·人工智能与模式识别·　　文章编号：1000-3428(2023)06-0062-09　　文献标志码：A　　中图分类号：TP311

面向安防系统的高效用语义轨迹模式挖掘

付嘉豪，杨嘉怡，李爱国

（西安科技大学 计算机科学与技术学院，西安 710054）

摘　要：在安防系统中，将大量目标轨迹先转化为语义轨迹后再进行频繁模式挖掘，有助于分析目标行为模式、识别危险源及增强安防系统内部防控。针对现有频繁模式挖掘方法未考虑目标停留点的效用差异问题，提出一种高效用语义轨迹模式挖掘算法。综合停留点兴趣度、目标停留时间以及目标语义轨迹支持度这3个参数定义语义轨迹效用值，采用蚁群算法挖掘高效用语义轨迹模式。利用精英蚂蚁策略改进蚂蚁种群的迭代方式，通过轮盘赌选择法优化蚂蚁对于下一个节点的选择策略，运用无效用编码向量剪枝策略提高算法执行效率。在Chess、Mushroom、Foodmart、Retail等4个公开数据集以及某安防系统的RFID定位数据集上的实验结果表明，相比于HUIM-ACS算法，该算法挖掘的高效用语义轨迹模式数量增加了10%~15%，运行时间减少了7%~12%。

关键词：安防系统；语义轨迹；高效用模式挖掘；蚁群算法；剪枝策略

开放科学（资源服务）标志码（OSID）：

源代码链接：https://gitee.com/fujiahao0823/high-utility-pattern-mining/blob/master

中文引用格式：付嘉豪，杨嘉怡，李爱国. 面向安防系统的高效用语义轨迹模式挖掘[J]. 计算机工程，2023，49(6)：62-70.

英文引用格式：FU J H，YANG J Y，LI A G. High-utility semantic trajectory pattern mining for security system[J]. Computer Engineering，2023，49(6)：62-70.

High-Utility Semantic Trajectory Pattern Mining for Security System

FU Jiahao，YANG Jiayi，LI Aiguo

（School of Computer Science and Technology，Xi'an University of Science and Technology，Xi'an 710054，China）

【Abstract】In security systems，transforming a large number of collected target trajectories into semantic trajectories and mining their frequency patterns are helpful in analyzing target behavior patterns，identifying hazard sources，and enhancing internal prevention and control of security systems. Existing frequent-pattern mining methods do not consider the utility difference in stay-point values. To address this issue，this study proposes a high-utility semantic trajectory pattern mining method. The concept of semantic trajectory utility value is defined by integrating three parameters：the interest of the stay point，stay time of the target at stay point，and the support of target semantic trajectory. To achieve this，an ant colony algorithm is used to mine high-utility semantic trajectory patterns. The algorithm involves the elite ant strategy to improve the iterative method of ant population and the strategy of the ant selecting the next node through the roulette selection method. Next，the invalid coding vector-pruning strategy is used to improve the execution efficiency of the algorithm. The proposed algorithm is tested on four public datasets，namely Chess，Mushroom，Foodmart，and Retail，as well as the Radio Frequency IDentification（RFID）location dataset of a certain security system. The experimental results show that the proposed algorithm increases the number of high-utility semantic trajectory patterns by 10%-15% and reduces the running time by 7%-12% compared with the Ant Colony Optimization（ACO）-based approach to mine high-utility itemsets（HUIM-ACS）.

【Key words】security system；semantic trajectory；high-utility pattern mining；ant colony algorithm；pruning strategy

DOI：10. 19678/j. issn. 1000-3428. 0064950

0　概述

安防系统在安防区域周界及内部部署大量RFID定位基站，为所有进入安防区域的人员佩戴RFID定位腕表，并通过RFID定位技术实时监测并收集人员在安防区域的移动轨迹[1-3]。如何从采集

基金项目：国家自然科学基金（62101432）。

作者简介：付嘉豪（1996—），男，硕士研究生，主研方向为数据挖掘；杨嘉怡，副教授、博士；李爱国，教授、博士。

收稿日期：2022-06-10　　**修回日期：**2022-08-06　**E-mail：**758633861@qq.com

的大量轨迹数据中提取有价值的信息提高安防系统的安全性与可靠性是安防系统面临的一个重要挑战。挖掘目标在安防系统内移动轨迹的频繁模式，可以发现人员的活动规律和行为模式，对于分析人员意图、识别危险源、找出安防薄弱点具有重要意义。

安防系统内各区域安全等级不同，现有语义轨迹频繁模式挖掘方法未能考虑到各区域之间安全等级的区别，无法挖掘出价值更高的模式[4-5]。频繁的轨迹往往都是在安全等级相对较低的区域中进行活动，在安全等级较高区域中的活动轨迹较为不频繁，但其价值更高，更值得关注。此外，目标在安防系统特定区域中停留时间的不同以及目标轨迹支持度的不同，所反映的目标意图也不同。高效用模式挖掘具有综合考虑项集支持度和价值来衡量项集效用值的特点，能弥补频繁模式挖掘只关注项集支持度的不足[6-8]。

对于高效用模式挖掘，LIU等[9-10]提出基于Apriori方法的两阶段高效用模式挖掘算法，该类算法会产生大量的候选项集，并且算法的执行时间和内存占用率过高。因此，一些学者提出基于树的两阶段算法[11-14]，该类算法的优点是减少了候选项集产生的数量与数据库的扫描次数，缺点是会产生候选项集并且数量较大。在此基础上，还有学者提出基于列表的一阶段算法来改善算法的时空效率[15-18]，该类算法的优点是不会产生候选项集，无需多次扫描数据库，缺点是构建效用列表及连接操作的代价均较大。因此，结合启发式生物优化算法来挖掘高效用模式的思路被提出。文献[19]设计基于遗传算法的启发式HUPE-GARM算法来挖掘高效用模式，该算法能通过不断迭代改进候选解决方案，但是每一次迭代中随机选择和变异生成的新项集会和上一代项集有明显区别，因此算法收敛速度较慢，而且该算法在一定的迭代次数中只能挖掘少量的高效用项集。文献[20]提出一种基于粒子群优化的高效用模式挖掘算法HUIM-BPSO，该算法相比于HUPE-GARM算法所需的参数更少，通过离散的粒子群优化机制将粒子编码为二进制向量，使用OR/NOR-Tree结构来存储粒子，从而有效地挖掘高效用模式。文献[21]提出一种基于蚁群优化算法的高效用模式挖掘算法HUIM-ACS，该算法通过特定的路径图可以避免计算同一项集的效用值，并能够检查挖掘过程中是否考虑了所有项集，通过正修剪策略和递归修剪策略来减少路径图，提高了算法的执行效率，但其挖掘的高效用模式不够全面，可能会遗漏部分高效用模式。

