arXiv:2301.02182v1 [cs.DB] 4 Jan 2023

基于综合规则进行流程发现的排序策略比较

Tsung-Hao Huang 和 Wil M. P. van der Aalst

流程与数据科学（PADS），亚琛工业大学，德国亚琛 {tsunghao.huang, wvdaalst}@pads.rwth-aachen.de

摘要。 流程发现旨在从信息系统中观察到的行为（即事件日志）中学习流程模型。发现的模型可作为流程挖掘技术的起点，用于解决性能和合规问题。与最先进的归纳式挖掘器相比，应用自由选择网理论中的综合规则的算法能够发现结构更加灵活（非块状）的流程模型，同时确保模型具有相同的理想健全性和自由选择性。此外，这一研究方向的最新进展表明，发现的模型具有相当的质量。按照综合规则，该算法通过每次添加事件日志中的一个活动，逐步修改现有流程模型。由于规则的应用高度依赖于现有模型的结构，添加活动的顺序会显著影响模型质量和计算时间。本文使用真实世界的事件数据，研究不同排序策略对发现模型（就适应度和精确度而言）及计算时间的影响。结果表明，与单纯依据活动频率的排序策略相比，所提出的排序策略能够提升最终流程模型的质量，同时耗时更少。

关键词： 流程发现 · 综合规则 · 排序策略。

1 引言

流程挖掘是一门连接流程科学与数据科学的学科， [2], 提供了分析流程执行过程中生成的事件数据（即事件日志）的技术和工具。流程挖掘技术所产生的分析结果，为利益相关者提供了宝贵的数据驱动洞见。

流程发现是流程挖掘的三大主要研究领域之一，另外两个领域是符合性检查和流程增强。流程发现技术旨在从事件数据中学习端到端流程模型。借助发现的模型，知识工作者可以应用其他流程挖掘技术，进一步获取优化洞见。

虽然已有各种算法被提出，但只有少数算法能确保健全性和自由选择性等理想属性。一方面，健全性保证了：(1) 流程始终可以完成 (2)

流程可以正确完成 (3)，且模型中不存在无法执行的变迁 [1]。另一方面，自由选择性将流程模型（佩特里网）中的选择构造与同步构造分离开来。这一属性很理想，因为它可以轻松地将发现的模型转换为 BPMN 等广泛使用的表示法 [3]。此外，自由选择网还可使用大量基于相关理论开发的分析技术 [5]。

最先进的技术，例如归纳式挖掘器（IM） [9] 系列，能够发现保证健全且自由选择的流程模型。IM 通过利用其内部流程表示——流程树——来提供此类保证。然而，这种表示也可能是一把双刃剑。由于表示偏差，IM 发现的模型注定是块结构化的，即模型必须由具有单一入口和出口的部分组成 [9]。这意味着 IM 只能发现健全的自由选择工作流网的一个子集。

为了在保持相同保证的同时提供更灵活的流程表示，我们在 [8] 中提出了一种新颖的发现算法，即所谓的 Synthesis Miner。Synthesis Miner 利用自由选择网理论中的综合规则 [5]。通过预定义模式，将事件日志中的活动逐步添加到正在构建的模型中。遵循这些规则可确保发现的流程模型始终健全且自由选择。此外，研究表明，与归纳式挖掘器发现的模型相比，Synthesis Miner 发现的模型质量相当。然而，综合规则的可能应用在很大程度上取决于现有模型结构。以不同顺序添加活动可能会产生不同的模型。因此，一个尚待研究的问题是：向现有模型添加活动的顺序会如何影响最终流程模型的质量。本文通过比较 Synthesis Miner 的排序策略，并深入考察活动添加顺序对模型质量和计算时间的影响，来探讨这一问题。使用四个公开可用的真实事件日志进行的实验表明，先进的排序策略能够显著提升模型质量并缩短计算时间。

本文其余部分安排如下。第 2 节介绍相关工作。第 3 节介绍全文所用的必要符号和概念。随后，第 4 节介绍所提出的排序策略。第 5 节展示使用公开的真实事件日志进行的评估。最后，第 6 节对本文进行总结。

2 相关工作

关于流程挖掘的一般介绍，请参阅 [2]。此外， [4] 对流程发现领域的近期进展进行了综述和基准评测。本文重点关注流程发现技术，这类技术会对正在构建的模型进行增量修改，以推导出最终流程。

增量式流程挖掘允许用户通过将不同的轨迹逐步整合到现有模型中，从事件日志中学习流程模型 [14]。由于排序策略会对模型质量产生重大影响，因此开展了一项研究 [13] ，以

探究两者之间的相互作用。不过，算法迭代添加的是轨迹，而非活动。因此，这与本文的相关性较低。

Dixit 等人 [6] 是最早使用自由选择网理论中的综合规则 [5] 来发现流程模型的研究者之一。受此启发， [6], [8] 提出了 SynthesisMiner，通过引入预定义模式和搜索空间剪枝机制，实现了发现过程的自动化。 [6]和 [8]都为各自的方法提出了几种排序策略。然而，排序的选择被留给用户作为输入参数。排序策略对模型质量和计算时间的影响尚未得到充分研究。此外，排序策略与搜索空间剪枝之间的相互作用也尚未得到解释。最后但同样重要的是，有必要对不同的排序策略进行比较。本文旨在解决这一尚未解答的研究问题，并为用户提供一条经验法则。

