Negative Sequence Analysis: A Review

WEI WANG, University of Technology Sydney
LONGBING CAO*, University of Technology Sydney

Negative sequential patterns (NSPs) produced by negative sequence analysis (NSA) capture more informative and actionable knowledge than classic positive sequential patterns (PSPs) due to involving both occurring and non-occurring items, which appear in many applications. However, the research on NSA is still at an early stage and NSP mining involves very high computational complexity and a very large search space, there is no widely accepted problem statement on NSP mining, and different settings on constraints and negative containment have been proposed in existing work. Among existing NSP mining algorithms, there are no general and systemic evaluation criteria available to assess them comprehensively. This paper conducts a comprehensive technical review of existing NSA research. We explore and formalize a generic problem statement of NSA, investigate, compare and consolidate the definitions of constraints and negative containment, and compare the working mechanisms and efficiency of existing NSP mining algorithms. The review is concluded by discussing new research opportunities in NSA.

CCS Concepts: • **General and reference → Surveys and overviews**;

Additional Key Words and Phrases: Negative Sequential Pattern Mining, Negative Sequence Analysis, Non-occurring Behavior Analysis, Behavior Analytics

ACM Reference Format:
Wei Wang and Longbing Cao. 2010. Negative Sequence Analysis: A Review. *ACM Comput. Surv.* 9, 4, Article 39 (March 2010), 14 pages. https://doi.org/0000001.0000001

A SUPPLEMENTARY MATERIALS

In this supplementary material, the representative NSP mining algorithms are evaluated on 35 synthetic sequence datasets, based on the evaluation criteria listed in Table 1, which are proposed to evaluate NSP mining algorithms in terms of three evaluation aspects, *NSP count*, *NSP runtime* and *Total length of NSCs*.

A.1 Synthetic Datasets for Evaluation

35 synthetic sequence datasets are used for this evaluation, which are generated by the IBM data generator [1]. Table 2 describes these datasets in terms of different data factors. The base dataset is $C10\_T6\_S8\_I8\_DB10k\_N0.1k$. C is the average number of elements per sequence; T is the average number of items per element; S is the average length of potentially maximal frequent positive sequences; I is the average number of items per element in potentially maximal frequent positive sequences; DB is the number of data sequences in a sequence dataset; N is the number of different items.

*Corresponding author.

Authors' addresses: Wei Wang, University of Technology Sydney, Wei.Wang-8@student.uts.edu.au; Longbing Cao, University of Technology Sydney, longbing.cao@uts.edu.au.

0360-0300/2010/3-ART39 $15.00
https://doi.org/0000001.0000001

Table 1. NSP Evaluation Aspects and Description

Evaluation Aspect	**Notation**	**Description**
NSP count	Nct	The number of discovered NSPs
NSP runtime	Nrt	Runtime consumed to discover NSPs
Total length of NSCs	Tlnc	The total length of the generated NSCs

Table 2. Description of Synthetic Sequence Datasets w.r.t. Data Factor Adjustment

Data Factor	**Dataset Name**	**Data Factor Adjustment**
Base Dataset	C10_T6_S8_I8_DB10k_N0.1k	$C = 10, T = 6, S = 8, I = 8, DB = 10k, N = 0.1k$
C_1	C6_T6_S8_I8_DB10k_N0.1k	$C = 6, T = 6, S = 8, I = 8, DB = 10k, N = 0.1k$
C_2	C8_T6_S8_I8_DB10k_N0.1k	$C = 8, T = 6, S = 8, I = 8, DB = 10k, N = 0.1k$
C_3	C12_T6_S8_I8_DB10k_N0.1k	$C = 12, T = 6, S = 8, I = 8, DB = 10k, N = 0.1k$
C_4	C14_T6_S8_I8_DB10k_N0.1k	$C = 14, T = 6, S = 8, I = 8, DB = 10k, N = 0.1k$
T_1	C10_T4_S8_I8_DB10k_N0.1k	$C = 10, T = 4, S = 8, I = 8, DB = 10k, N = 0.1k$
T_2	C10_T8_S8_I8_DB10k_N0.1k	$C = 10, T = 8, S = 8, I = 8, DB = 10k, N = 0.1k$
T_3	C10_T10_S8_I8_DB10k_N0.1k	$C = 10, T = 10, S = 8, I = 8, DB = 10k, N = 0.1k$
T_4	C10_T12_S8_I8_DB10k_N0.1k	$C = 10, T = 12, S = 8, I = 8, DB = 10k, N = 0.1k$
S_1	C10_T6_S4_I8_DB10k_N0.1k	$C = 10, T = 6, S = 4, I = 8, DB = 10k, N = 0.1k$
S_2	C10_T6_S6_I8_DB10k_N0.1k	$C = 10, T = 6, S = 6, I = 8, DB = 10k, N = 0.1k$
S_3	C10_T6_S10_I8_DB10k_N0.1k	$C = 10, T = 6, S = 10, I = 8, DB = 10k, N = 0.1k$
S_4	C10_T6_S12_I8_DB10k_N0.1k	$C = 10, T = 6, S = 12, I = 8, DB = 10k, N = 0.1k$
I_1	C10_T6_S8_I4_DB10k_N0.1k	$C = 10, T = 6, S = 8, I = 4, DB = 10k, N = 0.1k$
I_2	C10_T6_S8_I6_DB10k_N0.1k	$C = 10, T = 6, S = 8, I = 6, DB = 10k, N = 0.1k$
I_3	C10_T6_S8_I10_DB10k_N0.1k	$C = 10, T = 6, S = 8, I = 10, DB = 10k, N = 0.1k$
I_4	C10_T6_S8_I12_DB10k_N0.1k	$C = 10, T = 6, S = 8, I = 12, DB = 10k, N = 0.1k$
DB_1	C10_T6_S8_I8_DB20k_N0.1k	$C = 10, T = 6, S = 8, I = 8, DB = 20k, N = 0.1k$
DB_2	C10_T6_S8_I8_DB30k_N0.1k	$C = 10, T = 6, S = 8, I = 8, DB = 30k, N = 0.1k$
DB_3	C10_T6_S8_I8_DB40k_N0.1k	$C = 10, T = 6, S = 8, I = 8, DB = 40k, N = 0.1k$
DB_4	C10_T6_S8_I8_DB50k_N0.1k	$C = 10, T = 6, S = 8, I = 8, DB = 50k, N = 0.1k$
N_1	C10_T6_S8_I8_DB10k_N0.2k	$C = 10, T = 6, S = 8, I = 8, DB = 10k, N = 0.2k$
N_2	C10_T6_S8_I8_DB10k_N0.3k	$C = 10, T = 6, S = 8, I = 8, DB = 10k, N = 0.3k$
N_3	C10_T6_S8_I8_DB10k_N0.4k	$C = 10, T = 6, S = 8, I = 8, DB = 10k, N = 0.4k$
N_4	C10_T6_S8_I8_DB10k_N0.5k	$C = 10, T = 6, S = 8, I = 8, DB = 10k, N = 0.5k$

