XWAVE: Optimal and Approximate Extended Wavelets for Streaming Data

Sudipto Guha
Univ. of Penn.

Chulyun Kim
Seoul Nat'l Univ.

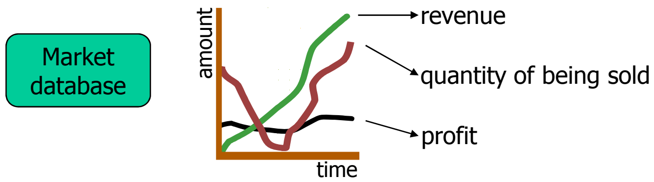
Kyuseok Shim
Seoul Nat'l Univ.

Outline

- Introduction
- Preliminaries & Previous Work
- XWAVE Algorithms
  - Optimal Algorithms
  - Approximation Algorithm
  - Adapting to Stream data
- Experimental Result
- Summary

Introduction

- Decision support system applications
  - Large database with multiple measures are common
  - Stringent response-time is required
  - Approximate query processing can provide a viable solution
  - Approximate wavelet synopses with multiple measures are frequently used



Wavelet – Haar wavelet

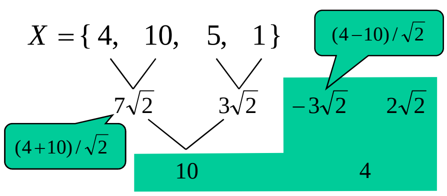
- Decomposition – takes linear time complexity
  - Basic operation

p	q

$\longrightarrow (p-q)/\sqrt{2}$

$\searrow (p+q)/\sqrt{2}$

  - Example



$$w = \{10, 4, -3\sqrt{2}, 2\sqrt{2}\}$$

Wavelet – Haar wavelet (Cont.)

- To Compress the original N data with Minimum L_2 error using B(<<N) Haar wavelet coefficients,
  - choose B coefficients with largest absolute value
- For given B = 2 and X = {4, 10, 5, 1}
  - $w = \{10, 4, -3\sqrt{2}, 2\sqrt{2}\}$
  - Two largest coefficients are $10, -3\sqrt{2}$
  - $w' = \{10, 0, -3\sqrt{2}, 0\}$
  - X' = {2, 8, 5, 5}
  - L_2 error = $4^2 + (2\sqrt{2})^2 = 24$

Traditional Wavelet Methods for Multiple Measures

Coordinate

- Multi-measure coefficients

measure

$$w = \left\{ \begin{bmatrix} 0 \\ 4 \\ 1 \end{bmatrix}, \begin{bmatrix} 3 \\ 5 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \\ -4\sqrt{2} \end{bmatrix}, \begin{bmatrix} 5 \\ 0 \\ 0 \end{bmatrix} \right\}$$

- Individual coefficient method
  - Select individual coefficient independently
  - Individual coefficient: <coordinate, measure, value>
    e.g. <2, 1, 3> <2, 2, 5> <4, 1, 5>

Traditional Wavelet Methods for Multiple Measures

- Multi-measure coefficients

Coordinate

measure

$$w = \left\{ \begin{bmatrix} 0 \\ 4 \\ 1 \end{bmatrix}, \begin{bmatrix} 3 \\ 5 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \\ -4\sqrt{2} \end{bmatrix}, \begin{bmatrix} 5 \\ 0 \\ 0 \end{bmatrix} \right\}$$

- Individual coefficient method
  - Select individual coefficient independently
  - Individual coefficient: <coordinate, measure, value>
    e.g. <2, 1, 3> <2, 2, 5> <4, 1, 5>
- Combined coefficient method
  - Select combined coefficient vector
  - Combined coefficient: <coordinate, coefficient-vector>
    e.g. <2, {3,5,0}> <4, {5,0,0}>

Drawbacks of Traditional Wavelet Methods for Multiple Measures

- Both method may result in suboptimal solutions
  - In individual coefficient method, the same coordinate information may be stored multiple times
  - In combined coefficient method, only a few coefficients in a vector may reduce the error
- Example: 2 coordinates, 3 measures, space of 7 numbers (i.e. store 2 individual or 1 combined coefficients)

Example	Individual	Combined
Coord. Values 1 100 50 0 2 0 100 0	<1,100,1><2,100,2> L_2 Error = 2500	<1,{100,50,0}> L_2 Error = 10000
Coord. Values 1 100 50 100 2 0 100 0	<1,100,1><1,100,3> L_2 Error = 12500	<1,{100,50,100}> L_2 Error = 10000