3 预备知识

本节将介绍本文中通篇使用的必要概念和符号。

对于任意集合 A，我们将所有可能的序列组成的集合记为 A^*，将 A 上所有多重集组成的集合记为 $\mathcal{B}(A)$。给定 σ_1， $\sigma_2 \in A^*$， $\sigma_1 \cdot \sigma_2$ 表示这两个序列的连接。设 A 为一个集合， $X \subseteq A$ 为 A 的子集。对于 $\sigma \in A^*$ 和 $a \in A$，我们递归地定义 $\restriction_X \in A^* \to X^*$ 为一个投影函数，其中若 $a \in X$，则 $\langle a \rangle \cdot \sigma$ 的 $\langle\rangle\restriction_X = \langle\rangle$；若 $a \notin X$，则 $\langle a \rangle \cdot \sigma$ 的 $\restriction_X = \sigma\restriction_X$。例如， $\langle x, y, x\rangle\restriction_{\{x,z\}} = \langle x, x\rangle$。投影函数也可应用于序列的多重集。例如，
$[\langle x, y, x\rangle^4, \langle x, y\rangle^2, \langle y, x, z\rangle^6]\restriction_{\{y,z\}} = [\langle y\rangle^6, \langle y, z\rangle^6]$。我们将 $\mathcal{U}_A$ 记为活动标签全集。

定义 1（轨迹与日志）。轨迹 $\sigma \in \mathcal{U}_A^*$ 是活动标签的序列。日志是轨迹的多重集，即 $L \in \mathcal{B}(\mathcal{U}_A^*)$。

定义 2（日志属性 [8]）。设 $L \in \mathcal{B}(\mathcal{U}_A^*)$ 以及 a、$b \in \mathcal{U}_A$ 为两个活动标签。我们定义以下日志属性：

– $\#(a, L) = \Sigma_{\sigma \in L} |\{i \in \{1, 2, ..., |\sigma|\} | \sigma(i) = a\}|$ 是 L 中 a 出现的次数。– $\#(a, b, L) = \Sigma_{\sigma \in L} |\{i \in \{1, 2, ..., |\sigma| - 1\} | \sigma(i) = a \wedge \sigma(i+1) = b\}|$ 是 L 中从 a 到 b 的直接后继关系的数量。– $caus(a, b, L) = \begin{cases} \frac{\#(a,b,L) - \#(b,a,L)}{\#(a,b,L) + \#(b,a,L) + 1} \text{ if } a \neq b \\ \frac{\#(a,b,L)}{\#(a,b,L) + 1} \text{ if } a = b \end{cases}$ 是因果关系 (a, b) 的强度。– $A_c^{pre}(a, L) = \{a_{pre} \in \mathcal{U}_A | caus(a_{pre}, a, L) \geq c\}$ 是由阈值 c 确定的 a 的前序活动集合。– $A_c^{fol}(a, L) = \{a_{fol} \in \mathcal{U}_A | caus(a, a_{fol}, L) \geq c\}$ 是由阈值 c 确定的 a 的后续活动集合。

定义 3（佩特里网）。设 $N=(P,T,F,l)$ 为一个佩特里网，其中 P 是库所集合，T 是变迁集合，$P\cap T=\emptyset$。$F\subseteq(P\times T)\cup(T\times P)$ 是弧的集合，$l\in T\to\mathcal{U}_A\cup\{\tau\}$ 是一个将活动标签分配给变迁的标记函数。如果 $l(t)=\tau$，则变迁 $t\in T$ 不可见（或静默）。

定义 4（路径与初等路径）。佩特里网 $N=(P,T,F)$的路径是一个非空节点序列 $\rho=\langle x_1,x_2,...,x_n\rangle$，满足 $(x_i,x_{i+1})\in F$对于 $1\leq i<n$。如果 $x_i\neq x_j$ 对于 $1\leq i<j\leq n$，则 ρ 是初等路径。对于$X,X'\in P\cup T$，$elemPaths(X,X',N)\subseteq(P\cup T)^*$是从某个 $x\in X$ 到某个 $x'\in X'$的所有初等路径的集合。

定义 5（工作流网（WF-net）[1]）。设 $N=(P,T,F,l)$ 是一个佩特里网。$W=(P,T,F,l,i,o,\top,\perp)$ 是一个 *WF-net*，当且仅当：*(1)* 它有一个专用源库所 $i\in P$：$\bullet i=\emptyset$，以及一个专用汇库所 $o\in P$：$o\bullet=\emptyset$；*(2)* $\top\in T$：$\bullet\top=\{i\}\wedge i\bullet=\{\top\}$ 和 $\perp\in T$：$\perp\bullet=\{o\}\wedge\bullet o=\{\perp\}$ *(3)* 每个节点 x 都位于某条从 i到 o的路径上，即 $\forall_{x\in P\cup T}(i,x)\in F^*\wedge(x,o)\in F^*$，其中 F^*是 F的自反传递闭包。

