P2P 环境中的 skyline 查询综述

孙 志 孙雪姣

(烟台大学计算机与控制工程学院 山东 烟台 264005)

摘 要 随着数据规模的增长以及网络技术的发展,对等网络(P2P)作为一种分布式信息共享与搜索的平台引起了越来越广泛的关注。基于对等网络高度动态、高度分散、扩展性强等特点,P2P 上的 skyline 计算方法不仅需要满足集中式 skyline 计算方法的各种要求,还需要考虑减小网络通讯量、减少平均节点访问数、保持负载平衡等。文中对这个发展领域的最新技术进行了研究,并且描述了分布式 skyline 方法的目的和主要原理,概括了适用于 P2P 环境中的现有方法,并进行了性能比较分析。最后,给出了 P2P 环境 skyline 计算的未来发展方向。

关键词 skyline 处理,skyline 变化,对等网络,结构化 P2P,非结构化 P2P

中图法分类号 TP3-05 **文献标识码** A

Survey of Skyline Processing in P2P Environments

SUN Zhi SUN Xue-jiao

(School of Computer and Control Engineering, Yantai University, Yantai, Shandong 264005, China)

Abstract With the growth of data scale and the development of network technology, peer-to-peer (P2P) has attracted more and more attention as a platform for distributed information sharing and searching. Based on the highly dynamic, highly decentralized and extensible features of peer-to-peer networks, the skyline calculation method on P2P not only needs to meet the requirements of centralized skyline calculation method, but also needs to reduce the network traffic, reduce the average number of nodes, maintain load balancing and other important metrics. This paper examined this ongoing research area so that readers can easily understand the most advanced overview. This paper described the purpose and main principles of the distributed skyline method, and summarized the existing methods applicable to the P2P environment and provided performance comparison analysis. Finally, this paper gave the future development direction of skyline calculation in P2P environment.

Keywords Skyline processing, Skyline variants, P2P, Structured P2P, Unstructured P2P

1 引言

在将 skyline 查询引入数据库研究之前,我们将其称为最大向量问题或帕累托最优问题。近年来,skyline 查询处理已经成为数据库研究中的一个重要问题。skyline 运算的普及主要是由于其对决策应用的适用性。在这样的应用中,数据库元组表示为一组多维度的数据点,而 skyline 集则包含那些在不同维度上都处于最佳权衡的点。

skyline 处理首先在单一数据库环境(即集中式环境)中进行研究。随着数据越来越多地以分布式方式存储和处理,对分布式数据的 skyline 处理在近年来引起了极大的关注。本文对 P2P 环境中 skyline 查询处理的方法进行了概述。对等网络(P2P)是指用于组织和搜索分布在独立资源上的大型数据存储库。比如一个基于网络的酒店预订系统,其服务器分散在全国各地;旅行社可以通过服务器宣传他们的酒店,其中每个服务器可以为全国各地的酒店提供报价,例如在北京的服务器可能为拉萨、上海和青岛的酒店提供不同的报价。这样的系统可以通过酒店的数据库为用户提供预订服务,而不需要用户在每个服务器上进行注册。用户能够对服务器网络提出个性化的查询(如 skyline 查询),并得出与用户查询匹配的元组。

本文第 2 节给出了 skyline 处理和对等网络(P2P)的基础知识;第 3 节介绍了分布式 skyline 处理的目的和主要原理,以及 P2P 环境中的 skyline 处理方法,并对其进行了性能比较分析;第 4 节给出了 P2P 环境中 skyline 的发展趋势。

2 基础知识

2.1 skyline 查询处理

在研究分布式 skyline 处理之前,首先介绍 skyline 的相关定义以及 skyline 查询的演变。

(1)skyline 查询

skyline 运算符在文献[1]中首次引入,通过可选择的 SKYLINE OF 扩展了 SQL 中的 SELECT 语句,以便用户可以指定维度以及使用 skyline 查询的函数(MIN, MAX, $DIFF$)进行查询。给定一组 d 维$\{d_1, d_2, \cdots, d_d\}$的数据空间 D 以及基数为 n 的 D 上的数据集 S,点 $p \in S$ 可以表示为

本文受山东省自然科学基金(ZR2014FL009),山东省高等学校科技计划项目(OJ14LN23)资助。

孙 志(1989—),男,硕士生,主要研究方向为偏好处理,E-mail:sunzhi2016@126.com;孙雪姣(1979—),女,硕士,副教授,CCF 会员,主要研究方向为人工智能偏好处理,E-mail:sunxuejiao6@sina.com(通信作者)。

$\{p_1, p_2, \cdots, p_d\}$，其中 p_i 是维度 d_i 上的值。在不失一般性的情况下，假定任一维度 d_i 的值 p_i 大于或等于 0($p_i \geqslant 0$)，并且对于所有维度，最小值是最优的。

定义 1(skyline) 如果在每个维度 $d_i \in D$ 上都有 $p_i \leqslant q_i$，并且至少在一个维度 $d_j \in D$ 上，有 $p_i < q_i$，则称点 $p \in S$ 支配点 $q \in S$，表示为 $p \prec q$。skyline 是一组不受任何其他点支配的点的集合，即 $SKY(S) \subseteq S$。$SKY(S)$ 中的点称为 skyline 点。

skyline 查询的概念可以扩展到子空间，其中子空间的 skyline 查询仅指用户选择的属性子集。每个非空子集 $U(U \subseteq D)$ 均为数据空间 D 的子空间。D 是数据集 S 的完整空间。子空间 U 的 skyline 集是一组不被子空间 U 上的任何其他点支配的点的集合，记为 $SKY_U(S) \subseteq S$。

skyline 查询也可用于有约束的情况，其中每个约束均被表示为沿着 d 维度的范围，并且所有约束的连接在 d 维属性空间中形成超矩形。一个有约束的 skyline 查询返回满足给定约束的 skyline 集。给定约束，从满足约束的所有点中检索出 skyline 集。

定义 2(动态 skyline) 给定数据空间 D、数据集 S 以及 m 维函数 $f_1, f_2, \cdots, f_m$，其中每个函数 $f_i (1 \leqslant i \leqslant m)$ 采用数据点的坐标作为参数，则动态 skyline 查询根据由函数 $f_1, f_2, \cdots, f_m$ 定义的新数据空间返回 S 的 skyline 集。

