Efficient Computation of Parsimonious Temporal Aggregation

Giovanni Mahlknecht, Anton Dignös, Johann Gamper

Free University of Bozen-Bolzano, Italy

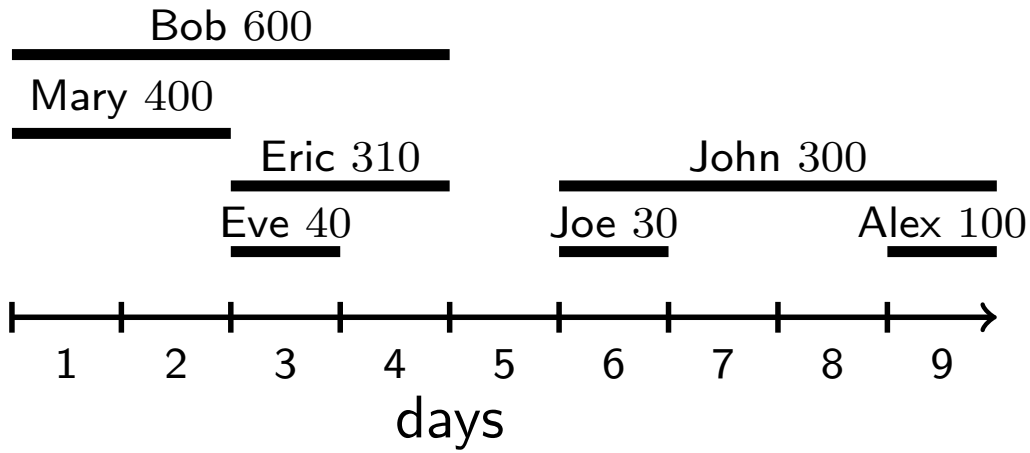
ADBIS 2015
September 8-11, 2015 - Futuroscope, Poitiers, France

Outline

Instant Temporal Aggregation (ITA)

Patient treatment periods with daily cost

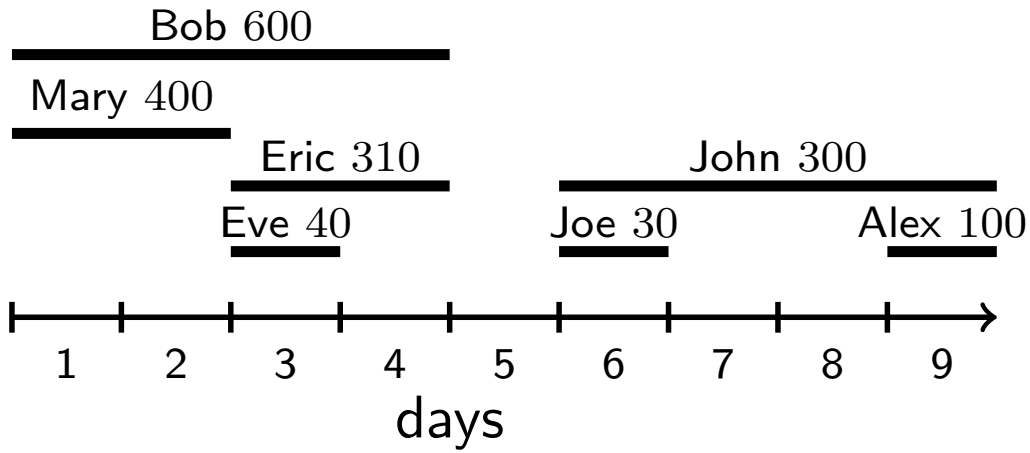
	P	C	T
r_1	Bob	600	[1,4]
r_2	Mary	400	[1,2]
r_3	Eve	40	[3,3]
r_4	Eric	310	[3,4]
r_5	Joe	30	[6,6]
r_6	John	300	[6,9]
r_7	Alex	100	[9,9]



Instant Temporal Aggregation (ITA)

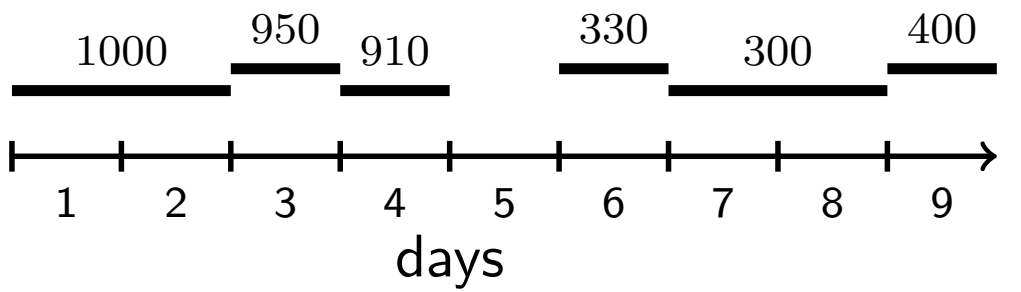
Patient treatment periods with daily cost

	P	C	T
r_1	Bob	600	[1,4]
r_2	Mary	400	[1,2]
r_3	Eve	40	[3,3]
r_4	Eric	310	[3,4]
r_5	Joe	30	[6,6]
r_6	John	300	[6,9]
r_7	Alex	100	[9,9]



ITA: at each timepoint SUM(C)

	Val	T
s_1	1000	[1,2]
s_2	950	[3,3]
s_3	910	[4,4]
s_4	330	[6,6]
s_5	300	[7,8]
s_6	400	[9,9]