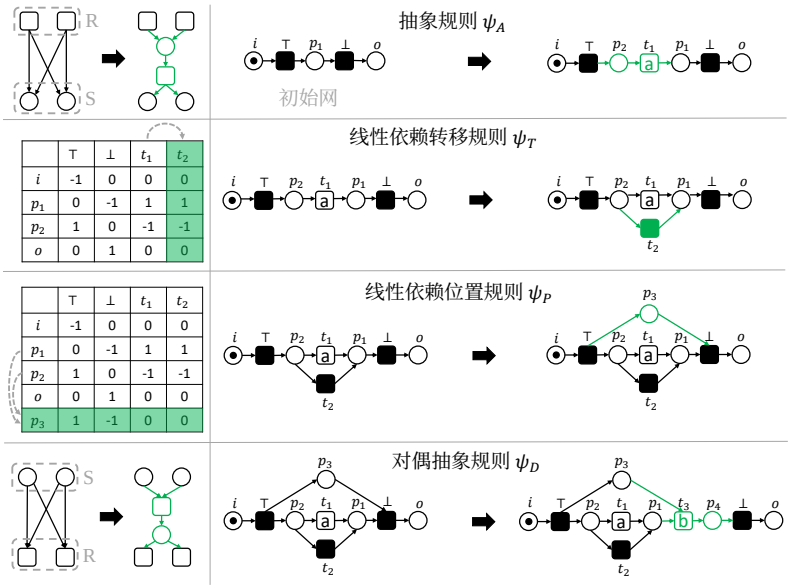
定义 6（活动顺序）。设 $L\in\mathcal{B}(\mathcal{U}_A^*)$ 和 $A=\bigcup_{\sigma\in L}\{a\in\sigma\}$。$\gamma\in A^*$ 是 L 的活动顺序，当且仅当 $\{a\in\gamma\}=A$ 且 $|\gamma|=|A|$。

Synthesis Miner：使用综合规则进行流程发现在先前的工作 [8], 中，我们介绍了 Synthesis Miner，它通过应用 [5] 中定义的综合规则以及额外的对偶抽象规则 [8]，保证能够发现健全的自由选择工作流网。

给定一个工作流网 W，抽象规则（ψ_A）允许在一组变迁 $R\subseteq T$ 和一组库所$S\subseteq P$之间添加一个库所p 和一个变迁 t，前提是它们完全连接，即（$R\times S\subseteq F$）$\wedge$（$R\times S\neq\emptyset$）。线性变迁/库所规则（ψ_T/ψ_P）允许添加一个变迁 t/库所 p，前提是它在相应的关联矩阵中与其他变迁/库所线性相关。对偶抽象规则（ψ_D）可在一组库所 S和一组变迁 R之间添加一个变迁 t 和一个库所 p，前提是（$S\times R\subseteq F$）$\wedge$（$S\times R\neq\emptyset$）。这四条规则[1] 均能保持健全性和自由选择性质 [5,8]。图 1 展示了规则应用的几个示例。

给定日志 L，Synthesis Miner 首先确定活动顺序 γ。然后启动迭代。在第 i 次迭代（其中 $1\leq i\leq|\gamma|$）中，将活动 γ（i）添加到来自第 $i-1$ 次迭代的现有网 [2] 中。每次迭代的步骤如下：(1) 使用投影日志 $L_i=L{\restriction}_{\{\gamma(1),\gamma(2),...\gamma(i)\}}$ 中的启发式信息，在现有工作流网（W_i）上找出最可能添加待加入活动 γ（i）的位置；(2) 应用预定义模式（由综合规则推导而来），获得候选网集合；(3) 从候选项集合中选出适应度和精确度最佳的网，用于下一次迭代。

[1] 有关这些规则的正式定义，请参阅 [5,8]。[2] 第一次迭代中的现有网以初始网为起点，如图 1 中抽象规则的示例所示。



	⊤	⊥	t_1	t_2
i	-1	0	0	0
p_1	0	-1	1	1
p_2	1	0	-1	-1
o	0	1	0	0

	⊤	⊥	t_1	t_2
i	-1	0	0	0
p_1	0	-1	1	1
p_2	1	0	-1	-1
o	0	1	0	0
p_3	1	-1	0	0

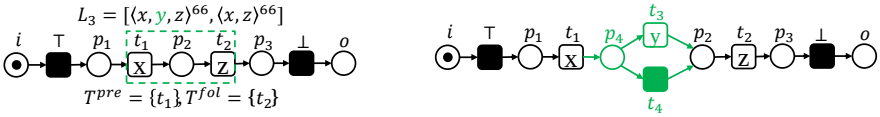
图 1： 综合规则应用示例。ψ_A 允许通过 $R = \{\top\}$ 和 $S = \{p_1\}$添加 p_2 和 t_1。t_2 由 ψ_T 添加，因为它与 t_1线性相关。p_3 由 ψ_P添加，因为它是 p_1 和 p_2的线性组合。ψ_D 允许通过 $S = \{p_1, p_3\}$ 和 $R = \{\bot\}$添加 t_3 和 p_4。

