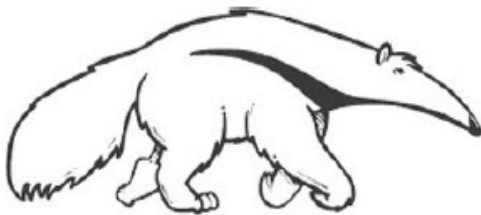


# Pushing the Power of Stochastic Greedy Ordering Schemes for Inference in Graphical Models

Kalev Kask, Andrew Gelfand, Lars Otten, Rina Dechter  
Dept. of Computer Science, UC Irvine

AAAI 2011 - San Francisco, CA  
Tuesday, August 9<sup>th</sup> 2011



# Graphical Model Inference

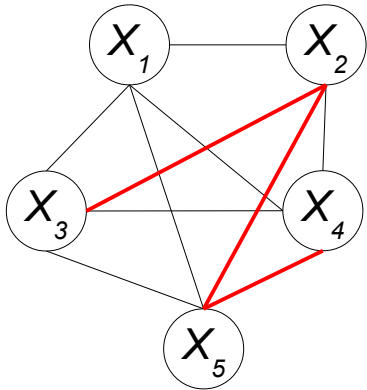
- Underlying graph structure encodes conditional independencies
  - Exploited in many inference algorithms:
    - Junction Tree (JT) [Lauritzen & Spiegelhalter 88]  
Bucket Elimination (BE) [Dechter 99]
    - Generalized BP [Yedidia, Freeman, & Weiss 05]  
AND/OR Sampling [Gogate & Dechter 08]
  - Complexity highly dependent on a given variable ordering and its *(tree)width*.
    - $O(k^w)$  –  $k$  domain size,  $w$  treewidth

# Problem Decomposition

- Captured by elimination/variable ordering
  - Eliminate variable and connect neighbors, repeat

$\pi_A = x_1, x_2, x_3, x_4, x_5$

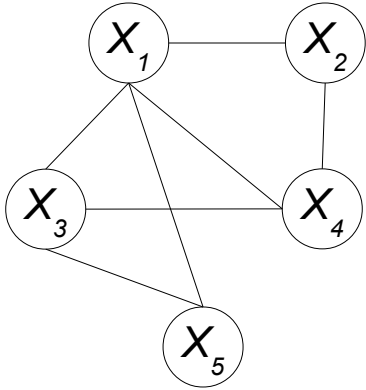
$C_1 = \{x_1, x_2, x_3, x_4, x_5\}$   
 $C_2 = \{x_2, x_3, x_4, x_5\}$   
 $C_3 = \{x_3, x_4, x_5\}$   
 $C_4 = \{x_4, x_5\}$   
 $C_5 = \{x_5\}$



$width(\pi_A) = \max_i |C_i| - 1 = 4$

$\pi_B = x_2, x_5, x_3, x_4, x_1$

$C_1 = \{x_1, x_2, x_4\}$   
 $C_2 = \{x_1, x_3, x_5\}$   
 $C_3 = \{x_3, x_1, x_4\}$   
 $C_4 = \{x_1, x_4\}$   
 $C_5 = \{x_1\}$



$width(\pi_B) = \max_i |C_i| - 1 = 2$

# Computing “good” orderings

- Finding minimal order is NP-hard [Arnborg et al. 87]
  - Many anytime and approximate algorithms
    - B&B: [Gogate & Dechter 04] [Bachooore & Bodlaender 06]
    - Tabu Search: [Clautiaux et al. 04]
    - Simulated Annealing: [Kjaerulff 92]
- Greedy schemes are effective and popular
  - Not yet pushed to their limits
  - Preview:  $n=15,319$ , domain size  $k=5$ 
    - $w = 36 / \underline{19 \text{ TB}} \rightarrow w = 30 / \underline{41 \text{ GB}}$

