接的第 i 个分离器为 $S_i(i=1,2,\cdots,n)$,对应信念势表的长度为 L_i,则一次信念传播需要计算的次数 $N_t = \sum_{i=1}^{n} L_i$,对于单核 CPU 假设一个信念势表每个条目的计算时间为 t_0,一次信念传播的时间

$$t_c = N_t * t_0 = \sum_{i=1}^{n} \prod_{i=1}^{w_i} r_i * t_0 \tag{3}$$

由上可知,随着信念势表的不断增大,信念传播时间呈指数级增加。由于 GPU 拥有大量的线程计算资源,同时信念势表各条目的概率计算可并发执行,这里提出一种基于 GPU 多线程的节点级并行化方法。每个 GPU 线程负责信念势表的一个条目。首先,CPU 端将分离器的信念势表和待更新的团节点信念势表发送至 GPU 端;然后,CPU 端分配合适的线程块数用于计算当前任务,GPU 并发计算完成待更新节点的信念势表更新;最后,GPU 端将更新后的信念势表回传至 CPU,该团节点信念势表完成更新。假设 GPU 端每个线程计算信念势表的单个条目时间也为 t_0,理想情况下,对于 N_t 次计算需要的时间为 $t_g = t_0$,获得的性能提升

$$\tau = \frac{t_c - t_g}{t_c} = \frac{N_t * t_0 - t_0}{N_t * t_0} = 1 - \frac{1}{N_t} \tag{4}$$

GPU 端并行算法流程如下:

```
input:Node table Xp, Separator table Yp, Table length N
tid_in_grid = blockDim.x * blockIdx.x+threadIdx.x;
tid_in_block = threadIdx.x;
bid = blockIdx.x;
__shared__ Xp_s[BLOCKSIZE];
__shared__ Yp_s[BLOCKSIZE];
if tid_in_grid < N
Xp_s[tid_in_block] = Xp[tid_in_grid];
Yp_s[tid_in_block] = Yp[tid_in_grid];
__syncthreads();
Update Xp_susing Yp_s;
__syncthreads();
Xp[tid_in_grid] = Xp_s[tid_in_block]
end if
```

算法输入为待更新节点信念势表 Xp,子节点分离器信念势表 Yp,信念势表长度 N。首先,线程块内各线程根据自身全局索引从全局内存中读取信念势表条目到共享内存,由于 GPU 每次发射 32 个线程,需调用同步函数保证数据读入的完整性;然后,待更新节点信念势 Xp_s 吸收分离器信念势 Yp_s;最后,将共享内存中更新后的节点信念势表 Xp_s 写回全局内存供 CPU 读取。

2.2　拓扑级并行化

团树传播算法中,无论是信念的收集阶段还是分发阶段,本质是节点间的信念传播,即节点首先吸收与孩子节点间分离器的信念更新自身的信念势表,然后,边缘化更新的信念至父节点间的分离器。因此对无直接依赖关系的节点,信念可并行传播。在团树结构中,根节点相同的子树节点与根节点所在树的其余节点存在依赖关系,但各子树节点间无依赖关系,因此不同子树的信念可并行传播。如图 2 所示,传播过程①,②(或①,③)的两对节点间,无依赖关系,可并行传播;但过程②,④间不可并行,因为对于节点 f 只有在吸收

h, i 的信念后,才能更新父节点信念势,存在依赖关系。这里提出拓扑级的并行化方法,即让不同子树无依赖关系的节点间并行传播信念。图 2 中,只有当③、④都完成时,②过程才能开始,即对于同一节点,其子树信念全部传播至该节点时,后续的信念传播才能进行。