(2)Skyline 查询的演变

Borzsony 在文献[1]中首次介绍了 skyline 运算符，并提出了两个基本的内存算法：BNL(块嵌套循环)和 D&C(分治法)。BNL 算法使用块嵌套循环将数据库中的每个元组与其他每个元组进行比较，仅当元组未被其他任何元组支配时才作为结果。D&C 算法采用递归方法将输入元组划分为更小的集合，分别计算每个集合的单独 skyline，并将它们合并成最终的 skyline。文献[2]中的 SFS(Sort-First-Algorithm)以及文献[3]中的 LESS 根据单调函数对元组进行排序来提高 BNL 的性能。基于排序方法的主要原则是，如果元组基于单调评分函数进行排序，则元组不能被其后的元组支配。

Borzsonyi 等首次提出使用索引结构的 skyline 查询处理，主要思想是使用一个索引来确认元组间的维数以及从早期阶段修剪元组。Borzsonyi 等提出了使用 R-树的算法，即 NN 搜索(最近邻)[4]和 BBS(分支和约束 skyline)[5]。这些算法首先计算原点的近邻，以确保其是 skyline 结果集的一部分。显然，由近邻支配的区域可以安全地从结果集中删除。通过重复查看非支配区域中的下一个近邻来确定完整的 skyline 集。结果表明，BBS[5]通过使用 R-树索引保证了数据集的最小成本 I/O。

Papadias 等首先介绍了 skyline 运算符的不同变化，如约束、子空间和动态 skyline 查询。文献[6-7]中也讨论了有约束的 skyline 查询。文献[8]主要从查询语义的角度讨论了子空间 skyline 查询。文献[9]也研究了子空间 skyline 查询，并提出了 SUBSKY 算法。

2.2 对等系统

对等系统 P2P 是通过直接交换来共享资源的分布式计算机体系结构，不需要中央协调服务器。在 P2P 系统中，彼此连接的对等体以分布式和自组织方式在网络中组织并共享资源，同时尊重对等体的自主权，这意味着对等体相对于决策是独立的，比如把哪部分数据传递给邻居、是否更新本地数据、何时离开或加入网络等。P2P 系统的主要特点是能够适应对等体故障(容错)并容纳大量参与的对等体(可扩展性)，同时将网络的性能保持在可接受的水平并维持对等体的连接。

一般情况下，所有对等体在它们执行任务和功能方面都是对等的，每个对等体都有共享的文件或数据。两个互相连接的对等体称为邻居，对等体邻居的数量称为它的度。任何对等体都可以发出查询请求以检索数据，发出查询的对等体称为查询对等体。查询信息仅在相邻的对等体之间传送。通过发出查询并将其发送给其邻居，每个对等体原则上都可以访问系统中的所有数据，甚至可以查询位于几跳距离(查询为到达其目的地被转发的次数)的对等体上的数据，而不需要查询对等体进行查询规划和优化。然而，查询相应对等体上的数据跳数越大，查询执行的成本就越高。

P2P 系统根据其位于网络中的方式分为两类：结构化 P2P 系统和非结构化 P2P 系统。结构化 P2P 系统以控制方式构建，并且在对等体数据和网络拓扑之间增加相应的关系。加入网络的对等体提供的数据根据通用的规则进行重新分布，该规则将数据映射到对等体上，并确定各对等体负责数据的相应部分。与结构化 P2P 系统相反，非结构化 P2P 系统中的对等体保持较高的自主性，因此它们保持数据独立，即对等体提供的数据保持于对等体中，并且一般不会重新分发给系统中的其他对等体。此外，网络结构仅依赖于对等体对邻居的选择，并且对等体可任意选择其邻居。由于网络拓扑和存储的数据对象之间不存在任何关系，对等体仅具有有限的关于存储在其他对等体的数据对象的信息。因此，搜索可能会使得对等体向它们所有的邻居检索与查询匹配的数据对象——这种方法被称为洪泛。为了降低网络的昂贵成本，非结构化 P2P 系统中的对等体通常需要构建和维护查询路由[10-11]。由于缺乏全局知识，非结构化 P2P 系统中的每个对等体必须根据本地可用的信息对查询路由进行优化。

3 P2P 环境中的 skyline 处理

3.1 分布式 skyline 处理的目的和原理

本节首先介绍分布式 skyline 处理的目的，然后介绍分布式 skyline 处理的主要原理。

分布式 skyline 处理的主要目的是最小化查询的执行时间。我们把从发出查询请求到所有结果都返回给用户之间经过的时间称为执行时间。影响执行时间的几个因素包括：

1)处理时间。处理时间是执行时间的一部分，是对等体在本地进行评估查询处理的时间。因此，最小化执行时间需要最小化处理时间，后者可以通过最小化每个对等体的单独处理时间来实现。

2)查询对等体的数量。如果对等体的本地数据属于结果集或对等体是传送相关数据的路径的一部分，则对等体对结果有贡献。查询对等体的数量是指处理该查询的对等体的数量。通过避免接触对 skyline 结果集无贡献的对等体来最小化查询对等体的数量。

3)网络流量：旨在最小化传输的数据量，以减少网络的传输时间，从而减少执行时间。因此，尽可能多地在本地进行评

估查询,而不是将整个数据集传输到查询对等体。

然而,这些因素往往是相互矛盾的,我们需要在它们之间找到一个平衡点。

下文给出了用于分布式 skyline 处理中的 3 个主要原理:1)skyline 运算符的加性;2)通过过滤修剪本地的数据;3)基于本地信息修剪对等体。

分布式 skyline 处理中的关键属性是 skyline 运算符的加性。给出对应 n 个对等体的 n 个数据集 $S_i(1\leqslant i\leqslant n)$,对 skyline 运算符进行如下评估:1)对 n 个数据集的并集进行评估;2)先在每个数据集上评估,然后再对结果集的并集进行评估,skyline 集是一样的。

定义 3(skyline 运算符的加性)　给定数据集 S 和 n 个数据集 S_i,使得 $S=S_1\cup\cdots\cup S_n$,以下等式成立:$SKY(S_1\cup\cdots\cup S_n)=SKY(SKY(S_1)\cup\cdots\cup SKY(S_n))$。

分布式 skyline 处理的第二原理是通过过滤(或过滤点)来对数据进行修剪,主要思想是对等体在查询过程中也转发附加信息(过滤信息),其简要地描述了已经检索的本地结果集。通常,该信息是局部 skyline 集的子集,称为过滤点。

最后,用于分布式环境中 skyline 处理的第三原理是利用支配关系修剪对等体。在大多数方法中,每个对等体使用本地 skyline 点,以确定邻居对等体是否存储未被支配的数据点。在每个邻居对等体上存储的数据信息称为路由信息,其用于对等体的修剪。在非结构化 P2P 系统中,对等体可以通过每个对等体的概要信息来检测邻居对等体是否可被修剪。在结构化 P2P 系统中,路由信息本质上是数据首次分布的规则。该规则与网络结构一起产生可以修剪对等体的信息。此外,本地 skyline 点可以与过滤信息组合,使其可以修剪邻近对等体。