由于步骤 (1) 直接受到排序策略的影响，我们正式定义 [3] 如何利用日志启发式将搜索空间限制为工作流网上节点的一个子集。

定义 7（缩减搜索空间）。设 $a \in \mathcal{U}_A^*$ 为一个活动，$L \in \mathcal{B}(\mathcal{U}_A^*)$ 为一个日志，$W = (P, T, F, l, i, o, \top, \bot)$ 为一个工作流网，且 $0 \leq c \leq 1$。T^{pre} 是日志 L中由 a 的前序活动标记的变迁集合。若 $A_c^{pre}(a, L) \neq \emptyset$，则 $T^{pre} = \{t \in T | l(t) \in A_c^{pre}(a, L)\}$；否则为 $T^{pre} = \{\top\}$。T^{fol} 是日志 L中由 a 的后续活动标记的变迁集合。若 $A_c^{fol}(a, L) \neq \emptyset$，则 $T^{fol} = \{t \in T | l(t) \in A_c^{fol}(a, L)\}$；否则为 $T^{fol} = \{\bot\}$。缩减后的搜索空间为 $\mathbf{reduce}(a, L, W, c) = \{x \in \rho | \rho \in \mathbf{elemPath}(T^{pre}, T^{fol}, W)\}$。

函数 $reduce$ 首先查找待添加活动 $\gamma(i)$ 的前置和后续活动，以及对应的带标签变迁集。然后，返回位于前置变迁与后续变迁之间路径上的节点集，记为 V_i。V_i 用于限制综合规则的应用范围。更确切地说，用作应用规则 ψ_A 和 ψ_D 的前提条件的变迁集 R 和库所集 S 必须是 V_i的子集，即 $S \subseteq V \wedge R \subseteq V$。至于规则 ψ_T/ψ_P，新变迁/库所（t'/p'）不能与 V_i以外的任何节点通过弧相连。此步骤有助于将搜索空间限制在工作流网中最有可能添加活动 $\gamma(i)$ 的节点上。图 2 展示了一个缩减搜索空间的示例。

[3] 由于步骤（2）和（3）的形式化定义不在本文讨论范围内，请参阅 [8]。



(a) W_2，上一轮迭代中的现有网　　**(b)** W_3，添加 y后的网

图 2 $L_3 = [\langle x, y, z\rangle^{66}, \langle x, z\rangle^{66}]$ y W_2 $c = 0.9$

展示如何缩减搜索空间的示例。考虑日志 。 是我们想要添加到网 中的活动。使用 ， 我们得到 $T^{pre} = \{t_1\}$ 和 $T^{fol} = \{t_2\}$。因此，函数 $reduce$ 将返回 t_1 和 t_2之间的节点集，也就是如 (a) 中绿色虚线所示的 $V_3 = \{t_1, p_2, t_2\}$ 。随后，综合规则的应用只会考虑这三个节点。最后，从候选项中选出最佳网 W_3 ， 并在 (b) 中展示。

4 排序策略

本节将介绍不同的排序策略。为了说明排序策略，请考虑以下日志 $L_s = [\langle b, c, d, e, f, g\rangle, \langle b, e, c, d, f, g\rangle, \langle b, e, c, f, g, d\rangle, \langle b, c, e, d, f, g\rangle, \langle b, c, e, f, d, g\rangle, \langle b, c, e, f, g, d\rangle, \langle e, b, c, d, f, g\rangle, \langle e, b, c, f, d, g\rangle]$.
相应的直接跟随图（DFG）如图 3 所示。

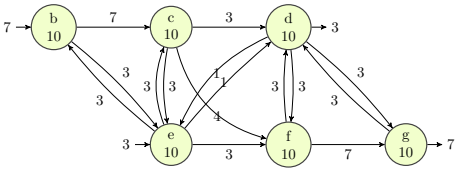


图 3: 日志 L_s 的直接跟随图。

第一种排序策略基于频率，且相对简单。只需按照活动在日志中出现的频率对活动进行排序。

定义 8（基于频率的排序）。令 $L \in \mathcal{B}(\mathcal{U}_A^*)$。基于频率的排序函数为 $order^{freq}(L) = \gamma$ ，满足 γ 是活动顺序，且 $\forall_{1 \le i < j \le |\gamma|} \#(\gamma(i)\ L) \ge \#(\gamma(j)\ L)$。

如果活动的频率相同，则按字母顺序排列。以示例事件日志 L_s 为例，顺序为 $order^{freq}(L_s) = \langle b, c, d, e, f, g\rangle$。

其他排序策略更为复杂，因为它们不仅考虑活动频率，还考虑活动之间的连接。在介绍其他排序策略之前，我们先定义一个辅助函数，根据连接强度对直接后继活动进行排序。

定义 9（直接后继活动排序）。令 $L \in \mathcal{B}(\mathcal{U}_A^*)$ 和 $a \in \mathcal{U}_A . A = \{b \in \mathcal{U}_A | \#(a, b, L) > 0\}$ 为在 L 中至少出现一次直接后继于 a 的活动集合，且 $\sigma \in A^*$。直接后继活动排序是 $sortDFA(a, L) = \sigma$ ，满足 $\{b \in \sigma\} = A$ 且 $|\sigma| = |A|$ ，并且 $\forall_{1 \le i < j \le |\sigma|} \#(a, \sigma(i)\ L) \ge \#(a, \sigma(j)\ L)$。

