FPGA Acceleration of Gene Rearrangement Analysis

Jason D. Bakos

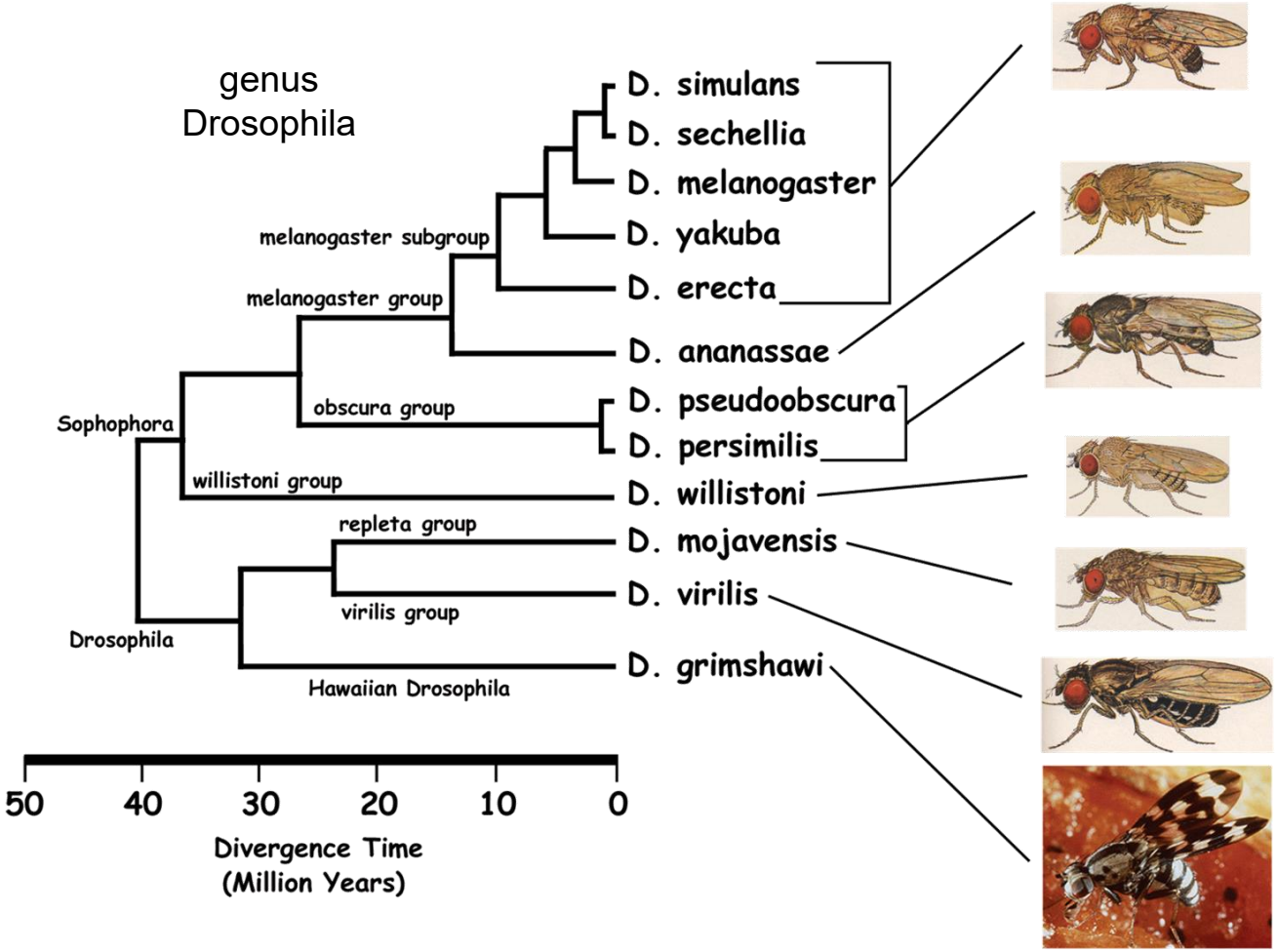
Dept. of Computer Science and Engineering
University of South Carolina
Columbia, SC USA

