计算机研究与发展
Journal of Computer Research and Development
DOI:10.7544/issn1000-1239.2017.20170187
54(8):1665-1681,2017

D^3MOPSO:一种基于用户偏好的元搜索排序聚合演化方法

汤小月[1] 余 伟[2] 李石君[2]
[1](武汉轻工大学数学与计算机学院 武汉 430023)
[2](武汉大学计算机学院 武汉 430079)
(sharontang@whu.edu.cn)

D^3MOPSO:An Evolutionary Method for Metasearch Rank Aggregation Based on User Preferences

Tang Xiaoyue[1], Yu Wei[2], and Li Shijun[2]
[1](*School of Mathematics and Computer Science*, *Wuhan Polytechnic University*, *Wuhan* 430023)
[2](*Computer School*, *Wuhan University*, *Wuhan* 430079)

Abstract Much work has been done to implement metasearch engines with different rank aggregation methods. However, those methods do not have the ability to deal with the exploding data from huge amount of Web sources as well as the multiplying requirements of metasearch users. In this paper, we take the view that the rank aggregation problem can be solved with a multi-objective optimizer if the quality requirements of a user are considered along with the queries, and we find that the user's preferences among those quality requirements can help reduce the solution space. Accordingly, we propose an evolutionary rank aggregation algorithm based on user preferences. We bring a new encoding scheme for MOPSO, leverage new definitions of position and velocity, modify initialization methods of the particle swarms, improve the turbulence operator, and adjust strategies of external archive updating and leader selection, aiming at building a discrete multi-objective optimizer based on decomposition and dominance (D^3MOPSO) to map out the best aggregated ranking quickly and accurately from a large-scale discrete solution space. We have the proposed algorithm along with several state-of-the-art rank aggregation methods tested on 4 datasets of different sizes: the LETOR MQ2008-agg dataset, a Web dataset, a synthetically simulated dataset and an extended Web dataset. The experiment results demonstrate that our method outperforms machine-learning-based algorithms and other multi-objective evolutionary algorithms by convergence, performance and efficiency especially when dealing with the large-scale metasearch rank aggregation tasks.

Key words rank aggregation; metasearch; user preference; multi-objective optimization; discrete particle swarm optimization

摘 要 随着网络数据的爆发式增长和用户需求的多元化发展,现有元搜索排序聚合方法在精度和性能上面临着巨大挑战.以满足用户的多重需求和个性化偏好为目标,提出了一种新的元搜索排序聚合算法.

收稿日期:2017-03-20;**修回日期**:2017-06-07
基金项目:国家自然科学基金项目(61502350);湖北省教育厅科研计划项目(Q-20161702)
This work was supported by the National Natural Science Foundation of China (61502350) and the Research Program of Hubei Provincial Department of Education (Q-20161702).
通信作者:余伟(yuwei@whu.edu.cn)

通过重新定义多目标粒子群优化算法(multi-objective particle swarm optimization, MOPSO)中粒子的属性,调整速度变化因子,改进种群初始化和演化机制,设计新的存档与更新策略以及引导微粒选择策略,提出了一个基于支配分解的离散多目标优化(D^3MOPSO)算法,使其能根据用户的质量需求偏好在大规模离散解空间中快速准确地找出最优解集.在多个数据集上的实验结果表明:当数据规模较小时,D^3MOPSO 算法的精度和性能接近机器学习排序聚合方法;在大规模数据环境下,其精度和性能优于机器学习方法以及同类多目标优化方法.

关键词 排序聚合;元搜索;用户偏好;多目标优化;离散粒子群优化

中图法分类号 TP391

元搜索引擎[1-7]的用户对检索结果的质量要求通常包含着多重约束条件或意图.不同的用户对检索结果的各项质量指标也会有不同的偏好与侧重.为了更好地满足用户个性化的检索需求,元搜索引擎应当将用户对排序结果的质量偏好作为其排序聚合过程的重要参考.

现有的元搜索排序聚合算法大多是通过基于机器学习的方法来适应用户的个性化偏好的,训练一个合适的排序学习模型[8-17],使其能输出符合用户期望的排序聚合结果.然而,随着互联网数据源的爆发式增长,基于学习模型的排序聚合算法在聚合与重排大规模数据源的检索结果时,其排序效率和性能都受到了极大的挑战.针对这一问题,有一种解决思路是将元搜索的排序融合视为一个寻找排序最优解的问题,然后采用学习效率更高、优化速度更快的演化算法来求解.然而,现有的基于演化的排序聚合算法[18-28]大都实现的是较为简单的单目标优化,以单一的质量或效率度量值作为寻找最优解的依据,即使考虑了多个度量指标,也只是对其进行简单的综合,依然无法满足用户对排序结果的多元化、个性化的质量偏好.

为此,本文提出一种基于用户自定义的质量偏好的元搜索排序聚合方法,在多目标粒子群优化算法的基础框架上,通过重新定义其粒子状态、更新策略与映射机制,实现对海量数据项的聚合与多维度排序问题的演化求解.本文的主要贡献有 3 方面:

1) 提出了一种基于用户自定义的质量偏好的检索结果聚合模型,使聚合结果能够在多个维度上满足用户的偏好;

2) 提出了一种适用于大规模数据项排序过程的全局最优解发现方法,通过引入用户的质量偏好来提高解空间的搜索效率和改善优化效果;

3) 通过重新定义多目标粒子群算法中的映射机制,建立了粒子与实际数据项的关联,进而提出一种新的基于演化的排序聚合算法 D^3MOPSO,该算法具有较强的收敛性,能够快速地得到满足用户多重质量需求的排序聚合结果.

1 相关工作

从元搜索问世至今,已有大量的文献记录了对其进行的研究[1-7].对于其中决定检索质量的关键问题——排序聚合问题,目前的研究重点集中在基于排序学习模型的方法上.在无监督的排序学习方法中,文献[8]提出了一种基于位置得分的方法,通过检查数据来动态地向每个位置分配权重;文献[9]用基于 Markov 链的方法来实现排序聚合;文献[10]提出根据平均 Kendall-Tau 距离或 Kemeny 距离来判断排序聚合最优解;文献[11]提出了 Condorcet 熔丝格式搜索方法;文献[12]提出一种使用分数线性聚合的数据融合方法;文献[13]用 Mallows 模型聚合来自领域专家的类型排名.在有监督的排序聚合学习方法中,文献[14]提出了一种基于监督学习的排序聚合概率模型;文献[15]提出了基于查询相似性的监督排序聚合框架;文献[16]讨论了绝对和相对在线学习的概念,并比较了多种排名方法;文献[17]将从特征值生成的额外中间特征用于排序函数的两阶段学习.然而,随着互联网数据的爆发式增长,无论是监督还是非监督算法都很难满足海量数据源环境下的数据聚合与排序的性能和精度要求.于是,一类更易于实现、搜索速度更快的人工智能方法——演化方法——引起了学界的重视,如模拟退火算法[18]、遗传算法[19]、蚁群优化算法[20]、禁忌搜索算法[21]和粒子群优化(PSO)算法[22].对元搜索而言,其排序聚合问题是一种离散优化,而对离散粒子群优化(DPSO)算法的研究已有不少先例.第 1 个 DPSO 算法是 Kennedy 和 Eberhart 在文献[23]中提出的二进制粒子群优化算法(BPSO)算法;文献

[24]讨论将连续空间映射到离散空间的方法并提出了若干以空间变换技术为特征的离散粒子群优化算法;文献[25]通过分析粒子的飞行轨迹和引入广义中心粒子和狭义中心粒子,提出一种双中心粒子群优化;文献[26]提出一种基于集合的粒子群优化算法(SPSO),将候选解和速度分别定义为一个清晰集合和一个基于概率的模糊集合;文献[27]提出了一种模糊粒子群优化算法;文献[28]提出了一种基于加权和的离散粒子群优化算法;文献[29]把遗传算子引入粒子群优化算法中,提出了一种收敛精度较高的混合型智能优化算法.

尽管上述离散粒子群演化算法在许多排序应用中都获得了良好的效果,但由于用户需求多元化的发展特性,元搜索排序聚合问题不仅是一个离散演化问题,更是一个多目标粒子群优化问题,而由于缺乏充分的映射机制,目前也鲜有研究讨论如何运用多目标粒子群的演化算法来满足元搜索用户的多维度需求.

本文提出的基于多目标粒子群的元搜索排序聚合演化算法将粒子与待排序数据项进行一一映射,通过粒子的运动在多维解空间中寻找排序最优解集合,并结合用户对排序聚合结果质量的个性化偏好以缩小最优解的搜索范围.不仅满足了用户对检索结果多维度的质量需求,提高了解空间的搜索效率,还改善了粒子群演化算法在离散环境下的数据收敛性.同时,该方法也可以有效地弥补学习模型在大规模数据环境下排序聚合精度和性能上的退化.

2 基于用户偏好的元搜索排序聚合表示法

本节介绍与元搜索排序聚合问题相关的定义与描述.

2.1 基本定义

元搜索引擎的各类用户由于职业、年龄、知识领域、专业背景等条件的不同,其对检索结果的数据质量也有不同的偏好.例如对于同一检索词"奥巴马",有些用户对排序结果的时效性要求较为严格,而另一些用户则期望将更具权威性的结果排在前列.因此,本文将满足用户对数据的检索需求和个性化偏好视为数据采集和整合过程中需要同时达成的双重目标.用户的检索需求来源于用户输入的查询限定条件,而用户的个性化偏好(本文将重点研究用户对数据质量各项指标的偏向性)则取决于用户的历史检索行为.在详细描述本文算法与排序聚合模型之前,首先需要对与"基于用户偏好的排序聚合"问题相关的一些用语和符号进行定义和解释.

定义 1. 带质量标签的数据源.对于元搜索引擎的任意一个数据源(即独立搜索引擎),可用集合$DS=\{ID,UR,OD,\boldsymbol{DQ}\}$对其进行描述,其中,$ID$表示数据源名称,$UR$表示用户输入的检索词集合,$OD=\{\boldsymbol{d}_1,\boldsymbol{d}_2,\cdots,\boldsymbol{d}_N\}$为当前数据源中的全部待排序数据项(即文档,为保持术语的通用性本文统称为数据项)的集合,N为数据项总数,该集合中任意一个数据项可表示为$\boldsymbol{d}=(w_1,w_2,\cdots,w_L)$,$w_i$表示长度为$L$的词表中第$i$个词汇在数据项$\boldsymbol{d}$中对应的度量值.$\boldsymbol{DQ}$为数据源的质量标签向量,通常可表示为$\boldsymbol{DQ}=(v_1,v_2,\cdots,v_M)$,其中$v_i$为该标签中第$i$个质量指标的值,$M$表示考察的质量指标的个数.

根据元搜索的通用质量标准,本文选取准确性、可靠性、时效性、访问速度、领域相关性这5个指标对数据源质量进行考察.假设某数据源的质量标签为(0.2,0.3,0.1,0.6,0.1),这表示该数据源的准确性权值为0.2,可靠性权值为0.3,时效性权值为0.1,访问速度权值为0.6,而领域相关性权值为0.1.对于元搜索采用的所有数据源,可采用文献[30]中的评估方法对其质量标签进行计算.

定义 2. 用户偏好.元搜索用户对搜索结果各项质量指标的偏向性可以用向量$\boldsymbol{UP}=(c_1,c_2,\cdots,c_M)$表示.其中,$M$为考察的质量指标的个数,$c_i$表示用户对第$i$个质量指标的偏好程度.例如,用户定义的偏好向量(0.2,0.1,0.3,0.4,0)表示该用户对搜索结果的访问速度最为重视,其余依次重视时效性、准确性和可靠性,而对于领域相关性指标则未作特别要求.此外,本文定义与$\boldsymbol{UP}$唯一对应的序列$\boldsymbol{R}=(r_1,r_2,\cdots,r_M)$,使得$\forall 1\leqslant r_i,r_j\leqslant M,1\leqslant i<j\leqslant M$,有$c_{r_i}>c_{r_j}$.由定义可知,序列$\boldsymbol{R}$对应用户对排序结果各质量指标的偏好顺序.例如,上述用户质量偏好向量(0.2,0.1,0.3,0.4,0)对应的偏好顺序为$\boldsymbol{R}=(3,4,2,1,5)$.

