基于极小独立支配集的多样化排序算法

印 佳 程春玲 周 剑
(南京邮电大学计算机学院 南京 210003)

摘 要 为了满足用户的多元化需求和提高用户查询的满意度,出现了多样化排序算法的研究,但是目前多样化排序算法在多样化和相关性之间不能达到很好的平衡,且查询处理效率不能完全适应实际的交互需求,为此提出了一种基于极小独立支配集的多样化排序算法。将多样化子集选取问题转化为无向加权图的极小独立支配集的求解问题,以此兼顾查询结果的多样化和相关性;在求解过程中通过引入抛弃子集的概念来减少冗余顶点对之间距离的比较,加快算法求解的速度。仿真实验表明,所提算法在多样化性能和查询处理效率方面有一定的提升。

关键词 多样化排序,查询处理,极小独立支配集,抛弃子集
中图法分类号 TP311 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2017.08.032

Diverse Ranking Algorithm Based on Minimal Independent Dominating Set

YIN Jia CHENG Chun-ling ZHOU Jian
(School of Computer Science, Nanjing University of Posts and Telecommunications, Nanjing 210003, China)

Abstract In order to meet the diversified needs and improve the satisfaction of user's query, the research of the diverse ranking algorithm has been developed. However, the current diverse ranking algorithm cannot achieve good balance between diversification and relevance, and the efficiency of query processing cannot fully meet the actual needs of the interaction. Thus, a new diverse ranking algorithm based on minimal independent dominating set(MIDS-DR) was proposed. First, the problem of selecting diversified subset is transformed into the minimal independent dominating set of the undirected weighted graph, so as to balance the diversification and relevance of query results. To speed up the algorithm, the concept of abandoned subset is introduced to reduce the comparison of distances between redundant vertex pairs during the solving process. The simulation results show that the proposed algorithm can improve the performance and query efficiency.

Keywords Diverse ranking, Query processing, Minimal independent dominating set, Abandoned subset

1 引言

随着大数据时代的发展,信息出现多元化,用户的查询需求也越来越丰富,信息检索系统需要朝多元化趋势发展,并提供多种类型的查询结果来满足不同层次用户的潜在需求,从而提升用户查询的体验度和满意度。其中,对于查询结果的排序问题一直是信息检索领域关注的重点,为此,国内外在查询结果组织方面提出了多样化排序的概念[1],即根据某种规则在有限数目的查询结果中实现多样化,使得靠前排序列表的查询结果的多样化效果明显。查询结果多样化的研究在消除用户初始查询的模糊性以及进一步帮助用户缩小查询范围方面非常具有研究意义和研究价值。

目前,多样化需求主要来源于两个方面:1)用户查询的不确定性,即不明确自己的查询意图;2)查询本身的语义模糊性,即大多数查询可能包含多种解释,涵盖多个方面的信息。在上述情况下,通过提供多样化的排序列表来满足不同用户的查询需求至关重要。然而,多样化排序的目标不仅仅是提高查询结果的丰富性,还要保证最终查询结果和用户初始查询需求之间的相关性,现有的多样化排序算法大都基于评分机制来返回多样化的排序列表,要么过多地侧重于查询的相关性而导致冗余情况的出现,要么倾向于多样化而导致查询结果脱离初始查询需求,在相关性和多样化之间不能达到显著的平衡,因此需要设计合理的选取规则来改善这种状况。同时,大多数算法在选取多样化子集时忽视了查询处理效率低的问题,不能保证实时准确地满足用户的查询需求,因此需要提高查询处理的效率。

针对上述分析,本文基于图论的核心思想,设计了一种基于极小独立支配集的多样化排序算法(Diverse Ranking algo-

到稿日期:2016-07-01 返修日期:2016-11-09 本文受国家自然科学基金(71301081,61403208),江苏省自然科学基金(BK20130877),国家博士后基金(2014M551637),江苏省博士后基金(1401046C)资助。

印 佳(1992-),女,硕士生,主要研究方向为信息检索,E-mail:yinjia920117@sina.com;程春玲(1972-),女,教授,硕士生导师,CCF会员,主要研究方向为数据管理、资源管理和优化等,E-mail:chengcl@njupt.edu.cn(通信作者);周 剑(1984-),男,博士,副教授,硕士生导师,CCF会员,主要研究方向为决策支持系统。

rithm based on Minimal Independent Dominating Set, MIDS-DR)，主要工作为：1)用无向加权图来表示初始查询结果集；2)利用无向加权图的支配集和独立集的特点，把多样化子集选取问题抽象为求解无向加权图的极小独立支配集的问题；3)求解多样化子集选取问题时，改进初始选取条件并引入抛弃子集的概念来处理冗余顶点，最终选取满足条件的顶点加入多样化子集中。