本文针对安防环境中停留点安全等级不同的特点，综合停留点兴趣度、目标在停留区域的停留时间和目标轨迹的支持度3个参数定义语义轨迹效用值。在HUIM-ACS算法的基础上，提出一种面向安防系统的高效用语义轨迹模式挖掘算法HUPM-IACO，挖掘安防系统中的高效用语义轨迹模式，并在4个公开数据集和真实RFID数据集上验证该算法的有效性。

1　问题描述与形式化定义

在安防系统中各个不同的停留点被称为项目。项目集合$I=\{i_1,i_2,\cdots,i_n\}$是n个不同项目i的有限集合，并且每个项目i都有对应的兴趣度。

语义轨迹数据库$D=\{T_1,T_2,\cdots,T_m\}$，其中每条语义轨迹$T_j\in D,j=1,2,\cdots,m$都是包含I的子集，并且每条语义轨迹T都有唯一标识符T_{id}。每个$T\in D$中的项目i都有其对应的停留时间Δt，Δt表示这条轨迹中目标在该停留点i的停留时间。

定义1　项目集合I中每个项目i都对应一个兴趣度$L(i)$，且$L(i)\in L$，$L=\{L(1),L(2),\cdots,L(l)\}$，则项目$i\in I$的兴趣度效用$p(i)$可表示如下：

$$p(i)=\frac{L(i)}{\sum_{j=1}^{l}L(i_j)} \tag{1}$$

定义2　$T\in D$表示D中一条语义轨迹，可表示如下：

$$T=\{(i_1,\Delta t_1),(i_2,\Delta t_2),\cdots,(i_r,\Delta t_r)\} \tag{2}$$

其中：变量$\Delta t_j,j=1,2,\cdots,r$表示$T$中第$j$个项目$i_j$的停留时间。

在T中同一个项目可重复出现。例如$T=\{(i_1,\Delta t_1),(i_2,\Delta t_2),(i_1,\Delta t_3)\}$表示目标首先在$i_1$停留了$\Delta t_1$时间，然后在$i_2$停留了$\Delta t_2$时间，最后在$i_1$停留了$\Delta t_3$时间。

定义3　项目i在T中出现的次数占T中总项目数的比值称为项目i的支持度效用，记为$q(i,T)$，可表示如下：

$$q(i,T)=\frac{\left|\{i|i\in T\}\right|}{|T|} \tag{3}$$

若$T=\{(i_1,\Delta t_1),(i_2,\Delta t_2),(i_1,\Delta t_3)\}$，则$q(i_1,T)=\frac{2}{3}$，$q(i_2,T)=\frac{1}{3}$。

定义4　项目i在T中停留时间Δt所对应的权重值称为项目i的停留时间效用，记为$\theta(i,T)$，可表示如下：

$$\theta(i,T)=\frac{\Delta t}{\max\left\{\Delta t_j\middle|\Delta t_j\in T,T\in D\right\}} \tag{4}$$

其中：$j=1,2,\cdots,|T|$。

定义5　项目$i\in I$在$T\in D$中的效用值，记为$u(i,T)$，可表示如下：

$$u(i,T)=w_1p(i)+w_2q(i,T)+w_3\theta(i,T) \tag{5}$$

其中：w_1、w_2、w_3为非负常数，$w_1+w_2+w_3=1$。

若项集$X\subseteq I$，$K=|X|$，则称X为K项集。

定义6 项集$X\subseteq I$在$T\in D$中的效用值为项集X中每个项目i的效用之和，记为$u(X,T)$，可表示如下：

$$u(X,T)=\sum_{i_j\in X\wedge X\subseteq T}u(i_j,T) \tag{6}$$

定义7 项集$X\subseteq I$在语义轨迹数据库D中的总效用值称为项集X的总效用，记为$u(X)$，可表示如下：

$$u(X)=\sum_{j=1}^{|D|}u(X,T_j) \tag{7}$$

定义8 语义轨迹$T\in D$的效用值是T中所有项目i的效用之和，记为$t_u(T)$，可表示如下：

$$t_u(T)=\sum_{i_j\in T}u(i_j,T) \tag{8}$$

定义9 语义轨迹数据库D的总效用$t_u(D)$可表示如下：

$$t_u(D)=\sum_{j=1}^{|D|}t_u(T_j) \tag{9}$$

定义10 给定语义轨迹数据库D，最小效用阈值t_{umin}可表示如下：

$$t_{umin}=\mu\cdot t_u(D) \tag{10}$$

其中：μ为用户设定的常数，$\mu\in[0,1]$。

定义11 在语义轨迹数据库D中，$\forall X\subseteq I$，设$K=|X|$，当$u(X)\geqslant t_{umin}$时，则称X是K项高效用语义轨迹项集，记为H_{UI}，可表示如下：

$$H_{UI}=\{X|u(X)\geqslant t_{umin}\} \tag{11}$$

定义12 给定语义轨迹数据库D，项集$X\subseteq I$的语义轨迹加权效用值表示D中包含X的所有语义轨迹效用值的总和，记为$t_{wu}(X)$，可表示如下：

$$t_{wu}(X)=\sum_{X\in T\wedge T\in D}t_u(T) \tag{12}$$

定义13 当项集$X\subseteq I$的语义轨迹加权效用值$t_{wu}(X)\geqslant t_{umin}$时，则$X$是高效用语义轨迹加权项集，记为$H_{TWUI}$，可表示如下：

$$H_{TWUI}=\{X|t_{wu}(X)\geqslant t_{umin}\} \tag{13}$$

性质1 事务加权向下闭包：如果项集X的语义轨迹加权效用值$t_{wu}(X)<t_{umin}$，则X的超集都不是高效用语义轨迹项集；如果项集X的语义轨迹加权效用值$t_{wu}(X)\geqslant t_{umin}$，则$X$的非空子集都是高效用语义轨迹项集。