定义 3. Y项-支配(PY-占优).假设存在函数$\boldsymbol{F}$,可对任意数据源中的任意数据项$\boldsymbol{d}$计算其质量标签向量,即$\boldsymbol{F}(\boldsymbol{d})=(f_1(\boldsymbol{d}),f_2(\boldsymbol{d}),\cdots,f_M(\boldsymbol{d}))$,其中$M$为质量指标数.当2个数据项$\boldsymbol{d}_1$和$\boldsymbol{d}_2$的质量标签向量$\boldsymbol{F}(\boldsymbol{d}_1)$和$\boldsymbol{F}(\boldsymbol{d}_2)$的各分量满足以下2个条件时,可称数据项$\boldsymbol{d}_1$"$Y$项-支配(PY-占优)"数据项$\boldsymbol{d}_2$,记为$\boldsymbol{d}_1\succ_{\mathrm{PY}}\boldsymbol{d}_2$:

1) $\forall i\in\{1,2,\cdots,M\}:f_i(\boldsymbol{d}_1)\geqslant f_i(\boldsymbol{d}_2)$;

2) $\exists Set=\{s_1,s_2,\cdots,s_Y\},s_j\in\{1,2,\cdots,M\}:f_{s_j}(\boldsymbol{d}_1)>f_{s_j}(\boldsymbol{d}_2)$.

其中,Y 为集合 Set 的大小. 当 $Y=1$ 时,Y 项-支配即为 Pareto 支配. 需要注意的是,当 $Y\leqslant\lfloor M/2\rfloor$ 时,Y 项-支配是一种非劣支配关系.

结合上述定义,可以正式给出本文所研究的基于用户偏好的元搜索排序聚合问题的描述.

2.2 基于用户偏好的元搜索排序聚合问题描述

第 2.1 节提到,为了在关键词匹配检索的前提下满足不同用户对检索结果质量的个性化偏好,可以将用户对数据的检索需求和个性化偏好视为元搜索数据采集和整合过程中需要同时达成的多重目标. 因此,基于用户偏好的元搜索排序问题可看作是一个根据用户对数据的偏好而指定目标权重的多目标优化问题. 假设用户输入的查询限定条件集合为 UR,从用户的历史检索行为中可以得知用户偏好向量为 $\boldsymbol{UP}$,那么将多个数据源 $DS_1, DS_2, \cdots, DS_n$ 中的海量数据项根据用户偏好 $\boldsymbol{UP}$ 进行元搜索排序聚合就是要从满足查询条件 UR 的数据项集合中找出以用户偏好 $\boldsymbol{UP}$ 为优化目标的全局 Y 项-支配(PY 占优)最优解集合 $S_{\mathrm{PY}}=\{\boldsymbol{d}^* \mid \neg\exists \boldsymbol{d}\in \Gamma^{UR}_{DS_1,DS_2,\cdots,DS_n}-S_{\mathrm{PY}}:\boldsymbol{d}\succ_{\mathrm{PY}}\boldsymbol{d}^*\}$ 以及该集合元素的 Top-K 排序结果 $\boldsymbol{L}_{\mathrm{PY}}=(\boldsymbol{d}_1,\boldsymbol{d}_2,\cdots,\boldsymbol{d}_K)$,使得在用户较为偏重的质量指标上占优的数据项排序靠前,而在用户不够偏重的质量指标上占优的数据项排序靠后,即 $\forall \boldsymbol{d}_i\in \boldsymbol{L}_{\mathrm{PY}}, 1\leqslant i\leqslant K$, $\forall 1\leqslant j\leqslant M$,有 $f_{r_j}(\boldsymbol{d}_i)>f_{r_j}(\boldsymbol{d}_{i+1})$ 或 $|f_{r_j}(\boldsymbol{d}_i)-f_{r_j}(\boldsymbol{d}_{i+1})|<\theta$ 且 $f_{r_{j+1}}(\boldsymbol{d}_i)>f_{r_{j+1}}(\boldsymbol{d}_{i+1})$,同时满足:

$$\max \boldsymbol{F}(\boldsymbol{d})=(f_1(\boldsymbol{d}),f_2(\boldsymbol{d}),\cdots,f_M(\boldsymbol{d})),\quad \boldsymbol{d}\in\Gamma^{UR}_{DS_1,DS_2,\cdots,DS_n}, \tag{1}$$

其中,$f_{r_j}(\boldsymbol{d})$ 表示数据项 $\boldsymbol{d}$ 的质量特征向量 $\boldsymbol{F}(\boldsymbol{d})$ 的第 r_j 个分量,θ 为一个足够小的数.

上述演化过程旨在从 n 个数据源中对应的符合查询条件 UR 的数据项集 $\Gamma^{UR}_{DS_1,DS_2,\cdots,DS_n}$ 中找到在 M 个目标上的 Y 项-支配最优解,容易知道,在一般情况下,上述优化问题求解的时间复杂度为 $O(n^M)$.

由于互联网上的数据源数目庞大,若先按查询条件对多个数据源的海量数据项进行筛选,再从符合条件的数据项中挑选符合用户多元化、个性化的质量偏好的集合进行排序,其运算规模是巨大的,现有的排序聚合技术在效率上难以应对此类高维度、大数据量的问题. 为此,本文提出一种基于支配分解的多目标离散粒子群优化方法(discrete multi-objective particle swarm optimizer based on decomposition and dominance, D^{3}MOPSO)来解决这一问题.

3 D^{3}MOPSO:一种改进的离散多目标粒子群算法

本节详细介绍如何改进多目标粒子群优化算法,以适应求解元搜索排序聚合问题的需要,将从数据项与粒子的映射方法、粒子群初始化方法、粒子的位置与速度定义、粒子状态更新规则、迭代存档策略等方面深入介绍本文提出的 D^{3}MOPSO 算法.

3.1 多方向的目标搜索策略

传统的离散多目标粒子群优化方法需要将所有的离散点在某个维度上以线性方式按一定方向排序,以保证搜索在前后 2 个方向上的随机游走. 然而对于规模极大的互联网数据来说,由于离散点数量骤增,这种搜索方法效率不佳. 本文根据数据源和元搜索结果的特性,对数据源集合中符合用户输入条件 UR 的全部数据项取其质量标签向量 $\boldsymbol{DQ}$ 某一特定分量的值,按其大小将数据源排列成网格. 由此可以定义排序优化搜索网格的结构如图 1 所示:

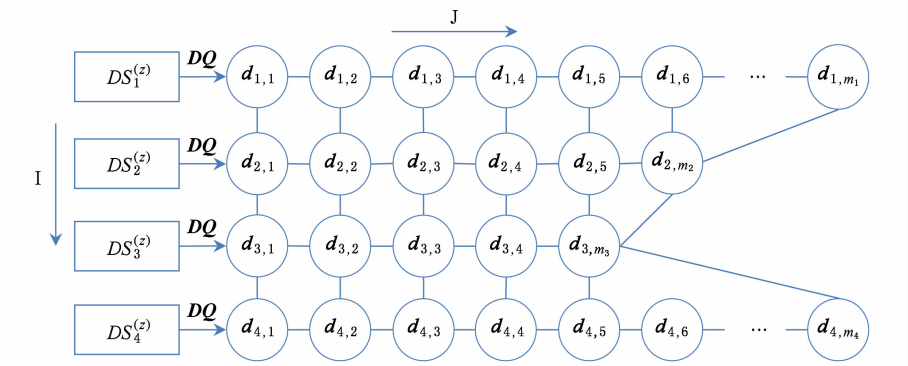


Fig. 1 Searching grid of particles on objective dimension z

图 1 质量目标维度 z 上的最优解集搜索网格示意图

以第 z 个质量维度为例,n 个数据源根据其先验数据源质量标签向量 $\boldsymbol{DQ}$ 的分量 v_z 的值降序排列,形成序列($DS_1^{(z)}$, $DS_2^{(z)}$, …, $DS_n^{(z)}$). 数据源 $DS_i^{(z)}$ 中满足用户输入条件 UR 的数据项集合为 $\Gamma_{DS_i^{(z)}}^{UR}=\{\boldsymbol{d}_{i,1},\boldsymbol{d}_{i,2},\cdots,\boldsymbol{d}_{i,m_i}\}$,集合的大小为 m_i. 每个维度的粒子可以在如图 1 所示的 I,J 两个方向上前后移动,为了表示粒子的移动方式,可以为搜索网格中每个数据项 $\boldsymbol{d}$ 定义 4 个虚拟的方向指针 $\boldsymbol{d}\rightarrow$Inext,$\boldsymbol{d}\rightarrow$Iprior,$\boldsymbol{d}\rightarrow$Jnext,$\boldsymbol{d}\rightarrow$Jprior(实际中可以对搜索网格采用连续的二维矩阵进行存储,通过内存的移动来进行寻址,以提高搜索效率):

$$\boldsymbol{d}_{i,j}\rightarrow\text{Inext}=\begin{cases}\boldsymbol{d}_{(i+1),j}, & i<n, m_{i+1}\geqslant j;\\ \boldsymbol{d}_{(i+1),m_{i+1}}, & i<n, m_{i+1}<j;\\ \text{Null}, & i\geqslant n.\end{cases}\quad(2)$$

$$\boldsymbol{d}_{i,j}\rightarrow\text{Jnext}=\begin{cases}\boldsymbol{d}_{i,(j+1)}, & j<m_i;\\ \text{Null}, & j\geqslant m_i.\end{cases}\quad(3)$$

$$\boldsymbol{d}_{i,j}\rightarrow\text{Iprior}=\begin{cases}\text{Null}, & i\leqslant 1;\\ \boldsymbol{d}_{(i-1),j}, & i>1, m_{i-1}\geqslant j;\\ \boldsymbol{d}_{(i-1),m_{i-1}}, & i>1, m_{i-1}<j.\end{cases}\quad(4)$$

$$\boldsymbol{d}_{i,j}\rightarrow\text{Jprior}=\begin{cases}\text{Null}, & j\leqslant 1;\\ \boldsymbol{d}_{i,(j-1)}, & j>1.\end{cases}\quad(5)$$

由于 I,J 两个方向的有序,这种分布方式能使每个粒子第 z 个质量维度上的目标值都在一个小领域内保持相对平滑,有利于粒子在局部搜索中找到的极值. 更重要的是,粒子在该搜索网格中能够多方向游走,从而提高搜索效率. 算法 1 描述了针对 M 个目标分别生成对应的搜索网格的过程.

算法 1. 最优解集搜索网格生成算法.

输入:n 个数据源 DS_1,DS_2,…,DS_n、用户查询词集合 UR、各数据源质量标签向量 $\boldsymbol{DQ}$ 形如(v_1,v_2,…,v_M);

输出:对应于 M 个质量偏好排序目标的搜索网格(矩阵).

① generate $\Gamma_{DS_1,DS_2,\cdots,DS_n}^{UR}$ from DS_1, DS_2, …, DS_n;

② $Maxitem=\max(m_i)$; /* m_i 为集合 $\Gamma_{DS_i}^{UR}$ 的大小 */

③ $ParticleGraph(z)=$new $matrix(Maxitem, n)$; /* $z=1,2,\cdots,M$ */

④ $SortIndex(z)=$new $matrix(1,n)$; /* 存储按数据源的特征 z 降序排列后的数据源索引值 */

⑤ for $z=1:M$

⑥ $SortIndex(z)=SORT(DS_1, DS_2, \cdots, DS_n)$ by $\boldsymbol{DQ}.v_z$;

⑦ $ParticleGraph(z)=(DS_{SortIndex(z)(1)}, DS_{SortIndex(z)(2)}, \cdots)$. /* 重新排序各数据源的结果 */

⑧ end for

3.2 种群的初始化策略

传统的排序聚合算法更多地考虑高质量的数据源和数据项的排名,因而很容易丢失更多的候选数据. 而一般的粒子群算法采用随机初始化策略,也容易造成收敛困难. 本文提出一种加权随机初始化方法,原理是结合数据源的质量标签和数据项位置的分布规律随机进行概率分配.