2 相关工作

目前多样化排序方法从不同的角度分析和解决了用户的多样化需求，根据查询与蕴含的用户意图的关系可分为两大类[2]，即显式多样化排序和隐式多样化排序。

显式多样化排序[2]建模各个查询与各种潜在意图之间的相关度，并根据相应的目标函数来实现多样化子集的选取，典型的有离线和在线两大类。离线的显式多样化排序利用子主题和查询的相关性来达到多样化程度，Santos 基于 xQuAD 框架[1]挖掘出了查询所蕴含的子查询，利用子查询的重要性、文档对子查询的覆盖度、新颖度和查询相关度来预估给定查询的代表性细节的重要性；任鹏杰[3]综合考虑语义维度和时效性维度，通过分析查询日志来预估给定查询的时效性意图的概率分布；Wu[4]提出数据融合的方法，利用线性组合，在权值分配中考虑结果的有效性和结果间的互补性以促进排序列表的生成；Yu[5]把选取多样化子集的任务看成是 0-1 多个子主题背包问题，利用图模型和最大置信传播算法来选取最优子集并将其填入多个子主题背包中；Hu 提出了层次化意图的多样化模型[6]，基于主题新颖和比例模型来计算子主题每个层次的多样化分数以选取最优结果。这类离线方式一定程度上提高了查询结果的多样化，但是其子查询的生成过程比较复杂，比如上述 xQuAD 框架中的子查询通过 k-means 算法对查询结果集进行聚类并利用查询扩展模型从每一类簇中选取代表性的结果作为子查询，其构建时间较长，且有些方法获取训练数据的代价较高，很难应用于实际。

在线的显式多样化排序算法主要基于用户的交互来优化排序模型，主要有多臂赌徒机排序算法[7](Rank Bandits Algorithm, RBA)、Schuth 提出的多叶梯度下降算法[8](Multileave Gradient Descent, MGD)、概率多叶梯度下降算法[9](Probabilistic Multileave Gradient Descent, P-MGD)等。RBA 算法基于多臂赌徒机的策略优化用户的点击反馈，在线调整排序模型，实时保证用户点击的查询意图；Harrie 基于 MGD 提出了 P-MGD 算法，利用概率交错的方法实现算法的高敏感和不带偏见并加快了学习的速度。这类方法注重用户的交互影响，在一定程度上提高了算法的更新速率，但没有深入考虑查询结果间的多样化。

隐式多样化排序主要基于相似的查询包含相似的细节来选择多样化子集，早期的最大边际相关算法[10](Maximal Marginal Relevance Algorithm, MMR)根据文档相关性和新颖度来迭代选取最大化目标函数的多样化子集，Xia[11]基于此提出了使用评价作为边际的感知算法(Perception Algorithm using Measures as Margins, PAMM)，利用序列文档选择的过程，直接优化多样化评价指标来学习最大边际相关模型；Zhu[12]提出的 Grasshopper 算法通过考虑吸收马尔可夫随机游走的特性来加强查询结果的多样化；而文献[13]的关系排序学习框架(Rational Learning-To-Tank, R-LTR)基于排序函数和损失函数来预测新的多样化排序列表，将判别学习函数定义为双准则的目标函数，首次建立了统一的多样化排序学习框架。

综上所述，虽然显式多样化排序算法在多样化程度上能有效地满足不同用户的查询需求，但其昂贵的训练数据和算法执行代价使其不能很好地适用于实际，因此本文基于隐式多样化排序的基本思想，为保证查询结果的相关性与多样化之间的平衡以及算法的执行效率，设计了一种新的多样化排序算法。

3 基于极小独立支配集的多样化排序算法

3.1 算法思想

多样化排序的目的是使得在排序列表靠前的位置上返回多样化的查询结果，这些查询结果之间包含差异性最优。根据用户的初始查询 *Query* 得到初始查询结果集 $QR=\{qr_1, qr_2, \cdots, qr_m\}$，其中 $qr_i=(ar_1, ar_2, \cdots, ar_n)$，$1\leqslant i\leqslant m$，对应于 n 维欧氏度量空间 X^n 中的一个向量或一个点，表示含有 n 个属性的任意一条结果元组，m 为查询结果集的个数。无向加权图 $RDG=\langle RV, RE, RW\rangle$，两个顶点之间的距离通过连边权重集合来表示，而 X^n 中任意两个向量之间的关系通过距离来表示，距离的值越大，差异性越大，反之则相似度越大。因此，度量空间中的向量可以对应于 RDG 中的顶点，两个向量之间的关系对应于两个顶点之间的连边关系，通过连边权重值的大小来比较两个向量之间的包含差异性；同时，从 RDG 中选取的独立集是指选取的顶点在 RDG 中没有边相连，对应于选取的向量集之间的差异性较明显；从 RDG 中选取的支配集是指选取的顶点和剩余顶点中至少有一条边相连，对应于选取的向量集之间存在相似性，因此多样化排序问题可以转化为给定 n 维欧氏度量空间 X^n 中的多个向量，根据向量之间的相似性和差异性，求解包含差异性最优的向量集并排序输出。无向加权图计算问题可以转化为给定 n 维欧氏度量空间 X^n 中的多个顶点，根据顶点之间的连边关系，求解 RDG 的极小独立支配集。