例如，$sortDFA(b,\ L_s) = \langle c, e\rangle$。这是因为活动 c 和 e 的入弧来自 b，且强度 $\#(b,\ c,\ L_s) \ge \#(b,\ e,\ L_s)$。定义了用于排序直接后继活动的函数后，我们现在可以在算法1中定义基于广度优先搜索的排序策略。

算法 1： 基于广度优先搜索的排序，排序 $_{BFS}$

输入 ：一个日志 $L \in \mathcal{B}(\mathcal{U}_A^*)$
输出： 一个活动顺序 γ 适用于 L
$A \leftarrow \bigcup_{\sigma \in L} \{a \in \sigma\}$；// L 中活动的集合
$A^s \leftarrow \{\sigma(1) \mid \sigma \in L \wedge \sigma \neq \langle\rangle\}$；// L 中起始活动的集合
$\sigma \leftarrow order_{freq}(L)\restriction_{A^s}$；// 按频率排序的起始活动序列
$i \leftarrow 1$;
当 $|\sigma| \neq |A|$ **时：**
  $A' \leftarrow A \setminus \{a \in \sigma\}$；// 不在 σ 中的活动集合
  $\sigma' \leftarrow sortDFA\ (\sigma\ (i)\quad L)\restriction_{A'}$；// 对 $\sigma(i)$ 的后续活动进行排序，并投影到 A'
  $\sigma \leftarrow \sigma \cdot \sigma'$；// 更新 σ
  $i \leftarrow i + 1$;
$\gamma \leftarrow \sigma$;
return γ;

基于广度优先搜索的排序策略首先构建日志中的起始活动序列，然后使用定义 9 中的函数，迭代地追加直接后继活动序列。将该函数应用于示例事件日志 L_s，得到排序 $_{BFS}(L_s) = \langle b, e\rangle \cdot \langle c\rangle \cdot \langle f, d\rangle \cdot \langle\rangle \cdot \langle g\rangle = \langle b, e, c, f, d, g\rangle$。$\sigma$ 以 $\langle b, e\rangle$初始化。然后，在第 i次迭代中，将 $\sigma(i)$的直接后继活动按$sortDFA(\sigma(i)\ L_s)$排序后组成的序列追加到 σ 中，并过滤掉已在σ 中的活动。循环持续进行，直到 σ 包含日志中的所有活动。顾名思义，该排序优先探索直接后继活动。

接下来，我们在算法 2 中介绍另一种基于深度优先搜索的排序策略。与基于广度优先搜索的排序策略一样，该策略也会考虑活动之间的连接关系，但基于深度优先搜索的排序优先考虑深度而非广度。也就是说，只有在探索完深度更高的活动后，才会充分探索其直接后继活动。将基于深度优先搜索的排序应用于日志 L_s，可得顺序 $_{DFS}(L_s) = \langle b, c, f, g, d, e\rangle$。

请注意，尽管我们将基于广度优先搜索和基于深度优先搜索的排序策略定义为从起始活动开始，但也可以从另一个方向开始探索，即从结束活动开始，随后探索直接前驱活动以进行排序。以 L_s 为例，如果从结束活动集合开始，使用基于深度优先搜索的排序处理日志 L_s，将得到 $\langle g, f, c, b, e, d\rangle$；使用基于广度优先搜索的排序则将得到 $\langle g, d, f, c, e, b\rangle$。

为说明不同排序策略如何影响流程发现的进展，图 4 展示了使用三种不同排序策略将 Synthesis Miner 应用于日志 L_s 时生成的所有中间网。基于深度优先搜索的排序倾向于在开始时先从开始构建流程直至结束，然后再添加并行／选择分支中的活动。相反，基于广度优先搜索的排序优先构建局部控制流。例如，从迭代 1 到迭代 2 的差异就很明显。虽然所有排序策略在迭代 1 中生成的网相同，但基于广度优先搜索的排序建议在迭代 2 中为 b 添加并发活动 e，而基于深度优先搜索的排序则先添加

算法 2： 基于深度优先搜索的排序排序 $_{DFS}$

输入 ：一个日志 $L \in \mathcal{B}(\mathcal{U}_A^*)$
输出： L的活动顺序 γ
$A \leftarrow \bigcup_{\sigma \in L} \{a \in \sigma\}$; // L 中活动的集合
$A^s \leftarrow \{\sigma(1) \mid \sigma \in L \wedge |\sigma| \neq 0\}$; // L 中起始活动的集合
$\sigma^s \leftarrow order_{freq}(L)\restriction_{A^s}$; // 按频率排序的起始活动序列
$\sigma \leftarrow \langle \sigma^s(1) \rangle$; // 使用出现频率最高的开始活动来初始化序列
$\sigma^s \leftarrow \sigma^s \restriction_{\{A^s \setminus \{\sigma^s(1)\}\}}$; // 将 σ^s 更新为栈
当 $|\sigma| \neq |A|$ **：**
  $A' \leftarrow A \setminus \{a \in \sigma\}$; // 不在 σ 中的活动集合
  $\sigma^f \leftarrow sortDFA(\sigma(|\sigma|)\ L)\restriction_{A'}$; // 对 $\sigma(|\sigma|)$ 的后续活动进行排序
  if $|\sigma^f| = 0$ **：**
    $\sigma \leftarrow \sigma \cdot \langle \sigma^s(1) \rangle$; // 将栈中的第 1 个元素追加到 σ σ^s
  否则：
    $\sigma \leftarrow \sigma \cdot \langle \sigma^f(1) \rangle$; // 将 σ^f 中的第 1 个元素追加到 σ
  $\sigma^s \leftarrow (\sigma^f \restriction_{A \setminus \{a \in \sigma \vee a \in \sigma^s\}}) \cdot (\sigma^s \restriction_{A \setminus \{a \in \sigma\}})$; // 更新栈 σ^s
$\gamma \leftarrow \sigma$;
返回 γ;