对于搜索网格中的任意一个数据项 $\boldsymbol{d}_{i,j}$,定义其初始概率为:$P(\boldsymbol{d}_{i,j},z)=P(DS_i|z)P(Rank=j)$,其中,$P(DS_i|z)=DS_i.\boldsymbol{DQ}.v_z\Big/\sum_{k=1}^{n}DS_k.\boldsymbol{DQ}.v_z$. 同时规定数据项 $\boldsymbol{d}_{i,j}$ 在其对应数据源 DS_i 中排第 j 位的概率 $P(Rank=j)$ 服从齐夫分布,即对任意排序位置 j,$1\leqslant j\leqslant m_i$ 有:$\sum_{j=1}^{m_i}h\times j^{-\beta}=1$,$h\times m_i=a$,其中 a 为当前数据源中排在最后一位的数据项的初始概率,其值在实验中可通过训练调优得到. 参数 h 和 β 则可通过逼近法进行数值求解. 算法 2 描述了生成目标维度 z 上的一个初始化粒子,在生成粒子群时,则根据质量目标的个数多次调用算法 2.

算法 2. 粒子群初始化算法.

输入:z /* 生成初始粒子群的质量目标维度编号 */;

输出:搜索第 z 个质量目标网格矩阵的粒子群.

① $rand_1=random()$; /* 生成[0,1)上的随机数 */

② $SortIndex=SORT(DS_1, DS_2, \cdots, DS_n)$ by $\boldsymbol{DQ}.v_z$; /* 向量用来存储按数据源的特征 z 降序排列后的数据源索引值 */

③ $TotalScore=\sum_{i=1}^{n}DS_i.\boldsymbol{DQ}.v_z$; /* 所有数据源在特征 z 上先验评分的累加值 */

④ $s_i=0$; /* 存储随机选择的数据源编号(排序后的编号) */

⑤ for $i=1:n$

if $rand_1<DS_{SortIndex(i)}.\boldsymbol{DQ}.v_z/TotalScore$

$s_i=i$;break;

else $rand_1-=DS_{SortIndex(i)}.\boldsymbol{DQ}.v_z/TotalScore$;

```
    end if
  end for
⑥ t_i=0; /* 存储随机选择的数据编号 */
⑦ DataNumber=m_{s_i}; /* 所选数据源的查询结果数 */
⑧ rand_2=random(); /* 生成[0,1)上的随机数 */
⑨ for i=1:DataNumber
    if rand_2<hi^{-beta}
      t_i=i;break;
    else rand_2-=hi^{-beta};
    end if
  end for
⑩ return d(s_i,t_i). /* 选中第 s_i 个数据源的查询结果中的第 t_i 个数据作为粒子 */
```

3.3 粒子的位置与速度变化规律

为了提高从大规模数据项集合中选取多目标最优解的效率,本文重新定义了传统粒子群优化算法中的粒子位置与速度.

1) 粒子位置. 在本文提出的排序搜索优化算法中,一个数据项 $\boldsymbol{d}_{i,j}$ 所在的数据源的序号及其在该数据源中的排序值(i,j)对应了搜索网格中的一个粒子位置,即一个粒子位置可以用搜索网格中的一个节点 $\boldsymbol{d}_{i,j}$ 来表示. 因此,一个由 s 个粒子所组成的粒子群 $P=\{\boldsymbol{p}_1,\boldsymbol{p}_2,\cdots,\boldsymbol{p}_s\}$ 可以用其粒子的位置集合表示,由于粒子位置与数据项网格中数据项的一一对应关系,粒子群 P 可用如下数据项集合来进行描述:$\{\boldsymbol{p}_1,\boldsymbol{p}_2,\cdots,\boldsymbol{p}_s\}=\{\boldsymbol{d}_{i_1,j_1},\boldsymbol{d}_{i_2,j_2},\cdots,\boldsymbol{d}_{i_s,j_s}\}$ 其中 $\forall 1\leqslant k\leqslant s,\boldsymbol{d}_{i_k,j_k}\in\Gamma_{DS_{i_k}^{(z)}}^{UR}$ 且 $1\leqslant j_k\leqslant size(\Gamma_{DS_{i_k}^{(z)}}^{UR})$.

2) 粒子速度. 粒子的运动速度规定了粒子的运动方向和位移,是决定粒子能否快速有效地收敛于最佳目的地的关键因素. 用速度向量 $\boldsymbol{V}_x=(v_{x\mathrm{I}},v_{x\mathrm{J}})$ 来定义粒子群中任一粒子 $\boldsymbol{p}_x$,$1\leqslant x\leqslant s$ 的运动速度,其中 $v_{x\mathrm{I}}$,$v_{x\mathrm{J}}$ 分别是粒子在搜索网格中 I 和 J 方向上的速度分量,其值都是整数. 如果 $v_{x\mathrm{I}}$ 为正值,则按 $\boldsymbol{d}_{i,j}\rightarrow$Inext 的方向移动 $v_{x\mathrm{I}}$ 次,如果为负值,则按 $\boldsymbol{d}_{i,j}\rightarrow$Iprior 的方向移动 $v_{x\mathrm{I}}$ 次;如果 $v_{x\mathrm{J}}$ 为正值,则按 $\boldsymbol{d}_{i,j}\rightarrow$Jnext 的方向移动 $v_{x\mathrm{J}}$ 次,如果为负值,则按 $\boldsymbol{d}_{i,j}\rightarrow$Jprior 的方向移动 $v_{x\mathrm{J}}$ 次.

3.4 离散粒子的状态更新过程

粒子群优化算法通过更新粒子的速度来改变粒子的移动距离. 由于本文算法中粒子的位置和速度均为整数向量,因此本文也相应地重新定义了离散化的粒子速度变化规则,以适应离散化的元搜索排序聚合问题的需要. 假设粒子 $\boldsymbol{p}_x$ 的速度 $\boldsymbol{V}_x$ 在 I 和 J 方向上的分量分别为 $v_{x\mathrm{I}}$ 和 $v_{x\mathrm{J}}$,则每次粒子发生移动后速度就更新为式(6)所示形式.

$$v'_{x\mathrm{I}}=\left\lfloor\omega_{\mathrm{I}}v_{x\mathrm{I}}+c_{\mathrm{I}1}r_{\mathrm{I}1}DI(\boldsymbol{Pb}_x-\boldsymbol{p}_x)+c_{\mathrm{I}2}r_{\mathrm{I}2}DI(\boldsymbol{Ob}_x-\boldsymbol{p}_x)+c_{\mathrm{I}3}r_{\mathrm{I}3}DI(\boldsymbol{Gb}_x-\boldsymbol{p}_x)\right\rfloor,$$

$$v'_{x\mathrm{J}}=\left\lfloor\omega_{\mathrm{J}}v_{x\mathrm{J}}+c_{\mathrm{J}1}r_{\mathrm{J}1}DJ(\boldsymbol{Pb}_x-\boldsymbol{p}_x)+c_{\mathrm{J}2}r_{\mathrm{J}2}DJ(\boldsymbol{Ob}_x-\boldsymbol{p}_x)+c_{\mathrm{J}3}r_{\mathrm{J}3}DJ(\boldsymbol{Gb}_x-\boldsymbol{p}_x)\right\rfloor. \quad (6)$$

在式(6)中,$\boldsymbol{Pb}_x$ 是粒子 $\boldsymbol{p}_x$ 的个体历史最优位置,$\boldsymbol{Ob}_x$ 是粒子群在当前质量维度上的局部最优位置,$\boldsymbol{Gb}_x$ 则是粒子群在全部质量维度上的全局历史最优位置(描述参见 3.6 节);ω_{I} 和 ω_{J} 为$[0,1]$区间内的随机数,分别表示粒子在 I 和 J 方向上的惯性系数;$c_{\mathrm{I}1}$,$c_{\mathrm{I}2}$ 和 $c_{\mathrm{I}3}$ 为 I 方向上的关系系数,表示粒子的自身历史轨迹与群体运动规律对其速度的影响,类似的还有 J 方向上的关系系数 $c_{\mathrm{J}1}$,$c_{\mathrm{J}2}$ 和 $c_{\mathrm{J}3}$,其值可通过训练调优获得;系数 $r_{\mathrm{I}1}$,$r_{\mathrm{I}2}$,$r_{\mathrm{I}3}$ 以及 $r_{\mathrm{J}1}$,$r_{\mathrm{J}2}$,$r_{\mathrm{J}3}$ 是 2 组$[0,1]$区间内的随机数. DI 和 DJ 分别为方向 I 和方向 J 上的距离函数,其计算方法如例所示:设粒子 $\boldsymbol{p}_x$ 的当前位置对应数据项 $\boldsymbol{d}_{15,32}$,其 $\boldsymbol{Pb}_x$ 对应数据项 $\boldsymbol{d}_{23,17}$,则可以计算得到函数值 $DI(\boldsymbol{Pb}_x-\boldsymbol{p}_x)=8$,$DJ(\boldsymbol{Pb}_x-\boldsymbol{p}_x)=-15$.

根据上述粒子速度的变化规则,将粒子位置更新规则重新定义的离散形式为

$$\boldsymbol{p}'_x=\boldsymbol{p}_x\odot\boldsymbol{V}'_x=\boldsymbol{p}_x\odot(v'_{x\mathrm{I}},v'_{x\mathrm{J}}), \quad (7)$$

式(7)中的运算符$\odot$表示对粒子位置状态的更新运算,其值域范围决定了粒子移动的方向和位移大小,因此该运算也是决定算法性能的关键. 一个成功的位置更新策略必须保证粒子的每次移动都能更加靠近目的地. 据此本文对粒子位置的更新运算$\odot$进行重新定义. 假设粒子 $\boldsymbol{p}_x$ 更新前的起始位置对应于数据项 $\boldsymbol{d}_{k,l}$,经过更新操作后粒子将移动到数据项 $\boldsymbol{d}_{k',l'}$ 对应的目标位置. 则运算符$\odot$对应下列计算:

$$k'=\begin{cases}k+v'_{x\mathrm{I}}, & 1\leqslant k+v'_{x\mathrm{I}}<n;\\ n+v'_{x\mathrm{I}}-k, & k+v'_{x\mathrm{I}}\geqslant n;\\ 2-v'_{x\mathrm{I}}-k, & k+v'_{x\mathrm{I}}<1.\end{cases} \quad (8)$$

$$l'=\begin{cases}l+v'_{x\mathrm{J}}, & 1\leqslant l+v'_{x\mathrm{J}}<m_{k'};\\ m_{k'}+v'_{x\mathrm{J}}-l, & l+v'_{x\mathrm{J}}\geqslant m_{k'};\\ 2-v'_{x\mathrm{J}}-l, & l+v'_{x\mathrm{J}}<1.\end{cases} \quad (9)$$

其中,n 为元搜索的数据源个数,$m_{k'}$ 是从数据源 $DS_{k'}^{(z)}$ 中获得的符合查询条件集合 UR 的数据项数目,即 $size(\Gamma_{DS_{k'}^{(z)}}^{UR})$. 如果搜索网格中不存在计算得到的目标位置 $\boldsymbol{d}_{k',l'}$,则随机寻找该坐标邻域内最近

的一个落点.如果 $\boldsymbol{d}_{k',l'}$ 是粒子已经搜索过的位置,则在整个搜索网格中随机选择一个落点作为粒子的目标位置,以避免陷入局部搜索.如图 2 所示,如果粒子从起始点 $\boldsymbol{d}_{1,4}$ 出发,且有 $v_{x\mathrm{I}}=1, v_{x\mathrm{J}}=-2$,则粒子移动到离 $\boldsymbol{d}_{2,2}$ 最近的领域点的落点处.