由此，多样化排序问题的初始查询结果集可以用无向加权图的顶点集来表示，即 $RV=QR=\{qr_1, qr_2, \cdots, qr_m\}$，顶点之间的连边关系对应于查询结果元组之间的距离关系，多样化子集根据无向加权图的独立支配集来产生，因此本文将求解多样化排序中的多样化子集问题转化为求解无向加权图的极小独立支配集问题。

下面以用户输入的 *Query*=monte carlo 为例来说明该算法的思想。该查询包含多种不同的意思，比如蒙特卡罗方法、蒙特卡洛城市、蒙特卡罗电影、蒙特卡罗纸牌游戏、蒙特卡罗歌名等，刚开始用户可能不清楚所要查询的具体范围，因此需要在排序列表靠前的位置提供多样化的查询结果来帮助用户进一步缩小其查询范围。以初始查询结果集中的 7 条查询结果集为例，即 $QR=\{qr_1, qr_2, qr_3, qr_4, qr_5, qr_6, qr_7\}$，其中 $qr_i=\{ID, URL, title, snippet\}$，具体内容见 AMBIENT 数据集[14]。

首先将查询结果集对应到欧氏度量空间 X^n，以 qr_1 和 qr_2 为例，将 QR 归一化处理后，qr_1 对应 X^n 中的向量 $u_1(0.3,-0.1)$，qr_2 对应 X^n 中的向量 $u_2(0.1,0.4)$，则 $d(u_1,u_2)\approx 0.53$，这为向量 u_1 和 u_2 的欧氏距离，表示 qr_1 和 qr_2 的差异性；同样地，可以得到 QR 中其他查询结果集之间的关系。QR 中的查询结果集分别对应无向加权图的顶点集，查询结果集之间的距离对应无向加权图中顶点的连边关系和权重，如图 1 所示。

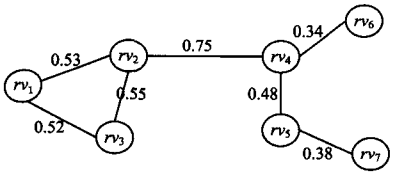

图 1　查询结果集的无向加权图

3.2　无向加权图的构建

本文采用简单的无向加权图来表示初始查询结果集，即 $RDG=\langle RV,RE,RW\rangle$ 是一个三元组，其中：

(1) $RV=\{rv_1,rv_2,\cdots,rv_m\}$ 为 RDG 的顶点集合，对应于欧氏度量空间中的向量集合，即 $RV=QR=\{qr_1,qr_2,\cdots,qr_m\}$，每一个顶点表示一条初始查询结果，$m$ 表示顶点个数，其中 $rv_i=qr_i=(ar_1,ar_2,\cdots,ar_n)$，$1\leqslant i\leqslant m$。

(2) $RE=\{re_1,re_2,\cdots,re_c\}$ 为顶点之间的边集合，如果 $d(rv_i,rv_j)\leqslant T$，则顶点 rv_i 和 rv_j 之间存在一条边，其中 $d(rv_i,rv_j)$ 表示顶点 rv_i 和 $rv_j\in RV$ 之间的欧氏距离，T 表示距离阈值，c 表示边的总数，且 $1\leqslant c\leqslant m(m-1)/2$。

为了得到距离阈值 T，根据欧氏度量空间距离可度量的特点以及欧氏距离能体现出个体数值特征的绝对差异的特点，计算出 RDG 中各个顶点之间的欧氏距离，构成距离集合 $DS=\{d_1,d_2,\cdots,d_{m(m-1)/2}\}$，则距离阈值 T 的计算公式为：

$$T=\frac{1}{m(m-1)/2}\sum_{i=1,j=1}^{m}d_{ij} \tag{1}$$

(3) $RW=\{rw_1,rw_2,\cdots,rw_c\}$ 表示顶点之间的连边权重集合，c 表示边的数目，如果两个顶点 rv_i 和 rv_j 之间存在一条边，那么 rw_{ij} 为两个顶点之间的欧氏距离 d_{ij}，否则为 0，即 rw_{ij} 满足式(2)：

$$rw_{ij}=\begin{cases}d_{ij}, & d(rv_i,rv_j)\leqslant T\\ 0, & d(rv_i,rv_j)>T\end{cases} \tag{2}$$

进而，无向加权图 RDG 的求解过程为：

(1) 利用式(1)得到每个顶点 $rv_i\in RV$ 的近邻集，表示为 $RVN(rv_i)=\{rv_j\mid rv_i\neq rv_j,d(rv_i,rv_j)\leqslant T,1\leqslant i,j\leqslant m\}$。