3.2　P2P 环境中的 skyline 处理方法

本节介绍 P2P 环境中处理 skyline 查询问题的主要方法。我们将这些方法分为两类:第一类是适用于结构化 P2P 系统的方法,第二类是适用于非结构化 P2P 系统的方法。

3.2.1　结构化 P2P

本文第 2 节已经介绍了结构化 P2P 系统的主要特点。在分布式 skyline 处理中,覆盖网络用于查询路由,而过滤点用于提高整体查询处理的性能。下文将详细介绍现有的方法,并对主要方法进行比较分析。

(1)DSL

Wu 等[7]提出 DSL(Distributed Skyline)来计算 CAN 网络中带有约束的 skyline 查询[12]。DSL 使用 CAN 覆盖将数据映射到区域,并将这些区域分配给对等体(基于内容的数据分区)。给定带有约束的 skyline 查询,可以直接修剪那些对约束没有贡献的对等体,并构建连接所有剩余对等体的自组织组播树。通过以多播层次结构方式组织对等体并使用它来传播查询和局部结果集,每个查询对等体在其本地数据上计算 skyline 集,中间对等体合并其邻居的结果集。图 1 显示了在运行时构建的层次结构,其中查询区域为((0.3,0.3),(0.9,0.9)),对等体 3 保证含有全局 skyline 集的部分数据,因此作为多播层次结构的根。查询沿着层次结构的边传播;对等体执行本地计算,并将结果按相同的路径返回——在返回的路径上,利用 skyline 的加性合并结果集。

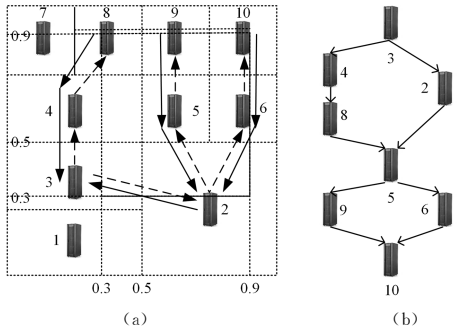

图 1　DSL:CAN 区域及其多播层次结构

在查询处理期间,DSL 构建了多播层次结构,其中包含左下角约束区域的对等体为根。存储在该对等体的数据点保证属于全局 skyline 集,因为这些数据点不能由存储在其他对等体上的点来支配。在 DSL 中,层次结构是动态构建的,每个查询对等体通过使用动态区域划分和编码来决定查询哪些相邻对等体。因此,收到查询的对等体先等待接受在层次结构之前的所有相邻对等体的本地 skyline 集;然后根据其本地数据和接收的数据来计算 skyline 集;此后将本地 skyline 点转发给相邻对等体,使得仅有数据点互不支配的对等体被并行地查询,被本地 skyline 点支配的邻近对等体不被查询,因为它们对全局 skyline 集没有贡献;最后将所有的本地结果集在不再转发查询的对等体上相加,得到全局 skyline 集,并将全局 skyline 集返回给查询的发起者。

(2)SSP 和 skyframe

Wang 等[13]提出了 SSP(Skyline Space Partitioning),并将其用于在 BATON[14]网络中 skyline 查询的分布式处理。BATON 中的对等体以平衡二叉树的结构覆盖网络,其中每个对等体负责数据空间中的一个区域。通过使用分解和合并的技术可使得对等体之间达到负载均衡。随机从对等体中动态地采样以检测负载是否均衡,如果不均衡,则使用分解和合并的技术将数据迁移到其他对等体,以抵消不均衡。

由于 BATON 网络最初是为一维数据设计的,因此 Wang 等通过使用 Z-曲线将多维数据空间映射为一维。图 2 显示了 BATON 网络中把数据区域映射到对等体的示例。通过将现有区域连续地分割成 2^n 维来创建 BATON 中的区域,其中每个区域由识别该区域的二进制字符串表示,并与该区域的 Z 次序一致,如图 2(a)所示。每个对等体都知道其区域的来源(即维度、分割值)。此外,每个对等体的路由表包含指向其他对等体(父级、子级、相邻对等体和同级别的其他对等体)的连接,使得查询有效地传送到特定的对等体。

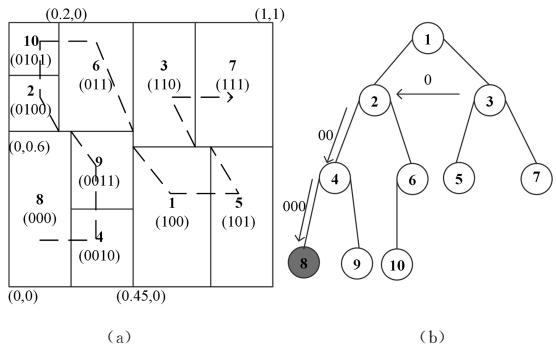

图 2　Skyframe:分配区域及其 BATON 覆盖中的路由

图 2(b)说明了 BATON 覆盖中路由的原理。假设对等体 3 需要检索分配给对等体 8 的数据。负责该区域的标识符从 0 开始,因为它包含在第一个分解的下半部分(在 $x=0.45$ 处分解)。因此,对等体 3 将查询转发到在其分配区域中具有前缀 0 的已知对等体(对等体 2)。使用相同的策略,对等体 2 将查询转发给对等体 4,对等体 4 再次将查询转发到持有查询数据的对等体 8。

BATON 网络中的 skyline 处理通过识别相关区域并将查询转发到负责这些区域的对等体。更准确地说,skyline 计算从对等体 p_{start} 开始,对等体 p_{start} 是包含数据空间来源的对等体。对等体计算包含在全局 skyline 集中的本地 skyline 点。然后,p_{start} 选择最主要的点 p_{md}(即支配最大区域的点[15])用于优化搜索空间和修剪支配区域。如果其区域的最佳点(即左下角)由 p_{md} 支配,那么可以安全地修剪该对等体。随后,查询对等体将查询转发给未修剪的对等体,并收集其本地 skyline 集。最后,查询启动器通过丢弃被支配的本地 skyline 点来计算全局 skyline 集。