UNIVERSITY OF SOUTH CAROLINA

Talk Outline

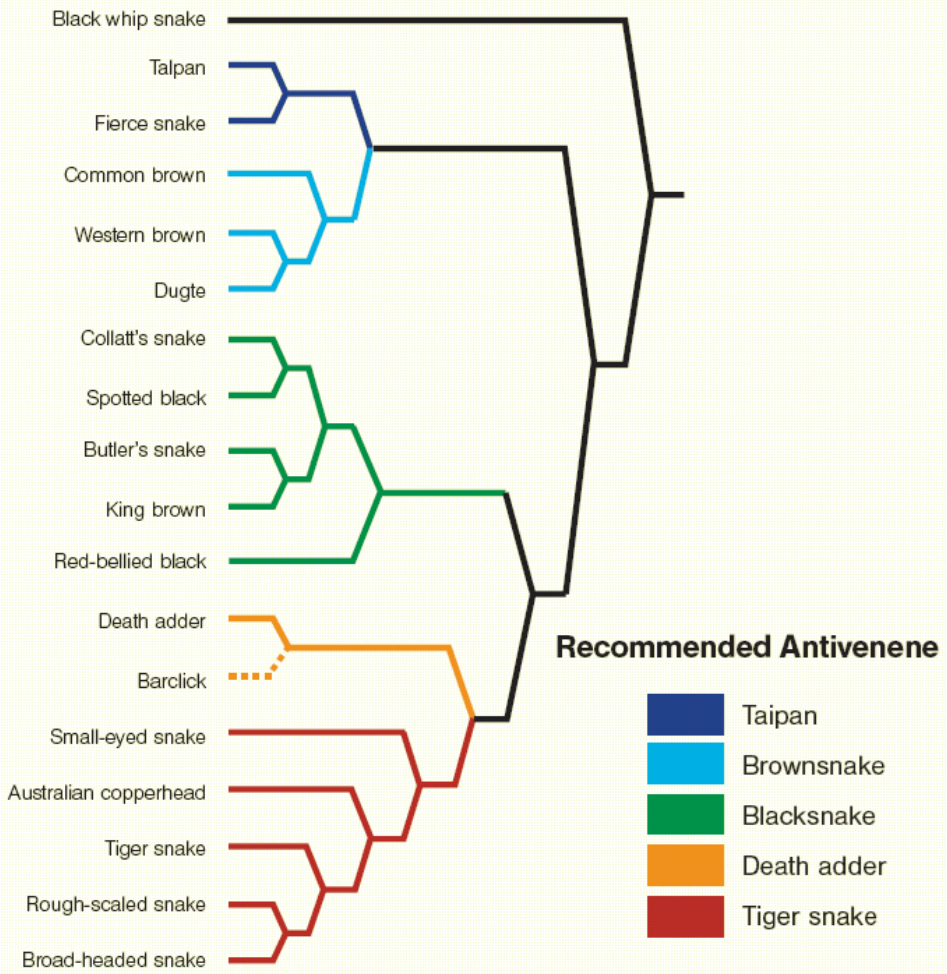
- Application:
  - Phylogenetic Reconstruction
  - Gene Rearrangement Data
- Breakpoint Median Computation
- Core Architecture
- Extracting Parallelism
- Performance Results
  - Core Computation Speedup
  - Application Speedup
- Conclusion and Future Work