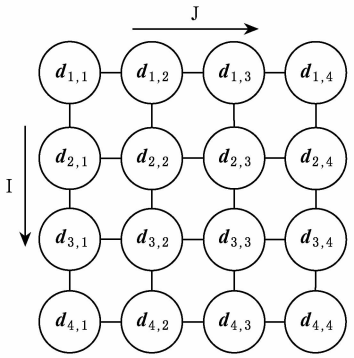


Fig. 2 Searching directions of a particle

图 2 搜索过程中的粒子运动方向示意图

3.5 保留多样性与用户偏好的存档策略

现有的粒子群优化算法通常借助外部存档(external archive, EA)来存储搜索过程中产生的 Pareto 解[31],其基本思路是通过对算法当前迭代过程中产生的最优解与 EA 中的元素进行 Pareto 支配的比较来获得 Pareto 最优解,即:当 EA 中的元素数目尚未达到最大容量 $MAXP$ 时,直接将新生的非被支配解插入;而当 EA 已满时,随机决定是否用新生的非被支配解来优化 EA 中的某个随机元素.但随着优化目标增多,在迭代优化的过程中会形成大量的非被支配解(相互不满足 Pareto 支配关系),使得 EA 容易被迅速填满,导致 EA 更新频繁.在连续的优化问题中,这个问题可以通过使最优解匹配优化目标的平均期望来避免.解决方法包括基于 KL 散度(Kullback-Leibler divergence)的均衡性度量[32]和基于自适应网格法的最优点选择[33]等方法.

在离散多目标最优化问题中,多样性具有不同的涵义和意义.在大规模元搜索排序聚合问题中,保留解的多样性主要是指减少最优解集中的相似解.有别于连续最优化问题中的 Pareto 最优前沿,离散最优化问题中的距离不一定能反映相似程度.因此,在本文提出的 D³MOPSO 算法中,档案保存策略考虑了多样性和用户偏好等特征,避免了最优解集过大的情况.本文主要从 3 方面对档案保存策略进行改进:

1) 用定义 3 中的 Y 项-支配(PY 占优)代替 Pareto 支配机制,以保证存档结果的领先性.Y 的取值在实验中将会进行训练调优.

2) 定义粒子相似度时综合考虑粒子的文本特征、质量特征和位置特征 3 方面的相似度,保留粒子的多样性,以避免存档的粒子过于集中.每个档案(粒子)$\boldsymbol{p}_x$ 实际上对应搜索网格中的一个数据项 $\boldsymbol{d}$,由于数据项 $\boldsymbol{d}$ 同时具有文本特征、质量特征和位置特征,因此,粒子 $\boldsymbol{p}_x$ 也同样具有这 3 类特征:

① $\boldsymbol{W}(\boldsymbol{p}_x)=(w_1, w_2, \cdots, w_L)$ 表示 $\boldsymbol{p}_x$ 的文本特征;

② $\boldsymbol{F}(\boldsymbol{p}_x)=(f_1(\boldsymbol{d}), f_2(\boldsymbol{d}), \cdots, f_M(\boldsymbol{d}))$ 表示 $\boldsymbol{p}_x$ 在质量标签多个维度上的匹配度,是粒子的质量特征;

③ $\boldsymbol{P}(\boldsymbol{p}_x)=(k_x, l_x)$ 表示 $\boldsymbol{p}_x$ 对应搜索网格中的数据项 $\boldsymbol{d}_{k_x,l_x}$,是粒子的位置特征.

综合上述 3 类特征,2 个粒子 $\boldsymbol{p}_A$ 和 $\boldsymbol{p}_B$ 的相似度计算为

$$S(\boldsymbol{p}_A,\boldsymbol{p}_B)=S_1(\boldsymbol{W}(\boldsymbol{p}_A),\boldsymbol{W}(\boldsymbol{p}_B))+ S_2(\boldsymbol{F}(\boldsymbol{p}_A),\boldsymbol{F}(\boldsymbol{p}_B))+S_3(\boldsymbol{P}(\boldsymbol{p}_A),\boldsymbol{P}(\boldsymbol{p}_B)), \quad (10)$$

其中,S_1 为 2 个粒子的文本特征相似度计算函数,度量方法已有许多[34],本文采用的是向量余弦值.为了避免最优解都集中在适应某个单一目标的局部最优解附近,需计算粒子的质量特征相似度,其计算函数为

$$S_2(\boldsymbol{F}(\boldsymbol{p}_A),\boldsymbol{F}(\boldsymbol{p}_B))= \sum_{i=1}^{M}\left(1-\frac{|f_i(\boldsymbol{p}_A)-f_i(\boldsymbol{p}_B)|}{|f_i(\boldsymbol{p}_A)|+|f_i(\boldsymbol{p}_B)|}\right);$$

位置特征相似度计算函数 $S_3(\boldsymbol{P}(\boldsymbol{p}_A),\boldsymbol{P}(\boldsymbol{p}_B))=\mathrm{e}^{-|k_A-k_B|}$ 计算两粒子的位置相似度.

3) 双窗口滑动的有序比较.由于元搜索具有集成查询结果的特性,过多的全局最优解对于用户没有意义,因此保证存档中的粒子的领先性非常重要,不仅要使档案中粒子保证绝对的 Y 项-支配,而且要保持领先和有序,以满足排序的效率要求和用户的质量偏好.假设档案中的粒子以 $(\boldsymbol{p}_1, \boldsymbol{p}_2, \cdots, \boldsymbol{p}_{MAXP})$ 的形式排列,并按照匹配用户查询词 UR 以及保持用户偏好 $\boldsymbol{UP}$ 多个目标优先级 $\boldsymbol{R}=(r_1, r_2, \cdots, r_M)$ 的顺序排序,即对于档案中任意的前后 2 个粒子 $\boldsymbol{p}_x$ 和 $\boldsymbol{p}_{x+1}$,一定有 $f_{r_i}(\boldsymbol{p}_x)>f_{r_i}(\boldsymbol{p}_{x+1})$ 或 $|f_{r_i}(\boldsymbol{p}_x)-f_{r_i}(\boldsymbol{p}_{x+1})|<\theta$ 且 $f_{r_{i+1}}(\boldsymbol{p}_x)>f_{r_{i+1}}(\boldsymbol{p}_{x+1})$,其中 $1\leqslant i, r_i\leqslant M$.为此,本文设置 2 个指向档案中粒子的指针 $\boldsymbol{q}_1$ 和 $\boldsymbol{q}_2$,如图 3 所示.当新一轮迭代产生的局部最优粒子 $\boldsymbol{p}'$需要加入时,首先依次遍历档案中的所有粒子,将每个粒子 $\boldsymbol{p}_x$ 都与 $\boldsymbol{p}'$进行 Y 项-支配比较,若 $\boldsymbol{p}'\succ_{\mathrm{PY}}\boldsymbol{p}_x$ 则 $\boldsymbol{p}_x\leftarrow\boldsymbol{p}'$后退出;若 $\boldsymbol{p}_x\succ_{\mathrm{PY}}\boldsymbol{p}'$则直接退出.同时,比较粒子 $\boldsymbol{p}_x$ 和 $\boldsymbol{p}'$的质量函数,若满足 $f_{r_i}(\boldsymbol{p}_x)>f_{r_i}(\boldsymbol{p}')$ 或 $|f_{r_i}(\boldsymbol{p}_x)-f_{r_i}(\boldsymbol{p}')|<\theta$ 且 $f_{r_{i+1}}(\boldsymbol{p}_x)>f_{r_{i+1}}(\boldsymbol{p}')$,则移动指针 $\boldsymbol{q}_1$ 到粒子 $\boldsymbol{p}_x$ 处,

否则停止移动. 然后比较 $\boldsymbol{p}'$ 和 $\boldsymbol{p}_x$ 的相似度,记录其相似度值,并保持指针 $\boldsymbol{q}_2$ 指向与 $\boldsymbol{p}'$ 相似度最大的粒子. 最后遍历完档案后,若没有退出,则根据此时的最大相似度值,判断 $\boldsymbol{p}'$ 是否与 $\boldsymbol{q}_2$ 指向的粒子进行二选一保留策略. 如果相似度差值未超过阈值 λ,则将 $\boldsymbol{p}'$ 插入在 $\boldsymbol{q}_1$ 后. 如果档案容量超过 $MAXP$,则随机选择一个粒子退出档案.

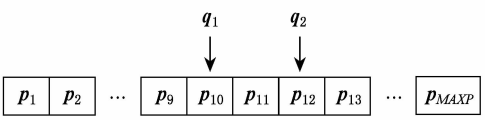


Fig. 3 Two pointers traversing through EA

图 3 EA 更新过程中的双指针遍历

3.6 引导微粒的选取

选取合适的引导微粒可以有效提高粒子的多样性和算法的收敛性. 在传统的粒子群优化算法中,存档集合 EA 中的解两两不相互支配,因此必须从中选择一个档案成员作为全局最优目标位置 $\boldsymbol{Gb}$. 通常的选择策略有随机选择法和就近选择法等[35-36].

在本文提出的 D^{3}MOPSO 算法中,根据 3.5 节得到的存档 EA 满足相对有序性,即粒子在 EA 中的顺序必然在某些目标质量维度上满足一定的优劣支配关系. 在这一存档策略的基础上,本文提出一种带生命周期的全局最优位置淘汰机制,结合加权初始生存期赋值,进而有效地选出档案成员得到引导微粒.

本文为存档中的非劣个体设置生存期属性,生存期是档案成员充当全局最优位置角色并引导粒子寻找最优解的活动周期,其长度可以用历经的迭代次数来表示,即生存期值. 带生存期的全局最优位置的淘汰机制的基本思路是:如果粒子群中某粒子 $\boldsymbol{p}_x$ 随机选中档案中某个非支配个体 $\boldsymbol{E}_j$ 作为全局最优解,则赋予档案成员 $\boldsymbol{E}_j$ 一个初始的生存期值 $L_j = 10j^{-\alpha}L_0$,其中 j 表示 $\boldsymbol{E}_j$ 在 EA 中的位置,α 为该成员的位置系数,表示 $\boldsymbol{E}_j$ 的生存期长度与 EA 中当前存档的最优解总数 Num 有关,本文取 $\alpha = \ln Num$. L_0 为全局初始生存期,L_i 表示初始生存期会随着 EA 中位置逐渐递减,一旦当档案成员 $\boldsymbol{E}_j$ 的生存期值 L 减至预设的最小阈值 $L_{\min}$,甚至更小时,档案成员 $\boldsymbol{E}_j$ 的活动期结束,进而从档案中重新为粒子 $\boldsymbol{p}_x$ 随机选择一个非劣解作为它的全局最优位置并引导粒子 $\boldsymbol{p}_x$ 继续飞行. 此处生存期的值需要根据粒子之间的支配关系进行调整. 对于档案成员 $\boldsymbol{E}_j$,其生存期值按照 3 种情况进行调整:

1) 如果粒子 $\boldsymbol{p}_x$ 的目标位置 $\boldsymbol{p}'_x$($\boldsymbol{p}'_x$ 为由式(6)和(7)生成的目标位置粒子)与 $\boldsymbol{p}_x$ 的历史最优位置之间相互非支配,则当前作为全局最优解的 $\boldsymbol{E}_j$ 的生存权 L 减少 $\Delta L/2$;

2) 如果粒子 $\boldsymbol{p}_x$ 的目标位置 $\boldsymbol{p}'_x$ 被 $\boldsymbol{p}_x$ 的历史最优位置 Y 项-支配,则当前作为全局最优解的 $\boldsymbol{E}_j$ 的生存期 L 减少 ΔL;

3) 如果粒子 $\boldsymbol{p}_x$ 的目标位置 $\boldsymbol{p}'_x$ Y 项-支配 $\boldsymbol{p}_x$ 的历史最优位置,则当前作为全局最优解的 $\boldsymbol{E}_j$ 的生存期 L 增加 ΔL.