证明 设项集X^k为K项集，X^{k+1}为$K+1$项集，满足$X^k\subseteq X^{k+1}$，则包含项集X^{k+1}的语义轨迹为包含项集X^k语义轨迹的超集。因为$t_{umin}>t_{wu}(X^k)=\sum_{X^{k+1}\subseteq T\wedge T\in D}t_u(T)\geqslant\sum_{X^{k+1}\subseteq T\wedge T\in D}t_u(T)=t_{wu}(X^{k+1})\geqslant u(X^{k+1})$，所以对于任意$X^k$的任意超集和非空子集都满足事务权重向下闭包的性质。

2 高效用语义轨迹模式挖掘算法

2.1 HUPM-IACO算法

根据相关研究可知，使用蚁群优化算法来挖掘高效用语义轨迹模式的效果比其他启发式生物优化算法的效果更好[19-21]。因此，基于蚁群优化算法思想，本文提出高效用语义轨迹模式挖掘算法HUPM-IACO。使用蚁群算法挖掘高效用语义轨迹模式的基本原理为：根据对语义轨迹效用值的定义计算出每条语义轨迹模式的效用值，通过蚁群优化算法找出所有满足最小效用阈值的语义轨迹模式。将项目的效用值作为目标函数，当项目效用值取最大值时得到最优解。解空间由所有高效用语义轨迹加权1项集所组成的路径构成。高效用语义轨迹模式挖掘问题转化为蚁群优化问题的步骤为：1）根据语义轨迹数据库D构建蚂蚁的路径图，并结合停留点兴趣度、目标在停留点的停留时间和目标语义轨迹的支持度3个参数来计算语义轨迹的效用值和语义轨迹加权效用值；2）更新局部信息素和全局信息素；3）当路径效用值高的蚂蚁释放的信息素较多，随着迭代次数的增加，效用值较高的路径上积累的信息素浓度逐渐增高，选择该路径的蚂蚁个数逐渐增多。若蚂蚁所经过路径的效用值大于最小效用阈值，则称该路径为高效用语义轨迹。算法创新之处为：定义语义轨迹效用值的计算方法，并采用精英蚂蚁策略、轮盘赌选择法和无效用编码向量剪枝策略来改进蚁群优化算法，从而挖掘出尽可能多的高效用语义轨迹模式。HUPM-IACO算法的伪代码具体如下：

算法1 HUPM-IACO算法

输入 语义轨迹数据库D，非负常数w_1、w_2和w_3，信息素矩阵$\boldsymbol{Q}$，种群大小S，信息素挥发因子η，最小效用阈值t_{umin}，最大迭代次数M_{AXiter}

输出 高效用语义轨迹集合H_{UP}

1.初始化Q、H_{UPs}、i_{ter}、A、H_{UP}

2.扫描语义轨迹数据库D一次，计算每个项目的语义轨迹加权效用，删除低效用语义轨迹加权1项集，只保留高效用语义轨迹加权1项集H_{TWUI-1}

3.根据H_{TWUI-1}的个数构建信息素矩阵Q

4.根据式(15)更新信息素矩阵Q中每个元素的值

5.$H_{UPs}=\varnothing$、$H_{UP}=\varnothing$、$i_{ter}=1$

6.while $i_{ter}<M_{AXiter}$

7.for 每个蚂蚁A

8.if 迭代次数$i_{ter}=1$

9.根据算法2构建蚂蚁路径图G

10.根据算法3删除不可能存在高效用语义轨迹模式的分支

11.end if

12.if A访问过G中一个项目

13.根据算法4选择下一个项目

14.if $t_{wu}(i_p,i_q)\geqslant t_{umin}$

15.使用式(16)更新局部信息素值

16.if $u(i_p,i_q)\geqslant t_{umin}$

17.$H_{UPs}=H_{UPs}\cup u(i_p,i_q)$
18.使用式(17)更新全局信息素
19.根据算法5更新蚂蚁种群
20.end if
21.end if
22.end if
23.end for
24.$i_{ter}++$
25.end while
26.$H_{UP}=H_{UP}\cup H_{UPs}$
27.return H_{UP}

算法1中的步骤1为初始化，步骤2~步骤27为挖掘高效用语义轨迹模式的主要步骤，直到没有蚂蚁可以找到高效用语义轨迹模式。步骤2扫描语义轨迹数据库D得到所有的高效用语义轨迹加权1项集H_{TWUI-1}。步骤3根据H_{TWUI-1}的个数来构建信息素矩阵$\boldsymbol{Q}$。步骤4根据式(15)计算出信息素矩阵$\boldsymbol{Q}$中每个元素的值。步骤6初始化迭代次数为1。步骤7~步骤22递归地描述了下一次迭代中蚂蚁更新信息素值和挖掘高效用语义轨迹H_{UPs}的过程。步骤10根据算法2构建蚂蚁的路径图G。步骤11对G剪枝，根据算法3将不可能挖掘出高效用语义轨迹模式的节点从G中删除。步骤13蚂蚁根据轮盘赌选择法选择下一个节点。步骤15根据式(16)更新局部信息素。步骤18根据式(17)更新全局信息素。步骤19根据精英蚂蚁策略更新蚂蚁种群。步骤25将发现的新的高效用语义轨迹模式加入集合H_{UP}。步骤27输出所有的高效用语义轨迹模式集合H_{UP}。

2.2　蚂蚁路径构建

由于语义轨迹加权效用值$t_{wu}(X)$具有向下闭包的性质，因此创建路径图时只需要包含所有的高语义轨迹加权1项集H_{TWUI-1}，不需要包含语义轨迹数据库中所有的项集。在计算得到所有的H_{TWUI-1}后，将其递增排序，然后通过算法2生成蚂蚁路径图。蚂蚁路径图构建的算法伪代码具体如下：

算法2　蚂蚁路径构建算法
输入　增序排列的H_{TWUI-1}列表l_{ist}
输出　蚂蚁路径图G
1.初始化G=∅
2.在G中建立起点s
3.for each H_{TWUI-1} in l_{ist}
4.$G\leftarrow H_{TWUI-1}$
5.生成一个从s到H_{TWUI-1}的有向链接
6.if H_{TWUI-1}不是l_{ist}的最后一个项目
7.在s后建立一个子列表l_{ist_i}
8.循环执行步骤3~步骤7，i=i+1
9.else 输出G
10.end if
11.end for
12.return G