A.2 Experimental Result and Analysis

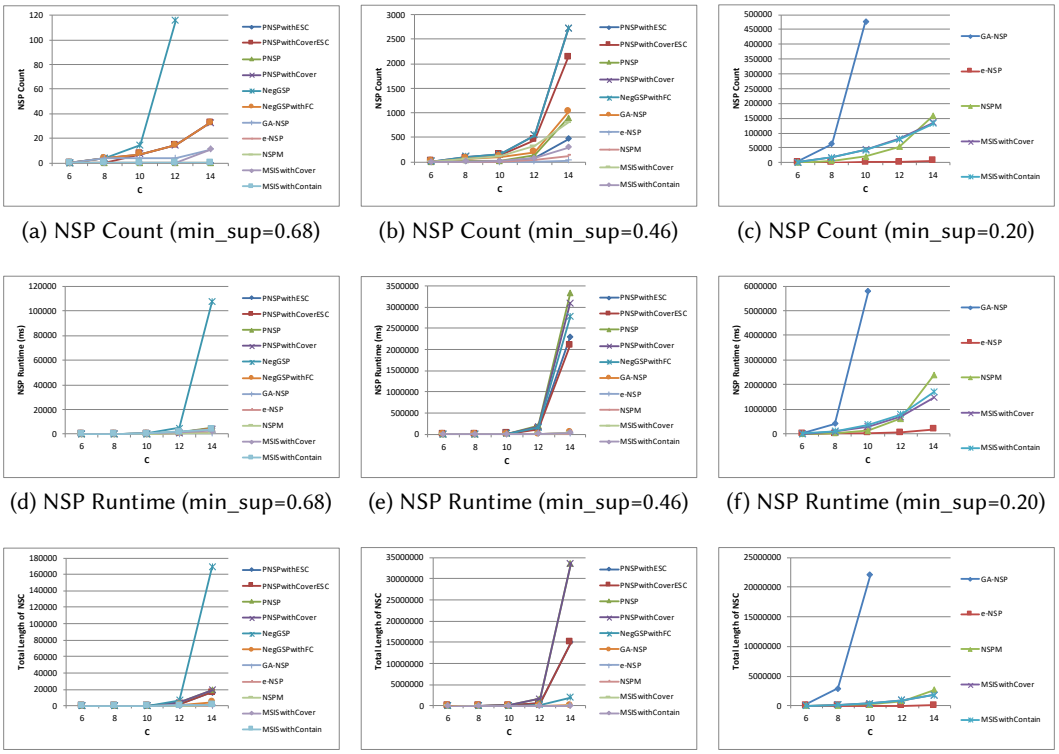
Below, we evaluate the algorithms by adjusting the corresponding data factors. It is noted that GA-NSP is implemented with the crossover rate at 100%, mutation rate at 10% and decay rate at 10%. In the following figures, the X-axis stands for the value of minimum support and the Y-axis stands for the value of the evaluation criteria, the unit of NSP runtime (*Nrt*) is the millisecond,

and the unit of NSP count (*Nct*) and total length of NSCs (*Tlnc*) is one. Finally, since difference algorithms aim to discover NSP in divergent search space, their value of the evaluation criteria is in different orders of magnitude. For the convenience of display, we select three thresholds, i.e., high, medium and low thresholds, to present the experimental results respectively.

A.2.1 Algorithm Efficiency Analysis w.r.t. C. The NSP count *Nct* for different NSP mining algorithms on factor C under divergent thresholds are shown in Figures 1a, 1b and 1c. Here we set the high threshold as 0.68, because all evaluation aspects of NegGSP experience an exponential rapid on big C and will be in different orders of magnitude with those of other algorithms under lower thresholds. Therefore, we show the evaluation results of NegGSP under $min\_sup = 0.68$ as high threshold. In addition, we set medium threshold as 0.46 to show the results of PNSP-based variants and NegGSPwithFC, and set low threshold as 0.20 to show those of other algorithms.

We can see that as factor C increases from 6 to 14, the NSP count *Nct* for most algorithms increases under all thresholds. when threshold is relatively high, as illustrated in Figure 1a, these algorithms discover similar number of NSP when factor C is less than 8, but when C is larger than 12, the *Nct* for most algorithms increases drastically while e-NSP maintains a relatively stable *Nct*. Among these algorithms, the *Nct* for NegGSP experiences the fastest increase since it adopts a loose FreC, IFC, and thus can discover much more number of NSP than other algorithms even when threshold is as high as 0.68. Since adopting the same constraint and negative containment, EFC and negative containment, PNSPwithCover and NegGSPwithFC have same *Nct* when $min\_sup = 0.46$ and C is less than 12, however, when C get higher, *Nct* for NegGSPwithFC is a little less than that of PNSPwithCover suffering from its joining-base NSC generation strategy. Comparing PNSPwithCover with PNSPwithCoverESC and PNSP, it can be seen that both of the two latter algorithms discover much less NSP since they adopt a stricter constraint, ESC, and a stricter containment, N-containment, respectively. Relatively speaking, PNSP can mine only less than half number of NSP compared with PNSPwithCoverESC, which means that the impact of N-containment on the *Nct* for PNSP is much more than that of ESC. And PNSPwithESC which adopts ESC and N-containment can only discover almost half number of NSP than PNSP. As is seen in Figure 1b, with C increasing from 8 to 10, the *Nct* for PNSPwithESC, PNSP and e-NSP decreases when $min\_sup = 0.46$, because adopting strict negative containment, such as N-containment and strictly-negative containment, may break the positive correlation between *Nct* and factor C. when threshold is low ($min\_sup = 0.20$), MSISwithContain can discover more NSP than NSPM when C is less than 12, shown in Figure 1c, since more IPS can be generated when threshold is low and thus more 2-neg-size NPS can be found by MSISwithContain, which cannot be discovered by NSPM since constraint CFC is adopted. However, when factor is larger than 14, the *Nct* for NSPM surpasses that of MSISwithContain since more long-size NSP are discovered by NSPM.

The NSP runtime *Nrt* for different algorithms on factor C under divergent thresholds is shown in Figures 1d, 1e and 1f. When threshold is high ($min\_sup = 0.68$), similar with its *Nct*, the *Nrt* for NegGSP is similar with that of PNSPwithCover, PNSP, PNSPwithCoverESC and PNSPwithESC when C is less than 10, but then increases much rapidly since search space get larger dramatically, as illustrated in Figure 1d. When $min\_sup = 0.46$, shown in Figure 1e, even though PNSP can only discover a quarter of NSP than PNSPwithCover, the *Nrt* for PNSP is slightly more than that of PNSPwithCover, since PNSP generates and tests same set of NSC as PNSPwithCover and needs to compare data sequences with not only each NSC but also its maximum positive subsequence because of adopting N-containment. Compared with PNSPwithCover, PNSPwithCoverESC consumes much less runtime to mine NSP since adopting ESC shrinks the set of NSC and seeds and less NSC need to be tested. As for NegGSPwithFC, its *Nrt* is less than the *Nrt* for PNSPwithCover because joining-based strategy generates NSC length by length and avoids the generation of some



(a) NSP Count (min_sup=0.68) (b) NSP Count (min_sup=0.46) (c) NSP Count (min_sup=0.20)

(d) NSP Runtime (min_sup=0.68) (e) NSP Runtime (min_sup=0.46) (f) NSP Runtime (min_sup=0.20)

(g) Total Length of NSC (min_sup=0.68) (h) Total Length of NSC (min_sup=0.46) (i) Total Length of NSC (min_sup=0.20)

Fig. 1. Algorithm Efficiency Analysis w.r.t. C (X-axis stands for C, and Y-axis stands for algorithm efficiency)

invalid NSC. But compared with PNSPwithESC and PNSPwithCoverESC, NegGSPwithFC still has a larger *Nrt* because it generates more NSC. In addition, when $min\_sup = 0.20$, shown in Figure 1f, the increasing trend of NSPM is faster than that of MSISwithCover and MSISwithContain, and NSPM consumes much more runtime when C gets higher, since NSPM can generate long-size NSC and is required to conduct more comparison when testing NSC with the increase of C. It can be seen that the number of generated NSC and comparison times are two major attributes impacting *Nrt*, and thus to design a strategy which generates less NSC or can test the generated NSC by less comparison is an effective path to improve the efficiency of NSP mining algorithms.