Wang 等[16]通过提出 Skyframe 来概括 SSP。更详细地说,Wang 等提出了一种用于 skyline 处理的替代算法,而不需要在查询处理开始之前确定对等体 p_{start}。查询对等体将查询转发到边界对等体。至少在一个维度上具有最小值的区域的对等体被称为边界对等体。例如,在图 2 中,对等体 1,2,4,5,8 和 10 是边界对等体。一旦启动器接收到本地 skyline 的结果,它将计算 p_{md} 并确定是否需要查询其他对等体。如果需要,查询对等体查询其他对等体,并收集本地 skyline 的结果;当不需要查询其他对等体时,查询启动器计算全局 skyline 集。Wang 等在文献[16]中提出,Skyframe 也适用于 CAN 网络。

(3)iSky

Chen 等[17]提出了在结构化 P2P 系统中进行 skyline 处理的 iSky 算法。与 Skyframe[13,16]算法类似,iSky 依靠 BATON[14]网络覆盖,但采用 iMinMax 转换将数据分配给对等体,即先确定所有维中各元组的最大值,再将每个多维元组映射为一维值(*iMinMax* 值)。假设每个维度的范围标准化为[0,1),则 *iMinMax* 值被定义为该最大值和其维数的总和。每个对等体负责 *iMinMax* 值的特定非重叠范围,以便将每个数据点分配给特定的对等体。图 3 给出了 iSky 在 BATON 网络中的示例,也为每个对等体提供了 *iMinMax* 值的范围。应注意,iSky 假设在每个维度上较大的值是更优的。

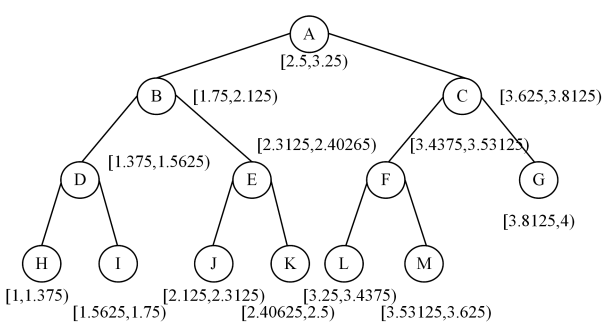

图 3 iSky:BATON 网络及分给对等体的 *iMinMax* 范围

给出一个 skyline 查询,iSky 首先确定一组初始 skyline 对等体,其中包括数据点在任何维度上具有最大值的对等体。选择这些对等体是因为在某个维度上具有最大值的点肯定是全局 skyline 集的一部分。由于使用 *iMinMax* 转换分发网络中的数据,初始 skyline 对等体负责接近整数的数据范围,例如负责范围[1.75,2.125)的对等体。此后,查询初始的 skyline 对等体,并将其本地 skyline 结果合并为一组初始的 skyline 点。然后,查询对等体确定阈值和过滤点。对等体接收到查询时,首先使用阈值来检查是否对数据进行修剪,此时对等体只需将查询转发给具有大于阈值的相邻对等体;否则,对等体对其本地数据进行查询处理,并使用过滤点来丢弃被支配的元组。最后,每个对等体在转发查询之前优化阈值和过滤点,并将本地结果集发送给查询对等体,然后合并该查询对等体和本地结果,并在所有查询对等体处理查询后获取全局 skyline 集。

(4)SSW

Li 等[18]使用基于底层语义覆盖网络(Semantic Small World-SSW)的空间分割法。在这样的网络中,多维数据空间被分割成不重叠的区域,也称为簇。基于语义标签将对等体分配给非重叠区域。对等体的语义标签与包含其最大数据簇质心的区域相对应。对于不包含在与对等体的语义标签相对应的区域中的数据,在该对等体处创建外部索引。对等体的外部索引保存区域中包含的数据的相关信息。每个对等体维护分配相同簇的其他对等体的连接,并维护在每个相邻簇中至少一个对等体的连接。

Skyline 的计算从肯定包含 skyline 点的区域开始,然后根据该区域提供的数据来评估 skyline 查询。我们把与原点最近邻的局部 skyline 点作为过滤点。更详细地说,我们不需要考虑由过滤点完全支配的所有区域,在查询剩余区域后,将所有本地结果集传送给查询启动器对等体,并检查相互优势,然后将全局 skyline 集返回给用户。

除了这种算法,Li 等还提出了一种不需要语义覆盖网络的近似算法。对等体仍然具有处理 skyline 查询的语义标签。由于假设对等体知道其邻居的语义标签,因此将查询转发给具有最佳语义标签的邻居,最佳语义标签是指最接近原点的语义标签。一旦对等体找不到比其更好的语义标签的邻居,则 skyline 计算结束,并将结果返回给用户。

比较:结构化 P2P 网络中方法的性能主要取决于:1)底层覆盖网络;2)发现数据的规则;3)算法的各个特征,如过滤、支配区域的修剪,以及结果传播。

DSL 和 SSP 通过发现负责数据空间原点附近区域的对等体来发起查询处理。对于一个 CAN 覆盖,需要 $O(d \cdot N_p^{\frac{1}{d}})$ 跳。此后,DSL 动态地构建多播层次结构,在最坏的情况下,它将具有 N_p 的最大长度。由于多播层次中的所有对等体代表相邻的分区,因此遍历的成本为 $O(N_p)$。另一方面,SSP 并行地连接所有的非支配区域,导致 $O(2 \cdot d \cdot N_p^{\frac{1}{d}})$ 跳的成本。Skyframe 先并行地查询所有的边界区域,然后查询剩余的未被支配的区域。因此,它产生与 SSP 相同的成本。在传输数据方面,DSL 通过多播层次结构(即多跳)传输所有的本地 skyline 点,而 SSP/Skyframe 仅路由过滤点,并通过直接传输(即单跳)收集本地结果。因此,DSL 比 SSP/Skyframe 消耗更高的带宽。

SSP/Skyframe 采用 Z-曲线方法对 BATON 进行一维映射。该算法与 CAN 网络上的 SSP/Skyframe 相同,但发现区域的成本发生了变化。如文献[16]所述,均匀分布的 skyline 查询的成本为 $O(1+2\cdot d\cdot(1-\frac{1}{\sqrt[d]{N_p}}))$跳。另一方面,iSky 采用将数据转换为一维值(使用所有维度中的最大值)进行分区而不是采用空间分区。如文献[19]所述,iSky 比 SSP/Skyframe 更好地利用了 BATON 协议,因为它更有意地在对等体上分发数据。对于查询处理,iSky 用 $O(d\cdot\log N_p)$的成本检测存储边界分区的对等体。首先,由这些对等体处理 skyline 查询,并且基于阈值修剪对等体。为了进一步减少传输的数据量,使用一个过滤点,类似于 SSP/Skyframe。然后,查询传播到那些基于阈值尚未修剪的对等体,直到所有对等体被查询或修剪为止。因此,与 SSP/Skyframe 相比,只有一些对等体被并行地处理,并且把查询转发给相邻的对等体。在每个步骤中,阈值和过滤点都被改进。iSky 的自适应过滤技术增强了其修剪能力,这可能导致比 SSP/Skyframe 更少的传输数据。然而,在最坏的情况下,iSky 需要连接所有的对等体。由于通过从 d 个对等体开始将查询转发给邻近对等体,因此我们将该成本估计为 $O(\frac{N_p}{d})$。

