计算机科学
COMPUTER SCIENCE

具有周期间隙约束的负序列模式挖掘

王珠林, 武优西, 王月华, 刘靖宇

引用本文

王珠林, 武优西, 王月华, 刘靖宇.具有周期间隙约束的负序列模式挖掘[J]. 计算机科学, 2023, 50(3): 147-154.

WANG Zhulin, WU Youxi, WANG Yuehua, LIU Jingyu. Mining Negative Sequential Patterns with Periodic Gap Constraints [J]. Computer Science, 2023, 50(3): 147-154.

计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.211200248

具有周期间隙约束的负序列模式挖掘

王珠林[1] 武优西[1,2] 王月华[3] 刘靖宇[1,2]
1 河北工业大学人工智能与数据科学学院 天津 300401
2 河北省大数据计算重点实验室 天津 300401
3 河北工业大学经济管理学院 天津 300401
(1299056565@qq.com)

摘 要 间隙约束的序列模式挖掘是一种特殊形式的序列模式挖掘方法,该方法能够揭示一定间隔下的频繁出现(发生)的子序列。但当前间隙约束的序列模式挖掘方法只关注正序列模式的挖掘,忽略了事件中的缺失行为。为解决该问题,探索了周期间隙约束的负序列模式(Negative Sequential Pattern with Periodic Gap Constraints,NSPG)挖掘方法,该方法能够更灵活地反映元素与元素之间的关系。为高效求解 NSPG 挖掘问题,提出了 NSPG-INtree(Incomplete Nettrees)算法,该算法主要包括两个步骤:候选模式生成和支持度计算。在候选模式生成方面,为了减少候选模式的数量,该算法采用模式连接策略;在支持度计算方面,为了提高模式支持度计算效率并减少空间消耗,该算法采用不完整网树结构计算模式支持度。实验结果表明,NSPG-INtree 算法不仅具有较高的挖掘效率,而且能同时挖掘间隙约束的正序列模式和负序列模式。与其他间隙约束的序列模式挖掘算法相比,NSPG-INtree 能够多发现 209%~352%的模式;与不同策略的对比算法相比,NSPG-INtree 能够缩短 6%~38%的运行时间。

关键词:序列模式挖掘;负序列模式;频繁模式;间隙约束;不完整网树
中图法分类号 TP311

Mining Negative Sequential Patterns with Periodic Gap Constraints

WANG Zhulin[1],WU Youxi[1,2],WANG Yuehua[3] and LIU Jingyu[1,2]
1 School of Artificial Intelligence,Hebei University of Technology,Tianjin 300401,China
2 Hebei Key Laboratory of Big Data Computing,Tianjin 300401,China
3 School of Economics and Management,Hebei University of Technology,Tianjin 300401,China

Abstract Sequential pattern mining with gap constraints is a special form of sequential pattern mining,which can reveal frequent subsequences in a certain gap. However,the current sequential pattern mining methods with gap constraints only focus on positive sequential pattern mining,and ignore the missing behavior in a series of events. To solve this problem,a negative sequential pattern method with periodic gap constraints(NSPG) mining is explored,which can reflect the relationship between elements more flexibly. To solve the problem of NSPG mining,this paper proposes an NSPG-INtree(incomplete nettrees) algorithm,which includes two key steps:candidate pattern generation and support calculation. For candidate pattern generation,to reduce the number of candidate patterns,the algorithm uses a pattern join strategy. For support calculation,to improve the efficiency and reduce space consumption,the algorithm employs an incomplete nettree structure to calculate the supports of patterns. Experimental results show that NSPG-INtree not only has high mining efficiency,but also can mine positive and negative sequential patterns with gap constraints. NSPG-INtree can find 209% ~ 352% more patterns than other gap-constrained sequential pattern mining algorithms. Moreover,NSPG-INtree can reduce the running time by 6%~38% than other competitive algorithms with different strategies.

Keywords Sequential pattern mining,Negative sequential pattern,Frequent pattern,Gap constraint,Incomplete nettree

1 引言

序列模式挖掘(Sequential Pattern Mining,SPM)旨在从给定的序列数据库中寻找支持度大于给定阈值的所有频繁模式(子序列)。该方法在现实生活中具有广泛的应用[1-2],如生物信息挖掘[3]、时间序列分析[4]、居民出行特征研究[5]、顾客购买信息分析[6]等。传统的 SPM 大多只关注模式在一条序列中是否出现,未考虑该模式可能多次重复出现的情况,这

到稿日期:2021-12-23 返修日期:2022-06-21
基金项目:国家自然科学基金(61976240)
This work was supported by the National Natural Science Foundation of China(61976240).
通信作者:武优西(wuc567@163.com)

可能会导致一些重要的模式被忽略。例如，在传统的 SPM 中，模式 $P=xz$ 在序列 $S=xzzxzz$ 中出现了，因此记为出现一次，但实际上模式 P 在序列 S 中出现了多次。为了更全面地统计模式在序列中重复出现的次数，具有间隙约束(也称可变通配符)的 SPM 应运而生[7]。

具有间隙约束的 SPM 能够根据实际需求设定间隙范围，更具灵活性和针对性。具有间隙约束的序列模式可以表示成 $P=p_1[M_1,N_1]p_2\cdots[M_{j-1},N_{j-1}]p_j\cdots[M_{m-1},N_{m-1}]p_m$ 的形式[8]，其中 M_j 和 N_j 分别为元素 p_j 与 p_{j+1} 之间允许出现的最小和最大元素数量。例如，在模式 $x[0,1]y[0,2]z$ 中，间隙约束 $[0,1]$ 代表 x 和 y 之间可以包含 0 或 1 个元素。如果模式 P 中的间隙约束满足条件 $M_1=M_2=\cdots=M_m$ 以及 $N_1=N_2=\cdots=N_m$，那么我们称 P 为具有周期间隙约束的序列模式，例如模式 $x[0,1]y[0,1]z$ 即为一条具有周期间隙约束的序列模式。与传统的间隙约束相比，周期间隙更方便用户设定间隙值，而且能够提高挖掘算法的计算效率。

然而，当前具有周期间隙约束的 SPM 研究大多关注的是由频繁出现的元素(也称为正元素)构成的模式，即具有周期间隙约束的正序列模式(Positive Sequential Pattern with Periodic Gap Constraint，PSPG)，忽略了由序列中未出现的元素(也称为负元素)[9]构成的频繁模式，即负序列模式，导致难以发现潜在的重要信息。鉴于此，本文提出了具有周期间隙约束的负序列模式挖掘。与 PSPG 相比，NSPG 的限定条件更多，例如模式 $x[0,1](\neg y)z$ 是一条 NSPG，该模式在满足 $x[0,1]z$ 的基础上，还要求 x 和 z 的间隙内不能存在 y[10]。此外，NSPG 挖掘能同时挖掘具有周期间隙约束的正、负序列模式，因此与 PSPG 挖掘相比，NSPG 挖掘能捕获更多的信息。举例说明如下。