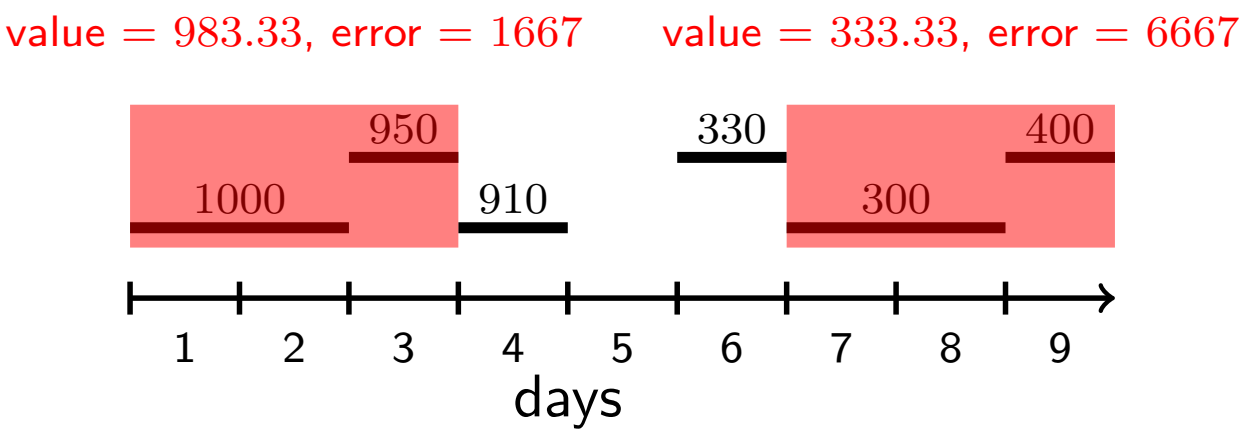


Parsimonious Temporal Aggregation (PTA)

Input: ITA result
Output: merged tuples to size c with minimum error

Rules:

- Merge only adjacent tuples
- Merged values are weighted mean
- Error is Squared Sum Error
- Result is of size c

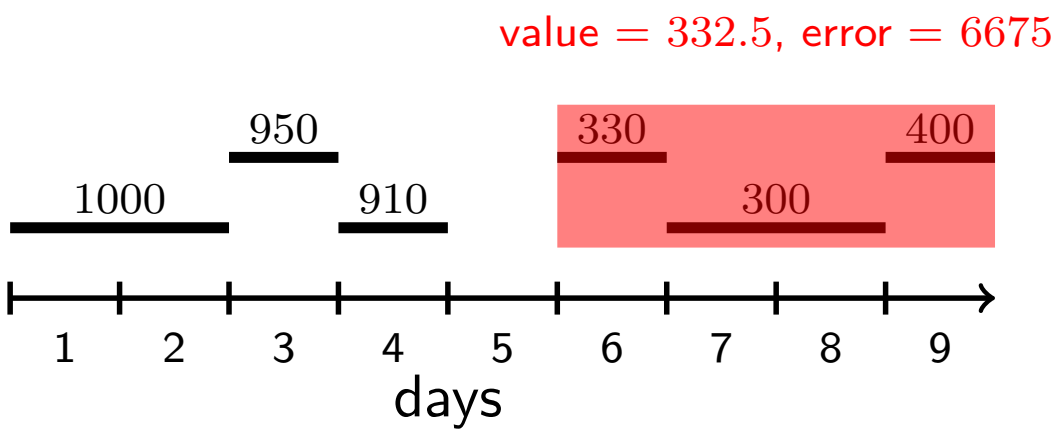


Parsimonious Temporal Aggregation (PTA)

Input: ITA result
Output: merged tuples to size c with minimum error

Rules:

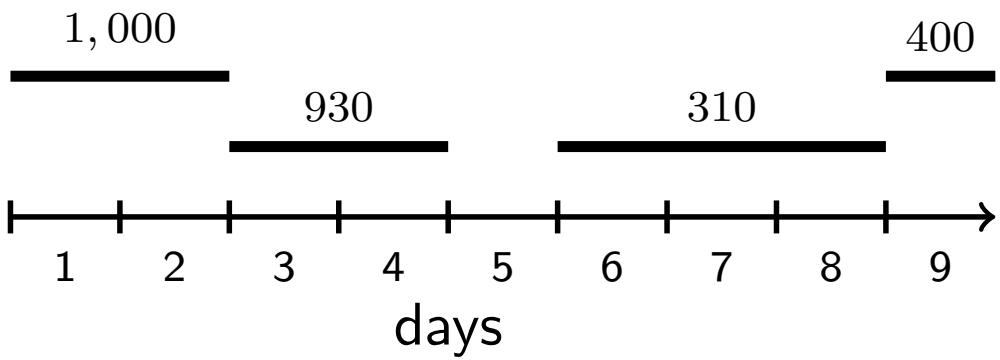
- Merge only adjacent tuples
- Merged values are weighted mean
- Error is Squared Sum Error
- Result is of size c



PTA Optimal Solution

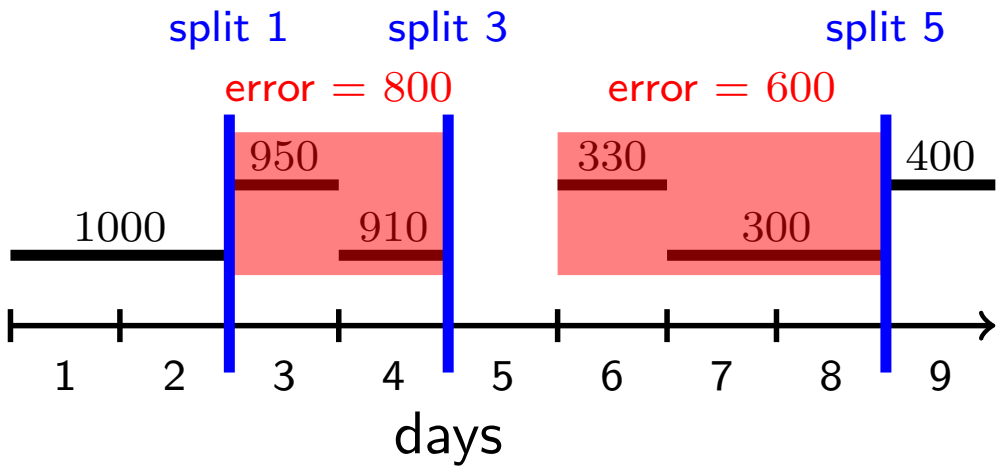
Result of ITA for SUM(C) (size $n = 6$)

