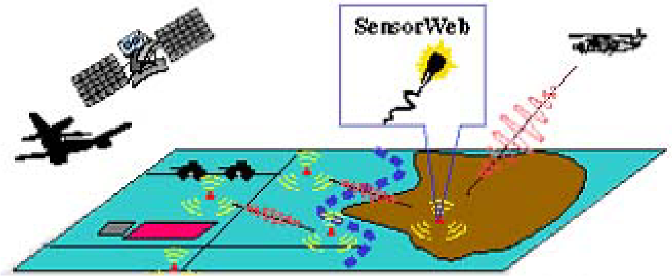


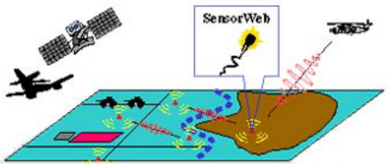
Estimating Entropy and Divergence of Sensor Data

Haixiao Cai, Sanjeev Kulkarni, Sergio Verdu

Princeton University

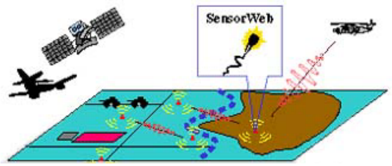
SensorWeb MURI Review Meeting

June 14, 2002



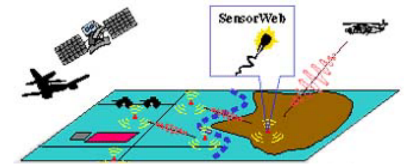
Problem and Motivation

- Many simple, myopic sensors
- Would like to fuse myopic information to attain more global view of battlefield scenario
- Need to understand relationships between sensor outputs



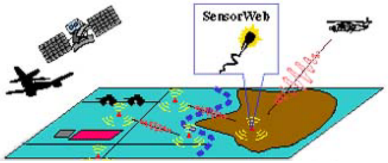
Difficulties and Complications

- Need fast, low-complexity algorithms (but can exploit temporal information)
- Unknown scene and sensor geometry
- Complex, dynamic environment
- Multiple, widely separated, possibly dynamic sensors
- Uncalibrated, possibly multi-modal, sensors (unknown parameters, geometry)
- Noise



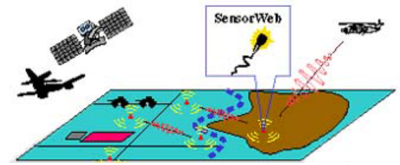
A Key Task and Possible Approach

- Correspondence and fusion are
  - difficult with many parameters (or nonparametric) and with nonlinear transformations
  - meaningless without common information in data streams
- A key task is to determine how related are the outputs of two sensors
- A possible approach:
  - Determine sensor pairs/groups that are highly related
  - Perform correspondence and fusion first with maximally related sensors
  - Incorporate other sensor outputs



A “Distilled” Problem

- *The Problem:* How to estimate the entropy and divergence of two sources based only on one realization from each source ?
- *Assumption:* Both are finite-alphabet, finite-memory, stationary sources.
- *Our goal:* Want good estimates, fast convergence, and reasonable computational complexity.



Definitions

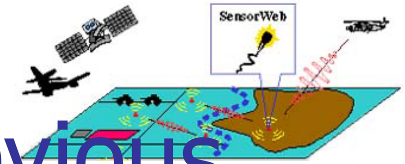
Entropy and Divergence rates are defined as follows:

$$H(q_z) = \lim_{n->\infty} \frac{1}{n} E[\log \frac{1}{q_z(z^n)}]$$

$$= \sum_{s \in S} q_z(s) \sum_{a \in \chi} q_z(a \mid s) \log \frac{1}{q_z(a \mid s)}.$$

$$D(q_z \parallel p_x) = \lim_{n->\infty} \frac{1}{n} E[\log \frac{q_z(z^n)}{p_x(z^n)}]$$

$$= \sum_{s \in S} q_z(s) \sum_{a \in \chi} q_z(a \mid s) \log \frac{q_z(a \mid s)}{p_x(a \mid s)}.$$



Possible Approaches and Previous Work

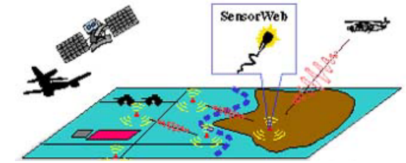
- Model Based: Assume model to get empirical distribution. Then plug-in to formulas.
- Universal Algorithms: Use universal compression algorithm or related methods to estimate entropy. E.g., can use Lempel-Ziv string matching method inspired by LZ data compression algorithm (Wyner, Ziv '89, Kontoyiannis et al. '94,'96,'98, Quas '95, Shields '92).

  L_n =1 + length of the longest prefix that recurs

$$\hat{H}_n = \frac{\log_2 n}{L_n}$$

  e.g., for "abaabaaa", L_8=5.

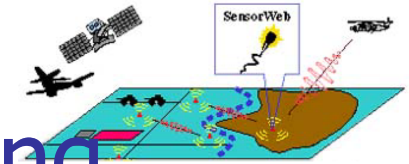
- Very little previous work on divergence estimation (Ziv, Merhav '93)



Weaknesses and Overcoming Them

- Model-based methods need to know order of model and are not universal.
- Existing universal methods (e.g., LZ-based methods) converge very slowly.
- Can we retain advantages of both?