(2) 初始选取 $i=1$ 时，确定 $RVN(rv_1)$ 中的顶点和 rv_1 相连，直到 $RVN(rv_1)$ 中所有顶点均被访问到，然后执行 $i=i+1$，重复执行上述操作，直到 RV 中的所有顶点都被访问到为止。

3.3　多样化子集的选取

现有多样化子集选取条件基于相关性和多样化的联合评分来选取分数最高的子集加入多样化子集，其平衡参数不能很好地兼顾相关性和多样化，为此本文基于无向加权图的极小独立支配集来构成多样化子集，使其能主动适应多样化和相关性之间的平衡，保证了查询结果集的包含差异性最优。

根据图 $RDG=\langle RV,RE,RW\rangle$ 可以得出多样化子集，并将其表示为 $MIDS$，其中：$MIDS=\{dv_1,dv_2,\cdots,dv_s\}$，且 $MIDS\subseteq RV$，表示满足以下定义的顶点集合，s 表示多样化子集的总数且 $1\leqslant s\leqslant m$。

定义 1(RDG 的独立集 I)　对于任意的 $rv_i,rv_j\in RV$，$1\leqslant i,j\leqslant m$，如果 $d(rv_i,rv_j)>T$，那么 rv_i,rv_j 属于 RDG 的独立集 I 中的顶点。

定义 2(RDG 的支配集 D)　对于任意的 $rv_x\in RV$，如果图 RDG 中存在一个剩余顶点 rv_y，满足 $d(rv_x,rv_y)\leqslant T$，那么 rv_x 属于 RDG 的支配集 D 中的一个顶点。

将多样化子集求解问题建模为简单无向加权图 RDG 的极小独立支配集求解问题，该问题是一个单目标多约束问题，其数学描述如下：

$$\min\sum_{i=1}^{m}u_{rv_i}v_{rv_i} \tag{3}$$

$$\text{s. t. } u_{rv_k}=\begin{cases}1, & rv_k\in I\\ 0, & rv_k\notin I\end{cases},1\leqslant k\leqslant s \tag{4}$$

$$v_{rv_k}=\begin{cases}1, & rv_k\in D\\ 0, & rv_k\notin D\end{cases},1\leqslant k\leqslant s \tag{5}$$

$\sum_{i=1}^{m}u_{rv_j}x_{rv_irv_j}\geqslant 1$，其中：

$$x_{rv_irv_j}=\begin{cases}1, & rv_i\notin RVN(rv_j)\\ 0, & rv_i\in RVN(rv_j)\end{cases},1\leqslant j\leqslant s \tag{6}$$

$\sum_{p=1}^{m}v_{rv_q}y_{rv_prv_q}\geqslant 1$，其中：

$$y_{rv_prv_q}=\begin{cases}1, & rv_p\in RVN(rv_q)\\ 0, & rv_p\notin RVN(rv_q)\end{cases},1\leqslant q\leqslant s \tag{7}$$

式(3)表示所有顶点中满足独立支配集条件的数目，并且数目最少；式(4)和式(5)表示式(3)中 u,v 的取值，如果顶点满足独立集或支配集条件，则取值为 1，否则取值为 0；式(6)保证 RDG 中独立集存在且式(1)合理，其中 $x_{rv_irv_j}$ 表示保证独立集中的顶点在图 RDG 中相互之间没有边相连，减少了相关性的冗余；同样地，式(7)保证存在支配集，其中 $y_{rv_prv_q}$ 保证支配集中至少存在一个顶点与图 RDG 中剩余顶点有边相连，间接保证了多样化子集中所有顶点与剩余顶点之间的关联性，不脱离原始查询的特点。

求解该最优化问题时，为了减少距离对之间的比较次数，引入抛弃子集 AS 的概念，并改进初始的随机选取策略，使得在后继选取目标顶点时能够减少与初始顶点比较的次数。

定义 3(抛弃子集 AS)　极小独立支配集 MIDS 中所有顶点的近邻集组成的顶点集合称为抛弃子集，特点是在下一次选取时无需考虑该子集中的顶点，即：

$AS=\{rv_i\mid rv_i\in RVN(rv_j),rv_j\in MIDS\}$

求解上述最优化问题的具体步骤为：

(1) 构造顶点等级次序表，根据每个顶点的近邻集数目，并利用广度优先搜索 BFS 将 RDG 中的所有顶点分成不同的等级从而生成一张次序表，$rate=|RVN(rv_i)|$，$1\leqslant i\leqslant m$，次序表根据 $rate$ 由大到小列出每个等级中的顶点。

(2) 初始选取 $rate=\max|RVN(rv_i)|$ 的顶点 rv_i 加入 MIDS 中，将 $RVN(rv_i)$ 中的所有顶点加入 AS 中并更新次序表，即删除次序表中初始顶点及其近邻集中的顶点，如果存在多个 rv_i，则选取 i 较小的那一个作为初始顶点。

