Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

FLCHUS-X: Cost-Aware Pruning for Mining Frequent Sequences with Low Average Cost and High Average Utility

Quan Van

vanhaminhquan2406@gmail.com

Independent Researcher https://orcid.org/0009-0002-8222-0076

Research Article

Keywords: Sequential pattern mining, High utility, Cost, Lower bound, Pruning

Posted Date: October 22nd, 2026

DOI: https://doi.org/10.21203/rs.3.rs-11450858/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: The authors declare no competing interests.

FLCHUS-X: Cost-Aware Pruning for Mining Frequent Sequences with Low Average Cost and High Average Utility

Quan Van

Independent Researcher

Abstract. High utility sequential pattern mining finds sequences of items that yield a high profit, but it ignores the cost of obtaining that profit. To consider both aspects, Truong et al. proposed the problem of mining frequent sequences having a low average cost and a high average utility (FLCHUSM), together with the FLCHUSPM algorithm, whose pruning relies on upper bounds on the average utility and on a lower bound on the average cost. This paper studies lower bounds on the average cost for this problem. It presents *MLB*, a lower bound based on the cheapest occurrence of a sequence, and proves that *MLB* is monotone for all super-sequences. Consequently, the cost can be used in the same pruning strategies as the support and the utility upper bound: depth pruning, removal of items, filtering of co-occurrence maps and tightening of the candidate item sets. The analysis also shows that the first-occurrence lower bound used in FLCHUSPM is not monotone for itemset extensions, so that pruning with it is not safe in general; a second bound, *MDB*, restricted to the subtree of a node, recovers most of its pruning power with a proof. Based on these results, the FLCHUS-X algorithm is presented. Its output was identical to that of a brute-force implementation of the problem definition on 1,900 random databases. On six real sequence datasets, FLCHUS-X was 1.4 to 3.1 times faster than FLCHUSPM and 1.6 to 266 times faster than FLCHUSPM without its unsafe bound, with a similar memory usage.

Keywords: Sequential pattern mining · High utility · Cost · Lower bound · Pruning.

1 Introduction

Sequential pattern mining consists of finding subsequences that appear frequently in a database of sequences [1]. Since many patterns that are frequent are not valuable for the user, high utility sequential pattern mining was proposed to find sequences having a high utility (e.g. profit) [9,5]. But in many applications, the utility of a pattern is obtained by spending some resources such as money or time. For example, in an e-learning platform, a sequence of learning activities may lead to good results (high utility) but require much study time (high

cost). Patterns that combine a high utility with a low cost are then more interesting, which motivated the study of cost-effective patterns [4] and of low-cost high-utility itemsets [6].

Recently, Truong et al. [8] defined the problem of mining frequent sequences with a low average cost and a high average utility (FLCHUSM), where the utility and the cost of a sequence are both measured by their maximum over its occurrences, and averaged over the input sequences that contain it. Since these measures are neither monotone nor anti-monotone, they designed two upper bounds on the average utility (*AMUB* and *FUB*), a lower bound on the average cost (*Flb*), width and depth pruning strategies, and the FLCHUSPM algorithm, which relies on a vertical structure named CUL (cost-utility list).

In FLCHUSPM, the cost is used only for depth pruning, through a lower bound computed from the *first* occurrence of a sequence, while width pruning (removal of items, co-occurrence maps, candidate sets) uses only the support and an upper bound on the utility, because these are monotone for all super-sequences. A cost bound with the same property would allow the cost threshold to take part in all pruning strategies. This paper makes the following contributions:

- It proposes *MLB*, a lower bound on the average cost based on the *cheapest* occurrence of a sequence, and proves that it is monotone for all super-sequences (Section 5). This property allows using the cost in the width pruning strategies: removing items, filtering the co-occurrence maps and tightening the candidate sets.
- It analyses the first-occurrence bound and shows, with a counterexample, that it is not monotone for i-extensions, so that it cannot be used safely for depth pruning (Section 4). It proposes *MDB*, a lower bound valid for the subtree of a node, which uses the most expensive occurrences ending at the positions where the subtree can extend (Section 5.1).
- It presents the FLCHUS-X algorithm, which extends the CUL structure with the cost of the cheapest occurrence and applies the new strategies (Section 6).
- It reports an experimental evaluation of correctness against a brute-force oracle and of performance on real datasets (Section 7).

The rest of this paper is organized as follows. Section 2 reviews related work. Section 3 defines the problem. Sections 4 to 7 present the contributions, and Section 8 draws a conclusion.

2 Related Work

Sequential pattern mining was introduced by Agrawal and Srikant [1]. Efficient algorithms explore the search space of sequences with a pattern-growth approach such as PrefixSpan [7] or with vertical structures such as SPADE [10]. The CMAP structure stores which items co-occur in the database to avoid generating candidates that cannot be frequent [2]. High utility sequential pattern mining

Table 1. The toy QSDB used as running example (q-items are written *item(cost, utility)*).

SID	q-sequence	$u(\Psi')$
Ψ'_1	$a(9,4) \to a(1,6)\, b(2,20) \to c(3,10)$	40
Ψ'_2	$a(8,5) \to a(2,6)\, b(1,18) \to e(30,40)$	69
Ψ'_3	$a(2,7)\, b(2,15) \to d(1,5) \to e(25,30)$	57
Ψ'_4	$b(3,10) \to c(2,12)\, d(4,6)$	28

was popularized by USpan [9], which defines the utility of a sequence in an input sequence as the maximum utility of its occurrences; many algorithms followed (see the survey [5]).

Cost has been considered in several pattern mining tasks. Fournier-Viger et al. proposed mining cost-effective patterns in event logs, where each event has a cost and each sequence has a utility outcome [4]. Nawaz et al. proposed mining low-cost high-utility itemsets in transaction databases [6]. Truong et al. proposed the FLCHUSM problem for quantitative sequence databases where each item occurrence has both a cost and a utility, and the FLCHUSPM algorithm [8]. In [8], FLCHUSPM is evaluated by comparing versions of itself with different pruning strategies; no other algorithm for FLCHUSM was found in the surveyed literature. The present work keeps the search space, the CUL structure and the utility bounds of FLCHUSPM, and replaces its cost bound.