# Key Contributions

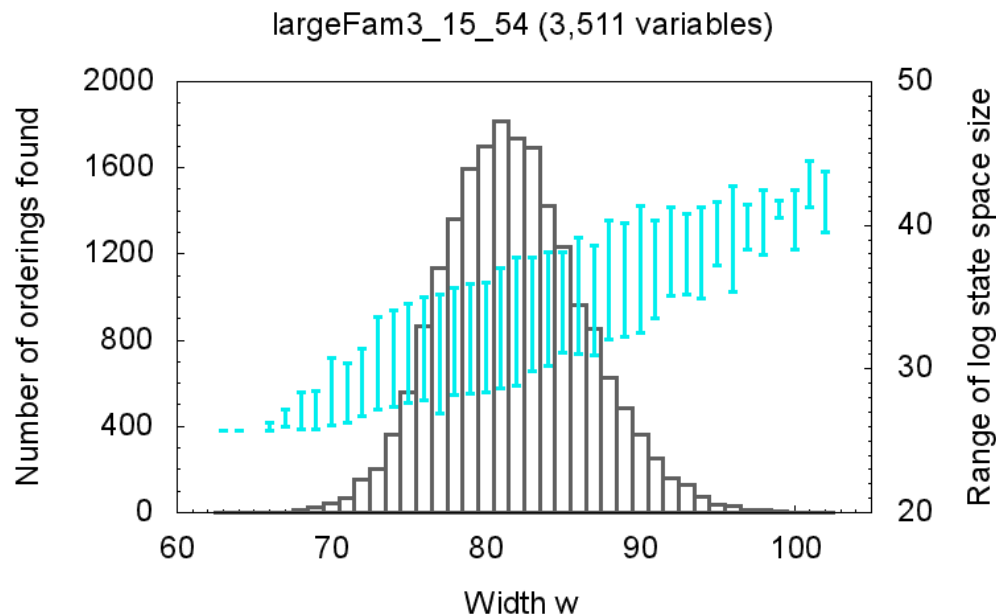
- Present comprehensive overview
- Develop unifying algorithm *IGVO*
  - Algorithmic enhancements:
    - Randomization through pooling
    - Early termination
    - Optimized data structures
    - Parallelization
- Perform extensive empirical evaluation
  - Obtain significant improvements

# Greedy Variable Ordering

- Algorithm: GVO
  - For  $i=1$  to number of variables
    - $\pi(i) \leftarrow$  variable with smallest elimination cost
    - Eliminate  $\pi(i)$
- Cost functions to consider:
  - *Min-Fill*: number of fill edges added
  - *Min-Degree*: degree of node in current graph
  - *Min-Complexity*: cost of variable elimination

# Empirical Observation

- “Smallest cost” leads to many ties
  - Large variance in quality of resulting orders
- 20K Min-Fill iterations, random tie breaking:



# Iterative GVO (IGVO)

- Break ties randomly and repeat! [Fishelson & Geiger 03]
- Algorithm: Iterative GVO (IGVO)
  - For  $n=1$  to number of iterations
    - $\pi_n \leftarrow \text{GVO}(G)$  with random tie breaking
    - If  $C(\pi_n, G) < C(\pi^*, G)$ , then  $\pi^* \leftarrow \pi_n$
- Possible complexity objectives:
  - Width:  $C(\pi, G) \equiv \text{width}(\pi, G)$
  - State space:  $C(\pi, G) \equiv s(\pi, G) = \sum_i s(\pi(i), G_i)$



# Pooling & Early Termination

- **Pooling** with parameters  $p$  and  $e$  :
  - Select node  $\pi(i)$  from pool  $T$  of size  $p$ 
    - Can include nodes with non-minimal cost
  - Non-uniform sampling distribution over  $T$ :
    - Sample node  $v$  with probability  $p(v) = VC(v)^e / \sum_{t \in T} VC(t)^e$
  - Similar in [Fishelson & Geiger 03]
- **Early Termination:**
  - Abort iteration if cost of new ordering exceeds current optimum.