(3) 继续检查次序表的 $rate=\max|RVN(rv_i)|$ 等级中是否存在顶点，如果存在且不止一个，则分别比较其中的顶点与

初始顶点之间的距离，距离小的优先加入 MIDS 中，同时将其近邻集的顶点加入 AS 中并更新次序表，然后检查该等级中是否有剩余顶点，如果存在则重复上述操作直到该等级中不存在顶点为止；否则执行步骤(4)。

(4)如果不存在顶点，则执行 $rate=rate-1$，重复执行步骤(3)，直到次序表中不存在顶点为止。

以上述 *Query*=monte carlo 为例来说明多样化子集选取的过程及其结果，基于 3.2 节和 3.3 节的算法设计，首先得到 $RVN(rv_1)=\{rv_2,rv_3\}$，$RVN(rv_2)=\{rv_1,rv_3,rv_4\}$，$RVN(rv_3)=\{rv_1,rv_2\}$，$RVN(rv_4)=\{rv_2,rv_5,rv_6\}$，$RVN(rv_5)=\{rv_4,rv_7\}$，$RVN(rv_6)=\{rv_4\}$，$RVN(rv_7)=\{rv_5\}$，根据每个顶点包含的近邻集数目构造顶点等级次序表，初始选取 $rate=\max|RVN(rv_i)|=3$，即 rv_2，因此将 $RVN(rv_2)$ 中的剩余顶点加入 AS 中，继续检查次序表中的顶点，还剩 rv_5，rv_6，rv_7；按照上述迭代进程，依次选取 rv_5，rv_6 加入 MIDS 中，rv_7 加入 AS 中，因此可得 $MIDS=\{rv_2,rv_5,rv_6\}$，$AS=\{rv_1,rv_3,rv_4,rv_7\}$，则得到的极小独立支配集为 $\{rv_2,rv_5,rv_6\}$，最终的排序列表为 $qr_2,qr_5,qr_6,qr_1,qr_3,qr_4,qr_7$。

3.4 算法描述

根据 3.1—3.3 节的算法设计，可以将 MIDS-DR 算法分为 3，即无向加权图构建、多样化子集选取和排序输出部分，具体的算法伪代码如算法 1 所示。

算法 1 MIDS-DR 算法伪代码

```
Input:查询 Query 的初始查询结果集 QR={qr1,qr2,…,qrm}
Output:查询 Query 的多样化排序结果列表 list
step1 构建无向加权图
1.  for(i=1;i≤m;i++){
2.    QR→X^n;//将 QR 映射到 R^n 中,得到集合 QR={qr1,qr2,…,qrm}
3.  }
4.  for(i=1;i≤m;i++){
5.    for(j=1;j≤m;j++){
6.      double d(qri,qrj)=Euclidean_dist(qri,qrj);//计算对象之间的欧氏距离
7.    }
8.  }
9.  for(i=1;i≤m;i++){//构建无向加权图
10.     construct RDG;
11. }
step2 选取多样化子集
1.  MIDS=∅,AS=∅;
2.  for(i=1;i≤m;i++){
3.      RVN(rvi)=RangeQuery(rvi,T);
4.      Count| RVN(rvi)|;//统计每个顶点的近邻集数目
5. }
6. if (|RVN(rvi)| is max) then
7.    rate=max|RVN(rvi)|,i=1;
8.    MIDS={rvi};//选择包含近邻集最多的顶点作为初始顶点
9.    AS=RVN(rvi);//将初始顶点的近邻集加入抛弃子集
10.   Use BFS to construct L;//利用广度优先搜索算法和|RVN(rvi)|来构造顶点等级次序表 L
11. while(L is null){//根据顶点等级来构造 MIDS 和 AS
12.     rate=rate-1;
13.     select rvk from the L;
14.     for(each rvk){
15.       compare d(rvi,rvk);
16.       select the rvk when min d(rvj,rvk);
17.       MIDS=MIDS∪{rvk};
18.       AS=AS∪RVN(rvk);
19.     }
20. }
step3 排序输出:根据选取的 MIDS 和 AS 按序输出最终的多样化排序列表 list
```

3.5 几种算法之间的比较

本节将本文算法 MIDS-DR 与 MMR 算法、Grasshopper 算法以及 MSKP 算法进行了比较。

MMR 算法是经典的最大边际相关的启发式算法，它以最大化多样化分数为目标选取多样化子集，虽然选取的查询结果目标是最大的，但是并不能保证相关性和多样化之间的平衡；同时，每次选取多样化子集时，需要遍历初始查询结果集中的所有对象，造成查询处理效率低下。为此，本文算法 MIDS-DR 利用独立支配集这一特性来选取多样化子集，在一定程度上保证了两者之间的平衡，同时改进初始化策略，减少了冗余对象之间的比较次数。