其中的 $\Delta L = (L_1 - L_{\min})/l$,$\Delta L$ 为生存期 L 的调整步长,L_0,$L_{\min}$ 和 l 为 ΔL 的计算参数.

4 D^{3}MOPSO 算法框架与复杂度分析

本文提出的 D^{3}MOPSO 算法沿用了传统粒子群优化算法的框架和交叉、变异的演化思想,同时在解空间搜索模式、种群初始化方法、粒子位置与速度变化规则、排序聚合规则以及外部存档更新策略、全局最优解位置选择策略等方面进行了较大改进,以适应元搜索排序聚合问题.

4.1 算法描述

综合各项步骤,可以得到本文提出的 D^{3}MOPSO 模型的整体框架.

算法 3. D^{3}MOPSO 算法框架.

输入:n 个数据源 $DS_1, DS_2, \cdots, DS_n$,用户检索词集合 UR,各数据源质量标签向量 $\boldsymbol{DQ}$ 形如 $(v_1, v_2, \cdots, v_M)$;

输出:有序的 Top-K 排序聚合最优解.

① 根据算法 1 针对每个目标维度对 $\Gamma^{UR}_{DS_1, DS_2, \cdots, DS_n}$ 中的所有数据项(待排序结果)进行重新编码,生成 M 个搜索网格矩阵;

② 根据算法 2 初始化 M 个大小为 $SwarmSize$ 的粒子群;

③ 根据 3.3 节定义粒子的结构体并初始化粒子群中个体的位置和速度;

④ 初始化粒子群中的每个个体最优位置矩阵 ***PbMatrix***、粒子群在每个质量目标维度上的局部最优位置矩阵 ***ObMatrix***;

⑤ 根据 3.6 节初始化计算全局最优位置矩阵 ***GbMatrix*** 的函数;

⑥ 开始迭代,根据 3.3~3.6 节,为粒子群中的每个个体生成新的位置值与速度值并更新 ***PbMatrix***, ***ObMatrix*** 和 ***GbMatrix*** 矩阵;

⑦ 如果迭代中产生的任何一个粒子可以 Y 项-支配全局最优位置 $\boldsymbol{Gb}$,则根据 3.5 节判断并将其加入到存档 EA 中;

⑧ 每次迭代后,根据 3.6 节更新全局最优位置 $\boldsymbol{Gb}$ 及其生存期,根据具体情况进行变化;

⑨ 返回步骤⑥,直至达到最大迭代次数,结束算法并输出存档 EA.

4.2 复杂度分析

本文提出的 D³MOPSO 运行时有 2 个过程需要占用存储空间:1)解搜索空间搜索网格的初始化,需要占用 $M\times n\times Maxitem\approx M\times N$ 大小的存储空间,空间复杂度为 $O(M\times N)$;2)粒子群搜索最优解的迭代过程,假设粒子群规模为 $SwarmSize$,则空间复杂度为 $3SwarmSize\times M$,即 $O(SwarmSize\times M)$. 因此,算法总的空间复杂度不超过 $O(M\times N)$.

从算法 3 可以看出,由于初始化过程可以在常数时间内完成,D³MOPSO 算法的最大时间耗费发生在迭代搜索最优解的过程中. 考虑到在每次迭代中,粒子群更新粒子位置与速度的时间开销为 $O(SwarmSize\times M)$,存档最优解的时间开销为 $O(MAXP)$(假设档案最大容量为 $MAXP$),更新全局最优位置矩阵的时间开销也是 $O(SwarmSize\times M)$,可以推算出 D³MOPSO 算法在最坏情况下的时间复杂度为 $O(maxgen\times SwarmSize\times M)$,其中 $maxgen$ 为迭代次数. 由于粒子群的大小和迭代次数与海量的待排序数据项规模相比起来非常小,因此,与线性多目标优化算法的复杂度 $O(N^M)$ 相比,D³MOPSO 可以看作是一种快速的多目标排序算法.

5 实验与结果

由于本文提出的 D³MOPSO 算法采用了改进的离散多目标优化思想来解决由用户自定义质量偏好的元搜索排序聚合问题,为了方便比较,本文选取基于 ANN 网络的 ListNet 排序算法[37]和基于蒙特卡罗模型的 WT-INDEG 算法[38]作为元搜索排序聚合的基线算法,同时选用 NPDA[39]算法和 BHTPSO 算法[40]作为离散多目标优化基线算法,通过在排序学习数据集、真实网络数据集、综合仿真数据集和扩展真实数据集 4 种数据集上进行对比实验来检测 D³MOPSO 算法的有效性和性能.

5.1 度量标准

1) Kullback-Tau 距离(Kendall Tau distance, KTD)

为了衡量检索结果和期望结果真值的偏差,采用 KTD 来评价排序聚合结果与期望结果的距离. 假设用户对输入查询词集合 UR 后元搜索的排序结果存在有期望真值,则此次检索结果的 KTD[41]可表示为

$$K(\tau_1,\tau_2)=|\{(i,j):i<j,(\tau_1(i)<\tau_1(j))\wedge(\tau_2(i)>\tau_2(j))\vee(\tau_1(i)>\tau_1(j))\wedge(\tau_2(i)<\tau_2(j))\}|, \tag{11}$$

其中,$\tau_1(i)$和 $\tau_2(i)$分别表示数据项 i 在实际排序结果与期望排序结果中的序号. KTD 的值越小,实际排序结果越符合用户的期望.

2) 归一化折损积累增益(normalized discounted cumulative gain, NDCG)

为了给网络数据源中的每个检索结果数据项找到聚合排序结果的最佳可能排名,可以采用归一化折损累积增益(NDCG)指标来评估排序结果的准确度[42]. 假设有 k 个待排序的数据项,那么排序结果的 NDCG@k 指标计算为

$$Gain(k)=N_k\sum_{i=1}^{k}\frac{2^{ref(\sigma^{-1}(i))}-1}{\mathrm{lb}(1+i)}, \tag{12}$$

其中,函数 ref 计算数据项 i 与检索词的相关性,σ 是 i 的理想排名,归一化参数 N_k 使得最优排序的 NDCG@k 始终等于 1.

3) 多样性指标(diversity metric, DM)

由于离散数据和排序聚合问题的特殊性,传统的间距指标(spacing metric)[43]无法有效的描述近似最优解集 EA 中的解与解之间的距离. 考虑到解之间的位置距离、语义距离和顺序距离,假设 $\boldsymbol{L}$ 为排序算法得到的近似最优解向量,针对本文的数据项排序聚合过程定义一个专门的多样性指标 DM 如下:

$$Div(\boldsymbol{L})=\sqrt{\frac{2}{|\boldsymbol{L}|(|\boldsymbol{L}|-1)}\sum_{i=1}^{|L|}\sum_{j=1}^{|L|}\left[\left(\frac{C(\boldsymbol{p}_i,\boldsymbol{p}_j)}{i-j}\right)^2+D(\boldsymbol{p}_i,\boldsymbol{p}_j)^2+\sum_{k=1}^{M}(M-k)\left(\frac{f_k(\boldsymbol{p}_i)-f_k(\boldsymbol{p}_j)}{i-j}\right)^2\right]}, \tag{13}$$

其中,$\boldsymbol{p}_i$ 和 $\boldsymbol{p}_j$ 都是排序聚合结果最优解 $\boldsymbol{L}$ 中的粒子,与之对应的是相应位置上的待排序数据项. 距离函数 C 计算粒子 $\boldsymbol{p}_i$ 和 $\boldsymbol{p}_j$ 所在位置之间的 Chebyshev 距离,差异性函数 D 计算粒子 $\boldsymbol{p}_i$ 和 $\boldsymbol{p}_j$ 对应的数据项之间的语义距离,即文档内容的差异性. M 则是用户的质量偏好向量的秩,函数 f 的含义参见定义 2. 因此,DM 值越大,排序聚合结果的多样性就越好.

4) 收敛性指标(convergency metric, CM)

在传统的连续多目标优化问题中,收敛性指标是描述近似最优解集中于理想前沿面的归一化欧氏

距离[44]. 而在离散多目标优化问题中,存在有非 Pareto 支配最优解 $\boldsymbol{L}^*=(\boldsymbol{d}_1^*,\boldsymbol{d}_2^*,\cdots,\boldsymbol{d}_{|\boldsymbol{L}^*|}^*)$(即采用非 Pareto 支配比较获得的最优解),因此,对于由 D^{3}MOPSO 算法得到的近似最优解向量 $\boldsymbol{L}$ 中的每一项 $\boldsymbol{p}_i$,可以计算得到 $\boldsymbol{p}_i$ 与 $\boldsymbol{L}^*$ 之间归一化欧氏距离的最小值:

$$cm_i=\min_{j=1}^{|\boldsymbol{L}^*|}\sqrt{\sum_{k=1}^{M}\left(\frac{f_k(\boldsymbol{p}_i)-f_k(\boldsymbol{d}_j^*)}{\max\limits_{\boldsymbol{d}_t^*\in\boldsymbol{L}^*}f_k(\boldsymbol{d}_t^*)-\min\limits_{\boldsymbol{d}_t^*\in\boldsymbol{L}^*}f_k(\boldsymbol{d}_t^*)}\right)^2}. \tag{14}$$

由此,本文考察的收敛性指标 CM 可以被定义为 D^{3}MOPSO 算法近似最优解向量中所有项的归一化距离的平均值:

$$Con(\boldsymbol{L})\triangleq\frac{1}{|\boldsymbol{L}|}\sum_{i=1}^{|\boldsymbol{L}|}cm_i. \tag{15}$$

收敛性指标代表算法得到的近似最优解集和支配最优解全集. 因此,该指标值越低就表明算法得到的解的收敛性越好,越接近理想结果.

5) 收敛迭代次数(iteration converging generation metric, ICGM)

使用多目标进化算法来求解问题时,达到收敛所需的迭代次数直接反映了算法的收敛速度,IC_i 表示第 i 次迭代结束后档案 EA 中被更新的存档粒子数,则每次进化算法执行过程中开始收敛的迭代次数 ICGM 为满足约束条件 $\forall i>Gen, IC_i<\varphi$ 的迭代数 Gen 的最小值. 其中 φ 为档案 EA 更新存档数的最小阈值,即在经过了 ICGM 次迭代之后的迭代过程中,如果每一次更新的存档数少于 φ 个,则认为本算法在第 ICGM 次迭代后达到收敛. 本文通过参数试验设置 $\varphi=2$. ICGM 越小,说明收敛速度越快. 如果迭代次数达到 $maxgen$ 时 EA 更新的存档粒子数仍不小于 φ,则说明整个迭代周期都没有收敛. 值得注意的是,如果 ICGM 比较小,而收敛性指标 CM 不理想,则说明该算法陷入早熟收敛.

6) 执行时间(Time)

为了比较各方法的效率,在同样的数据集情况下,直接使用执行时间来进行比较. 执行时间越小,说明算法效率越高.

5.2 数据集

本文将在排序学习公共数据集、真实网络数据集、综合仿真数据集和扩展真实数据集这 4 种类型的数据集上测试本文所提出的方法. 下面从数据集大小和数据特点等方面分别介绍这 4 类数据集.

1) 排序学习公共数据集(LETOR dataset)

本文选取微软的 LETOR 排序聚合数据集 MQ2008-agg 进行公共数据集上的算法测试. 该数据集包含了在 25 个匿名数据源对 800 次查询请求进行响应的结果. 由于 LETOR 数据集对每条查询结果数据项均给出了 46 个特征值,根据这些特征值可以计算出各数据项的多个质量特征值和各数据源的质量标签向量,从而构造本文提出的 D^{3}MOPSO 算法所需的实验数据集.

2) 真实网络数据集(real-world dataset)

本文选取的真实网络数据集是从 20 个真实 Web 搜索引擎数据源的检索结果中采集得到的,数据项来自于 30 次不同的检索请求,平均每次检索对每个数据源抓取约 200 条数据项. 实验中针对每个数据源、每条有效检索结果数据项计算其文本特征向量、数据源质量标签向量等特征值的方法来自于文献[30].