算法2输入所有H_{TWUI-1}递增排列的列表l_{ist}，输出蚂蚁的路径图G，G中每个节点对应一个项目。步骤1为初始化路径图G。步骤2在路径图G中建立一个起点s。步骤3~步骤8为在路径图G中循环建立有向链接，直到所有的H_{TWUI-1}组成的路径都被建立。步骤12输出路径图G。

2.3　位图矩阵表示的语义轨迹数据库

HUIM-IACO算法使用位图的形式来表示语义轨迹数据库D，有利于后续进行剪枝和计算目标语义轨迹的效用值，有助于挖掘高效用语义轨迹模式。$D=\{T_1,T_2,\cdots,T_m\}$是一个语义轨迹数据库，项目集合$I=\{i_1,i_2,\cdots,i_n\}$，将D表示成矩阵的形式称为D的位图矩阵，记为$\boldsymbol{M}_D$。$\boldsymbol{M}_D$是一个$m\times n$的布尔型矩阵。$\boldsymbol{M}_D$的第c行、第j列元素表示轨迹T_c是否包含项目i_j，其定义如式(14)所示。$\forall X\subseteq I,X$的编码向量$\boldsymbol{B}_{it,x}$为X中的每个项目$\boldsymbol{B}_{it,i}$通过按位与运算得到。$\forall X\subseteq I,\forall Y\subseteq I,\boldsymbol{B}_{it,XUY}$由$\boldsymbol{B}_{it,X}$和$\boldsymbol{B}_{it,Y}$按位与计算得到。

$$\boldsymbol{M}_{c,j}=\begin{cases}1,i_j\in T_c\\0,i_j\notin T_c\end{cases}\tag{14}$$

其中：$c=1,2,\cdots,m;j=1,2,\cdots,n$。

2.4　无效用编码向量修剪策略

如果$\boldsymbol{M}_D$中项集$X\subseteq I$的位图编码向量$\boldsymbol{B}_{it,X}$全部由0元素构成，则称$\boldsymbol{B}_{it,X}$为无效用编码向量，反之，则称$\boldsymbol{B}_{it,X}$为有效用编码向量。如果$\boldsymbol{B}_{it,X}$是无效用编码向量，根据性质1则该X不可能是高效用项集。将无效用编码向量代表的项集修剪策略称为无效用编码向量修剪策略。无效用编码向量修剪策略的算法伪代码如下：

算法3　无效用编码向量修剪算法
输入　所有项集的编码向量$\boldsymbol{v}$
输出　有效用编码向量$\boldsymbol{u}$
1.计算所有v中元素1的数量，记为b
2.用$i_1,i_2,\cdots,i_b$表示编码向量中包含的项目
3.$H_U=B_{it}(i_1)$
4.for $K\in(1,b)$
5.$H_{U'}=H_U IB_{it}(i_k)$
6.if $H_{U'}$是一个有效用编码向量
7.$H_{U'}=H_U$
8.将v中对应的$i_k\rightarrow 0$
9.end if
10.$H_U=H_{U'}$
11.end for
12.return有效用编码向量u

算法3输入所有的项集编码向量，输出所有的有效用编码向量，将无效用编码向量去除，并且输出的有效用编码向量包含在原始项集编码向量中，不会产生原本不存在的项集编码向量，确保该向量在语义轨迹数据库中真实存在。步骤1将每个编码向量$\boldsymbol{v}$中1的数量记为b。步骤2用$i_1,i_2,\cdots,i_b$表示编码向量中包含的项目。步骤3用第一个项目的编码向量初始化最终结果$\boldsymbol{H}_U$。步骤4~步骤11循环修剪无效用编码向

量。步骤12输出所有有效用编码向量$\boldsymbol{u}$。

2.5 信息素更新规则

使用定义5所定义的项目效用值来更新局部信息素,使用定义12所定义的语义轨迹加权效用值来更新全局信息素。首先建立一个矩阵$\boldsymbol{Q}$用来记录两个项目之间的信息素值。高语义轨迹加权效用1项集$H_{\text{TWUI-1}}$的个数记为h,则$\boldsymbol{Q}$为一个$h\times h$的矩阵。$\boldsymbol{Q}$中每个元素$\boldsymbol{Q}_{c,j}$,$c=1,2,\cdots,h$,$j=1,2,\cdots,h$代表两个项目i和i'之间路径的信息素。$\boldsymbol{Q}$对角线上的元素均为0,其他元素的信息素值初始化如下:

$$\boldsymbol{Q}_{c,j}=\begin{cases}u(i,i')/2, t_{\text{wu}}(i,i')\geqslant t_{\text{umin}}\\ 0, t_{\text{wu}}(i,i')<t_{\text{umin}}\end{cases} \tag{15}$$

在局部信息素更新时,$\boldsymbol{M}_D$中的每个元素按照式(16)进行更新:

$$\boldsymbol{Q}_{c,j}=\boldsymbol{Q}_{c,j}+u(i,i')/\eta \tag{16}$$

在全局信息素更新时,按照式(17)进行更新:

$$\boldsymbol{Q}_{c,j}=\boldsymbol{Q}_{c,j}+t_{\text{wu}}(i,i')/\eta \tag{17}$$

其中:η为信息素挥发因子,用来稀释信息素浓度,$\eta\geqslant 0$。

2.6 轮盘赌选择法

在蚁群算法中,将每条路径的概率看作是轮盘的一个扇面,旋转轮盘指针落在哪个扇面,蚂蚁在选择下一个路径时就选择该扇面对应的路径。轮盘赌选择法的伪代码如下:

算法4 轮盘赌选择法

输入 所有的路径节点集合$I'=[i_1,i_2,\cdots,i_r]$,种群S

输出 选择的路径节点i_v

1.通过定义5计算出I′中所有项目的效用值u(i,T)

2.计算u(i,T)/t_u(T)得到节点被选中概率P(i)