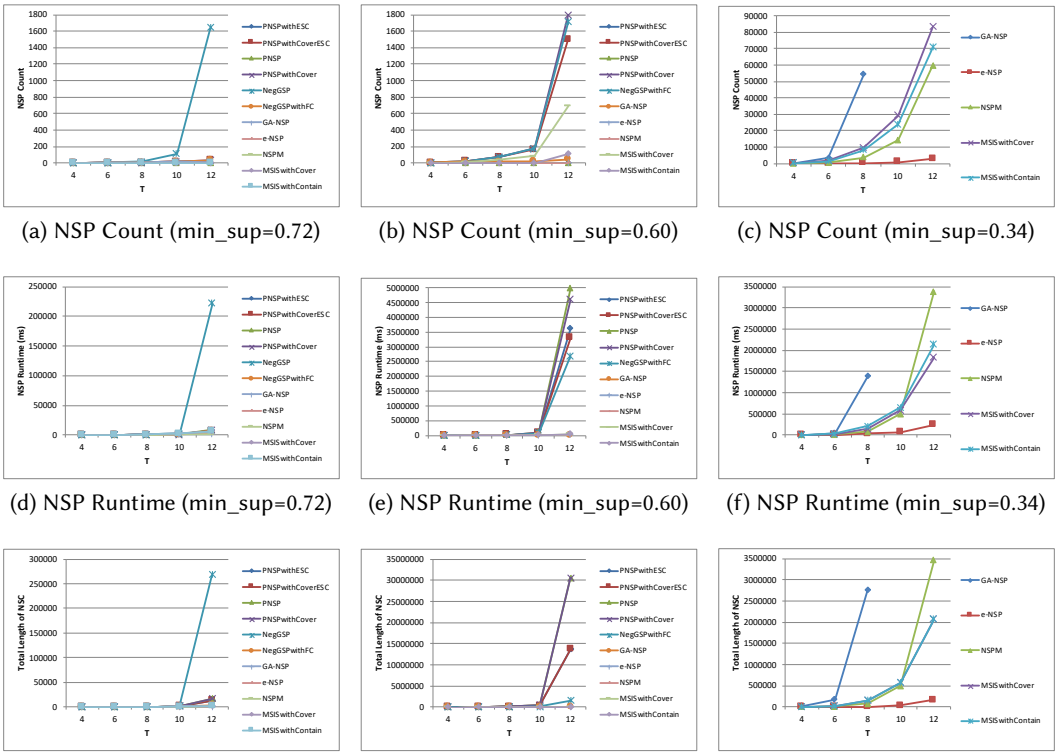
The total length of NSCs *Tlnc* for mining algorithms on factor C under different thresholds is illustrated in Figures 1g, 1h and 1i. When $min\_sup = 0.68$, PNSPwithESC, PNSPwithCoverESC, PNSP, PNSPwithCover, NegGSP and NegGSPwithFC consume similar memory space to save the generated NSC and the *Tlnc* for these algorithms increases smoothly with factor C rising from 6 to 10, and then rises sharply as C gets increased. As shown in Figure 1g, PNSPwithESC has the same *Tlnc* with PNSPwithCoverESC, which is much less than that of PNSP and PNSPwithCover when C is high. That is because that adopting different negative containment cannot cause the change of seeds and thus has no impact on the set of generated NSC, which can also be seen by comparing MSISwithCover and MSISwithContain. And when cover pruning strategy is adopted, some constraints can result in the decrease of the base-support of NSC and thus cause a smaller

seed set, as a result fewer long-size NSC are generated and less memory usage is required. Similarly, compared with NegGSP, NegGSPwithFC maintains a far less *Tlnc* when C is larger than 12, because it adopts EFC and avoids generating amount of invalid NSC defying EFC, especially for less long-size NSC. As shown in Figure 1h, as threshold *min_sup* get decreased to 0.46, the *Tlnc* difference of PNSPwithESC and PNSP is get much larger, and when factor C gets larger than 12, PNSPwithESC only requires less than half of the memory usage of PNSP. As shown in Figure 1i, although GA-NSP tends to generate high potential NSC in a relatively smaller search, when threshold is low, it still consumes much more memory compared with NSPM and MSIS. In addition, e-NSP consumes the smallest memory and keeps a better scalability in all thresholds. It is clear that adopting effective constraints and pruning strategy to reduce the number of generated NSC can efficiently decrease the memory space required in the mining process, especially when threshold is low.

A.2.2 Algorithm Efficiency Analysis w.r.t. T. Here we characterize the efficiency of these algorithms on T. We set the high threshold as 0.72 to show the evaluation of NegGSP, set the medium threshold as 0.60 to show that of PNSP-based variants and NegGSPwithFC, and set the low threshold as 0.34 to show the results of other algorithms.

Figures 2a, 2a and 2c show the number of NSP mined by algorithms under different thresholds and T. We can see from Figure 2a that four global mining algorithms with MPS-based negative containment, PNSPwithCoverESC, PNSPwithCover, NegGSP and NegGSPwithFC, experience a trend towards higher *Nct* as T increases. Though this increasing trend looks similar with that in Figure 1a, but it is achieved in a much higher threshold and the *Nct* for NegGSP grows much more rapidly when T gets to 12, because as each element in data sequences has more items a NSC may be contained by more data sequences and further have a higher support. From Figure 2b indicating algorithms' *Nct* under medium threshold, we note that NegGSPwithFC loses more NSPs when T gets to 12 compared with PNSPwithCover, because as T grows dataset gets more denser and NSCs involving the elements larger in size have potentially higher support, but they may not be generated by NegGSP suffering from the adoption of Cover Pruning Strategy and joining-based NSC generation . In addition, even if *Nct* for PNSPwithCoverESC is still lower than that of PNSPwithCover, it obtains a greater proportion of NSP compared with that on C, since an increasing T does not lead to the rising in size of NSP and ESC mainly works in filtering long-size NSC. Different from that in Figures 1a, 1b and 1c, two N-containment-based variants of PNSP, PNSPwithESC and PNSP discover no NSP in relatively medium threshold suffering from the mechanism requiring the maximum positive sub-sequence of a NSP not supported by any data sequence negative supporting it. Furthermore, Figure 2c demonstrates that under low threshold ($min\_sup = 0.34$), two variants of MSIS, MSISwithCover and MSISwithContain, have a much less difference in *Nct* and both of them can discover much more NSP than NSPM, because with dataset getting denser MSIS can generate more NSC from IPS. In contrast to other algorithms having a clearly increasing *Nct* w.r.t T, e-NSP maintains a stable but low *Nct* in most situation.