3 Preliminaries and Problem Definition

The definitions of this section are those of [8]. Let A be a set of items with a total order $\prec$. A *quantitative item* (q-item) is a triple (a, co, ut) where $a \in A$ and $co, ut > 0$ are its cost and utility. A q-itemset is a set of q-items with distinct items, and a *q-sequence* $\Psi' = E'_1 \to E'_2 \to \cdots \to E'_q$ is a list of q-itemsets. A quantitative sequence database (QSDB) D' is a set of q-sequences. A *sequence* $\alpha = E_1 \to \cdots \to E_p$ is a list of itemsets (sets of items). The utility (cost) of a q-itemset or q-sequence is the sum of the utilities (costs) of its q-items.

Definition 1 (Occurrence). *A sequence $\alpha = E_1 \to \cdots \to E_p$ occurs in a q-sequence $\Psi' = F'_1 \to \cdots \to F'_q$ if there are positions $1 \le j_1 < \cdots < j_p \le q$ such that E_k is a subset of the items of F'_{j_k} for $k = 1, \ldots, p$. The q-sequence formed by the matched q-items is an occurrence of α; $O(\alpha, \Psi')$ denotes the set of occurrences of α in Ψ', and $\rho(\alpha)$ the set of q-sequences containing α. The support of α is $supp(\alpha) = |\rho(\alpha)|$.*

Definition 2 (Utility and cost of a sequence [8, Def. 4]). *The utility and the cost of α in $\Psi' \in \rho(\alpha)$ are $u(\alpha, \Psi') = \max_{\alpha' \in O(\alpha, \Psi')} u(\alpha')$ and $c(\alpha, \Psi') = \max_{\alpha' \in O(\alpha, \Psi')} c(\alpha')$. The average utility and the average cost of α are $au(\alpha) = \sum_{\Psi' \in \rho(\alpha)} u(\alpha, \Psi')/supp(\alpha)$ and $ac(\alpha) = \sum_{\Psi' \in \rho(\alpha)} c(\alpha, \Psi')/supp(\alpha)$.*

Example 1. In Table 1, the sequence $\langle(ab)\rangle$ (items a and b in the same itemset) appears in Ψ'_1, Ψ'_2 and Ψ'_3, with one occurrence each: $a(1,6)b(2,20)$, $a(2,6)b(1,18)$ and $a(2,7)b(2,15)$. Thus $supp(\langle(ab)\rangle) = 3$, $au(\langle(ab)\rangle) = (26+24+22)/3 = 24$ and $ac(\langle(ab)\rangle) = (3+3+4)/3 \approx 3.33$. The sequence $\langle a\rangle$ appears in the same three q-sequences; in Ψ'_1 its occurrences have costs 9 and 1, so $c(\langle a\rangle, \Psi'_1) = 9$, and $ac(\langle a\rangle) = (9+8+2)/3 \approx 6.33$.

Definition 3 (FLCHUSM [8]). *Given thresholds mins, minu and maxc, a sequence α is a frequent, low average cost and high average utility (FLCHU) sequence if $supp(\alpha) \geq mins$, $au(\alpha) \geq minu$ and $ac(\alpha) \leq maxc$. The FLCHUSM problem is to find all FLCHU sequences.*

Example 2. For $mins = 2$, $minu = 20$ and $maxc = 4$, the only FLCHU sequence of the toy QSDB is $\langle(ab)\rangle$. For example, $\langle b\rangle$ has $ac = 2$ but $au = 15.75 < minu$, and $\langle b \to e\rangle$ has $au = 51.5$ but $ac = 29 > maxc$. (This result was obtained with the brute-force implementation described in Section 7.)

The search space is a prefix tree whose root is the empty sequence; the children of a node α are its *i-extensions* $\alpha \diamond_i y$ (item y added to the last itemset, y greater than all its items) and its *s-extensions* $\alpha \diamond_s y$ (a new itemset $\{y\}$ appended). FLCHUSPM uses the following bounds, where $m = mins$ [8]: $AMUB(\alpha)$ is the average of the m largest values $u(\Psi')$ for $\Psi' \in \rho(\alpha)$; $FUB(\alpha)$ is the average of the m largest values $\max_{\alpha'}(u(\alpha') + u(rem(\alpha')))$, where $rem(\alpha')$ is the part of Ψ' after the occurrence α'. *AMUB* is anti-monotone for all supersequences and *FUB* for forward extensions [8, Theorem 1].

4 Analysis of the First-Occurrence Cost Bound

For the cost, FLCHUSPM uses the following lower bound.

Definition 4 (Flb [8, Def. 6]). *Let α'_{first} be the first occurrence of α in Ψ' (the occurrence with the smallest ending position) and $lb(\alpha, \Psi') = c(\alpha'_{first})$. $Flb(\alpha)$ is the average of the m smallest values $lb(\alpha, \Psi')$, $\Psi' \in \rho(\alpha)$.*

Flb is a lower bound on *ac* [8, Theorem 2a], and FLCHUSPM prunes the subtree of a node α when $Flb(\alpha) > maxc$ (depth pruning). This relies on [8, Theorem 2b], which states that $Flb(\beta) \geq Flb(\alpha)$ for every forward extension β of α. The proof assumes that the first occurrence of β contains the first occurrence of α. This does not hold for i-extensions: the first occurrence of $\alpha \diamond_i y$ can end at a later itemset, where the occurrence of α is cheaper.

Proposition 1. *Flb is not monotone for i-extensions. Consequently, a depth-pruning test based on Flb may discard the subtree of a sequence that has FLCHU i-extensions.*

Proof. Consider Table 1 with $mins = 2$, $minu = 20$ and $maxc = 4$. The first occurrences of $\langle a\rangle$ in $\Psi'_1, \Psi'_2, \Psi'_3$ cost $9, 8, 2$, so $Flb(\langle a\rangle) = (2+8)/2 = 5 > maxc$

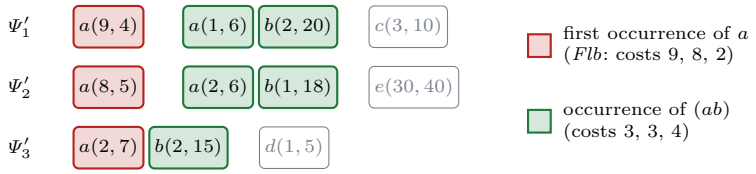

Fig. 1. Why *Flb* is not monotone. With $m = 2$, $Flb(\langle a \rangle) = 5 > Flb(\langle (ab) \rangle) = 3$: the occurrences of (ab) do not contain the (expensive) first occurrence of a. In Ψ'_3 the first occurrence of a is also part of (ab).

and FLCHUSPM prunes the subtree of $\langle a \rangle$. The first (and only) occurrences of its i-extension $\langle (ab) \rangle$ cost $3, 3, 4$, so $Flb(\langle (ab) \rangle) = 3 < Flb(\langle a \rangle)$. Since $\langle (ab) \rangle$ is an FLCHU sequence (Example 2), pruning the subtree of $\langle a \rangle$ with *Flb* excludes it. □

Figure 1 illustrates the situation: the first occurrence of a is expensive in Ψ'_1 and Ψ'_2, while the occurrences of (ab) use the cheaper occurrences of a that come later. The bound *Flb* remains a valid lower bound on the average cost of a sequence itself [8, Theorem 2a]; only its use for pruning descendants is affected.

