海量不完整数据的核心数据选择问题的研究

刘永楠 李建中 高 宏
(哈尔滨工业大学计算机科学与技术学院 哈尔滨 150001)

摘 要 在大数据时代,越来越多的带有缺失值的数据需要处理,因而数据不完整成为一种常见的数据质量问题.不完整的数据给大数据的查询、挖掘和分析带来了困难.在某些情况下,数据中的很多缺失值是无法被确定的.只能根据用户的需求,在不完整的数据上选择一部分用户感兴趣的核心数据集合,来提高不完整数据的可用性.完整度较高,规模较小,在用户感兴趣的属性上给出更多完整信息的核心数据集合,能够支持高效的查询处理,提高查询结果的准确性和完整性.该文形式化了核心数据选择问题,证明了这至少是一个NP-难问题.由于需要同时优化核心数据集合的完整度、集合的规模以及对于感兴趣属性的覆盖性,现有的基于集合覆盖问题的方法无法解决文中提出的问题.该文提出了一个采用贪心策略,具有理论保证的近似核心数据选择算法ACS. ACS首先判断当前的数据集合是否存在一个满足覆盖性要求的子集合.当这样的子集合存在时,ACS尽量选择完整的元组来组成核心数据集合,当使用完整元组无法满足覆盖性的要求时,ACS选择较少的不完整元组. ACS通过限制选择的次数来获得一个集合大小的上界是运行次数常数倍的子集合,并且保证了对于感兴趣的属性的覆盖比例.通过理论分析可知,ACS能够在近似线性的时间内,找到一个大小至多在给定的大小对数因子内的近似核心数据集合,其中被覆盖的感兴趣的属性的比例至少为$(1-1/e)$,包含的不完整元组的个数至多为给定的核心数据集合的大小,其中e是自然对数的底数.通过在DBLP和NBA球员信息这两个真实数据集合上的实验,表明了所提出的算法ACS的有效性和高效性;通过在规模更大的合成数据上的实验,表明了ACS的良好的扩展性.

关键词 数据质量;数据完整性;不完整数据;核心数据选择;近似算法
中图法分类号 TP311 **DOI号** 10.11897/SP.J.1016.2018.00915

Research on Core-Sets Selection on Massive Incomplete Data

LIU Yong-Nan LI Jian-Zhong GAO Hong
(*School of Computer Science and Technology, Harbin Institute of Technology, Harbin* 150001)

Abstract In the big data era, more and more data with missing values are to be handled, which makes data incompleteness be a common problem of data quality. Incomplete data brings challenges to querying, mining and analyzing on massive data. In some scenarios, many missing values cannot be determined. According to a user's requirement, only a subset of entire data set with missing values, called core-sets, has to be selected to improve the data usability of the entire incomplete data set. With higher data completeness, smaller size, and more complete information on attributes of a user's interest, core-sets can support efficient query processing and provide more accurate and complete answers to queries. Core-sets selection problem is formalized in this paper, and such problem is proven to be at least NP-Hard. Because there are three objects to be optimized at the same time: the core-set's completeness, size and coverage on attributes of a user's interest, existing methods based on set cover problems cannot solve the core-sets selection problem proposed in this paper. An approximate core-set selection algorithm with theoretical

收稿日期:2016-09-01;在线出版日期:2017-04-23.本课题得到国家自然科学基金(61632010,61502116,61370217,U1509216)、科技部重点研发计划(2016YFB1000703)资助.**刘永楠**,男,1986年生,博士研究生,主要研究方向为数据质量、数据完整性. E-mail: cellsi2011@gmail.com. **李建中**,男,1950年生,教授,博士生导师,中国计算机学会(CCF)会员,主要研究领域为数据库、无线传感器网络. **高 宏**,女,1966年生,教授,博士生导师,中国计算机学会(CCF)会员,主要研究领域为数据库、无线传感器网络.

guarantee based on greedy, called ACS, is proposed in this paper. The proposed algorithm ACS firstly determines whether or not there is a subset of the entire data set with required coverage on the attributes of a user's interest. If such subsets satisfying requirement on coverage exist, ACS selects tuples containing complete values as many as possible to obtain an approximate core-set. But if required coverage cannot be satisfied by only tuples containing complete values, ACS will select a few tuples containing missing values to cover the remaining attributes, but the number of such tuples containing missing values is small. The number of selections in ACS is limited so that ACS can obtain a subset of tuples of which the size is bounded by a constant factor of the number of the selections, and the ratio of attributes of a user's interest covered is guaranteed as well. By theoretical analysis, within nearly linear time, ACS can select an approximate core-set of size within the logarithm factor to the given size, where the ratio of attributes of interest covered is at least $(1-1/e)$, with incomplete tuples of at most the size of the required core-set, where e is the base of the natural logarithm. Experimental results conducted on two real-world datasets of DBLP and NBA players show effectiveness and efficiency of the proposed algorithm ACS, and experimental results on synthetic datasets with larger scale show scalability of ACS.

Keywords data quality; data completeness; incomplete data; core-sets selection; approximate algorithms

1 引 言

高速增长的大数据给了人们更多认识世界,发现知识的机会.随之而来的数据质量问题也日益凸显,给大数据的使用和分析带来了困难[1].有报告显示,在世界顶级的公司中也有超过25%的重要数据是不准确的[2],每年给美国商业造成了6千亿的损失[3].数据质量问题已经是一个数据科学家面临的基本问题①.

不完整数据是常见的低质量的数据,它给很多问题的算法设计带来了挑战[4-7].由于信息更新不及时[8],抽取算法的错误[4]等原因,很多应用面对的数据都包含不完整信息,比如影片的打分数据[6].某些应用中,用户往往只需要有限个对象,比如向用户推荐某些商品[9],或者从众多工人中选择一些合适某项工作的团队[10,11].对于这类查询,由于某些空值无法填充,因此只能选择完整度较高,用户感兴趣的元组作为查询结果,返回给用户.下面通过一个例子来说明.

例1. 图1中给出了电影信息关系的部分属性和部分元组,其中包含了电影的名字,上映的时间,导演的姓名,观众对电影的评分,从第5列开始给出了电影包含的题材.关系中的空白的部分可以视为缺失值,其中某些空值可以填充比如《谍影重重5》的导演的姓名,而有些空值无法填充,比如《谍影重重5》不是儿童题材的影片.如果用户要求选择3部电影,其中要包含爱情、剧情、惊悚、儿童、商业、动作题材.那么,A_1={捉妖记,使徒行者,谍影重重5},A_2={捉妖记,使徒行者,湄公河行动},A_3={捉妖记,谍影重重5,湄公河行动},都可以作为查询结果返回给用户.可以发现集合AC_1={捉妖记,谍影重重5},AC_2={捉妖记,湄公河行动}包含的元组覆盖了用户指定的感兴趣的属性,并且其大小没有超过用户给定的大小限制(3部).但是,为了能够提供较为完整的信息,除了指定的属性集合外,应该尽可能少的包含含有空值的元组.因此,AC_2包含了较AC_1更为完整的元组,AC_2可以看作一个能够回答用户查询的核心数据集合.可以发现,如果能高效地选择核心数据集合AC_2,就能快速地响应用户的查询.

电影名字	上映时间	导演姓名	评分	爱情	剧情	惊悚	儿童	商业	动作	…
捉妖记	2015	许诚毅	6.9	✓	✓		✓		✓	…
使徒行者	2016	文伟鸿	6.8		✓			✓	✓	…
谍影重重5	2016		7.3		✓	✓		✓	✓	…
湄公河行动	2016	林超贤	8.2		✓	✓		✓	✓	…
…	…	…	…	…	…	…	…	…	…	…

图1 电影信息关系

① For Big-Data Scientists, 'Janitor Work' is Key Hurdle to Insights — The New York Times. http://www.nytimes.com/2014/08/18/technology/for-big-data-scientists-hurdle-to-insights-is-janitor-work.html. (Accessed on 06/02/2016)

核心数据集合有很多用途：(1) 通过核心数据集合可以判断，现有的数据集合是否包含了感兴趣的属性的信息，可以以此来判断现有的数据集合的完整程度，并作为衡量数据质量的一个指标. 因为如果无法选择一个类似于例子 1 中的 AC_2 这样的核心数据集合，那么用户指定的属性是无法被完全覆盖的，或者很难有一个完整度较高的查询结果；(2) 可以为后续的操作提供完整度高，规模小的初始集合. 由于核心数据集合规模较小，使得一些操作可以高效地进行；(3) 同时核心数据集合覆盖了用户感兴趣的属性的集合，这个集合往往包含更少的空值，可以提供利于计算的有效信息. 比如，可以根据历史查询信息，找到用户最近喜欢查询哪些类型的影片，然后在核心数据集合上运行推荐算法，来进一步选出满足用户喜好的影片，或者直接回答某些计算密集型的查询[6]，免去了对于大量不在结果集合中的不完整数据的处理，从而提高了效率. 类似地，可以通过指定必备的技能集合，来缩小候选人集合的规模，来高效地求解队伍组建问题[10,11]. 如何从海量的不完整数据中抽取这样的核心数据集合，是本文所研究的问题.

将数据根据用户要求划分，来回答用户的查询，已经有了一些研究结果，但是这些方法都存在不同的缺点，下面分别进行说明：

通过核心数据来快速地回答某些查询[12,13]，广泛地应用于计算几何和机器学习领域. 然而这些方法都针对数值型数据，无法应用于推荐系统等包含分类属性的数据中.

建立索引[14]和近似查询[15,16]可以提高计算的效率. 由于没有将数据按照完整程度进行区分，完整度低的元组一方面无法提供有效的信息，进而导致近似结果产生较大的误差；另一方面浪费了计算的资源. 文献[17]分析了数据中错误类型，文献[18]给出了近似结果的误差分析. 但是这些方法都没有给出选择高质量数据的方法.