Figures 2d, 2e and 2f show the algorithm *Nrt* on T. As seen from Figure 2d, the *Nrt* for NegGSP is much higher than that of others and rises exponentially as T increases even under a threshold as high as 0.72. Figure 2e illustrates that even though NegGSPwithFC keeps almost same *Nct* as PNSPwithCover, it consumes nearly half runtime when T is 12. Different from the cases on C, NegGSPwithFC takes less runtime than that of all the PNSP-based variants with ESC, because the growth of T does not cause PNSP generating long-size NSC and ESC leaves less influence on reducing NSC on large T. Another observation is that Figure 1e shows *Nrt* for NSPM increases gradually as C increases and NSPM takes less *Nrt* than e-NSP under big C, while Figure 2e shows NSPM has a rapidly increasing *Nrt* with T growing and takes approximately twice the runtime consumed by e-NSP when T is greater than 10. That is because as average size of elements in



(a) NSP Count (min_sup=0.72) (b) NSP Count (min_sup=0.60) (c) NSP Count (min_sup=0.34)

(d) NSP Runtime (min_sup=0.72) (e) NSP Runtime (min_sup=0.60) (f) NSP Runtime (min_sup=0.34)

(g) Total Length of NSC (min_sup=0.72) (h) Total Length of NSC (min_sup=0.60) (i) Total Length of NSC (min_sup=0.34)

Fig. 2. Algorithm Efficiency Analysis w.r.t. T (X-axis stands for T, and Y-axis stands for algorithm efficiency)

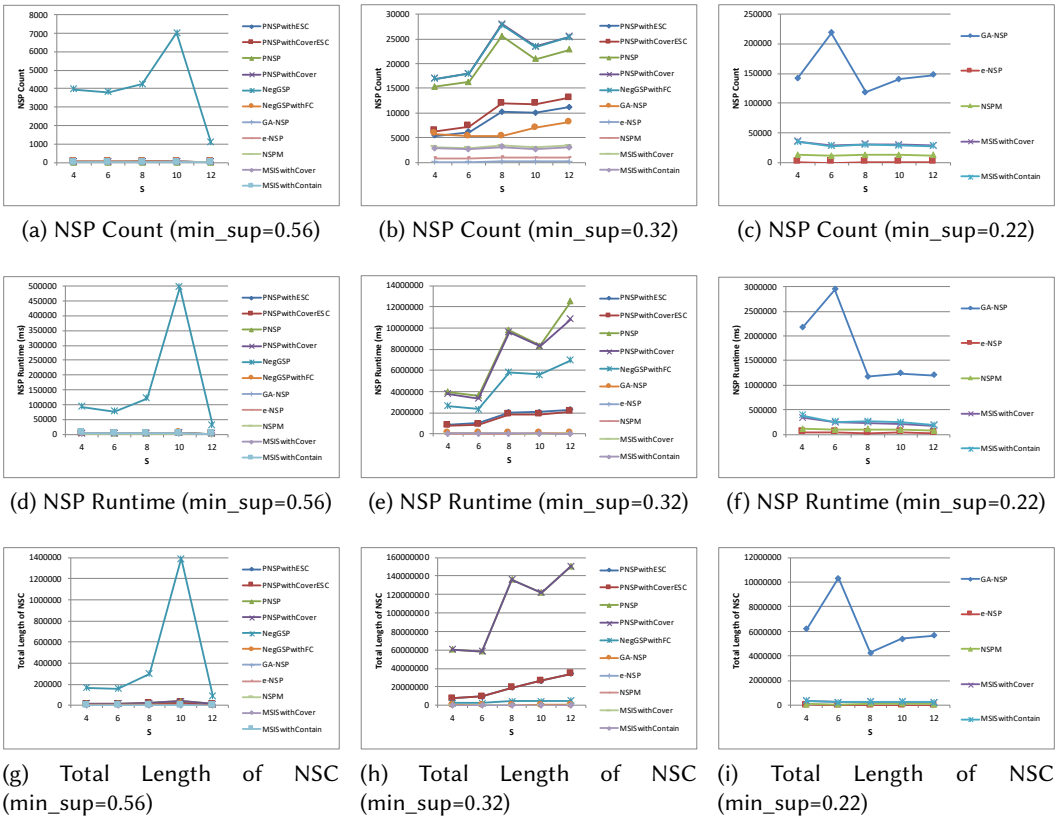
data sequences grows NSPM generates more long-length NSC and thus takes more runtime. This phenomenon is much more obvious in Figure 2f, which indicates under low threshold NSPM consumes much more runtime than two MSIS-based variants, even if it obtains much less NSP.

Figures 2g, 2h and 2i reveal the total length of NSCs of algorithms on T. Similar with the cases of *Tlnc* on C, the *Tlnc* for each algorithm shows a clear growing trend as T is increased, and NegGSP maintains the *Tlnc* most drastically, followed by two variants of PNSP, PNSPwithCover and PNSP, followed by another two PNSP-based variants with ESC, PNSPwithCoverESC as well as PNSPwithESC, and followed by NegGSPwithFC. Among the algorithms targeting at mining a subset of NSP, two MSIS-based variants, MSISwithContain and MSISwithCover, consume the maximum amount of memory space, and followed by NSPM and e-NSP. In addition, we note that the increasing trend of *Tlnc* on T is much rapidly than that on C.

A.2.3 Algorithm Efficiency Analysis w.r.t. S. Here we analyse the algorithms efficiency on S. The high threshold is set as 0.56 to show the evaluation of NegGSP, medium threshold is set as 0.32 to show that of PNSP-based variants and NegGSPwithFC, and low threshold is set as 0.22 to show the results of other algorithms.

Figures 3a, 3b and 3c shows the *Nct* for these algorithms on S, which illustrates a non-monotonic and relatively gentle trend of most algorithms. The reason is that since NSP do not satisfy downward

property and the length of NSC can be far much greater than that of PSP, the increasing S will not lead to a monotonic growth in the maximum length of NSP. We can note from Figure 3a that compared with other algorithms, NegGSP always discovers a far greater number of NSP because it adopts a loose ISC constraint and is required to generate NSC in a much larger search space than others. Figure 3b shows when threshold is 0.56, NegGSPwithFC and PNSPwithCover obtain almost the same and maximum number of NSP, followed by PNSP, and two PNSP-based variants with ESC discovers only nearly half number of NSP as that of PNSPwithCover, and that is because a given S and I leaves a restriction on the size of data sequence and the adoption of ESC filters many long-size NSC. Among other algorithms, GA-NSP can always find more NSP than two MSIS-based algorithms, both of which mine more than NSPM. And e-NSP discovers minimum NSP and is the least sensitive to the changing of S. The same trend of *Nct* can be also illustrated in Figure 3c.