Phylogenies

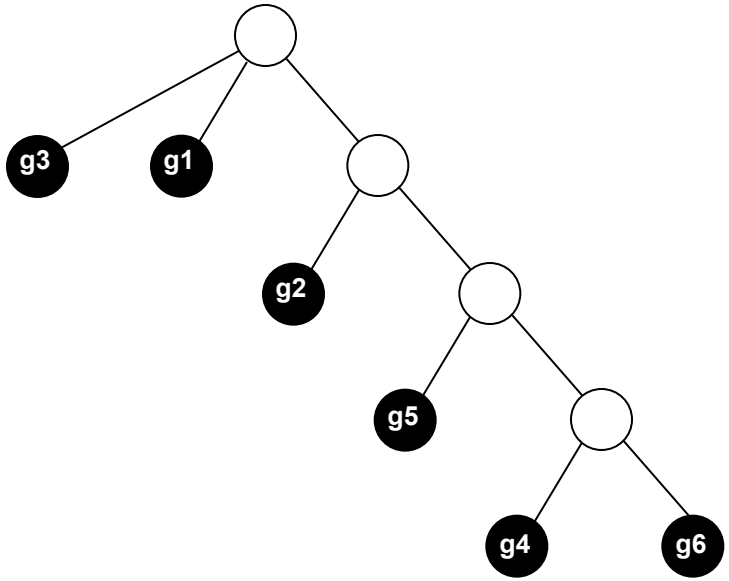


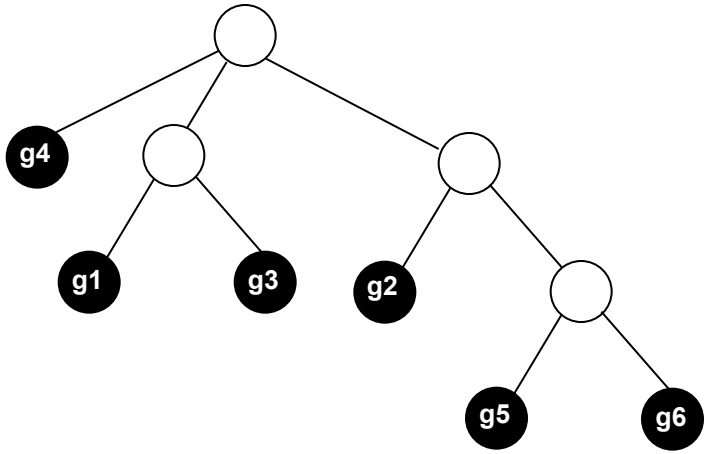
Phylogenetic Analysis

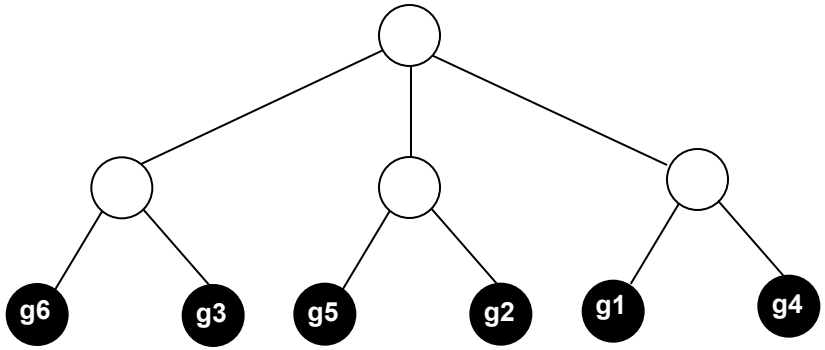
- Phylogenies are used to infer common characteristics among related species



Phylogeny Data Structure







- Unrooted binary tree
- *n* leaf vertices
- *n* - 2 internal vertices (degree 3)

- Tree configurations =
  $(2n - 5) * (2n - 7) * (2n - 9) * \ldots * 3$
- 200 trillion trees for 16 leaves