# Optimized Data Structures

1) Adding fill edges has complexity  $O(deg^3)$

- Sorting adjacency lists reduces this to  $O(2 \cdot deg^2)$

2) Updating Min-Fill costs when eliminating  $x$

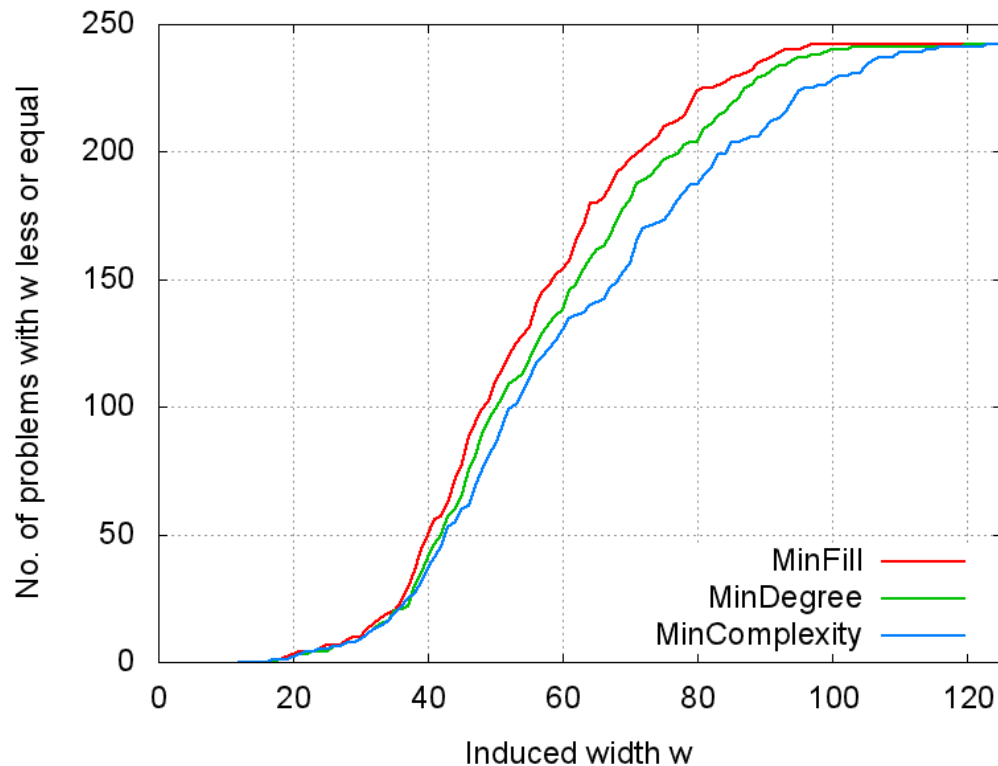
- Full reevaluation of  $N[x]$  and  $N[N[x]]$  expensive
- Instead, start from previous Min-Fill costs:
  - If  $(w, u) \in E, (u, x) \in E$  and  $(w, x) \notin E$  subtract 1 from  $u$
  - $\forall$  fill-edges,  $(u, v)$  if  $(w, u) \in E$  and  $(w, v) \notin E$  add 1 to  $u$
  - $\forall$  fill-edges  $(u, v)$  if  $(w, u) \in E$  and  $(w, v) \in E$   
not added as fill-edge, subtract 1 from  $w$

# Experiments

- Large set of real-world benchmarks:
  - “largeFam”, 242 problems, haplotype queries
    - 2000-6000 variables, domain size 2-6
  - “type4”, 82 problems, genetic linkage analysis
    - Up to 15,000 variables, domain size 2-5
  - “protein”, 138 problems, side-chain prediction
    - Up to 2000 variables, max. domain size  $k=81$
- Compare against baseline implementation
  - Standard Min-Fill with tie breaking