Extended Wavelet Coefficients

- [Deligiannakis and Roussopoulos: SIGMOD'03]
- A more flexible representation: <Bit, i, V>
  - Bit : A bitmap consisting of M bits
  - i : i-th coordinate
  - V : The list of stored coefficient values
  - e.g. <110, 2, {3,5}> <100, 4, {5}>

Coordinate

measure

$$w = \left\{ \begin{bmatrix} 0 \\ 4 \\ 1 \end{bmatrix}, \begin{bmatrix} 3 \\ 5 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \\ -4\sqrt{2} \end{bmatrix}, \begin{bmatrix} 5 \\ 0 \\ 0 \end{bmatrix} \right\}$$

Motivation for Extended Wavelet Coefficients

- For the given 2 coordinates having 3 measures
- Assuming that we have only space to store 7 numbers
  - We can store 2 individual, 1 combined, or 2 extended coefficients

Example	Individual	Combined	Extended
Coord. Values 1 100 50 0 2 0 100 0	<1,100,1><2,100,2> L_2 Error = 2500	<1,{100,50,0}> L_2 Error = 10000	<110,1,{100,50}> <010,2,{100}> L_2 Error = 0
Coord. Values 1 100 50 100 2 0 100 0	<1,100,1><1,100,3> L_2 Error = 12500	<1,{100,50,100}> L_2 Error = 10000	<101,1,{100,100}> <010,2,{100}> L_2 Error = 2500

Extended Wavelet Coefficients

- [Deligiannakis and Roussopoulos: SIGMOD'03]
- The goal isn't to minimize L_2 error but to maximize L_2 benefit

$$\sum_{i,j:w_{ij} \text{ is stored}} W_j w_{ij}^2$$

  - w_{ij}: the coefficient of j-th measure at i-th coordinate
- Proposed algorithms
  - DynL2: an optimal alg. with O(NMB) time and O(NMB) space
  - GreedyL2: an approximation alg. with $O(NM^2\log(NM))$ time and O(NM) space

DynL2

- Dynamic programming to get optimal L2 benefit
- OPT[u,b] is the optimal benefit with upto u-th coefficient using b space
- Optimal substructure

$$Opt[u,b].ben = \max\begin{cases} Opt[u-1,b].ben \\ Opt[u-1,b-space(w_{ij})].ben + W_j w_{ij}^2 \end{cases}$$

  - u = (i-1)M+j for w_{ij}
- Time complexity : O(NMB)
- Space complexity : O(NMB)

GreedyL2

- Greedy algorithm to get 2-approximation for L2 benefit
- Greedy choice property
  - Let per space benefit be benefit/required-space
  - Select a subset of a combined coefficient with maximum per space benefit
  - Use a data structure guaranteeing logarithmic time per operation (e.g. AVL tree or Heap)
- Time complexity : $O(NM^2\log(NM))$
- Space complexity : $O(NM)$

Drawback of the Previous Work

- Both optimal and approximate algorithms require at least linear space
- These algorithms cannot work for stream data
- Approximation for benefit doesn't guarantee error bound
  - Suppose a 2-approximation of the benefit is 50
  - The error is actually 50 times of optimum error

	benefit	error
optimal	99	1
2-approx.	50	50
max(o/a,a/o)	**1.98**	**50**

Our Contributions

- Propose faster optimal and approximation algorithms with less space
- Our approximation guarantees that its quality in terms of L2 error is at most that of the optimum solution with relaxed space
- Extend our algorithms to streaming data
- Experimental result confirms the our algorithms are much faster with less space requirement

Problem Formulation

- Given a *N* data points in D-dimensions with *M* measures, a storage constraint B, and a set of weights *W*,
- Select the extended wavelet coefficients to be stored within B space to minimize the error

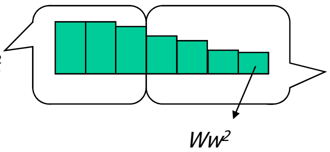
$$\sum_{i=1}^{N}\sum_{j=1}^{M} W_j \cdot e_{ij}^2 = \sum_{i,j:w_{ij} \text{ is not stored}} W_j w_{ij}^2$$

in a single pass over the data

OptWaveI – Optimal Algorithm

- We can *independently* select subsets of measures for each coordinate
- To store a subset C⊆{1,…,M} of i-th coordinate with |C|=p
  - Let BOTTOM[i,j] = $\sum_{j\ smallest\ items} W_k w_{ik}^2$ and TOP[i,j] = $\sum_{j\ largest\ items} W_k w_{ik}^2$
  - TOP[i,p] gives the maximum benefit
  - BOTTOM[i,M-p] gives the minimum error