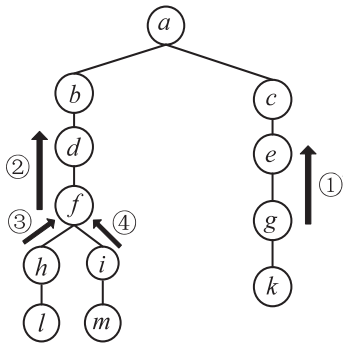

图 2　并行信念传播

为达到全局推理时间最短,应使各子树信念传播时间最短,使整个团树的推理过程按层级进行。为实现这一目标,提出基于计算复杂度的优先级队列的方法。对于团树结构的各个分支,其节点总数和各节点宽度(信念势表长度)是影响该分支信念传播速度的关键因素,定义计算复杂度 $O(\tau)=1$,对于团树结构中的任一节点,其计算复杂度与该节点所在子树的祖先点数和祖先节点宽度有关。由团树结构和计算复杂度可得对应的计算树:

$$T_c=\{T_1,T_2,\cdots,T_i,\cdots,T_n\},T_i=\{O(\tau_1),O(\tau_2),\cdots,O(\tau_i),\cdots,O(\tau_n)\} \quad (5)$$

其中,T_i 为计算树 T_c 的一子树;$O(\tau_i)$ 为上任一节点 T_i 的计算复杂度。

基于计算树结构的信念收集阶段可描述为:首先在遍历树的基础上记录所有叶子节点,叶子节点入待处理队列 Q;然后,以各节点的计算复杂度 $O(\tau_i)$ 为优先级,按优先级 $O(\tau_i)$ 对 Q 排序,由高到低依次从队列中取出合适的节点发送给 GPU,GPU 根据节点级并行化策略对节点进行处理,回送处理结果;最后,CPU 存储节点已更新的信念势表,边缘化与父节点间的分离器,该节点的父节点入待处理队列,直到整个推理过程完成。

为了达到 CPU 与 GPU 的异步并行执行,采用基于流(CUDA Stream)的开发模式[13-15]。CUDA Stream 是按顺序执行的指令序列,不同流之间是乱序执行或并行执行。

这样,一个流的计算与另外一个流的数据传输同时进行,在一定程度上隐藏数据 I/O 的时间开销。节点信念传播采用基于流的异步执行方式。基于流的拓扑级并行算法流程如下:

```
input:Node queue Taskqueue,Queue lengthN,Junction tree J
Scheduler:
pthread_cond_wait(Taskqueue->event);
std::sort(Taskqueue, Taskqueue+N, Complexity_com);
Query GPU available resources;
nodes = Taskqueue.Dequeue();
Send nodes to Node Sender;
Node Sender:
for i=0:N
cudaMalloc(Taskqueue(i)->data);
cudaStreamCreate(Taskqueue(i)->stream);
cudaMemcpyAsync(Taskqueue(i)->data, HostToDevice,
Taskqueue(i)->stream);
Kernel<<<n_blocks,n_threads,Mem,Taskqueue(i)->stream
>>>>(data);
cudaMemcpyAsync(J->node(Taskqueue(i))->P, DeviceTo-
Host, Taskqueue(i)->stream);
cudaStreamAddCallback(task->stream,Marcallback);
end for
Signal Processing:
while uitflag ==0
sigwaitinfo(&set,&info);
if info.si_signo == SIG_NEXT
if info->parent == root
if ncrement(root) quitflag = 1;
else pthread_cond_signal(Taskqueue->event);
end if
else
Taskqueue.Enqueue(task->parent);
pthread_cond_signal(Taskqueue->event);
end if
end if
end while
```

算法输入为待处理节点队列 Taskqueue,队列长度为 N,团树结构为 J。首先调度器(Scheduler)收到队列非空信号后,获得当前待处理队列,按优先级 $O(\tau_i)$ 将队列排序,根据当前 GPU 处理能力,按优先级由高到低的顺序,发送节点至发送器(Node Sender);然后,节点发送器为每个待计算节点创建流,各个流并行处理对应节点,先将节点数据拷贝至 GPU,GPU 对节点概率更新后回送至 CPU 完成节点信念更新;最后流回调边缘化函数 Marcallback 完成该节点信念传播。在节点边缘化中,若节点的父节点无子树,该节点直接完成边缘化计算发送父节点信息至信号处理线程(Signal Processing);若父节点存在子树,优先到达父节点的子节点进行边缘化计算,更新父节点标志位,向等待父节点边缘化完成的子节点发送开始信号;后续子节点通过查询父节点标志位,等待父节点中另一子节点边缘化完成,在收到开始信号后,进行边缘化计算,最后发