3.初始化i_v

4.计算每个项目的累计概率$q_j=\sum_{j=1}^{S}P(j)$

5.for 所有的路径节点

6.在[0,1]内产生一个均匀分布的随机数σ

7.if $q_{j-1}<\sigma<q_j$

8.$i_j\rightarrow i_v$

9.end if

10.end for

11.return i_v

在算法4中,输入所有的路径节点集合I'(I'实际上是项目集合的子集),输出选择的路径节点i_v,$i_v\in I'$,$u(i,T)$是项目i在语义轨迹T中的效用值,$t_u(T)$是一条语义轨迹T的效用值,$u(i,T)/t_u(T)$是项目i在语义轨迹T中被选中的概率。步骤1计算所有项目的效用值$u(i,T)$。步骤2计算每个项目被选中的概率$P(i)$。步骤3初始化要选择的路径节点i_v。步骤4计算每个项目的累计概率。步骤5~步骤10循环通过轮盘赌法选择路径节点i_v。步骤11输出选择的路径节点i_v。

2.7 精英蚂蚁策略

经过多个节点后仍未找到高效用语义轨迹模式的蚂蚁,称为低效用蚂蚁。精英蚂蚁策略就是要将部分低效蚂蚁删除,将精英蚂蚁复制,让更多的精英蚂蚁去寻找高效用语义轨迹模式,提高算法效率。具体做法为:在每次迭代完成后,将蚂蚁按照访问过的节点效用值进行排序,将排名靠后的若干蚂蚁删除,并将排名靠前的蚂蚁进行复制,复制的数量与删除的数量相等。精英蚂蚁策略的算法伪代码如下:

算法5 精英蚂蚁算法

输入 每个蚂蚁经过节点的效用值之和t_u,需要更新的低效用蚂蚁数量n

输出 更新后的蚂蚁种群S_N

1.建立一个列表l_{ist}存储t_u

2.降序排列列表l_{ist}

3.for j=0,j++

4.delete l_{ist}的最后一个元素

5.copy l_{ist}的第一个元素

6.update l_{ist}

7.if j=n

8.$S_N\leftarrow l_{ist}$

9.end if

10.end for

11.return S_N

算法5为一次迭代完成后更新蚂蚁种群的操作。在每次迭代完成后都对蚂蚁种群进行更新。通过精英蚂蚁策略使得蚂蚁种群更加高效,在有限时间内能找到更多的高效用语义轨迹模式。算法5输入为每只蚂蚁已经过的路径节点效用之和t_u和将要更新的低效蚂蚁数量n,输出为更新后的蚂蚁种群S_N。步骤1建立一个列表l_{ist}来存储t_u。步骤2降序排列列表l_{ist}。步骤3~步骤10循环更新低效用蚂蚁直到次数达到n。步骤11输出更新后的蚂蚁种群S_N。

3 实验结果与分析

3.1 数据集与环境设置

使用4个公开数据集Chess[22]、Mushroom[23]、Foodmart[24]以及Retail[25]进行实验。通过对比本文提出的HUPM-IACO算法与具有代表性的HUPE-GARM[19]、HUIM-BPSO[20]以及HUIM-ACS[21]算法的运行时间和挖掘的高效用语义轨迹模式数量来评估HUPM-IACO算法的性能。

同时,使用从某安防系统的RFID定位子系统中采集的真实目标轨迹数据集(简称为RFID数据集)进行实验来进一步验证本文提出的HUPM-IACO算法的有效性。RFID数据集的采集方法为:通过目标佩戴RFID定位腕表在区域内进行移动,计算基站返回的RSSI值来确定目标位置。该过程共采集10个目标、5 488个采样点、916条轨迹,定位区域分为6个区域,编号为a~f。表1给出了5个数据集的数据

基本信息。

表1　数据集基本信息

Table 1　Basic information of the datasets

数据集	轨迹总数/条	项目总数/个	项目平均数/个
Chess[22]	3 196	76	37.00
Mushroom[23]	8 416	128	23.00
Foodmart[24]	4 141	1 559	4.42
Retail[25]	88 162	16 470	10.30
RFID	916	6	3.00

为了公平地对比各算法的运行时间,将4种算法的最大迭代次数都设置为200,蚂蚁种群规模设置为30。由于Chess、Mushroom、Foodmart、Retail等4个数据集只有事务和项目等属性,没有停留时间等数据,无法使用定义5的方法来计算效用值,因此使用商品价值和商品个数相乘的方式计算项目的效用值。在所有实验中,语义轨迹的效用值权重常数均为$w_1=w_2=w_3=1/3$。

实验在配置为酷睿i5-6500(3.20 GHz双核)、8 GB内存的计算机上进行,系统为Win10,使用Python 3.8.0与PyCharm编写程序。

3.2　算法运行时间对比

图1分别给出了HUPE-GARM、HUIM-BPSO、HUIM-ACS和HUPM-IACO算法在Chess、Mushroom、Foodmart、Retail等4个公开数据集上的运行效率,通过改变最小效用阈值来观察算法的运行时间。

