SEQUENTIAL PATTERN MINING WITH THE MICRON'S AUTOMATA PROCESSOR

Ke Wang, Elaheh Sadredini, Kevin Skadron
Center for Automata Computing
Department of Computer Science
University of Virginia

THE CENTER FOR AUTOMATA PROCESSING

Outline

- Micron's Automata Processor (AP)
- SPM on the AP: Opportunities and Challenges
- AP Accelerated SPM and optimizations
- Performance Evaluation
- Conclusions and the Future Work

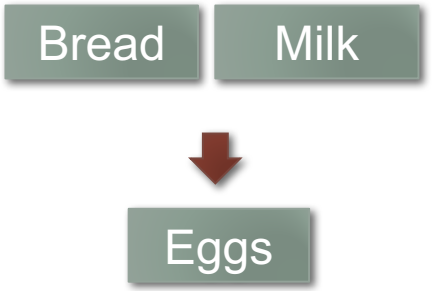
Sequential Pattern Mining

Sequential Pattern Mining(SPM):

- discovers frequent sequence of transactions
- the order of items within a transaction doesn't matter, but the order of transaction is important

- Customer purchase patterning analysis
- Correlation analysis of storage system
- Web log analysis
- Software bug tracking
- Software API usage tracking

Trans.	Items
1	<{Bread, Milk}, {Coke}>
2	<{Bread, Milk, Diaper}{Beer, Eggs}{Diaper}>
3	<{Milk} {Diaper} {Beer, Coke}>
4	<{Bread, Milk, Diaper}{Beer, Diaper}{Beer, Coke, Eggs}>
5	<{Bread, Milk}{Coke}{Diaper}{Eggs}>



Seq. Pat. Mining vs. Freq. Set. Mining

❑ Similarity

- Can be solved by Apriori-based algorithm
- Massive pattern matching is the bottleneck

❑ Differences

❖ Hierarchical patterns in SPM

- The sequence of itemsets (transactions) are considered
  - The items within an itemset could be unordered and discontinuous
  - The itemsets within an a sequence could be discontinuous but the order should be checked

❖ Larger search space

- For a given number of total items in a sequential pattern - k, the total number of structures is 2^(k-1)
- Search space needs to be reduced by Apriori-based algorithm (GSP)
- Automaton design space needs to be reduced to save reconfiguration time

AP Accelerated SPM - Flowchart

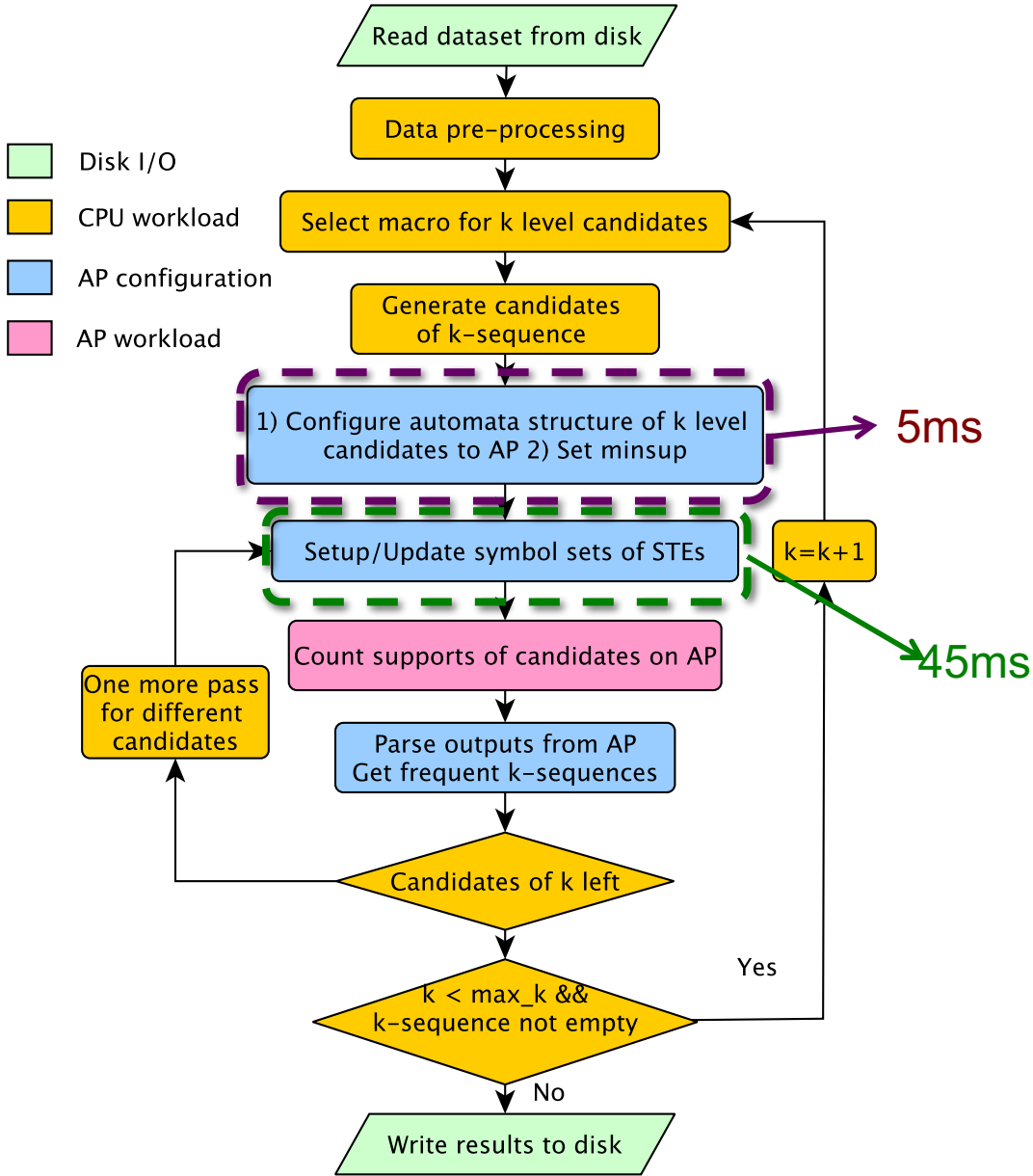
Data preprocessing:

1) Filter out infrequent items
2) Recode -> 8-bit / 16-bit symbols
3) Recode transactions
4) Sort items in transactions
5) Connect transactions by an itemset delimiter（\x253）
6) Connect sequences by a sequence delimiter（\x254）