Result PTA ($c = 4$) total error 1,400 (optimal solution)

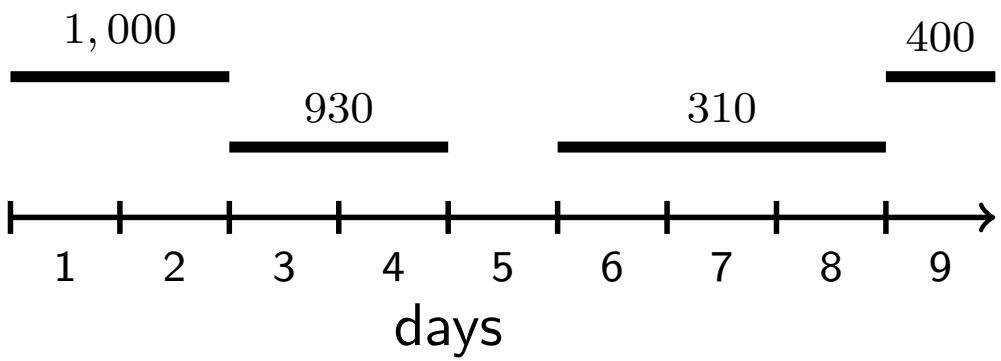


Split Points / Split Path

- PTA computes a split path (sequence of split points)
- Tuples between split points are merged



Split Path: [1, 3, 5]



PTA Existing Algorithm

Dynamic Programming Algorithm

Error Matrix $\mathbf{E}$

	i=1	2	3	4	5	6
k=1	0	1667	5700	∞	∞	∞
2	-	0	800	5700	6300	12375
3	-	-	0	800	1400	6300
4	-	-	-	0	600	1400

$\mathbf{E}_{i,k}$ minimum error in reducing the first i tuples to size k

only the last two rows are used

Split Point Matrix $\mathbf{J}$

	i=1	2	3	4	5	6
k=1	0	0	0	0	0	0
2	0	1	1	3	3	3
3	0	0	2	3	3	5
4	0	0	0	3	3	5

$\mathbf{J}_{i,k}$ optimum split point when reducing the first i tuples to size k

whole matrix

Problem and Contribution

Problem

- Runtime and space requirements of existing algorithm not scalable

Contribution

- **Diagonal Pruning**: Reduces the computational complexity by avoiding unnecessary computations
- **Split Point Graph**: Reduces the space complexity
- Result remains **optimal**

Diagonal Pruning

Lemma (Diagonal Pruning)

For the computation of the error matrix $\mathbf{E}$ *and split point matrix* $\mathbf{J}$ *there exists an upper bound for variable* i*.*

	i=1	2	3	4	5	6
k=1	0	0	0	0	0	0
2	-	1	1	3	3	3
3	-	-	2	3	3	5
4	-	-	-	3	3	5

- Red cells can be avoided, reduces runtime
- allows to eliminate parts of the matrices, reduces memory

Introduction

Diagonal Pruning

Split Point Graph

Experimental Evaluation

Split Point Graph

Challenge

Substitution of Split Point Matrix $\mathbf{J}$ by alternative structure to reduce memory consumption

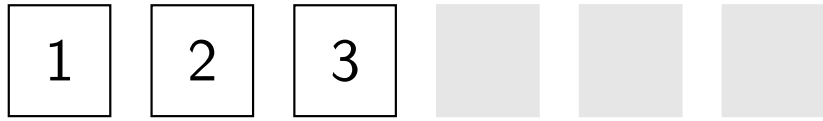
Idea

unnecessary nodes are not stored

Split Point Graph

- Only necessary nodes are inserted
- Nodes are removed when they become obsolete (Path Pruning)