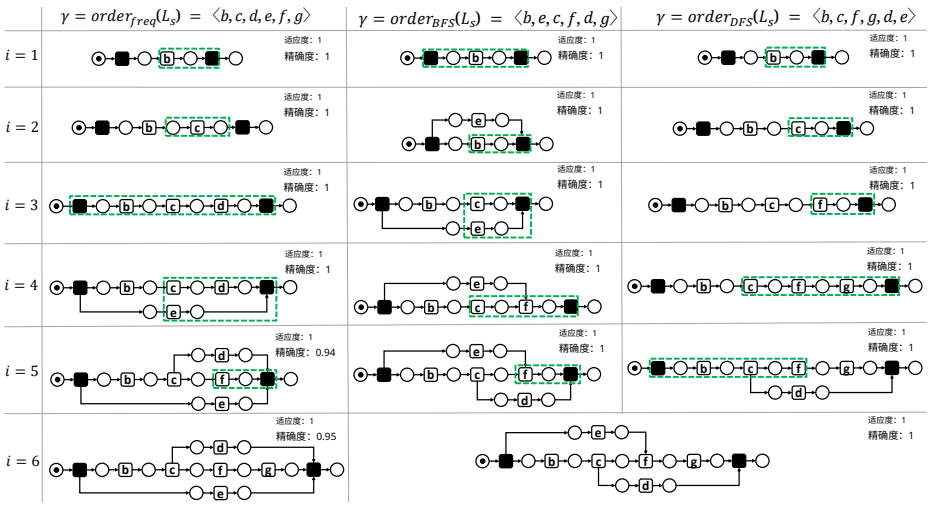


图 4： 针对日志 L_s 对不同排序策略进行的比较。每一列代表一种排序策略，每一行对应于在第 i 次迭代中添加 γ（i）后的中间工作流网。绿色虚线标出了代表缩减搜索空间的节点。适应度和精确度指标使用相应的投影日志 $L_i = L\restriction_{\{\gamma(1),\gamma(2),\ldots\gamma(i)\}}$ 进行测量。请注意，在此示例中，由广度优先搜索和基于深度优先搜索的排序策略发现的最终模型相同。

b 的直接后继活动 c 。频率排序似乎没有呈现出明确的发现模式。

我们预计，排序方式的选择会显著影响发现所需的计算时间。主要差异在于检查线性依赖规则可行性所需的时间。

随着工作流网不断扩展，检查候选库所/变迁是否存在线性依赖所需的时间成本也会随之增加。因此，最好尽可能缩减搜索空间，尤其是在后续迭代中。请回顾，缩减后的搜索空间（定义 7）是一个节点集，用于限定综合规则的应用范围。图 4 中的绿色虚线标出了第 i 次迭代的缩减搜索空间 V_i。如图 4 所示，通常，基于广度优先搜索的排序能够比其他策略保持更小的搜索空间，因为它优先处理相连活动。相反，基于深度优先搜索的排序在后续迭代中更容易产生较大的搜索空间。由于并行/可选活动是在较后阶段添加的，待添加活动 $\gamma(i)$ 的前序和后续活动极有可能分散在现有网中。再加上搜索空间缩减的影响，最终形成相对较大的搜索空间，这意味着需要考虑更多节点。图 4 中基于深度优先搜索的排序在第 4 次和第 5 次迭代中的示例可见一斑。

尽管我们假设基于广度优先搜索的排序具有相对较低的计算时间，但搜索空间缩减可能会在最优解与时间之间引入权衡。在下文中，我们旨在通过实验探究排序策略对模型质量以及发现流程模型所需时间的影响。

5 评估

本节将介绍用于评估排序策略的实验，包括实验设置以及对结果的讨论[4]。

5.1 实验设置

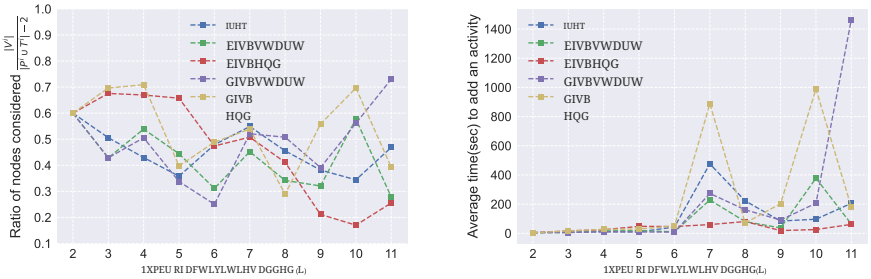
实验中，我们使用了四个公开可用的真实事件日志 [7,10,11,12]。我们对这些日志进行过滤，聚焦于主流行为（至少占轨迹的 95%），并使用出现频率最高的轨迹变体。对于 BPI2017 日志 [7],，我们根据活动前缀（A、W、O）将其拆分为三个日志。因此，总计得到六个日志。