∀C where |C|=p,
TOP[i,p] ≥ $\sum_{j\in C} W_j w_{ij}^2$



∀C where |C|=p,
BOTTOM[i,M-p] ≤ $\sum_{j\in\{1,\ldots,M\}-C} W_j w_{ij}^2$

OptWaveI (Cont.)

```
Procedure OptWaveI()
begin
   for i:=1 to N do
     for b:=1 to B do
       for p:=0 to M do
           if b-H-S*p ≥ 0
             NEWOPT[i,b]:=min(NEWOPT[i,b],
                   NEWOPT[i-1,b-H-S*p]+BOTTOM[i,M-p])
end
```

space for top p coefficients

- O(NMB) time
- O(B^2) space
  - To evaluate NEWOPT[i,j], only need the array of NEWOPT[i-1,j] for 1 ≤ j ≤ B
  - Each NEWOPT[i,j] needs O(B) space to store chosen coefficients

OptWaveII – Optimal Algorithm

- We *do not need* to examine all coordinates
- The optimum solution can store at most
  L= $\left\lfloor \frac{B}{S+H/M} \right\rfloor$ coefficients
  - Each coefficient w_{ij} takes up *at least* S+H/M space
- BEST[p] is the set of coordinate i in top-L largest TOP[i,p]
- There *exists* an optimum solution which has the coefficients in the coordinates only in

$$\bigcup_{p=1}^{M} BEST[p]$$

OptWaveII - Example

H: space for Bit and i,
S: space for a coefficient
H+S|V|: space for an extended coefficient

When H=2, S=1, and B=5,
L = $\left\lfloor \frac{5}{1+2/3} \right\rfloor$ = 3

$\bigcup_{p=1}^{3} BEST[p]$ = { 1, 2, 3, 6 }

Candidate Coefficients

Coord.	Values		
1	100	1	2
2	90	70	50
3	50	50	40
4	10	20	30
5	5	7	4
6	60	20	10
7	10	9	8
8	2	4	6

BEST[1]	BEST[2]	BEST[3]
1 (TOP[1,1]=10000)	2 (TOP[2,2]=13000)	2 (TOP[2,3]=15500)
2 (TOP[2,1]=8100)	1 (TOP[1,2]=10004)	1 (TOP[1,3]=10005)
6 (TOP[6,1]=3600)	3 (TOP[3,2]=5000)	3 (TOP[3,3]=6600)

OptWaveII (Cont.)

```
Procedure OptWaveII()
begin
1.  for each i∈∪BEST[p]
2.     for b:=1 to B do
3.        for p:=0 to M do
4.            if b-H-S*p ≥ 0
5.               NEWOPT[i,b]:=min(NEWOPT[i,b],
                     NEWOPT[prev_i,b-H-S*p]+BOTTOM[i,M-p]
end
```

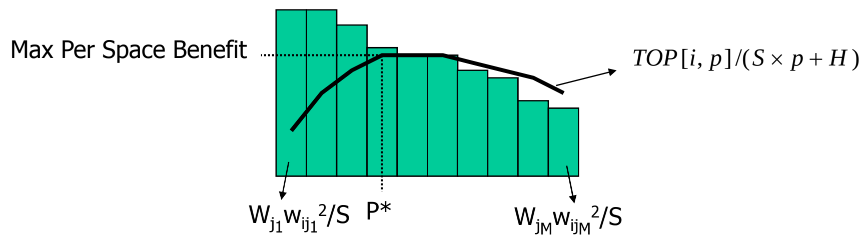
- $O(NM(\log M+\log L)+M^2LB)$ time
- $O(ML+B^2)$ space
  - Array BEST[p] needs O(ML) space

ApproxWave - Approximate Alg.

- Relax space constraint B slightly by (MS+H)
- Guarantee the quality in terms of *L2 error*
- Reduce space and time using a *O(B)-space* heap to store a *solution*
  - c.f. GreedyL2 uses O(NM)-space heap to store coefficient sets
- Each coefficient is inserted to the heap at most once - allows a single scan

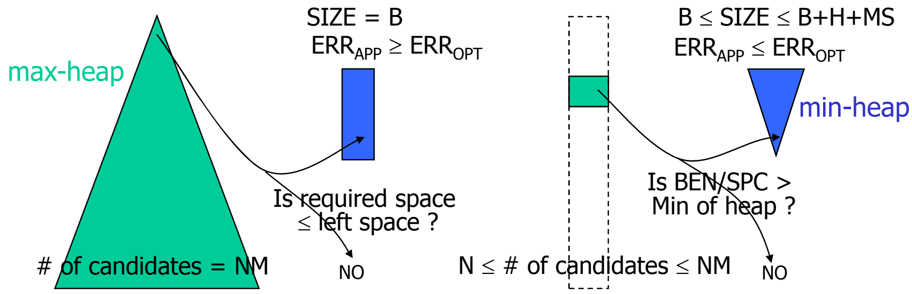
Approximation Algorithm (Cont.)