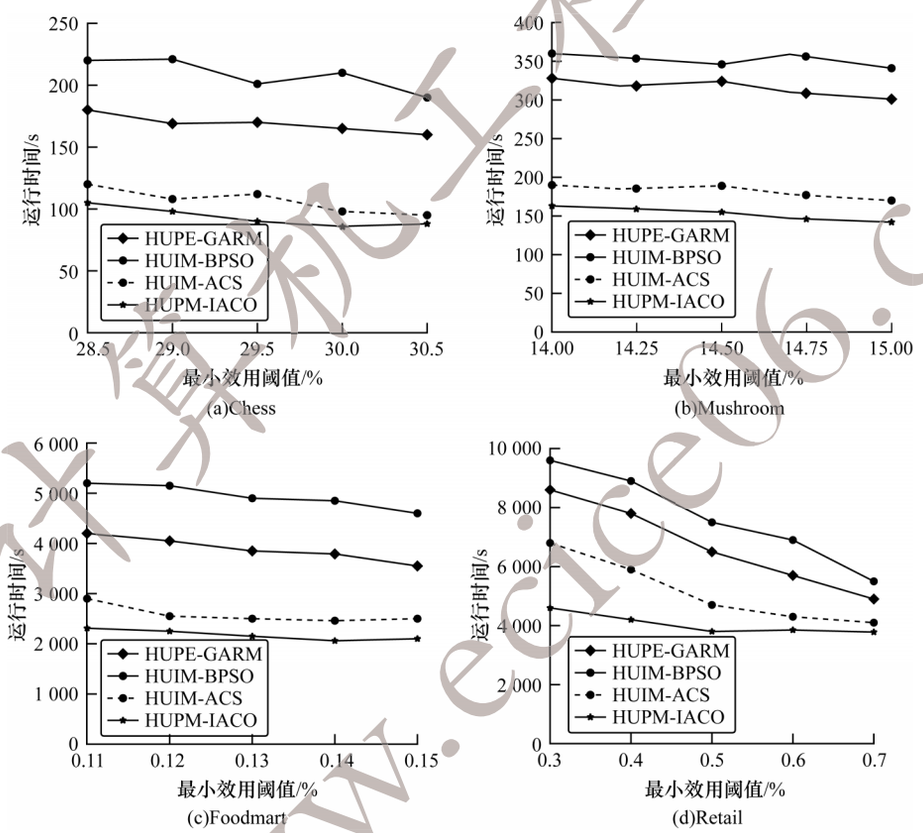

图1　算法运行时间比较

Fig.1　Comparison of the running time of algorithms

由图1(a)可以看出,在Chess数据集上,随着最小效用阈值的增加,HUPE-GARM、HUIM-BPSO和HUIM-ACS算法的运行时间逐渐减少,HUPM-IACO算法的运行时间变化不大。由图1(b)可以看出,在Mushroom数据集上,随着最小效用阈值的增加,HUPE-GARM和HUIM-BPSO算法的运行时间略微减少,HUIM-ACS和HUPM-IACO算法运行时间变化不大。由图1(c)可以看出,在Foodmart数据集上,随着最小效用阈值的增加,4种算法的运行时间变化不大。由图1(d)可以看出,在Retail数据集上,随着最小效用阈值的增加,HUPE-GARM、HUIM-BPSO和HUIM-ACS算法的运行时间显著降低,HUPM-IACO算法的运行时间略微降低。

综合4个数据集的实验结果来看,HUPM-IACO算法在运行时间上比HUIM-ACS算法快7%~12%,比HUPE-GARM算法快40%~45%,比HUIM-BPSO算法快50%~55%,说明了HUPM-IACO算法在运行时间上优于3个对照算法。

3.3　挖掘的高效用语义轨迹模式数量对比

图2给出了HUPE-GARM、HUIM-BPSO、HUIM-ACS和HUPM-IACO算法分别在Chess、Mushroom、Foodmart、Retail等4个数据集上挖掘的高效用语义

轨迹模式数量，通过改变最小效用阈值来观察算法挖掘的高效用语义轨迹模式数量的变化情况。

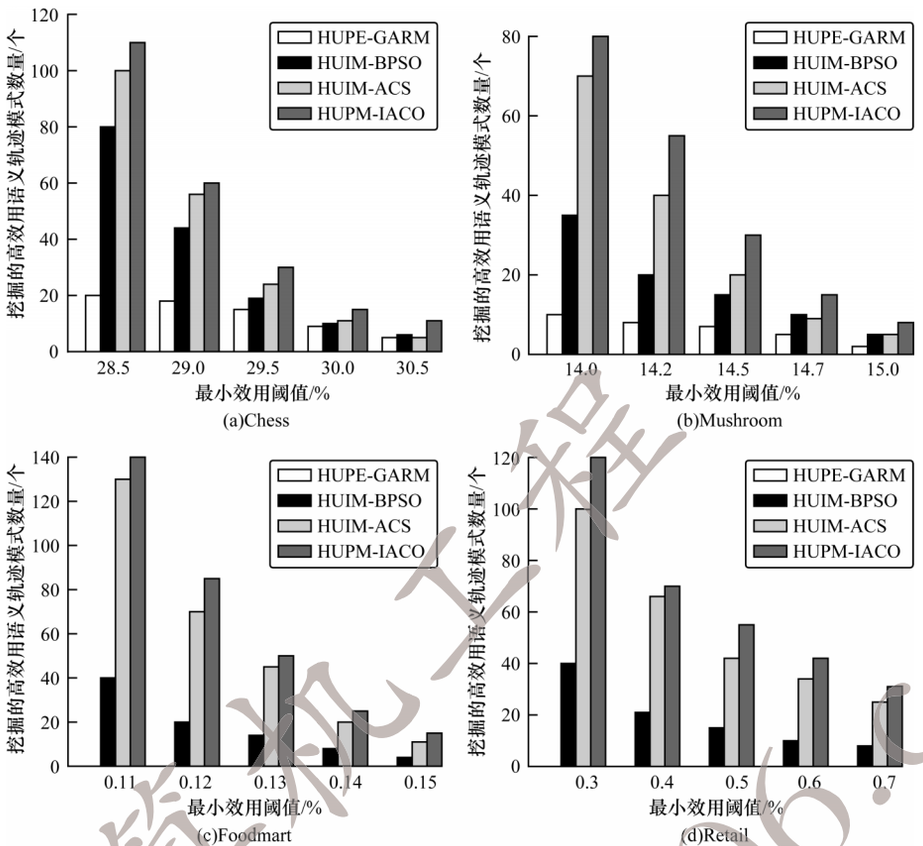

图2 算法挖掘的高效用语义轨迹模式数量比较

Fig.2 Comparison of the number of high-utility semantic trajectory patterns mined by the algorithms

由图2(a)可以看出，在Chess数据集上，随着最小效用阈值的增加，4种算法挖掘的高效用语义轨迹模式数量减少，并且HUPM-IACO、HUIM-ACS和HUIM-BPSO算法的挖掘数量远大于HUPE-GARM算法。由图2(b)可以看出，在Mushroom数据集上，4种算法的挖掘情况和Chess数据集相似。由图2(c)和图2(d)可以看出，在Foodmart和Retail数据集上，HUPE-GARM算法无法挖掘出高效用语义轨迹模式，HUIM-BPSO、HUIM-ACS和HUPM-IACO算法挖掘出的高效用语义轨迹模式数量随着最小效用阈值的增加逐渐减少，并且HUIM-ACS和HUPM-IACO算法挖掘的数量远多于HUIM-BPSO算法。

综合4个数据集的实验结果来看，HUPM-IACO算法在挖掘高效用语义轨迹模式数量上比HUIM-ACS算法多10%~15%，比HUIM-BPSO算法多27%~45%，比HUPE-GARM算法多70%~80%，说明了HUPM-IACO算法在挖掘高效用模式数量上优于3个对照算法。