(a) NSP Count (min_sup=0.56) (b) NSP Count (min_sup=0.32) (c) NSP Count (min_sup=0.22)

(d) NSP Runtime (min_sup=0.56) (e) NSP Runtime (min_sup=0.32) (f) NSP Runtime (min_sup=0.22)

(g) Total Length of NSC (min_sup=0.56) (h) Total Length of NSC (min_sup=0.32) (i) Total Length of NSC (min_sup=0.22)

Fig. 3. Algorithm Efficiency Analysis w.r.t. S (X-axis stands for S, and Y-axis stands for algorithm efficiency)

Figures 3d, 3e and 3f show the *Nrt* for mining algorithms on S under thresholds. Once again NegGSP is the most time-consuming NSP mining algorithm, suffering from its exponentially increasing search space. Similar with the previous ranks, PNSPwithCover and NegGSPwithFC consume similar runtime, which is a little less than those of PNSP and PNSPwithESC and slightly more than that of PNSPwithCoverESC. In contrast with the increasing trend in high thresholds, we note that the *Nrt* for NSPM and two MSIS-based variants keeps decreasing with S growing, and the *Nrt* for these algorithms on $S = 12$ is almost half of that on $S = 4$. From Figure 3e we can see that

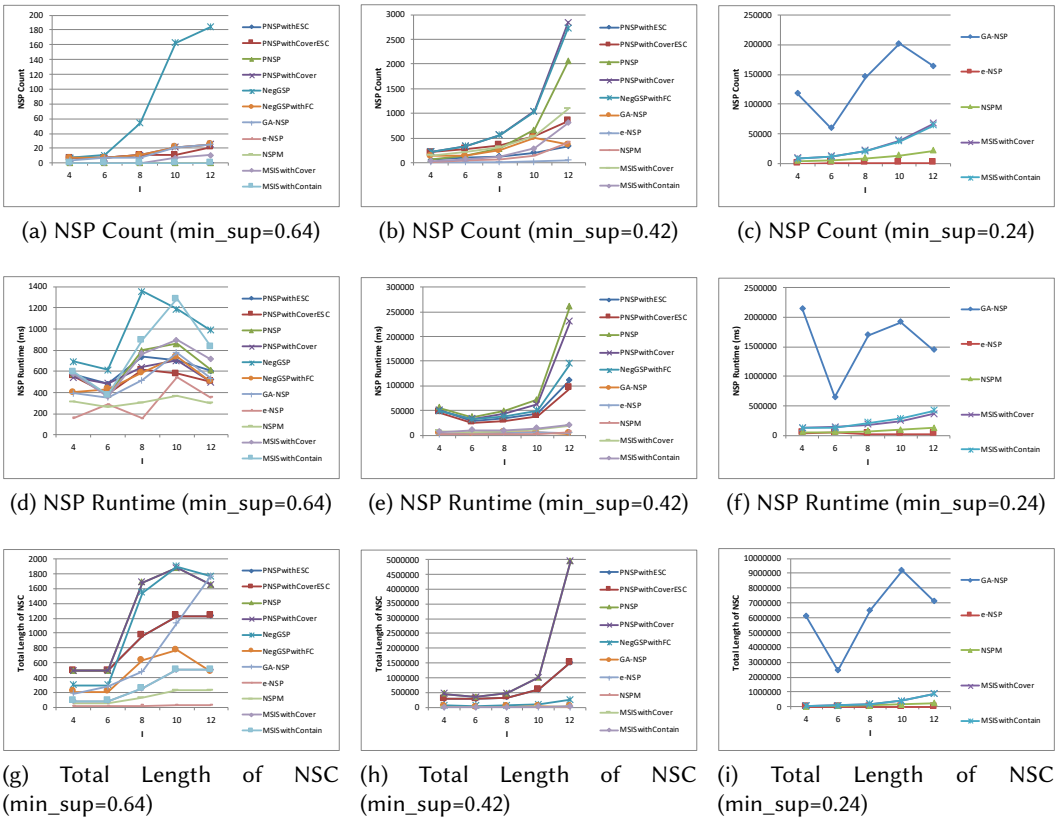
quite different from the *Nrt* performance under high threshold, when threshold gets to 0.32, the *Nrt* for NegGSPwithFC only occupies two thirds of that of PNSPwithCover, which is no less than that of PNSP. Another similar observation is that PNSPwithCoverESC takes a quite similar *Nrt* as that of PNSPwithESC, which are both clearly less than those of previous two PNSP-based variants without ESC and means the adoption of negative containment leaves a much less impact on *Nrt* compared with that of ESC on large S. Another interesting observation is that, different from the phenomenon on C and T, NegGSPwithFC consumes a dramatically higher runtime than those of PNSPwithESC and PNSPwithCoverESC. Figure 3f shows that, on the contrary to the monotonically increasing trend, all the four algorithms targeting at discovering particular subset of NSP experience a rough decrease as S goes down, and the *Nrt* for e-NSP is significantly less than those of other algorithms.

Figures 3g, 3h and 3i illustrate the total length of NSC of algorithms on S. As shown in Figure 3g, *Tlnc* ranks on S under high threshold ($min\_sup = 0.56$) are similar with those on T, and NegGSP achieves a highest *Tlnc*, followed by four PNSP-based variantsm, and also followed by NegGSPwithFC, which consumes only one third of memory usage of two PNSP-based variants with ESC and one fifth of PNSP and PNSPwithCover. Compared with the high *Tlnc* for NegGSP and PNSP-based variants, GA-NSP has a much lower *Tlnc*. Different from the dramatic changes of *Tlnc*, e-NSP, NSPM and two MSIS-based variants keep a relatively stable scalability on *Tlnc*. Figure 3h demonstrates that, when threshold gets to 0.32, *Tlnc* for PNSP and PNSPwithCover is far more than that of PNSPwithESC and PNSPwithCoverESC. Comparatively speaking, the PNSP-based algorithms with ESC can discover more than half number of NSP mined by PNSP and PNSPwithCover, but can reduce almost 80% memory usage and computation runtime, which shows that the adoption of ESC can be regarded as an effective design to save resource consumption when mining NSP on datasets with big S. Compared with the obvious growth on *Tlnc*, e-NSP, NSPM and two MSIS-based variants keep relatively stable *Tlnc* on all S. In addition, we note that when S is 12, e-NSP has a slightly more *Tlnc* than NSPM, because with the average length of potentially maximum PSP getting increased much more long-length NSC can be generated by e-NSP and thus more memory usage is required. Figure 3i shows that under low threshold ($min\_sup = 0.22$) e-NSP has a far less *Tlnc* than NSPM, since the potentially maximum size of frequent negative itemsets gets increased as threshold decreases and thus NSPM generates more long-neg-size NSC and consume more memory usage.

A.2.4 Algorithm Efficiency Analysis w.r.t. I. Here we characterize the efficiency of these algorithms on I. The high threshold is set as 0.64 to illustrate the evaluation of NegGSP, medium threshold is set as 0.42 to show that of PNSP-based variants and NegGSPwithFC, and low threshold is set as 0.24 to show the results of other algorithms.