- In each i-th coordinate, we find TOP[i, p*] with max. per space benefit for $1 \le p \le M$
- If insertion into the heap succeeds, try insertion for rest of the coefficients
- Otherwise, move to the next coordinate



GreedyL2 vs. ApproxWave

[GreedyL2] **[ApproxWave]**

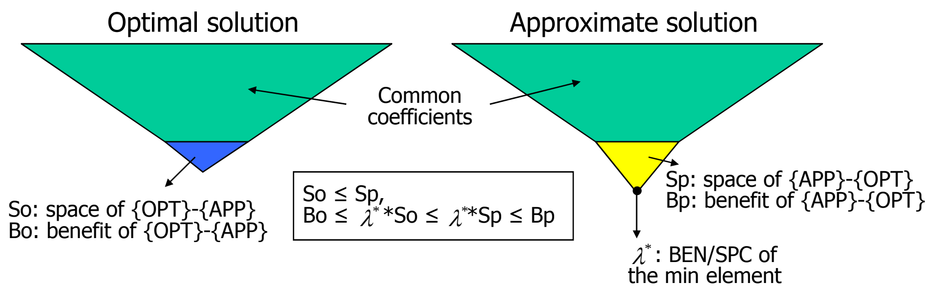


Time Complexity	$O(NM^2\log(NM))$
Space Complexity	O(NM)

Time Complexity	O(NM(logM+logB))
Space Complexity	O(B)

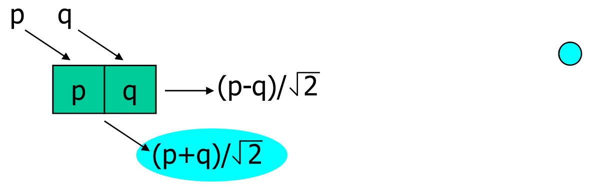
Approximation Algorithm (Cont.)

- We can show that the benefit of our solution is no less than the benefit of the optimum solution



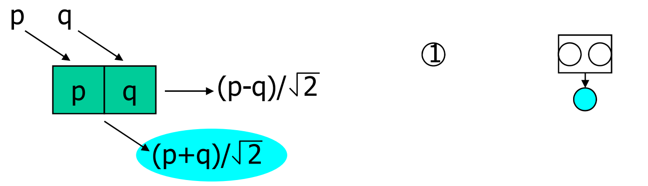
Adapting to Stream data

- Using O(logN) space we can compute the wavelet decomposition in a single pass.
- The order of the output coefficients will correspond to a post-order traversal of the tree



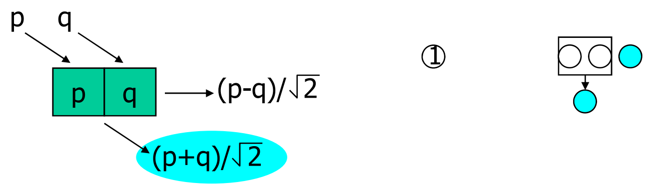
Adapting to Stream data

- Using O(logN) space we can compute the wavelet decomposition in a single pass.
- The order of the output coefficients will correspond to a post-order traversal of the tree



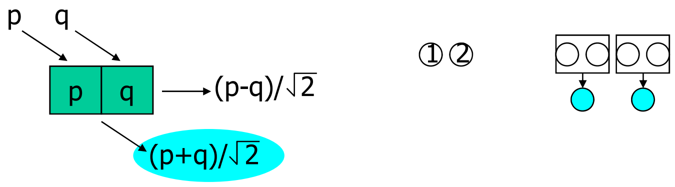
Adapting to Stream data

- Using O(logN) space we can compute the wavelet decomposition in a single pass.
- The order of the output coefficients will correspond to a post-order traversal of the tree



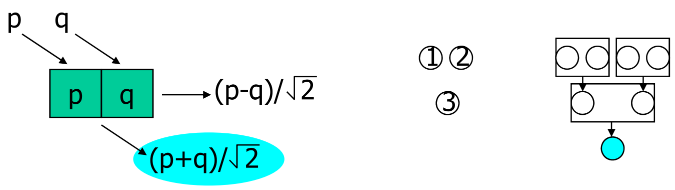
Adapting to Stream data

- Using O(logN) space we can compute the wavelet decomposition in a single pass.
- The order of the output coefficients will correspond to a post-order traversal of the tree