通过数据摘要[19,20]和数据压缩方法能提供简洁的数据表示方式，减少了需要处理数据的规模，但是，如果直接用于计算通常需要额外的计算代价，来获得数据的特性[21]. 而且进行变换后，数据的特性有很多损失，多数只能回答聚集操作[16,22]，无法支持某些基于属性值的特点进行的分析，例如 group-by 操作[23].

如例 1 所示，核心数据的选择问题扩展了集合覆盖问题[24]以及最大覆盖问题[25,26]，因此这两个问题的现有方法和结论无法直接应用于核心数据选择问题. 文献[27]考虑了同本文相似的问题，但是，它研究的问题假设总能找到一条元组覆盖所有的属性，在实际的问题中，这个假设不一定能被满足，尤其是在不完整数据上. 此外，它提出的算法的性能依赖于参数的选择，而且无法通过简单地扩展，支持不完整数据上的核心数据选择，更详细的讨论参见相关工作部分，以及实验对比部分.

为了高效地选择包含用户感兴趣属性的较为完整的数据，解决已有方法的问题，本文研究了核心数据的性质，分析了核心数据选择问题的时间复杂性，证明了其弱化版本是一个 NP-完全问题；提出了具有质量保证的近似算法，通过理论分析可知，这个算法能够在近似线性的时间内，抽取覆盖用户感兴趣属性总数一定比例的核心数据集合. 本文的主要贡献如下：

(1) 形式化了不完整数据上的核心数据选择问题，分析了核心数据选择问题的计算复杂性，并证明了其弱化版本是一个 NP-完全问题；

(2) 提出了一个有误差保证的高效的核心数据抽取算法. 当用户给定包含 N 条元组的数据集合，感兴趣的属性的大小是 n，要求至少覆盖这些属性的比例是 θ 时，近似算法能在 $O(Nn\ln n)$ 的时间内，输出一个同给定的大小比值在 $O(\ln n)$ 内的近似核心数据集合，这个集合至少覆盖了 $(1-1/\mathrm{e})\theta n$ 的感兴趣的属性，其中 e 是自然对数的底数；

(3) 通过在 NBA 数据集合和 DBLP 数据集合上的大量实验结果，表明了所提出的算法能够在较短的时间内，在不完整数据上抽取满足覆盖要求的近似核心数据. 通过在合成数据上的实验结果，说明了本文提出的近似方法具有良好的扩展性，能够在海量不完整数据上高效地抽取近似核心数据集合.

本文在第 2 节给出核心数据选择问题的形式化定义，并给出这个问题的时间复杂性的分析结果以及证明；在第 3 节给出近似核心数据的选择算法 ACS 对算法的理论分析和改进的策略；在第 4 节给出近似算法在真实数据集合以及合成数据集合上的实验结果以及分析；在第 5 节综述相关的研究工作；在第 6 节给出结论.

2　问题的定义和复杂性

2.1　核心数据选择问题的形式化定义

给定属性集合 $Att=\{A_j\}$，$1\leq j\leq m$，感兴趣的属性集合 $AI\subseteq Att$，数据集合 T 是定义在 Att 上的关系数据集合，其中若一条元组 $t\in T$ 中至少一个不属于 AI 的属性值为空值，则称 t 是一条“不完

整元组”，否则称为“完整元组”. 如果 t 在属性 A_i 上的属性值不是空值，并且 $A_i \in AI$，则称 t 覆盖了属性 A_i，元组 t 覆盖的属性是其覆盖的所有的属性的并集，表示为 $Ben(t)$. 元组集合 T' 覆盖的属性集合，是其中所有元组覆盖的属性的并集，即 $Ben(T')=\bigcup_{t\in T'} Ben(t)$.

根据上面的定义，下面给出核心数据集合选择问题的形式化定义.

定义 1. 核心数据选择问题.

输入：定义在属性集合 Att 上的关系数据集合 D，感兴趣的属性集合 $AI \subseteq Att$，正整数 k，实数 $\theta \in (0,1]$.

输出：至多 k 条元组的集合 T_{CD}，要求 T_{CD} 中包含不完整元组的数目最少，并且 $|Ben(T_{CD})| \geq \theta |AI|$.

2.2 核心数据选择问题的时间复杂性

由于需要同时优化集合大小，覆盖性和不完整元组的个数，因此核心数据集合的选择问题是一个很难的问题. 下面给出其弱化的版本，来说明这个问题至少的难度. 弱化的版本不限制核心数据中不完整元组的数目，并且只要求 $|Ben(T_{CD})|=\theta|AI|$. 其时间复杂性由下面的定理给出.

定理 1. 弱化的核心数据选择问题是 NP-完全的.

证明. 容易知道，可以猜测一个原始数据集合的子集作为核心数据集合，并在多项式时间内验证其中元组的数目，对于感兴趣属性集合的覆盖程度是否符合条件，因此核心数据选择问题在 NP 中.

下面给出一个从集合覆盖问题的多项式时间的规约方法来证明核心数据选择问题的复杂性：

给定一个集合覆盖问题的实例 $P=\{U,S\}$，其中 U 是一个元素的集合 $U=\{a_1,a_2,\cdots,a_n\}$，以及由 U 中元素的某些子集合组成的集合 $S=\{S_i \mid S_i \in 2^U\}$，即 $S \subseteq 2^U$，其中 2^U 表示集合 U 的幂集合. 集合覆盖问题是要寻找一个最小的由 S 中元素组成的子集合，使得这个子集合中的元素的并集能够覆盖 U 中所有的元素.

设核心数据选择问题的实例是 $C=\{D,AI,k,\theta\}$，其中 D 是给定的数据集合，AI 是感兴趣的属性的集合，k 是核数据中包含元组数目的上限，θ 是至少覆盖的感兴趣的属性的比例. 下面给出一个从集合覆盖问题的实例 P 到核心数据选择问题的实例 C 的规约方法 f：

D 中的属性包含两个部分，一个部分是 U 中全部的元素，另一个部分是属性 M，即 D 中的属性集合是 $Att(D)=U \cup \{M\}$. 指定感兴趣的属性集合 $AI=Att(D)$，指定需要覆盖的比例 $\theta=|U|/(|U|+1)=n/(n+1)$，核数据集合大小的上限 $k=|S|$. 下面给出 D 中元组的具体形式：对于 S 中的每个集合 S_i，向 D 中插入一条元组 t_i，如果 U 中的元素 a_j 在 S_i 中，那么对应的 t_i 的属性 a_j 的属性值为 1，表示这个属性值是完整的. t_i 的属性 M 的值是空值，属性值为 0，将其余的非 1 的属性值都视为空值，将其属性值设为 0. 可以知道，上面的规约 f 在多项式时间内可以完成. 可以发现，由于 AI 指定了 D 中的所有属性，因此 D 中的元组都可以看作完整的.

下面证明核心数据选择问题的实例 C 的一个解，与集合覆盖问题的实例 P 的一个解的对应关系，即利用上面的规约 f，当给定一个 C 的解的实例 $S(C)$ 后，可以得到一个 P 的解的实例 $S(P)$，反之亦然.

(1) $(S(P) \Rightarrow S(C))$. 当给定一个集合覆盖问题的解的实例 $S(P)$ 时，由于 $S(P)$ 已经指定了一个 S 的子集合，由规约 f 中建立数据集合的方法可以知道，S 中的每一个集合 S_i 都对应了一个 D 中的元组 t_i，在解集中的每一个集合 S_j 也对应了一个 D 中的元组 t_j，因此，$S(P)$ 中的每个子集合对应的元组集合，就指定了一个 D 中元组的集合 $S(C)$. 由于 $S(P)$ 已经覆盖了所有的 U 中的元素，相应的，$S(C)$ 也覆盖了 D 中除了 M 以外的所有的属性，即 $S(C)$ 覆盖的属性的比例是 $|Att(D)\backslash\{M\}|/|Att(D)|=n/(n+1)$，显然 $S(C)$ 的大小与 $S(P)$ 的大小相同，不超过 $|S|=k$，并且 D 中的元组都是完整的，所以 $S(C)$ 是核心数据选择问题的一个解.

(2) $(S(C) \Rightarrow S(P))$. 当给定一个核心数据集合选择问题的解的实例 $S(C)$ 时，根据对核心数据的要求，$S(C)$ 已经覆盖了至少 $n/(n+1)$ 的 AI 中的属性，由于 AI 中 M 属性在所有的元组上都是空值，无法被覆盖，所以，$S(C)$ 一定覆盖了除了 M 以外的所有属性，即全部的 U 中元素对应的属性. 由规约 f 的过程可以知道，每条 $S(C)$ 中的元组 t_j，对应一个 S 中的集合 S_j，因此，$S(C)$ 中的元组也指定了 S 的一个子集合 $S(P)$，由于 $S(C)$ 覆盖了全部的 U 中元素对应的属性，因此，由规约 f 可以知道，$S(P)$ 覆盖了 U 中的全部元素. 由于 $S(C)$ 的大小没有超过 $k=|S|$，因此，$S(P)$ 是 S 的一个子集，即 $S(P)$ 是集合覆盖问题的一个解. 证毕.

3 核心数据选择问题的近似算法

通过前面的定理可以知道，核心数据选择问题

是一个很难的问题.因此,需要设计一个高效率的近似算法来选择一个近似的核心数据集合.为了得到一个高效率的近似算法,需要放松核心数据选择问题的三个优化目标.采用同最大覆盖问题[25]相似的思想,我们设计了近似核心数据选择算法 ACS,ACS 采用贪心的策略,当用户指定感兴趣属性个数后,放松对不完整元组数目的限制,选择比最优解多一些的近似核心数据集合,能够保证至少覆盖 $(1-1/e)$ 的比例.

