Supplementary material for "Detecting Patterns of Attacks to Network Security in Urban Air Mobility with Answer Set Programming"

Gioacchino Sterlicchio[a] **and Francesca Alessandra Lisi**[b]

[a]DMMM, Polytechnic University of Bari, Italy
[b]DIB and CILA, University of Bari Aldo Moro, Italy
ORCID (Gioacchino Sterlicchio): https://orcid.org/0000-0002-2936-0777, ORCID (Francesca Alessandra Lisi): https://orcid.org/0000-0001-5414-5844

Abstract

This document presents the supplementary material for the paper entitled "Detecting Patterns of Attacks to Network Security in Urban Air Mobility with Answer Set Programming" accepted for presentation at the 27th European Conference on Artficial Intelligence (ECAI 2024). It contains the Answer Set Programming (ASP) encoding for the problem considered in the paper and details of further experiments that could not be included due to lack of space.

1 MASS-CSP encoding

In this section we show the complete Answer Set Programming (ASP) encoding for the Contrast Sequential Pattern Mining (CSPM) tasks which from now on will be referred to as Mining with Answer Set Solving - Contrast Sequential Patterns (MASS-CSP). First of all, basic encoding will be introduced and then in a dedicated section the gap and span constraints as described in the main article.

1.1 CSPM

The full ASP encoding for CSPM is reported below. It encompasses two phases; The first aims at the discovery of frequent sequential patterns, the latter checks which among the discovered patterns are of contrast with some class. Listing 1 shows how to find sequential patterns. We start by acquiring all the elements other than the input sequences (Line 1). Lines 3-7 generate all possible candidate patterns combining all different elements of the sequences. Candidate patterns are generated taking into account a minimum and maximum length called *minlen* and *maxlen* respectively. From 9 to 11, pattern candidate occurrences are computed by analyzing all sequences. Lines 13-15 find support of a candidate pattern and if its support is less than a minimun support threshold *misup*, it will not be a sequential pattern.

Listing 1. ASP encoding for finding sequential patterns

```
1   item(I) :- seq(_, _, I).
2
3   patpos(1).
4   { patpos(X+1) } :- patpos(X), X < maxlen.
5   patlen(L) :- patpos(L), not patpos(L+1).
6   :- patlen(L), L < minlen.
7   1 {pat(X, I): item(I)} 1 :- patpos(X).
8
9   occ(T, 1, P) :- seq(T, P, I), pat(1, I).
10  occ(T, L, P) :- occ(T, L,   P-1), seq(T, P, _).
11  occ(T, L, P) :- occ(T, L-1, P-1), seq(T, P, C), pat(L, C).
12
13  seqlen(T, L) :- seq(T, L, _), not seq(T, L+1, _).
14  support(T) :- occ(T, L, LS), patlen(L), seqlen(T, LS).
15  :- { support(T) } < minsup.
```

Listing 2 shows the second phase of MASS-CSPM. In the code below, Lines 1-2 compute the cardinality of the datasets $\mathcal{D}_1$ and $\mathcal{D}_2$ whereas Lines 4-5 compute the support of a pattern s in $\mathcal{D}_1$ and $\mathcal{D}_2$ respectively. Lines 8-9 calculate $GR_{C_1}(s)$ in accordance with the formula in Section 3.1 of the main paper, while Line 7 capture the case of $GR_{C_1}(s) = \infty$. ASP does not support the computation of formulas that return decimal values. For this reason, a Python script has been developed which can be called from within ASP (with the @ command followed by the function name). The result will no longer be treated in ASP as a constant but rather as a string. Analogously, Lines 12-13 encode the computation of GR_{C_2} as written in Section 3.1 of the main article and Line 11 concerns the infinite case for GR_{C_2}. Finally, Lines 15-16 check if the sequence s in hand is a contrast pattern for either C_1 or C_2 by means of a Python function because it compares decimal numbers. If the growth rate is less than *mincr*, a constant *no* is returned, *yes* otherwise. Line 15 sets the first term of the *contrast_pattern* to *yes* in accordance with the formulas in section 3.1 of the main paper. The integrity constraint at Line 17 discards all answer sets that do not represent contrast patterns for any of the two classes.