小结:如果底层网络是 CAN,则 SSP/Skyframe 更合适;而在底层网络为 BATON 的情况下,iSky 的性能优于 Skyframe。应当注意的是,我们在讨论中忽略了 SSW,原因是它在 CAN 和 BATON 中完全不同,因为 SSW 中的数据没有分配给对等体。相反,每个对等体与用于路由的语义标签相关联。

3.2.2　非结构化 P2P

与结构化 P2P 系统中的 skyline 查询处理相比,在非结构化 P2P 系统中,需要查询所有对等体或需要构建路由索引。应注意,一些方法假定查询对等体与所有其他对等体具有直接连接(完全连接的网络拓扑)。

(1)单过滤点(SFP)

Huang 等[15]假定移动设备通过自组织网络(MANET)进行通信,并且研究涉及空间约束的 skyline 查询。虽然他们主要研究移动环境,但自组织网络和非结构化 P2P 网络之间具有相似之处。因此,用于减少传输数据的技术(单过滤点-SFP)可直接应用于 P2P 网络。

SFP 的主要特点是使用本地 skyline 点作为过滤点来丢弃其他对等体的 skyline 点。过滤点的选择取决于支配区域的范围,支配区域是指由 skyline 点支配的区域。假设 skyline 点均匀分布,则较大的支配区域意味着更有可能支配其他点。当对等体接收到查询请求时,它首先在本地进行处理,然后通过将过滤点附加到查询中将其传播到相邻对等体。在将本地结果发送回查询对等体之前,使用过滤点丢弃本地 skyline 点。如果本地 skyline 点具有更大的支配区域,则每个对等体将更新过滤点。

(2)DDS

Hose 等[11,20]提出了一种使用路由索引来查询相关对等体的 skyline 处理方法。路由索引或分布式数据摘要(DDS)是指可通过相邻对等体访问到的数据的摘要。摘要不仅总结其本地数据,而且总结了位于几跳但经由相邻对等体可达的数据。假定网络中的每个对等体都具有这样的摘要。

在这种情况下的 skyline 查询处理如下:首先,查询对等体根据其本地数据计算 skyline 集;然后,根据数据摘要决定其相邻对等体的相关性。主要思想是修剪由本地 skyline 点支配的所有相邻对等体的数据。基于直方图的数据摘要是指由矩形表示的一组区域。给定一个 skyline 查询和一个矩形区域,那么最佳点应位于矩形的左下角。最佳点可能支配该区域包含的所有数据点。如果最佳点支配对等体的本地 skyline 点,则该区域可被修剪。如果某一相邻对等体的数据摘要被修剪,则不会将查询转发给该对等体;否则,查询被转发,并且本地 skyline 点也被转发到相邻对等体以便修剪其邻居。为了最小化启动器的负载,本地 skyline 点在查询传播的同一路径上路由;查询对等体将从相邻对等体收到的本地结果集与其本地 skyline 集进行合并,检查相互的优势,并将获得的结果发送给查询对等体。

(3)SKYPEER 和 SKYPEER+

Vlachou 等[21]提出一种分布式框架 SKYPEER 用于计算超级对等体上的子空间 skyline 处理。为此,将支配的概念定义为扩展的 skyline 集,其包含在任意子空间中有效响应 skyline 查询的所有数据点。在预处理阶段,每个超级对等体计算并存储其关联对等体的扩展 skyline 集。当超级对等体接收到子空间 skyline 查询时,SKYPEER 将查询传播到所有超级对等体并收集本地 skyline 集。此外,SKYPEER 利用有效的阈值方案修剪对等体中被支配的数据。为了支持基于阈值的查询处理,数据需转换为一维值。然后,在查询处理期间,根据已经计算的子空间 skyline 点来定义阈值,阈值在网络传播之前附加到查询中。

文献[22]扩展 SKYPEER 得到了 SKYPEER+,其重点是在超级对等网络上得到 skyline 查询的有效路由,目的是减少连接到的超级对等体的数量。与洪泛网络相反,建立路由机制是为了仅连接那些可能对全局 skyline 集有贡献的超级对等体。更准确地说,在预处理阶段,每个超级对等体对其本地存储的扩展 skyline 集应用聚类算法。然后,基于一维映射来存储扩展的 skyline 集,如图 4 所示。

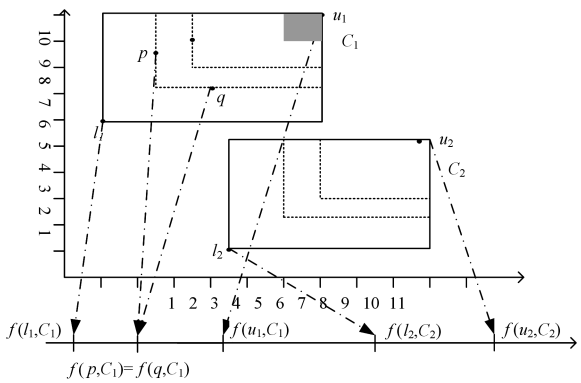

图 4　SKYBEER+:扩展 skyline 点的索引及其一维映射

簇由 MBR 表示,并且每个点被映射为一维值(如点 p 映射为 $f(p,C_1)$,点 q 映射为 $f(q,C_1)$),而属于相同虚线的所有点具有相同的一维值(如 $f(p,C_1)=f(q,C_1)$)。然后,根据 SKYPEER+采用的阈值,使用点 p 修剪阴影区域。簇信息通过超级对等网络进行传播。每个超级对等体收集所有超级对等体的簇信息,并根据它们构建路由索引。通过将一维映

射与聚类信息相结合，提出一种建立路由索引的新技术，有效支持了 SKYPEER 的阈值方案。在查询处理期间，路由索引用于将查询传播到网络路径，超级对等体存储可能有助于 skyline 集的数据点。此外，路由信息用于重新定义阈值。因此，SKYPEER+进一步改善了阈值方案，并大大减少了传输数据的数量。