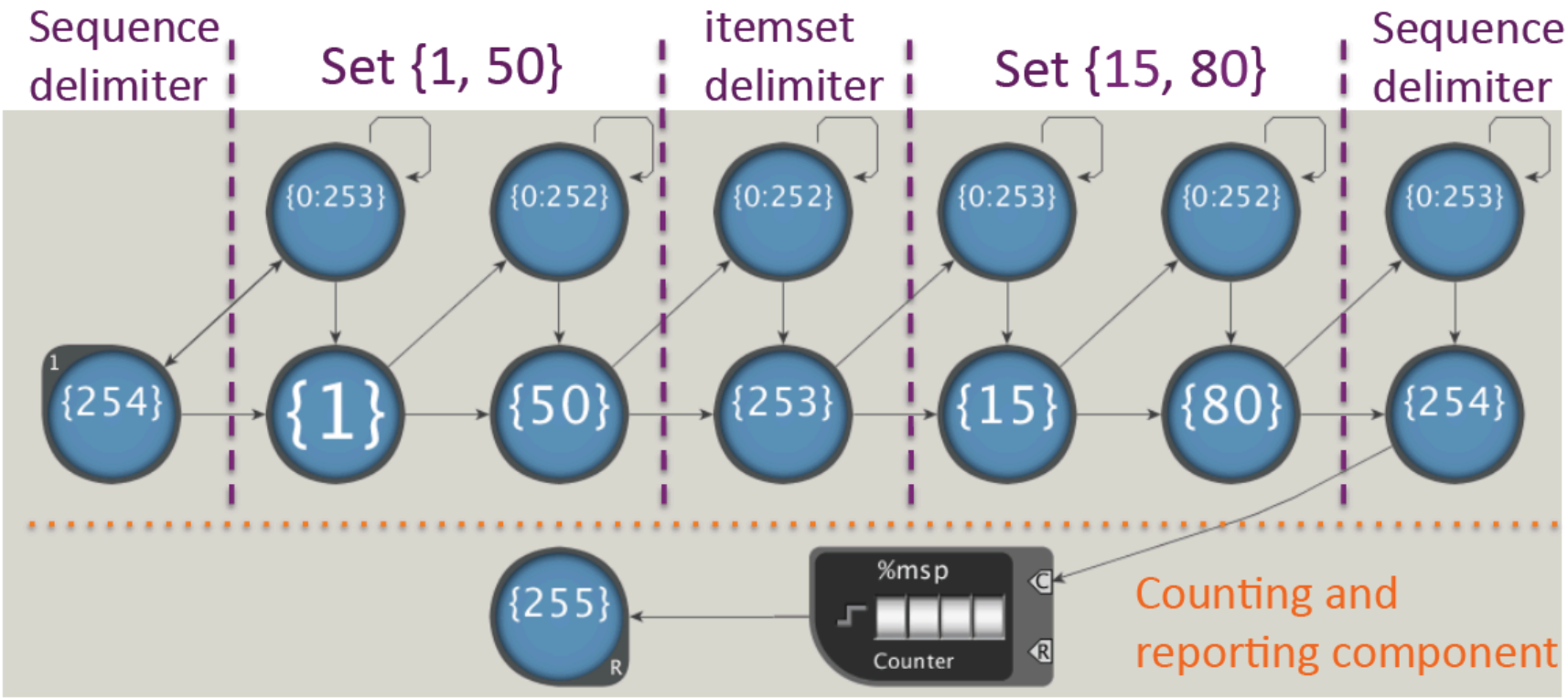
Encoding:

freq_item# <254: 8-bit

253< freq_item# <64009:16-bit

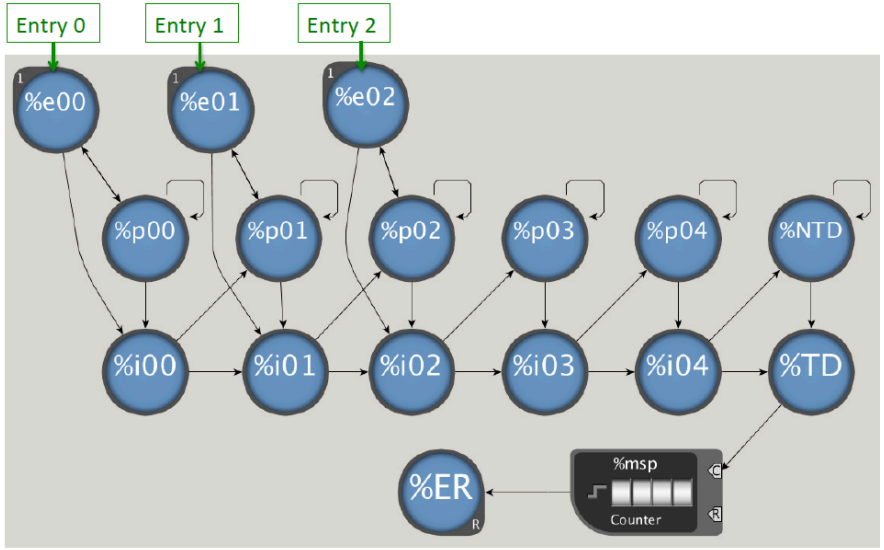


Automata Design for SPM: Flattened

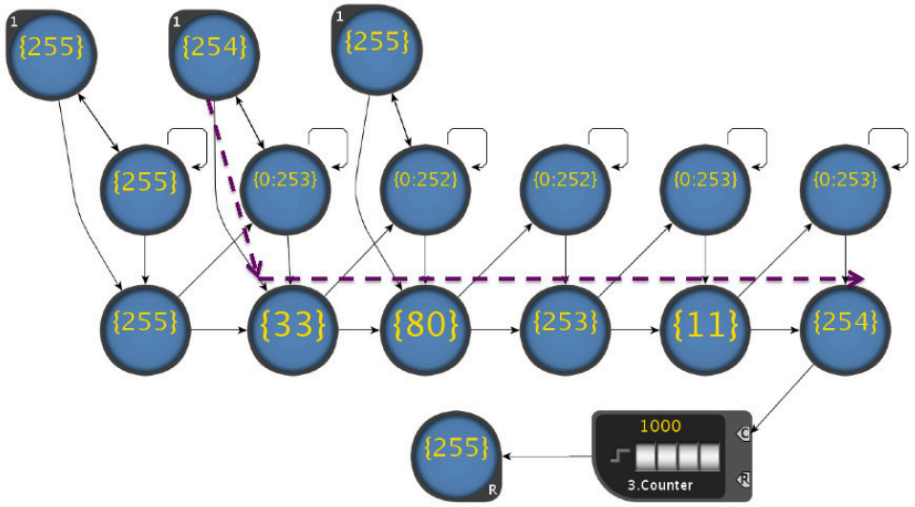


(a) Automaton for matching sequence $< \{1, 50\}, \{15, 80\} >$

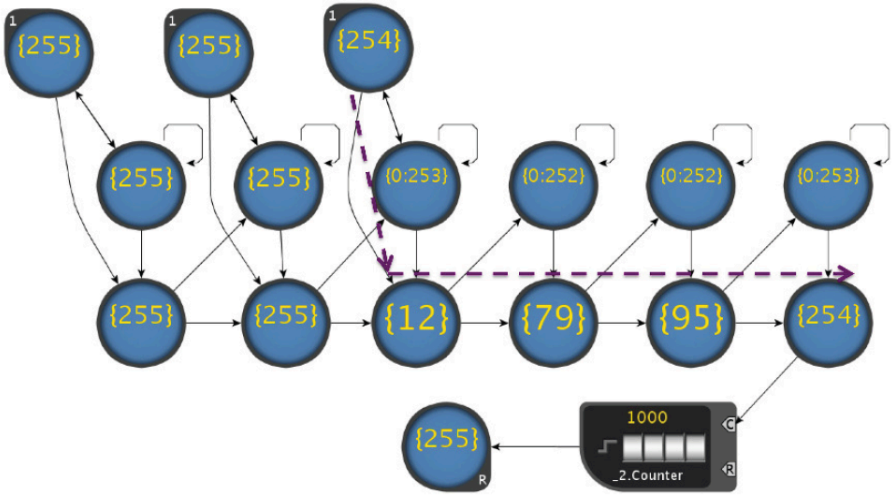
Automata Design for SPM: Multi-entry



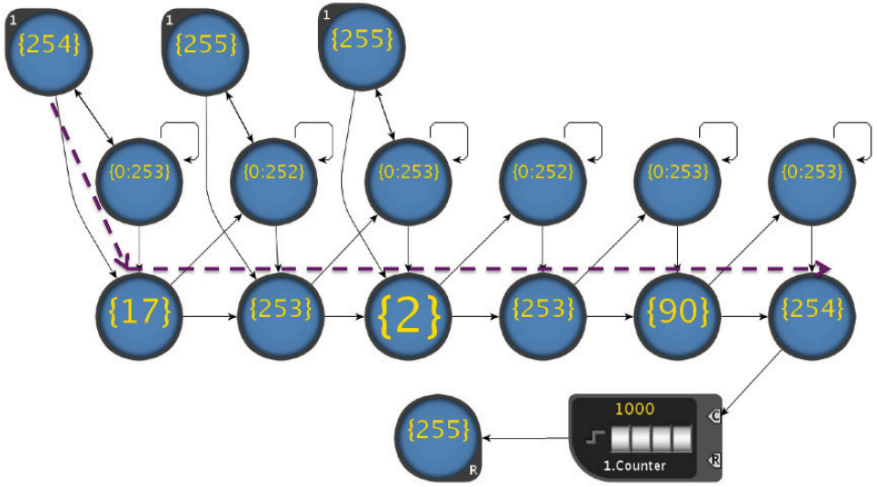
(a) AP macro for sequential pattern



(c) Automaton for sequence $< \{33, 80\}\{11\} >$



(b) Automaton for sequence $< \{12, 79, 95\} >$



(d) Automaton for sequence $< \{17\}\{2\}\{90\} >$

Performance Evaluation – AP

- Target hardware: D480 X 32
- 8-bit encoding:
  - Speed: 7.5ns/ item

	$k <= 10$	$10 < k <= 20$	$20 < k <= 40$
$sup < 4096$	4	2	1
$sup >= 4096$	2	2	1

One D480 AP chip has 192 AP blocks. One 32-chip AP board has 6144 blocks.

- 16-bit encoding:
  - Speed: 15ns/item

	$k <= 5$	$5 < k <= 10$	$10 < k <= 20$
$sup < 4096$	4	2	1
$sup >= 4096$	2	2	1

- Connection reconfiguration time: 5ms for an entire board

  Symbol replacement time: 45ms for an entire board

Performance Evaluation - Comparison

❑ Compare with other implementations

1. Java multi-threading implementation of GSP: GSP-Java
2. C sequential implementation of GSP: GSP-1C
3. OpenMP multi-threading implementation of GSP: GSP-6C
4. GPU implementation of GSP: GSP-1G
5. Java multi-threading implementation of PrefixSpan
6. Java multi-threading implementation of SPADE

❑ Testing platform

- CPU: Intel CPU i7-5820K (6 physical cores, 3.30GHz)
- Mem: 32GB, 1.333GHz
- GPU: Nvidia Kepler K40, 706 MHz clock, 2888 CUDA cores, 12 GB global memory

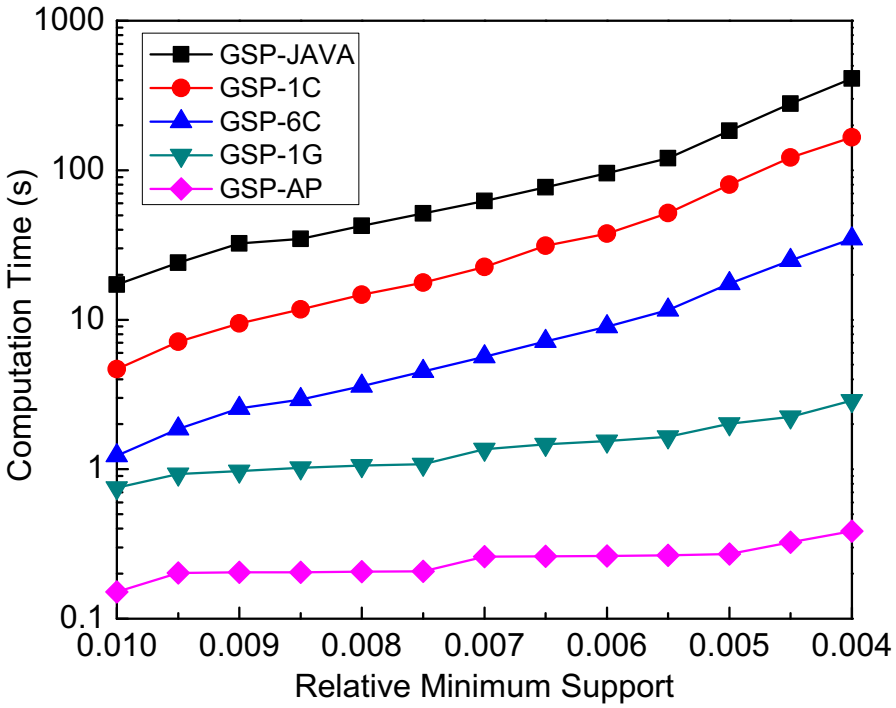
Performance Evaluation - Datasets

- Six real-world datasets

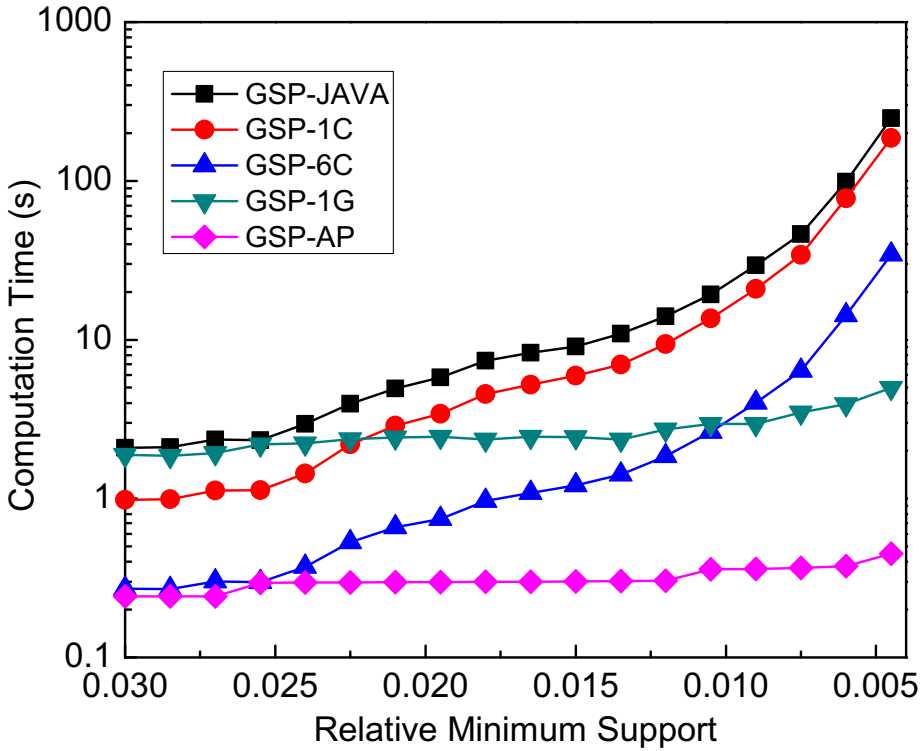
Name	Sequences#	Aver. Len.	Item#	Size (MB)
BMS1	59601	2.42	497	1.5
BMS2	77512	4.62	3340	3.5
Kosarak	69998	16.95	41270	4.0
Bible	36369	17.84	13905	5.4
Leviathan	5834	33.8	9025	1.3
FIFA	20450	34.74	2990	4.8