在这一部分中,在 3.1 节给出近似核心数据选择算法 ACS,在 3.2 节分析这个算法,并给出算法的时间复杂性分析结果和对近似结果的质量分析,在 3.3 节给出对于算法的改进的策略.

3.1　近似核心数据选择算法 ACS

在下面的描述中,A 表示完整的元组集合,B 表示不完整的元组集合.

算法的直观思想是:尽可能地从 A 集合选择元组,来减少不完整元组的个数.如果通过选择 A 集合中的元组,已经能够满足覆盖要求,就不必使用 B 集合中的元组,否则,用 A 集合的元组尽可能的覆盖后,再选择 B 集合中的元组.

算法 1 给出核心数据集合近似选择算法,扫描一遍 A 集合,得到 A 集合能够覆盖的元素集合 R_0,即能够覆盖感兴趣的属性的数目.再扫描一遍 B 集合,得到 B 集合能够覆盖的元素集合 R_1,如果 $|R_0 \cup R_1| < (1-1/e)\theta n$,说明即使全部选择这些元组,也无法完成指定比例的覆盖,因此返回“无法完成”.否则,在 A 集合执行贪心的集合覆盖[25],即每次选择覆盖剩余 AI 属性个数最多的元组,如果两条元组覆盖的个数相同,任意选择一条,然后从 AI 中去掉已经被覆盖的属性,共执行 $(\ln((1-1/e)\theta n)+1)$ 次,每次选择 k 条,期间如果已经覆盖完成就提前终止.如果 A 集合完成后,覆盖的元素个数小于 $(1-1/e)\theta n$,那么在 B 集合执行贪心的集合覆盖一次,共选择 k 条元组,如果能提前达到覆盖要求,就提前终止.

在近似核心数据选择算法 ACS 中,假设每次的选择都能覆盖尽可能多的元素,在大数据背景下,由于数据是丰富的,具有覆盖相同感兴趣属性的元组是充足的,这一假设能够被满足,比如很多用户都看过类似的电影[6],很多员工有相似的技能[11].如果最优解能用 k 条元组覆盖要求的感兴趣属性的个数,那么,接下来每次需要覆盖的属性个数,一定也可以被 k 条元组覆盖.

算法 1.　近似核心数据选择算法 ACS.

输入:定义在属性集合 Att 上的关系数据集合 T,其中 $|T|=N$,感兴趣的属性集合 $AI \subseteq Att$,$|AI|=n$,正整数 k,实数 $0<\theta\leq 1$

输出:近似核心数据集合 T_{CD}

```
1.  T_CD ← ∅;
2.  扫描一遍 A 集合,得到 A 集合能够覆盖的元素集合 R_0;
3.  扫描一遍 B 集合,得到 B 集合能够覆盖的元素集合 R_1;
4.  IF |R_0 ∪ R_1| < (1-1/e)θn THEN
5.      RETURN "无法完成";
6.  ELSE
7.  //先在 A 集合上进行选择
8.      FOR i ← 1 TO (ln((1-1/e)θn)+1) DO
9.        FOR j ← 1 TO k DO
10.         e_j ← GreedySetCover(A,AI);
11.         AI ← AI \ {Ben(e_j)}
12.         T_CD ← T_CD ∪ e_j;
13.         IF Coverage(T_CD) ≥ (1-1/e)θn THEN
14.           RETURN T_CD;
15.         ENDIF
16.       ENDFOR
17.     ENDFOR
18. //再在 B 集合上进行选择
19.     FOR j ← 1 TO k DO
20.       e_j ← GreedySetCover(B,AI);
21.       AI ← AI \ {Ben(e_j)}
22.       T_CD ← T_CD ∪ e_j;
23.       IF Coverage(T_CD) ≥ (1-1/e)θn THEN
24.         RETURN T_CD;
25.       ENDIF
26.     ENDFOR
27. ENDIF
28. RETURN T_CD;
```

3.2　近似核心数据选择算法 ACS 的分析

由于衡量近似核心数据选择算法的优劣涉及 4 个目标:时间复杂性、核心数据集合的大小、核心数据集合包含不完整元组的个数、核心数据集合覆盖 AI 中属性的个数,因此,从这 4 个方面分别给出分析,分析的结果由下面的定理给出.

定理 2.　近似算法能够在 $O(Nn\ln n)$ 时间内选择 $O(\ln n)k$ 条元组,至少覆盖 $(1-1/e)\theta n$ 个感兴趣的属性,得到的近似核心数据集合的不完整元组的个数与最优个数的比值不超过 k.其中,n 是感兴趣的属性的数目,N 是元组的总数.

证明.　下面分 4 个方面给出分析,来证明上面

的定理.

以下证明中假设最优解能够根据用户的需求,最多用 k 条元组完成对 AI 中指定比例属性的覆盖.

(1) ACS 的时间复杂性. 用 N_A 表示 A 集合中元组的数目,N_B 表示 B 集合中元组的数目. 首先,每次在 A 集合上进行贪心的集合覆盖,选择覆盖剩余属性个数最多的元组,这一步在 $O(N_An)$ 时间内可以完成,每次贪心选择要选择 k 条元组,总共进行 $(\ln((1-1/e)n)+1)$ 次,因此,在完整元组集合上的操作在 $O(kN_An(\ln((1-1/e)n)+1))$ 内,即 $O(kN_An\ln n))$ 可以完成. 然后,如果还需要在 B 集合上执行一次大小为 k 的贪心选择,也可以在 $O(kN_Bn)$ 的时间内完成. 由于 N_A 和 N_B 都不会超过 N. 因此,近似算法可以在 $O(Nn\ln n)$ 内完成.

(2) 核心数据集合的大小. 如果最优解只用 A 集合可以完成核心数据的选择,那么,近似算法也一定只用 A 集合完成核心数据的选择,因为在最坏的情况下,ACS 选择的元组已经覆盖了 A 中元组能够覆盖的所有属性. 因此近似核心数据集合的大小与给定的大小的比值就是运行的次数,即 $(\ln((1-1/e)\theta n)+1)=O(\ln n)$.

如果最优解需要用 B 集合完成,那么,优化解至少引入一条不完整的元组. 如果这样,一定是因为某些没有被覆盖的属性被某条 B 中的元组覆盖,那么,因为近似算法采用贪心的方法,也一定选中了这条. 否则,最优解至少需要在 B 集合中选择 2 条元组,最多用 k 条元组,而近似算法最多用 $k(\ln((1-1/e)\theta n)+1+1)$ 条. 因此,近似比是 $O(\ln n)$.

因此,由于用户指定了最多使用 k 条元组来完成核心数据的选择,如果最优算法在 k 条元组内就能完成核心数据的选择,近似算法需要用 $k(\ln((1-1/e)\theta n)+1+1)$ 条,即 $kO(\ln n)$ 条.

(3) 核心数据集合中不完整元组的个数. 如果最优解只用 A 集合中的元组完成,那么近似算法也同样只选择了 A 集合中的元组,因此,近似算法选择的元组也都是完整的.

如果最优解用 k 条元组完成了覆盖,其中包含了 B 集合中的元组. 那么,按照近似算法,当 A 中的元组无法满足覆盖要求时,在 B 中最多选择 k 条元组,那么,近似算法的不完整元组的个数就是 k,而最优解是选择了 B 中元组的至少一条,因此,近似比不超过 k.

(4) 覆盖的 AI 中属性的个数. 以下分析按照最优解能够完全覆盖 R_0,此时覆盖的属性最少. 当不能完全覆盖时,由于每次会覆盖新的属性,同下面的分析类似,会得到更多的覆盖的个数. 按照近似算法,在 A 集合上执行贪心的集合覆盖,如果已知 A 集合能够覆盖 $|R_0|$ 的属性,那么,进行一次覆盖之后,可以覆盖的属性个数记为 M_1,应该满足 $M_1\geq(1-1/e)|R_0|$[25],因此,经过一次覆盖之后,剩余的属性不超过 $|R_0|/e$,同理,再次覆盖之后,剩余的属性不超过 $|R_0|/e^2$,那么经过 l 次之后,剩余的属性不超过 $|R_0|/e^l$. 假设 l 次之后已经剩余至多一个属性了,那么 $|R_0|/e^l\leq 1$,因此有 $l\geq\ln(|R_0|)$,即最多经过 $(l+1)$ 次就能完成全部属性的覆盖.

如果设置要覆盖的比例为 $(1-1/e)$,即需要覆盖 $(1-1/e)\theta n$ 的属性,如果只用 A 集合就能完成,应该有 $|R_0|\geq(1-1/e)\theta n$,所以,代入上面的式子,最多经过 $(\ln((1-1/e)\theta n)+1)$ 次,就能完成预计的覆盖要求.

如果只用 A 集合无法完成,那么就应该有 $|R_0|\leq(1-1/e)\theta n$,所以,代入上面的式子,最多经过 $(\ln((1-1/e)\theta n)+1)$ 次,已经覆盖了所有的 A 中的元素,剩下的就只能从 B 中选择.

在最坏的情况下,A 集合只能覆盖 i 个属性,其中 $0\leq i<\theta n$,那么,B 集合可以覆盖剩余的 $(\theta n-i)$ 个元素,那么同理,经过 p 次之后,应该满足至少覆盖了 $(1-1/e^p)$ 比例的属性,即应该有 $(\theta n-i)(1-1/e^p)+i\geq(1-1/e)\theta n$,假设 $\theta n\geq i$,即采用 B 集合能够完成覆盖. 那么满足 $p\geq\ln((\theta n-i)/\theta n)+1$,那么,由于 $\ln((\theta n-i)/\theta n)<\ln(\theta n/\theta n)=\ln 1=0$,所以 p 至少为 1 即可,即只需在 B 集合中进行一次大小为 k 的贪心集合覆盖即可,因为最优解最多用 k 条元组就可以覆盖全部属性. 证毕.