5 A Lower Bound Based on the Cheapest Occurrence

Definition 5 (MLB). *For $\Psi' \in \rho(\alpha)$, let $lb_{\min}(\alpha, \Psi') = \min_{\alpha' \in O(\alpha, \Psi')} c(\alpha')$ be the cost of the cheapest occurrence of α in Ψ'. For a sequence with $supp(\alpha) \geq m$, $MLB(\alpha)$ is the average of the m smallest values $lb_{\min}(\alpha, \Psi')$, $\Psi' \in \rho(\alpha)$.*

Example 3. In Table 1, $lb_{\min}(\langle a \rangle, \Psi'_1) = 1$, $lb_{\min}(\langle a \rangle, \Psi'_2) = 2$ and $lb_{\min}(\langle a \rangle, \Psi'_3) = 2$, so $MLB(\langle a \rangle) = 1.5$, while $Flb(\langle a \rangle) = 5$. For the item e, $MLB(\langle e \rangle) = (25 + 30)/2 = 27.5$.

Lemma 1. *Let β be a super-sequence of α and $\Psi' \in \rho(\beta)$. Then $c(\beta, \Psi') \geq lb_{\min}(\beta, \Psi') \geq lb_{\min}(\alpha, \Psi')$.*

Proof. The first inequality holds because a maximum is at least a minimum over the same non-empty set. Let β' be an occurrence of β in Ψ'. Since α is contained in β, the q-items of β' that match the items of α form an occurrence α' of α in Ψ', and $c(\beta') \geq c(\alpha')$ because costs are positive. Hence $c(\beta') \geq lb_{\min}(\alpha, \Psi')$ for every occurrence β', and the result follows by taking the minimum over β'. □

Theorem 1 (Properties of MLB). *Let α be a sequence with $supp(\alpha) \geq m$. (a) $MLB(\alpha) \leq ac(\alpha)$. (b) For every super-sequence β of α with $supp(\beta) \geq m$, $MLB(\beta) \geq MLB(\alpha)$, hence $ac(\beta) \geq MLB(\alpha)$.*

Proof. (a) Let $s = supp(\alpha) \geq m$. By Lemma 1 (with $\beta = \alpha$), each value $c(\alpha, \Psi')$ is at least $lb_{\min}(\alpha, \Psi')$, so $ac(\alpha)$ is at least the average of the s values $lb_{\min}(\alpha, \Psi')$, which is at least the average of the m smallest of them. (b) Since β contains α,

$\rho(\beta) \subseteq \rho(\alpha)$. Let $\Psi'_1, \ldots, \Psi'_m$ be the q-sequences of $\rho(\beta)$ with the m smallest values $lb_{\min}(\beta, \cdot)$. By Lemma 1, $lb_{\min}(\beta, \Psi'_j) \geq lb_{\min}(\alpha, \Psi'_j)$ for each j, and the m distinct q-sequences Ψ'_j belong to $\rho(\alpha)$, so $\sum_j lb_{\min}(\alpha, \Psi'_j)$ is at least the sum of the m smallest values $lb_{\min}(\alpha, \cdot)$. Thus $MLB(\beta) \geq MLB(\alpha)$, and $ac(\beta) \geq MLB(\beta) \geq MLB(\alpha)$ by (a). □

Theorem 1(b) holds for *all* super-sequences, which is the same kind of property as the anti-monotonicity of *AMUB* and of the support used by the width pruning strategies of FLCHUSPM. It therefore yields the following strategies, whose proofs repeat those of the corresponding strategies of [8, Section 4.3] with *MLB* in place of *AMUB*.

Corollary 1 (Cost-aware pruning strategies).

1. Depth pruning. *If $MLB(\alpha) > maxc$, no super-sequence of α (in particular no node of the subtree of α) is an FLCHU sequence.*
2. Item removal. *If $supp(\langle y \rangle) < m$, $AMUB(\langle y \rangle) < minu$ or $MLB(\langle y \rangle) > maxc$, no FLCHU sequence contains y, and y can be removed from D'. Removing items lowers the utilities $u(\Psi')$ and thus AMUB for the remaining items, while supports and MLB values are unchanged; the removal can therefore be repeated until no item is removed.*
3. Co-occurrence maps. *If $MLB(\langle (xy) \rangle) > maxc$ (resp. $MLB(\langle x \to y \rangle) > maxc$), no i-extension (resp. s-extension) with y of a sequence whose last item is x leads to an FLCHU sequence, so y can be excluded from $iCMAP(x)$ (resp. $sCMAP(x)$).*
4. Candidate sets. *If $MLB(\alpha \diamond y) > maxc$, y can be excluded from the candidate set $I(\alpha)$ or $S(\alpha)$, and therefore from the candidate sets of all descendants of α (tightening strategy of [8]).*

Example 4. With $mins = 2$, $minu = 20$ and $maxc = 4$, item e passes the width tests of FLCHUSPM ($supp = 2$ and $AMUB(\langle e \rangle) = 63$) but $MLB(\langle e \rangle) = 27.5 > maxc$, so FLCHUS-X removes e. The utilities of Ψ'_2 and Ψ'_3 become 29 and 27, and $AMUB(\langle a \rangle)$ decreases from 63 to 34.5. In the same way, $MLB(\langle b \to c \rangle) = (5 + 5)/2 = 5$ and $MLB(\langle b \to d \rangle) = (3 + 7)/2 = 5$ exceed $maxc$, so $c, d \notin sCMAP(b)$.