3) 综合仿真数据集(synthetic simulation dataset)

本文选取的另一个数据集是根据真实数据源的分布特征所生成的模拟数据集合. 综合了来自于 1 000 个仿真数据源的检索结果数据,每个数据源的平均记录条数约为 150 条,共计进行了 100 次模拟检索请求,累计生成 15 837 493 条记录. 每条记录都需要进行数值化处理,即将每条记录都表示为在 M 个质量目标维度上的取值,即如 $(v_1,v_2,\cdots,v_M)$ 的形式.

4) 扩展真实数据集

为了比较不同规模的真实数据集对算法效率的影响,本文抓取了更大规模的真实数据作为扩展真实数据集进行实验. 该数据集采纳了 135 个数据源(55 个 Web 搜索引擎和 80 个在线电子图书馆)的搜索数据,共提交查询请求 30 次(查询词与真实数据集相同),对每次查询均从 135 个数据源返回的结果中抓取前 50 条数据项,获取数据项共计 173 279 条.

结合上述 4 类数据集各自的数据规模与特点,本文对各数据集进行以下划分. 考虑到排序学习数据集 LETOR MQ2008-agg 和综合仿真数据集 2 个数据集中包含标准排序结果,适用于基于机器学习排序聚合算法的训练和参数调整过程,因此,本文选取这 2 个数据集中的 60%作为训练集,20%作为交叉确认集,20%作为测试集. 各项算法在上述 2 个数据集上的测试结果可以用以比较排序效果指标. 对于缺乏标准排序结果的真实数据集和扩展真实数据集,直接取其全部数据项作为测试集,仅计算排序结果的性能指标.

另一方面,为了方便控制数据规模,对4个数据集中的数据源和数据项也要进行筛选.对于包含800查询词的LETOR数据集,取其中60%查询词(480个)的对应结果作为训练集,另取20%查询词(160个)的对应结果作为交叉确认集,余下20%查询词(160个)的搜索结果作为测试数据集.每条查询均涉及25个数据源,对每个数据源取其前15条检索结果.对于包含30个查询词的真实数据集,对其涉及到的20个数据源,各取前50条检索结果.对于包含100个查询词的仿真数据集,按同样比例进行划分,取其中60查询词的对应结果作为训练集,另取20查询词的对应结果作为交叉确认集,余下20个查询词的检索结果作为测试集.每条查询涉及1000个数据源,各取前15条查询结果.对于包含30个查询词的扩展真实数据集,对其涉及的135个数据源各取前50条检索结果.

5.3 实验参数设置

在本文提出的D³MOPSO算法中,有一些参数需要通过训练实验来获取最佳的取值.例如定义3"Y项-支配"中的Y值将直接影响到最优解的判定,通过对综合仿真数据集在$Y\in\{1,2,3,4,5\}$中的各个可能取值上进行训练实验,得知$Y=3$时的优化效果最好.实验发现,$Y=1$会导致大量的粒子相互支配,而$Y=5$时,满足"5项-支配"关系的粒子又太少.只有当$Y=3$时,最优解集合的大小适中.

根据3.2节,某数据源检索结果数据项中最后一名的最小概率a反映了递减函数的最小值,通过训练实验发现,$a\approx 1/(3m_i)\approx 0.001\,67$为其最佳取值.而在粒子速度的变化规则中(见3.4节),I方向上的关系系数最佳取值为:$c_{I1}=n/100$,$c_{I2}=n/80$以及$c_{I3}=n/50$,J方向上的关系系数的最佳取值为:$c_{J1}=1.08$,$c_{J2}=1.49$以及$c_{J3}=2$,其中n为数据源的数量.外部存档EA的最大容量取值$MAXP=200$时效果最好,如果档案容量继续增大则会影响算法收敛速度.3.5节中介绍的目标维度上的容错误差$\theta=0.01$,表示当2个粒子在某个目标维度上的值的距离小于θ,则可认为其值相同.粒子相似阈值$\lambda=0.7$,表示当2个粒子的相似度很接近时,需进行筛选操作.最后,对于3.6节所述的引导微粒生存期的计算参数,通过多次试验调整,最终设置为$L_0=1$,$L_{\min}=1$,$l=20$.

5.4 实验与结果

本文基于5个维度的质量目标(准确性、可靠性、时效性、访问速度和领域相关性)对用户偏好和数据源质量标签进行定义和计算,在上述4类数据集上均构建D³MOPSO算法、基于ANN的ListNet算法、基于蒙特卡罗模型的WT-INDEG算法、基于多目标优化的NPDA算法、BHTPSO算法这5种算法的排序模型,进行对比实验.

采用基于listwise的ANN算法ListNet实现排序聚合的基本思路是将各数据源中的所有非重复数据项看作待排序数据项,采用选取各数据项的5维质量特征,以LETOR公共数据集和仿真数据集中提供的带标准排序结果的训练集为基础对ListNet中的神经网络进行训练,最后用训练好的网络模型对各个数据集中的全部待排序数据项进行综合排序.为了避免计算量过大,本文选取层数为1的线性神经网络,使用交叉熵表示损失函数,学习步长设为0.3,最大迭代次数$maxgen=1\,000$,采用梯度下降法来对排序函数的梯度ω进行训练,进而计算各排序序列的Top-3概率.

基于蒙特卡罗模型的WT-INDEG算法的基本思路ANN方法类似,对所有数据源中的非重复数据项进行整体排序从而实现排序聚合.将数据项的5个维度的质量特征值结合用户偏好程度作为权重,以KTD距离最小化为学习目标,每轮生成样本数量1000个,变异分位数rho设为0.1,更新概率设为0.25,最大迭代次数$maxgen=1\,000$,停止阈值为30次,即当中间结果在30次内KTD没有改进时停止学习.

基于多目标优化的NPDA算法和BHTPSO算法无法从目标的优先顺序角度实现用户对各个质量目标的偏好,因此需要以结果中的数据项的质量特征向量与用户提供的质量偏好向量在各维度上的差异最小为目标来实现对多个数据项的重新排序与聚合.为了便于比较,在本文的实验中,D³MOPSO,NPDA和BHTPSO这3种基于多目标优化的算法的最大迭代次数均设为200次,即$maxgen=200$,优化目标均为5个.其中BHTPSO算法中的惯性系数取[0.2,0.6]区间内的随机数,而加速系统分别从区间[0.5,2.0],[1.0,2.0]和[0.5,1.5]中随机生成和调整.

根据5.1节,本文选取的性能指标主要包括KTD,NDCG,DM,CM这4项指标.针对4类数据集,各排序聚合算法对每个数据集的测试子集中的所有查询词逐一完成排序聚合实验,对多次实验结果的KTD,NDCG,DM,CM指标值取平均值和标准差,以比较各算法的排序聚合表现.

1) 在排序学习数据集上的表现

LETOR MQ2008-agg 数据集是一个公共数据集,每个查询词都有对应的标准排序列表. 表 1 展示了在该数据集上各算法的排序聚合指标的比较结果.

Table 1 Performance of Algorithms on LETOR Dataset

表 1 排序学习数据集上各算法的排序聚合效果指标

Algorithms	KTD		NDCG@3		DM		CM	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std
D^3MOPSO	0.189	0.069	0.573	0.187	4.897	1.615	0.018	0.002
ANN	0.181	0.072	0.518	0.105	2.298	0.921	0.112	0.011
WT-INDEG	0.192	0.089	0.582	0.213	2.489	1.056	0.078	0.009
NPDA	0.372	0.128	0.387	0.139	6.982	2.193	0.021	0.002
BHTPSO	0.312	0.108	0.412	0.134	5.498	1.706	0.019	0.002

在与排序效果相关的 KTD 和 NDCG 指标上,D^3MOPSO 算法与 ANN 和 WT-INDEG 算法接近,这是因为 ANN 和 WT-INDEG 方法采用的排序算法具有良好的排序结果比较特性;另一方面,D^3MOPSO 算法的 KTD 和 NDCG 指标均值要远优于 NPDA 和 BHTPSO 方法,这是因为 NPDA 和 BHTPSO 方法得到的最优解是一个无序的集合,在排序上表现不佳. 在反映多样性的 DM 指标上,D^3MOPSO 算法要略差于 NPDA 和 BHTPSO 方法,这是因为 NPDA 和 BHTPSO 方法在多样性选择时的主要依据就是粒子位置,因而导致了最优解集在粒子位置上的差异很大,导致 DM 指标值更高;同时,在 DM 指标上 D^3MOPSO 算法又优于 ANN 和 WT-INDEG 方法,这是因为 ANN 和 WT-INDEG 方法对排序结果的多样性考虑不足. 最后,在体现排序结果收敛性的 CM 指标上,D^3MOPSO 算法表现接近 NPDA 和 BHTPSO 方法,并且 5 个方法在 CM 指标上都表现良好,说明各方法得到的结果都能有效地接近最优解集合. 因此,在针对 LETOR 公共数据集的实验中可以发现,D^3MOPSO 算法的排序效果比较理想,能够有效地接近最佳的聚合排序结果.

2) 在真实网络数据集上的表现

由于真实网络数据集不便获得最佳期望排序结果作为排序效果参照值,因此,在该数据集上主要分析各个算法在 DM 和 CM 两个指标上的表现. 表 2 展示了在真实网络数据集上各算法的多样性和收敛性指标的比较结果.

可以看出,在数据规模较小的真实网络数据集的实验中,D^3MOPSO 和 BHTPSO,NPDA 在收敛速度上表现十分接近,并未明显体现出优势,甚至不及 BHTPSO 算法收敛速度快,这是因为 D^3MOPSO 算法与 BHTPSO 相比,在迭代过程中需要多一步计算历史最优位置,而历史最优位置只有在数据规模较大、迭代次数较多时才能显著地加速搜索最优解的过程,抵消计算带来的时间开销.

Table 2 Performance of Algorithms on Real-world Dataset

表 2 真实网络数据集上各算法的排序聚合效果指标

Algorithms	DM		CM	
	Mean	Std	Mean	Std
D^3MOPSO	4.613	1.601	0.017	0.001
ANN	2.497	1.196	0.315	0.064
WT-INDEG	2.721	1.463	0.253	0.041
NPDA	6.395	2.179	0.019	0.001
BHTPSO	5.956	2.011	0.016	0.001

3) 在综合仿真数据集上的表现

综合仿真数据集中每个查询词对应的标准排列顺序来自于数据项自动生成后的特征值比较结果. 表 3 展示了在综合仿真数据集上各算法排序聚合指标的比较结果.

表 3 中的数据表明,在大规模的多数据源数据项的排序聚合中,ANN 和 WT-INDEG 方法在体现排序效果的 KTD 和 NDCG 指标上表现不佳,其收敛性指标 CM 的值更是表明其排序结果严重偏离最优解集. 而 NPDA 和 BHTPSO 方法虽然在多样性指标 DM 上表现良好,但是以降低了收敛性指标 CM 的值为代价的. 由于解空间过大,NPDA 和 BHTPSO 算法为了遍历解空间搜索最优解需要扩大粒子的运动幅度,容易使粒子跳跃过快,从而导致难以收敛. 通过在综合仿真数据集上的实验发现,在解空间规模骤增的情况下,只有 D^3MOPSO 算法可以在各项排序指标上能保持较为均衡而平稳的表现,其排序聚合的结果也远远优于其他方法.