送父节点信息至信号处理单元,信号处理单元使新节点入队列。该机制避免了重复发送父节点信息引起的计算冗余。

3　实验方法

为了验证上述方法的加速效果,实验平台使用 NVIDIA GTX660 图形处理器和 Intel Core E7500 组成的异构平台。为了验证节点级并行化和拓扑级并行化对团树传播算法的加速效果,这里采用两种实验方法进行验证。

实验 1:节点级并行化实验方法。

在团树结构中,节点信念势表长度取决于团节点宽度,影响团节点宽度的参数是团节点包含的随机变量 w 和每个随机变量对应的状态数 r,这里固定随机变量 w,只改变对应状态数 r。为了排除其余并行化方法的影响,输入团树结构为线性结构。图 3(a)是在给定网络结构不变的条件下,当团节点中各随机变量的状态数变化而节点包含的随机变量数不变时,CPU 与 CPU-GPU 异构平台的推理时间。

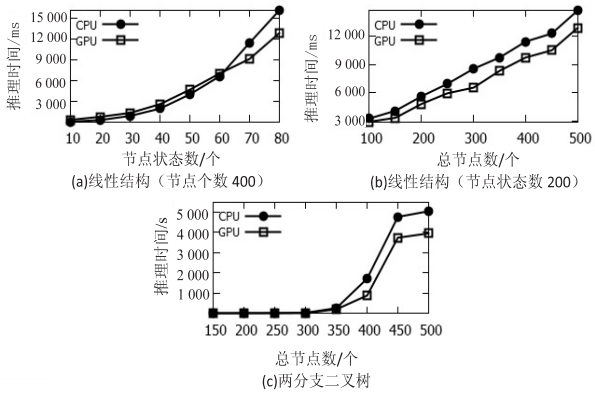

图 3　线性结构实验结果

图 3(a)中当网络结构恒为 400 个节点的线性结构时,拓扑结构不变,与原算法相比只有节点级并行化。当节点状态数较少时(10 ~ 50),CPU-GPU 异构平台计算时间大于 CPU,这是因为 GPU 利用率较低,参与并行更新信念势表的线程块数较少,启动 GPU、CPU-GPU 数据拷贝、GPU 计算的总耗时大于原推理计算时间。而节点状态数较多时(>60),参与计算的 GPU 线程块数增大,GPU 利用率得到显著提升,大量 CUDA 线程并发执行、隐藏了 GPU 全局存储器的访问延迟,加速了团树传播算法。

实验 2:拓扑级并行化实验方法。

影响拓扑级并行化的因素是团树结构中无依赖关系的节点对和节点宽度,团树的不同输入结构能够影响无依赖关系的节点对,改变输入团树结构和各团节点宽度可影响全局团树推理速度。这里输入三种团树结构:线性结构、两分支二叉树、满二叉树结构,来验证该方法实际效果。图 3、4 为给定输入结构分别为线性结构、二叉树结构、满二叉树结构时,团节点宽度不变,总节点数变化的推理时间对比。

图 3、4 当团节点包含的随机变量数和状态数不变时,改变网络结构会影响拓扑级并行化效果。图 3(b)是改变线性结构的总节点数,算法加速效果源于节点级并行化。图 3(c)输入结构为两分支二叉树,随着节点数增加,拓扑级并行化方法对于存在分支的团树结构具有一定加速效果,但由于分支数较少,优先级队列中待并行更新的团节点数不足,拓扑级并行化方法无法达到最佳效果。对于图 4 的满二叉树,总层数为 n 时,该层总节点数为 2^n 个,这些节点信念势表可并行更新,随着满二叉树层数(1 ~ 6)增加,加速效果得到提升,验证了拓扑级并行化方法的有效性。