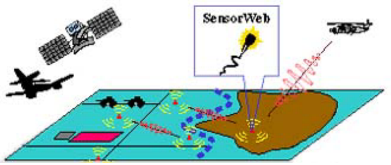
- Recent interest in Burrows-Wheeler Transform (block sorting) for data compression (Burrows, Wheeler '94)
- Can this be adapted for entropy and divergence estimation?



Burrows-Wheeler Block Sorting Transform

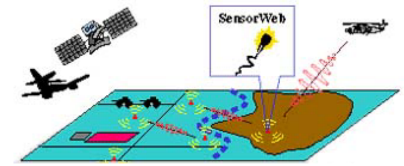
1. Get every shift of the input sequence.
2. Sort them alphabetically.
3. Output the last column of the sorted table, as well as the position of the original sequence.

- Reversible transform.
- Can be implemented in *O(n)* time/space complexity.
- Sorts input sequence according to context.
- Output is close to a piecewise i.i.d. distribution (Effros, Visweswariah, Kulkarni, Verdu ’02).



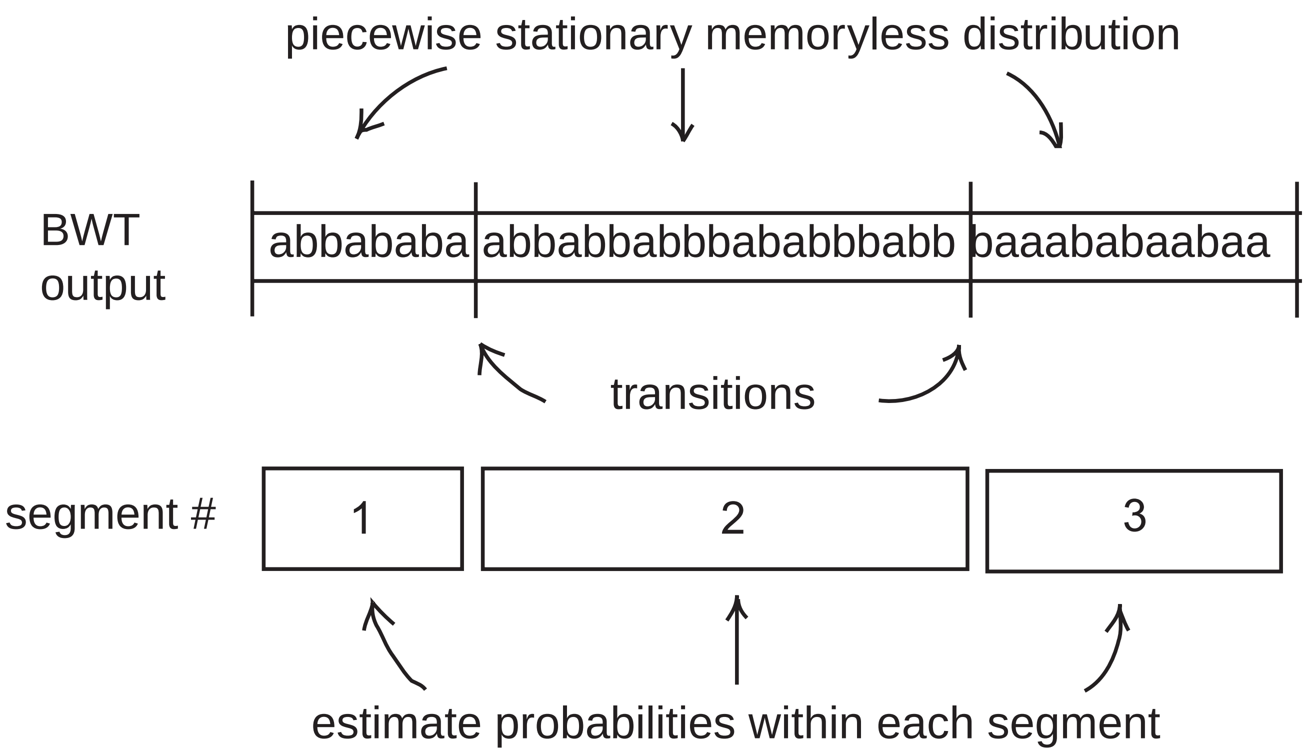
BWT: Example

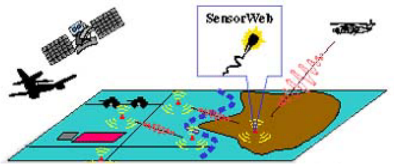
	EOF symbol		sorted table			output:
input →	**banana$**		anana$	b		b
cyclic shifts	$banana		ana$ba	n		n
	a$banan	sort	a$bana	n	output	n
	na$bana	⟹	**banana**	**$**	⟹	**$**
	ana$ban		nana$b	a		a
	nana$ba		na$ban	a		a
	anana$b		$banan	a		a



Entropy Estimation – Basic Idea

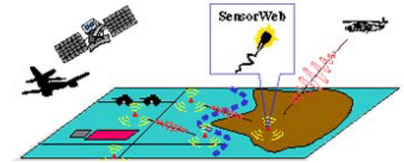
The BWT sorts the sequence according to context, and the output is close to piecewise i.i.d.