Graph Evolution

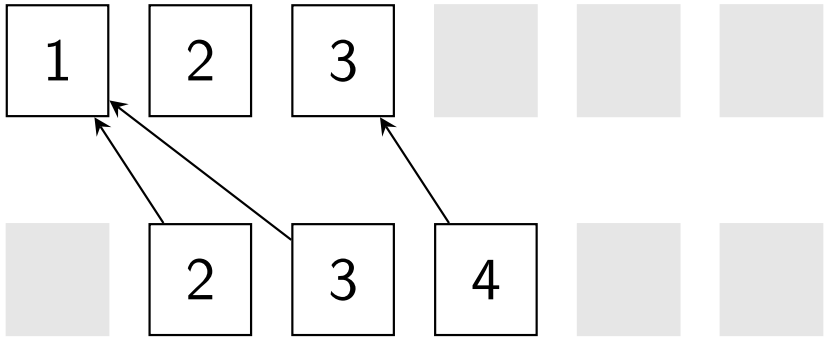


diagonal pruned node

active path pruned nodes

path pruned nodes

Graph Evolution

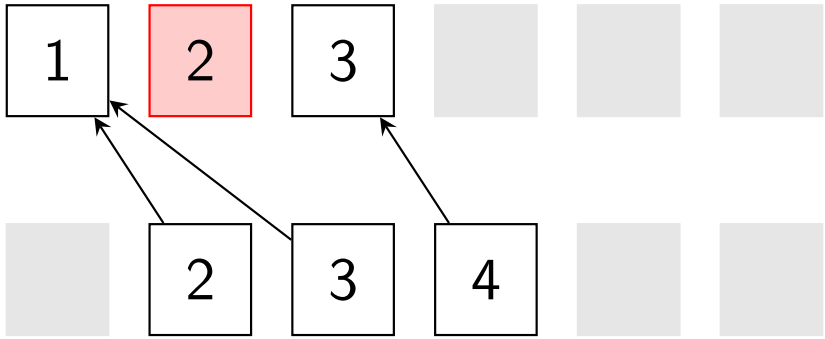


diagonal pruned node

active path pruned nodes

path pruned nodes

Graph Evolution

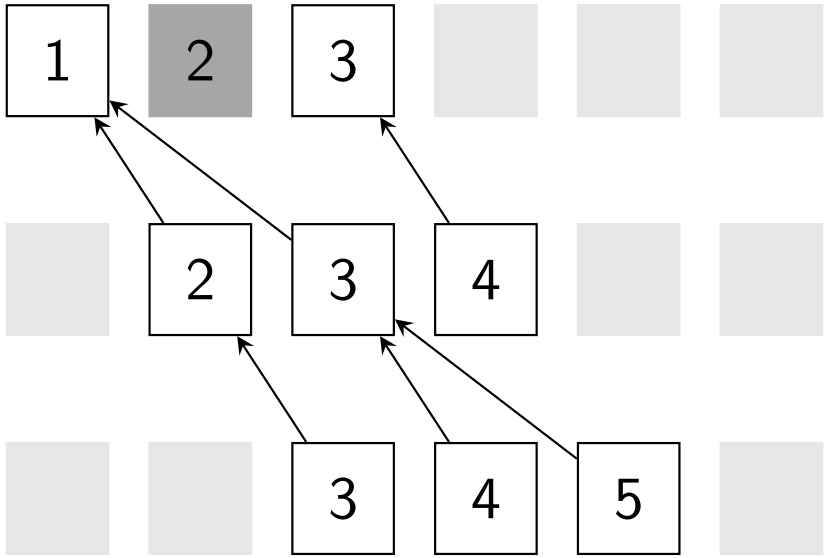


diagonal pruned node

active path pruned nodes

path pruned nodes

Graph Evolution

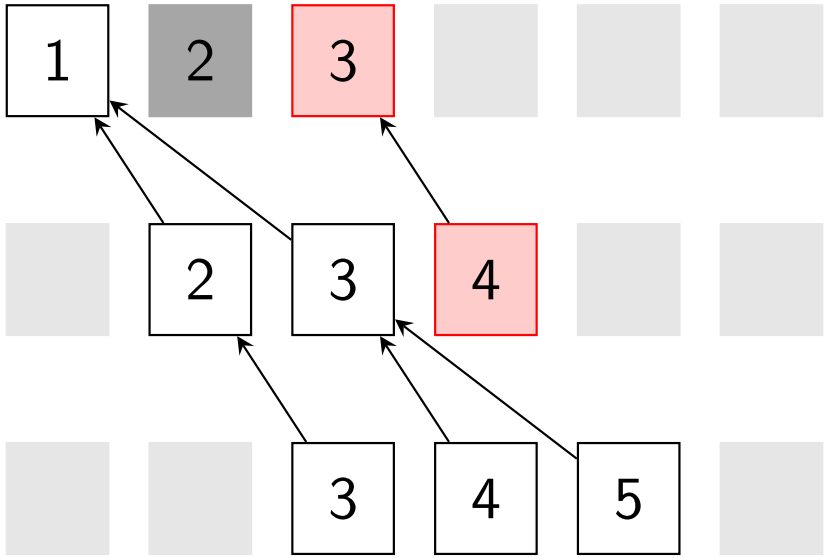


diagonal pruned node

active path pruned nodes

path pruned nodes

Graph Evolution

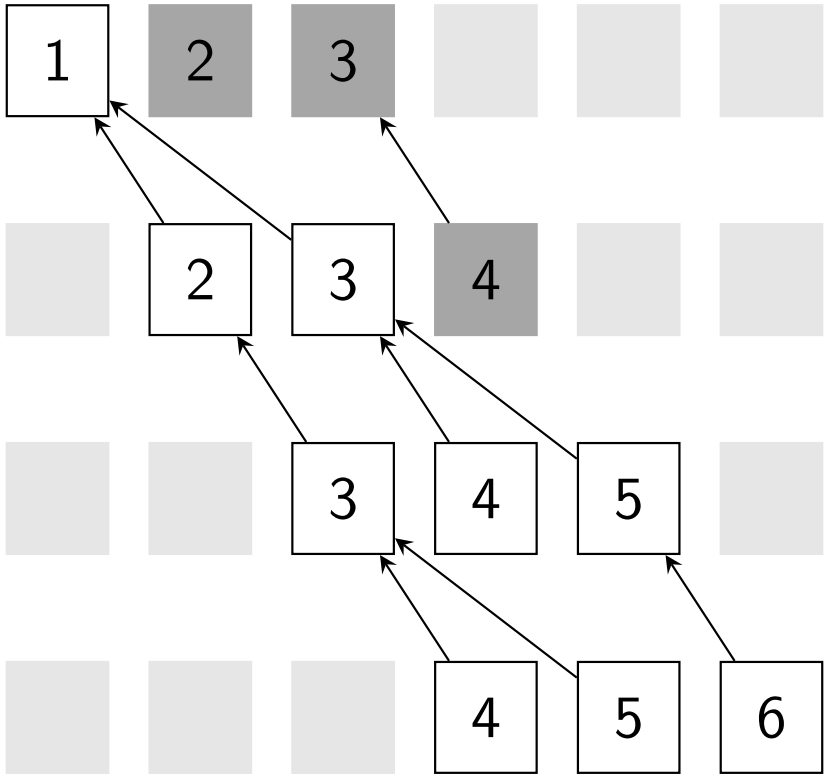


diagonal pruned node

active path pruned nodes

path pruned nodes

Graph Evolution

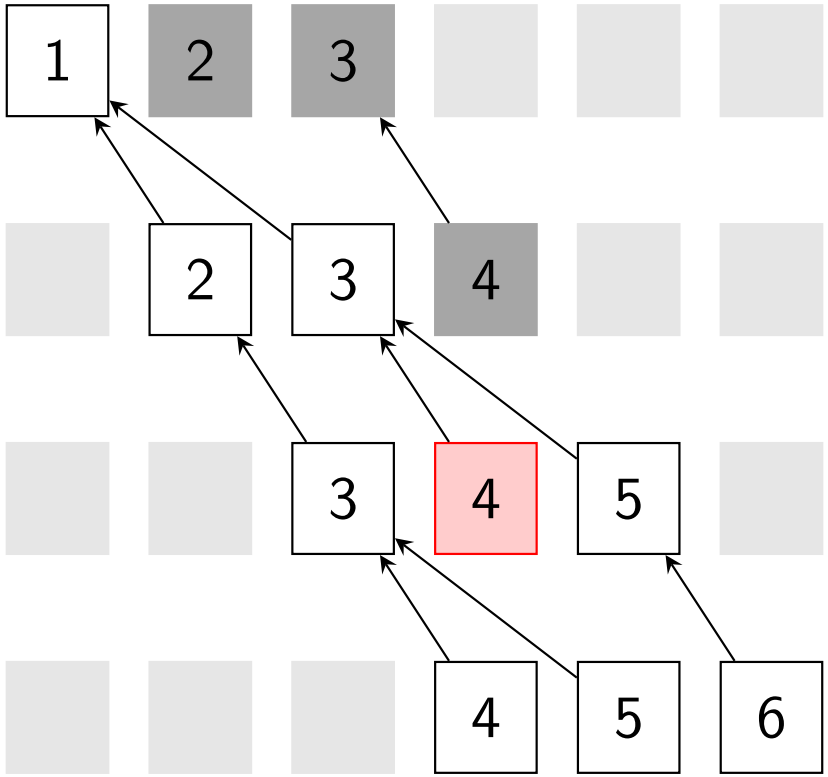