例 1 假定序列 $S=xzzxzz$ 为某车辆保险用户的活动记录，元素 x,y,z 分别代表用户向平台提交的车辆损坏、车辆维修和理赔活动。如果给定 PSPG 为 $P_1=x[0,1]z$，可以发现 P_1 在序列 S 中出现 4 次：$\langle 1,2\rangle$，$\langle 1,3\rangle$，$\langle 4,5\rangle$ 和 $\langle 4,6\rangle$。给定阈值 $\rho=4$，由于 P_1 的出现数大于阈值 ρ，因此 P_1 是频繁模式。如果此时给定 NSPG 为 $P_2=x[0,1](\neg y)z$，则可以发现 S 中的位置 1 和 2、1 和 3、4 和 5、4 和 6 之间都不存在元素 y，也就是说 P_2 在 S 中同样出现 4 次，即 P_2 也是频繁模式。

例 1 中，模式 P_1 说明该用户在登记车辆损坏后的一段时间内理赔了保险，这符合保险理赔的逻辑。但是通过分析模式 P_2 发现，该用户在车辆损坏后的一段时间内没有进行车辆维修就理赔了保险，这说明该用户存在保险欺诈或者夸大损失的可能。由此可见，NSPG 具有很高的研究与应用价值。

本文第 2 节讨论了相关工作；第 3 节给出了相关定义；第 4 节提出了本文算法并对其进行分析；第 5 节进行了实验验证；最后总结全文。

2 相关工作

间隙约束的 SPM 不仅难于求解[11]，而且存在多种形式。例如，文献[12]要求模式 P 中的间隙约束都必须为正，这样的间隙约束条件被称为非负间隙，而文献[13]则允许间隙约束为负，这样的间隙约束条件被称为一般间隙。例如 $x[0,1]z$ 是非负间隙，而 $x[0,-1]z$ 是一般间隙，本文研究非负间隙。为提高挖掘效率，文献[14]要求所有字符间的间隙约束都是相同的，这样的间隙约束条件被称为周期间隙约束，例如 $b[0,1]a[0,1]c$ 是周期间隙约束，而 $b[0,2]a[0,1]c$ 不是。本文专注于周期间隙约束。

与传统的 SPM[15]不同，间隙约束的 SPM 需要计算模式在一条序列中多次出现。而间隙约束计算出现数的方法可以分为 3 种类型，分别是无重叠[16-17]、一次性[18]、无条件[7]。无重叠要求同一位置的字符不能在出现的相同位置重复使用，假设 $I_1=\langle 1,2,4\rangle$ 和 $I_2=\langle 1,3,4\rangle$ 是两个出现，由于 I_1 和 I_2 的相同位置都使用了 1 和 4，因此它们不满足无重叠条件。一次性要求序列中的任何一个字符都只能出现一次，假设 $I_1=\langle 1,2,4\rangle$ 和 $I_2=\langle 4,5,6\rangle$ 是两个出现，由于 I_1 和 I_2 都用使用了 4，因此它们不满足一次性条件。无重叠和一次性条件对出现的约束相对严格，导致挖掘出的模式不够丰富，因此本文的研究不对出现进行约束，属于无条件约束[7]。

不同于无重叠和一次性条件，无条件的间隙约束的 SPM 不满足先验性。因此，为提高算法效率，文献[7]使用 Apriori-like 性质提出了 MAPB 算法，该算法能够有效挖掘周期间隙约束的序列模式。Wang 等[14]提出的基于索引树的 ITM 算法进一步提高了挖掘效率。而文献[19]中的方法通过重新定义偏移序列的方式使得周期间隙约束的 SPM 满足先验性。本文采用文献[19]中的定义，实现了高效的模式挖掘。

但是上述挖掘算法都只能挖掘出正序列模式，未考虑负元素的存在[20]，负序列模式挖掘的出现解决了这个问题。在负序列模式挖掘的早期算法中，Lin 等[21]提出只允许序列中的最后一个元素为负元素的 NSPM 算法；Ouang 等[22]也提出了挖掘模糊负序列模式的方法，能够挖掘出 $\langle\neg X,Y\rangle$，$\langle X,\neg Y\rangle$ 和 $\langle\neg X,\neg Y\rangle$ 3 种形式的负序列模式；Hsueh 等[23]提出了不限定负元素位置的算法 PNSP，并且为了提高算法效率给予了负序列模式约束条件；Zheng 等[10]提出了基于 GSP 算法来挖掘负序列模式的 NegGSP 算法。在上述算法中，Lin 等和 Ouyang 等提出的算法挖掘出的负序列模式都带有一定的局限性；而 PNSP 和 NegGSP 都需要先挖掘一遍正序列模式再挖掘负序列模式，导致这两种算法的时间性能不佳。

为了解决上述问题，Cao 等[24]和 Dong 等[25-26]提出了可以根据正序列模式的相关信息计算负序列模式的支持度的算法 e-NSP，以及基于自适应数据存储的算法 F-NSP+。这两种算法在求出频繁的正序列模式后不需要重新扫描数据库即可得出负序列模式的支持度，因此时间性能优于早期算法。然而，e-NSP 和 F-NSP+使用的集合论思想对负序列的约束过于严格，导致这两种算法挖掘出的负序列模式不够丰富，并且不适用于平均序列长度较长的大型数据集。此外，e-NSP 和 F-NSP+均未考虑间隙约束。因此，Guyet 等[9]提出了带间隙约束的负序列模式挖掘算法 NegPSpan，该算法在保持时间效率的同时适度降低了对负序列的约束，因此能够更灵活地挖掘负序列模式。然而，NegPSpan 未将模式在序列中重复的出现考虑在内，这可能会导致一些重要信息被遗漏。

3 问题定义

定义 1 长度为 l 的序列表示为 $S=d_1d_2\cdots d_l$，序列数据

库(Sequence DataBase,SDB)由 n 个序列构成,可以表示为 $SDB=\{S_1,S_2,\cdots,S_n\}$,SDB 中所有序列的长度之和即为 SDB 的长度,表示为 L。SDB 中不同字符的集合表示为 Σ,$|\Sigma|$代表 Σ 中字符的个数。

例 2　表 1 中,序列数据库 SDB 包含 S_1 和 S_2 两个序列,S_1 的长度 $l_1=6$,S_2 的长度 $l_2=8$,因此 SDB 的长度 $L=l_1+l_2=14$。该序列数据库的字符集 $\Sigma=\{x,y,z\}$,因此$|\Sigma|=3$。

表 1　序列数据库实例

Table 1　Sequence database example

Sid	Sequence
1	$xzzxzz$
2	$zxyzyxyy$

定义 2　给定元素 $e(e\in\Sigma)$,e 对应的负元素为$\neg e$,代表缺失元素 e。在序列 $S=d_1\cdots d_j\cdots d_k\cdots d_l$ 中,下标 j 为 d_j 在 S 中的位置,如果 S 的位置 j 和 k 之间不存在元素 e,那么 S 的元素 d_j 和 d_k 之间存在负元素$\neg e$。

例 3　给定序列 $S_1=d_1d_2d_3d_4d_5d_6=xzzxzz$,$\Sigma=\{x,y,z\}$,$d_4$ 和 d_6 之间的元素为 $d_5=z$,也就是说 d_4 和 d_6 之间不存在元素 x 和 y,换言之 d_4 和 d_6 之间存在负元素$\neg x$ 和$\neg y$。