3.4 在安防系统中挖掘的高效用语义轨迹模式

在某安防系统中，将6个安防区域(停留点)的安全等级划分为5个级别，按安全程度由低到高记为1~5。用安全等级来表示该停留点的兴趣度，如表2所示。由表2可以看出：停留点c和f的安全等级为1；停留点b的安全等级为2；停留点a的安全等级为3；停留点e的安全等级为4；停留点d的安全等级为5。可见，各停留点的安全程度关系为d>e>a>b>c=f。

表2 兴趣度效用

Table 2 Interest utility

停留点	安全等级
a	3
b	2
c	1
d	5
e	4
f	1

表3为RFID数据集的语义轨迹数据库D前10条语义轨迹，每条语义轨迹包含停留点和停留时间信息。由表3可以看出，轨迹T_1含有3个停留点：首先在停留点a停留了10 min，然后前往停留点c停留了15 min，最后在停留点e停留了7 min。语义轨迹数据库的位图矩阵表示如表4所示，其中停留时间单位为min。

表3　语义轨迹数据库

Table 3　Semantic trajectory database

轨迹编号	停留点和停留时间
T_1	(a,10),(b,15),(c,7)
T_2	(b,14),(c,25),(e,17),(d,8)
T_3	(a,13),(c,24),(d,31)
T_4	(d,26),(e,18)
T_5	(c,16),(d,24)
T_6	(b,23),(f,24)
T_7	(b,27),(d,21),(e,28)
T_8	(a,5),(c,12),(e,17),(d,24)
T_9	(a,17),(b,22),(f,30)
T_{10}	(b,60),(c,25),(e,30),(f,24),(d,30)

表4　语义轨迹数据库的位图表示

Table 4　Bitmap representation of semantic trajectory database

轨迹编号	a	b	c	d	e	f
T_1	1	1	1	0	0	0
T_2	0	1	1	1	1	0
T_3	1	0	1	1	0	0
T_4	0	0	0	1	1	0
T_5	0	0	1	1	0	0
T_6	0	1	0	0	0	1
T_7	0	1	0	1	1	0
T_8	1	0	1	1	1	0
T_9	1	1	0	0	0	1
T_{10}	0	1	1	1	1	1

设项目在语义轨迹中的效用值权重常数为$w_1=w_2=w_3=1/3$。图3给出了HUPE-GARM、HUIM-BPSO、HUIM-ACS和HUPM-IACO算法在RFID数据集上挖掘的高效用语义轨迹模式数量及运行时间情况。由图3(a)可以看出,4种算法在RFID数据集上随着最小效用阈值增加,运行时间逐渐减少,HUPM-IACO算法在运行效率上最优。由图3(b)可以看出,4种算法在RFID数据集上随着最小效用阈值增加,挖掘的高效用语义轨迹模式数量逐渐减少,HUPM-IACO算法挖掘的高效用语义轨迹模式数量最多。由图3(c)可以看出,4种算法在最小效用阈值为10%的情况下,在RFID数据集上,随着最小支持度阈值逐渐增加,各算法运行时间逐渐减少,HUPM-IACO算法在运行效率上最优。由图3(d)可以看出,4种算法在最小效用阈值为10%的情况下,在RFID数据集上,随着最小支持度阈值逐渐增加,各算法挖掘的高效用语义轨迹模式数量逐渐减少,本文算法挖掘的高效用语义轨迹模式数量最多。

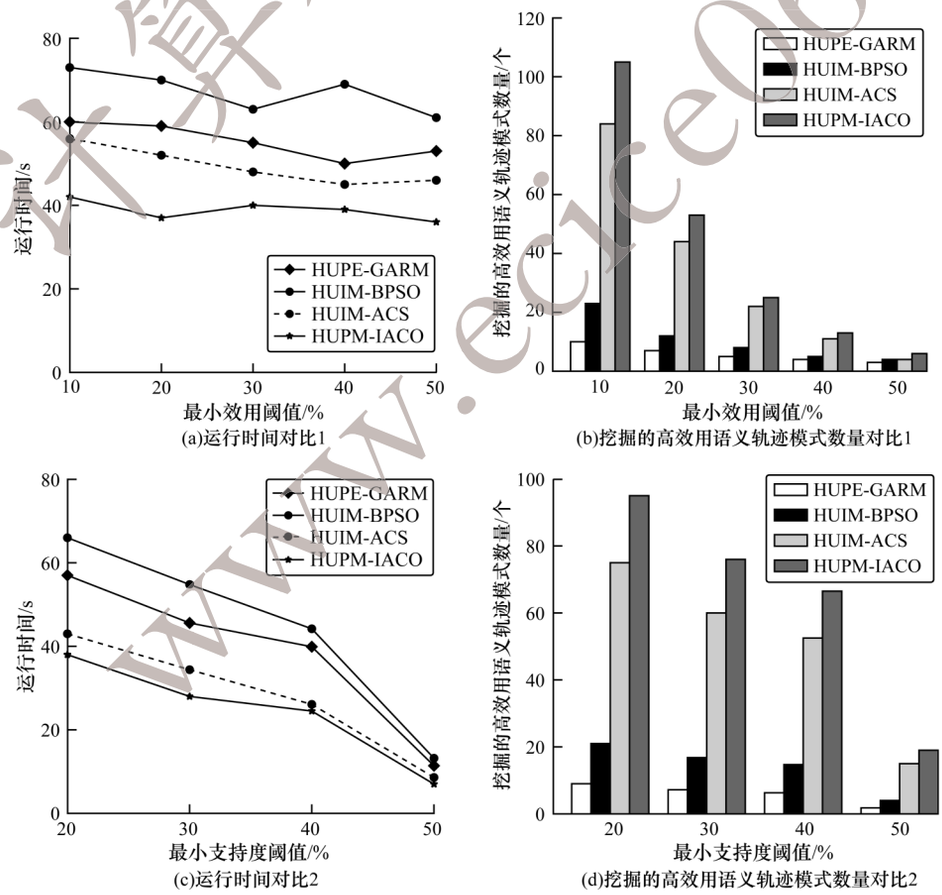

图3　RFID数据集上算法运行时间与挖掘的高效用语义轨迹模式数量比较

Fig.3　Comparison of the running time and the number of high-utility semantic trajectory patterns mined by the algorithms on the RFID dataset