(4)BITPEER

Fotiadou 等[23]提出了对超级对等体进行子空间 skyline 查询的 BITPEER。与 SKYPEER 类似，每个超级对等体存储其对等体的扩展 skyline 集。不同于 SKYPEER，Fotiadou 等主要研究基于 BITMAP 的分布式 skyline 计算[24]。因此，BITPEER 使用位图表示法，总结所有扩展的 skyline 点。给定一个子空间 skyline 查询，该查询在超级对等网络中洪泛，并且本地结果集在中间超级对等体上进行合并后发送给查询超级对等体。为了实现子空间 skyline 结果的可重用性，在查询处理期间，查询超级对等体收集扩展子空间的 skyline 集[21]而不是子空间的 skyline 集。因此，BITPEER 能够对子空间 skyline 查询使用缓存中的 skyline 集。

比较：在非结构化 P2P 网络中，几乎所有的算法都旨在最小化响应时间。假定连接的对等体和超级对等体的最大数量分别为 N_p 和 N_{sp}。基于超级对等体的方法需要一个预处理阶段，其中对等体的扩展 skyline 集需转移到相关联的超级对等体，成本为 O(1)。SKYPEER+构造路由索引，这需要在超级对等体网络中传播 MBR，导致 $O(N_{sp})$的额外成本。在查询处理期间，所有方法的最大跳数为 $O(N_{sp})$，因为在最坏的情况下，所有超级对等体都需依次地连接。

SFP 和 DDS 都假设是一个 P2P 网络。SFP 使用洪泛进行查询传播，因此最大跳数为 $O(N_p)$。DDS 构建成本为 $O(N_p)$的路由索引。在查询处理期间，DDS 的最大跳数为 $O(N_p)$，因为在最坏的情况下，所有的对等体都需依次联系。

除了最长的路径，传输的数据量也影响响应时间。在最坏的情况下，所有方法都需要传递所有的本地 skyline 点(SKY_i)，最佳的情况是仅转移那些属于全局 skyline 集的本地 skyline 点，即 SKY 点，但这在分布式环境中是不可行的。一般来说，$\sum_{1\cdots N_p} SKY_i \approx N_p * SKY \gg SKY$，因为 skyline 集的基数主要取决于数据分布、维数以及数据基数。所有方法都尝试使用过滤或查询计划来减少传输的数据量。修剪数据量难以估计，因为它取决于数据分布、数据维度、给定的网络拓扑以及数据分发给对等体的方式。SKYPEER，SKYPEER+，BITPEER，DDS 和 SFP 使用过滤(或阈值)来丢弃被支配的点，但路由路径属于给定的网络拓扑。此外，文献[22]表明，SKYPEER+导致比 SKYPEER 更少的传输数据，因为阈值是根据路由信息进行优化的。

小结：在完全连接的网络拓扑结构中，我们可以更好地进行研究，原因是路由路径基于数据之间的支配关系，从而修剪大量的本地数据。由于 $N_{sp}<N_p$，如果存在专用服务器，则基于超级对等体的方法比纯 P2P 方法更有效。文献[22]的实验表明，SKYPEER+胜过 SKYPEER。如果没有专门的服务器存在，并且对等体故障是常见的，那么在超级对等体上构建路由信息的代价将是昂贵的。在这种情况下，DDS 或 SFP 更合适。DDS 具有较小的成本，预期会取得比 SFP 更好的性能。

4 发展趋势

在本节中，我们详细介绍为 skyline 变化提出的各种方法，并讨论了各种方法是否支持其他 skyline 变化。一种方法是否也支持一种 skyline 变化主要取决于其路由机制和过滤方法。表 1 给出了各种分布式方法支持 skyline 变化的各种情况。

表 1 每种方法支持的 skyline 变化

方法	skyline	子空间	有约束	动态
DSL	*			
SSP/Skyframe	√		*	
iSky	√			
SSW	√		*	
SFP	√	*	*	*
DDS	√	*	√	*
SKYPEER/SKYPEER+	*	√		
BITPEER	*	√		

注：* 表示提出；√ 表示支持

在展开研究之前，我们指出方法 SFP 支持本文中的所有 skyline 变化。因为 SFP 将查询转发给所有对等体，所以 SFP 可以直接支持任何 skyline 的查询变化。唯一需要注意的是，需根据给定的查询变化正确地选择过滤点。

4.1 子空间 skyline 查询

上文提及的 skyline 方法大都支持子空间 skyline 查询。最初提出的 SKYPEER 和 BITPEER 都是为了有效地支持子空间 skyline 查询。对于其他的方法，出现的问题是当仅考虑维度时，它们是否可以对相关对等体执行高效的查询路由。

使用 MBR 进行查询路由的方法，如 DDS，可以支持子空间 skyline 查询，因为在 skyline 处理之前需把给定的子空间投影到 MBR。然后，对等体根据投影的 MBR 进行修剪。DSL，Skyframe 和 SSW 假设一个类似的空间分区，并通过结构化覆盖将每个分区映射到特定的对等体。这些方法不能有效地支持子空间 skyline 查询，主要原因如下：首先，这些方法假定对等体的本地 skyline 点可以不进行合并而直接返给用户。当处理子空间 skyline 查询时，由于多个数据分区在投影空间中可能重叠，因此不成立。其次，这些方法所需的另一个属性是数据分区不重叠，这使得这些方法能够有效地路由查询，并避免与同一对等体联系两次。如果数据分区投影到一个子空间中，那么投影的数据分区可能会重叠，从而导致不能有效地路由查询，所提出的方法不能有效地解决这些问题。iSky 根据每个数据点具有其坐标最大值的维度将数据点分配给对等体，虽然这提高了 skyline 查询计算的效率，但是它使得 iSky 不适用于子空间 skyline 查询。数据点到对等体的映射不允许仅处理部分维度，因为某些数据点将被错误地丢弃。

4.2 有约束的 skyline 查询

DSL 和 DDS 的提出不仅针对 skyline 查询，而且也针对有约束的 skyline 查询。因此，这些方法能有效地支持有约束的 skyline 查询。为了有效地支持有约束的 skyline 查询，必须要处理的问题是路由机制和过滤方法必须适用于仅考虑整个数据空间部分区域的情况。

类似于 DSL，Skyframe 和 SSW 也支持有约束的 skyline 查询，因为它们都假设一个类似的空间分区。有约束的 skyline 查询处理从对等体开始，该对等体对给定约束的左下角

进行索引(而不是索引数据源的对等体),并且修剪与查询不重叠的对等体。Skyframe 和 SSW 面对的挑战是高效地检测起始对等体,这是可行的。

另一方面,iSky 对于有约束的 skyline 查询效率不高。虽然查询处理可以从在每个维度上存储最高值但小于约束上限的对等体开始,但 iSky 的效率依赖于索引最大值的对等体是已知的。当每个查询与不同的约束相关联时,这是不可行的,并且在查询处理期间有效地检测这些对等体并不简单。SKYPEER 和 BITPEER 不能简单地扩展到有约束的 skyline 查询,因为用于过滤的阈值方案依赖于基于数据空间计算的一维映射,从而使阈值适应于给定的约束。