定义 3　NSPG 表示为 $P=p_1[M,N](\neg e_1)p_2\cdots p_j[M,N](\neg e_j)p_{j+1}\cdots[M,N](\neg e_{m-1})p_m$,其中 p_j 是正元素,$p_j\in\Sigma$,模式中正元素的个数 m 称为 P 的长度。$\neg e_j$ 是负元素,$e_j\in\Sigma$ 或为空。当 $e_j\in\Sigma$ 时,表示在 p_{j-1} 和 p_j 之间的元素中不存在 e_j;当 e_j 为空时,p_j 和 p_{j+1} 之间的间隙可省略负元素$(\neg e_j)$,可表示为 $p_j[M,N]p_{j+1}$ 的形式。

例 4　$P=p_1[M,N](\neg e_1)p_2=x[0,1](\neg y)z$ 是一条 NSPG。P 中包含两个正元素 x 和 z,因此长度为 2。P 中的负元素$\neg e_1=\neg y$,表示 $p_1=x$ 和 $p_2=z$ 之间的元素不存在 y。

定义 4　如果位置序列 $I=\langle i_1,i_2,\cdots,i_m\rangle$中任意两个位置 i_{j-1}和 $i_j(2\leqslant j\leqslant l)$都满足 $M\leqslant i_j-i_{j-1}-1\leqslant N$,那么称 I 为长度为 m 的模式 P 在 S 中的一条偏移序列。P 在序列 S 中的所有偏移序列总数记为 $ofs(P,S)$,在 SDB 中所有序列的偏移序列总数之和记为 $ofs(P,SDB)$,即 $\sum_{i=1}^{n}ofs(P,S_i)$ $\sum_{i=1}^{n}ofs(P,S_i)$。

本文采用文献[19]的定义,长度为 m 的模式 P 在长度为 L 的序列数据库 SDB 上的偏移序列数可通过公式计算:$ofs(P,S)=L\times W^{m-1}$,其中 $W=N-M+1$。

例 5　给定模式 $P=x[0,1](\neg y)z$,序列数据库 SDB 如例 2。由于 P 的长度 $m=2$,因此 P 的偏移序列长度也为 2。已知位置序列 $I=\langle 4,6\rangle$的长度为 2,且位置 4 和 6 满足条件 $0\leqslant 6-4-1\leqslant 1$,因此 I 是 P 在 S_1 中的一条偏移序列。而 P 在 SDB 中的偏移序列数可通过公式计算:$ofs(P,SDB)=L\times W^{m-1}=14\times(1-0+1)^{2-1}=28$。

定义 5　$I=\langle i_1,i_2,\cdots,i_m\rangle$是 P 在 S 中的一条偏移序列,如果满足如下两个条件,则可认为 I 是 P 在 S 中的一条出现。

条件 1　$\forall i_j(1\leqslant j\leqslant m)$,都有$d_{i_j}=p_j$。

条件 2　$\forall i_j(1\leqslant j\leqslant m-1)$,都有$d_{i_j}$和$d_{i_{j+1}}$之间不存在$e_j$,如果 e_j为空,则无需判断条件 2。

例 6　给定 $P=p_1[M,N](\neg e_1)p_2=x[0,1](\neg y)z$,序列 $S_1=d_1d_2\cdots d_6=xzzxzz$,$I=\langle 4,6\rangle$为 P 在 S_1 中的一条偏移序列。由于 $d_4=p_1=x$,$d_6=p_2=z$,因此 I 满足条件 1。由于$\neg e_1$不为空,并且 S_1 的 d_4 和 d_6 之间不包括元素 $e_1=y$,因此 I 满足条件 2,故 I 是 P 在 S_1 中的一条出现。P 在 S_1 中所有的出现包括$\langle 1,2\rangle$,$\langle 1,3\rangle$,$\langle 4,5\rangle$,$\langle 4,6\rangle$。

定义 6　模式 P 在序列 S 中的支持度记为 $sup(P,S)$;在 SDB 中的支持度记为 $sup(P,SDB)$。$sup(P,SDB)$是模式 P 在各个序列的支持度总和。P 在 SDB 中的支持率用 $r(P,SDB)$表示,$r(P,SDB)=sup(P,SDB)/ofs(P,SDB)$,如果 $r(P,SDB)$大于等于给定的阈值 ρ,那么 P 就是一个频繁模式。

例 7　给定模式 $P=x[0,1](\neg y)z$,阈值 $\rho=0.13$,序列数据库 SDB 如例 2。根据例 6 可知 P 在 S_1 中的支持度 $sup(P,S_1)=4$,同理可以求出 P 在 S_2 中的支持度 $sup(P,S_2)=0$,因此 P 在 SDB 中的支持度 $sup(P,SDB)=4$,根据例 5 可知 P 在 SDB 中的偏移序列数 $ofs(P,SDB)=28$,故 P 的支持率$r(P,SDB)=4/28=0.143$。因为 $r(P,SDB)>\rho$,所以 P 是频繁模式。

定义 7(NSPG 挖掘)　问题的求解目标是在给定的序列数据库和间隙约束的情况下,挖掘出所有频繁的 PSPGs 和 NSPGs。

例 8　给定序列数据库 SDB 如例 2,间隙约束为$[0,1]$,阈值 $\rho=0.13$。经过 NSPG 挖掘,$x,y,z,x[0,1]z,x[0,1](\neg x)z,x[0,1](\neg y)z,z[0,1]x,z[0,1](\neg x)x$ 等模式的支持率大于阈值,因此这些模式为挖掘的频繁模式。

4　NSPG-INtree 算法

在挖掘正负序列模式时,主要面临着两个挑战:候选模式剪枝和支持度计算。为了有效地剪枝候选模式,本文采用模式连接策略生成候选模式,并在 4.1 节进行了详细的介绍。为了提高支持度计算的效率,第 4.2 节提出了基于不完整网树结构的 NegINtree 算法,该算法基于子模式的支持度来计算超模式的支持度,从而提高算法的挖掘效率。第 4.3 节进一步给出了能够同时挖掘 PSPGs 和 NSPGs 的 NSPG-INtree 算法。

4.1　候选模式生成

由于 NSPG 挖掘的特殊性,因此我们无法使用一种候选模式生成方法求出所有的 PSPGs 和 NSPGs。本文将在 4.1.1 节提出长度为 2 的候选模式生成方法,在 4.1.2 节提出长度为 $m+1(m\geqslant 2)$的候选模式生成方法。

4.1.1　长度为 2 的候选模式生成

在求出了全部长度为 1 的频繁模式后,采用两两拼接的方式生成长度为 2 的候选模式 $T=p_1[M,N]p_2$,此时模式 T 的间隙不包含负元素。采用模式匹配技术计算模式 T 的支持度。如果模式 T 为频繁模式,则在 p_1 和 p_2 中插入负元素$\neg e(e\in\Sigma)$生成新的候选模式 $T_1=p_1[M,N](\neg e)p_2$,并判断 T_1 是否为频繁模式。