对于每个事件日志，我们都为 Synthesis Miner [8] 应用不同的排序策略，而其他参数采用默认值。对于基于 BFS 和 DFS 的排序策略，我们分别从两个方向（开始活动和结束活动）应用排序。因此，我们评估了五种排序策略。为衡量排序策略对搜索空间剪枝的影响，我们跟踪缩减搜索空间的比例。该比例通过 $\frac{|V_i|}{|P_i \cup T_i| - 2}$ 评估，其中 V_i 是缩减后的节点集合，~~P_i~~ 和 T_i 是现有 WF 网 W_i 中的库所集合和变迁集合。分母中的 -2 用于排除两个库所（源库所和汇库所），根据定义 5，这两个库所无法通过新增节点连接。以图 4 为例，在迭代 3 中，频率排序策略的 $\frac{|V_3|}{|P_3 \cup T_3| - 2}$ 值为 $\frac{9}{11-2} = 1$。这表明，在应用综合规则以添加下一个活动时，所有可能的节点都已纳入考量。此外，我们还从适应度、精确度和 F1分数（适应度与精确度的调和平均数）方面评估最终模型。

[4] https://github.com/tsunghao-huang/synthesisRulesMiner

5.2 结果与讨论

搜索空间缩减与计算时间 图 5 展示了五种排序策略在搜索空间缩减方面的效果对比。y 轴上的数值 $\frac{|V_i|}{|P_i \cup T_i|-2}$ 是六个事件日志的平均值。如图所示，该指标跟踪添加下一个活动时缩减后的搜索空间比例，该比例反映了可能的综合规则应用数量。总体而言，我们可以从图中观察到，



(a) 搜索空间缩减的平均比例 **(b)** 添加一个活动所需的平均时间

图 5： 比较各排序策略在缩减搜索空间方面的效果，以及每个步骤所需的计算时间。请注意，$\frac{|V_i|}{|P_i \cup T_i|-2}$ 的值越低越好。

排序策略的表现符合预期。如图 5a 所示，在发现的后期阶段（$i \geq 8$），广度优先搜索排序策略（bfs_start、bfs_end）将搜索空间缩减的比例维持在较低水平，而频率排序和基于深度优先搜索的排序策略则表明，它们更倾向于纳入搜索空间中的大量节点。

图 5b 展示了六个日志中每一步向现有工作流网添加一个活动所需的平均时间。对比两张图，可以看出搜索空间缩减对计算时间的影响。如图 5b 所示，bfs_end 策略使每一步的平均计算时间保持在相当低的水平。bfs_start 策略也有类似表现，尽管在添加第 7 个和第 10 个活动时出现了两个峰值。对于 dfs_end 策略，第 7 步和第 10 步的两个峰值尤为显著。两次添加单个活动到现有模型都耗时超过 10 分钟。此外，添加活动所需的最长时间也出现在 dfs_start 策略的第 11 步。

简而言之，由于与搜索空间缩减相互作用，基于广度优先搜索的排序策略在计算时间方面具有显著优势。

模型质量 表 1[5] 展示了使用五种不同排序策略时的模型质量结果。正如预期，在全部六个事件日志中，我们观察到基于广度优先搜索的排序策略计算时间最短。这

[5] 为提供当前最先进水平的参考，我们还展示了 IMf 的结果（以灰色标示）。IMf 生成的最佳模型（按 F1 分数衡量）是从一组网中选出的，这些网使用了过滤器的五种不同取值（$[0.1, 0.2, 0.3, 0.4, 0.5]$）。

表1： 不同排序策略发现的模型质量。

Log	排序策略& IMf	适应度	精确度		F1	时间（秒）
BPI2017A	频率	0.971	0.947		0.958	685
	BFS_开始	0.973	1.000		0.986	893
	BFS_end	0.990	0.935		0.961	**334**
	DFS_开始	0.963	0.868		0.913	1850
	DFS_end	0.999	0.986		**0.993**	1248
	IMf(0.2)	0.999	0.936		0.967	10
BPI2017O	频率	0.993	0.962		0.978	537
	BFS_开始	0.985	0.963		0.974	**165**
	BFS_end	0.989	1.000		0.995	231
	DFS_开始	0.996	1.000		**0.998**	498
	DFS_end	0.993	0.962		0.978	360
	IMf(0.2)	0.997	0.907		0.950	7
BPI2017W	频率	0.993	0.726		0.838	3617
	BFS_开始	0.974	0.864		0.914	1626
	BFS_end	0.993	0.888		0.936	**579**
	DFS_开始	0.974	0.864		0.914	1732
	DFS_end	0.993	0.901		**0.944**	5397
	IMf(0.2)	0.923	0.897		0.910	14
服务台	频率	0.974	0.984		0.978	51
	BFS_开始	0.974	0.984		0.978	52
	BFS_end	0.983	0.976		**0.979**	**43**
	DFS_开始	0.974	0.984		0.978	49
	DFS_end	0.989	0.963		0.976	64
	IMf(0.2)	0.967	0.950		0.958	1
医院计费	频率	0.945	0.810		0.879	509
	BFS_开始	0.931	0.922		0.936	**314**
	BFS_end	0.988	0.935		**0.961**	383
	DFS_开始	0.931	0.970		0.961	2154
	DFS_end	0.943	0.883		0.920	2359
	IMf(0.2)	0.982	0.906		0.943	45
流量	频率	0.967	0.930		0.945	274
	BFS_开始	0.967	0.930		0.945	**202**
	BFS_end	0.972	0.720		0.825	388
	DFS_开始	0.991	0.933		**0.960**	366
	DFS_end	0.942	0.858		0.903	443
	IMf(0.4)	0.904	0.720		0.801	28