Aver. Len. = Average number of items per sequence.

Performance Evaluation – GSP implementations

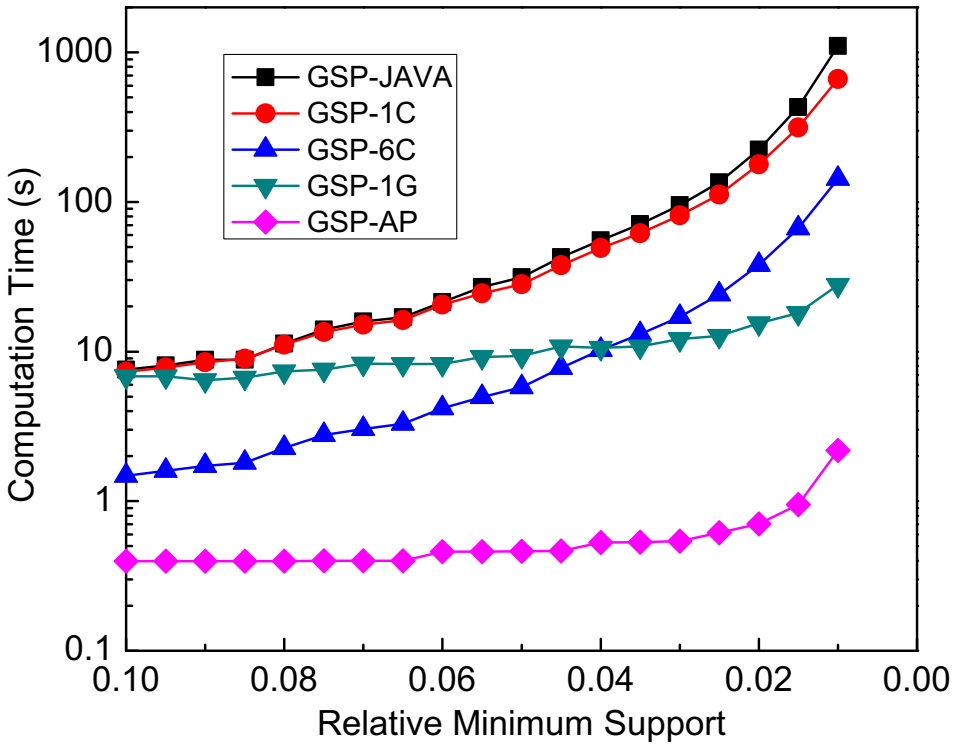


BM2

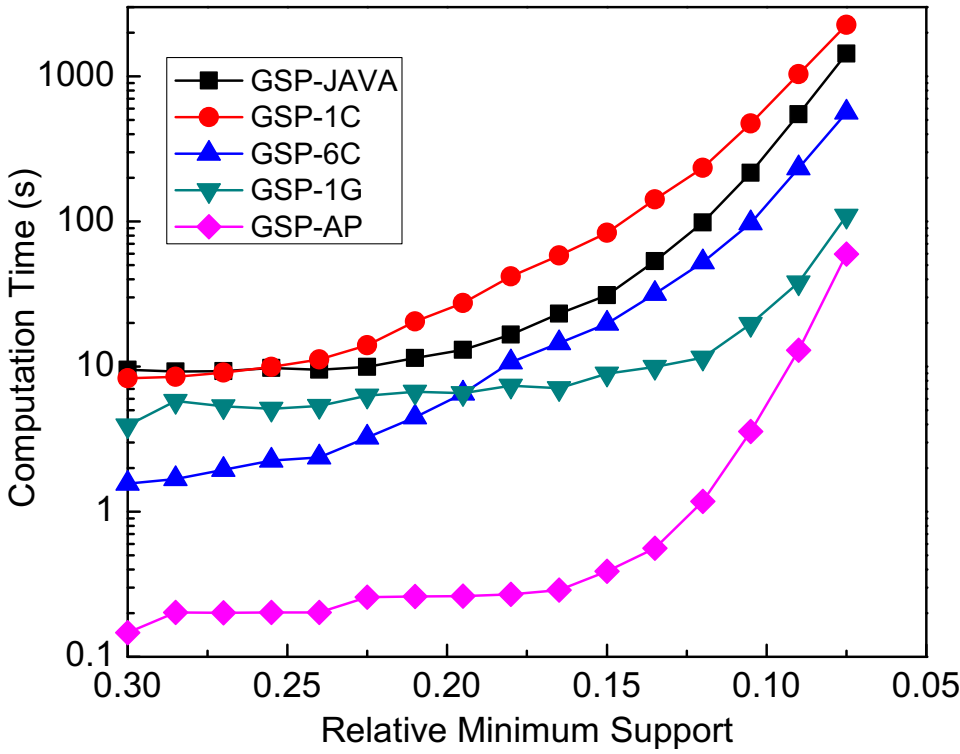


Kosarak

Performance Evaluation – GSP implementations (2)