diagonal pruned node

active path pruned nodes

path pruned nodes

Graph Evolution

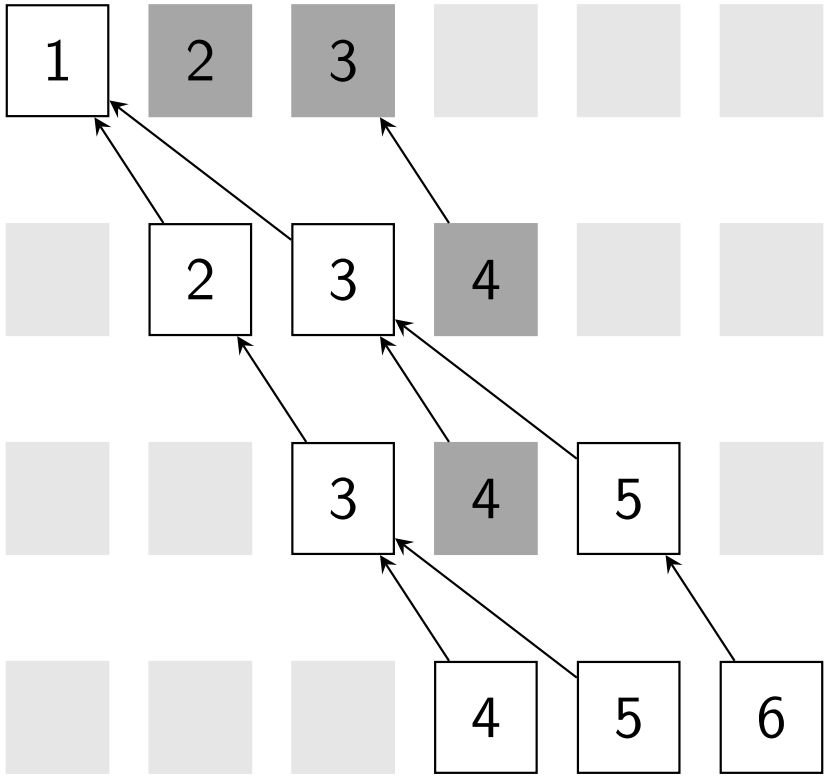


diagonal pruned node

active path pruned nodes

path pruned nodes

Graph Evolution



diagonal pruned node

active path pruned nodes

path pruned nodes

Total number of nodes: 24

Not computed nodes: 12

Path pruned nodes: 4

Experimental Configuration

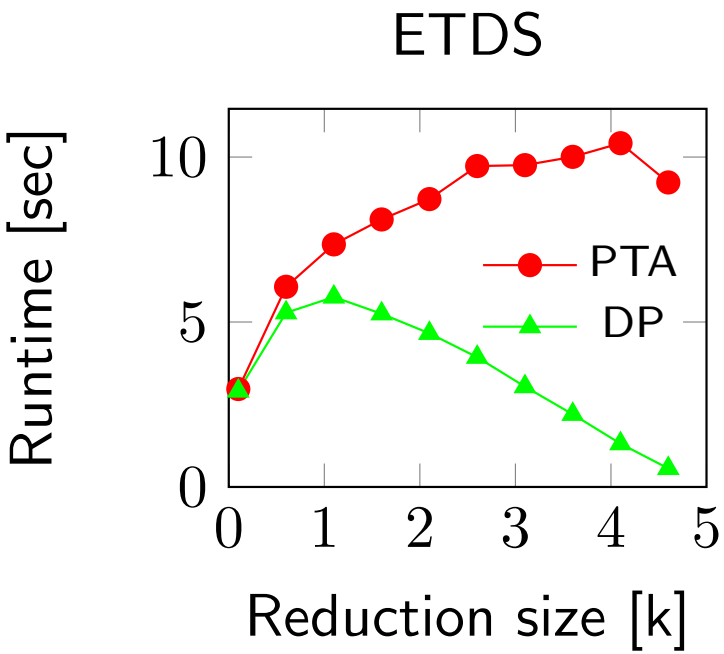
Synthetic Datasets

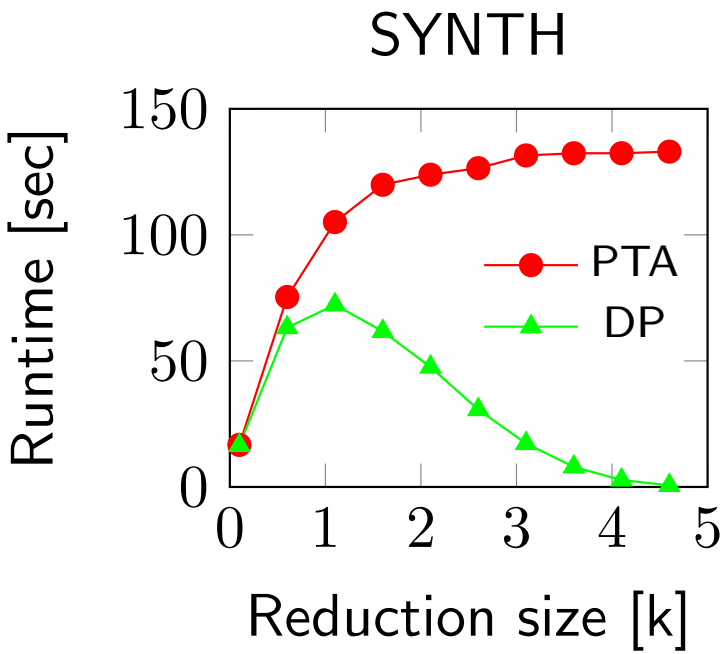
- SYNTH: random distributed values
- ETDS: evolution of employees in a company

Algorithm Comparisons

- PTA: original Algorithm
- DP: PTA with diagonal pruning
- SGP: Split point graph with diagonal and path pruning