Listing 2. ASP encoding for finding contrast sequential patterns

```
1   card(Card, c1) :- Card = #count{T : cl(T, c1)}.
2   card(Card, c2) :- Card = #count{T : cl(T, c2)}.
3
4   sup(Sup, c1) :- Sup = #count{T : support(T), seq(T, _, _), cl(T, c1)}.
5   sup(Sup, c2) :- Sup = #count{T : support(T),  seq(T, _, _), cl(T, c2)}.
6
7   gr_rate(inf, c1) :- sup(Sup1, c1), Sup1 != 0, sup(0, c2).
8   gr_rate(@gr(Sup1, Card1, Sup2, Card2), c1) :- sup(Sup1, c1), card(Card1, c1), sup(Sup2, c2), card(Card2, c2),
9                                                 Sup1 > 0, Sup2 > 0.
10
11  gr_rate(inf, c2) :- sup(Sup2, c2), Sup2 != 0, sup(0, c1)
12  gr_rate(@gr(Sup2, Card2, Sup1, Card1), c2) :- sup(Sup1, c1), card(Card1, c1), sup(Sup2, c2), card(Card2, c2),
13                                                Sup1 > 0, Sup2 > 0.
14
15  contr_pat(yes, Class) :- gr_rate(inf, Class).
16  contr_pat(@csp(Cr, mincr), Class) :- gr_rate(Cr, Class), Cr != inf.
17  :- contr_pat(no, c1), contr_pat(no, c2).
```

1.2 CSPM constraints

Below the constraints described in the main article to be replaced in the pattern embedding section of the MASS-CSPM reported in Listing 1 (Lines 9-11).

Listing 3. ASP encoding of the span constraint.

```
occS(T,1,P,P) :- seq(T,P,I), pat(1,I).
occS(T,L,P,IP) :- occS(T,L,P-1,IP), seq(T,P,_).
occS(T,L,P,IP) :- occS(T,L-1,P-1,IP), seq(T,P,C),
      pat(L,C), P-IP+1>=minspan, P-IP+1<=maxspan.
```

Listing 4. ASP encoding of the gap constraint.

```
occG(T,1,P) :- seq(T,P,I), pat(1,I).
occG(T,L,P) :- occG(T,L,P-1), seq(T,P,_).
occG(T,L,P) :- occG(T,L-1,P-1), seq(T,P,C), pat(L,C), pat(L-1,C1), seq(T,P2,C1), P-P2-1>=mingap, P-P2-1<=maxgap.
```

Listing 5. ASP encoding of the span and gap constraints.

```
occSG(T,1,P,P) :- seq(T,P,I), pat(1,I).
occSG(T,L,P,IP) :- occSG(T,L,P-1,IP), seq(T,P,_).
occSG(T,L,P,IP) :- occSG(T,L-1,P-1,IP), seq(T,P,C), pat(L,C), pat(L-1,C1), seq(T,P2,C1), P-P2-1>=mingap,
                   P-P2-1<=maxgap, P-IP+1>=minspan, P-IP+1<=maxspan.
```

2 Further experiments

This section contains further experiments which did not fit into the paper.

minlen=2 & maxlen=2				
minsup	mincr	#pat	time	memory
10%	1	162	22.58	415.91
20%	1	97	19.81	415.91
30%	1	72	21.34	415.91
40%	1	63	15.44	415.91
50%	1	60	14.78	415.91
minlen=2 & maxlen=6				
minsup	mincr	#pat	time	memory
10%	1	50,596	T.O	667.14
20%	1	83,962	T.O	594.66
30%	1	63,152	T.O	598.47
40%	1	34,853	2,372.92	552.66
50%	1	29,080	1,785.24	475.62

Table 1. Number of patterns, runtime (seconds), grounder time (seconds), solver time (seconds) and memory consumption (MB) on Auth_failure_1000. *minsup* was varied and *mincr* was left unchanged. T.O. means time-out

minlen=2 & maxlen=2				
mincr	minsup	#pat	time	memory
1	30%	72	21.49	415.91
2	30%	19	35.91	416.06
3	30%	19	2000.52	452.64
4	30%	19	310.45	417.48
6	30%	19	58.097	422.5

minlen=2 & maxlen=6				
mincr	minsup	#pat	time	memory
1	30%	63,142	T.O	599.25
2	30%	19,197	T.O	751.66
3	30%	10,826	T.O	997.11
4	30%	11,067	T.O	657.53
5	30%	3,289	T.O	563.07

Table 2. Number of patterns, runtime (seconds), grounder time (seconds), solver time (seconds) and memory consumption (MB) on Auth_failure_1000. *mincr* was varied and *minsup* was left unchanged. T.O. means time-out

(a)

minsup	mincr	#pat	time	memory
10%	1	121	41.09	508.9
20%	1	64	46.92	508.9
30%	1	53	112.49	508.99
40%	1	44	44.73	508.89
50%	1	39	33.35	508.89