Table 3 Performance of Algorithms on Synthetic Simulated Dataset

表 3 综合仿真数据集上各算法的排序聚合效果指标

Algorithms	KTD		NDCG@3		DM		CM	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std
D³MOPSO	0.265	0.098	0.594	0.107	4.411	1.372	0.035	0.003
ANN	0.398	0.167	0.472	0.184	2.072	0.959	0.626	0.149
WT-INDEG	0.339	0.163	0.458	0.179	2.503	1.280	0.575	0.146
NPDA	0.367	0.132	0.353	0.136	8.804	2.845	0.053	0.005
BHTPSO	0.309	0.113	0.452	0.161	6.225	2.074	0.038	0.003

4) 在扩展真实数据集上的表现

与真实网络数据集类似,在扩展真实数据集上仅分析各个算法在 DM 和 CM 两个指标上的表现.表 4 展示了在扩展真实网络数据集上各算法的多样性和收敛性指标的比较结果.实验结果表明:在规模较大的扩展真实网络数据集上,D³MOPSO 算法在多样性与收敛性方面与基于排序学习的 ANN 和 WT-INDEG 算法相比具有明显优势,在多样性上虽略逊于同为基于多目标优化的 NPDA 和 BHTPSO 算法,但在收敛性上比这 2 个算法稍好.

Table 4 Performance of Algorithms on ExtendedReal Dataset

表 4 扩展真实数据集上各算法的排序聚合效果指标

Algorithms	DM		CM	
	Mean	Std	Mean	Std
D³MOPSO	4.527	1.542	0.023	0.002
ANN	2.459	1.295	0.539	0.071
WT-INDEG	2.294	1.236	0.492	0.077
NPDA	5.163	1.782	0.038	0.003
BHTPSO	4.807	1.651	0.029	0.002

上述实验结果表明,在有监督的学习环境下,即在 LETOR 公共数据集和综合仿真数据集的实验中,基于机器学习的排序聚合方法排序效果指标的均值较高并且标准差较小,表现较为稳定,但一旦脱离了学习目标,基于机器学习的排序聚合方法排序效果下降明显,稳定性也降低.基于演化的方法在各种环境之下的排序表现差异不大,就排序性能的稳定性而言,基于粒子群优化的 D³MOPSO 算法和 BHTPSO 算法要优于 NPDA 算法.而综合表现上,D³MOPSO 算法在对大规模数据进行排序聚合时性能最为稳定.

值得注意的是,D³MOPSO 算法在多样性指标上表现不及 BHTPSO,NPDA 算法.这是因为以穷尽式搜索为主导思想的多目标优化算法(包括 BHTPSO 和 NPDA 算法)会保留尽可能多的非劣解,以提高最终得到的最优解集的多样性.这样做的弊端是导致算法收敛的效率往往不高.为此,BHTPSO,NPDA 两种算法分别采取了有效的策略来避免收敛速度慢的问题[39-40].多方面的实验结果表明,在适度数据规模下,BHTPSO 和 NPDA 算法在收敛速度上具备现有的多目标优化算法中的较高水平,如表 1、表 2 所示.

针对元搜索排序融合这一特定问题,由于其用户对检索结果的各项质量指标的重视程度存在差异,具有偏向性.这就意味着在这个特殊的多目标优化问题中可以通过降低对多样性的要求来"补偿"算法的收敛性.通过引入用户偏好,一方面可以根据子目标的优先顺序有效地缩小多目标优化的解空间,加快迭代速度;另一方面还能结合全局最优、局部最优以及历史最优的位置信息来优化粒子的搜索路径,进一步提升收敛性.D³MOPSO 算法综合采纳了上述 2 种思路,虽然在存档最优解时将多样性纳入了考虑,避免了出现过多相似解,但由于采用了基于用户偏好设置的双窗口滑动有序比较来缩减最优解的存档大小,保留的非劣解相较 BHTPSO,NPDA 两种算法更少,因此多样性表现不及这 2 种算法(表 2),但在收敛性的提升效果上却比 BHTPSO,NPDA 两种算法更有优势,并在数据规模较大时体现得较为明显(表 3、表 4).

5.5 原因分析

本文提出的 D³MOPSO 算法的最大优势是在大规模数据环境下的时间性能要明显优于基于排序学习模型的 ANN 算法、WT-INDEG 算法以及同样基于演化的 NPDA 算法、BHTPSO 算法.

假设 N 为输入的待聚合排序结果个数.基于机器学习的排序聚合算法的实现思路是将所有的非重复的数据项进行重新排序得到最终结果.基于 ANN 网络的 ListNet 算法通过训练神经网络为所有可能

的排序序列计算其成为最终排序序列的概率，WT-INDEG 算法通过计算每种可能排序序列的交叉熵得到最优的排序结果. 由于上述方法计算开销巨大，实际中通常采用以前 K 项排序序列代替完整排序序列的方式来减少计算量，基于神经网络的 ListNet 算法仅计算 Top-K 概率，WT-INDEG 算法仅计算 Top-K 序列的交叉熵. 这一过程通常需要 $N!/(N-K)!$ 次计算，因此当 $K>2$ 时，基于神经网络的 ListNet 算法和 WT-INDEG 算法的时间复杂度可近似表示为 $O(N^3)$. 在基于多目标优化的演化排序聚合算法中，NPDA 算法为了得到在 M 个优化目标上的最优解，每确定一个最优解的分量都需要对一张 M 维的增益表进行更新，确定 N 个数据项的排序信息共需要更新 $N(N+1)/2$ 次，所以该算法的时间复杂度为 $O(M\times N^2)$；BHTPSO 算法和本文的 D^{3}MOPSO 算法都是基于粒子群搜索的演化算法，但 BHTPSO 在每轮迭代的存档过程中，为了保持解的多样性，存档的非支配解数目远超外部档案 EA 的容量(导致 EA 频繁更新)，在数量级上接近输入总量，因此，其时间复杂度为 $O(M\times maxgen\times N)$，其中 $maxgen$ 为迭代次数；对本文提出的 D^{3}MOPSO 算法而言，正如 4.2 节所分析，在最坏情况下的时间复杂度为 $O(maxgen\times SwarmSize\times M)$，其中 $SwarmSize$ 为粒子群大小. 在大规模数据环境下，粒子群的大小和迭代次数与海量的待排序数据项规模相比非常小，因此，从时间复杂度的表示来看，D^{3}MOPSO 算法要优于上述基于排序学习的算法以及基于多目标优化的 NPDA，BHTPSO 算法. 本文的实验结果也证实了上述分析结论. 图 4 展现了各排序聚合算法在 4 个不同数据集上的执行效率比较结果.

容易看出，在数据源数目较少的排序学习数据集和真实网络数据集上，基于无监督学习的 ANN 和 WT-INDEG 算法的耗时都非常少，执行效率总体上高于基于演化的 D^{3}MOPSO，NPDA 和 BHTPSO 算法. 这是由于 2 方面原因造成的：1)基于机器学习模型的排序聚合方法单次迭代处理排序向量的能力强、耗时少，相比多目标优化方法具有天然优势，因而总的执行效率更高. 2)基于排序学习的模型的训练过程的开销与输入的数据量正相关，当输入的数据规模较小时，训练过程可以在较短的时间内完成，能迅速使模型适应对当前规模数据项集合重新排序

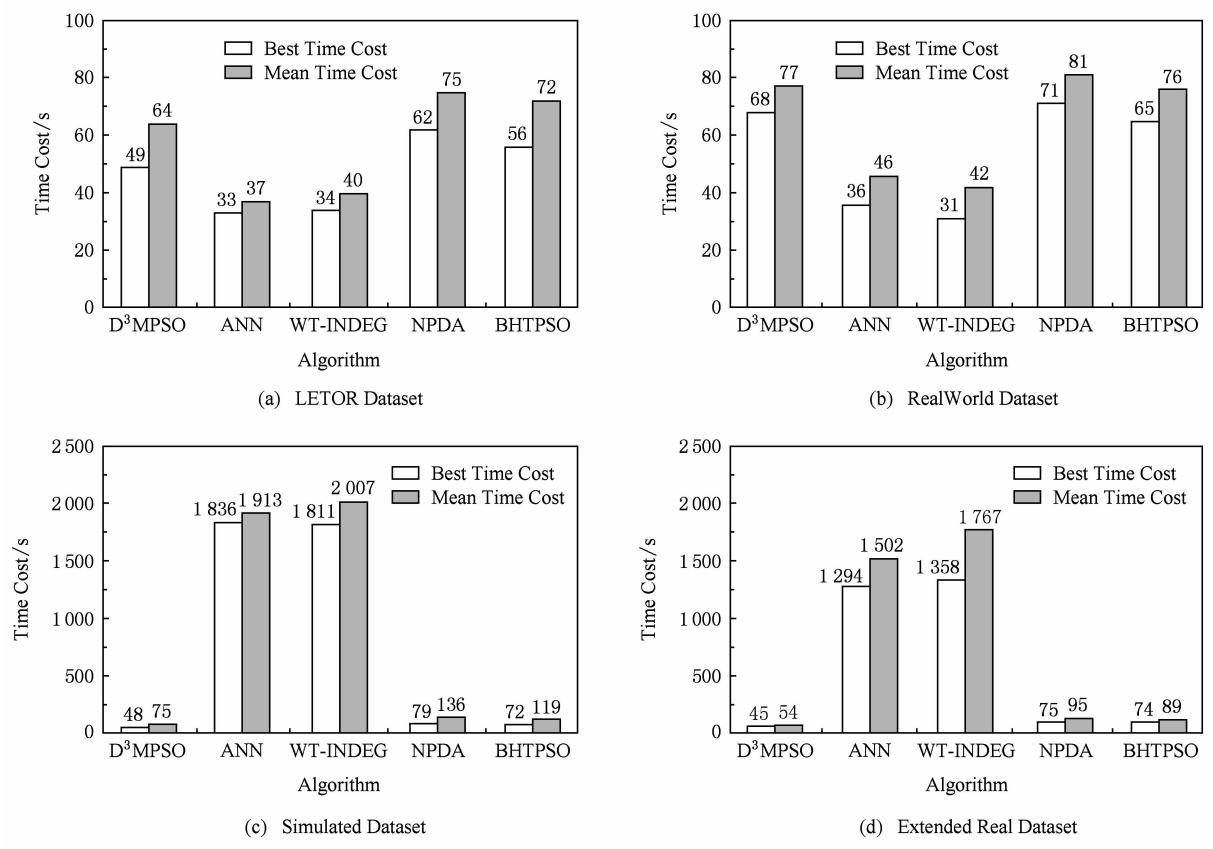


Fig. 4 Execution time comparison of algorithms on 4 datasets

图 4 4 类数据集上各算法的排序聚合执行时间对比图

的问题,而基于多目标优化的演化方法则无法从同类型问题的训练过程中获得性能优势.综合看来,当涉及的数据源较少时,基于演化的排序聚合算法在执行效率上普遍不及排序学习模型.其中,D³MOPSO算法的执行时间要略优于NPDA和BHTPSO这2种同样基于演化的方法,处于中等水平.

当排序聚合涉及到数据源数量明显增多时,如在本实验中的综合仿真数据集和扩展真实数据集上,增加的数据源和增多的数据项使得输入规模增加,基于排序学习的ANN和WT-INDEG算法所需的训练时间急剧增长,导致运行时间开销骤增.但基于多目标优化的D³MOPSO,NPDA和BHTPSO等演化算法由于其搜索解空间的方法的灵活性和敏捷性,排序聚合速度相对较快,算法的执行时间增长幅度与其数据规模的增长相比较为平缓,所以比排序学习方法的运行时间明显更短.

在基于演化的3种方法之中,D³MOPSO算法的运行效率高于NPDA和BHTPSO算法,且在数据源数目多、规模大时较为明显,如图5所示.图5中n表示数据源数量,m表示数据源规模.

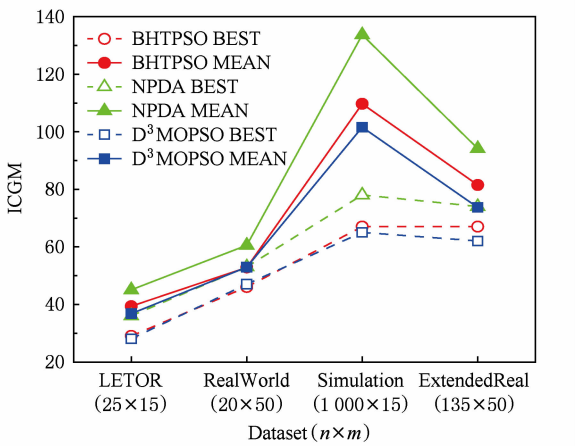


Fig. 5 Convergence time comparison of evolutionary algorithms on datasets of different sizes

图5 基于演化的算法在不同规模数据集上收敛时间对比

可以看出,在排序聚合规模较大的综合仿真数据集和扩展真实数据集上,D³MOPSO算法的收敛速度最快,明显耗时更短.这是由于D³MOPSO算法在迭代步长上比一般演化算法有所改进,同时也优化了迭代过程中粒子速度的变化策略和迭代结束时的存档策略,因此,单次迭代的效率较高,存档更新速度加快,整个算法的收敛性得以提升.