例如,在例 8 中挖掘到长度为 1 的频繁模式,即 $x,y,z,$

将 x,y,z 两两拼接可得到长度为 2 的候选模式 $x[0,1]x$，$x[0,1]y$，$x[0,1]z$，$y[0,1]x$，$y[0,1]y$，$y[0,1]z$，$z[0,1]x$，$z[0,1]y$，$z[0,1]z$。经过计算得出长度为 2 的正序列频繁模式，即 $x[0,1]z$ 和 $z[0,1]x$。在频繁正序列模式的基础上产生负序列候选模式。以 $x[0,1]z$ 为例，在字符集为 3 的情况下，可以生成 3 个负序列候选模式，即 $x[0,1](\neg x)z$，$x[0,1](\neg y)z$ 和 $x[0,1](\neg z)z$。因此，$x[0,1]z$ 和 $z[0,1]x$ 共可以生成 $2\times 3=6$ 个负序列候选模式。经过计算得出 $x[0,1](\neg x)z$，$x[0,1](\neg y)z$ 和 $z[0,1](\neg x)x$ 也为长度为 2 的频繁模式。

4.1.2 长度为 $m+1$ 的候选模式生成

当候选模式长为 $m+1(m\geqslant 2)$ 时，我们使用模式连接的方式生成候选模式。

定义 8 给定超模式 $P=p_1[M,N](\neg e_1)p_2\cdots[M,N](\neg e_{m-2})p_{m-1}[M,N](\neg e_{m-1})p_m$，模式 P 的前缀 $Prefix(P)=p_1[M,N](\neg e_1)p_2\cdots[M,N](\neg e_{m-2})p_{m-1}$ 是去掉模式 P 的最后一个正元素 p_m 和负元素 $\neg e_{m-1}$ 的模式。模式 P 的后缀 $Suffix(P)=p_2\cdots[M,N](\neg e_{m-2})p_{m-1}[M,N](\neg e_{m-1})p_m$ 是去掉模式 P 的第一个正元素 p_1 和负元素 $\neg e_1$ 的模式。

定义 9 给定模式 $P=p_1[M,N](\neg e_1)p_2\cdots[M,N](\neg e_{m-2})p_{m-1}[M,N](\neg e_{m-1})p_m$ 和 $Q=q_1[M,N](\neg f_1)q_2\cdots[M,N](\neg f_{m-2})q_{m-1}[M,N](\neg f_{m-1})q_m$，如果模式 $R=Suffix(P)=Prefix(Q)$，则 P 和 Q 可生成超模式 $T=P\oplus Q=p_1[M,N]\neg e_1R[M,N]\neg f_{m-1}q_m$，此方式称为模式连接。其中，超模式 T 称为长度 $m+1$ 的超模式候选。

例 9 给定模式 $P=x[0,1](\neg y)z$，$Q=z[0,1](\neg x)x$，由于 $R=Suffix(P)=Prefix(Q)=z$，因此 P 和 Q 可生成超模式候选 $T=P\oplus Q=x[0,1](\neg y)z[0,1](\neg x)x$。

例 8 中长度为 2 的频繁模式包括：$x[0,1]z$，$x[0,1](\neg x)z$，$x[0,1](\neg y)z$，$z[0,1]x$，$z[0,1](\neg x)x$，使用模式连接策略将前缀和后缀进行拼接可以得到 12 个候选模式，即 $x[0,1]z[0,1]x$，$x[0,1]z[0,1](\neg x)x$，$x[0,1](\neg x)z[0,1]x$，$x[0,1](\neg x)z[0,1](\neg x)x$，$x[0,1](\neg y)z[0,1]x$，$x[0,1](\neg y)z[0,1](\neg x)x$，$z[0,1]x[0,1]z$，$z[0,1]x[0,1](\neg x)z$，$z[0,1]x[0,1](\neg y)z$，$z[0,1](\neg x)x[0,1]z$，$z[0,1](\neg x)x[0,1](\neg x)z$，$z[0,1](\neg x)x[0,1](\neg y)z$，只需对这 12 个候选计算支持度就可以得到所有长度为 3 的频繁模式。

如果使用深度优先或广度优先策略由频繁模式 P 枚举生成候选 $T=P[M,N]\neg f_{m-1}q_m$，则每个频繁模式都将生成 $|\Sigma|$ 个正候选模式。又由于每个正候选模式可以生成 $|\Sigma|$ 个负候选模式，因此可以产生 $|\Sigma|\times|\Sigma|$ 个负候选模式。因此，总计可以生成 $(|\Sigma|+1)\times|\Sigma|$ 个候选模式。例 8 中 $|\Sigma|=3$，因此例 8 中的每个频繁模式都将生成 $4\times 3=12$ 种候选。例 8 中共有 5 个长度为 2 的频繁模式，根据深度优先或广度优先策略这 5 个频繁模式将生成 $5\times 12=60$ 个长度为 3 的候选模式，进而需要计算支持度 60 次。相比之下，深度优先和广度优先生成的长度为 3 的候选模式数量是模式连接策略的 6 倍。由此可见，模式连接策略显著优于深度优先和广度优先策略。

4.2 支持度计算

本节提出了 NegINtree 算法来计算模式的支持度。在提出算法前，需要先引入不完整网树结构。

定义 10 $I=\langle i_1,i_2,\cdots,i_m\rangle$ 为模式 P 在序列 S 中的某一条出现，i_m 为 I 的最后一个位置，我们称 i_m 为模式 P 在序列 S 中的一个结束位置。

设模式 P 在序列 S 中共有 k 个出现，由于可能有多个出现在同一个位置 i_m 结束，因此 P 在 S 中的结束位置的个数小于等于 P 在 S 中的出现数。我们设 P 在 S 中共有 x 个结束位置($x\leqslant k$)。

定义 11 每个模式 P 在序列 S 中都存在一个不完整网树 INtree。INtree 共包含 x 个结点，每个结点 $INtree[j]$ 的 $name$ 值保存了 P 在 S 中的不同结束位置，$INtree[j].NRP$ 保存了以 $INtree[j].name$ 为结束位置的出现数。

例 10 给定模式 $P=x[0,1](\neg y)z$，序列 $S_1=xzzxzz$。根据例 6 可知 P 在 S_1 中的所有出现为 I_1，I_2，I_3 和 I_4，它们的结束位置分别为 2，3，5 和 6。根据 I_1，I_2，I_3 和 I_4 得出 P 的不完整网树 INtree 如图 1 所示，图中包括 INtree 的 4 个结点，每个结点的白色和灰色圆圈中的数字分别表示结点的 name 及其 NRP。例如，图 1 中第一个结点 $INtree[1].name=2$，由于只有 I_1 在位置 2 结束，因此 $INtree[1].NRP=1$。

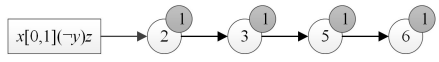


图 1 P 在 S_1 中的不完整网树 INtree

Fig. 1 Incomplete nettree INtree of P in S_1

模式 P 在序列 S 中的不完整网树中所有结点的 NRP 值之和即为 P 在 S 中的支持度。例如，在例 10 中 INtree 所有结点的 NRP 值之和为 4，表示模式 P 在 S_1 中的所有出现数。因此，求解出候选模式 P 在 S_1 中的不完整网树即可得出 P 的支持度。NegINtree 算法可根据前缀 $prefix(P)$ 的不完整网树 INtree1 求出模式 P 的不完整网树 INtree2。