(b)

mincr	minsup	#pat	time	memory
1	30%	53	112.60	508.89
2	30%	23	1,247.88	560.62
3	30%	23	461.04	541.97
4	30%	21	T.O	660.19
5	30%	21	T.O	628.59

Table 3. Number of patterns, runtime (seconds), and memory consumption (MB) on Auth_failure_1000 with *mingap=0*, *maxgap=0*, *minlen=2*, and *maxlen=6*. (a) by varying minsup and (b) by varying mincr. T.O. means time-out

(a)

minsup	mincr	#pat	time	memory
10%	1	6,204	T.O	1,750.6
20%	1	5,630	2,219.28	2,291.09
30%	1	2,556	1,985.18	1,696.91
40%	1	2,126	1,754.92	2,122.04
50%	1	1,385	1,105.34	1,746.43

(b)

mincr	minsup	#pat	time	memory
1	30%	2,256	1,993.68	1,696.91
2	30%	357	T.O	1,487.53
3	30%	488	T.O	1,543.18
4	30%	322	T.O	1,465.94
5	30%	297	T.O	1,464.66

Table 4. Number of patterns, runtime (seconds), and memory consumption (MB) on Auth_failure_1000 with *minspan=1*, *maxspan=10*, *minlen=2*, and *maxlen=6*. (a) by varying minsup and (b) by varying mincr. T.O. means time-out

(a)

minsup	mincr	#pat	time	memory
10%	1	98	137.85	1565.71
20%	1	52	89.58	1565.71
30%	1	44	85.686	1563.64
40%	1	44	91.66	1542.93
50%	1	39	87.28	1560.89

(b)

mincr	minsup	#pat	time	memory
1	30%	44	81.88	1552.45
2	30%	13	T.O	1544.85
3	30%	14	T.O	1566.44
4	30%	14	T.O	1566.79
5	30%	12	T.O	1567.12

Table 5. Number of patterns, runtime (seconds), and memory consumption (MB) on Auth_failure_1000 with *mingap=0*, *maxgap=0*, *minspan=1*, *maxspan=10*, *minlen=2*, and *maxlen=6*. (a) by varying minsup and (b) by varying mincr. T.O. means time-out

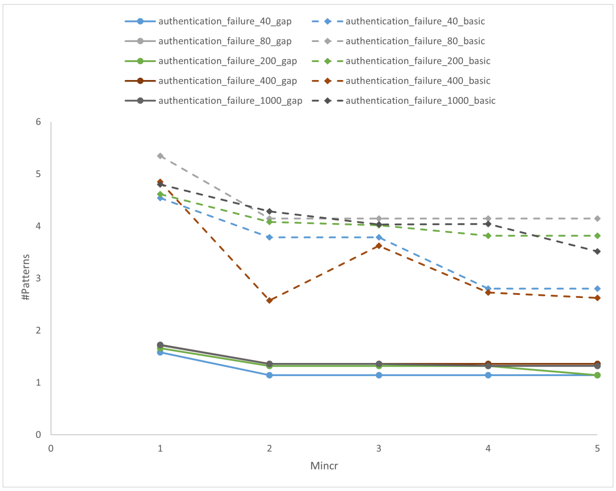

Figure 1. Comparison between basic encoding and gap constraint on extracted patterns *mingap=0*, *maxgap=0*, *minlen=2*, and *maxlen=6*. The number of patterns is in logarithmic scale to facilitate viewing and comparison

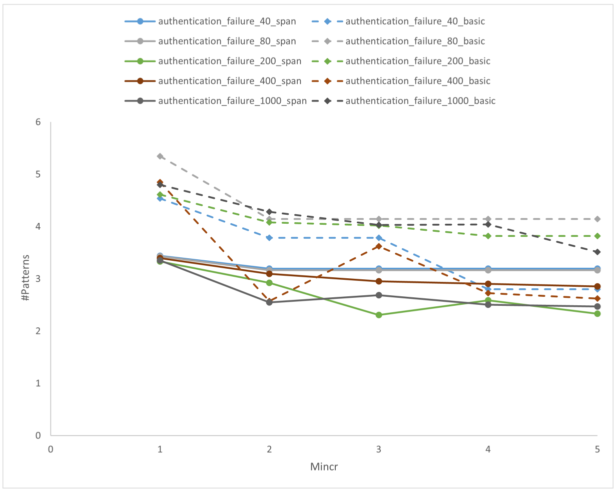

Figure 2. Comparison between basic encoding and span constraint on extracted patterns with *minspan=1*, *maxspan=10*, *minlen=2*, and *maxlen=6*. The number of patterns is in logarithmic scale to facilitate viewing and comparison