Grasshopper 算法利用马尔可夫链的平稳分布特性来选取第一个对象，既考虑了文档的集中性，又把文档与查询的相关性融入其中，具有一定的优势；但是在其选取过程中对象之间的比较基于吸收马尔可夫链中状态的平均访问次数，该策略偏向对象的集中性，容易造成选取的多样化子集之间信息的冗余。而本文算法 MIDS-DR 中改进的初始化策略虽然选取的第一个对象没有 Grasshopper 算法效果明显，但是在之后的选取过程中本算法比较占优势，通过抛弃子集的概念来减少对象之间的比较次数，而且独立支配集的特性也能很好地减少对象之间的冗余度。

MSKP 算法将选取多样化子集的问题看成是 0-1 多个子主题背包问题，利用最大总和置信传播算法来获取最大联合概率，其迭代次数和迭代时间相对减少，但是基于联合概率选取的多样化子集比较侧重于查询结果的相关性，多样化性能不是很明显；而 MIDS-DR 算法把多样化子集选取问题看成求解无向加权图的极小独立支配集问题，根据独立集和支配集的两个特点来衡量多样化和相关性，有一定的依据。

从算法 1 的描述可以看出，在 step1 中计算对象之间的欧氏距离的时间复杂度为 $O(m^2)$，构建无向加权图需遍历每个顶点及近邻集，其时间复杂度为 $O(m+m(m-1)/2)$；step2 中统计每个顶点的近邻集数目的过程需要进行 m 次，其时间复杂度为 $O(m)$，根据广度优先搜索算法构造顶点等级次序表需要遍历 m 个顶点，然后根据顶点等级来构造 MIDS 和 AS，在最坏情况下，在每个等级中需要比较距离对 $m\log m$ 次，则该过程的时间复杂度为 $O(m\log m)$，step2 中时间复杂度为 $O(m\log m+km)$，即 MIDS-DR 算法的时间复杂度为 $O(m\log m+km)$。根据上述几种算法的比较，表 1 列出了了 4 种算法的时间复杂度，从中可以看出本文算法在选取多样化子集时取得了较低的时间复杂度。

表 1　4 种算法的时间复杂度比较

算法	时间复杂度
MMR 算法[10]	$O(m^2)$
Grasshopper 算法[12]	$O(km^2)$
MSKP 算法[5]	$O(nm\log m+km)$
MIDS-DR 算法	$O(m\log m+km)$

4　仿真实验与结果分析

4.1　实验环境设置

实验采用公开的数据集 AMBINENT[14] 来衡量 MIDS-DR 算法的多样性，数据集包括 44 个主题，每个主题包含一定的子主题集合和 100 个已排序的初始查询结果文档。主题和子主题基于维基百科获取，查询结果集通过将主题名作为查询词从 Yahoo!搜索引擎中获取，基于维基百科上列出的每个主题对应的子主题信息通过人工来标注子主题的相关性。每个查询结果由查询结果的 URL、标题和摘要组成。

实验的软硬件环境：采用数据库 MySQL 5.0 对数据集进行存储，在 MyEclipse8.5 中对 MMR 算法[10]、Grasshopper 算法[12]、MSKP 算法[5] 和本文算法 MIDS-DR 进行仿真。实验采用 1 台 PC 机进行，其配置为 2.59GHz Intel(R) Core (TM) i5-4210M CPU、4GB 内存、450GB 硬盘。

实验从多样化和查询性能两方面对算法进行评估，多样化性能采用文献[15]中的子主题召回率和平均准确率两个指标来评价，子主题召回率和平均准确率越高说明其多样化性能越好；查询性能通过查询处理效率来评价，查询处理时间和距离对的比较次数越少说明查询性能越高。

4.2　实验结果及其分析

4.2.1　多样化性能

本节对 AMBINENT 数据集中的所有主题进行了实验。由于用户比较关注排序列表前 20 的查询结果，分别记录各个算法前 20 的查询结果与查询主题相关的子主题数目以及相关性，根据式(8)和式(9)分别计算出排序列表位置前 20 的子主题召回率和平均准确率，同时对所有主题的子主题召回率和平均准确率求均值，实验结果如图 2 和图 3 所示。