定义 12 给定 $INtree1[j]$ 为 $prefix(P)$ 的不完整网树 $INtree1$ 中的任意结点，d_k 为 S 中的任意元素，根据定义 5，如果 $INtree1[j].name$ 和 d_k 满足如下条件，则认为 $INtree1[j]$ 和 d_k 匹配。

条件 1 $M\leqslant k\text{-}INtree1[j].name-1\leqslant N$($k$ 为 d_k 的下标)。

条件 2 如果 P 中的最后一个负元素 $\neg e_{m-1}$ 不为空，则 S 的位置 $INtree1[j].name$ 和 k 之间不存在元素 e_{m-1}。

条件 3 $d_k=p_m$。

定义 13 如果 $INtree1[j]$ 和 d_k 匹配，则在 $INtree2$ 中插入结点 $INtree2[l]$，$INtree2[l].name=k$，$INtree2[l].NRP=INtree1[j].NRP$；假如 $INtree2$ 中已存在结点 $INtree2[l]$ 满足 $INtree2[l].name=k$，则更新 $INtree2[l]$ 的 NRP 值为：$INtree2[l].NRP=INtree2[l].NRP+INtree1[j].NRP$。

根据定义 12 和定义 13，我们可以通过扫描 INtree1 和序列 S 的方式建立 P 的不完整网树 INtree2，进而得出 P 在 S 中的支持度。本文通过例 11 进行了举例说明。

例 11 给定模式 $P=p_1[M,N](\neg e_1)p_2[M,N](\neg e_2)p_3=x[0,1](\neg y)z[0,1](\neg x)x$，序列 $S_1=d_1d_2\cdots d_6=xzzxzz$。$P$ 的前缀 $prefix(P)=x[0,1](\neg y)z$，$prefix(P)$ 的不完整网树 INtree1 如图 2 中的第一层所示。

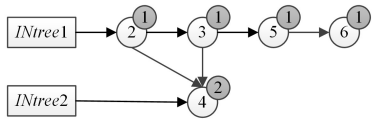


图 2 不完整网树 INtree1 和 INtree2

Fig. 2 Incomplete nettrees INtree1 and INtree2

读取 $INtree1[1]$，在序列 S 中寻找与 $INtree1[1]$ 匹配的元素，发现 $INtree1[1]$ 和 d_4 同时满足条件 1、条件 2 和条件 3。对于条件 1，$INtree1[1].name=2$，d_4 的下标为 4，2 和 4 满足条件 $0\leqslant 4-2-1\leqslant 1$；对于条件 2，$S$ 的位置 2 和 4 之间未出现 P 的最后一个负元素 $\neg x$ 对应的正元素 x；对于条件 3，$d_4=p_3=x$。因此，$INtree1[1]$ 和 d_4 匹配，在 $INtree2$ 中插入或更新结点。由于此时 $INtree2$ 为空，因此在 $INtree2$ 中插入结点 $INtree2[1]$，$INtree2[1].name=4$，$INtree2[1].NRP=INtree1[1].NRP=1$。

读取 $INtree1[2]$，发现 $INtree1[2]$ 和 d_4 同时满足条件 1、条件 2 和条件 3。因此，$INtree1[2]$ 和 d_4 匹配，在 $INtree2$ 中插入或更新结点。由于 $INtree2$ 中已经存在结点 $INtree2[1]$ 满足 $INtree2[1].name=4$，因此更新 $INtree2[1]$ 的 NRP 值为：$INtree2[1].NRP=INtree2[1].NRP+INtree1[2].NRP=2$。

依次读取 $INtree1[3]$ 和 $INtree1[4]$，发现 S 中没有元素与 $INtree1[3]$ 和 $INtree1[4]$ 匹配。至此，$INtree1$ 中的结点全部读取完毕，模式 P 在序列 S 中的不完整网树 $INtree2$ 建立完毕，如图 2 中的第二层所示。P 在 S 中的支持度为 $INtree2$ 中所有结点的 NRP 值之和，即 $sup=INtree2[1].NRP=2$。

NegINtree 算法如算法 1 所示。

算法 1 NegINtree

输入：模式 P，序列 S，间隙[M，N]，不完整网树 INtree1

输出：P 在 S 中的支持度 sup

```
1. for each node INtree1[j] in INtree1 do
2.    for k=INtree1[j].name+M+1 to INtree1[j].name+N+1 do
3.    ifI Ntree1[j]和 dk匹配 then
4.          根据定义 13 在 INtree2 中插入或更新结点；
5.       end if
6.    end for
7. end for
8. sup=INtree2 中结点的 NRP 值之和；
9. return sup.
```

4.3 NSPG-INtree 算法

挖掘 NSPG 的 NSPG-INtree 算法的步骤如下：

Step1 扫描数据库 SDB，得到所有长度为 1 的频繁模式并将其放入 F_1 中。按照 4.1.1 节的方法求出所有长度为 2 的频繁模式并将其放入 F_2 中。根据 F_2 生成长度为 3 的候选集 C。

Step2 当 C 不为空时，从 C 中依次取出长度为 k（$k>1$）的候选模式 P，计算出 P 在 SDB 中的支持率，如果 P 的支持率大于 ρ 则将 P 放入 F_k 中。

Step3 采用模式连接策略，根据 F_k 生成长度为 $k+1$ 的候选集 C 并重复 Step2，直到 C 为空，算法结束。

算法 2 NSPG-INtree

输入：序列数据库 SDB，阈值 ρ，间隙约束[M，N]

输出：频繁模式集 F

```
1.  遍历 SDB 并将所有频繁的元素存储在集合 F1中；
2.  用 4.1.1 节的方法求出所有长度为 2 的频繁模式，并将其存储在
    集合 F2中；
3. C←patterngrowth(F2)；// patterngrowth 指 4.2.2 节的模式连接
   方法
4. m←3；
5. while C〈〉null do
6.    offsup←L×W^(m-1)；
7.    for each P in C do
8.       sup←0；
9.       for each Skin SDB do
10.      sup←sup+NegINtree(P,Sk,[M,N],Prefix(P)->INtree)；
11.   end for
12.   if(sup/offsup>ρ) then Fm←Fm∪P；
13. end for
14. C←patterngrowth(Fm)；
15. m++；
16. end while
17. return F.
```