Phylogenetic Reconstruction

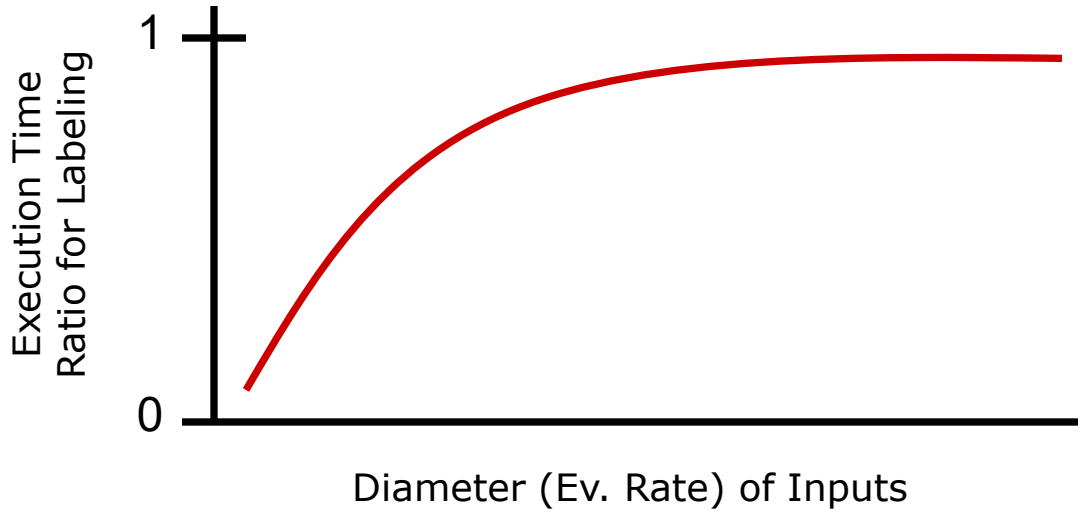
- Given biological data for a set of species, compute an accurate phylogeny

- Distance-based Methods
  - Construct tree using pair-wise distances (UPGMA, neighbor-joining)
  - Fast but inaccurate

- Maximum Parsimony Methods
  - Goal: Accuracy
  - Construct tree using evolutionary model
  - Search for most parsimonious tree
  - Candidate trees must be "*scored*"
    - All internal vertices are labeled with ancestral data
    - Edge distances are computed between vertex pairs and summed

GRAPPA Performance Bottleneck

- GRAPPA
  - "Genome Rearrangement Analysis under Parsimony and other Phylogenetic Algorithms"
  - Exhaustive branch-and-bound or heuristic search through tree space
  - Labeling computation is performance bottleneck
    - 95+% of execution time is spent computing labels for internal vertices



Gene Rearrangement Data

- Genome data is represented as a circular ordering of genes
  - "Gene orders"
- Each gene represents a known nucleotide sequence
  - Each gene has a positive or negative orientation

- Rare evolutionary events cause gene rearrangements
  - *Inversion*

    $g_0\ g_1\ g_2\ g_3\ g_4\ g_5 \longrightarrow g_0\ g_1\ -g_4\ -g_3\ -g_2\ g_5$

  - *Transposition*

    $g_0\ g_1\ g_2\ g_3\ g_4\ g_5 \longrightarrow g_0\ g_2\ g_3\ g_4\ g_1\ g_5$

  - *Inverted Transposition*

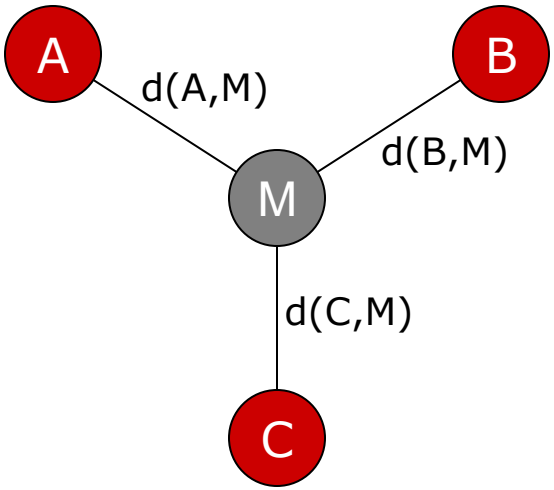
    $g_0\ g_1\ g_2\ g_3\ g_4\ g_5 \longrightarrow g_0\ -g_4\ -g_3\ -g_2\ g_1\ g_5$