$$S\text{-}rec@k(q)=\frac{|\bigcup_{i=1}^{k} Subtopic(qr_i)|}{n_t} \tag{8}$$

其中，n_t 表示对应的查询 q 相关的初始查询结果总数，$Subtopic(qr_i)$ 表示产生的 qr_i 和查询 q 相关的子主题数目。

$$AP@k(q)=\frac{\sum_{k=1}^{n} P@k(q)*l_k}{n_t} \tag{9}$$

其中，l_k 表示位置 k 上查询的相关性，$P@k(q)$ 表示位置 k 上的准确率($Precision@k$)，即：

$$P@k(q)=\frac{1}{k}\sum_{i=1}^{k} l_k \tag{10}$$

从图 2 可以看出，4 种算法在所有主题下的平均子主题召回率大约在 35%～60%之间，整体上，本文算法 MIDS-DR 在排序位置 3－20 上的平均子主题召回率高于其他 3 种算法；但是在排序位置 1 上，Grasshopper 算法的平均子主题召回率最高，达到了 60%以上，多样化性能最优，从而验证了 3.5 节的分析结果，基于随机游走特性选取的初始项取得了很好的多样化性能。然而，随着排序位置的不断变化，尤其是在排序位置 11－20，Grasshopper 算法的平均子主题召回率远远不如 MSKP 算法和 MIDS-DR 算法，而 MSKP 算法在排序位置前期波动较大，后期随着图模型的置信传播算法不断收敛，其子主题召回率趋于平缓，相比较而言，MIDS-DR 算法虽然在排序位置 1 上取得的平均子主题召回率不高，但是随着排序位置的不断变化，尤其是在排序位置 5－10 时其召回率远远高于其他算法，这说明独立支配集的特点符合多样化选取规则且多样化效果明显优于其他算法。而 MIDS-DR 算法在排序位置 10－16 上的平均子主题召回率有所下降，这说明 MIDS-DR 算法的多样化性能还与查询项包含的子主题数目相关，包含的数目越多，其子主题召回率越高。

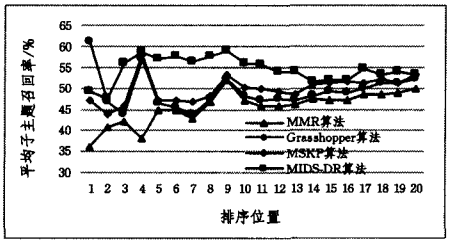

图 2　4 种算法在所有主题下的平均子主题召回率

从图 3 可以看出，4 种算法在所有主题下的平均准确率在 40%～65%左右波动，MIDS-DR 算法的平均准确率最高达到了 63%左右，相比于其他算法取得了最高的平均准确率，这说明其多样化效果较明显，而且在排序位置 2－20 上，其平均准确率均高于其他算法，特别是在排序位置 6－12 上，其平均准确率远远高于其他 3 种算法，这说明基于极小独立支配集特性选取的多样化子集的多样化性能较高。

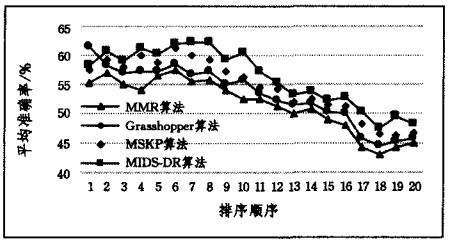

图 3　4 种算法在所有主题下的平均准确率

4.2.2　查询处理效率

同样地，查询所有主题时，记录各个算法执行时的距离对的比较次数以及算法执行时间。实验中设置 $m=100$，其中选取查询结果数目为 5，10，15，20，25，30，35，40，45，50 的情况进行展示，实验结果分别如图 4 和图 5 所示。

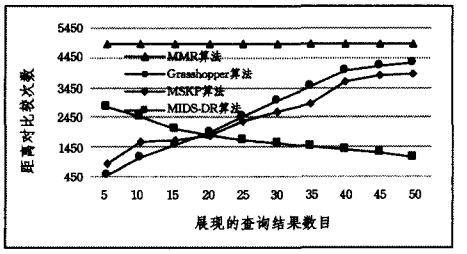

图 4　不同查询结果数目下 4 种算法的距离对的比较次数

当 $m=100$ 时，传统的 MMR 算法的距离对比较次数为 $\frac{m(m-1)}{2}=4950$，因此 MMR 算法的距离对比较次数与展现

的查询结果数目无关，每选取一个对象时都需要比较该对象与剩余对象之间的距离；而 Grasshopper 算法选取的开始项基于随机游走的特性，其距离对比较次数较少，查询处理的效率较高，但是随着展现的查询结果数目增多，其距离对比较次数不断增加；MSKP 算法基于图模型的置信传播算法来选取多样化子集，其距离对比较次数小于 Grasshopper 算法。

相比之下，MIDS-DR 算法虽然在 $m<20$ 时的距离对比较次数多于 Grasshopper 算法和 MSKP 算法，但在展现数目不断增加的情况下，其距离对比较的次数却在不断减少，这说明了引入抛弃子集概念的必要性。本方法减少了许多冗余操作，很好地控制了算法的计算量，比较适用于处理较大的数据量。