例 12 给定序列数据库 SDB 如表 1 所列，间隙约束为[0，1]，阈值 $\rho=0.13$。根据例 2 可知 SDB 的长度 $L=14$，字符集 $\Sigma=\{x,y,z\}$。扫描数据库获得模式 x，y 和 z 的不完整网树和支持度，x，y 和 z 的支持度分别为 4，4 和 6，根据定义 4 可以计算出 x 偏移序列数为 $L\times W^{m-1}=14\times(1-0+1)^{1-1}=14$，同理，$y$，$z$ 的偏移序列数也为 14，因此 x，y 和 z 的支持率分别为 $4/14=0.286$，$4/14=0.286$ 和 $6/14=0.429$。由于 x，y 和 z 的支持率都大于阈值 0.13，因此 x，y，z 均为频繁模式。按照 4.1.1 节的方法生成长度为 2 的候选，即 $x[0,1]x$，$x[0,1]y\cdots z[0,1](\neg z)x$，将长度为 2 的候选带入到 NegINtree 算法中计算它们的不完整网树和支持度，得出它们的支持度分别为 0，3 … 3，根据公式计算出它们的偏移序列数为 $L\times W^{m-1}=14\times(1-0+1)^{2-1}=28$，因此它们的支持率分别为 0，0.107，…，0.107。由于长度为 2 的候选中 $x[0,1]z$，$x[0,1](\neg x)z$，$x[0,1](\neg y)z$，$z[0,1]x$，$z[0,1](\neg x)x$ 的支持率大于阈值 0.13，因此它们是频繁的模式。如例 9 所示，采用模式连接策略生成 12 个长度为 3 的候选模式 $x[0,1]z[0,1]x$，$x[0,1]z[0,1](\neg x)x$，…，$z[0,1](\neg x)x[0,1](\neg y)z$，带入到 NegINtree 算法中计算它们的不完整网树和支持度，其中 $x[0,1](\neg y)z[0,1](\neg x)x$ 的支持度计算过程如例 11 所示，进一步计算长度为 3 的候选模式的支持率。由于没有长度为 3 的候选模式的支持率大于阈值 ρ，因此 NSPG-INtree 算法结束。

定理 1 本文的 NSPG 挖掘满足先验性

证明：给定模式 $T=P[M,N]\neg f_{m-1}q_m$，其中 P 为 T 的前缀，设 $Q=P[M,N]p_m$ 是仅比 P 多出一个正元素的模式。$I=\langle i_1,i_2,\cdots,i_{m-1}\rangle$ 为 P 在 S 中的任意一条出现，那么设 $I'=\langle i_1,i_2,\cdots,i_{m-1},i_m\rangle$ 是 Q 在 S 中以 I 为前缀的一条出现，多出的索引 i_m 最多出现 W 次，分别是 $i_{m-1}+M+1$ 至 $i_{m-1}+N+$

1，因此I'最多存在W种情况，进而推导出$sup(Q,S)\leqslant sup(P,S)\times W$。而$T$相比$Q$多出负元素$\neg f_{m-1}$，这代表着当$\neg f_{m-1}$不为空时，还要求$T$比$Q$多满足负包含的约束。鉴于此，$sup(T,S)\leqslant sup(Q,S)$，因此有$sup(T,S)\leqslant sup(P,S)\times W$。

在每个序列S中都有$sup(T,SDB)\leqslant sup(P,SDB)\times W$，则有$\sum_{i=1}^{n} sup(T,S_i)\leqslant \sum_{i=1}^{n} sup(P,S_i)\times W$，因此$sup(T,SDB)\leqslant sup(P,SDB)\times W$。

由定义4知$ofs(T,SDB)=LW^{m-1}$，$ofs(P,SDB)=LW^{m-2}$，故$ofs(T,SDB)=W\times ofs(P,SDB)$。又因为$sup(T,SDB)\leqslant sup(P,SDB)\times W$，所以得出$r(T,SDB)=sup(T,SDB)/ofs(T,SDB)\leqslant sup(P,SDB)/ofs(P,SDB)=r(P,SDB)$，超模式的支持率小于其前缀的支持率。同理可证超模式的支持率小于其后缀的支持率。因此，当超模式的前缀或后缀不是频繁模式时，该超模式一定不是频繁模式。证毕。

定理2 NSPG-INtree算法的时间复杂度为$O(D\times L\times W/|\Sigma|)$，其中参数$D$为不同长度候选的总数，$L$为SDB的长度，$|\Sigma|$为字符集$\Sigma$的大小，$W=N-M+1$。

证明：首先，NegINtree算法需要遍历前缀模式在序列S中的不完整网树INtree。由于SDB中的每个序列S的平均长度为L/n，因此每个不完整网树平均包含$L/n/|\Sigma|$个结点。其次，设$INtree[j]$为INtree中的某个结点，NegINtree需要对S中与位置$INtree[j].name$满足间隙约束$[M,N]$的元素进行判断，一共判断W次，因此NegINtree算法的时间复杂度是$O(L/n/|\Sigma|\times W)$。由于需要对D个候选进行计算，且数据库中的序列个数为n，因此NSPG-INtree算法的时间复杂度为$O(D\times L\times W/|\Sigma|)$。证毕。

定理3 NSPG-INtree算法的空间复杂度为$O(G\times L/|\Sigma|)$，其中G为频繁模式总数。

证明：由于频繁模式所占空间远远小于不完整网树所占空间的大小，因此本文在分析空间复杂度时主要考虑不完整网树的大小。共挖掘出G个频繁模式，每个频繁模式在序列数据库中包含n个不完整网树，每个不完整网树平均包括$L/n/|\Sigma|$个结点，因此NSPG-INtree算法的空间复杂度为$O(G\times L/|\Sigma|)$。证毕。

5 实验

实验运行的环境为：Intel(R)Core(TM)i7-8750CPU处理器，主频2.20GHz，内存8GB，Windows10，64位操作系统的计算机，程序开发环境为Visual Studio 2015。

5.1 基准数据集

表2列出了本文实验所用到的8个长度和内容各不相同的数据集，即数据集DNA1，DNA2，DNA3，DNA4，DNA5和DNA6[1)]，以及数据集COV2和SARS[2),3)]。

表2 基准数据集摘要

Table 2 Summary of benchmark datasets

Dataset	From	Length
DNA1	Homo Sapiens AL158070	6000
DNA2	Homo Sapiens AL158070	8000
DNA3	Homo Sapiens AL158070	10000
DNA4	Homo Sapiens AL158070	12000
DNA5	Homo Sapiens AL158070	14000
DNA6	Homo Sapiens AL158070	16000
COV2	Coronavirus 2(COVID-19)	29903
SARS	Coronavirus(SARS)	29751

5.2 基本方法

(1)NAMEIC-pro算法[27]。PSPG挖掘算法，该算法采用文献[27]中的NAMEIC算法计算模式的支持度，通过模式连接策略生成候选模式。为了比较NSPG挖掘和PSPG挖掘的区别，我们采用NAMEIC-pro作为对比算法。

(2)MAPB算法[7]。PSPG挖掘算法，为了比较NSPG挖掘和PSPG挖掘的区别，我们采用MAPB作为对比算法。

(3) NSPG-BFS算法。为了验证4.1节所采用的模式连接策略的效果，我们提出了NSPG-BFS算法，该算法基于广度优先策略的枚举方式生成候选模式，采用NegINtree算法计算支持度。

(4) NSPG-Nettree算法。为了验证4.2节所采用的NegINtree方法的计算效率，我们提出了NSPG-Nettree算法，该算法使用网树结构[7]计算模式的支持度，采用模式连接策略生成候选模式。

