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

Abstract. This document provides the complete ASP encoding for the contrast sequential pattern mining task along with the experimental results in tabular form.

1 ASP encoding

1.1 CSPM

The full ASP encoding for CSPM is reported below in Listing 1. 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. ASP encoding for finding contrast sequential patterns

```
1  item(I) :- seq(_, _, I).
2
3  % FREQUENT SEQUENTIAL PATTERN
4  %% sequential pattern generation
5  patpos(1).
6  { patpos(X+1) } :- patpos(X), X < maxlen.
7  patlen(L) :- patpos(L), not patpos(L+1).
8  :- patlen(L), L < minlen.
9
10 1 {pat(X, I): item(I)} 1 :- patpos(X).
11
12 %% pattern embeddings
13 occ(T, 1, P) :- seq(T, P, I), pat(1, I).
14 occ(T, L, P) :- occ(T, L,   P-1), seq(T, P, _).
15 occ(T, L, P) :- occ(T, L-1, P-1), seq(T, P, C),
16                 pat(L, C).
17
18 %% frequency constraint
19 seqlen(T, L) :- seq(T, L, _), not seq(T, L+1, _).
20 support(T) :- occ(T, L, LS), patlen(L),
21               seqlen(T, LS).
22 :- { support(T) } < minsup.
23
24 %% how many sequences contains pattern
25 len_support(N) :- N = #count{T : support(T)}.
26
27 % CONTRAST SEQUENTIAL PATTERN
28 % D1 cardinality
29 card(Card, c1) :- Card = #count{T : cl(T, c1)}.
30
31 % D2 cardinality
32 card(Card, c2) :- Card = #count{T : cl(T, c2)}.
33
34 % C1 support
35 sup(Sup, c1) :- Sup = #count{T : support(T),
36 seq(T, _, _), cl(T, c1)}.
37
38 % C2 support
39 sup(Sup, c2) :- Sup = #count{T : support(T),
40 seq(T, _, _), cl(T, c2)}.
41
42 % GR_c1(X):
43 % inf if sup(X, c2)=0 and sup(X, c1)!=0
44 gr_rate("inf", c1) :- sup(Sup1, c1),
45 Sup1 != 0, sup(0, c2).
46
47 % (sup(X, c1)/|D1|)/(sup(X, c2)/|D2|)
48 % otherwise
49 gr_rate(@gr(Sup1, Card1, Sup2, Card2), c1) :-
50 sup(Sup1, c1), card(Card1, c1), sup(Sup2, c2),
51 card(Card2, c2), Sup1 > 0, Sup2 > 0.
52
53 % GR_c2(X)
54 % "inf" if sup(X, c1)=0 and sup(X, c1)!=0
55 gr_rate(inf, c2) :- sup(Sup2, c2),
56 Sup2 != 0, sup(0, c1)
57
58 % (sup(X, c2)/|D2|)/(sup(X, c1)/|D1|)
59 % otherwise
60 gr_rate(@gr(Sup2, Card2, Sup1, Card1), c2) :-
61 sup(Sup1, c1), card(Card1, c1), sup(Sup2, c2),
62 card(Card2, c2), Sup1 > 0, Sup2 > 0.
63
64 contr_pat(yes, Class) :- gr_rate(inf, Class).
65 contr_pat(@csp(Cr, mincr), Class) :- gr_rate(Cr,
66 Class), Cr != inf
67
68 % delete a pattern if it does not contrast
69 % with any class
70 :- contr_pat(no, c1), contr_pat(no, c2).
```

1.2 CSPM constraints

Below the constraints described in the article to be replaced in the pattern embedding section of the ASP encoding for CSPM reported in Listing 1 (Lines 12-16).

Listing 2. 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 3. 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.
```

captioncaption