如定理 2 给出的,在某些数据集合上,近似算法可能选择超过 k 条元组的近似核心数据集合,最多会选择 $O(\ln n)k$ 条元组. 这是因为为了降低核心数据集合中不完整元组的数目,需要尽可能地从 A 集合中进行选择. 为了覆盖至少 $(1-1/e)\theta n$ 的感兴趣属性,可能需要多次从 A 集合进行选择,来覆盖新的感兴趣属性,由于每次选择 k 条元组,这样就会导致选择的近似核心数据集合的规模超过 k,但是正如分析中所指出的,超出的比例不会太多,是以 $O(\ln n)$ 为上界的. 达到上界时,A 集合所能覆盖的感兴趣属性已经都被覆盖了,利用了最多 $O(\ln n)k$ 的元组. 近似算法 ACS 通过多次在 A 集合中进行选择,虽然增加了输出的核心数据的规模,但是能够使得感兴趣属性更多地被完整元组所覆盖. 在实验 4.3 节的第(5)部分,通过实验说明了 ACS 输出的近似核心数据规模虽然较大,但是其中包含的不完

整元组的个数较少.

3.3　近似算法的改进策略

在这一部分中，讨论对算法 ACS 的改进策略，主要考虑 3 个问题：

(1) 选择的近似核心数据的规模小于给定的大小时，是否需要继续添加元组；

(2) 是否可以在整个数据集合上贪心地进行选择，而不是像本文的算法 ACS，先在完整的元组中进行选择，然后在不完整的元组中进行选择；

(3) 如果用户需要在同一个属性上，需要多个不同的值，如何修改本文提出的算法，修改后是否仍然具有质量保证.

对于第一个问题，如果数据的完整度比较高，或者用户给定的感兴趣的属性集合比较容易被满足时. 往往(近似)核心数据的大小会小于用户给定的大小，此时需要不同的策略来将结果返回给用户. 比如，可以直接给用户返回现有的集合，并告知 AI 集合中的大部分属性已经被覆盖. 或者，继续按照贪心的方式选择覆盖 AI 中覆盖比例高的元组，直至达到用户设定的大小，这样可以在已经覆盖的属性上面提供更多的候选值给用户，对于有序关系的数值类属性，可以便于用户进一步进行选择，但是对于两个属性值无法比较大小的类别属性，用户可能需要参考额外的信息进行选择.

对于第二个问题，本文的选择策略是，尽可能选择完整的元组，除非完整的元组无法满足覆盖要求. 如果不区分完整元组集合以及不完整元组集合，整体进行贪心选择，也能够达到至少$(1-1/e)$的对最优解的覆盖比例[25]，但是，这样往往无法保证选择较少的不完整元组，可能使得选出的结果集合，无法为用户或者后续的操作提供高质量的候选集合，影响后续操作的效果和效率.

对于第三个问题，用户指定在某个属性上需要提供多个不同的值，方便用户进行未来的选择. 可以在代码中添加一个计数器，控制这个属性被覆盖的次数. 将计数器的初始值设为用户指定的个数，每被覆盖一次，计数器的数值减一，只有计数器数值为零，才认为这个属性被覆盖. 但是，这样就要求这个属性一定被覆盖，因为近似算法选择的核心数据集合只保证能够覆盖指定个数的属性，并没有指定某一个属性一定被覆盖. 因为在每一轮进行贪心选择的时候，无法保证这个用户指定的属性，一定被覆盖剩余属性最多的元组所覆盖. 可以采取一种保守的策略，当每次进行贪心选择的时候，如果两条元组覆盖的剩余属性的个数相同，优先选择覆盖了较多指定个数的属性的元组，如果包含的这样的属性的个数相同，那么任意选择一条. 这样修改后，运行一次近似算法 ACS 仍能具有原来的覆盖属性的个数，近似核心数据大小，以及不完整元组的个数较少的保证. 然后，统计这次的选择结果中对用户指定了个数的属性的覆盖情况，根据覆盖的个数，可以选择是否再次运行算法，再次运行前，将已经覆盖的属性和相应的计数器进行更新. 但是，需要运行多少次能够达到用户满意的覆盖程度无法进行估计，与实际的数据的特性有关. 在 4.2 节的第(4)部分，通过多次运行 ACS 的方式，探索了选出的核心数据集合对指定的属性的覆盖次数.

4　性能评价

在这一部分中，通过在真实数据集合和合成数据集合上的实验来分析所提出的近似算法的有效性和性能. 根据核心数据选择问题的定义，通过实验来分析所提出的近似算法的时间效率，输出的近似核心数据集合的大小，对感兴趣属性的覆盖比例以及所包含的不完整元组的个数.

4.1　实验设置

我们用 C++ 语言编写了近似算法 ACS 的代码. 实验中进行对比的 Baseline 算法是文献[27]中的算法 CMC 和 CWSC，采用 C++ 编写了代码. 实验在一台联想台式机上执行，机器的配置是 64 位 Windows 7 SP1 旗舰版操作系统，8 核 3.40 GHz i7-3770 Core CPU，32 GB 内存，500 GB 机械硬盘. 每个设置的参数下，运行 10 次近似算法，下面报告的运行时间是平均时间.

(1) 数据集合

① NBA 数据集合. 为了模拟队伍组建问题，采用 NBA 球员信息集合，并以此为基础生成了合成的数据集合. NBA 球员数据集合来自于 NBA 球员信息网站①，包含了从 1946 年到 2014 年的 NBA 球队和球员信息，表中包含了 24 个属性：球员的编号、球员的名字、赛季的名字、球员的年龄、球队的简称、球队所在的联盟、参加的比赛的数目、赛季的得分、球队参加的赛季、队伍所在的联盟、球队的名称、球队的教练的名字、球队成立的时间、球队解散的时间(或者到现在)、球队的存在的时间、球队参加的比赛

① http://www.basketball-reference.com

次数、球队获胜的次数、球队失败的次数、球队获得冠军的次数、体育场的主场球队的名字、体育场的起止使用年、体育场的名称、体育场的所在地、体育场的容量.

② DBLP 数据集合. 为了模拟队伍组建问题,采用 DBLP 中的会议论文信息集合,并以此为基础生成了合成的数据集合. DBLP 会议论文信息来自于 DBLP 网站①,包含了截至到 2016 年 7 月的全部的会议论文信息,表中包含了 8 个属性:论文的编号、论文的作者、论文的题目、论文的发表时间、论文集的名字、交叉链接的键值、出版商的名字、论文集的编辑的名字.

两个真实数据集合的统计信息如表 1 所示.

表 1 真实数据集合统计信息

数据集合	属性个数	元组个数
NBA 数据集合	24	19197
DBLP 数据集合	8	1819273

③ 合成数据集合. 为了探究所提出的算法,在具有不同的完整度以及元组个数的集合上的性能,我们在真实的 NBA 集合中以 0.2 的概率,随机地将属性值设为空值,然后将每条元组按照一定倍数进行扩增,每条元组以 0.8 的概率加入生成的数据集合. 此外,为了分析更多属性和更多不完整属性值对近似算法的影响,在合成数据中加入了更多的属性,这些属性所包含的属性值以 0.6 的概率为空值,具体的参数会在下面的每个实验设置中详细说明.

(2) 评价指标

为了评估所提出的近似算法的有效性和效率,关注 4 个方面的实验效果:① 近似算法运行的时间,是否在可以容忍的范围内;② 所选择的近似核心数据集合的大小,是否没有过多地超出所要求的大小;③ 对感兴趣的集合的覆盖比例,是否至少达到了近似算法预期的比例;④ 包含的不完整元组的个数,是否尽可能得少.

在下面的实验中,我们将变化不同的参数,分别报告在真实数据和合成数据上的实验结果,观察上面的 4 个指标的满足情况.

4.2 在真实数据上的实验结果

为了研究本文提出的算法的有效性和效率,我们在真实的 NBA 数据集合和 DBLP 数据集合上模拟队伍组建问题,观察上面的 4 个指标的被满足情况. 其中,NBA 数据集合中,不完整属性值占全部属性值的比例是 12.3%,而 DBLP 数据集合中这个比例是 8.2%.

(1) k 的变化的影响

在这组实验中,通过改变用户要求的核心数据的大小 k,观察上面 4 个指标的满足程度. 实验中,在 NBA 数据集合中,随机选择 15 个用户感兴趣的属性作为 AI,在 DBLP 数据集合中,随机选择 6 个用户感兴趣的属性作为 AI,覆盖比例 θ 设置为 0.9,k 从 5 变化到 20. 实验的结果如图 2 和图 3 所示,其中横坐标给出了 k 的大小,纵坐标给出了算法运行的时间.