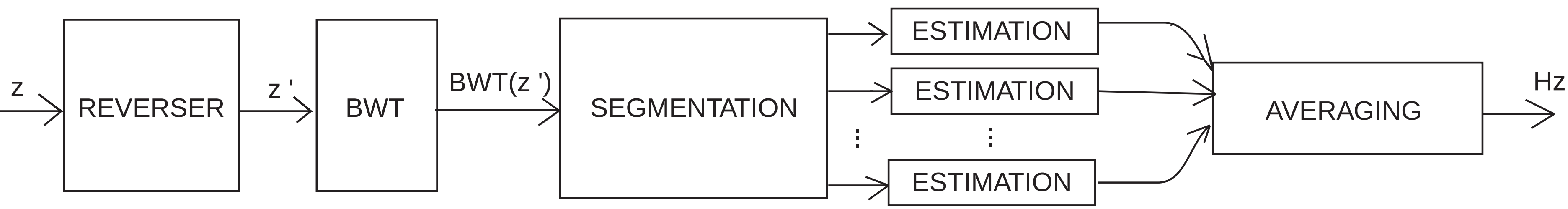


Entropy Estimation via BWT

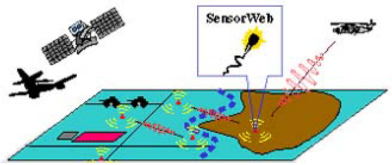
1. Reverse original sequence $\mathbf{z}^n$.
2. Run BWT on reversed sequence.
3. Divide BWT output into segments, according to transitions of distribution.
4. Estimate letter probabilities in each i.i.d. segment.
5. Combine estimates in each segment (average log of probability of each segment) to calculate probability of $\mathbf{z}^n$ and in turn the entropy.



Entropy Estimation via BWT: cont.



Block diagram of the entropy estimator

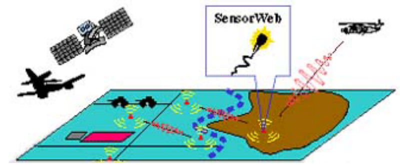


Divergence Estimation – Basic idea

- Recall,

$$D(q \| p) = \lim_{n->\infty} \frac{1}{n}\left(\log \frac{1}{p_x(z^n)} - \log \frac{1}{q_z(z^n)} \right)$$

- Second term is just entropy of **z**.
- First term (cross term) is probability of **z**, *but according to distribution for* **x**.
- Key idea to estimate cross term is *joint* BWT of both **x** and **z**.