综上所述,当排序聚合涉及到的数据源规模极大时,D³MOPSO算法是一种效率较为稳定的快速搜索最优解的方法.

6 总 结

用户对元搜索结果的多维度需求和个性化质量偏好使得元搜索排序聚合成为一个多目标优化问题,然而,传统的离散多目标优化算法缺乏从数据项到粒子的映射方法,导致其无法直接应用于排序聚合过程中.为此,本文提出了一种名为D³MOPSO的多策略演化算法,通过重新定义粒子的位置、速度与支配关系,引入速度变化因子,优化粒子群的初始化、演化和融合机制,设计新的种群更新、存档更新策略和引导微粒选取策略,改进了传统的离散多目标粒子群优化算法,并结合用户自定义的对元搜索结果的质量偏好缩小了解空间,优化了粒子群的搜索路径,使元搜索引擎能够快速地从海量数据源中的数据项中聚合得到符合用户质量偏好的最佳排序结果.实验结果表明:当数据规模极大时,D³MOPSO算法的排序效率要远远高于其他排序聚合算法,是一种稳定、高效的大规模元搜索排序聚合演化算法.

参 考 文 献

[1] Desarkar M S, Sarkar S, Mitra P. Preference relations based unsupervised rank aggregation for metasearch [J]. Expert Systems with Applications, 2016, 49(C): 86-98

[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 and Technology, 2015, 66(6): 1212-1228

[3] Ali R, Naim I. User feedback based metasearching using neural network [J]. International Journal of Machine Learning and Cybernetics, 2015, 6(2): 265-275

[4] Li Lin, Xu Guangdong, Zhang Yanchun, et al. Random walk based rank aggregation to improving Web search [J]. Knowledge-Based Systems, 2011, 24(7): 943-951

[5] Keyhanipour A H, Moshiri B, Kazemian M, et al. Aggregation of Web search engines based on users' preferences in Web Fusion [J]. Knowledge-Based Systems, 2007, 20(4): 321-328

[6] Amin G R, Sadeghi A E H. Metasearch information fusion using linear programming [J]. RAIRO-Operations Research, 2012, 46(4): 289-303

[7] Meng Weiyi, Wu Zonghuan, Yu C, et al. A highly scalable and effective method for metasearch [J]. ACM Trans on Information Systems, 2001, 19(3): 310-335

[8] Amin G R, Emrouznejad A. Optimizing search engines results using linear programming [J]. Expert Systems with Applications, 2011, 38(9): 11534-11537

[9] Dwork C, Kumar R, Naor M, et al. Rank aggregation methods for the Web [C] //Proc of the 10th Int Conf on World Wide Web. New York: ACM, 2001: 613-622

[10] Coppersmith D, Fleischer L K, Rurda A. Ordering by weighted number of wins gives a good ranking for weighted tournaments [C] //Proc of the 17th Annual ACM-SIAM Symp on Discrete Algorithm. New York: ACM, 2006: 776-782

[11] Montague M, Aslam J A. Condorcet fusion for improved retrieval [C] //Proc of the 11th Int Conf on Information and Knowledge Management. New York: ACM, 2002: 538-548

[12] Wu Shengli, Li Jieyu, Zeng Xiaoqin, et al. Adaptive data fusion methods in information retrieval [J]. Journal of the Association for Information Science and Technology, 2014, 65(10): 2048-2061

[13] Klementiev A, Roth D, Small K, et al. Unsupervised rank aggregation with domain-specific expertise [C] //Proc of the 21st Int Joint Conf on Artificial Intelligence. New York: ACM, 2009: 1101-1106

[14] Qin Tao, Geng Xiubo, Liu Tieyan. A new probabilistic model for rank aggregation [J]. Advances in Neural Information Processing Systems, 2010, 23(1): 1948-1956

[15] Wang Yang, Huang Yalou, Pang Xiaodong, et al. Supervised rank aggregation based on query similarity for document retrieval [J]. Soft Computing, 2013, 17(3): 421-429

[16] Chen Yiwei, Hofmann K. Online learning to rank: Absolute vs relative [C] //Proc of the 24th Int Conf on World Wide Web. New York: ACM, 2015: 19-20

[17] Keyhanipour A H, Moshiri B, Rahgozar M. CF-Rank: Learning to rank by classifier fusion on click-through data [J]. Expert Systems with Applications, 2015, 42(22): 8597-8608

[18] Attiya G, Hamam Y. Task allocation for maximizing reliability of distributed systems: A simulated annealing approach [J]. Journal of Parallel & Distributed Computing, 2006, 66(10): 1259-1266

[19] Falzon G, Li Maozhen. Enhancing list scheduling heuristics for dependent job scheduling in grid computing environments [J]. The Journal of Supercomputing, 2012, 59(1): 104-130

[20] Moradi P, Rostami M. Integration of graph clustering with ant colony optimization for feature selection [J]. Knowledge-Based Systems, 2015, 84(C): 144-161

[21] Carrasco R, Trinh A P, Gallego M, et al. Tabu search for the Max-Mean dispersion problem [J]. Knowledge-Based Systems, 2015, 85(C): 256-264

[22] Wang Lizhe, Geng Hao, Liu Peng, et al. Particle swarm optimization based dictionary learning for remote sensing big data [J]. Knowledge-Based Systems, 2015, 79(C): 43-50

[23] Kennedy J, Eberhart R C. A discrete binary version of the particle swarm algorithm [C] //Proc of the 27th IEEE Int Conf on Systems, Man and Cybernetics. Piscataway, NJ: IEEE, 1997: 4104-4108

[24] Sha D Y, Hsu C Y. A hybrid particle swarm optimization for job shop scheduling problem [J]. Computers and Industrial Engineering, 2006, 51(4): 791-808

[25] Tang Kezong, Liu Bingxiang, Yang Jingyu, et al. Double center particle swarm optimization algorithm [J]. Journal of Computer Research and Development, 2012, 49(5): 1086-1094 (in Chinese)
(汤可宗, 柳炳祥, 杨静宇, 等. 双中心粒子群优化算法[J]. 计算机研究与发展, 2012, 49(5): 1086-1094)

[26] Chen Weineng, Zhang Jun, Chung H S H, et al. A novel set-based particle swarm optimization method for discrete optimization problems [J]. IEEE Trans on Evolutionary Computation, 2010, 14(2): 278-300

[27] Abraham A, Liu Hongbo, Zhang Weishi, et al. Scheduling jobs on computational grids using fuzzy particle swarm algorithm [C] //Proc of the 10th Knowledge-Based Intelligent Information and Engineering Systems. Berlin: Springer, 2006: 500-507

[28] Izakian H, Ladani B T, Zamanifar K, et al. A novel particle swarm optimization approach for grid job scheduling [J]. Communications in Computer & Information Science, 2010, 31(9): 100-109

[29] Ma Chao, Deng Chao, Xiong Yao, et al. An intelligent optimization algorithm based on hybrid of GA and PSO [J]. Journal of Computer Research and Development, 2013, 50(11): 2278-2286 (in Chinese)
(马超, 邓超, 熊尧, 等. 一种基于混合遗传和粒子群的智能优化算法[J]. 计算机研究与发展, 2013, 50(11): 2278-2286)

[30] Yu Wei, Li Shijun, Yang Sha, et al. Automatically discovering of inconsistency among cross-source data based on Web big data [J]. Journal of Computer Research and Development, 2015, 52(2): 295-308 (in Chinese)
(余伟, 李石君, 杨莎, 等. Web大数据环境下的不一致跨源数据发现[J]. 计算机研究与发展, 2015, 52(2): 295-308)

[31] Nejat A, Mirzabeygi P, Panahi M S. Airfoil shape optimization using improved multi-objective territorial particle swarm algorithm with the objective of improving stall characteristics [J]. Structural & Multidisciplinary Optimization, 2013, 49(6): 1-15

[32] De-La-Torre M, Granger E, Sabourin R, et al. An adaptive ensemble-based system for face recognition in person re-identification [J]. Machine Vision and Applications, 2015, 26(6): 741-773

[33] Ahmadi A. Memory-based adaptive partitioning (MAP) of search space for the enhancement of convergence in Pareto-based multi-objective evolutionary algorithms [J]. Applied Soft Computing, 2016, 41: 400-417

[34] Zhang Xianchao, Xu Wen, Gao Liang, et al. Combining content and link analysis for local Web community extraction [J]. Journal of Computer Research and Development, 2012, 49(11): 2352-2358 (in Chinese)
(张宪超，徐雯，高亮，等. 一种结合文本和链接分析的局部Web社区识别技术[J]. 计算机研究与发展，2012，49(11)：2352-2358)

[35] Mostaghim S, Teich J. Strategies for finding good local guides in multi-objective particle swarm optimization (MOPSO)[C] //Proc of the 2003 Swarm Intelligence Symp. Piscataway, NJ: IEEE, 2003: 26-33

[36] Villalobos-Arias M A, Pulido G T, Coello C A C. A proposal to use stripes to maintain diversity in a multi-objective particle swarm optimizer [C] //Proc of the 2005 Swarm Intelligence Symp. Piscataway, NJ: IEEE, 2005: 22-29

[37] Ali R, Naim I. User feedback based metasearching using neural network [J]. International Journal of Machine Learning and Cybernetics, 2015, 6(2): 265-275

[38] Desarkar M S, Sarkar S, Mitra P. Preference relations based unsupervised rank aggregation for metasearch [J]. Expert Systems with Applications: An International Journal, 2016, 49(C): 86-98

[39] Kirlik G, Sayın S. Computing the nadir point for multi-objective discrete optimization problems [J]. Journal of Global Optimization, 2015, 62(1): 79-99

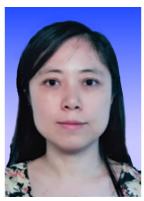
[40] Beheshti Z, Shamsuddin S M, Hasan S. Memetic binary particle swarm optimization for discrete optimization problems [J]. Information Sciences, 2015, 299(C): 58-84

[41] Buzaglo S, Etzion T. Bounds on the size of permutation codes with the kendall tau-metric [J]. IEEE Trans on Information Theory, 2015, 61(6): 3241-3250

[42] Clémencon S, Jakubowicz J. Kantorovich distances between rankings with applications to rank aggregation [C] //Proc of the 2010 European Conf on Machine Learning and Knowledge Discovery in Databases. Berlin: Springer, 2010: 248-263

[43] Tavana M, Li Zhaojun, Mobin M, et al. Multi-objective control chart design optimization using NSGA-III and MOPSO enhanced with DEA and TOPSIS [J]. Expert Systems with Applications, 2016, 50: 17-39

[44] Zitzler E, Thiele L. Multi-objective evolutionary algorithms: A comparative case study and the strength Pareto approach [J]. IEEE Trans on Evolutionary Computation, 1999, 3(4): 257-271



Tang Xiaoyue, born in 1983. PhD of Wuhan University. Lecturer of Wuhan Polytechnic University. Member of CCF. Her main research interests include Web mining, information retrieval, and artificial intelligence.

Yu Wei, born in 1987. PhD. Lecturer in Computer School of Wuhan University. Member of CCF. His main research interests include Web data ming.

Li Shijun, born in 1964. Professor and PhD supervisor in Computer School of Wuhan University. Member of CCF. His main research interests include data ming (shjli@whu.edu.cn).