Figures 4a, 4b and 4c illustrate the number of NSP mined by different algorithms on I. Figure 4a shows that under high threshold ($min\_sup = 0.64$), *Nct* for NegGSP increases substantially with I growing from 4 to 10, and then gets rising relatively gently. In addition, compared with NegGSPwithFC and PNSPwithCover, GA-NSP keeps a similar *Nct* on all I, especially when I is greater than 10 GA-NSP discovers same number of NSP. Figure 4b indicates that under medium threshold, NegGSPwithFC fails to generate more NSP than PNSPwithCover as I grows, because with I increasing the potential size of elements in long-length NSP get increased and thus NegGSPwithFC lose more NSP on big I. Moreover, different from the case on C and S, compared with PNSP, PNSPwithCoverESC has a higher *Nct* when I is less than 8 but a much lower *Nct* then. For example, *Nct* for PNSPwithCoverESC is three times as that of PNSP when $I = 4$ while is only less than half of that of PNSP when $I = 12$. The reason is that when S is fixed, the average size of potentially maximum PSP gets declined with I increasing, and when I is small, PSP potentially have a large average size and ESC will prune less NSC and thus PNSPwithCoverESC has a higher *Nct*. When

I is large, the N-containment support count of a NSC is potentially larger and thus PNSP has a higher *Nct*. These phenomenons show that the adoption of ESC can lose more NSP on small I while N-containment filter more NSP on big I. Another observation is that on contrary to the monotonous growth of *Nct* as I increase, *Nct* for e-NSP experiences a stable status with I rising from 6 to 10 and then gets increased, since with I growing PSP have a potentially smaller average size but consisting of more long-size elements and thus e-NSP may derives no more NSC from obtained PSP when I is small. Furthermore, *Nct* for NSPM and two MSIS-based variants gets rising much more rapidly as I grows. Figure 4c demonstrates that, under low threshold, *Nct* for e-NSP, NSPM and MSIS-based variants grows monotonically with I rising. But different from that on C and T, *Nct* for MSIS-based variants get increased more and more dramatically as I grows and is more than 3 times of that of NSPM. That is because MSIS only focus on obtaining 2-size NSP and less sensitive to the size decline of data sequences, and much more NSC can be derived from IPS and supported by sequence when I is high. Among these algorithms, e-NSP keeps a gently growing and lowest *Nct* on all I.



(a) NSP Count (min_sup=0.64) (b) NSP Count (min_sup=0.42) (c) NSP Count (min_sup=0.24)

(d) NSP Runtime (min_sup=0.64) (e) NSP Runtime (min_sup=0.42) (f) NSP Runtime (min_sup=0.24)

(g) Total Length of NSC (min_sup=0.64) (h) Total Length of NSC (min_sup=0.42) (i) Total Length of NSC (min_sup=0.24)

Fig. 4. Algorithm Efficiency Analysis w.r.t. I (X-axis stands for I, and Y-axis stands for algorithm efficiency)

Figures 4d, 4e and 4f illustrate the *Nrt* for mining algorithms on I. We note from Figure 4d that, under high threshold as 0.64, on contrary to the monotonically rising *Nrt* on C and T, *Nrt* for NegGSP experiments a fluctuation and has a similar maximum *Nrt* with MSISwithContain. And opposite to its design goal of discovering NSP efficiently, GA-NSP even consumes more runtime than NegGSPwithFC when I is larger than 10. Furthermore, *Nrt* for NSPM keeps the most stable

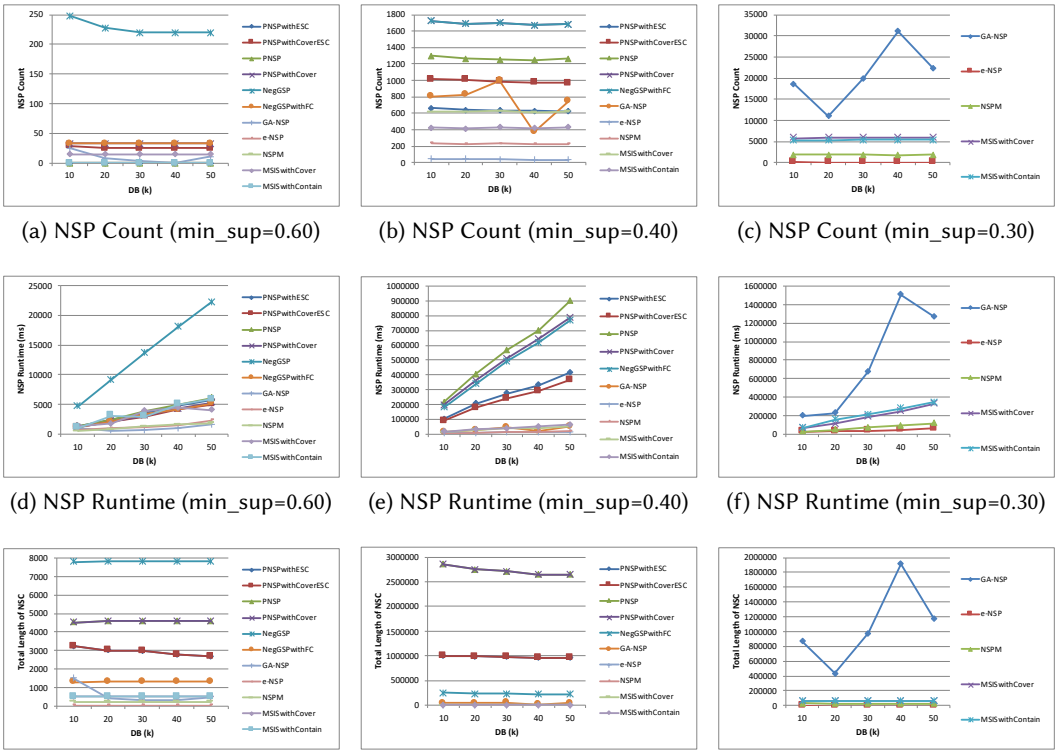
among these algorithms and is apparently lower than that of e-NSP on big I. Figure 4e indicates that when threshold gets to 0.42, *Nrt* for NegGSPwithFC and four PNSP-based variants gets decreasing at first and then increased with I growing, among which PNSPwithCover and PNSP consume the most runtime, followed by NegGSPwithFC, and then followed by the PNSP-based variants with ESC. In addition, *Nrt* for NSPM is the lowest when I is less than 10 but then gets growing drastically, and e-NSP has a stable *Nrt* but consumes slightly more time than GA-NSP on big I. Figure 4f indicates that, under low threshold, two MSIS-based variants maintain an increasing and highest *Nrt* on all I, followed by NSPM. Comparatively speaking, e-NSP keeps a fluctuating but lowest *Nrt* for all algorithms.

Here we analyse the *Tlnc* for mining algorithms on I, which is shown in Figures 4g, 4h and 4i. Figure 4g illustrates that under high threshold, in contrast to the phenomenons on C and T, NegGSP shares a similar *Tlnc* with PNSPwithCover and PNSP and even has a lower *Tlnc* when I is less than 8. That is because PNSP generates NSC by appending a seed sequence with a frequent itemset, and as I grows much more itemsets become frequent and their average size gets larger, as a result of which enormous long-length NSC are generated and thus *Tlnc* for PNSP-based variants without ESC gets much higher. Comparatively speaking, *Tlnc* for GA-NSP shows a clear rising trend as I is increased and even exceeds that of NegGSPwithFC when I is more than 10, which demonstrates that it is less competitive for NSP mining on big I under high threshold. Figure 4i indicates that under medium threshold, *Tlnc* for most algorithms experiences a slow growth with I rising from 4 to 8 and then gets increased substantially, since they adopts EFC as a FreC constraint and increasing I enlarges search space dramatically. Compared with other algorithms, e-NSP maintain a stable and lowest *Tlnc* since SFreC is adopted and it is less sensitive to the growth of I. Figure 4i shows that under low threshold, *Tlnc* for NSPM and MSIS-based variants shows an increasing trend rapidly with I rising and the difference between them gets larger continuously. NPSM consumes three quarters of the *Tlnc* used by MSIS-based variants when $I = 4$ while only occupies less than one third.