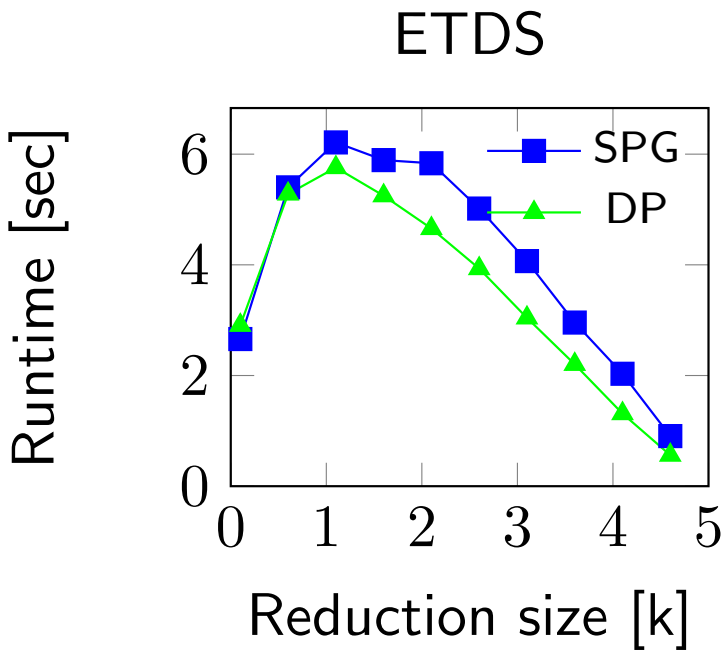
Runtime: PTA vs Diagonal Pruning

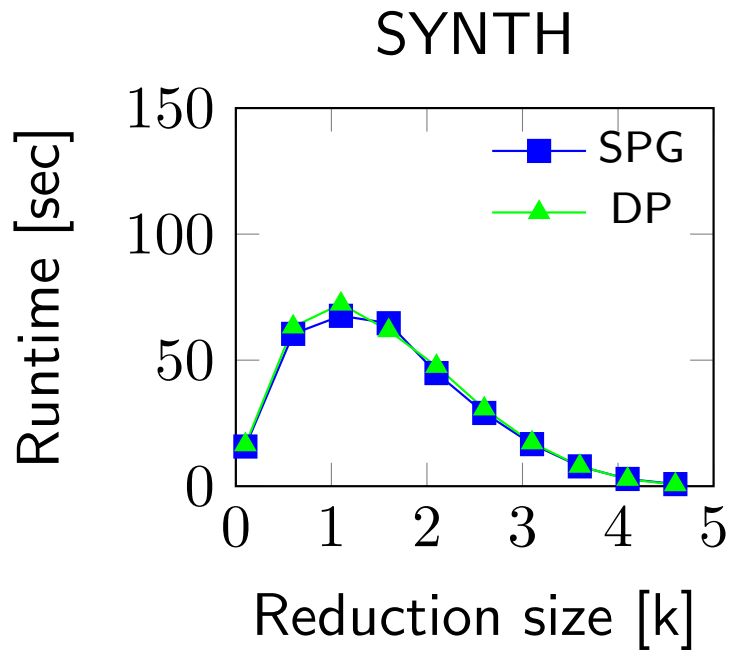




Diagonal pruning substantially reduces runtime

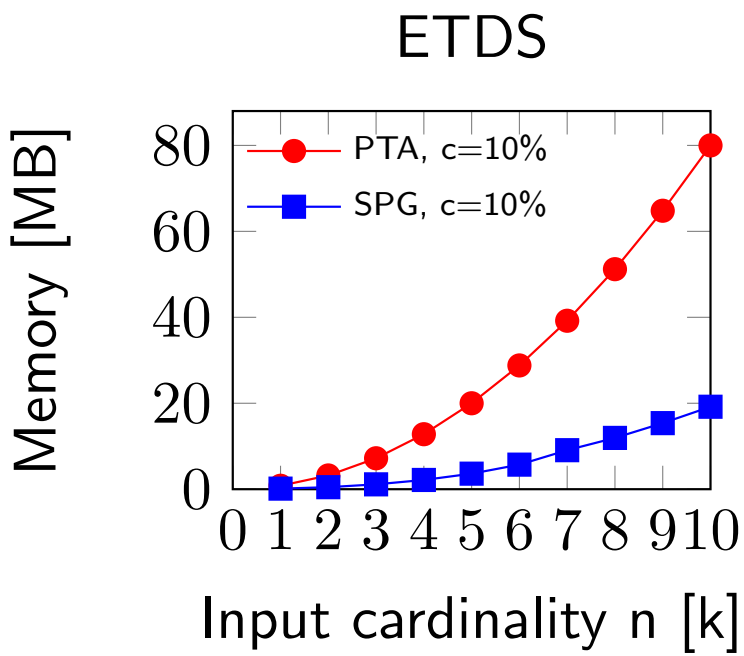
Runtime: Split Point Graph vs PTA with Diagonal Pruning

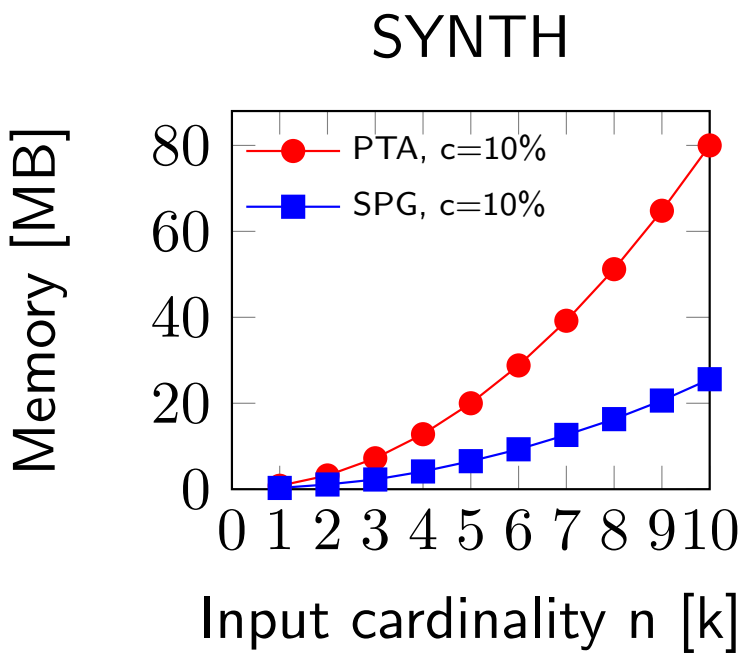