5.1 A cost bound for the subtree of a node

MLB is valid for all super-sequences, but on long q-sequences where items repeat, the cheapest occurrence of a sequence can be much cheaper than the occurrences used by its extensions. Since $c(\beta, \Psi')$ is a *maximum* over the occurrences of β, a stronger bound holds for the nodes of the subtree of α. Let c_e denote the maximum cost of the occurrences of α in Ψ' that end at itemset e, and e_1 the first such end.

Definition 6 (MDB). *For $\Psi' \in \rho(\alpha)$, let $lb_D(\alpha, \Psi')$ be the minimum of (i) c_{e_1}, if Ψ' has an itemset after e_1, and (ii) the values c_e for the ends e whose*

Table 2. Pruning tests of FLCHUSPM and FLCHUS-X (✓: used; ×: not used).

Use	Test	Valid for	FLCHUSPM	FLCHUS-X
Depth pruning	$FUB < minu$	forward ext.	✓	✓
Depth pruning	$Flb > maxc$	no (Prop. 1)	✓	×
Depth pruning	$MLB > maxc$	all super-seq.	×	✓
Depth pruning (descendants)	$MDB > maxc$	forward ext.	×	✓
Width, items, CMAPs	$supp < mins$, $AMUB < minu$	all super-seq.	✓	✓
Width, items, CMAPs	$MLB > maxc$	all super-seq.	×	✓
Item removal to a fixpoint	–	–	×	✓

itemset contains an item greater than the last item of α; it is $+\infty$ if neither exists. $MDB(\alpha)$ is the average of the m smallest finite values $lb_D(\alpha, \Psi')$, and $+\infty$ if there are fewer than m of them.

Proposition 2. *For every node $\beta \neq \alpha$ of the subtree of α with $supp(\beta) \geq m$, $ac(\beta) \geq MDB(\alpha)$.*

Proof. Let $\Psi' \in \rho(\beta)$ and β' an occurrence of β in Ψ'; the part of β' that matches α ends at some e. If the first extension step from α to β is an s-extension, all other q-items of β' lie after e; replacing the α-part by the most expensive occurrence of α ending at $e_1 \leq e$ gives an occurrence of β, so $c(\beta, \Psi') \geq c_{e_1}$, and an itemset follows e_1. If it is an i-extension, the other q-items of β' at e are greater than the last item of α, and the others lie after e; replacing the α-part by the most expensive occurrence of α ending at e gives an occurrence of β, so $c(\beta, \Psi') \geq c_e$ for an end e of type (ii). Hence $c(\beta, \Psi') \geq lb_D(\alpha, \Psi')$, which is finite, for every $\Psi' \in \rho(\beta)$, and the argument of Theorem 1(b) gives the result. □

Since $c_e \geq cm_e$, $MDB(\alpha) \geq MLB(\alpha)$. For descendants that start with an s-extension, term (i) is the first-occurrence cost used by *Flb*; term (ii) is what makes the bound valid for i-extensions. *MDB* holds only for the subtree, so it is used to stop the expansion of a node *after* the node itself has been tested. In the toy QSDB, $MDB(\langle a \rangle) = (1+2)/2 = 1.5$: in Ψ'_1 and Ψ'_2, a ends at the itemset (ab), where an i-extension with b is possible.

6 The FLCHUS-X Algorithm

6.1 The CUL structure with cheapest costs

FLCHUS-X uses the CUL (cost-utility list) structure of FLCHUSPM [8, Section 5.1]. For a sequence α and $\Psi' \in \rho(\alpha)$, the CUL stores one tuple per ending position e of α in Ψ', containing the maximum cost c_e and the maximum utility u_e of the occurrences of α that end at e, and the utility ru_e of the rest of Ψ'. FLCHUS-X adds one field: the cost cm_e of the cheapest occurrence ending at e. Then $u(\alpha, \Psi') = \max_e u_e$, $c(\alpha, \Psi') = \max_e c_e$, $lb_{\min}(\alpha, \Psi') = \min_e cm_e$, and the bound of *FUB* is $\max_e(u_e + ru_e)$.

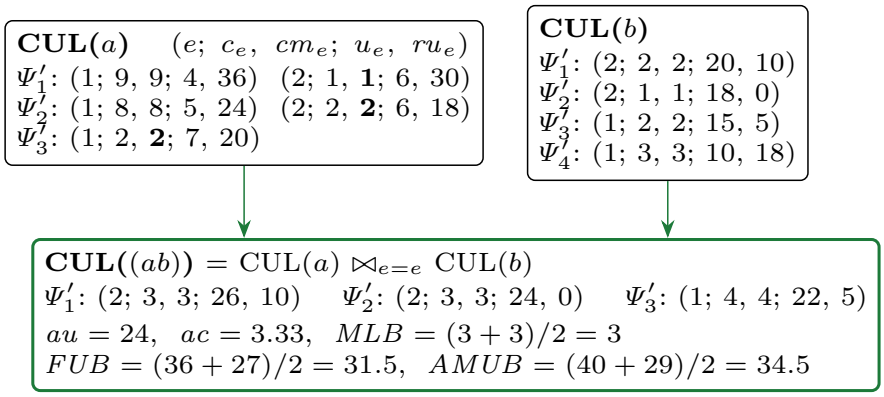

Fig. 2. CULs of a, b and of the i-extension (ab) on the toy QSDB after removing e. Bold values give $lb_{\min}(\langle a\rangle, \cdot) = 1, 2, 2$, hence $MLB(\langle a\rangle) = 1.5$, whereas the first occurrences of a give $Flb(\langle a\rangle) = 5$.

Lemma 2 (CUL of extensions). *Let t_y be the tuple of item y at position e in Ψ', with cost $c(y)$. For the i-extension $\beta = \alpha \diamond_i y$, the tuple of β at e exists if α has a tuple t_α at e, and $cm_e(\beta) = cm_e(\alpha) + c(y)$. For the s-extension $\beta = \alpha \diamond_s y$, it exists if α has a tuple at a position $e' < e$, and $cm_e(\beta) = \min_{e'<e} cm_{e'}(\alpha) + c(y)$.*

Proof. Every occurrence of β ending at e consists of an occurrence of α ending at e (i-extension) or at some $e' < e$ (s-extension), plus the q-item of y at e; conversely every such combination is an occurrence of β (for an i-extension, y is greater than the items of the last itemset of α). Since the cost of the combination is the sum of the two costs, the minimum over the occurrences ending at e is the minimum of the costs of α plus $c(y)$. □

The maximum cost and utility of extensions are computed in the same way with maxima [8, Section 5.1.3]. Figure 2 shows the CULs of a, b and (ab) of the toy QSDB after the removal of e.