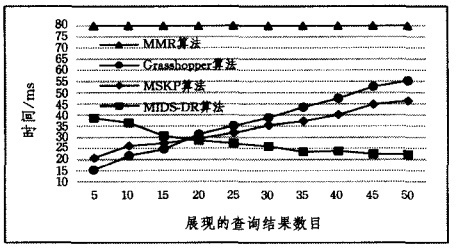

图 5 不同查询结果数目下 4 种算法花费的时间

从图 5 中各种算法花费的时间比较来看，MMR 算法与展现的查询结果数目无关，而 Grasshopper 算法和 MSKP 算法花费的时间随着展现的查询结果数目的增加而增加，MIDS-DR 算法花费的时间随着展现的查询结果数目的增加而减少，这说明 MIDS-DR 算法的查询处理效率高于其他算法，抛弃子集概念的引入使得查询处理效率不断提高。

结束语 针对目前用户实际的查询需求以及查询系统的实时应用要求，本文从查询结果的多样化和查询处理效率两个方面来设计多样化排序算法 MIDS-DR：基于图的极小独立支配集来选取多样化子集，保证相关性和多样化的平衡；改进初始化选取策略并引入抛弃子集来保证查询处理效率的提高，一定程度上减少了冗余距离对之间的比较次数。仿真结果表明，该算法能兼顾多样化性能和查询处理的效率，从而能快速展现出多样化的排序列表，适应实时快速的应用场景需求。

为了提高查询处理效率，MIDS-DR 算法利用范围查询来减少冗余查询对之间距离的比较，该范围查询所需的近邻集会被缓存下来，并占用较多的内存空间。同时在实际应用中，用户可能不止进行一次查询，多次连续查询之间可能存在联系，如何根据其中的联系来设计合适的多样化排序算法也是一个具有挑战性的问题。因此，下一步工作将着重研究范围查询的缓存占用问题以及连续查询的多样化排序问题。

参 考 文 献

[1] SANTOS R L T, PENG J, MACDONALD C, et al. Explicit Search Result Diversification through Sub-queries[C]// Proceedings of the 32nd European Conference on Advances in Information Retrieval. ACM, 2010: 87-99.

[2] OZDEMIRAY A M, ALTINGOVDE I S. Explicit search result diversification using score and rank aggregation methods[J]. Journal of the Association for Information Science & Technology, 2015, 66(2): 1212-1228.

[3] REN P J, CHEN Z M, MA J, et al. Search result diversification combing semantic and temporal intent[J]. Chinese Journal of Computers, 2015, 38(10): 2076-2091. (in Chinese)
任鹏杰，陈竹敏，马军，等. 一种综合语义和时效性意图的检索结果多样化方法[J]. 计算机学报，2015，38(10)：2076-2091.

[4] WU S L, HUANG C L. Search result diversification via data fusion[C]// Proceedings of the 37th International ACM SIGIR Conference on Research & Development in Information Retrieval. ACM, 2014: 827-830.

[5] YU H T, REN F. Search Result Diversification via Filling Up Multiple Knapsacks[C]// Proceedings of the 23rd ACM International Conference on Information and Knowledge Management. ACM, 2014: 609-618.

[6] HU S, DOU Z, WANG X, et al. Search Result Diversification Based on Hierarchical Intents[C]// Proceedings of the 24th ACM International on Conference on Information and Knowledge Management. ACM, 2015: 63-72.

[7] RADLINSKI F, KLEINBERG R, JOACHIMS T. Learning diverse rankings with multi-armed bandits[C]// Proceedings of the 25th International Conference on Machine Learning. ACM, 2008: 784-791.

[8] SCHUTH A, OOSTERHUIUS H, WHITESON S, et al. Multileave Gradient Descent for Fast Online Learning to Rank[C]// Proceedings of the Ninth ACM International Conference on Web Search and Data Mining. ACM, 2016: 457-466.

[9] OOSTERHUIUS H, SCHUTH A, DE RIJKE M. Probabilistic Multileave Gradient Descent[C]// Proceedings of the 38th European Conference on IR Research. 2016: 661-668.

[10] CARBONELL J, GOLDSTEIN J. The use of MMR, diversity-based re-ranking for reordering documents and producing summaries[C]// Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. ACM, 1998: 335-336.

[11] XIA L, XU J, LAN Y, et al. Learning Maximal Marginal Relevance Model via Directly Optimizing Diversity Evaluation Measures[C]// Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval. 2015: 113-122.

[12] ZHU X J, GOLDBERG, et al. Improving Diversity in Ranking using Absorbing Random Walks[C]// Proceedings of the North American Chapter of the Association for Computational Linguistics Conference on Human Language Technology. 2007: 97-104.

[13] ZHU Y, LAN Y, GUO J, et al. Learning for search result diversification[C]// Proceedings of the 37th International ACM SIGIR Conference on Research & Development in Information Retrieval. ACM, 2014: 293-302.

[14] CARPINETO. AMBIENT Dataset[EB/OL]. http://credo.fub.it/ambient.

[15] ZHAI C X, COHEN W W, LAFFERTY J. Beyond independent relevance: methods and evaluation metrics for subtopic retrieval [C]// Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. ACM, 2003: 10-17.