The overhead of the dynamic graph structure and path pruning is very small

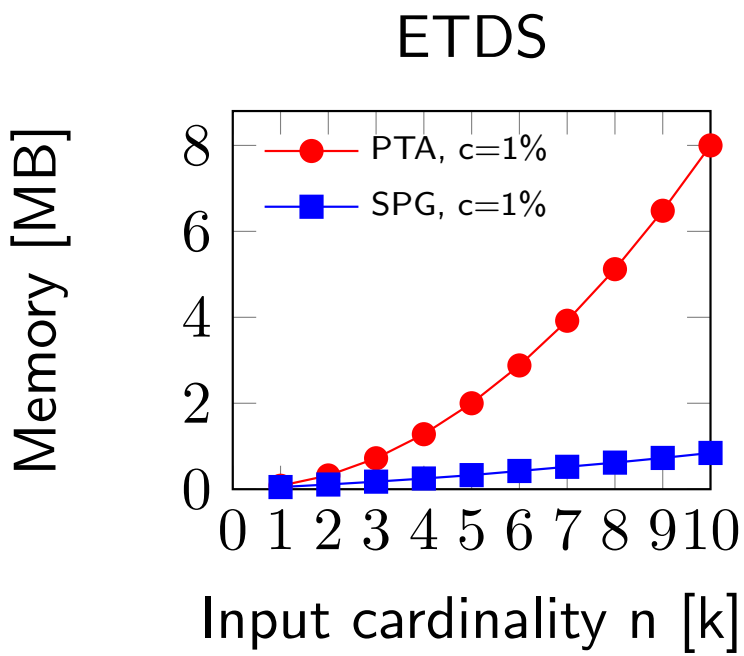
Space Efficiency: PTA vs SPG (compression to 10%)

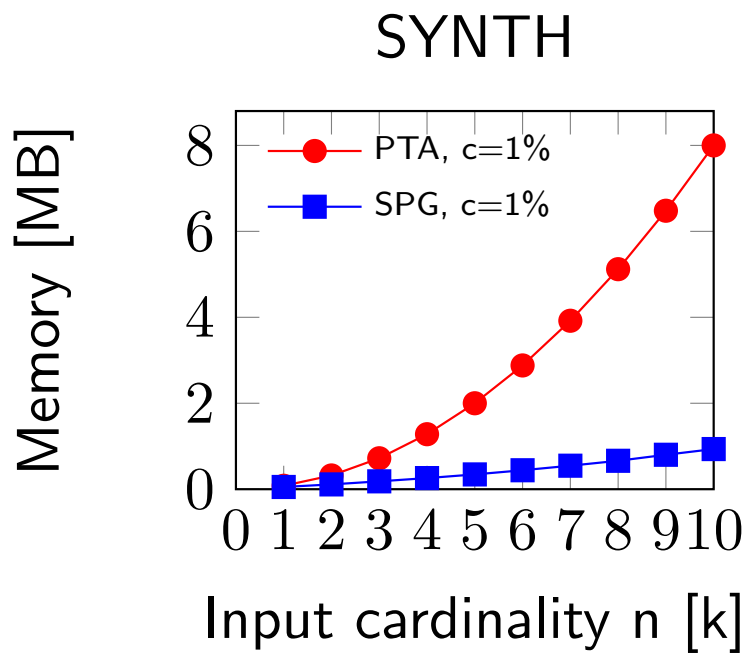




Graph Implementation with Diagonal Pruning and Path Pruning substantially reduces space consumption