6.2 The algorithm

Algorithm 1 shows FLCHUS-X. It first computes the CULs of the items and removes items with Corollary 1(2) until no item is removed (lines 1–5). It then builds the co-occurrence maps with the support, *AMUB* and *MLB* tests (line 6) and explores the prefix tree depth-first (procedure Search). For a node α, it applies depth pruning with *FUB* and *MLB* (line 9), outputs α if it is an FLCHU sequence (line 10), stops the expansion if $MDB(\alpha) > maxc$ (line 11), and builds the candidate sets of i- and s-extensions with the width tests (lines 12–17). The search space, the utility bounds and the tightening of candidate sets are those of FLCHUSPM; the differences are the use of *MLB* and *MDB* instead of *Flb*, the cost tests in lines 3, 6, 14 and 16, and the repetition of item removal.

Algorithm 1 FLCHUS-X

Require: QSDB D', thresholds $mins$, $minu$, $maxc$
Ensure: the set of FLCHU sequences
1: **repeat**
2: compute $u(\Psi')$ for all $\Psi' \in D'$ and CUL($\langle y \rangle$) for all items y
3: $R \leftarrow \{y \mid supp(\langle y \rangle) < mins \vee AMUB(\langle y \rangle) < minu \vee MLB(\langle y \rangle) > maxc\}$
4: remove the items of R from D'
5: **until** $R = \emptyset$
6: compute $iCMAP$ and $sCMAP$ with the tests on $supp$, $AMUB$ and MLB
7: $S \leftarrow$ items of D'; **for each** $x \in S$ **do** SEARCH($\langle x \rangle, S, S$)
8: **procedure** SEARCH(α, I, S)
9: **if** $FUB(\alpha) < minu \vee MLB(\alpha) > maxc$ **then return**
10: **end if**
11: **if** $au(\alpha) \geq minu \wedge ac(\alpha) \leq maxc$ **then** output α
12: **end if**
13: **if** $MDB(\alpha) > maxc$ **then return**
14: **end if**
15: $x \leftarrow$ last item of α; $I(\alpha) \leftarrow \emptyset$; $S(\alpha) \leftarrow \emptyset$
16: **for all** $y \in I$, $y \succ x$, $y \in iCMAP(x)$ **do**
17: **if** $supp(\alpha \diamond_i y) \geq mins \wedge AMUB(\alpha \diamond_i y) \geq minu \wedge MLB(\alpha \diamond_i y) \leq maxc$ **then** $I(\alpha) \leftarrow I(\alpha) \cup \{y\}$
18: **end if**
19: **end for**
20: **for all** $y \in S$, $y \in sCMAP(x)$ **do**
21: **if** $supp(\alpha \diamond_s y) \geq mins \wedge AMUB(\alpha \diamond_s y) \geq minu \wedge MLB(\alpha \diamond_s y) \leq maxc$ **then** $S(\alpha) \leftarrow S(\alpha) \cup \{y\}$
22: **end if**
23: **end for**
24: **for all** $y \in I(\alpha)$ **do** SEARCH($\alpha \diamond_i y, I(\alpha), S(\alpha)$)
25: **end for**
26: **for all** $y \in S(\alpha)$ **do** SEARCH($\alpha \diamond_s y, S(\alpha), S(\alpha)$)
27: **end for**
28: **end procedure**

Theorem 2. *FLCHUS-X outputs exactly the set of FLCHU sequences.*

Proof. Every output sequence satisfies the three conditions (line 10), with values computed exactly from the CULs (Lemma 2 and [8, Section 5.1]). Completeness: the search space is the prefix tree of all sequences over the items of D'; a subtree is pruned or an item is skipped only by a test that, by [8, Theorems 1 and 3] for the support, $AMUB$ and FUB, by Theorem 1 and Corollary 1 for MLB, and by Proposition 2 for MDB (which is applied after the node itself is tested), guarantees that the skipped sequences are not FLCHU sequences. The values c_e and the ends needed by MDB are available in the CUL. □

Implementation details. The bounds need the sum of the m largest (or smallest) values over the q-sequences of a CUL. FLCHUS-X obtains it with a linear-time

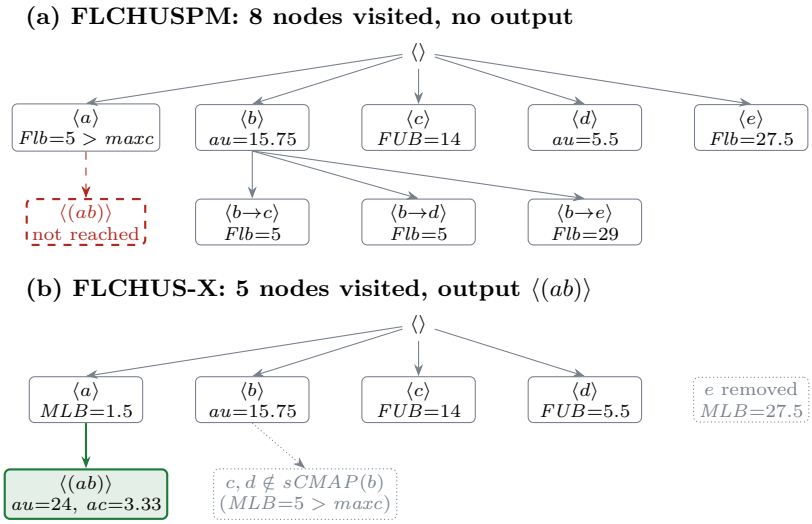

Fig. 3. Nodes visited on the toy QSDB with $mins = 2$, $minu = 20$, $maxc = 4$. Gray: node visited and not output (the value shown is the reason); green: FLCHU sequence; red dashed: FLCHU sequence not reached; dotted: pruned before the search.

selection instead of sorting, and stops the construction of an extension's CUL as soon as fewer than m q-sequences can still contain it. Neither change affects the output.