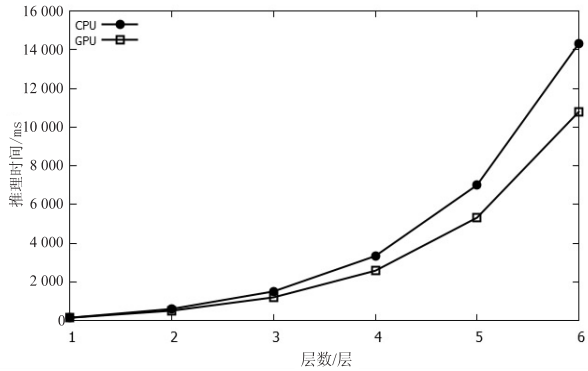

图 4　满二叉树结构实验结果

4　结束语

文中基于 CPU-GPU 异构计算平台提出了并行化的团树传播算法,实现了原算法的节点级并行化和拓扑级并行化。在节点级并行化中,首先研究了节点间信念传播的方法;然后提出了 GPU 多线程加速团节点信念势表更新的方法,达到了加速节点间快速传播信念的目的。在拓扑级并行化中,提出了基于计算复杂度的优先级队列方法,使不同子树无依赖关系的节点间并行传播信念,达到了全局推理时间最短。最后,通过实验证明两种并行化方法对团树传播算法有明显的加速效果。

参考文献:

[1] Xia Y. Parallel exact inference on a CPU-GPGPU heterogeneous system[C]//Proc of 39th international conference on parallel processing. [s. l.]:[s. n.],2010:61-70.

[2] Xia Yinglong, Prasanna V K. Node level primitives for parallel exact inference[C]//Proc of the 19th international symposium on computer architecture and high performance computing. Rio Grande do Sul:IEEE,2007.

(下转第 11 页)

tional conference on knowledge discovery and data mining. Washington, DC, USA: [s. n.], 2003: 286-295.

[35] ZhouXiaofeng, Gao Lin, Dong Anguo. An algorithm for finding frequent patterns in a large-parse graph[C]//Proc of international multi-conference of engineers and computer scientists. [s. l.]: [s. n.], 2007: 290-294.

[36] Huan Jun, Wang Wei, Prins J, et al. Spin: mining maximal frequent sub-graphs from graph databases[C]//Proc of KDD. [s. l.]: [s. n.], 2004: 581-586.

[37] Thomas L, Valluri S, Karlapalem K. MARGIN: maximal frequent sub-graph mining[C]//Proc of international conference on data mining. [s. l.]: [s. n.], 2006: 1097-1101.

[38] 李继腾,骆志刚,丁　凡,等. 最大频繁子图挖掘算法研究[J]. 计算机工程与科学,2009,31(12):67-70.

[39] 维基百科[EB/OL]. 2001. http://en. wikipedia. org/wiki/Graph_database.

[40] Vicknair C, Macias M, Zhao Zhendong, et al. A comparison of a graph database and a relational database[C]//Proc of the 48th annual southeast regional conference. Oxford: [s. n.], 2010.

[41] Holzschuher F, Peinl R. Performance of graph query languages: comparison of cypher, gremlin and native access in Neo4j [C]//Proc of international conference on extending database technology. [s. l.]: [s. n.], 2013: 195-204.

[42] Angles R. A comparison of current graph database models [C]//Proc of IEEE 28th international conference on data engineering. Arlington: IEEE, 2012: 171-177.

[43] Martinez-Bazan N, Gomez-Villamor S, Escale-Claveras F. DEX: a high-performance graph database management system [C]//Proc of 27th international conference on data engineering. Hannover: IEEE, 2011: 124-127.

[44] Ke Yiping, Cheng J, Ng W. Correlation search in graph databases[C]//Proc of 13th ACM SIMKOD international conference on knowledge discovery and data mining. San Jose: [s. n.], 2007: 390-399.