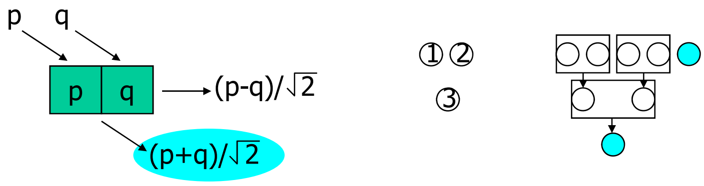
Adapting to Stream data

- Using O(logN) space we can compute the wavelet decomposition in a single pass.
- The order of the output coefficients will correspond to a post-order traversal of the tree



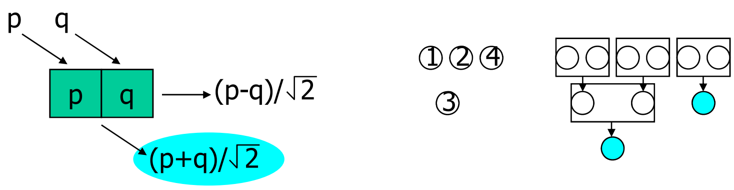
Adapting to Stream data

- Using O(logN) space we can compute the wavelet decomposition in a single pass.
- The order of the output coefficients will correspond to a post-order traversal of the tree



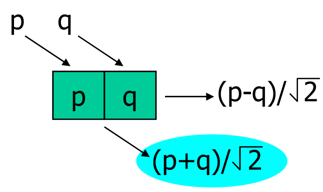
Adapting to Stream data

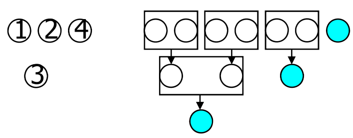
- Using O(logN) space we can compute the wavelet decomposition in a single pass.
- The order of the output coefficients will correspond to a post-order traversal of the tree



Adapting to Stream data

- Using O(logN) space we can compute the wavelet decomposition in a single pass.
- The order of the output coefficients will correspond to a post-order traversal of the tree



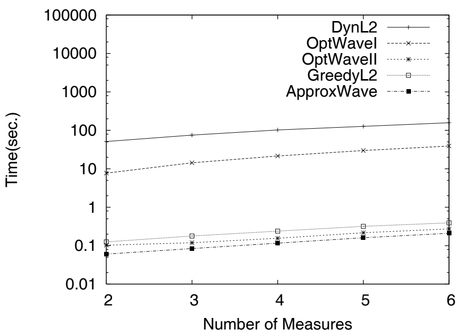


Experimental Result

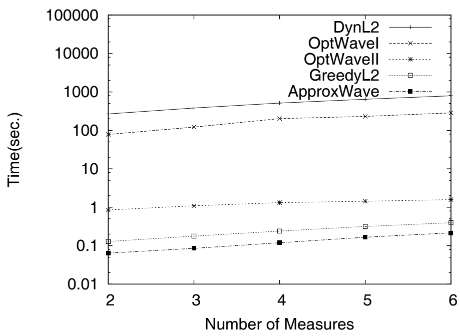
- Environment
  - Pentium-4 2.8 GHz with 512MB running Linux
  - GCC ver. 2.95.3
- Synthetic Data Sets
  - Data generator uses the Zipf functions to distribute values
  - Default parameters

Description	Value
# of dimensions	2
# of measures	30
Cardinality of every dimension	512
# of dense regions	10