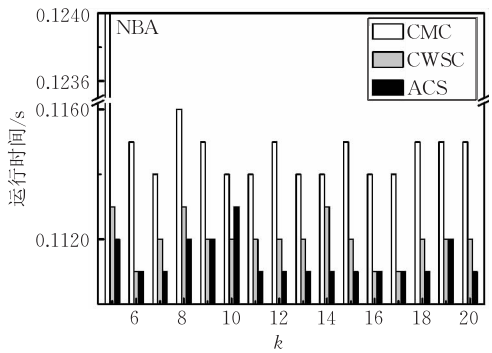

图 2 在 NBA 数据集合上 k 的变化对运行时间的影响

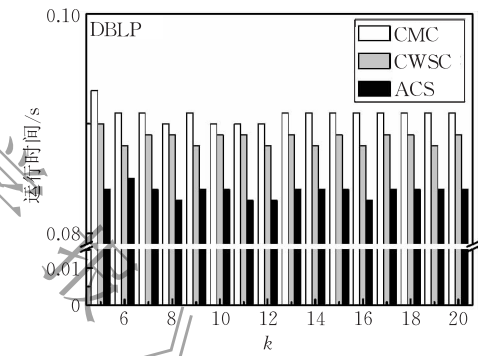

图 3 在 DBLP 数据集合上 k 的变化对运行时间的影响

对于本文提出的算法 ACS,通过在得到的近似核心数据计算上面的 4 个指标,可知这 4 个指标都能够被满足. 所选择的近似核心数据的大小在所要求大小范围内,比如在 NBA 数据集合上,当用户要求 k 等于 5 时,近似算法选择了 4 条元组已经能够覆盖全部的 15 个属性. 可以根据不同的策略,再选择 1 条元组,改进已有的选择结果,完成核心数据的选择. 即使数据中包含很多的空值,选出的 4 条元组都是完整的,共消耗 0.112 s. 而在 DBLP 数据集合上,当用户要求 k 等于 5 时,近似算法也选择了 4 条元组已经能够覆盖全部的 6 个属性. 可以根据不同的策略,再选择 1 条元组,完成核心数据的选择. DBLP 数据集合的完整度较高,选出的 4 条元组都

① http://dblp.dagstuhl.de/xml/release/

是完整的，共消耗 0.084 s.

通过图 2 和图 3 可以发现，虽然两个 Baseline 算法 CMC 和算法 CWSC 都能够满足 4 个指标，但是它们消耗了更多的时间，其中算法 CMC 所消耗的时间最多，因为算法需要多次尝试来猜测一个近似解，由于给定的 k 值不同，多次尝试需要消耗更多的时间.

通过图 2 和图 3 可以发现，近似算法能够高效地选择近似的核心数据集合，即使两个数据集合完整度有一定差距，还是能够找到满足覆盖比例的近似核心数据集合，并且随着 k 的变化，时间消耗的变化在很小的范围内波动. 主要原因是某些元组的完整属性值很多，因此一旦被选中，就能使得选择很快被完成，即使 k 给出的值很大，也可以提前结束选择过程.

(2) θ 的变化的影响

在这组实验中，通过改变用户要求的覆盖比例 θ，观察上面 4 个指标的满足程度. 在实验中，在 NBA 数据集合上随机选择 15 个用户感兴趣的属性作为 AI，在 DBLP 数据集合中，随机选择 6 个用户感兴趣的属性作为 AI，用户指定的核心数据的大小 k 设为 10，θ 从 0.6 变化到 1.0. 实验结果如图 4 和图 5 所示，其中横坐标给出了 θ 的大小，纵坐标给出了算法运行的时间.

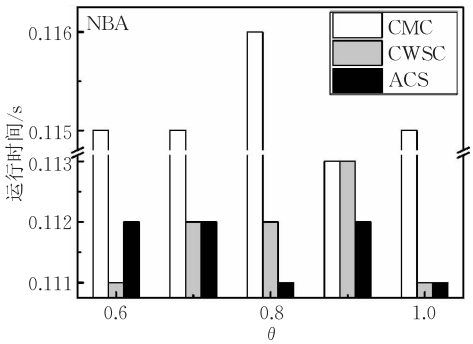

图 4　在 NBA 数据集合上 θ 的变化对运行时间的影响

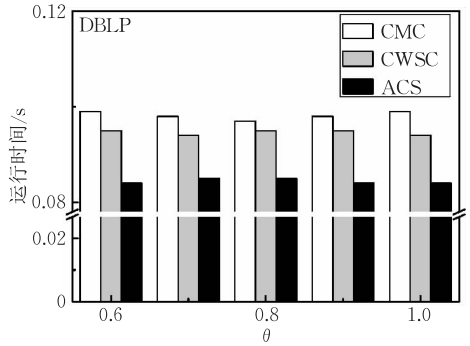

图 5　在 DBLP 数据集合上 θ 的变化对运行时间的影响

对于本文的算法 ACS，通过得到的近似核心数据计算上面的 4 个指标，可知这 4 个指标都能够被满足. 所选择的近似核心数据的大小在所要求大小范围内，比如在 NBA 数据集合上，当用户要求 θ 等于 0.8 时，近似算法选择了 8 条元组已经能够覆盖 15 个 AI 中的属性. 可以根据不同的策略，再选择 2 条元组，改进已有的选择结果，完成核心数据的选择. 即使数据中包含很多的空值，选出的 8 条元组都是完整的，共消耗 0.111 s. 在 DBLP 数据集合上，当用户要求 θ 等于 0.8 时，近似算法选择了 5 条元组已经能够覆盖 6 个 AI 中的属性. 可以根据不同的策略，再选择 5 条元组，改进已有的选择结果，完成核心数据的选择. 在完整度较高的 DBLP 数据集合上，选出的 5 条元组都是完整的，共消耗 0.085 s.

通过图 4 和图 5 可以发现，虽然两个 Baseline 算法 CMC 和算法 CWSC 都能够满足 4 个指标，但是它们消耗了更多的时间，其中算法 CMC 所消耗的时间最多，因为算法需要多次迭代来猜测一个近似解. 虽然 θ 不断变化，但是由于 DBLP 中的元组比较完整，因此容易找到满足条件的核心数据集合，因此整体来看在 DBLP 数据集合上消耗的时间较小.

通过图 4 和图 5 发现，近似算法能够高效地选择近似的核心数据集合，θ 的变化没有引起较大的时间消耗的波动. 主要原因是某些元组的完整属性值很多，因此一旦被选中，能够很快达到用户指定的覆盖要求，由于近似算法发现满足比例就提前了，因此选择往往很快就完成了.

(3) AI 的变化的影响

在这组实验中，通过改变用户感兴趣的属性集合 AI，观察上面 4 个指标的满足程度. 在实验中，在指定 AI 的个数后，随机选择感兴趣的属性. 用户指定的核心数据的大小 k 为 10，用户指定的覆盖比例 θ 设置为 0.9. AI 的大小从 4 个变化到 8 个. 实验结果如图 6 和图 7 所示，其中横坐标给出了 AI 的大小，纵坐标给出了算法运行的时间.

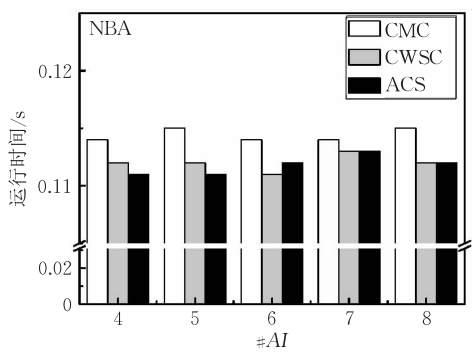

图 6　在 NBA 数据集合上 AI 的变化对运行时间的影响

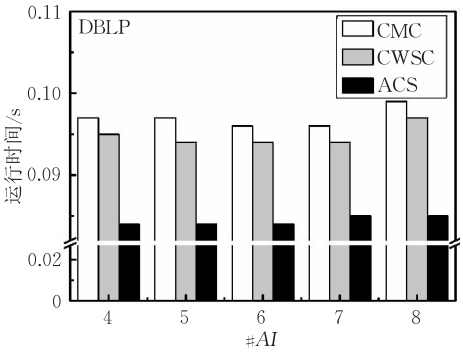

图 7 在 DBLP 数据集合上 AI 的变化对运行时间的影响

对于本文的算法 ACS,通过得到的近似核心数据计算上面的 4 个指标,这 4 个指标都能够被满足.所选择的近似核心数据的大小在所要求大小范围内,比如 NBA 数据集合上,当用户给出 AI 的大小为 7 时,近似算法选择了 6 条元组已经能够覆盖 7 个 AI 中的属性.可以根据不同的策略,再选择 4 条元组,改进已有的选择结果,完成核心数据的选择.即使数据中包含很多的空值,选出的 6 条元组都是完整的,共消耗 0.113 s.在 DBLP 数据集合上,当用户要求 AI 大小为 7 时,近似算法选择了 6 条元组已经能够覆盖 7 个 AI 中的属性.可以根据不同的策略,再选择 4 条元组,改进已有的选择结果,完成核心数据的选择.在完整属性值较多的 DBLP 数据集合上,选出的 6 条元组都是完整的,共消耗 0.085 s.

通过图 6 和图 7 可以发现,虽然两个 Baseline 算法 CMC 和算法 CWSC 都能够满足 4 个指标,但是它们消耗了更多的时间,其中算法 CMC 所消耗的时间最多,因为算法需要多次迭代来猜测一个近似解.虽然 AI 不断变化,但是由于 DBLP 中的元组比较完整,因此容易找到能够覆盖足够数目的 AI 中的属性的核心数据集合,因此整体来看在 DBLP 数据集合上消耗的时间较小.

通过图 6 和图 7 可知,近似算法能够高效地选择近似的核心数据集合,虽然 AI 有不同的变化,引起较小的时间消耗的波动,在可以接受的范围内.主要原因是某些元组完整度较高,可以较多地覆盖 AI 中指定的属性,因此,一旦被选中,可以使得选择很快完成.

(4)指定某个 AI 属性的个数

在这组实验中,通过 6 次运行算法来查看所选择的 AI 属性被覆盖的次数.在实验中,随机选择 8 个用户感兴趣的属性作为 AI,覆盖比例设置为 0.9,k 设为 10.实验结果如图 8 和图 9,其中横坐标是 8 个属性的编号,纵坐标给出了在 6 次运行后,每个属性被覆盖的个数.