5.3 算法性能对比

为了验证NSPG-INtree的挖掘性能，我们将NAMEIC-pro，MAPB，NSPG-BFS和NSPG-Nettree作为对比算法，在数据集DNA1，DNA2，DNA3，DNA4，DNA5和DNA6上进行实验。阈值参数$\rho=0.02$，间隙约束参数为$[0,3]$，频繁模式数量、运行时间及候选模式数量如图3—图5所示。

(1)与传统算法NAMEIC-pro和MAPB相比，虽然NSPG-INtree的时间效率相对较低，但是能发现更多有意义的模式。例如，在图3的DNA1中，NAMEIC-pro和MAPB算法只挖掘出了33个频繁模式，而NSPG-INtree挖掘出了102个频繁模式。其他数据集上也呈现出类似的现象。可以看出，NSPG-INtree比NAMEIC-pro和MAPB多发现了209%～352%的模式。这是因为NAMEIC-pro和MAPB只能挖掘出PSPG，而NSPG-INtree不仅能够挖掘出PSPG，还能挖掘出NSPG，其挖掘结果包含了NAMEIC-pro和MAPB的挖掘结果。

(2)NSPG-INtree的性能优于NSPG-BF，因为在挖掘出相同数量的频繁模式的前提下，NSPG-INtree的挖掘速度比NSPG-BFS快。造成此现象的原因是NSPG-INtree算法采用模式连接策略生成候选，比时间复杂度为$O(D\times L\times W\times|\Sigma|)$的NSPG-BF算法更有效。例如在图3—图5中，NSPG-BFS算法在数据集DNA2上的运行时间为43.30s，而NSPG-INtree算法的运行时间为29.17s。可以发现，在所有数据集

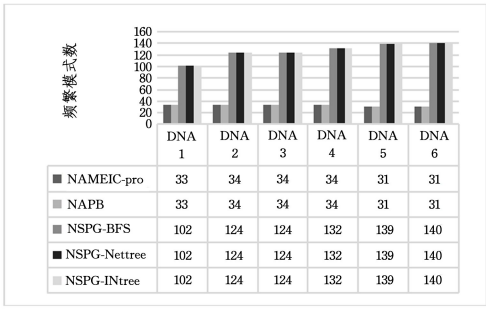
1) https://www.ncbi.nlm.nih.gov/nuccore/AL158070.11

2) https://www.ncbi.nlm.nih.gov/nuccore/MN908947

3) https://www.ncbi.nlm.nih.gov/nuccore/30271926

上的实验，NSPG-INtree 比 NSPG-BFS 节约了 26%～38%的时间。原因在于，NSPG-INtree 仅需要计算 1 724 个候选模式，而 NSPG-BFS 需要计算 3100 个候选模式，这表明 NSPG-BFS 算法需要计算更多冗余的候选模式，导致时间开销更多。造成这种现象的原因是 NSPG-INtree 采用模式连接策略生成候选模式，避免了冗余的候选生成，这与第 4.1 节中的分析是一致的。

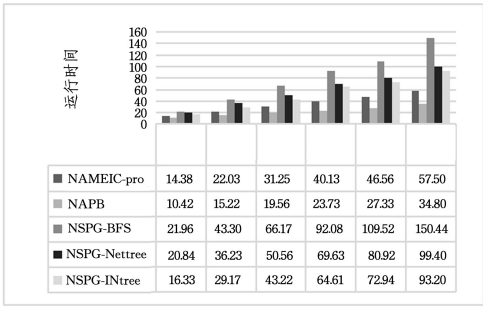
（3）NSPG-INtree 的性能优于 NSPG-Nettree，因为 NSPG-INtree 的挖掘速度比 NSPG-Nettree 快。造成此现象的原因是 NSPG-INtree 算法采用 NegINtree 计算模式支持度，比时间复杂度为 $O(D\times L\times W\times m)$ 的 NSPG-INtree 算法更有效。例如在图 3－图 5 中，NSPG-Nettree 在数据集 DNA3 中的运行时间为 50.56 s，而 NSPG-INtree 算法的运行时间为 43.22 s。可以看出，在其他数据集上也呈现出类似的现象，NSPG-INtree 的挖掘速度比 NSPG-Nettree 快了 6%～22%。原因在于，NSPG-INtree 采用不完整网树计算模式支持度，可以基于子模式的结果计算超模式的支持度，避免了冗余的计算过程，从而提高了算法的挖掘效率，这与第 4.2 节中的分析是一致的。



	DNA 1	DNA 2	DNA 3	DNA 4	DNA 5	DNA 6
NAMEIC-pro	33	34	34	34	31	31
NAPB	33	34	34	34	31	31
NSPG-BFS	102	124	124	132	139	140
NSPG-Nettree	102	124	124	132	139	140
NSPG-INtree	102	124	124	132	139	140

图 3 频繁模式数量对比

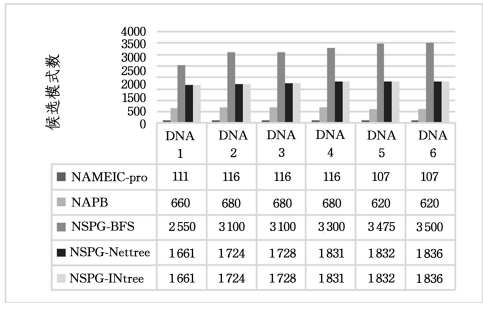
Fig. 3 Comparison of number of frequent pattern



NAMEIC-pro	14.38	22.03	31.25	40.13	46.56	57.50
NAPB	10.42	15.22	19.56	23.73	27.33	34.80
NSPG-BFS	21.96	43.30	66.17	92.08	109.52	150.44
NSPG-Nettree	20.84	36.23	50.56	69.63	80.92	99.40
NSPG-INtree	16.33	29.17	43.22	64.61	72.94	93.20

图 4 运行时间对比

Fig. 4 Comparison of running time



	DNA 1	DNA 2	DNA 3	DNA 4	DNA 5	DNA 6
NAMEIC-pro	111	116	116	116	107	107
NAPB	660	680	680	680	620	620
NSPG-BFS	2550	3100	3100	3300	3475	3500
NSPG-Nettree	1661	1724	1728	1831	1832	1836
NSPG-INtree	1661	1724	1728	1831	1832	1836

图 5 候选模式数量对比

Fig. 5 Comparison of number of candidate pattern

5.4 实例分析

肆虐全球的新冠病毒（COV2）与我们所熟知的 SARS 病毒同属于冠状病毒，还是官方认可的“近亲”。因此很多研究从直观的病毒序列分析两者的相似性，但是这些研究忽略了两者在缺失基因上的相似性。因此，本节使用 NSPG-INtree 同时从 PSPGs 和 NSPGs 的角度对病毒序列 SARS 和 COV2 进行研究。实验中，间隙约束设置为[0，15]，阈值 ρ 设置为 0.015，实验结果如表 3 所列，其中 union 为两者相同的挖掘结果。由于本次实验所挖掘出的模式长度大多小于等于 4，因此本次的实验结果只能分析两者基础结构的相似性。