Experiment on Varying M

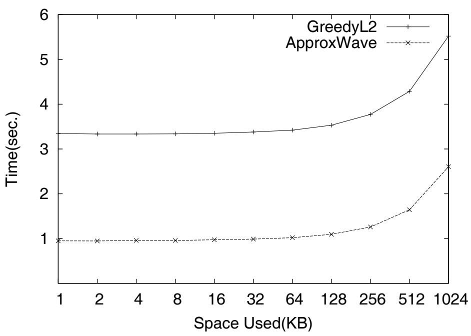


B=1K Bytes

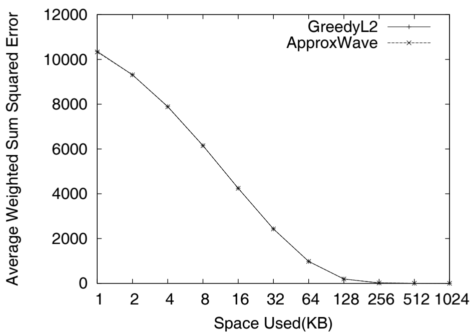


B=4K Bytes

Experiment on Varying B



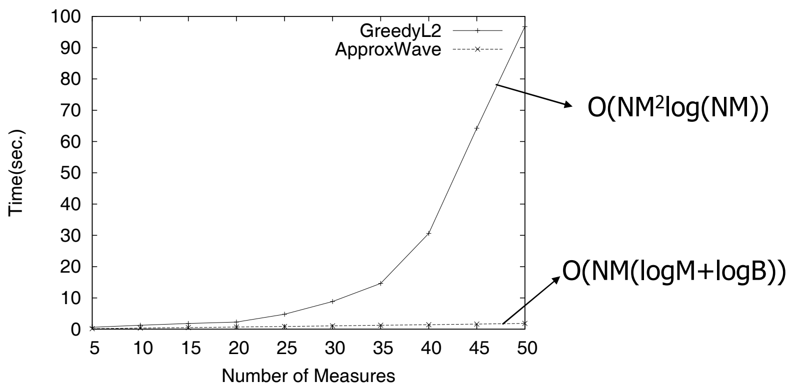
Execution Time



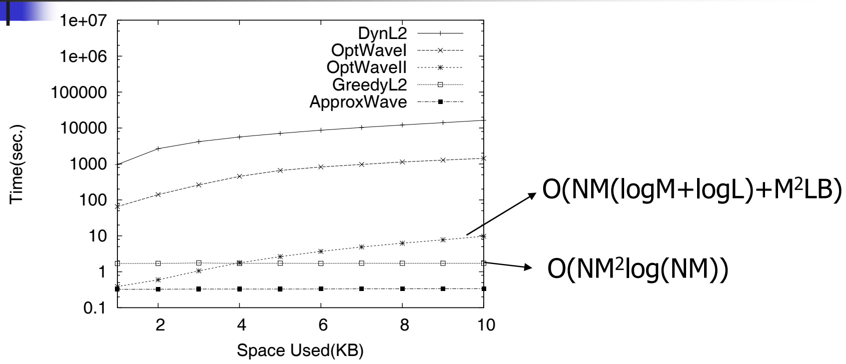
Error

Experiment on Large Values of M

- B = 1K Bytes



Experiment on Real-life Dataset



- Pacific Northwest 1-year weather measurement data
  - 525600 tuples and 6 measures (Solar irradiance, wind speed)
  - Weight vector : <3,3,2,2,1,1>

Summary

- [Deligiannakis and Roussopoulos: SIGMOD'03] gave an optimal algorithm and a 2-approximation algorithm with linear space for L2 benefit
  - Their approximation guarantees on benefit not on error
- We developed two faster optimal algorithms for L2 error with less space requirement
- We provide an approximation algorithm with at most the optimal error using relaxed space
- We demonstrated that our algorithms have significant performance benefits with much less space

Example: DynL2

Candidate Coefficients			
Coord.	Values		
1	100	1	2
2	90	70	50

H: space for Bit and i,
S: space for a coefficient
H+S|V|: space for an extended coefficient

When H=2, S=1, and B=5,
Benefit = 15500, Error = 10005

Opt

B / i	2 H	3 H+S	4 H+2S	5 H+3S
1	0	10000	10000	10000
2	0	10000	10001	10001
3	0	10000	10004	10005
4	0	10000	10004	10005
5	0	10000	13000	13000
6	0	10000	13000	15500

Force

B $H+S\|V\|$ / i	2 H	3 H+S	4 H+2S	5 H+3S
1	0	10000	10000	10000
2	0	10000	10001	10001
3	0	10000	10004	10005
4	0	8100	8100	8100
5	0	8100	13000	13000
6	0	8100	13000	15500

Example: GreedyL2

Candidate Coefficients			
Coord.	Values		
1	100	1	2
2	90	70	50

When H=2, S=1, and B=5,
Left Space = 5
Benefit = 0, Error = 25505

Coeff.	Sp.	Ben./Sp.	Selection
{100}	3	3333.3	-
{100,2}	4	2501	-
{100,2,1}	5	2001	-
{90}	3	2700	-
{90,70}	4	3250	-
{90,70,50}	5	3100	-

Example: GreedyL2

Candidate Coefficients			
Coord.	Values		
1	100	1	2
2	90	70	50

When H=2, S=1, and B=5,
Left Space = 2
Benefit = 10000, Error = 15505

Coeff.	Sp.	Ben./Sp.	Selection
{100}	3	3333.3	-
{100,2}	4	2501	-
{100,2,1}	5	2001	-
{90}	3	2700	-
{90,70}	4	3250	-
{90,70,50}	5	3100	-

Coeff.	Sp.	Ben./Sp.	Selection
{100}	3	3333.3	1
{2}	**1**	**4**	-
{2,1}	**2**	**2.5**	-
{90}	3	2700	-
{90,70}	4	3250	-
{90,70,50}	5	3100	-