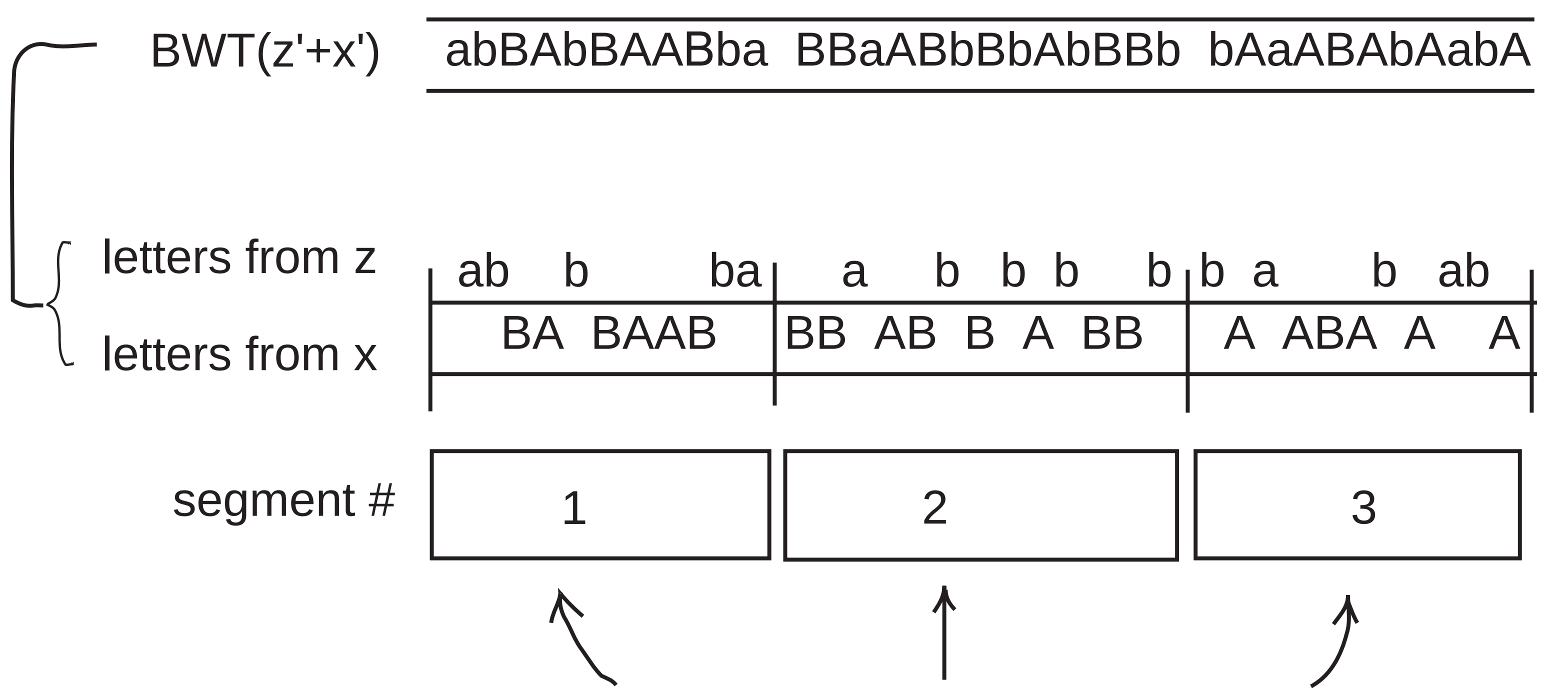


Estimation of Cross Term

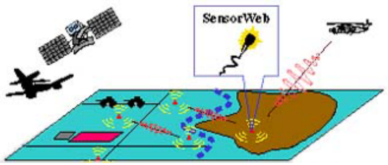
- Joint BWT of **x** and **z**:
  1. Concatenate **x** and **z**, adding '$' to the end of each.
  2. Sort the table of cyclic shifts. Note whether each letter comes from **x** or from **z** during the BWT sorting**.**
  3. Output the last column.

  This gather letters with same context from both **x** and **z** together.
- Segment the joint BWT output *according to* **x**.
- Compute probability of **z** *using probabilities estimated according to* **x**.

Estimation of Cross Term – continued

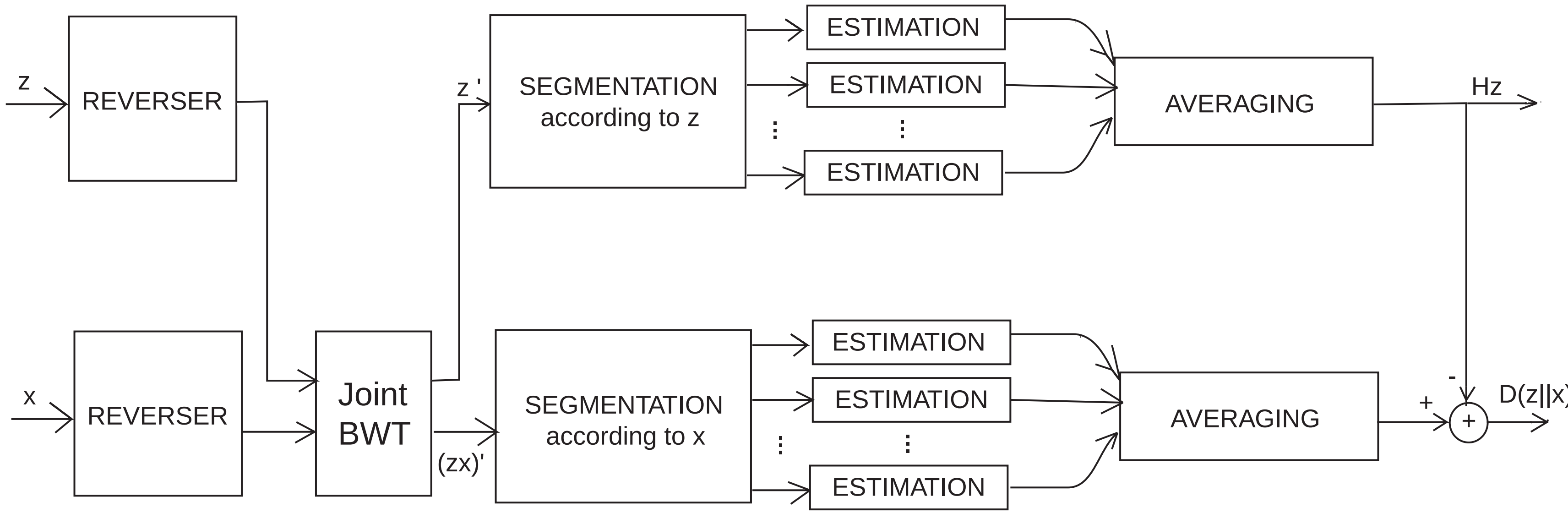


estimate $p_x(A)$, $p_x(B)$ within each segment

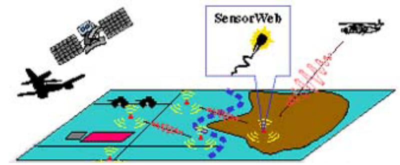


Divergence Estimation via BWT

1. Estimate entropy $H_z = -1/n \log q_z(z^n)$.
2. Estimate the cross term $-1/n \log p_x(z^n)$.
3. Subtract the entropy term from the cross term.