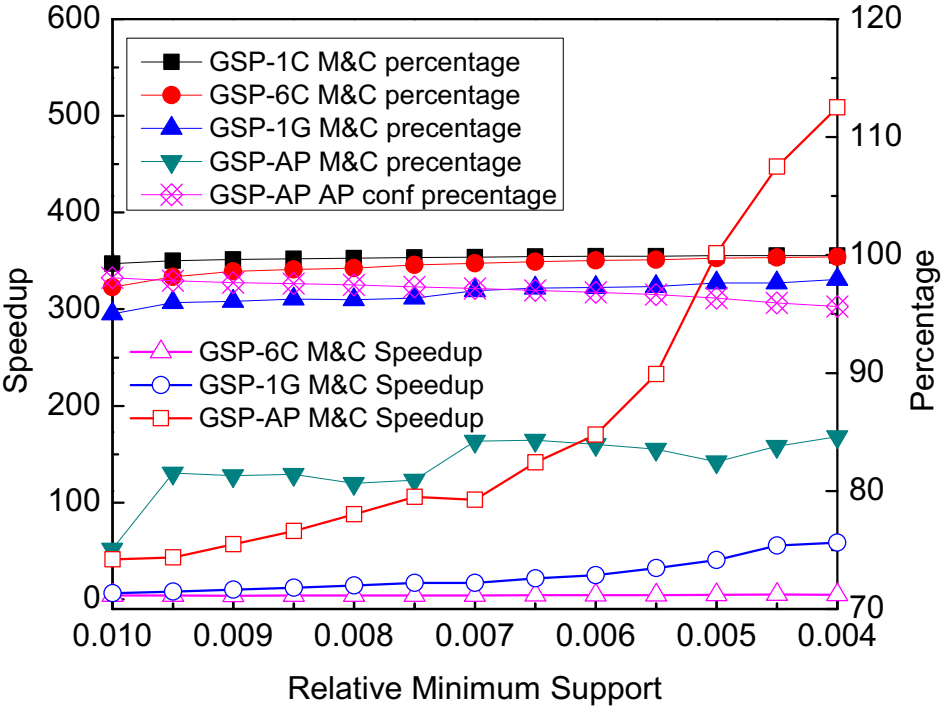


Bible

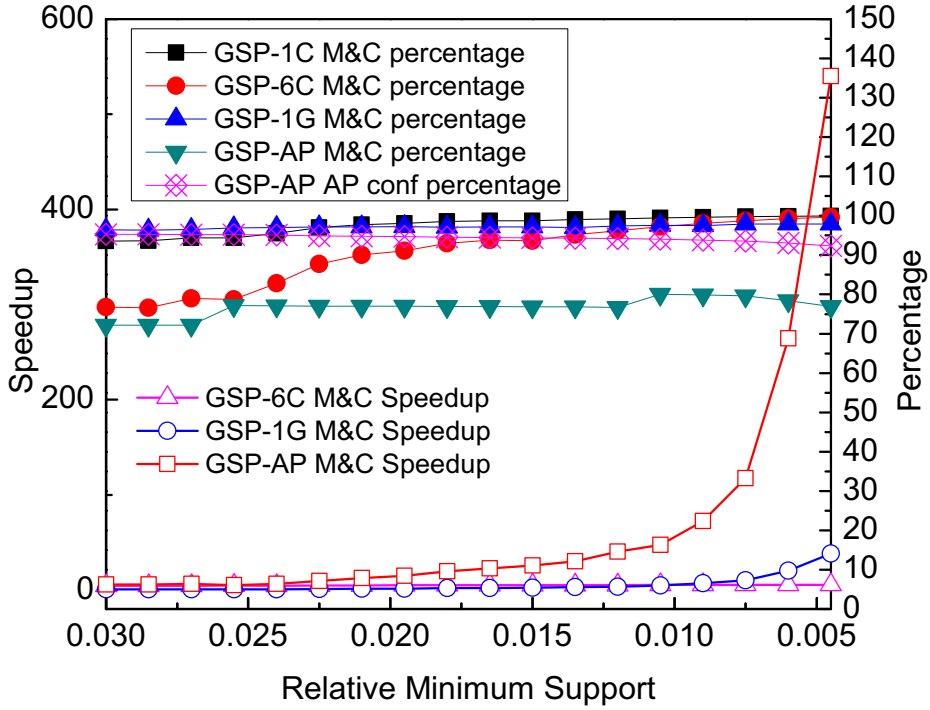


FIFA

Performance Evaluation – Analysis

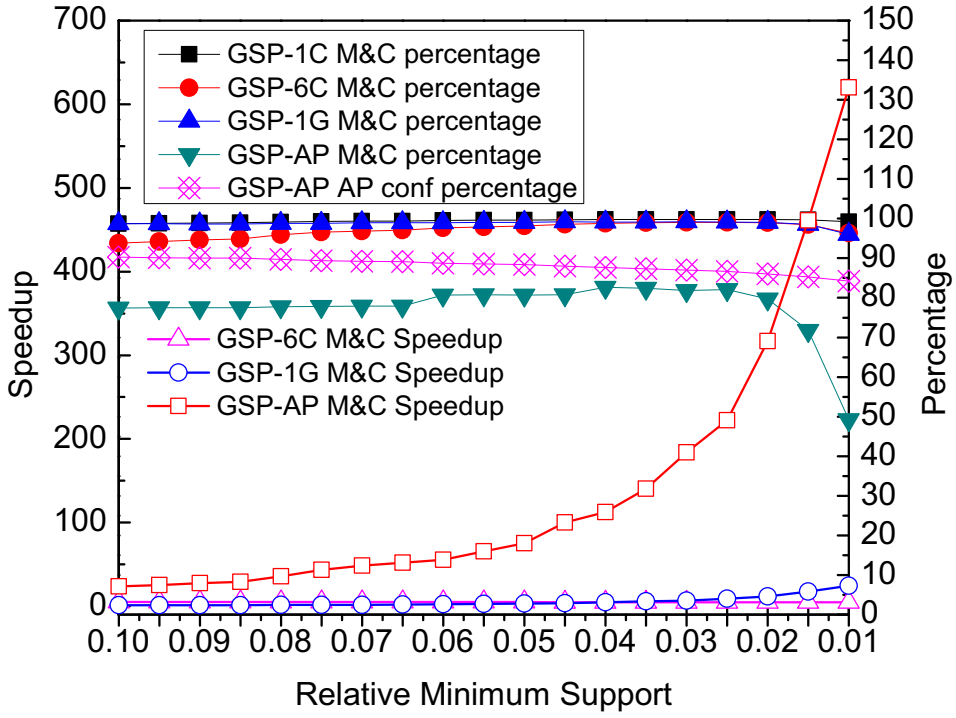


BM2

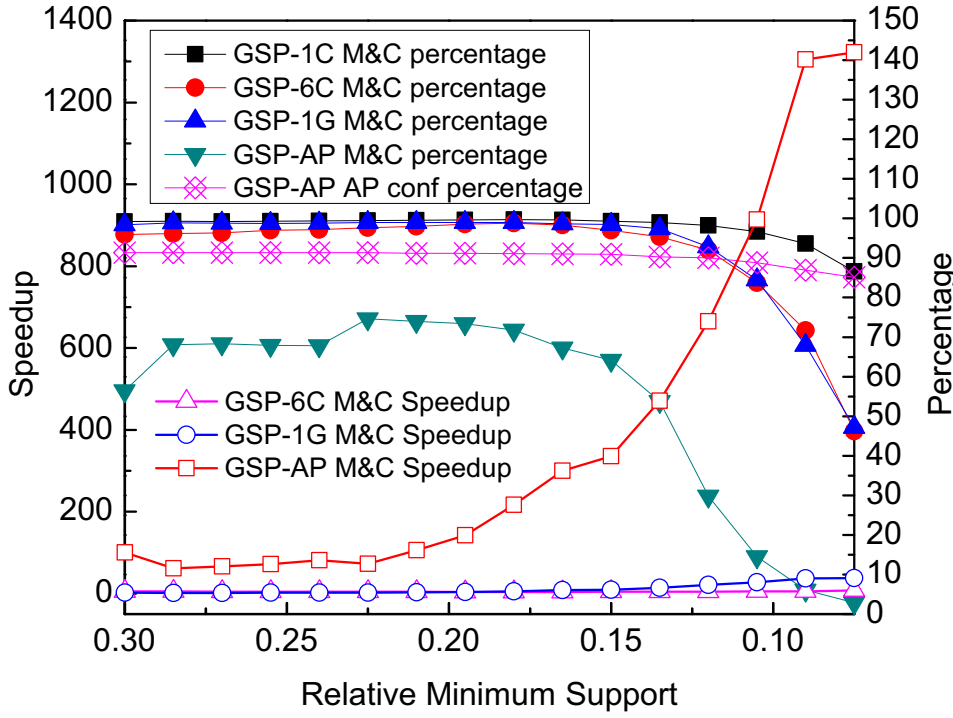


Kosarak

Performance Evaluation – Analysis (2)