6.3 Running example

Figure 3 compares the nodes visited by FLCHUSPM and by FLCHUS-X on the toy QSDB ($mins = 2$, $minu = 20$, $maxc = 4$). With Flb-based depth pruning, the subtree of $\langle a \rangle$ is pruned, so $\langle (ab) \rangle$ is not reached, and the nodes $\langle e \rangle$, $\langle b \to c \rangle$, $\langle b \to d \rangle$ and $\langle b \to e \rangle$ are visited before being pruned; 8 nodes are visited. FLCHUS-X removes e before the search and excludes c and d from $sCMAP(b)$, since their MLB values exceed $maxc$; it visits 5 nodes and outputs $\langle (ab) \rangle$. The values in the figure were computed by the implementations described in Section 7.

6.4 Complexity

As in FLCHUSPM, the CUL of an extension is built by a merge of two CULs in time linear in their number of tuples, and the measures of a node are computed in time linear in the size of its CUL; the selection of the m extreme values takes linear expected time instead of $O(n \log n)$ for a sort over n q-sequences. FLCHUS-X adds one value per tuple (cm_e) to the CULs of FLCHUSPM. Item removal is repeated at most $|A|$ times, each time with a scan of D'; on the six datasets of Section 7 it stopped after two rounds, the second removing no item.

Table 3. Comparison with the brute-force implementation (number of databases where the output differs; missing / extra patterns / patterns with different values).

Algorithm	Databases	FLCHU seq.	Differing	Missing	Extra	Values
FLCHUSPM	500	21,807	2	2	0	0
FLCHUSPM$^-$	500	21,807	0	0	0	0
FLCHUS-X	500	21,807	0	0	0	0
FLCHUS-X (second seed)	1,000	53,439	0	0	0	0
Long q-sequences over 2–3 items (3–8 itemsets each)						
FLCHUSPM	400	591,101	1	3	0	0
FLCHUS-X	400	591,101	0	0	0	0

7 Experimental Evaluation

FLCHUS-X and FLCHUSPM were implemented in C in the same framework, sharing the input parser and the CUL construction; FLCHUSPM follows Definitions 4–6, Theorems 1–3, Section 5.1 and Figs. 3–4 of [8]. As no implementation of FLCHUSPM is publicly available, it was implemented from [8] and validated on the running example of that paper: the implementation returns the published result, the CULs of all items of [8, Fig. 1], the published co-occurrence maps, and the 14 nodes of [8, Fig. 5]; the bound values agree except two *Flb* values of pruned nodes, for which the figure reports values equal to the *ac* values of other nodes. Two algorithms are compared: FLCHUSPM, and FLCHUSPM without the *Flb* test (denoted FLCHUSPM$^-$), which is sound and is the correct baseline. All programs are written in C, compiled with GCC 13.3.0 at `-O2`, and run single-threaded through the same standalone driver on an Intel Core i7-10700 (2.9 GHz) with 31 GiB of memory under Ubuntu 24.04.5. Each setting was run three times in sequence for every algorithm; the median time, from reading the input to the end of the search, and the peak resident memory measured by `/usr/bin/time` are reported. Runs were stopped after 300 s.

7.1 Correctness

A brute-force implementation of the problem definition (Definitions 1–4 of [8]) enumerates every sequence with support at least *mins* and computes its utility and cost from all its occurrences. It was compared with the algorithms on random QSDBs with 2 to 7 q-sequences of 1 to 5 itemsets over 2 to 5 items, random costs in $[1, 20]$, utilities in $[1, 50]$ and random thresholds. Table 3 reports the results.

The databases where FLCHUSPM differs contain an i-extension whose prefix is pruned by *Flb* as in Proposition 1. The second group of databases, with many repeated items, exercises the cases where the cheapest, first and most expensive occurrences differ, on which *MDB* relies. Each of the four components of FLCHUS-X (*MDB*, cost-aware width pruning, repeated item removal, linear-time selection) was also disabled in turn, and the output was identical to the brute force in all 500 databases of the first seed.

Table 4. Characteristics of the datasets.

Dataset	#sequences	#items	avg. length	(min; max) length
BMS	59,601	497	2.5	(1; 267)
Snake	163	20	60.6	(57; 61)
SIGN	730	267	52.0	(18; 94)
FIFA	20,450	2,990	36.2	(9; 100)
LEVIATHAN	5,834	9,025	33.8	(9; 100)
BIBLE	36,369	13,905	21.6	(9; 100)

Table 5. Median runtime at the default setting (*mins*%, *minu*%, *maxc*%) of each dataset; TL: time limit of 300 s exceeded.

Dataset	#FLCHUS	FLCHUSPM (s)	FLCHUSPM$^-$ (s)	FLCHUS-X (s)	Speed-up vs FLCHUSPM$^-$
Snake (60, 5, 5)	851	1.58	20.6	0.73	28.2×
BMS (0.03, 0.05, 1)	71,918	47.6	TL	20.8	>14×
SIGN (15, 2, 3)	6,311	0.99	1.18	0.40	3.0×
LEVIATHAN (2, 1, 3)	2,512	3.28	4.66	1.99	2.3×
BIBLE (2, 1, 3)	1,293	4.73	5.15	2.66	1.9×
FIFA (8, 1, 3)	12,584	115	143	46.1	3.1×

7.2 Datasets

The datasets of [8] include BMS, Snake and three synthetic datasets generated with the IBM Quest generator; the synthetic files are not publicly available. BMS and Snake and four other real sequence datasets commonly used in sequential pattern mining were used, all obtained from the SPMF library website [3] (Table 4; BMS and Snake match the characteristics reported in [8]). As in [8], costs and utilities of q-items were generated in $[1, 70]$ and $[10, 100]$ with a log-normal distribution; since its parameters are not given in [8], each q-item received $\exp(Y)$ rounded, where Y is normal with the interval of logarithms as ± 3 standard deviations, truncated to the interval.

7.3 Runtime, memory and search space

For each dataset, one of *mins*, *minu* and *maxc* (in the percentage form of [8]) was varied while the two others kept a default value (Table 5); the values of a sweep go from easy to hard, and once an algorithm exceeded the time limit, the harder values of that sweep were not run for it. Over the 69 settings, all algorithms that finished returned the same number of FLCHU sequences; in particular FLCHUSPM missed no pattern on these datasets, so the unsafe pruning of Section 4 was only observed on the random databases of Table 3.