A.2.5 Algorithm Efficiency Analysis w.r.t. DB. The number of NSP discovered by algorithms on DB are shown in Figures 5a, 5b and 5c. The high threshold is set as 0.60 to illustrate the evaluation of NegGSP, medium threshold is set as 0.40 to show that of PNSP-based variants and NegGSPwithFC, and low threshold is set as 0.30 to show the results of other algorithms.

Figure 5a shows that, under a high threshold, NegGSP has a much higher but slightly declining *Nct* as DB grows, while NegGSPwithFC, all PNSP-based variants with MPS-based negative containment and MSISwithCover maintain almost constant *Nct*. This phenomenon shows that NSP which dissatisfy EFC and are mined on small DB may have lower support than NSP in a dataset with big DB, and are less representative. Figure 5b shows that, when $min\_sup = 0.40$, the *Nct* for NegGSPwithFC and PNSP-based variants reduces slightly, while that of e-NSP, NSPM and MSIS-based variants remains relatively stable w.r.t. DB. Figure 5c shows that, under low threshold, the algorithms targeting particular types of NSP discover a far smaller number of patterns than GA-NSP on large DB, which reveals that these algorithms are not suitable for mining datasets containing enormous data sequences.

Next we compare the runtime consumed by algorithms in mining NSP on different sizes of DB, as illustrated in Figures 5d, 5e and 5f. Figure 5d reveals that, under high threshold, the *Nrt* for NegGSP is quite sensitive to the size of DB. There is a fast linear growing trend when the size of DB increases. In addition, PNSPwithCoverESC consumes less runtime consumed than PNSPwithCover and NegGSPwithFC on larger DB. Since they discover roughly same number of NSP, PNSPwithCoverESC outperforms PNSPwithCover and NegGSPwithFC in terms of time efficiency under high threshold. Moreover, NSPM and GA-NSP have a much lower *Nrt* than others on large DB, which is four fifths and five eighths respectively of the *Nrt* for e-NSP. Furthermore,



(a) NSP Count (min_sup=0.60) (b) NSP Count (min_sup=0.40) (c) NSP Count (min_sup=0.30)

(d) NSP Runtime (min_sup=0.60) (e) NSP Runtime (min_sup=0.40) (f) NSP Runtime (min_sup=0.30)

(g) Total Length of NSC (min_sup=0.60) (h) Total Length of NSC (min_sup=0.40) (i) Total Length of NSC (min_sup=0.30)

Fig. 5. Algorithm Efficiency Analysis w.r.t. DB (X-axis stands for DB, and Y-axis stands for algorithm efficiency)

compared with MSISwithContain, MSISwithCover consumes similar *Nrt* but discovers much more NSP, and similar phenomenon can be also seen in the comparison of PNSPwithCover and PNSP, which shows that MPS-based negative containment is more suitable for mining on large DB under high threshold. Figure 5e shows that, under medium threshold, the PNSP-based variants with ESC take less than half the runtime of NegGSPwithFC, PNSP and PNSPwithCover, which reveals that ESC can prune a significant number of data sequences in testing the generated NSC. In addition, the *Nrt* for e-NSP and NSPM is much lower and more stable than that of GA-NSP and MSIS-based variants. Figure 5f shows that, when the threshold is 0.30, NSPM retains a relatively stable and low *Nrt*, which is different from the rapid growth on other factors, and is only slightly higher than the *Nrt* for e-NSP. Based on the above analysis, NSPM and e-NSP are more time-efficient on large DB.

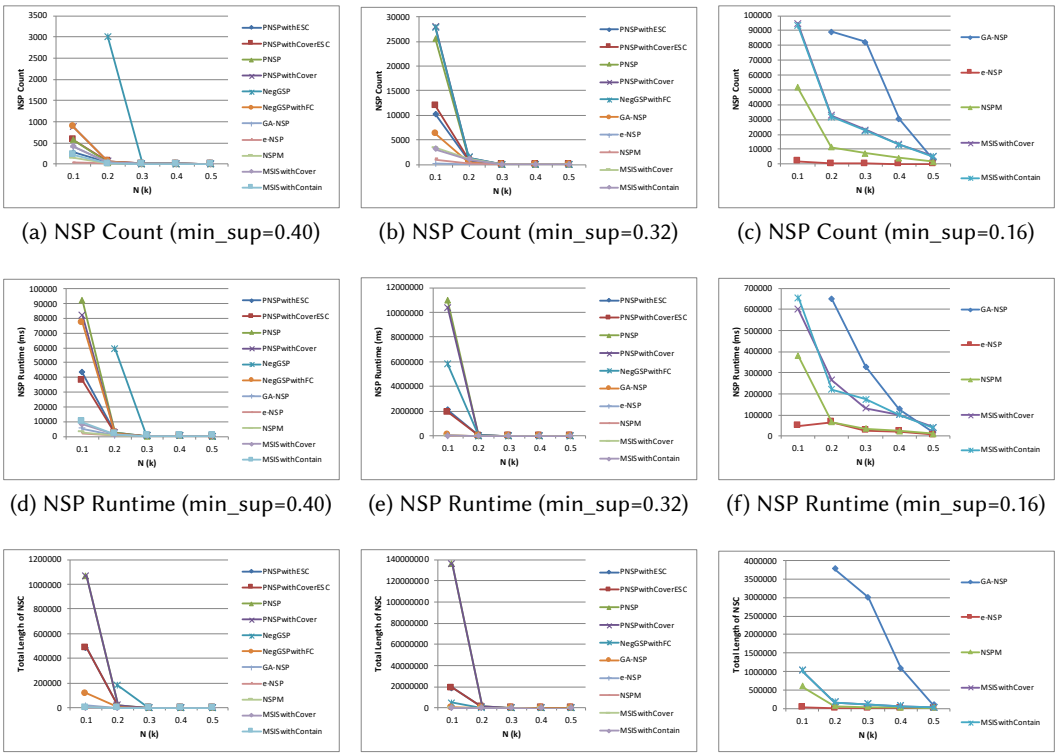
Figures 5g, 5h and 5i show the *Tlnc* w.r.t. DB change under different *min_sup* thresholds. Figure 5g indicates that the memory usage of most algorithms remains almost unaltered when the DB size changes, and the difference between the *Tlnc* for PNSP and PNSPwithESC becomes slightly larger as DB grows. Figure 5h shows that, when the threshold is 0.40, the *Tlnc* for PNSPwithCover and PNSP shows a gently declining trend compared to the relatively stable outcomes for others. It is noted that, PNSPwithESC takes roughly one third of *Tlnc* for PNSP for any size of DB, which shows that the *Tlnc* for PNSP-based variants is insensitive to DB size change under medium threshold. In addition, This indicates that the adoption of ESC can reduce memory usage to a great extent. Of all the

algorithms, e-NSP has a clearly lower *Tlnc*, which is less than one tenth of that of NSPM for any DB and is quite space-efficient. Furthermore, Figure 5i indicates e-NSP, NSPM and MSIS-based variants retain a stable and much lower *Tlnc* under low threshold than GA-NSP, with e-NSP consuming only a tiny proportion of the memory usage of NSPM.

A.2.6 Algorithm Efficiency Analysis w.r.t. N. Here we analyse the efficiency of these NSP mining algorithms on N. The high threshold is set as 0.40 to illustrate the evaluation of NegGSP, medium threshold is set as 0.32 to show that of PNSP-based variants and NegGSPwithFC, and low threshold is set as 0.16 to show the results of other algorithms.