4.3　动态 skyline 查询

动态 skyline 查询是 P2P 环境中最具有挑战性的 skyline 查询变化。主要的挑战是查询不是在原始的数据空间中执行的,而是在查询依赖的动态空间中执行,即对于不同的查询来说,它是不同的。最初提出的分布式 skyline 方法不能有效地处理动态 skyline 查询。

基于 MBR 的路由机制的方法,如 DDS,可以在这种情况下支持动态 skyline 查询:动态 skyline 查询的函数允许把每个 MBR 映射为一个转换的 MBR,即包含转换数据空间中的所有数据点。然后,用于查询路由和对等体修剪的动态空间中的 MBR 之间的支配关系导致正确的结果集。DSL,Skyframe 和 SSW 不支持动态 skyline 查询,因为对等体不一定是动态空间中的索引不相交的数据分区,并且相邻的数据分区不被邻居对等体索引。iSky 不能支持动态 skyline 查询,是因为用于查询处理的一维映射的属性在动态空间中不能保持。类似地,SKYPEER 和 BITPEER 不能轻易地扩展到动态 skyline 查询,这是因为用于过滤的阈值方案对于动态空间是不可扩展的。

结束语　尽管在 P2P 环境中已经提出了几种有效的 skyline 计算的方法,但是在相关文献中仍然存在迄今尚未研究的挑战性问题。我们在文中得出,现有的方法都没有研究动态 skyline 查询。处理动态 skyline 查询比传统 skyline 计算更具挑战性,这是因为 skyline 计算是基于一组用户指定的函数。在 P2P 环境中,尝试在数据插入或删除的情况下,保持已计算出的 skyline 集处于最新状态是合理的,而不是从头开始处理整个 skyline 查询。

目前,数据管理研究的趋势是具有不确定性数据的管理。目前,对不确定性数据的概率 skyline 查询主要集中在文献[25]中。在 P2P 环境中,数据的不确定性更容易发生,原因如下:首先,数据本身可能是不确定的;其次,对等体本身可能不被其他对等体信任。每个参与的对等体可能与可信度的值相关联,可信度表明其数据的合法性,从而使其查询结果不确定。因此,对于现实中广泛分布的应用程序来说,不确定数据的分布式查询处理是非常重要和具有挑战性的。文献[26]表明,skyline 集的基数可能很高[26],尤其是在高维或反向相关数据集的情况下。在分布式环境中,skyline 集的高基数不仅导致较高的处理成本,而且由于传输数据量不可忽略会导致较高的带宽消耗,因为在最佳情况下,至少 skyline 点必须被传递到查询对等体。此外,本地 skyline 点的总数远远大于全局 skyline 点的数量。因此,skyline 集的基数估计在 P2P 环境中非常重要。尽管研究者们已经提出了几种方法来评估集中式环境中 skyline 集的基数,但是这些方法在分布式环境中的适用性尚未得到证实。

本文总结了现有 P2P 环境下 skyline 计算方法的概况,给出了 P2P 环境中 skyline 处理的目标和主要原则,以及用于 P2P 系统的现有方法。因此,我们分析了基于 P2P 系统的现有方法,并阐明了每种方法的假设;此外,还提供了性能比较分析;我们也详细阐述了各种方法的 skyline 变化,并根据可支持的 skyline 变化对现有方法进行了分类;最后,我们提出了尚未探索的与 P2P 环境中 skyline 计算相关的开放问题。

参 考 文 献

[1] BÖRZSÖNYI S,KOSSMANN D,STOCKER K. The Skyline Operator[C]// International Conference on Data Engineering, 2001. IEEE,2002:421-430.

[2] CHOMICKI J,GODFREY P,GRYZ J,et al. Skyline with presorting[C]// International Conference on Data Engineering. IEEE,2003:717-719.

[3] GODFREY P,SHIPLEY R,GRYZ J. Maximal vector computation in large data sets[C]// International Conference on Very Large Data Bases. VLDB Endowment,2005:229-240.

[4] KOSSMANN D,RAMSAK F,ROST S. Shooting stars in the sky:an online algorithm for skyline queries[C]// International Conference on Very Large Data Bases. VLDB Endowment, 2002:275-286.

[5] PAPADIAS D,TAO Y,FU G,et al. An optimal and progressive algorithm for skyline queries[C]// ACM SIGMOD International Conference on Management of Data. ACM,2003:467-478.

[6] CUI B,LU H,XU Q,et al. Parallel Distributed Processing of Constrained Skyline Queries by Filtering[C]// International Conference on Data Engineering. IEEE,2008:546-555.

[7] WU P,ZHANG C,FENG Y,et al. Parallelizing Skyline Queries for Scalable Distribution[C]// International Conference on Advances in Database Technology-EDBT. DBLP,2006:112-130.

[8] PEI J,JIN W,ESTER M,et al. Catching the best views of skyline:a semantic approach based on decisive subspaces[C]// International Conference on Very Large Data Bases. 2005:253-264.

[9] TAO Y,XIAO X,PEI J. SUBSKY:Efficient Computation of Skylines in Subspaces[C]// International Conference on Data Engineering. IEEE,2006:65-65.

[10] CRESPO A,GARCIA-MOLINA H. Routing indices for peer-to-peer systems[C]// International Conference on Distributed Computing Systems. IEEE,2002:23-32.

[11] HOSE K,LEMKE C,SATTLER K U. Processing relaxed skylines in PDMS using distributed data summaries[C]// ACM International Conference on Information and Knowledge Management. DBLP,2006:425-434.

[12] RATNASAMY S,FRANCIS P,HANDLEY M,et al. A scalable content-addressable network[C]// Conference on Applications, Technologies,Architectures,and Protocols for Computer Communications. ACM,2001:161-172.

[13] WANG S,OOI B C,TUNG A K H,et al. Efficient Skyline Query Processing on Peer-to-Peer Networks[C]// IEEE, International Conference on Data Engineering. IEEE,2007:1126-1135.

[14] JAGADISH H V,OOI B C,VU Q H. BATON:a balanced tree structure for peer-to-peer networks[C]// International Con-

ference on Very Large Data Bases. VLDB Endowment, 2006: 661-672.

[15] HUANG Z, JENSEN C S, LU H, et al. Skyline Queries Against Mobile Lightweight Devices in MANETs[C]// International Conference on Data Engineering. IEEE, 2006: 66.