Table 5, Table 6 and Figure 4 show the results. FLCHUS-X was the fastest algorithm in every setting where an algorithm finished. Against the sound baseline FLCHUSPM$^-$, the speed-up ranges from 1.6 to 266: without a valid cost bound, FLCHUSPM$^-$ cannot exploit *maxc*, and on BMS it exceeded the time limit in all settings. Against FLCHUSPM, whose *Flb* test is not safe, FLCHUS-X was 1.4 to 3.1 times faster, and the speed-up grew when *mins* decreased on

Table 6. Range of the speed-up of FLCHUS-X over the settings of each sweep, and number of settings solved within the time limit.

Dataset	Sweep	vs FLCHUSPM	vs FLCHUSPM$^-$	Solved: FLCHUSPM / FLCHUSPM$^-$ / FLCHUS-X
Snake	ms	2.0–2.4	16.4–48.9	5 / 5 / 5 of 5
Snake	mu	2.1–2.1	28.1–28.4	5 / 5 / 5 of 5
Snake	mc	1.9–2.9	7.8–266.3	5 / 5 / 5 of 5
BMS	ms	1.9–2.9	–	5 / 0 / 5 of 5
BMS	mu	2.2–2.3	–	5 / 0 / 5 of 5
BMS	mc	1.7–2.1	–	2 / 0 / 2 of 3
SIGN	ms	2.3–2.5	2.5–3.2	5 / 5 / 5 of 5
SIGN	mu	2.2–2.4	2.9–3.0	5 / 5 / 5 of 5
SIGN	mc	2.1–2.4	2.1–4.6	4 / 4 / 4 of 4
LEVIATHAN	ms	1.5–1.7	2.0–2.6	5 / 5 / 5 of 5
LEVIATHAN	mu	1.4–1.6	2.0–2.3	5 / 5 / 5 of 5
LEVIATHAN	mc	1.4–1.6	1.6–3.5	4 / 4 / 4 of 4
BIBLE	ms	1.6–1.8	1.8–2.1	5 / 5 / 5 of 5
BIBLE	mu	1.4–1.7	1.7–2.0	5 / 5 / 5 of 5
BIBLE	mc	1.5–1.7	1.6–2.3	4 / 4 / 4 of 4
FIFA	ms	2.4–2.7	2.6–3.1	4 / 3 / 5 of 5
FIFA	mu	2.5–2.5	3.3–3.5	4 / 4 / 4 of 4
FIFA	mc	2.3–3.1	2.4–8.0	3 / 3 / 3 of 3

Table 7. Second sweep of *minu* on BMS ($mins = 0.03\%$, $maxc = 1\%$); median of three runs (min–max).

minu (%)	#FLCHUS	FLCHUSPM (s)	FLCHUS-X (s)	Speed-up
0.5	71,520	57.5 (57–58)	25.0 (25–25)	2.30
0.75	66,882	60.2 (57–62)	24.8 (25–25)	2.43
1	55,350	59.6 (59–64)	24.9 (25–25)	2.39
1.25	6,449	59.7 (58–60)	24.6 (25–25)	2.43
1.5	11	58.4 (58–59)	24.6 (25–25)	2.37

BMS and FIFA. On FIFA with $mins = 5\%$, FLCHUS-X was the only algorithm to finish. On BMS with $maxc = 2\%$ the number of FLCHU sequences explodes and all algorithms exceeded the time limit.

When *minu* varies, the runtimes hardly change (Table 6). The default values of *minu* on BMS turned out to be below the average utility of almost all frequent sequences (71,918 sequences at every value), so a second sweep with larger values was run on BMS (Table 7); it was run separately, and its absolute times are about 25% higher than in the main run for the same number of nodes, so only the ratios within it should be compared. The number of FLCHU sequences drops from 71,520 to 11 while the number of nodes visited by FLCHUS-X stays at about 497,000: on these datasets the utility bounds *AMUB* and *FUB* inherited from FLCHUSPM prune almost nothing, and the pruning comes from the support and from the cost bounds.

The peak memory of the algorithms was similar at the default settings, although the tuples of FLCHUS-X store one more value: 4.8 MB for FLCHUSPM and 4.9 MB for FLCHUS-X on Snake, and 1% to 5% less for FLCHUS-X on

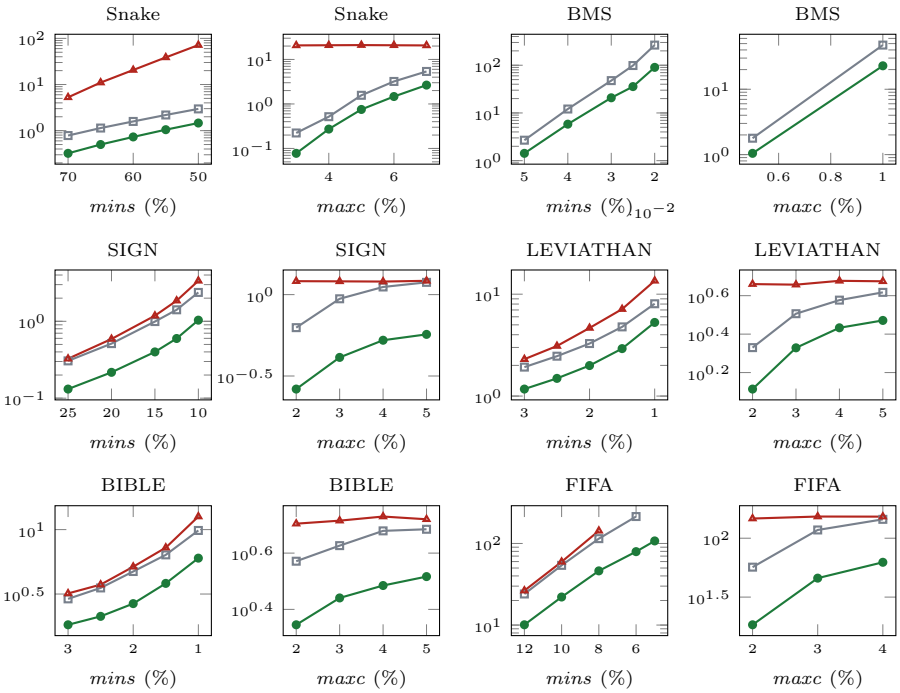

Fig. 4. Median runtime (s, log scale) when varying *mins* (harder to the right) and *maxc*. Gray squares: FLCHUSPM; red triangles: FLCHUSPM$^-$; green circles: FLCHUS-X. Missing points: time limit exceeded.

the five other datasets (for example 185 and 177 MB on BIBLE, 164 and 158 MB on FIFA).