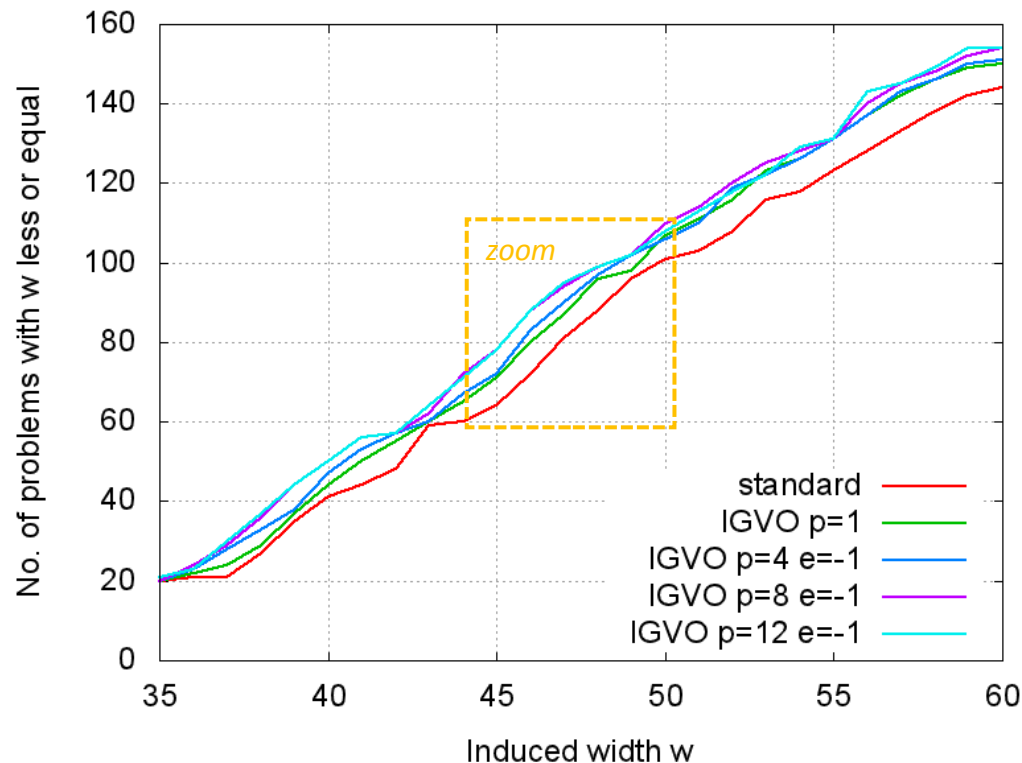
# Comparing Ranking Functions

- Cumulative IGVO results (1 hour, *largeFam*)
  - 242 problems, 2000-6000 variables,  $k = 6$



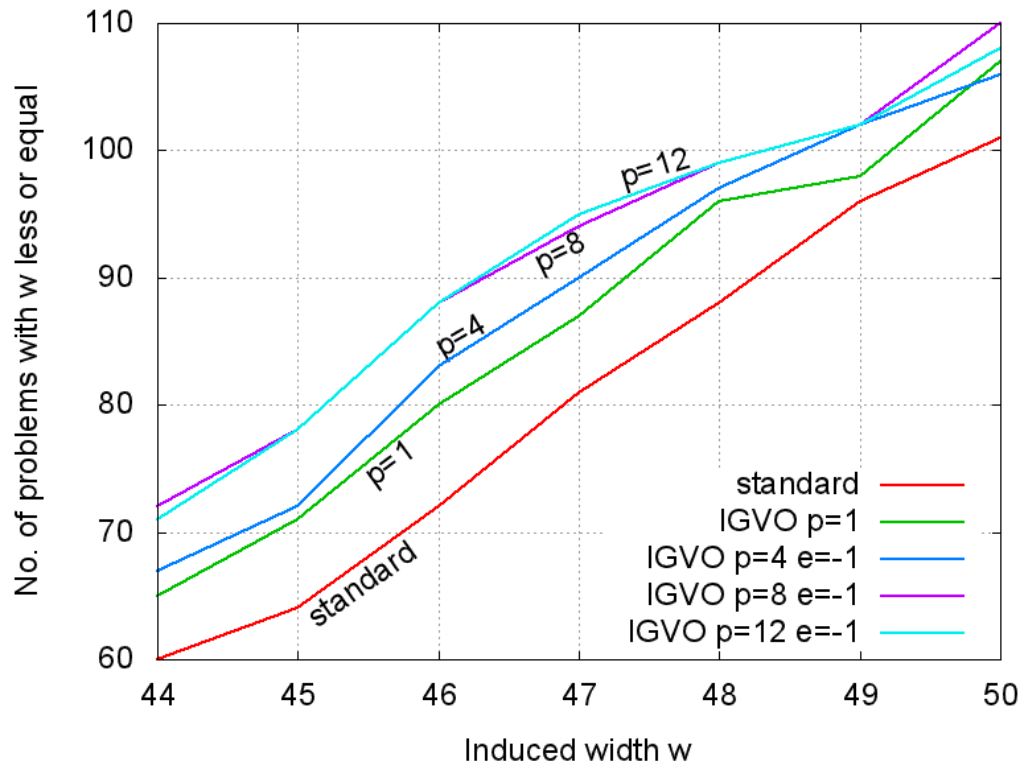
# Effect of Randomization

- Comparing pool sizes (30 minutes, *largeFam*)
  - 242 problems, 2000-6000 variables,  $k = 6$