[16] WANG S, VU Q H, OOI B C, et al. Skyframe: a framework for skyline query processing in peer-to-peer systems[J]. The International Journal on Very Large Data Bases, 2009, 18(1): 345-362.

[17] CHEN L, CUI B, LU H, et al. iSky: Efficient and Progressive Skyline Computing in a Structured P2P Network[C]// International Conference on Distributed Computing Systems. IEEE, 2008: 160-167.

[18] LI H J, TAN Q Z, LEE W-C, et al. Efficient progressive processing of skyline queries in peer-to-peer systems[C]// International Conference on Scalable Information Systems. DBLP, 2006: 26.

[19] CUI B, CHEN L, XU L, et al. Efficient Skyline Computation in Structured Peer-to-Peer Systems [J]. IEEE Transactions on Knowledge & Data Engineering, 2009, 21(7): 1059-1072.

[20] HOSE K, LEMKE C, SATTLER K U, et al. A Relaxed But Not Necessarily Constrained Way from the Top to the Sky[J]. Lecture Notes in Computer Science, 2007, 4803: 399-407.

[21] VLACHOU A, DOULKERIDIS C, KOTIDIS Y, et al. SKYPEER: Efficient Subspace Skyline Computation over Distributed Data[C]// International Conference on Data Engineering. IEEE, 2007: 416-425.

[22] VLACHOU A, DOULKERIDIS C, KOTIDIS Y, et al. Efficient Routing of Subspace Skyline Queries over Highly Distributed Data[J]. IEEE Transactions on Knowledge & Data Engineering, 2010, 22(12): 1694-1708.

[23] FOTIADOU K, PITOURA E. BITPEER: continuous subspace skyline computation with distributed bitmap indexes[OL]. http://zeus.cs.uoi.gr/~pitoura/distribution/damap08.pdf.

[24] TAN K L, ENG P K, OOI B C. Efficient Progressive Skyline Computation[C]// International Conference on Very Large Data Bases. 2001: 301-310.

[25] PEI J, JIANG B, LIN X, et al. Probabilistic skylines on uncertain data[C]// International Conference on Very Large Data Bases. VLDB Endowment, 2007: 15-26.

[26] ZHANG Z, YANG Y, CAI R, et al. Kernel-based skyline cardinality estimation[C]// ACM Sigmod International Conference on Management of Data. DBLP, 2009: 509-522.

(上接第 57 页)

[3] Google Research Blog. AlphaGo: Mastering the ancient game of Go with Machine Learning[EB/OL]. (2016-01-27). https://google research. blogspot. com/2016/01/alphago-mastering-ancient-game-of-go. html.

[4] KUMAR A N. LEGO Robots and AI[J]. ACM SIGCSE Bulletin, 2005, 37(3): 418-418.

[5] NAKAMOTO S. Bitcoin: A Peer-to-Peer Electronic CashSystem [EB/OL]. https://bitcoin. org/bitcoin. pdf.

[6] HABER S, STORNETTA W S. How to time-stamp a digital document[J]. Journal of Cryptology, 1991, 3(2): 99-111.

[7] Wikipedia. Blockchain[EB/OL]. [2017-10-17]. https://de. wikipedia. org/wiki/Blockchain.

[8] ANDREASM. Mastering Bitcoin: Unlocking Digital Cyrpto-Currencies [M]. Sebastopol, CA: O'Reilly Media, 2014: 160-170.

[9] SINHA A. Client-server computing [J]. Communications of the ACM, 1992, 35(7): 77-98.

[10] FAN J, YI L T, SHU J W. Research on the Technologies of Byzantine System [J]. Journal of Software, 2013, 24(6): 1346-1360.

[11] LOI L U U, NARAYANAN V, ZHENG C D, et al. A Secure Sharding Protocol For Open Blockchains[C]// Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. 2016: 17-30.

[12] HYPERLEDER. About Hyperledger [EB/OL]. [2017-10-17]. https://www. hyperledger. org/about.

[13] YUAN Y, ZHOU T, ZHOU A Y, et al. Blockchain: From Data Intelligence to Knowledge Automation [J]. Acta Automatica Sinica, 2016, 42(4): 481-494.

[14] BANKO M, BRILL E. Scaling to very very large corpora for natural language disambiguation[C]// Meeting of the Association for Computational Linguistics. 2001.

[15] CIRESAN D C, MEIER U, GAMBARDELLA L M, et al. Deep Big Simple Neural Nets Excel on Handwritten Digit Recognition [J]. Neural Computation, 2010, 22(12): 3207-3220.

[16] MCCONAGHY T. How Blockchains Could Transform Artificial Intelligence[EB/OL]. (2016-12-21)[2017-10-17]. http://dataconomy. com/2016/12/blockchains-for-artificial-intelligence.

[17] MACCONAGHY T, GIELEN G. Canonical form functions as a simple means for genetic programming to evolve human-interpretable functions[C]// Conference on Genetic and Evolutinary Computation. 2006: 855-862.

[18] RICK OMAC G. AI on the Blockchain: Dystopian or Utopian Future? [EB/OL]. (2014-10-16)[2017-10-16]. https://www.cryptocoinsnews. com/ai-blockchain-dystopian-utopian-future.

[19] ZHU Z W. Node. js Blockchain Development [M]. Beijing: Mechanical Industry Press, 2017: 14-15.

[20] BITCOINWIKI. ASIC[EB/OL]. (2015-03-29). https://en. bitcoin. it/wiki/ASIC.

[21] BYTOM. BytomWhitePaperV1. 0[EB/OL]. http://bytom. io/BytomWhitePaperV1. 0. pdf.

[22] GridCoin[EB/OL]. [2017-10-17]. http://www. gridcoin. us.

[23] YEGULALP S. ONNX makes machine learning models portable, shareable [EB/OL]. (2017-09-08). https://www. infoworld. com/article/3223401/machine-learning/onnx-makes-machine-learning-models-portable-shareable. html.

[24] ATMATRIX. Atmatrix whitepaper[EB/OL]. https://www. atmatrix. org/ system/whitepaper-cn. pdf.

[25] SONG S Z, ZANG C. Calm Thinking Under the Blockchain Hot [J]. Chinese Banking, 2016(12): 33-35.

[26] POON J, DRYJA T. The Bitcoin Lightning Network: Scalable Off-Chain Instant Payments[EB/OL]. (2016-02-14). https://lightning. network/lightning-network-paper. pdf.

[27] CAI W D, YU L, WANG R, et al. Blockchain Application Development Technical [J]. Journal of Software, 2017, 28(6): 1474-1487.