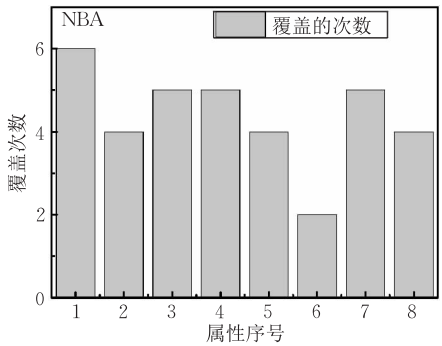

图 8 在 NBA 数据集合上对 AI 中属性的覆盖次数

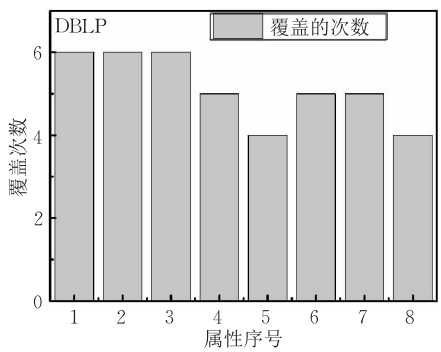

图 9 在 DBLP 数据集合上对 AI 中属性的覆盖次数

通过图 8 和图 9 可以发现,指定 AI 中的属性后,每次运行都会选择一些被覆盖的属性.由于 ACS 在选择满足覆盖比例的属性后就会提前终止,因此,对各个属性的覆盖次数并不相同,同时覆盖的次数也由数据集合本身的性质决定.可以发现在图 8 中对第 6 个属性仅仅覆盖了 2 次,而对第(1)个属性覆盖了 6 次.整体来看,由于 DBLP 数据的完整程度比较高,因此对属性的覆盖次数很多,NBA 数据集合较不完整,特别某些属性空值很多,因此不是对每个属性都覆盖了很多次.

正如 3.3 节的第三点改进策略所分析的,由于 ACS 要保证每次选择时覆盖一定比例的属性,并不能一定覆盖某个属性,同时由于数据集合本身的原因,无法直接保证某个属性被覆盖的次数.但是,可以通过多次运行的方式对某些属性多次覆盖,来达到用户的要求.

(5)真实数据集合实验结果总结

在真实数据上的实验中,本文提出的算法 ACS 能够根据不同的用户需求,高效地选择近似的核心数据.ACS 能够满足 4 个评价指标,并且不同的参数变化没有引起性能上太大的波动.

通过上面的实验图可以发现,ACS 的性能要优

于 2 个 Baseline 算法 CMC 和 CWSC，其中 CMC 算法通过试探的方式来进行近似核心数据的选择，因此在某些情况下，多次的试探会极大地增加选择的时间，导致性能会有所降低.

4.3　在合成数据上的实验结果

为了研究本文提出的算法的效率，以及选择的近似核心数据集合的质量，我们在合成的 NBA 数据集合上模拟队伍组建问题，观察上面的 4 个指标的被满足情况. 由于 DBLP 数据集合的实验结果类似，并且同 NBA 数据集合都是合成的集合，因此我们在这部分只给出在合成的 NBA 数据集合上的实验结果.

为了更好的观察我们提出的算法的效率，下面的前三组实验在同一个合成的 NBA 数据集合上进行，之后的两组实验观察不同的元组总数的影响时，采用不同的合成数据集合. 前三组实验所使用的合成的数据集合包含了 74 个属性，6 910 080 条元组，大小为 2.31 G.

(1) k 的变化的影响

在这组实验中，通过改变用户要求的核心数据的大小 k，观察上面 4 个指标的满足程度. 在实验中，随机选择 8 个用户感兴趣的属性作为 AI，覆盖比例设置为 0.9，k 从 5 变化到 20. 实验结果如图 10 所示，其中横坐标给出了 k 的大小，纵坐标给出了算法运行的时间.

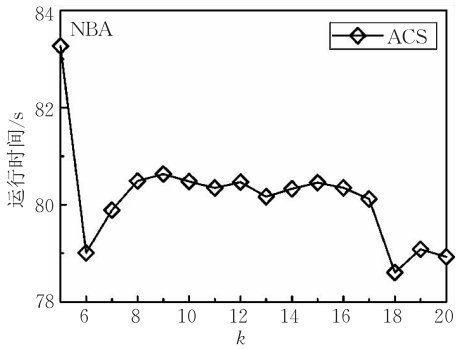

图 10　NBA 合成数据集合上 k 的变化对运行时间的影响

通过得到的近似核心数据计算上面的 4 个指标，这 4 个指标都能够被满足. 所选择的近似核心数据的大小在所要求大小范围内，比如当用户要求 k 等于 5 时，近似算法选择了 2 条元组已经能够覆盖全部的 8 个属性. 可以根据不同的策略，再选择 3 条元组，改进已有的选择结果，完成核心数据的选择. 由于数据集合中包含很多的空值，完整的元组对 AI 的覆盖比例很低，因此选择的都是不完整的元组，共消耗 83.27 s.

可以发现，近似算法能够高效地选择近似的核心数据集合，随着 k 的变化，时间消耗的变化在可以容忍的范围内. 虽然在合成数据集合中加入了很多空值，但是由于 AI 属性的指定具有随机性，并且给定的个数比较小，而且合成的某些元组仍然可以覆盖多数的 AI 属性，因此选择的过程较快.

(2) θ 的变化的影响

在这组实验中，通过改变用户要求的覆盖比例 θ，观察上面 4 个指标的满足程度. 在实验中，随机选择 8 个用户感兴趣的属性作为 AI，用户指定的核心数据的大小 k 设为 10，θ 从 0.6 变化到 1.0. 实验结果如图 11 所示，其中横坐标给出了 θ 的大小，纵坐标给出了算法运行的时间.

通过得到的近似核心数据计算上面的 4 个指标，这 4 个指标都能够被满足. 所选择的近似核心数据的大小在所要求大小范围内，比如当用户要求 θ 等于 0.8 时，近似算法选择了 4 条元组已经能够覆盖 7 个 AI 中的属性，而近似算法只需覆盖 5 个属性就可以了. 可以根据不同的策略，再选择 6 条元组，改进已有的选择结果，完成核心数据的选择. 由于数据集合中包含很多的空值，选出的 4 条元组都是不完整的，共消耗 78.31 s.

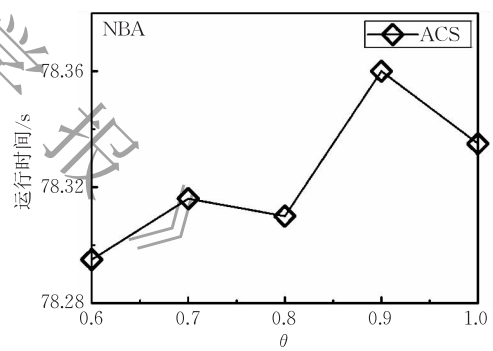

图 11　NBA 合成数据集合上 θ 的变化对运行时间的影响

可以发现，近似算法能够高效地选择近似的核心数据集合，θ 的变化，没有引起较大的时间消耗的波动. 在这组实验中，由于加入了额外的属性，很多的元组都是不完整的，但是也有着比较高的对 AI 的覆盖率，因此虽然 θ 在变化，但是实际需要覆盖的个数变化不大，因此选中 4 条覆盖率较高的元组后，选择就完成了.

(3) AI 的变化的影响

在这组实验中，通过改变用户感兴趣的属性集合 AI，观察上面 4 个指标的满足程度. 在实验中，在指定 AI 的个数后，随机选择感兴趣的属性. 用户指定的核心数据的大小 k 设为 10，用户指定的覆盖比

例 θ 设置为 0.9，AI 的大小从 6 个变化到 15 个. 实验结果如图 12 所示，其中横坐标给出了 AI 的大小，纵坐标给出了算法运行的时间.

通过得到的近似核心数据计算上面的 4 个指标，这 4 个指标都能够被满足. 所选择的近似核心数据的大小在所要求大小范围内，比如当用户要求 AI 大小为 11 时，近似算法选择了 8 条元组已经能够覆盖 10 个 AI 中的属性，而近似算法只需覆盖 7 个属性就可以了. 可以根据不同的策略，再选择 2 条元组，改进已有的选择结果，完成核心数据的选择. 由于数据集合中包含很多的空值，选出的 8 条元组都是不完整的，共消耗 80.34 s.

可以发现，近似算法能够高效地选择近似的核心数据集合，AI 的变化，引起较小的时间消耗的波动，在可以接受的范围内. 在这组实验中，AI 变化比较大，因此需要较多次的贪心选择来完成，但是由于仍然有完整度较高的元组，能够很快地覆盖 AI 中的属性，因此消耗的时间并没有比之前的实验增多太多.

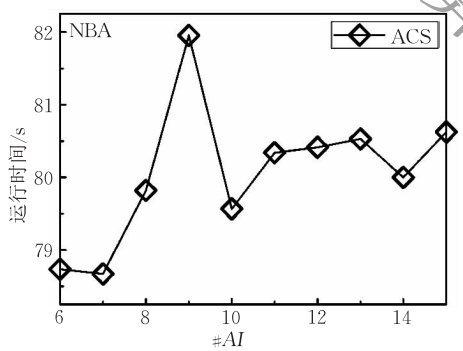

图 12 NBA 合成数据集合上 AI 的变化对运行时间的影响

(4) 元组的总数的变化的影响

在这组实验中，通过改变数据集合大小，观察上面 4 个指标的满足程度. 在实验中，随机选择 8 个属性作为感兴趣的属性集合 AI，用户指定的核心数据的大小 k 设为 10，用户指定的覆盖比例 θ 设置为 0.9. 通过设置每条元组的扩增倍数 Aug，然后按照前面描述的方式修改和加入元组得到数据集合. 扩增倍数 Aug 从 500 变化到 4000，根据前面的生成方式，修改后的元组以 0.8 的概率加入到数据集合中. 用于实验的文件的统计信息如表 2 所示，实验结果如图 13 所示，其中横坐标给出了 Aug 的大小，纵坐标给出了算法运行的时间.