表 3 COV2 和 SARS 的相似性研究

Table 3 Study on the similarity of COV2 and SARS

	COV2	SARS	union
PSPGs	31	31	31
NSPGs	34	31	31

从表 3 可以发现，在 COV2 和 SARS 中挖掘出的 PSPGs 完全相同，因此我们可以认为，COVID-19 和 SARS 的基本结构具有极高的相似性。如果同时挖掘 NSPGs，我们还可以发现 COV2 和 SARS 基因基础结构上的缺失也具有很高的相似性，基因缺失的相似性表明它们可能具有相似的遗传缺陷。通过研究 COV2 和 SARS 中共同缺失的基因，我们能够更深刻地了解冠状病毒的特性与缺点，为人类战胜冠状病毒提供参考。

结束语 为了挖掘满足间隙约束的负序列模式，本文提出了 NSPG 挖掘。为了高效求解该问题，本文提出了一种有效的算法 NSPG-INtree，该算法能够在判断两个正元素的间隙是否满足间隙约束的同时判断其中是否包含负元素，得出某个 NSPG 在序列中的支持度，进而判断其是否为频繁模式。在候选模式生成方面，本文采用模式连接策略有效地减少了候选模式数量，降低了时间和空间开销。在支持度计算方面，本文通过使用不完整网树结构计算模式的支持度，提高了算法的挖掘效率。为了验证算法的正确性和有效性，本文在真实的 DNA 数据集上进行了对比实验。实验结果表明，与传统的 PSPG 挖掘算法 NAMEIC-pro 和 MAPB 相比，NSPG-INtree 能够挖掘出更多有意义的频繁模式；与不同策略的对比算法相比，采用多种策略的 NSPG-INtree 算法可以有效提高算法效率。

参 考 文 献

[1] ZHANG G L，YANG Q H，CHENG X M，et al. Application of sequence pattern mining in communication network alarm prediction[J]. Computer Science，2018，45(11A)：535-538.

[2] LIU X B，WANG L Z，ZHOU L H. MLCPM-UC：A Multi-level co-location pattern mining algorithm based on uniform coefficient of pattern instance distribution [J]. Computer Science，2021，48(11)：208-218.

[3] WU Y X，ZHU C R，LI Y，et al. NetNCSP：Nonoverlapping closed sequential pattern mining [J]. Knowledge-Based Systems，2020，196：105812.

[4] WU Y X，LUO L F，LI Y，et al. NTP-Miner：Nonoverlapping three-way sequential pattern mining [J]. ACM Transactions on

Knowledge Discovery from Data, 2022, 16(3): 1-21.
[5] WU C F, CAI L, LI J, et al. Frequent pattern mining of resident's travel based on multi-source location data [J]. Computer Science, 2021, 48(7): 155-163.
[6] HUANGT C. Mining the change of customer behavior in fuzzy time-interval sequential patterns [J]. Applied Soft Computing, 2012, 12(3): 1068-1086.
[7] WU Y X, WANG L L, REN J D, et al. Mining sequential patterns with periodic wildcard gaps [J]. Applied Intelligence, 2014, 41(1): 99-116.
[8] WU Y X, LIU X, YAN W J, et al. Efficient algorithm for solving non-overlapping strict sequential pattern matching [J]. Journal of Software, 2021, 32(11): 3331-3350.
[9] GUYET T, QUINIOU R. NegPSpan: Efficient extraction of negative sequential patterns with embedding constraints [J]. Data Mining and Knowledge Discovery, 2020, 34(3): 563-609.
[10] ZHENG Z G, ZHAO Y C, ZUO Z Y, et al. Negative-GSP: An efficient method for mining negative sequential patterns [C]// Eighth Australasian Data Mining Conference. 2009: 63-67.
[11] DING B, LO D, HAN J W, et al. Efficient mining of closed repetitive gapped subsequences from a sequence database [C]// IEEE International Conference on Data Engineering. 2009: 1024-1035.
[12] WU Y X, TONG Y, ZHU X Q, et al. NOSEP: Nonoverlapping sequence pattern mining with gap constraints [J]. IEEE Transactions on Cybernetics, 2018, 48(10): 2809-2822.
[13] SHI Q S, SHAN J S, YAN W J, et al. NetNPG: Nonoverlapping pattern matching with general gap constraints [J]. Applied Intelligence, 2020, 50(1): 1832-1845.
[14] WANG L, WANG S, LIU S L, et al. An algorithm of mining sequential pattern with wildcards based on Index-Tree [J]. Chinese Journal of Computers, 2019, 42(3): 555-565.
[15] ZHOU Z Y, PI D C. Rare sequence pattern mining algorithm for shopping basket data [J]. Journal of Chinese Computer Systems, 2019, 40(3): 683-688.
[16] WU Y X, GENG M, LI Y, et al. HANP-Miner: High average utility nonoverlapping sequential pattern mining [J]. Knowledge-Based Systems, 2021, 229, 107361.
[17] WU Y X, WANG Y H, LI Y, et al. Top-k self-adaptive contrast sequential pattern mining [J]. IEEE Transactions on Cybernetics, 2022, 52(11): 11819-11833..
[18] WU Y X, LEI R, LI Y, et al. HAOP-Miner: Self-adaptive high-average utility one-off sequential pattern mining [J]. Expert Systems with Applications, 2021, 184, 115449.
[19] WU Y X, ZHOU K, LIU J Y, et al. Mining sequential patterns with periodic general gap constraints [J]. Chinese Journal of Computers, 2017, 40(6): 1338-1352.
[20] HUANG G, GAN W, HUANG S, et al. Negative pattern discovery with individual support [J]. Knowledge-Based Systems, 2022, 251, 109194.
[21] LIN N P, CHEN H J, HAO W H. Mining negative sequential patterns [C]// Conference on Applied Computer Science. 2007: 654-658.
[22] OUYANG W M, HUANG Q H, LUO S H. Mining positive and negative fuzzy sequential patterns in large transaction databases [C]// Fifth International Conference on Fuzzy Systems and Knowledge Discovery. 2008: 18-23.
[23] HSUEH S C, LIN M Y, CHEN C L. Mining negative sequential patterns for e-commerce recommendations [C]// IEEE Asia-Pacific Services Computing Conference. 2008: 1213-1218.
[24] CAO L B, D X J, ZHENG Z G. E-NSP: Efficient negative sequential pattern mining [J]. Artificial Intelligence, 2016, 235(1): 156-182.
[25] DONG X J, GONG Y S, CAO L B. F-NSP +: A fast negative sequential patterns mining method with self-adaptive data storage [J]. Pattern Recognition, 2018, 84: 13-27.
[26] DONG X J, GONG Y S, CAO L B. e-RNSP: An efficient method for mining repetition negative sequential patterns [J]. IEEE Transactions on Cybernetics, 2018, 50(5): 2084-2096.
[27] WU Y X, WU X D, MIN F, et al. A Nettree for Pattern Matching with Flexible Wildcard Constraints [C]// IEEE International Conference on Information Reuse & Integration. 2010: 109-114.

WANG Zhulin, born in 1997, postgraduate, is a member of China Computer Federation. His main research interests include data mining and so on.

WU Youxi, born in 1974, Ph.D, professor, Ph.D supervisor, is a distinguished member of China Computer Federation. His main research interests include data mining and machine learning.

(责任编辑:杨雪敏)