Example: GreedyL2

Candidate Coefficients			
Coord.	Values		
1	100	1	2
2	90	70	50

When H=2, S=1, and B=5,
Left Space = 1
Benefit = 10004, Error = 15501

Coeff.	Sp.	Ben./Sp.	Selection
{100}	3	3333.3	1
{2}	1	4	2
{1}	**1**	**1**	-

Coeff.	Sp.	Ben./Sp.	Selection
{100}	3	3333.3	1
{2}	**1**	**4**	-
{2,1}	**2**	**2.5**	-
{90}	3	2700	-
{90,70}	4	3250	-
{90,70,50}	5	3100	-

Example: GreedyL2

Candidate Coefficients			
Coord.	Values		
1	100	1	2
2	90	70	50

When H=2, S=1, and B=5,
Left Space = 0
Benefit = 10005, Error = 15500

Coeff.	Sp.	Ben./Sp.	Selection
{100}	3	3333.3	1
{2}	1	4	2
{1}	**1**	**1**	-

Coeff.	Sp.	Ben./Sp.	Selection
{100}	3	3333.3	1
{2}	1	4	2
{1}	1	1	3

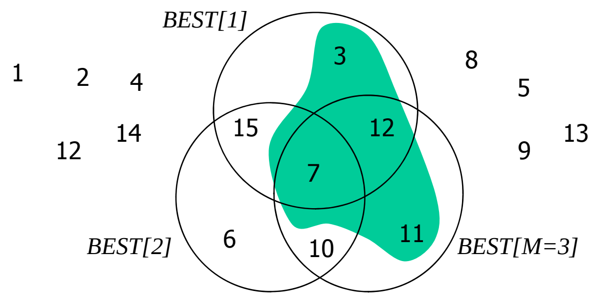
Example: OptWaveI

Candidate Coefficients			
Coord.	Values		
1	100	1	2
2	90	70	50

When H=2, S=1, and B=5,
Benefit = 15500, Error = 10005

i \ B	0 -	1 -	2 H	3 H+S	4 H+2S	5 H+3S
0	25505 {}	25505 {}	25505 {}	25505 {}	25505 {}	25505 {}
1	25505 {}	25505 {}	25505 {}	15505 {(0,1)}	15501 {(0,1),(0,3)}	15500 {(0,1),(0,3),(0,2)}
2	25505 {}	25505 {}	25505 {}	15505 {(0,1)}	12505 {(1,1),(1,2)}	10005 {(1,1),(1,2),(1,3)}

OptWaveII (Cont.)



Approximation Algorithm (Cont.)

- Suppose $w_{ij_1}, w_{ij_2}, \ldots, w_{ij_M}$ are the coefficients with index i such that $W_{j_u} w_{ij_u}^2 \geq W_{j_v} w_{ij_v}^2$ whenever $u \leq v$.
- RATIO[i].wt = $\max_{p=1,\ldots,M} TOP[i,p]/(S \times p + H)$
- RATIO[i].p is the smallest value p of the maximum per space benefit
- If RATIO[i].wt does not enter to the heap,
- We don't need to examine the rest of coefficient index from RATIO[i].p to j_M

Approximation Algorithm (Cont.)

- Keep an approximate solution in a heap with the space ≤ (B+M*S+H)
- For each i-th coordinate, suppose that
  - $w_{ij_1}, w_{ij_2}, \ldots, w_{ij_M}$ are the coefficients with index i such that $W_{j_u} w_{ij_u}^2 \geq W_{j_v} w_{ij_v}^2$ whenever $u \leq v$.
  - wt = $\max_{p=1,\ldots,M} TOP[i,p]/(S \times p + H)$
  - p is the smallest value j_k with the above wt
- If w_{ip} cannot be inserted into the heap, the coefficients $\{w_{i(p+1)}, w_{ij_2}, \ldots, w_{ij_M}\}$ are guaranteed not to be inserted
- Otherwise, we put $\{w_{ij_1}, w_{ij_2}, \ldots, w_{ip}\}$ into the heap and examine the rest of coefficients $\{w_{i(p+1)}, w_{ij_2}, \ldots, w_{ij_M}\}$

Approximation Algorithm (Cont.)