Figures 6a, 6b and 6c illustrate these algorithms' *Nct* under three thresholds on N, and we can see that different from the *Nct* shown in previous factors, their *Nct* on N gets decreased dramatically with N rising. From Figure 6a we note that when threshold is 0.40, NegGSP has a obviously higher *Nct* on small N but maintains a similar *Nct* with others when N gets larger than 0.3k, because as more number of different items occur in dataset the average support of each entity declines and number of long-length NSP gets decreased rapidly. In addition, compared with NegGSPwithFC and PNSPwithCover, PNSPwithCoverESC can discover most of NSP while PNSPwithESC and PNSP only obtain roughly half number of patterns when N is larger than 0.2k, which demonstrates that N-containment can filter large number of NSP under relatively high threshold. Furthermore, we note that *Nct* for GA-NSP maintains relatively high on each N and even equals to that of PNSPwithCover, because as N goes up fewer items gets frequent and its GA-based NSC generation strategy can derive enough valid NSC. Moreover, MSISwithCover keeps a high *Nct* on small N but obtain no pattern when N is more than 0.3k, and it shows that MSIS-based variants are not suitable on big N. Figure 6b indicates that under $min\_sup = 0.32$, all of these algorithms discover far less number of NSP when N is larger than 0.2k, which demonstrates that algorithm's *Nct* is quite sensitive to N. And it can be seen that NegGSPwithFC has a slightly lower *Nct* than PNSPwithCover on $N = 0.1k$ while maintains a same *Nct* on big N. Comparatively speaking, PNSP keeps a much higher proportion of *Nct* than PNSPwithCoverESC and PNSPwithESC on small N and it shows that ESC leads to the lost of many NSP. In contrast to the high *Nct* under high threshold, GA-NSP has a much lower *Nct* when N is less than 0.3k but then keeps a same *Nct* with PNSPwithCover, which shows that it is less competent for NSP mining on small N under low threshold. Compared with e-NSP, NSPM and MSIS-based variants discover much larger number of NSP on small N but finds no pattern when N is more than 0.4k, which shows they cannot work on big N. From Figure 6c we note that under low threshold, as N grows *Nct* for e-NSP, NSPM and MSIS-based variants declines quickly at first and then get decreased gently, and NSPM discovers much less proportion of NSP on big N than MSISwithContain. *Nct* for NSPM is over half of that of MSISwithContain on $N = 0.1k$ while becomes less than one third on $N \geqslant 0.3k$.

Nrt for these algorithms on N is illustrated in Figures 6d, 6e and 6f. Figure 6d indicates that when threshold is 0.40, NegGSP consumes far more time in mining NSP when N is less than 0.2k, however, its *Nrt* is even slightly lower than that of PNSP-based variants, since on big N few frequent items are derived to generate long-length NSC. In addition, we note that NSPM has the fastest declining *Nrt* and only consumes half of the runtime taken by e-NSP, because much less NSC can be generated by NSPM benefit of its strict LFC on big N. Figure 6e shows that when threshold gets to 0.32, PNSPwithCoverESC and PNSPwithESC consume almost same runtime on $N = 0.1k$, which is roughly one third of that of NegGSPwithFC and less than one fifth of that of PNSPwithCover and PNSP. However, as N grows to 0.2k, PNSPwithESC consumes a much higher *Nrt* than PNSPwithCoverESC, which is almost same as that of NegGSPwithFC and over six sevenths of that of PNSP. That demonstrates that N-containment increases NSP mining runtime to a high extent on a higher N while ESC can reduce *Nrt* much more obviously on a small N. In addition,



(a) NSP Count (min_sup=0.40) (b) NSP Count (min_sup=0.32) (c) NSP Count (min_sup=0.16)

(d) NSP Runtime (min_sup=0.40) (e) NSP Runtime (min_sup=0.32) (f) NSP Runtime (min_sup=0.16)

(g) Total Length of NSC (min_sup=0.40) (h) Total Length of NSC (min_sup=0.32) (i) Total Length of NSC (min_sup=0.16)

Fig. 6. Algorithm Efficiency Analysis w.r.t. N (X-axis stands for N, and Y-axis stands for algorithm efficiency)

e-NSP has a much lower *Nrt* on $N = 0.1k$ while consumes similar runtime with GA-NSP and NSPM on big N, because many invalid NSC are derived from obtained PSP by set-theory-based mining algorithms on big N. Figure 6f illustrates that under low threshold, *Nrt* for GA-NSP is much higher than that of e-NSP, NSPM and MSIS-based variants when $N \leqslant 0.4k$ but is only less than half of that of MSIS-based variants on $N = 0.5k$. In addition, e-NSP keeps a relatively stable and low *Nrt*, which is much lower than others on $N = 0.1k$ but almost same as that of NSPM on big N. As can be seen from these figures, e-NSP and NSPM has a clear superiority on NSP runtime only on big N under a quite low threshold.

Figures 6g, 6h and 6i illustrate the *Tlnc* for these algorithms on N correspondingly. From Figure 6g we note that when threshold is 0.40, *Tlnc* for NegGSP is much higher than others on $N = 0.2k$ but then declines rapidly. *Tlnc* for NegGSP is over eight times as that of PNSPwithCover when N is 0.2k while are roughly half of those of PNSP-based variants when N is larger than 0.3k. That is because on big N less long-length seeds can be derived and thus less long-length NSC can be generated by NegGSP, which shows joining-based NegGSP is quite memory-efficient and has a smaller research space on big N even though it has a large *Tlnc* on small N. In addition, it can be seen that compared with PNSP-based and NegGSP-based variants, GA-NSP has relatively lower *Tlnc* on small N, but keeps much higher *Tlnc* when $N \geqslant 0.3k$, which shows that GA-based NSC generation strategy contributes less to the reduction of number of NSC and save memory

space on big N. Furthermore, we notice that MSIS-based variants consume over twice memory space compared with NSPM on small N but get no NSC on big N, and it demonstrates that the derivation-based NSC generation strategy adopted by MSIS becomes invalid on big N. It can be noted from Figure 6h that when threshold declines to 0.32, PNSPwithCover and PNSP maintain a far higher *Tlnc* than PNSPwithCoverESC and PNSPwithESC on $N = 0.1k$. Compared with PNSPwithCoverESC and PNSPwithESC on $N = 0.1k$, PNSPwithCover and PNSP has a roughly twice *Tlnc* when $min\_sup = 0.40$ while occupies an over seven times *Tlnc* when $min\_sup = 0.32$. However, on big N, their *Tlnc* is only slightly higher than those of PNSP-based variants with ESC under both thresholds. This reason is that as N grows, far less long-length NSC are generated and thus ESC helps less on filtering NSCs. Figure 6i indicates that under low threshold, GA-NSP maintains much higher *Tlnc* than e-NSP, NSPM and MSIS-based variants. When N is 0.3k, compared with MSISwithCover, GA-NSP has about twice *Tlnc* under $min\_sup = 0.32$ while holds an over 28 times *Tlnc* under $min\_sup = 0.16$, which shows that GA-NSP generates enormous number of NSC and is much more memory-consuming under low threshold on relatively big N.

REFERENCES

[1] Rakesh Agrawal and Ramakrishnan Srikant. 1995. Mining sequential patterns. In *Proceedings of the Eleventh International Conference on Data Engineering*. IEEE, 3–14.

Received February 2007; revised March 2009; accepted June 2009