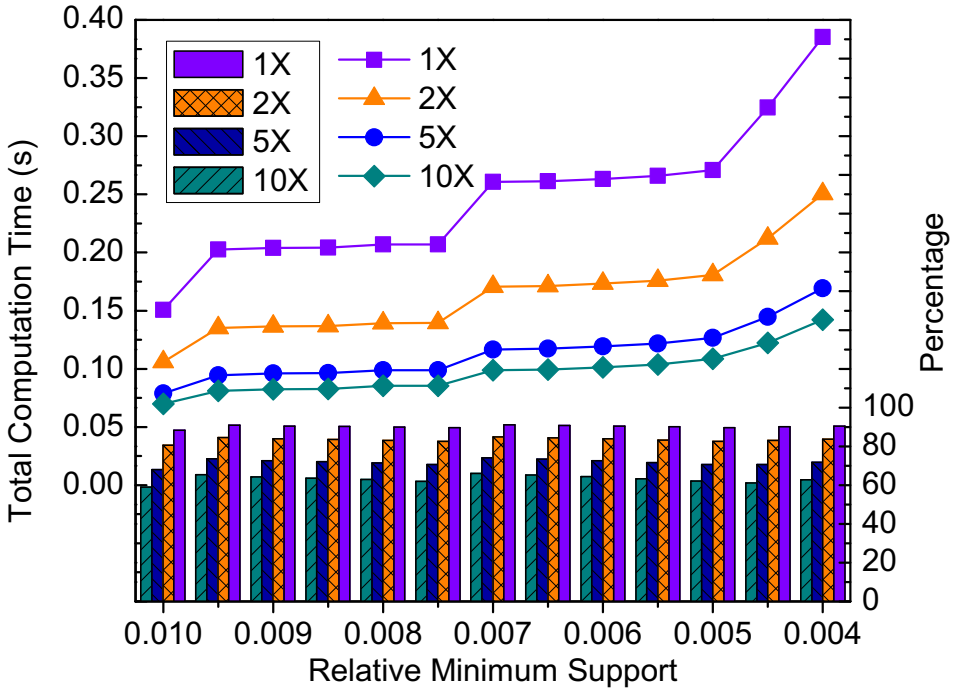
Bible



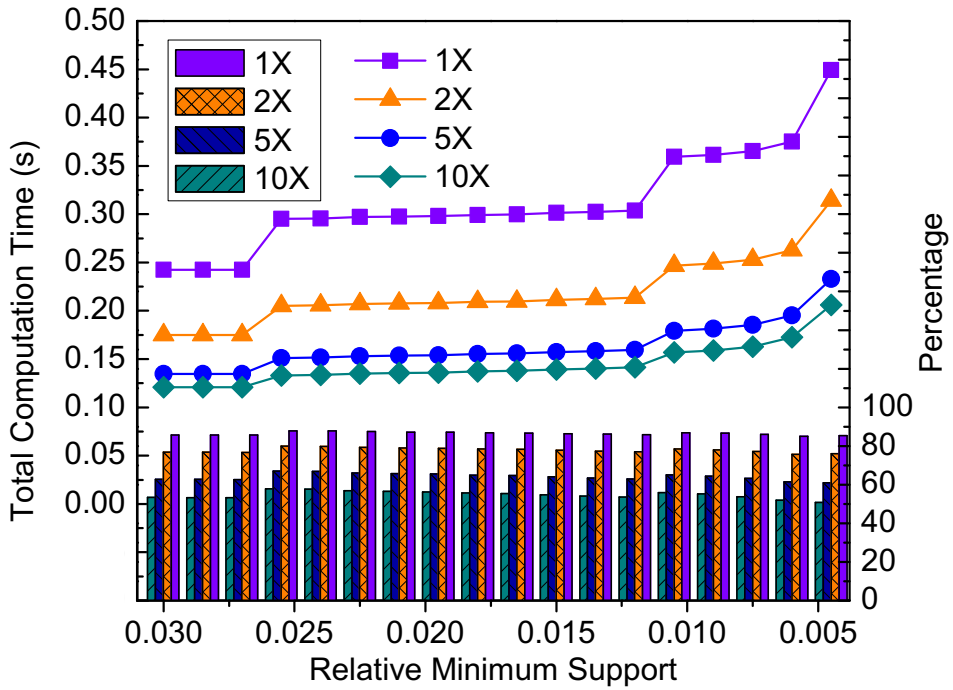
FIFA

- As sup decreasing, un-parallelized candidate generation becomes a new bottleneck
- The STE symbol replacement dominates the AP processing time