Breakpoint Distance Metric

- Estimation of number of rearrangement events between gene orders A and B
- # of adjacencies:

  g h in A that doesn't correspond to *g h* or *–h –g* in B
- Example:
  - A = 1 2 3 4 5
  - B = -2 -1 -5 -4 3
  - Breakpoint distance = 2

Median



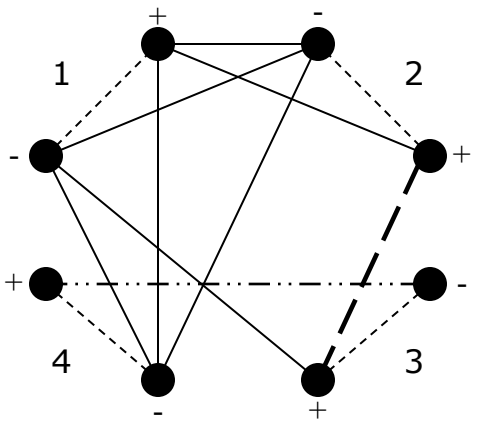
- Ancestral vertices are computed using a *median computation*
- All internal vertices have degree 3
- Find M that optimally minimizes median score

  score = d(A,M) + d(B,M) + d(C,M)
- Breakpoint median:
  - d() is breakpoint distance

Breakpoint Median Algorithm as a TSP

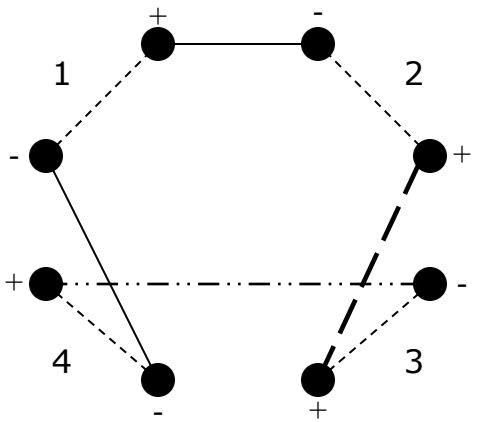
- Construct a fully connected graph containing all *g* and *–g* for each gene
- w(*g*,*-g*) = **-∞**
- Initialize all other weights to be 3
- For each adjacency *gh* in the three genomes, decrement weight between vertex *–g* and *h*
- Solve TSP, median is the even-indices of the tour (score = tour cost)

A = -1 +2 -4 -3
B = -1 -2 +3 +4
C = -2 +3 +4 +1



Edges not shown have cost = 3

cost = -∞
cost = 0
cost = 1
cost = 2



An optimal solution corresponding to genome +1 +2 -3 -4

Breakpoint Median Implementation

- Optimal TSP is feasible due to small graph
- Implemented as a depth-first branch-and-bound search
- Upper bound is the current best tour
- Lower-bound is computed using a linear greedy algorithm
  - Select a set of minimal-weight edges to complete a partially-constructed tour
  - To tighten: edges not considered that…
    - have been pruned at or above the current level of the search tree
    - that would create a cycle not including all cities

Breakpoint Median Algorithm

Edge list:

edges
(-3,4,w=0)
(2,3,w=1)
(1,2,w=2)
(-1,-2,w=2)
(1,-2,w=2)
(-2,-4,w=2)
(-1,3,w=2)
(-1,-4,w=2)
(1,-4,w=2)