```
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.
```

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

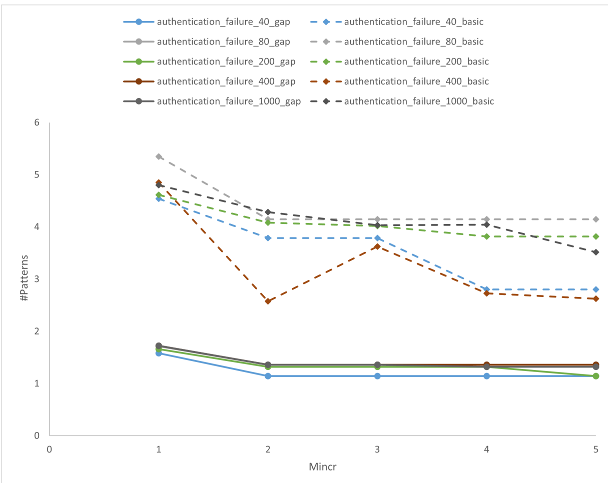

Figure 1. Comparison between basic encoding and gap constraint on extracted patterns *mingap=0*, *maxgap=0*, *minlen=2*, and *maxlen=6*. The number of patterns is in logarithmic scale to facilitate viewing and comparison

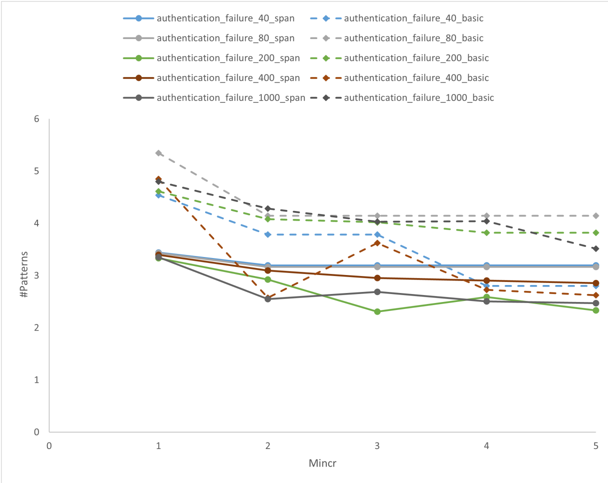

Figure 2. Comparison between basic encoding and span constraint on extracted patterns with *minspan=1*, *maxspan=10*, *minlen=2*, and *maxlen=6*. The number of patterns is in logarithmic scale to facilitate viewing and comparison

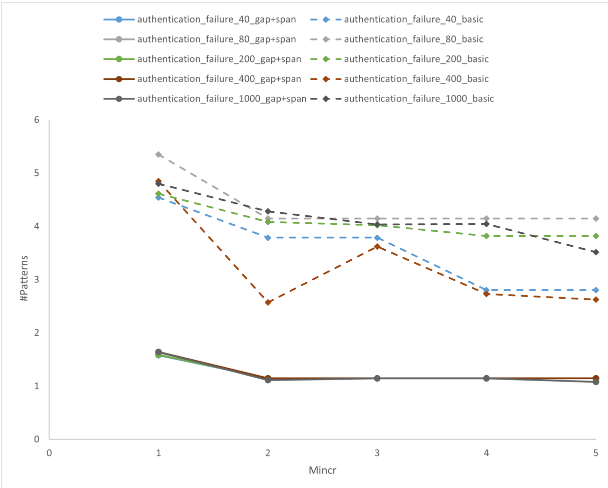

Figure 3. Comparison between basic encoding and gap+span constraint on extracted patterns with *mingap=0*, *maxgap=0*, *minspan=1*, *maxspan=10*, *minlen=2*, and *maxlen=6*. The number of patterns is in logarithmic scale to facilitate viewing and comparison

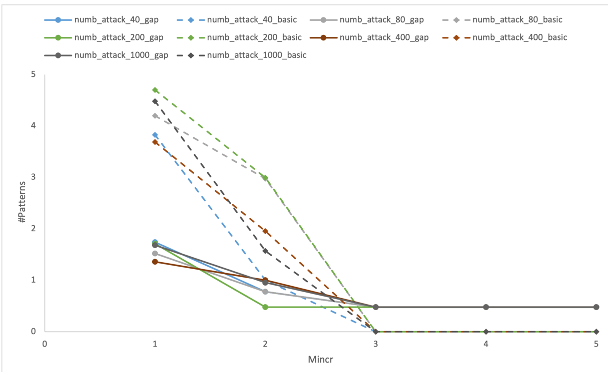

Figure 4. Comparison between basic encoding and gap constraint on extracted patterns *mingap=0*, *maxgap=0*, *minlen=2*, and *maxlen=6*. The number of patterns is in logarithmic scale to facilitate viewing and comparison

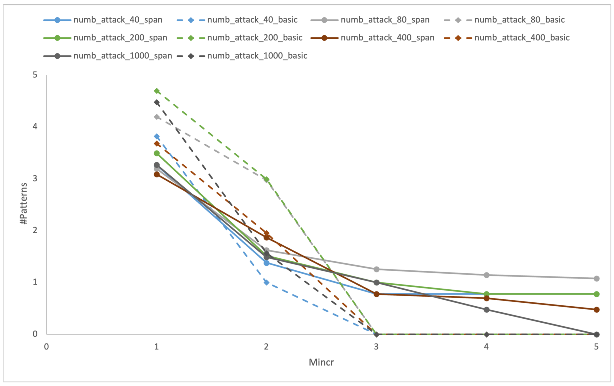

Figure 5. Comparison between basic encoding and span constraint on extracted patterns with *minspan=1*, *maxspan=10*, *minlen=2*, and *maxlen=6*. The number of patterns is in logarithmic scale to facilitate viewing and comparison

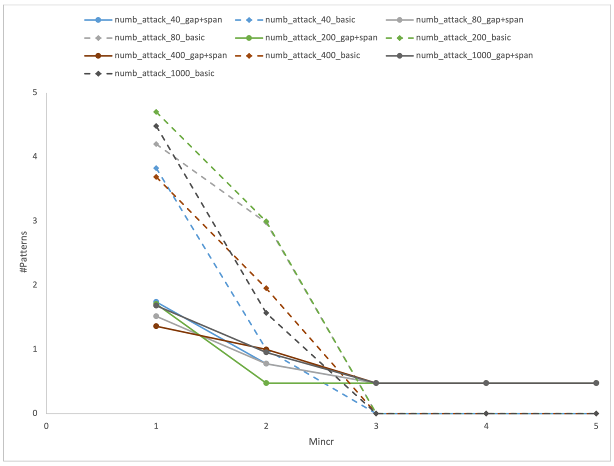

Figure 6. Comparison between basic encoding and gap+span constraint on extracted patterns with *mingap=0*, *maxgap=0*, *minspan=1*, *maxspan=10*, *minlen=2*, and *maxlen=6*. The number of patterns is in logarithmic scale to facilitate viewing and comparison