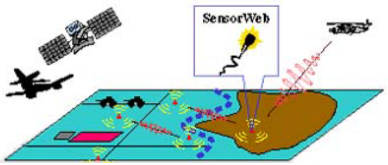


Block diagram of the divergence estimator



Approaches to Segmentation

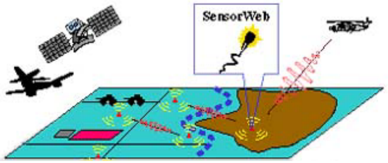
- Uniform segmentation: divide the BWT output sequence into equal-length segments.
- Adaptive segmentation: make a new segment only when we detect a transition. The number and length of segments are adapted to the source.



Uniform Segmentation

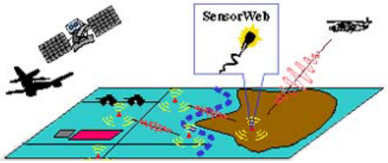
Divide the sequence (of length n) into equal-length segments of length w(n). Want those segments containing transitions to be negligible, but don't want too many segments.

segment#		j	j+1	j+2	j+3
		←w(n)→	←w(n)→	←w(n)→	
BWT(z')......	aabbb	aabbababba	abbababbab	babbbbabab	bababaa ......
		$q_z(j,a)=0.5$	$q_z(j+1,a)=0.4$	$q_z(j+2,a)=0.3$	
		$q_z(j,b)=0.5$	$q_z(j+1,b)=0.6$	$q_z(j+2,b)=0.7$	



Adaptive Segmentation

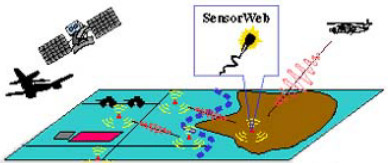
- Uniform segmentation ends up with many more segments than the actual number of states.
- Instead of using equal-length segments, estimate positions of transitions, and make new segment only when we detect a transition.
- Two-level blocks are introduced. In level-1 (with block length k_1), roughly locate the positions. In level-0 (with block length k_0), refine our estimate of the positions.