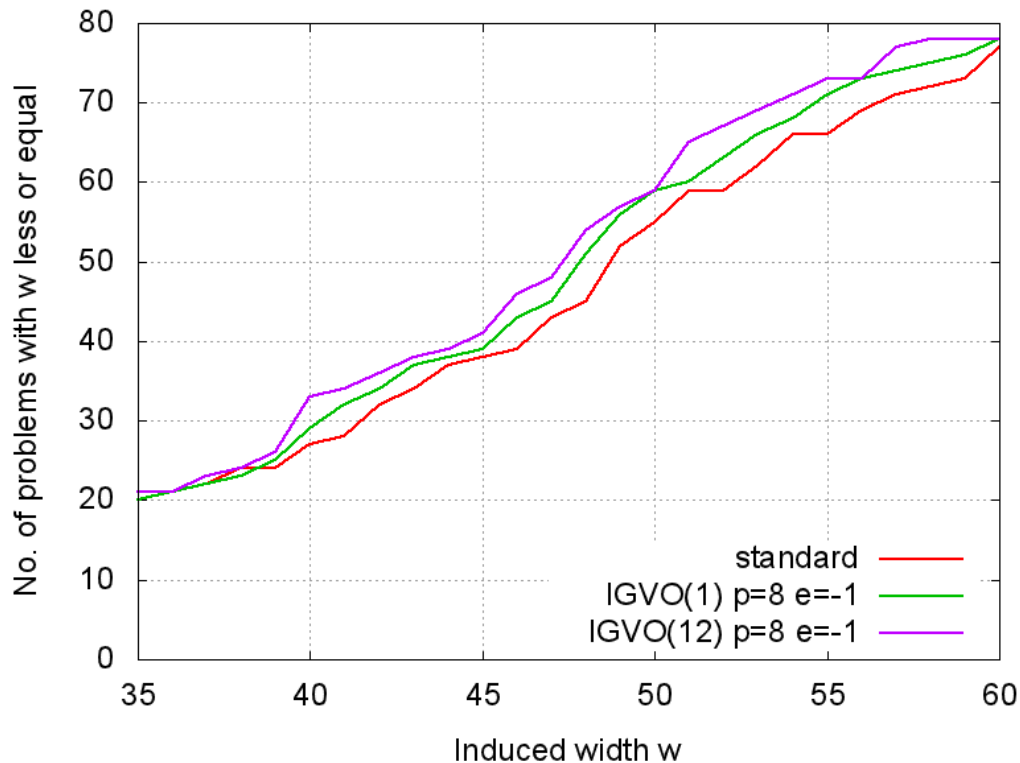
# Effect of Randomization

- Comparing pool sizes (30 minutes, *largeFam*)
  - 242 problems, 2000-6000 variables,  $k = 6$



# Effect of Parallelization

- Single- vs. 12-threaded (30 minutes, *type4*)
  - 82 problems, up to 15,000 variables,  $k = 5$



# Effect of Parallelization

- Select results (30 minutes, *type4*)

instance	$n$	standard		IGVO(1)		IGVO(12)		spd
		iter	$w$	iter	$w$	iter	$w$	
100-18	7,435	6,430	51	26,689	48	324,664	48	12.2
110-19	7,303	3,852	54	13,005	52	158,806	51	12.2
120-18	8,656	6,594	47	17,604	45	211,830	44	12.0
120-25	9,171	3,789	57	14,576	56	176,156	54	12.1
130-20	9,328	3,167	60	12,541	58	154,647	57	12.3
130-22	10,271	3,747	56	13,107	52	168,635	52	12.9
140-23	10,998	2,318	61	7,654	60	91,576	57	12.0
150-22	11,799	2,636	57	8,423	54	99,949	53	11.9
170-18	12,186	2,202	59	6,913	55	82,756	55	12.0
170-22	14,641	2,795	58	8,147	56	97,423	54	12.0
190-19	15,433	3,044	56	6,473	54	77,287	52	11.9
190-21	15,125	5,284	43	9,545	42	115,048	40	12.1



# Pushing Feasibility

- *BEEM*: Bucket Elimination with External Memory [Kask, Gelfand, & Dechter 10]
  - Utilizes hard drive storage to store tables
  - Four previously infeasible instances now solvable

instance	$n$	$k$	previous		new	
			$w$	space	$w$	space
110-21	7,675	5	37	16 TB	33	215 GB
140-20	9,355	5	35	10 TB	28	4 GB
180-21	14,157	5	38	9 TB	31	67 GB
200-18	15,319	5	36	19 TB	30	41 GB

# Summary

- Iterative Greedy Variable Ordering (IGVO):
  - Unifying framework for finding orderings
  - Flexible yet simple and easily parallelizable
  - Implementation engineered for efficiency, algorithmic optimizations
- Often yields significantly better orderings
  - Allowed solving previously infeasible instances