7.4 Ablation study

Table 8 reports the runtime of FLCHUS-X at the default settings with one component disabled. Repeated item removal never removed an item after the first round, so disabling it does not change the search; the differences of up to 8% in that column therefore indicate the variation of the measurements, and differences of this size elsewhere should not be interpreted. The components help on different datasets. *MDB* is decisive on Snake, the dense dataset with long sequences (4.2 times slower without it), but makes FLCHUS-X 5% slower on BMS and 17% slower on FIFA, where it prunes little and its computation is not free. Cost-aware width pruning matters on BMS (1.7 times slower without it); on the other datasets its effect is within 15% in either direction. The linear-time selection and early abandon of joins, which do not change the search space but the cost per node, give the largest gain on SIGN, LEVIATHAN, BIBLE and FIFA (1.8 to 2.4 times). Part of the speed-up of FLCHUS-X over FLCHUSPM

Table 8. Ablation: median runtime (s) of FLCHUS-X at the default setting and with one component disabled (ratio to FLCHUS-X in parentheses).

Dataset	FLCHUS-X	no *MDB*	no cost width	no fixpoint	no selection
Snake	0.73	3.05 (4.2×)	0.68 (0.9×)	0.74 (1.0×)	1.53 (2.1×)
BMS	20.8	19.9 (1.0×)	35.5 (1.7×)	20.9 (1.0×)	25.6 (1.2×)
SIGN	0.40	0.41 (1.0×)	0.46 (1.2×)	0.43 (1.1×)	0.97 (2.4×)
LEVIATHAN	1.99	2.31 (1.2×)	2.08 (1.0×)	2.14 (1.1×)	3.50 (1.8×)
BIBLE	2.66	2.74 (1.0×)	2.79 (1.0×)	2.83 (1.1×)	4.97 (1.9×)
FIFA	46.1	39.5 (0.9×)	45.4 (1.0×)	42.5 (0.9×)	108 (2.3×)

thus comes from this lower cost per node rather than from the new bounds; the bounds explain the gains on Snake and BMS and the settings solved only by FLCHUS-X.

8 Conclusion

This paper studied lower bounds on the average cost for the FLCHUSM problem of Truong et al. [8]. The lower bound *MLB*, based on the cheapest occurrence of a sequence, is monotone for all super-sequences, which allows using the cost not only for depth pruning but also for removing items, filtering the co-occurrence maps and tightening the candidate sets. The analysis also showed that a bound based on the first occurrence is not monotone for i-extensions. The resulting algorithm, FLCHUS-X, returned the same output as a brute-force implementation of the problem definition on all tested databases. A bound restricted to the subtree, *MDB*, recovers on dense data the pruning that the first-occurrence bound offered without its unsafety. On six real datasets, FLCHUS-X was 1.4 to 3.1 times faster than FLCHUSPM and up to 266 times faster than FLCHUSPM without its unsafe bound, with a similar memory usage; the ablation shows that the components help on different kinds of data and that part of the gain comes from a lower cost per node. The experiments also show that the utility bounds inherited from FLCHUSPM prune almost nothing on these datasets, so tighter utility bounds for this problem are a natural next step, as is the use of *MLB* and *MDB* in other pattern mining tasks whose measures are maxima over occurrences.

References

1. Agrawal, R., Srikant, R.: Mining sequential patterns. In: Proceedings of the Eleventh International Conference on Data Engineering, pp. 3–14 (1995). `https://doi.org/10.1109/ICDE.1995.380415`
2. Fournier-Viger, P., Gomariz, A., Campos, M., Thomas, R.: Fast vertical mining of sequential patterns using co-occurrence information. In: Advances in Knowledge

Discovery and Data Mining (PAKDD 2014), LNCS, pp. 40–52. Springer (2014). https://doi.org/10.1007/978-3-319-06608-0_4
3. Fournier-Viger, P., Lin, J.C.W., Gomariz, A., Gueniche, T., Soltani, A., Deng, Z.: The SPMF open-source data mining library version 2. In: Machine Learning and Knowledge Discovery in Databases (ECML PKDD 2016), LNCS, pp. 36–40. Springer (2016). https://doi.org/10.1007/978-3-319-46131-1_8
4. Fournier-Viger, P., Li, J., Lin, J.C.W., Truong Chi, T., Uday Kiran, R.: Mining cost-effective patterns in event logs. Knowledge-Based Systems **191**, 105241 (2020). https://doi.org/10.1016/j.knosys.2019.105241
5. Gan, W., Lin, J.C.W., Fournier-Viger, P., Chao, H.C., Tseng, V.S., Yu, P.S.: A survey of utility-oriented pattern mining. IEEE Transactions on Knowledge and Data Engineering **33**(4), 1306–1327 (2021). https://doi.org/10.1109/TKDE.2019.2942594
6. Nawaz, M.S., Fournier-Viger, P., Alhusaini, N., He, Y., Wu, Y., Bhattacharya, D.: LCIM: Mining low cost high utility itemsets. In: Multi-disciplinary Trends in Artificial Intelligence (MIWAI 2022), LNCS, pp. 73–85. Springer (2022). https://doi.org/10.1007/978-3-031-20992-5_7
7. Pei, J., Han, J., Mortazavi-Asl, B., Wang, J., Pinto, H., Chen, Q., Dayal, U., Hsu, M.C.: Mining sequential patterns by pattern-growth: the PrefixSpan approach. IEEE Transactions on Knowledge and Data Engineering **16**(11), 1424–1440 (2004). https://doi.org/10.1109/TKDE.2004.77
8. Truong, T., Duong, H., Le, B., Fournier-Viger, P., Yun, U.: Mining interesting sequences with low average cost and high average utility. Applied Intelligence **52**(7), 7136–7157 (2022). https://doi.org/10.1007/s10489-021-02505-0
9. Yin, J., Zheng, Z., Cao, L.: USpan: an efficient algorithm for mining high utility sequential patterns. In: Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 660–668 (2012). https://doi.org/10.1145/2339530.2339636
10. Zaki, M.J.: SPADE: An efficient algorithm for mining frequent sequences. Machine Learning **42**(1–2), 31–60 (2001). https://doi.org/10.1023/A:1007652502315