- Abstract pseudo code

```
Procedure ApproxWave()
begin
   for i:=1 to N do
      if Free ≥ 0
         Insert RATIO[i].p to heap
      else if RATIO[i].wt > min of heap
         Delete overflow elements and Insert RATIO[i].p to heap
      for u:=RATIO[i].p+1 to M
         if Free ≥ 0
            Insert Ww_u/S to heap
         else if Ww_u/S > min of heap
            Delete overflow elements and Insert Ww_u/S to heap
   Include all the coefficients stored in the heap
end
```

- Once we reach Free < 0
  - We keep $-(M*S+H) \le Free < 0$ in all step

ApproxWave - Example

Candidate Coefficients

Coord.	Values		
1	100	1	2
2	90	70	50

When H=2, S=1, and B=5,
RATIO[1].p = 1, RATIO[1].wt = 3333.3
RATIO[2].p = 1, RATIO[2].wt = 3250
Free space = 5

Heap <i, m, wt, IS_RATIO[].P>

ApproxWave - Example

Candidate Coefficients			
Coord.	Values		
1	100	1	2
2	90	70	50

When H=2, S=1, and B=5,
RATIO[1].p = 1, RATIO[1].wt = 3333.3
RATIO[2].p = 1, RATIO[2].wt = 3250
Free space = 2

Heap <i, m, wt, IS_RATIO[].P>
<1, 1, 3333.3, YES>

ApproxWave - Example

Candidate Coefficients			
Coord.	Values		
1	100	1	2
2	90	70	50

When H=2, S=1, and B=5,
RATIO[1].p = 1, RATIO[1].wt = 3333.3
RATIO[2].p = 1, RATIO[2].wt = 3250
Free space = 1

Heap <i, m, wt, IS_RATIO[].P>
<1, 2, 1, NO>
<1, 1, 3333.3, YES>

ApproxWave - Example

Candidate Coefficients			
Coord.	Values		
1	100	1	2
2	90	70	50

When H=2, S=1, and B=5,
RATIO[1].p = 1, RATIO[1].wt = 3333.3
RATIO[2].p = 1, RATIO[2].wt = 3250
Free space = 0

Heap <i, m, wt, IS_RATIO[].P>
<1, 2, 1, NO>
<1, 3, 4, NO>
<1, 1, 3333.3, YES>

ApproxWave - Example

Candidate Coefficients			
Coord.	Values		
1	100	1	2
2	90	70	50

When H=2, S=1, and B=5,
RATIO[1].p = 1, RATIO[1].wt = 3333.3
RATIO[2].p = 1, RATIO[2].wt = 3250
Free space = -4

Heap <i, m, wt, IS_RATIO[].P>
<1, 2, 1, NO>
<1, 3, 4, NO>
<2, 2, 3250, YES>
<1, 1, 3333.3, YES>

ApproxWave - Example

Candidate Coefficients

Coord.	Values		
1	100	1	2
2	90	70	50

When H=2, S=1, and B=5,
RATIO[1].p = 1, RATIO[1].wt = 3333.3
RATIO[2].p = 1, RATIO[2].wt = 3250
Free space = -3

Heap <i, m, wt, IS_RATIO[].P>
<1, 3, 4, NO>
<2, 2, 3250, YES>
<1, 1, 3333.3, YES>

ApproxWave - Example

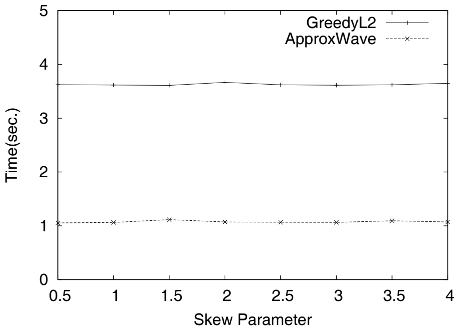
Candidate Coefficients

Coord.	Values		
1	100	1	2
2	90	70	50

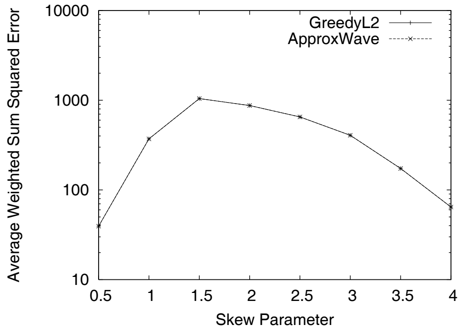
When H=2, S=1, and B=5,
RATIO[1].p = 1, RATIO[1].wt = 3333.3
RATIO[2].p = 1, RATIO[2].wt = 3250
Free space = -4, Used Space = 9
Benefit = 25505, Error = 1

Heap <i, m, wt, IS_RATIO[].P>
<1, 3, 4, NO>
<2, 3, 2500, NO>
<2, 2, 3250, YES>
<1, 1, 3333.3, YES>

Experiment on Varying Skew



Execution Time



Error