[45] Chairunnanda P, Forsyth S, Daudjee K. Graph data partition models for online social networks[C]//Proc of 23rd ACM conference on hypertext and social media. [s. l.]: [s. n.], 2012: 175-179.

[46] 第 32 次中国互联网络发展状况统计报告[R]. 出版地不详:中国互联网信息中心,2013.

[47] Aggarwal C, Wang H X. Managing and mining graph data [M]. Berlin: Springer-Verlag, 2010.

[48] Soussi R, Aufaure M, Baazaoui H. Towards social network extraction using a graph database[C]//Proc of second international conference on advances in databases, knowledge, and data application. [s. l.]: [s. n.], 2010: 28-34.

[49] Kadge S, Bhatia G. Graph based forecasting for social networking site[C]//Proc of international conference on communication, information and computing technology. [s. l.]: [s. n.], 2011.

[50] Cao Jin, Gao Hongyu, Li L E, et al. Enterprise social network analysis and modeling: a tale of two graphs[C]//Proc of INFOCOM. Turin: IEEE, 2013: 2382-2390.

[51] 李孝伟,陈福才,刘力雄. 一种融合节点与链接属性的社交网络社区划分算法[J]. 计算机应用研究,2013,30(5):1477-1480.

(上接第 5 页)

[3] Huang Jinbo, Chavira M, Darwiche A. Solving MAP exactly by searching on compiled arithmetic circuits[C]//Proceedings of the twenty-first national conference on artificial intelligence and the eighteenth innovative applications of artificial intelligence conference. Boston: IEEE, 2006: 1143-1148.

[4] Wu Dan, Wu Libing. Hierarchical junction trees as the secondary structure for inference in Bayesian networks[C]//Proc of eighth ACIS international conference on software engineering, artificial intelligence, networking, and parallel/distributed computing. Qingdao: IEEE, 2007: 706-712.

[5] Xia Yinglong, Prasanna V K. Parallel exact inference on the cell broadband engine processor[C]//Proc of the 2008 ACM/IEEE conference on supercomputing. Austin, Texas: IEEE, 2008: 1-12.

[6] Zheng Lu, Mengshoel O J, Chong J. Belief propagation by message passing in junction trees: computing each message faster using GPU parallelization[C]//Proc of the 27th conference on uncertainty in artificial intelligence. [s. l.]: [s. n.], 2011.

[7] 李　刚. 贝叶斯网络在制动系统故障中的应用及系统开发[D]. 沈阳:东北大学,2007.

[8] 宫义山,高媛媛. 基于信息融合的诊断贝叶斯网络研究[J]. 计算机技术与发展,2009,19(6):106-108.

[9] 黄锦增,陈　虎,赖路双. 异构 GPU 集群的任务调度方法研究及实现[J]. 计算机技术与发展,2012,22(5):32-36.

[10] 卢风顺,宋君强,银福康,等. CPU/GPU 协同并行计算研究综述[J]. 计算机科学,2011,38(3):5-9.

[11] 姚　平. CUDA 平台上的 CPU/GPU 异步计算模式[D]. 合肥:中国科学技术大学,2010.

[12] 黄友平. 贝叶斯网络研究[D]. 北京:中国科学院研究生院,2005.

[13] 张　舒. GPU 高性能运算之 CUDA[M]. 北京:中国水利水电出版社,2009.

[14] 崔　晨. 基于 GPU 的 H.264 编码器关键模块的并行算法设计与实现[D]. 大连:大连理工大学,2012.

[15] NVIDIA. NVIDIA CUDAtoolkit version[EB/OL]. 2013. http://developer. nvidia. com/cuda-toolkit-32-downloads.

GPU加速的贝叶斯网络精确推理方法研究

作者： 肖旭， 慕德俊， 张慧翔， 陈春雷， XIAO Xu， MU De-jun， ZHANG Hui-xiang， CHEN Chun-lei

作者单位： 西北工业大学 自动化学院,陕西 西安,710072

刊名： 计算机技术与发展 ISTIC

英文刊名： Computer Technology and Development

年，卷(期)： 2014(10)