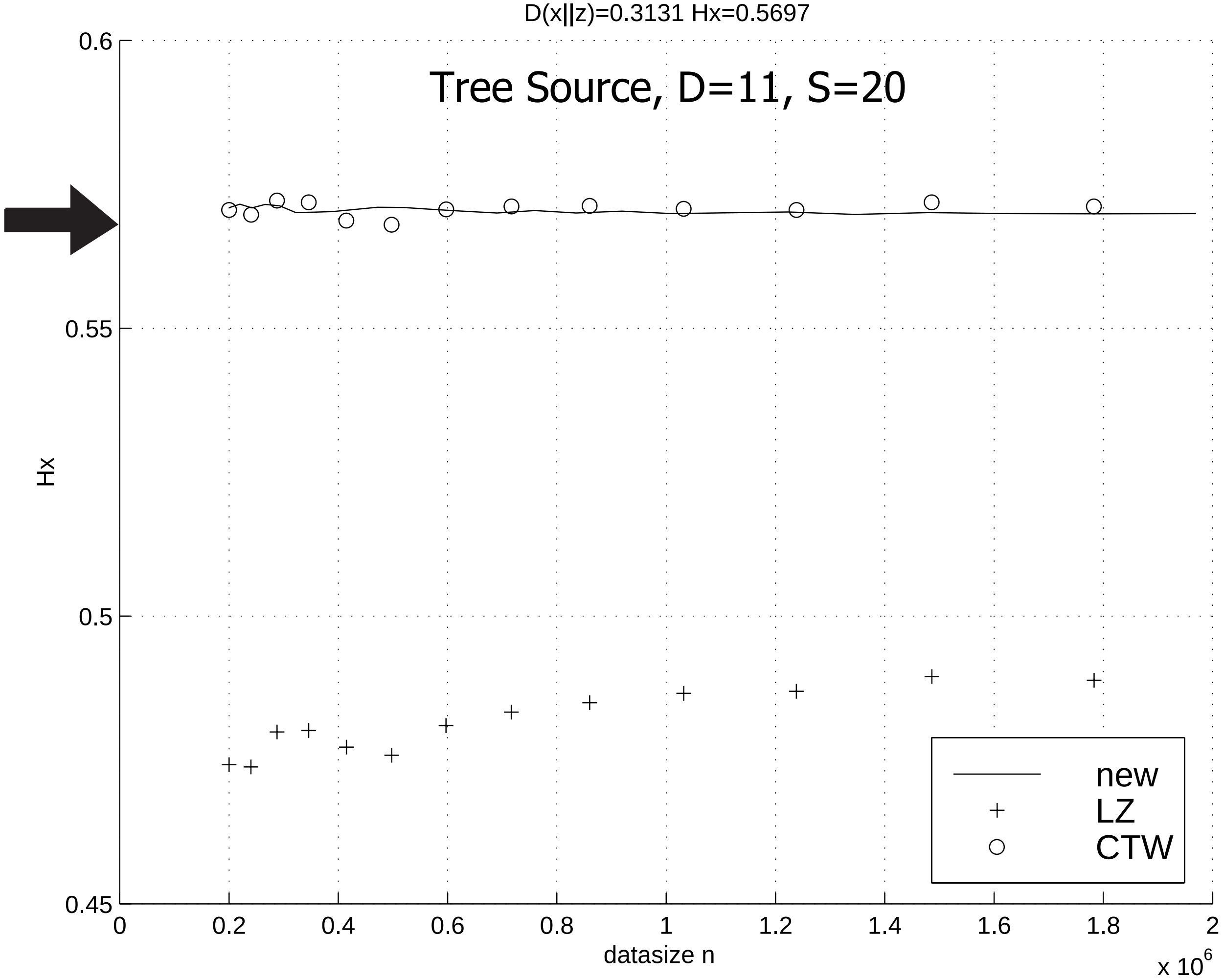


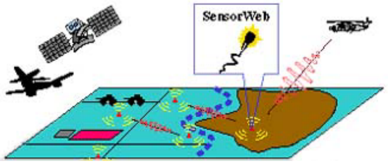
Experimental Results

- Have tested algorithm on simulated date (randomly generated binary tree sources).
- Compared new algorithm with LZ-based methods for both entropy and divergence estimation.
- Compared new algorithm with empirical distribution plug-in scheme for both entropy and divergence.
- Work with text files in progress, and sensor data forthcoming.

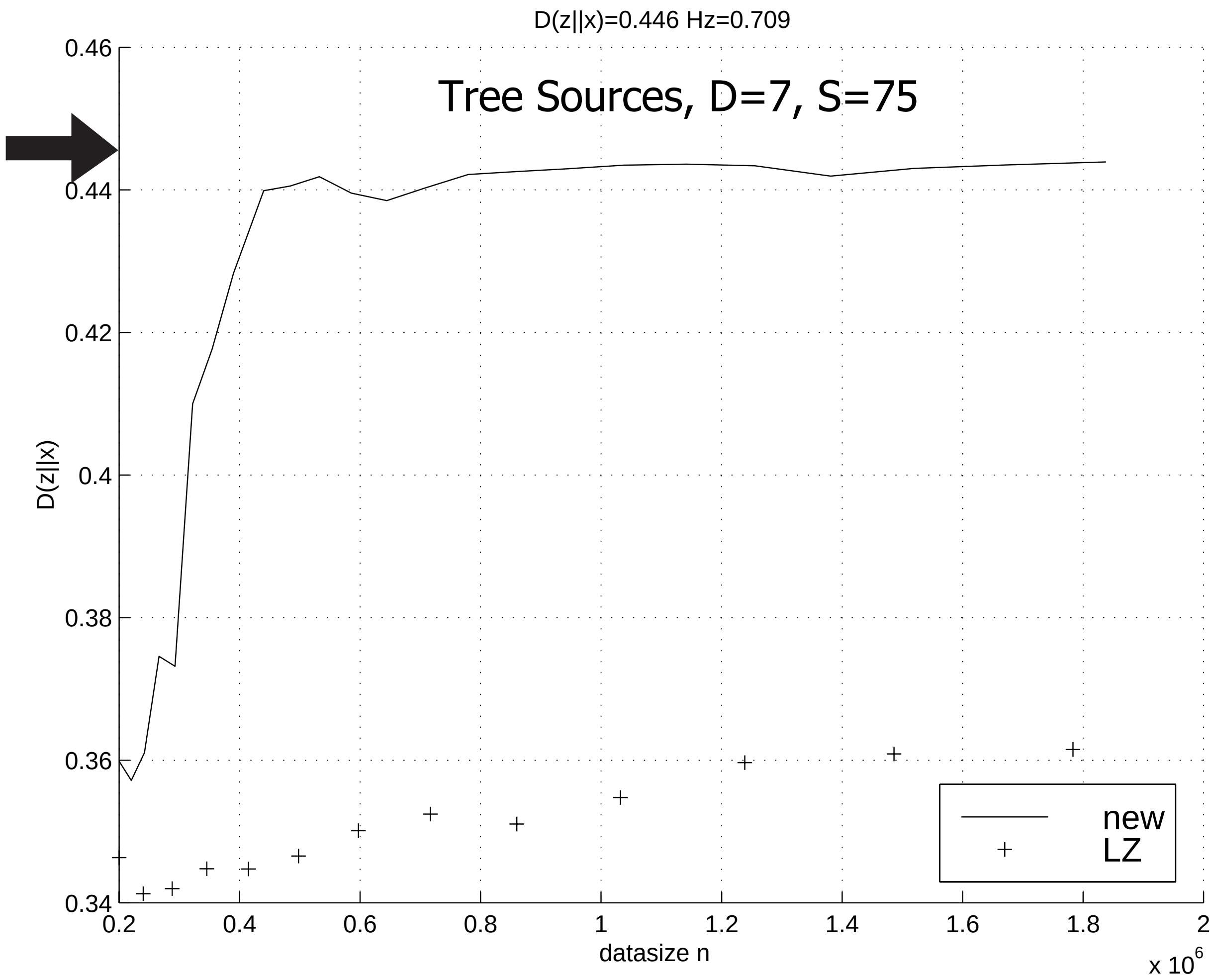
New algorithm vs. LZ – Entropy Estimator