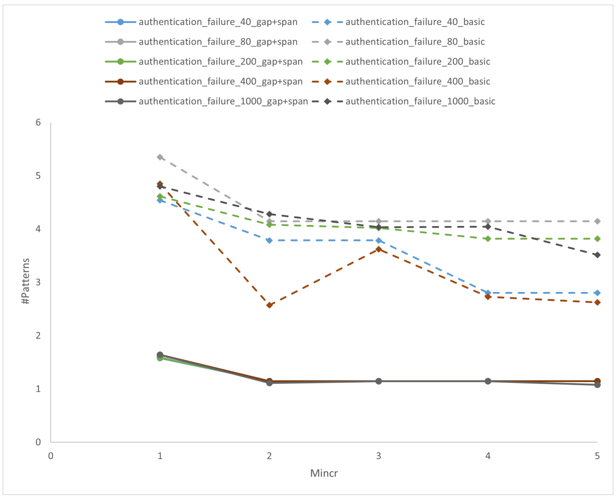

Figure 3. Comparison between basic encoding and gap+span constraint on extracted patterns with *mingap=0*, *maxgap=0*, *minspan=1*, *maxspan=10*, *minlen=2*, and *maxlen=6*. The number of patterns is in logarithmic scale to facilitate viewing and comparison

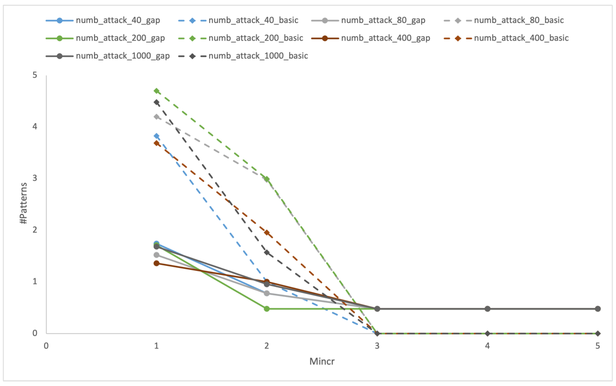

Figure 4. Comparison between basic encoding and gap constraint on extracted patterns *mingap=0*, *maxgap=0*, *minlen=2*, and *maxlen=6*. The number of patterns is in logarithmic scale to facilitate viewing and comparison

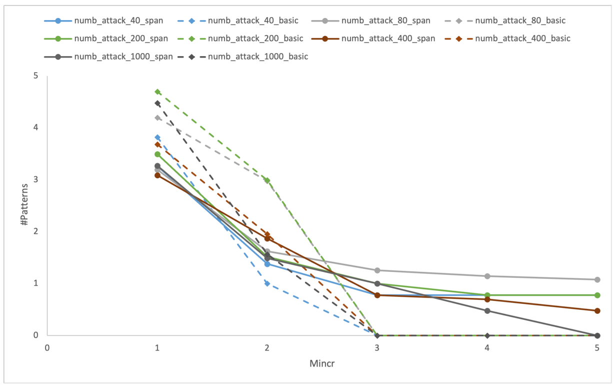

Figure 5. Comparison between basic encoding and span constraint on extracted patterns with *minspan=1*, *maxspan=10*, *minlen=2*, and *maxlen=6*. The number of patterns is in logarithmic scale to facilitate viewing and comparison

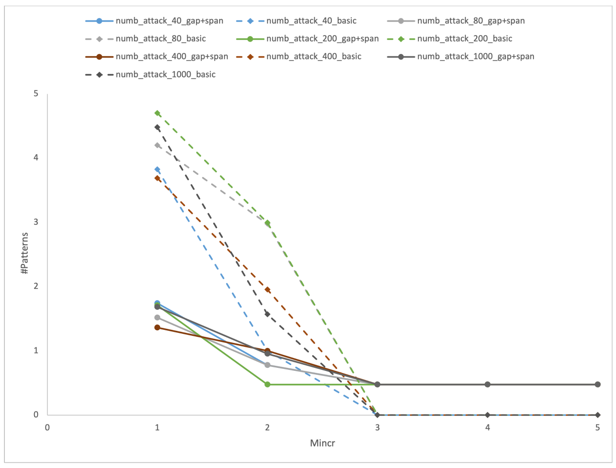

Figure 6. Comparison between basic encoding and gap+span constraint on extracted patterns with *mingap=0*, *maxgap=0*, *minspan=1*, *maxspan=10*, *minlen=2*, and *maxlen=6*. The number of patterns is in logarithmic scale to facilitate viewing and comparison