与上一节的发现相符。此外，尽管搜索空间大幅缩小，但在六个日志中，有两个日志使用广度优先搜索排序策略发现的模型取得了最高的 F1分数。

至于基于深度优先搜索的排序策略，其计算时间方面存在明显劣势，但在其他四个事件日志中取得了最高的 F1 分数。这一结果符合我们的假设，因为搜索空间缩减会在最优解与时间之间带来权衡。最后，频率排序策略在模型质量和计算时间方面均无显著优势。结果表明，考虑活动之间连接关系的排序策略比基于频率的排序策略更能提升 Synthesis Miner 的表现。

6 结论

本文介绍了五种使用综合规则的流程发现算法排序策略 [8]。我们研究了排序策略对模型质量和计算时间的影响。结果表明，与仅依据活动频率的排序策略相比，所提出的排序策略考虑了活动之间的连接关系（基于广度优先搜索和基于深度优先搜索），在时间和模型质量方面分别表现更优。

结果显示，所提出的基于广度优先搜索的排序策略能够加快计算速度。尽管 Synthesis Miner 能够发现质量更高的模型，但其整体发现时间仍无法与当前最先进的方法相媲美。因此，未来我们计划进一步利用日志启发式，并研究更复杂的排序策略，以加快 Synthesis Miner 的运行速度。另一个改进方向是提高其处理低频行为的能力，因为我们使用频率最高的轨迹变体来捕捉主流流程。若能为 Synthesis Miner 引入过滤机制，使其无需预先过滤日志即可直接处理原始日志，将会很有价值。

致谢。我们感谢亚历山大·冯·洪堡基金会（AvH）对本研究的支持。

参考文献

1. van der Aalst, W.M.P.：佩特里网在工作流管理中的应用。J. Circuits Syst. Comput. **8**(1), 21–66 (1998) 2. van der Aalst, W.M.P.：《流程挖掘——行动中的数据科学》，第二版。Springer (2016) 3. van der Aalst, W.M.P.：使用自由选择网进行流程挖掘和业务流程管理。载于：FedCSIS 2021. 第25卷，第9–15页 (2021) 4. Augusto, A., Conforti, R., Dumas, M., Rosa, M.L., Maggi, F.M., Marrella, A., Mecella, M., Soo, A.：从事件日志自动发现流程模型：综述与基准评测。IEEE Trans. Knowl. Data Eng. **31**(4), 686–705 (2019) 5. Desel, J., Esparza, J.：《自由选择佩特里网》。第40号，剑桥大学出版社 (1995) 6. Dixit, P.M., Buijs, J.C.A.M., van der Aalst, W.M.P.：Prodigy：人在回路中的流程发现。载于：RCIS 2018. 第1–12页。IEEE (2018) 7. van Dongen, B.：BPI 挑战赛 2017 (2017)。https://doi.org/10.4121/uuid:5f3067df-f10b-45da-b98b-86ae4c7a310b 8. Huang, T., van der Aalst, W.M.P.：发现具有非阻塞结构的健全自由选择工作流网。载于：EDOC 2022. 第13585卷，第200–216页。Springer (
2022 10 1007 978 3 031 17604 3_12)。https://doi.org/ 。/ - - - - 。
9
Leemans, S.J.J., Fahland, D., van der Aalst, W.M.P.：可扩展的流程发现与符合性检查。Softw. Syst. Model. **17**(2), 599–631 (2018) 10. de Leoni, M.M., Mannhardt, F.：道路交通罚款管理流程 (2015)。https://doi.org/10 5 11 4121 270 440 1057 4 9 89 9 699 47990 。/uuid: fd - - fb - a -b b f 。Mannhardt, F.：医院账单事件日志 (2017)。https://doi.org/10 741 12 4121 76 46 83 930 4798 1 9 4 94 。/uuid: c b -c - -a c - be dfeb 。Polato, M.：意大利公司服务台日志数据集 (2017)。https://doi.org/10.4121/uuid:0c60edf1-6f83-4e75-9367-4c63b3e9d5bb 13. Schuster, D., Domnitsch, E., van Zelst, S.J., van der Aalst, W.M.P.：用于增量流程发现的通用轨迹排序框架。载于：IDA 2022. 第13205卷，第264–277页。Springer (2022) 14. Schuster, D., van Zelst, S.J., van der Aalst, W.M.P.：分层流程模型的增量发现。载于：RCIS 2020. 第385卷，第417–433页。Springer (2020)