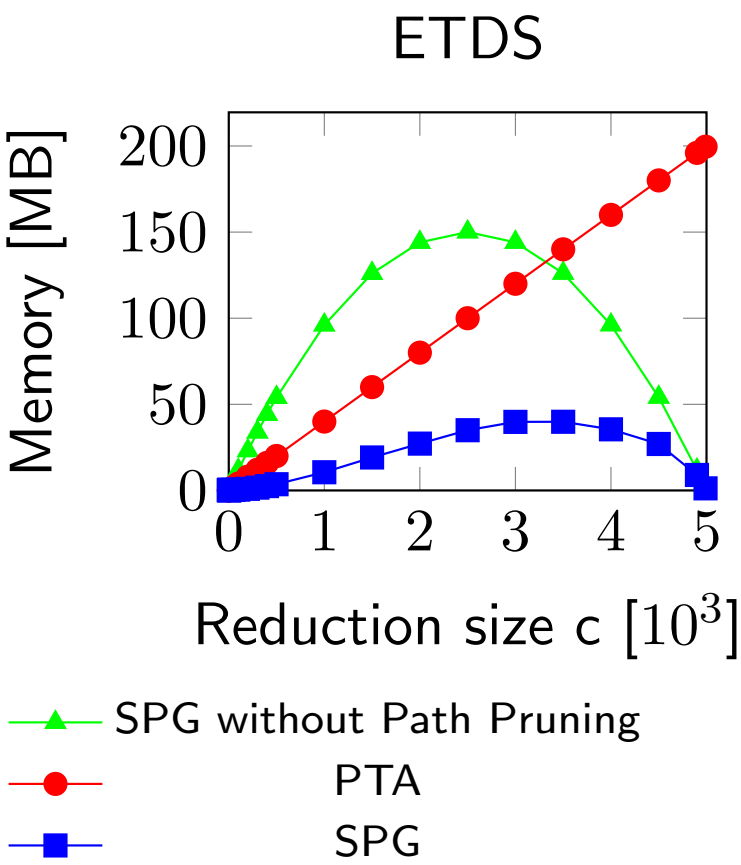
Space Efficiency: PTA vs SPG (compression to 1%)

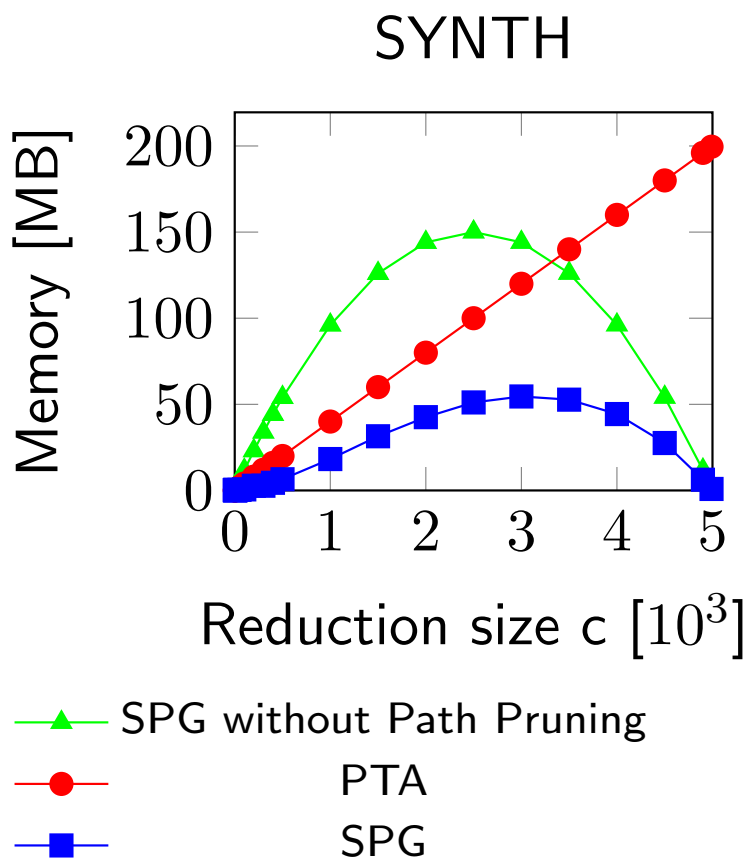




Graph Implementation with Diagonal Pruning and Path Pruning substantially reduces space consumption

Space Efficiency: Effect of Path Pruning





Path Pruning has a huge pruning effect. It prunes about 2/3 of the graph

Related Work

- Tuma, P.: Implementing Historical Aggregates in TempIS. Ph.D. thesis, Wayne State University, Detroit, Michigan (1992)
- Kline, N., Snodgrass, R.T.: Computing temporal aggregates. In: ICDE. pp. 222-231 (1995)
- Moon, B., Vega Lopez, I.F., Immanuel, V.: Efficient algorithms for large-scale temporal aggregation. IEEE Trans. Knowl. Data Eng. 15(3), 744-759 (2003)
- Tao, Y., Papadias, D., Faloutsos, C.: Approximate temporal aggregation. In: ICDE. pp. 190-201 (2004)
- Gordevičius, J., Gamper, J., Böhlen, M.H.: Parsimonious temporal aggregation. VLDB J. 21(3), 309-332 (2012)

Conclusion

- **Diagonal Pruning** reduces the **runtime** of the computation by reducing the search space of the DP scheme adopted by PTA
- **Split Point Graph** in combination with Path Pruning reduces **memory consumption**
- Experiments showed that the two optimizations reduce memory requirements to one third of the original PTA implementation

Future Work

- Generalization of split point graph for some classes of DP problems
- Computation of PTA queries while the ITA result is computed avoiding two step computation