综合以上实验结果可以得出，HUPM-IACO算法在运行时间和挖掘的高效用语义轨迹模式数量上比现有的启发式高效用挖掘算法更具优势。

4 结束语

本文针对安防系统特点，结合安防区域兴趣度、目标在停留区域的停留时间以及轨迹支持度来计算轨迹效用值，提出挖掘高效用语义轨迹模式算法HUPM-IACO，采用精英蚂蚁策略、轮盘赌选择法和无效用编码向量剪枝策略来提高算法执行效率。在4个公开数据集和某安防系统的RFID数据集上的实验结果证明了HUPM-IACO算法相比于对照算法具有更高的运行和挖掘效率。后续将考虑在安防环境中挖掘含有低效用的语义轨迹模式，进一步提高安防系统的安全性。

参考文献

[1] ZHU C, ZHAO S, XIA Y, et al. An improved three-point localization method based on RSS for transceiver separation RFID systems[J]. Measurement, 2022, 187: 110283.

[2] HAYWARD S J, EARPS J, SHARPE R, et al. A novel inertial positioning update method, using passive RFID tags, for indoor asset localisation[J]. CIRP Journal of Manufacturing Science and Technology, 2021, 35: 968-982.

[3] 陈博. 基于物联网的实物保护系统关键技术研究[D]. 西安: 西安科技大学, 2021.
CHEN B. Research on key technologies of physical protection system based on Internet of Things[D]. Xi'an: Xi'an University of Science and Technology, 2021. (in Chinese)

[4] ZHANG Z, ZHAO X G, ZHANG Y C, et al. Efficient mining of hotspot regional patterns with multi-semantic trajectories[J]. Big Data Research, 2020, 22: 100157.

[5] 吴瑕, 唐祖锴, 祝园园, 等. 近似到达时间约束下的语义轨迹频繁模式挖掘[J]. 软件学报, 2018, 29(10): 3184-3204.
WU X, TANG Z K, ZHU Y Y, et al. Frequent pattern mining with approximate arrival-time in semantic trajectories[J]. Journal of Software, 2018, 29(10): 3184-3204. (in Chinese)

[6] 张春砚, 韩萌, 孙蕊, 等. 高效用模式挖掘关键技术综述[J]. 计算机应用研究, 2021, 38(2): 330-340.
ZHANG C Y, HAN M, SUN R, et al. Survey of key technologies for high utility patterns mining[J]. Application Research of Computers, 2021, 38(2): 330-340. (in Chinese)

[7] AZIYA S V S, GEORGE J. An efficient high utility pattern mining for finding time based customer purchase behavior[C]//Proceedings of International Conference on Computer Networks and Inventive Communication Technologies. Berlin, Germany: Springer, 2019: 400-407.

[8] 李慧, 刘贵全, 瞿春燕. 频繁和高效用项集挖掘[J]. 计算机科学, 2015, 42(5): 82-87, 123.
LI H, LIU G Q, QU C Y. Mining frequent and high utility itemsets[J]. Computer Science, 2015, 42(5): 82-87, 123. (in Chinese)

[9] LIU Y, LIAO W, CHOUDHARY A. A two-phase algorithm for fast discovery of high utility itemsets[C]//Proceedings of Pacific-Asia Conference on Knowledge Discovery and Data Mining. Berlin, Germany: Springer, 2005: 689-695.

[10] LIU Y, LIAO W K, CHOUDHARY A. A fast high utility itemsets mining algorithm[C]//Proceedings of the 1st International Workshop on Utility-based Data Mining. New York, USA: ACM Press, 2005: 90-99.

[11] RYANG H, YUN U. Top-*k* high utility pattern mining with effective threshold raising strategies[J]. Knowledge-Based Systems, 2015, 76: 109-126.

[12] RYANG H, YUN U. High utility pattern mining over data streams with sliding window technique[J]. Expert Systems with Applications, 2016, 57: 214-231.

[13] KIM D, YUN U. Mining high utility itemsets based on the time decaying model[J]. Intelligent Data Analysis, 2016, 20(5): 1157-1180.

[14] KIM D, YUN U. Efficient algorithm for mining high average-utility itemsets in incremental transaction databases[J]. Applied Intelligence, 2017, 47(1): 114-131.

[15] DENG Z H. An efficient structure for fast mining high utility itemsets[J]. Applied Intelligence, 2018, 48(9): 3161-3177.

[16] DUONG Q H, LIAO B, FOURNIER-VIGER P, et al. An efficient algorithm for mining the top-*k* high utility itemsets, using novel threshold raising and pruning strategies[J]. Knowledge-Based Systems, 2016, 104: 106-122.

[17] DAM T L, LI K L, FOURNIER-VIGER P, et al. An efficient algorithm for mining top-*k* on-shelf high utility itemsets[J]. Knowledge and Information Systems, 2017, 52(3): 621-655.

[18] DAM T L, LI K L, FOURNIER-VIGER P, et al. CLS-Miner: efficient and effective closed high-utility itemset mining[J]. Frontiers of Computer Science, 2019, 13(2): 357-381.

[19] KANNIMUTHU S, PREMALATHA K. Discovery of high utility itemsets using genetic algorithm with ranked mutation[J]. Applied Artificial Intelligence, 2014, 28(4): 337-359.

[20] LIN J C W, YANG L, FOURNIER-VIGER P, et al. A binary PSO approach to mine high-utility itemsets[J]. Soft Computing, 2017, 21(17): 5103-5121.

[21] WU J M T, ZHAN J, LIN J C W. An ACO-based approach to mine high-utility itemsets[J]. Knowledge-Based Systems, 2017, 116: 102-113.

[22] BAIN M. Chess dataset[EB/OL]. [2022-05-14]. http://archive. ics. uci. edu/ml/datasets/Chess+%28King-Rook+vs. +King%29.

[23] SCHLIMMER J. Mushroom dataset[EB/OL]. [2022-05-14]. http://archive. ics. uci. edu/ml/datasets/Mushroom.

[24] VIGER P F. Foodmart[EB/OL]. [2022-05-14]. http://www. Philippe-fournier-viger. com/spmf/datasets/foodmart. txt.

[25] CHEN D. Online Retail dataset[EB/OL]. [2022-05-14]. http://archive. Ics. uci. edu/ml/datasets/Online+Retail.

编辑 陆燕菲