通过得到的近似核心数据计算上面的 4 个指标，这 4 个指标都能够被满足. 所选择的近似核心数据的大小在所要求大小范围内，比如当扩增倍数为 500 时，近似算法选择了 6 条元组已经能够覆盖 7 个 AI 中的属性，而近似算法只需覆盖 5 个属性就可以了. 可以根据不同的策略，再选择 4 条元组，完成核心数据的选择. 由于数据集合中包含很多的空值以及 AI 属性集合的指定，选出的 6 条元组都是不完整的，共消耗了 86.165 s.

表 2 合成数据集合统计信息

Aug	元组数目	文件大小/G
500	7677760	2.57
1000	15354711	5.15
1500	23033674	7.73
2000	30714519	10.30
2500	38388546	12.80
3000	46073100	15.40
3500	53758446	18.00
4000	61431065	20.60

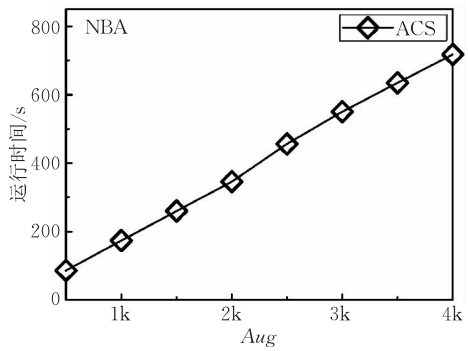

图 13 在 NBA 合成数据集合上数据规模对运行时间的影响

可以发现，近似算法能够高效地选择近似的核心数据集合. 虽然给定的 AI 集合的个数比较小，可以很快地被覆盖，但是由于元组的数目不断增多，因此每次贪心选择需要处理更多的属性和元组，因此时间消耗总体呈上升趋势，但是由于当给定 AI 集合时，近似算法的时间复杂性主要与元组的大小近似成线性关系，因此所消耗的时间的增长近似成线性，说明本文提出的算法具有良好的扩展性.

(5) 陌生数据集合上的核心数据集合的选择

对于一个陌生的数据集合，用户可能无法给出一个较好的核心数据集合的大小，那么选出的近似核心数据可能无法满足上面的 4 个指标. 在这组实验中，通过改变用户要求的核心数据的大小 k，观察算法选择的核心数据的大小以及覆盖指定的 AI 中的属性的个数，以及算法的运行时间. 在实验中，指定 9 个用户感兴趣的属性作为 AI，覆盖比例 θ 设置为 1，k 从 1 变化到 9. 在原有的 NBA 数据集合的基础上对每条元组扩增 10 次，对每个属性值以 0.2 的概率引入空值，产生的关系含有 191 970 条元组，实验结果见表 3、表 4 和表 5.

表 3　3 个算法的时间消耗

k	CMC 消耗时间/s	CWSC 消耗时间/s	ACS 消耗时间/s
1	431.242	1.113	1.095
2	432.125	1.109	1.119
3	433.485	1.109	1.123
4	131.602	1.110	1.123
5	87.921	43.930	1.126
6	87.826	43.897	1.124
7	87.789	43.972	1.124
8	87.831	44.892	1.126
9	87.961	44.767	1.125

表 4　3 个算法选出的核心数据集合覆盖的属性的个数

k	CMC 覆盖个数	CWSC 覆盖个数	ACS 覆盖个数
1	1	0	4
2	2	0	6
3	3	0	6
4	7	0	6
5	7	0	6
6	7	0	6
7	7	0	6
8	7	0	6
9	7	9	6

表 5　3 个算法选出的核心数据集合的大小

k	CMC 核心数据集合大小	CWSC 核心数据集合大小	ACS 核心数据集合大小
1	1	0	4
2	2	0	6
3	3	0	6
4	7	0	6
5	7	0	6
6	7	0	6
7	7	0	6
8	7	0	6
9	7	9	6

通过表 3 整体来看，算法 CMC 所用的时间最长，CWSC 所用的时间次之，本文所提出的算 ACS 所用的时间最短. 通过表 4 整体来看，CMC 算法在开始的时候没有达到 6 个属性的覆盖要求，从 k 超过 3 时可以达到要求；CWSC 只有 k 为 9 的时候所选出的近似核心数据才能达到覆盖的要求，并且全部覆盖了 9 个指定的属性；本文提出的算法 ACS 只有 k 为 1 时无法达到覆盖要求，此后选出的近似核心数据集合一直可以达到至少覆盖 6 个属性的要求. 通过表 5 整体来看，CMC 所选出的核心数据集合的大小在 k 不超过 3 时，能够满足指定的大小，此后选出的近似核心数据集合的大小不变，从超出指定的大小到在指定的大小以内；CWSC 除了 k 为 9 时能够选出满足大小的核心数据集合外，其余情况一直无法选出核心数据集合.

下面从算法角度进行分析.

CMC 算法具有理论上的覆盖指定属性的比例的保证，但是需要估计近似解的总的权值，只有估计到了这个权值，才能达到这个理论上的保证. 由于 CMC 算法采用试探的方法来猜测近似解的总的权值，并尝试在指定的大小内分层地贪心地选择可能的组合. 因此，每试探一次新的权值，就会枚举可能情况，虽然采用了分层的方式，过滤掉了很多不可能的情况，但是仍然需要大量的尝试. 因此，在 k 不超过 3 时，CMC 会进行多次尝试新的权值，由于数据本身的特点，核心数据集合大小在 3 以下时，无法满足覆盖条件，因此即使经历了很多的尝试，CMC 仍然无法选择一个满足覆盖要求的核心数据集合，却消耗了大量的时间. 当 k 超过 3 时，由于 CMC 认为完整的元组的权值是 0，因此可以尽量放到核心数据集合里面，因此选出了 7 条元组的核心数据集合，能够覆盖 7 个 AI 中的属性，因此，CMC 较快速地选择 7 条元组后就不再进行选择，但是这样的核心数据集合在 k 为 4，5，6 时是不符合用户指定的大小的.

CWSC 是一个快速的贪心式的核心数据选择算法，它没有理论上覆盖属性的个数的保证. 算法为了快速选择核心数据，要求每条元组都要满足一定比例的覆盖率，即在用户给定的 k 以内，一定能完成覆盖. 由于数据集合中的单条元组对指定属性集合的覆盖比例较低，因此 CWSC 简单计算后遍终止了，返回了“无法选择”来提示用户，等待用户给出一个合适的 k 值. 因此虽然 CMC 能在较短的时间内终止并给出提示，但是并没有选出一个近似核心数据集合，并且对于如何设置 k 值，也没有给出建议. 因此，只有用户给出了 k 为 9 时，CWSC 才快速地选择了一个近似核心数据集合，并覆盖率所有的属性.

本文提出的算法 ACS 在 k 为 1 时，无法选择满足覆盖要求的近似核心数据集合，只能尽可能地选择更多的完整元组来覆盖给定的属性，这里选择了 4 条完整元组来进行覆盖. 当 k 在 2 到 5 时，ACS 会尽量选择完整的元组来覆盖指定的属性，而每一次尝试都最多选择 k 条元组，因此选择的核心数据集合的大小会大于用户给定的大小，但是正如定理 2 给出的，选出的元组不会太多，有一个上界，并且选择的都是完整的元组. 因此，可以认为 ACS 对用户给定的 k 有一个修正作用，会在用户面对一个陌生的数据集合，无法确定核心数据集合大小时，给出一定的参考，并且会尽量选择完整的元组. 同时，如定理 2 所给出的，ACS 在覆盖指定的属性的比例上有

理论上的保证,因此除了 k 为 1 时,ACS 都能够高效地选出一个满足理论分析结果的近似核心数据集合.

为了进一步比较算法性能上的差异,对这个数据集合的每条元组进行了一定倍数的扩充,表 6 给出了在不同扩充倍数下的时间消耗,可以看出,正如前面的分析,面对陌生的数据,CMC 和 CWSC 的时间消耗增长很快,而本文的算法 ACS 时间消耗增长不是很快.其中扩充 1000 倍时,CMC 和 CWSC 的时间消耗太久,没有准确测量.

表 6 3 个算法在不同扩充倍数下的时间消耗

扩充倍数	CMC 消耗时间/s	CWSC 消耗时间/s	ACS 消耗时间/s
10	227.47	57.749	1.593
100	26486.90	6622.870	15.963
1000	360000+	360000+	187.462

5 相关工作

在缩减参与计算的数据量,从而提高计算的效率方面,已经有了很多的研究结果,主要的思想有数据摘要,数据压缩,索引方法和基于样本的近似计算方法.数据摘要[19,20]和数据压缩方法能提供简洁的数据表示方式,但是,如果直接用于计算通常需要额外的计算代价,来获得数据的某些具体的属性值[21].文献[14]研究了通过索引支持高效率的计算的方法.文献[19]介绍了用于大数据的数据摘要方法.文献[20]提出的方法将属性按照语义进行合并,得到层次的摘要结果.文献[23]提出了在样本中尽可能地保留原始数据的特点的抽样方法,但是,如果直接应用在不完整数据上,抽取的方法无法保证样本的完整程度.以上方法都没有考虑在不完整数据上进行计算时,如何减少信息缺失给查询结果的完整性带来的误差.