Performance Evaluation – Effect of Sym. Rep. time

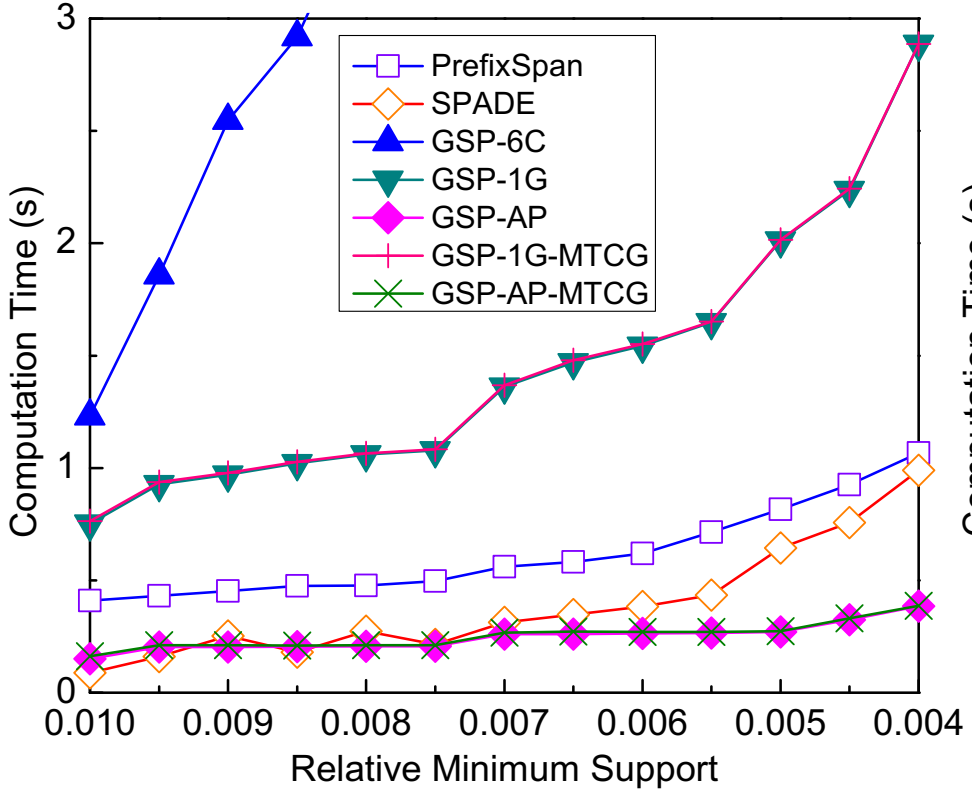


BM2

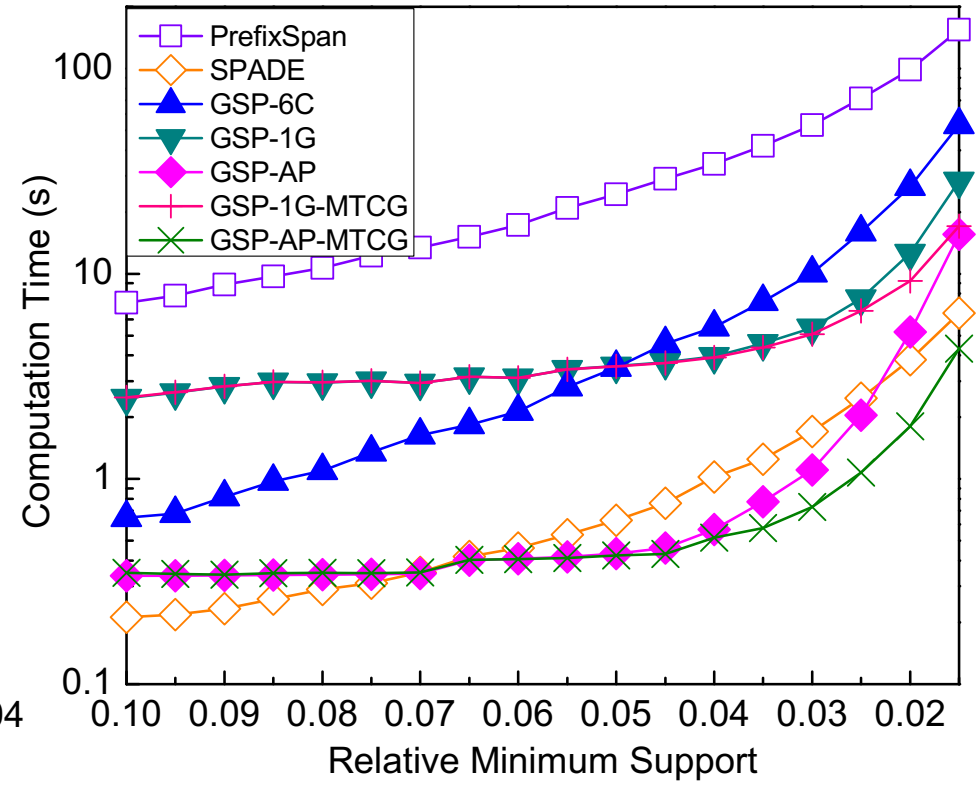


Kosarak

Performance Evaluation – Compare with PrefixSpan and SPADE

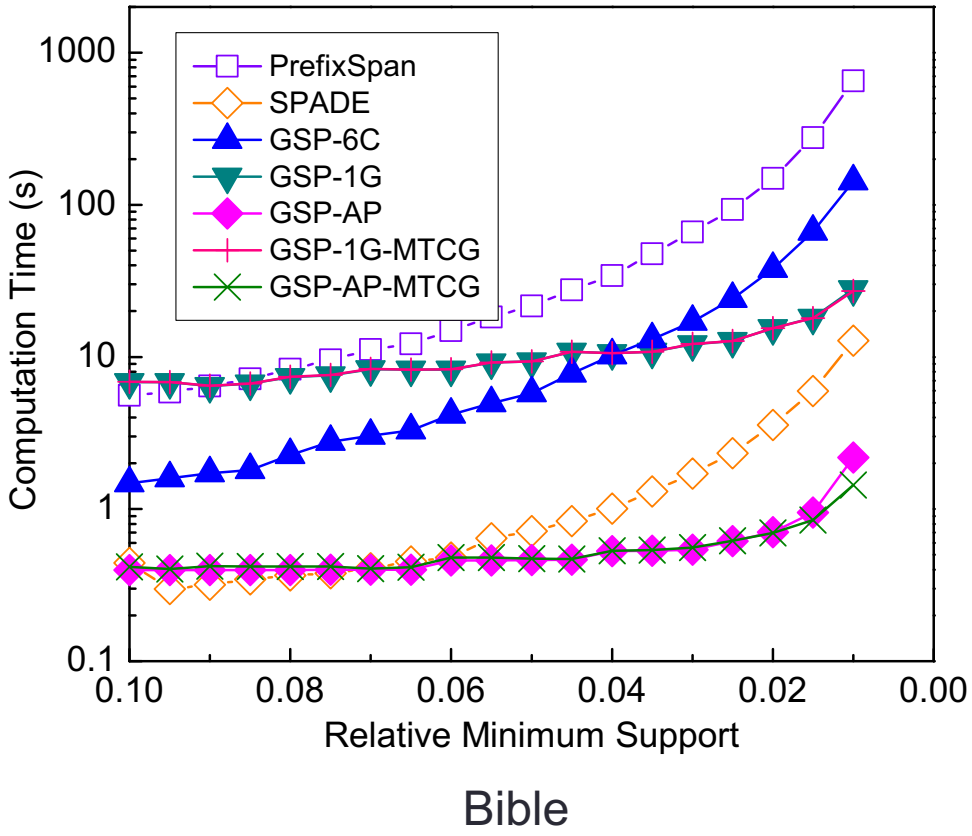


BM2

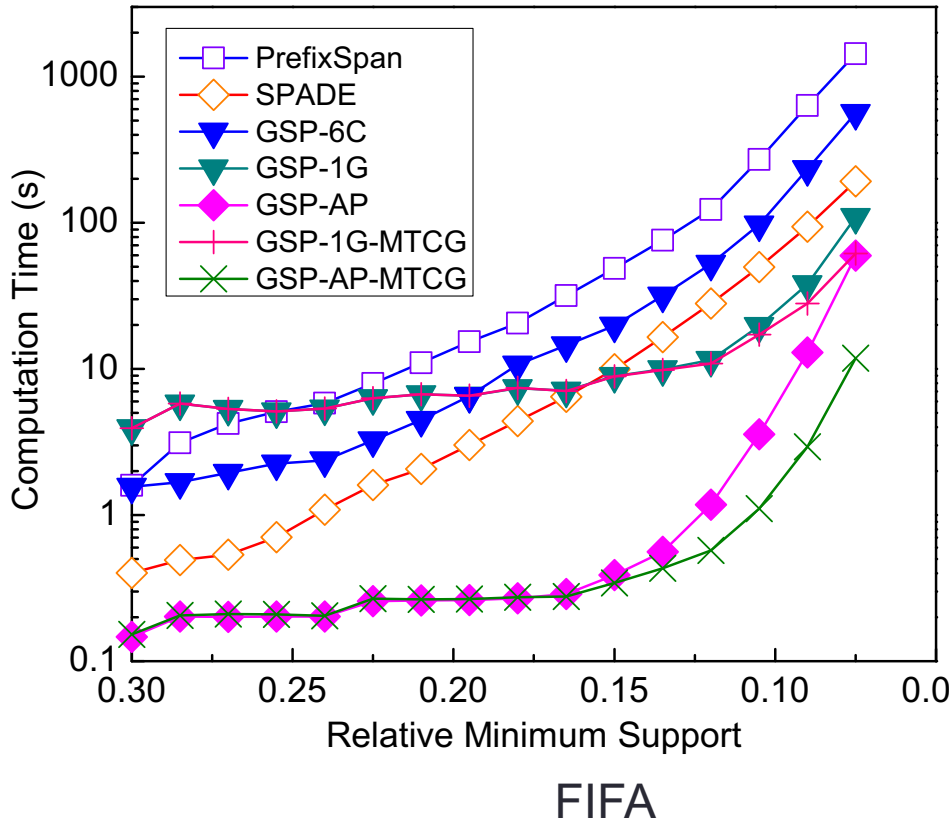


Leviathan

Performance Evaluation – Compare with PrefixSpan and SPADE (2)

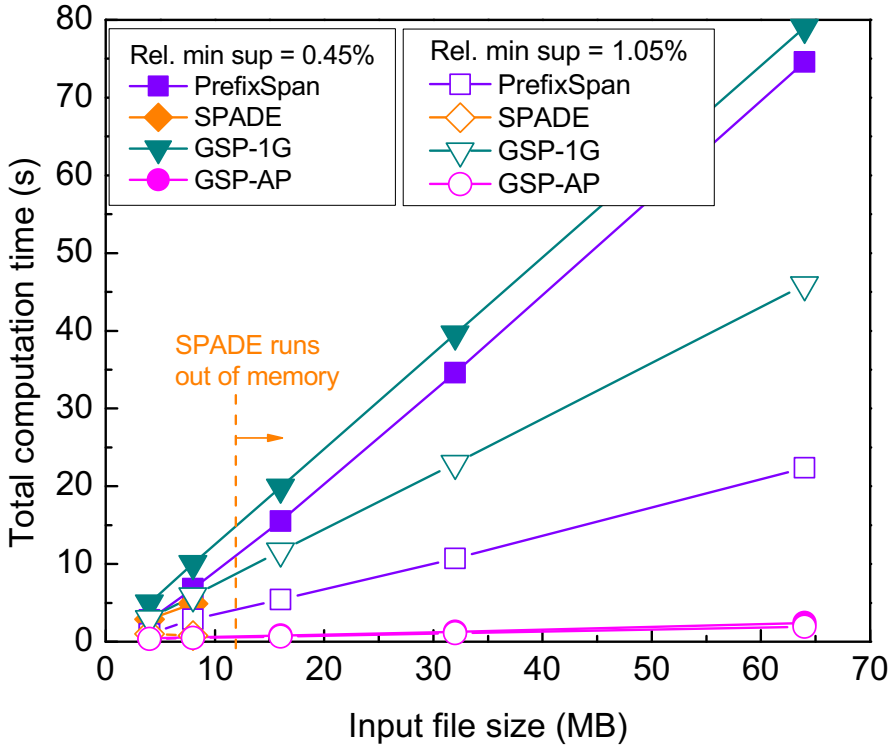


Bible

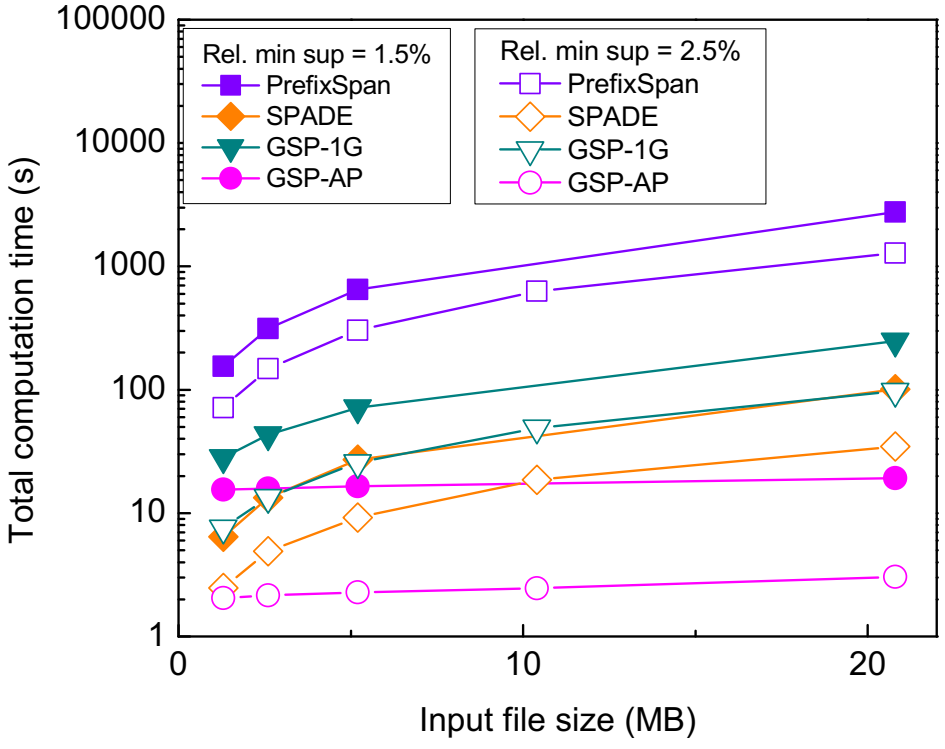


FIFA

Performance Evaluation – input size



Kosarak



Leviathan

Conclusions and Future Work

- We present a hardware-accelerated solution for sequential pattern mining (SPM), using Micron's AP
- Our proposed solution adopts the algorithm framework of the Generalized Sequential Pattern (GSP)
- We derive a compact automaton design for matching and counting frequent sequences.

1) flatten sequences into strings by using delimiters and place-holders

2) multiple-entry NFA strategy is proposed to accommodate variable-structured sequences

Together, this allows a single, compact template to match any candidate sequence of a given length, so this template can be replicated to make full use of the capacity and massive parallelism of the AP

- Up to 430X, 90X, and 29X speedups are achieved by the AP-accelerated GSP on six real-world datasets, when compared with the single-threaded CPU, multicore CPU, and GPU GSP implementations, respectively
- By parallelizing candidate generation, up to 452X and 49X over PrefixSpan and SPADE
- More speedup on larger datasets

Backups

GSP Algorithm

- Srikant, Ramakrishnan, and Rakesh Agrawal. *Mining sequential patterns: Generalizations and performance improvements*. Springer Berlin Heidelberg, 1996.