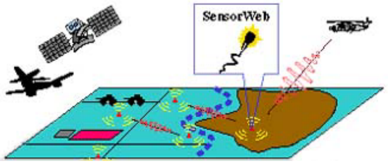




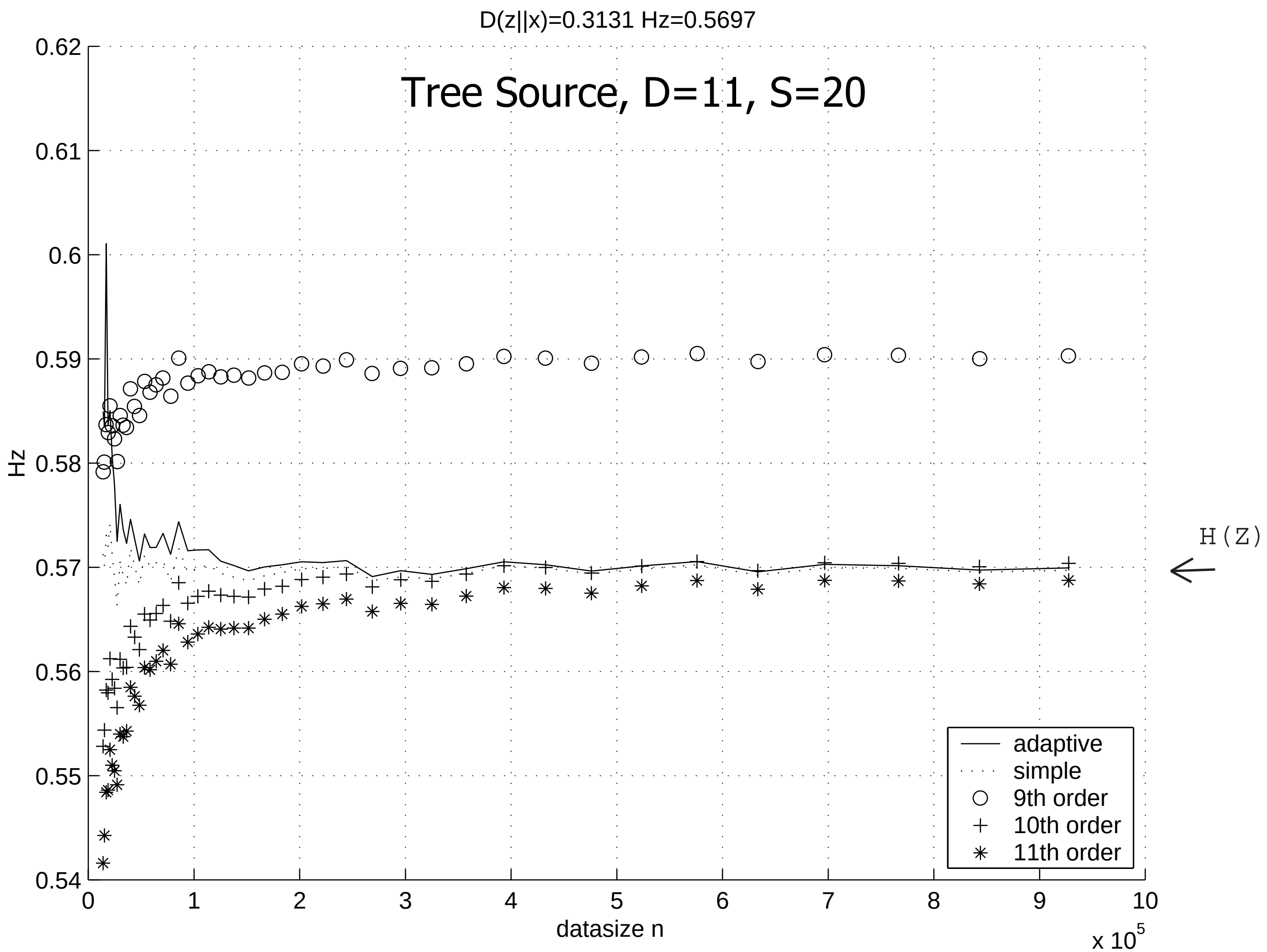


New algorithm vs. LZ – Divergence Estimator

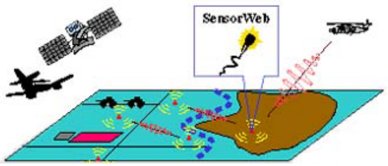


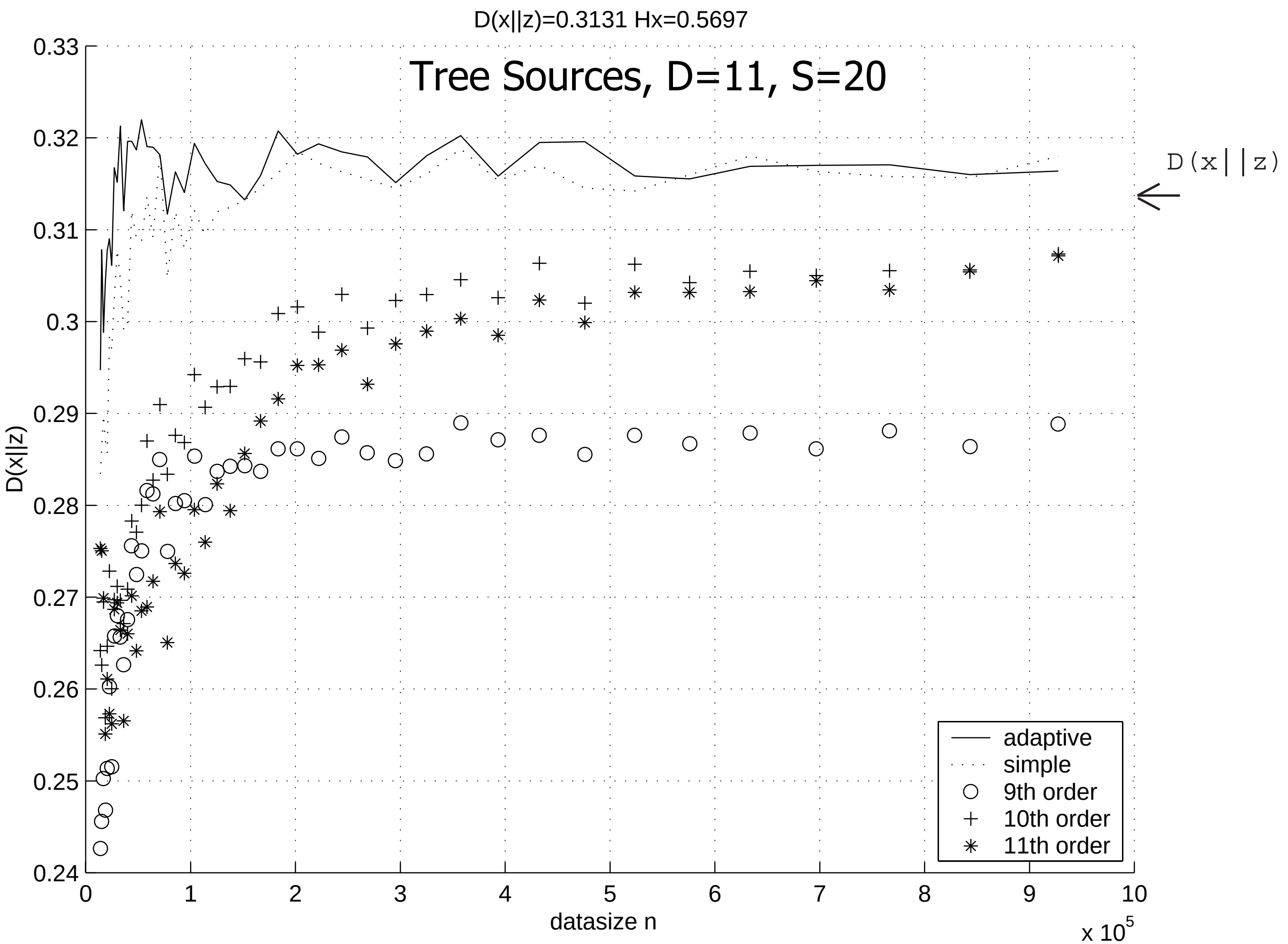


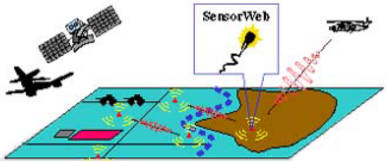
New algorithm vs. Plug-in – Entropy Estimator



New algorithm vs. Plug-in – Divergence Estimator

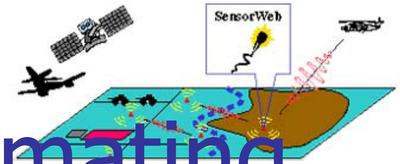






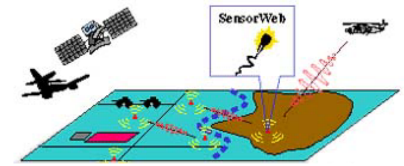
Experimental Results

- New algorithm converges much faster than LZ-based algorithm.
- Choosing "right order" is critical for the empirical distribution plug-in scheme.
- New algorithm has an intrinsic advantage by not assuming $|S| = |\chi|^D$. Also, doesn't assume prior knowledge about memory length or number of states.



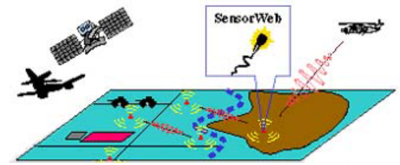
Preliminary Thoughts on Estimating Mutual Information

- Mutual information tells how much one information one source provides about another.
- Can estimate mutual information via entropy or divergence:
  - I(X;Y) = H(X) - H(X|Y) = H(X) + H(Y) – H(X,Y).
  - I(X;Y) = D(p(x,y) || p(x)*p(y)).
- Alphabet size increases complexity.



Summary

- A new entropy and divergence estimator based on BWT (block sorting).
- Doesn’t require knowledge of distribution or parameters of the sources.
- Efficient algorithm, good estimates, fast convergence.
- Significantly outperforms other algorithms tested.
- Expect this to be useful in a wide range of applications --- specifically, a key component in general correspondence and fusion algorithms.



Future Work

- Fine-tune and improve algorithm (e.g., segmentation procedure).
- Further analysis of performance.
- Extensions (e.g., continuous source, non-stationarity).
- Assess performance on actual sensor data.
- Implement algorithms for mutual information.
- Integrate as a component of general correspondence and fusion algorithms.