选择核心数据来近似回答用户查询的思想来自于文献[12].文献[13]研究了可以合并的核心数据的选择方法,同时考虑了覆盖性和多样性两个目标.文献[22]研究了来自传感器网络的数据中的支配数据的选择问题.但是,文中的方法去掉了大部分数据之间的差异,只能支持某些种类的查询.

本文要选择的核心数据集合要同时优化覆盖性,包含不完整元组的个数以及结果集合的大小三个目标.之前大部分工作,在完整的数据集合上进行选择,用代价表示不同数据的特点,同类数据虽然特点不一样,仍然可以混用;但是,本文要面对的数据是两类数据,即完整数据和不完整数据.需要通过控制在核心数据中的不完整元组的个数,来尽量减少信息缺失.之前的大部分工作,要么考虑覆盖性和代价,要么考虑覆盖性和结果集合大小,因此,不能直接用来解决本文所研究的问题,比如,加权的集合覆盖问题[24],研究的是在给定覆盖性下的代价优化问题;最大覆盖问题[25]研究的是,在固定结果集合大小时,如何优化覆盖率;而带有预算的最大覆盖问题[26],是在给定代价时,如何优化覆盖率.

文献[27]研究了类似于本文的问题,即具有结果大小约束下的加权集合覆盖问题,给出了两个启发式策略,以及优化的方法.但是,它的研究的数据仍是同一类有不同特点(权值)的数据,而本文研究的是完整的和不完整的两类数据,如果选择较多的不完整数据,那么会造成更多的信息丢失,因此,要尽量避免选择不完整的数据.此外,文献[27]假设总有一条元组能覆盖所有情况;文中的方法需要选择一个合适的迭代步长,来保证得到一个较好的近似比,但是这个步长通常很难选取;而且当待处理的元组带来的增益很分散时,文中给出的启发式算法所需要的迭代轮数是不可控制的.

6 结 论

本文研究了海量不完整数据的核心数据选择问题,证明了这个问题至少是一个 NP-难问题.基于贪心的策略,本文提出了一个具有质量保证的近似核心数据选择算法,并讨论了在不同的应用条件下的改进策略.通过不同数据集合上的实验表明,所提出的近似算法能够高效地选择一个高质量的近似核心数据集合,用来回答查询或者为后续的操作提供支持,所提出的近似算法具有良好的可扩展性.

参 考 文 献

[1] Saha B, Srivastava D. Data quality: The other face of big data //Proceedings of the 2014 IEEE 30th International Conference on Data Engineering. Chicago, USA, 2014: 1294-1297

[2] Swartz N. Gartner warns firms of 'dirty data'. Information Management Journal, 2007, 41(3): 6-7

[3] Eckerson W W. Data warehousing special report: Data quality and the bottom line. Application Development Trends, 2002, (5): 1-9

[4] Dong X L, Gabrilovich E, Murphy K, et al. Knowledge-based trust: Estimating the trustworthiness of web sources. Proceedings of the VLDB Endowment, 2015, 8(9): 938-949

[5] Li X, Dong X L, Lyons K, et al. Truth finding on the deep web: Is the problem solved?. Proceedings of the VLDB Endowment, 2012, 6(2): 97-108

[6] Khalefa M E, Mokbel M F, Levandoski J J. Skyline query processing for incomplete data//Proceedings of the 2008 IEEE International Conference on Data Engineering. Cancún, México, 2008: 556-565

[7] Miao X, Gao Y, Zheng B, et al. Top-k dominating queries on incomplete data. IEEE Transactions on Knowledge and Data Engineering, 2016, 28(1): 252-266

[8] Razniewski S, Nutt W. Completeness of queries over incomplete databases. Proceedings of the VLDB Endowment, 2011, 4(11): 749-760

[9] Deng T, Fan W, Geerts F. On the complexity of package recommendation problems. SIAM Journal on Computing, 2013, 42(5): 1940-1986

[10] Basu Roy S, Lakshmanan L V, Liu R. From group recommendations to group formation//Proceedings of the 2015 ACM International Conference on Management of Data. Melbourne, Australia, 2015: 1603-1616

[11] Lappas T, Liu K, Terzi E. Finding a team of experts in social neworks//Proceedings of the 15th ACM International Conference on Knowledge Discovery and Data Mining. Paris, France, 2009: 467-476

[12] Agarwal P K, Har-Peled S, Varadarajan K R. Approximating extent measures of points. Journal of the ACM, 2004, 51(4): 606-635

[13] Indyk P, Mahabadi S, Mahdian M, et al. Composable core-sets for diversity and coverage maximization//Proceedings of the 33rd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems. Utah, USA, 2014: 100-108

[14] Pirk H, Manegold S, Kersten M. Waste not … Efficient co-processing of relational data//Proceedings of the 2014 IEEE 30th International Conference on Data Engineering. Chicago, USA, 2014: 508-519

[15] Potti N, Patel J M. DAQ: A new paradigm for approximate query processing. Proceedings of the VLDB Endowment, 2015, 8(9): 898-909

[16] Agarwal S, Mozafari B, Panda A, et al. BlinkDB: Queries with bounded errors and bounded response times on very large data//Proceedings of the 8th ACM European Conference on Computer Systems. Prague, Czech Republic, 2013: 29-42

[17] Wang X, Dong X L, Meliou A. Data X-Ray: A diagnostic tool for data errors//Proceedings of the 2015 ACM International Conference on Management of Data. Melbourne, Australia, 2015: 1231-1245

[18] Agarwal S, Milner H, Kleiner A, et al. Knowing when you're wrong: Building fast and reliable approximate query processing systems//Proceedings of the 2014 ACM International Conference on Management of Data. Utah, USA, 2014: 481-492

[19] Cormode G, Garofalakis M, Haas P J, et al. Synopses for massive data: Samples, histograms, wavelets, sketches. Foundations and Trends in Databases, 2012, 4(1-3): 1-294

[20] Selçuk Candan K, Cao H, Qi Y, et al. AlphaSum: Size-constrained table summarization using value lattices//Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology. Saint-Petersburg, Russia, 2009: 96-107

[21] Granados A, Koroutchev K, de Borja Rodriguez F. Discovering data set nature through algorithmic clustering based on string compression. IEEE Transactions on Knowledge and Data Engineering, 2015, 27(3): 699-711

[22] Cheng S, Cai Z, Li J, et al. Drawing dominant dataset from big sensory data in wireless sensor networks//Proceedings of the 2015 IEEE Conference on Computer Communications. Hong Kong, China, 2015: 531-539

[23] Acharya S, Gibbons P B, Poosala V. Congressional samples for approximate answering of group-by queries. ACM SIGMOD Record, 2000, 29(2): 487-498

[24] Chvatal V. A greedy heuristic for the set-covering problem. Mathematics of Operations Research, 1979, 4(3): 233-235

[25] Hochbaum D S, Pathria A. Analysis of the greedy approach in problems of maximum k-coverage. Naval Research Logistics, 1998, 45(6): 615-627

[26] Khuller S, Moss A, Naor J S. The budgeted maximum coverage problem. Information Processing Letters, 1999, 70(1): 39-45

[27] Golab L, Korn F, Li F, et al. Size-constrained weighted set cover//Proceedings of the 2015 IEEE International Conference on Data Engineering. Seoul, South Korea, 2015: 879-890

LIU Yong-Nan, born in 1986, Ph. D. candidate. His research interests include data quality and data completeness.

LI Jian-Zhong, born in 1950, professor, Ph. D. supervisor. His research interests include databases and wireless sensor networks.

GAO Hong, born in 1966, professor, Ph. D. supervisor. Her research interests include databases and wireless sensor networks.

Background

Data quality is a heated research direction in the big data era, which has been investigated extensively in different perspectives. Incomplete data is a ubiquitous issue in data quality, which brings challenges to algorithms for querying, mining and analyzing. Incomplete data seriously reduce usability of data in real applications.

Selecting a subset of a user's interest from entire incomplete dataset is an effective and efficient mechanism to improve usability of incomplete data. Most prior work on selecting core-sets focuses on sampling numeric values for approximate aggregation queries, obtaining synopses or constructing histograms, which cannot handle categorical values and may lose many features of entire data. Using traditional methods based on set cover, tuples covering some attributes of a user's interest are selected, but there may be too many tuples for a user for further consideration or follow-up operations. In this paper, we investigate the problem of selecting core-sets with size constrained from incomplete data given attributes of a user's interest. Such problem is proved to be at least NP-Hard. Therefore, we design an approximate core-set selection algorithm based on greedy, called ACS, which is proposed in this paper. The proposed algorithm ACS selects tuples containing complete values as many as possible to obtain an approximate core-set, and the number of selections is limited to obtain a subset of tuples with size limited, and the ratio of attributes of interest covered is guaranteed as well. By theoretical analysis, within nearly linear time, ACS can select an approximate core-set of size within the logarithm factor to the given size, where the ratio of attributes of interest covered is at least $(1-1/e)$, with incomplete tuples of at most the size of the required core-set, where e is the base of the natural logarithm. Experimental results conducted on two real-world datasets of DBLP and NBA players show effectiveness and efficiency of the proposed algorithm ACS, and experimental results on synthetic datasets with larger scale show scalability of ACS.

To obtain an approximate core-set with high completeness, the algorithm mainly selects tuples from complete tuples as many as possible to cover attributes of interest. But there may still be attributes of interest not covered. Then the algorithm has to select incomplete tuples, but the number is strictly limited. Since the coverage can be guaranteed theoretically, the algorithm can return a good approximate core-set for further use.

This work is supported in part by the Key Research and Development Plan of National Ministry of Science and Technology under Grant No. 2016YFB1000703, and the Key Program of the National Natural Science Foundation of China under Grant Nos. 61632010, 61502116, 61370217, U1509216.