Search state:

solution
1 ◂···▸ -1
2 ◂···▸ -2
3 ◂···▸ -3
4 ◂···▸ -4

cost=0

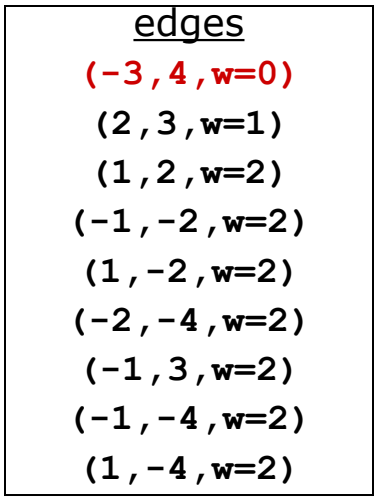
used

excluded

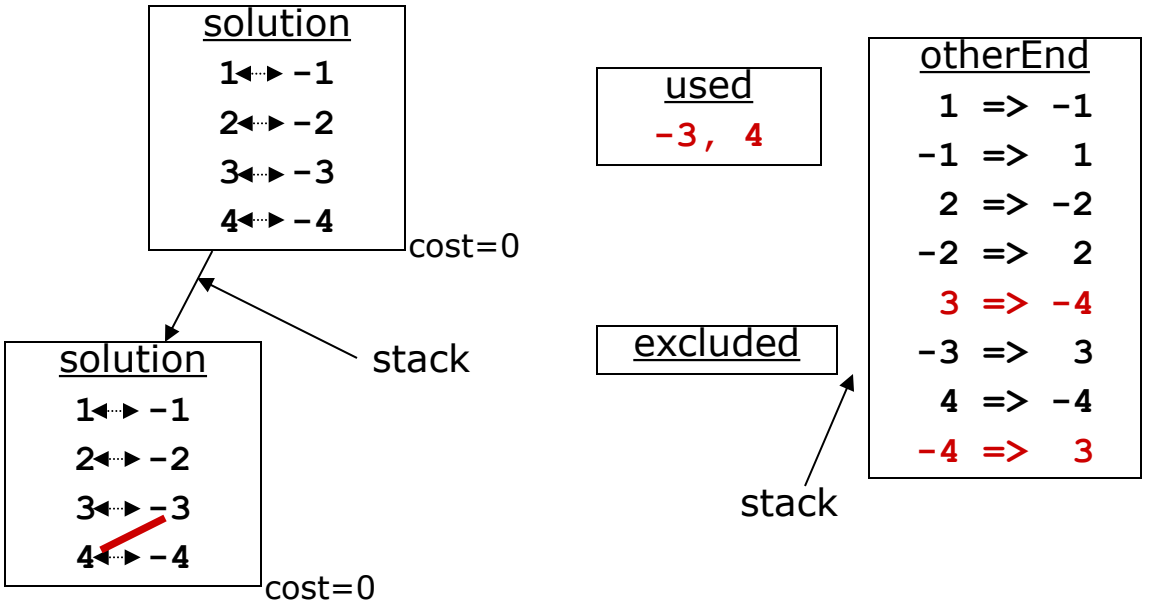
otherEnd
1 => -1
-1 => 1
2 => -2
-2 => 2
3 => -3
-3 => 3
4 => -4
-4 => 4

Breakpoint Median Algorithm

Edge list:

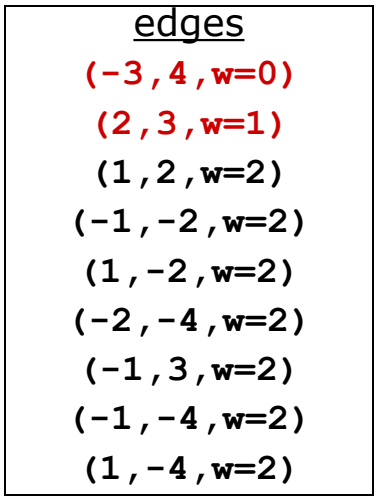


Search state:



Breakpoint Median Algorithm

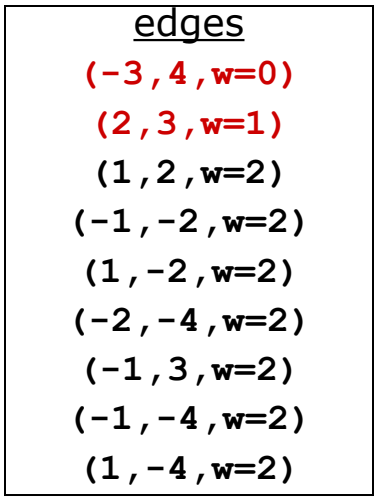
Edge list:



Search state:

Breakpoint Median Algorithm

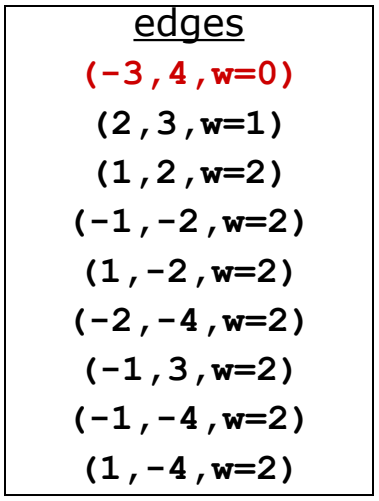
Edge list:



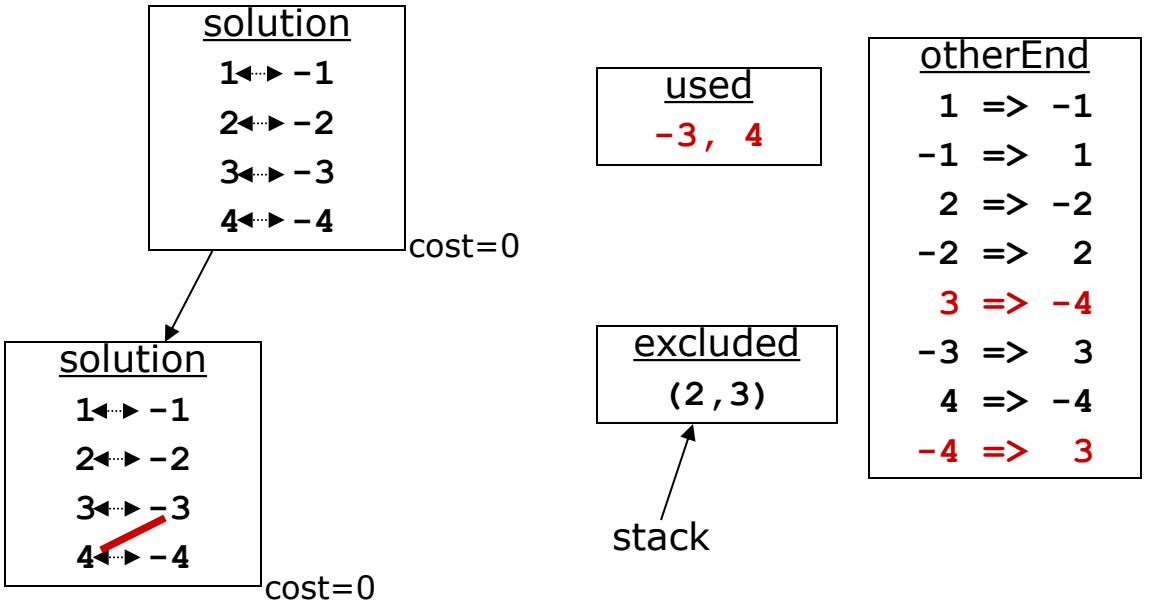
Search state:

Breakpoint Median Algorithm

Edge list:

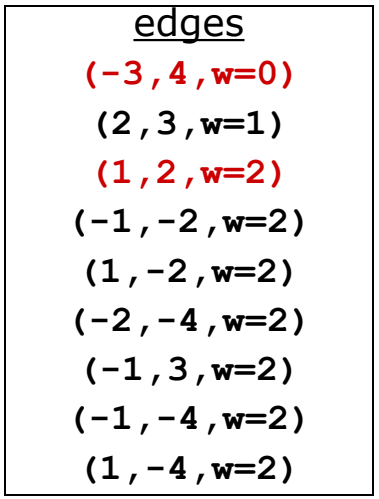


Search state:

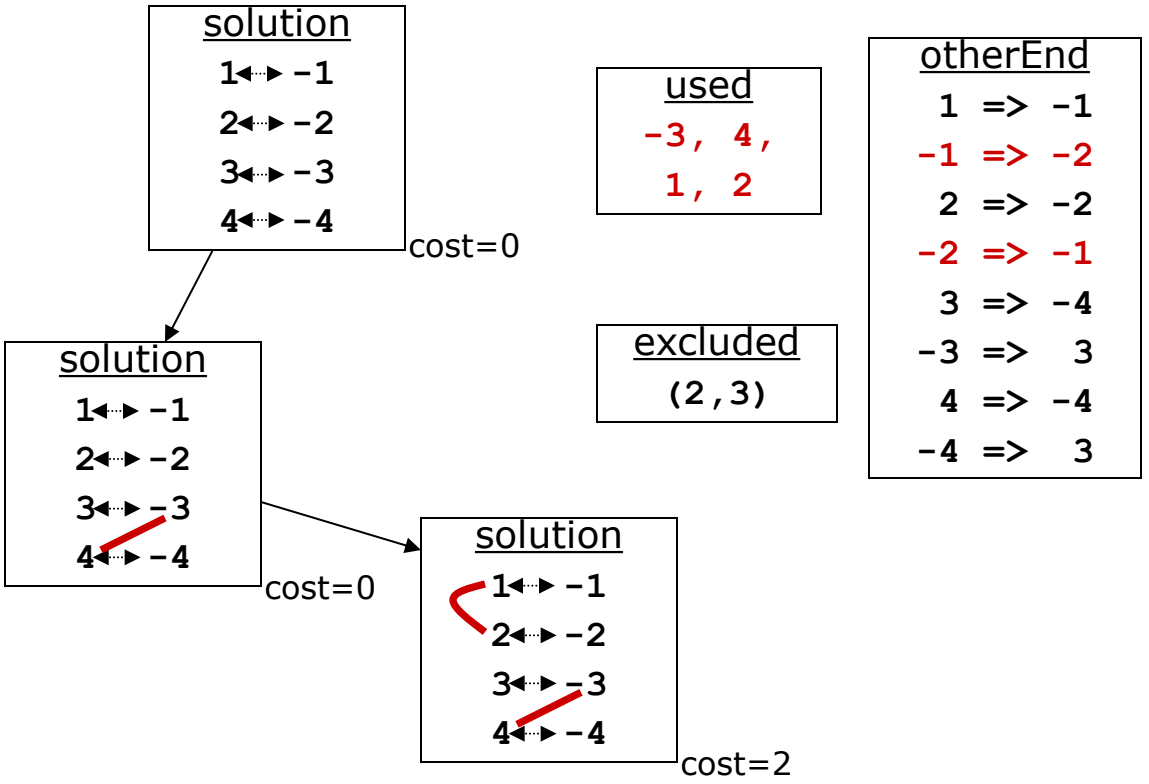


Breakpoint Median Algorithm

Edge list:

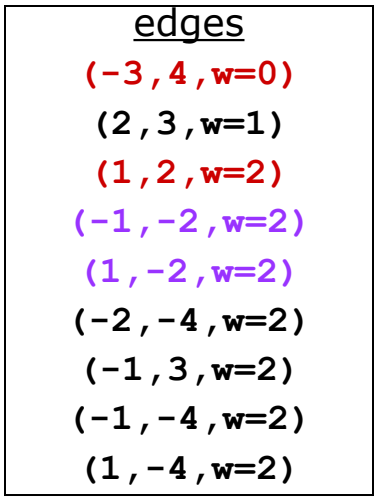


Search state:

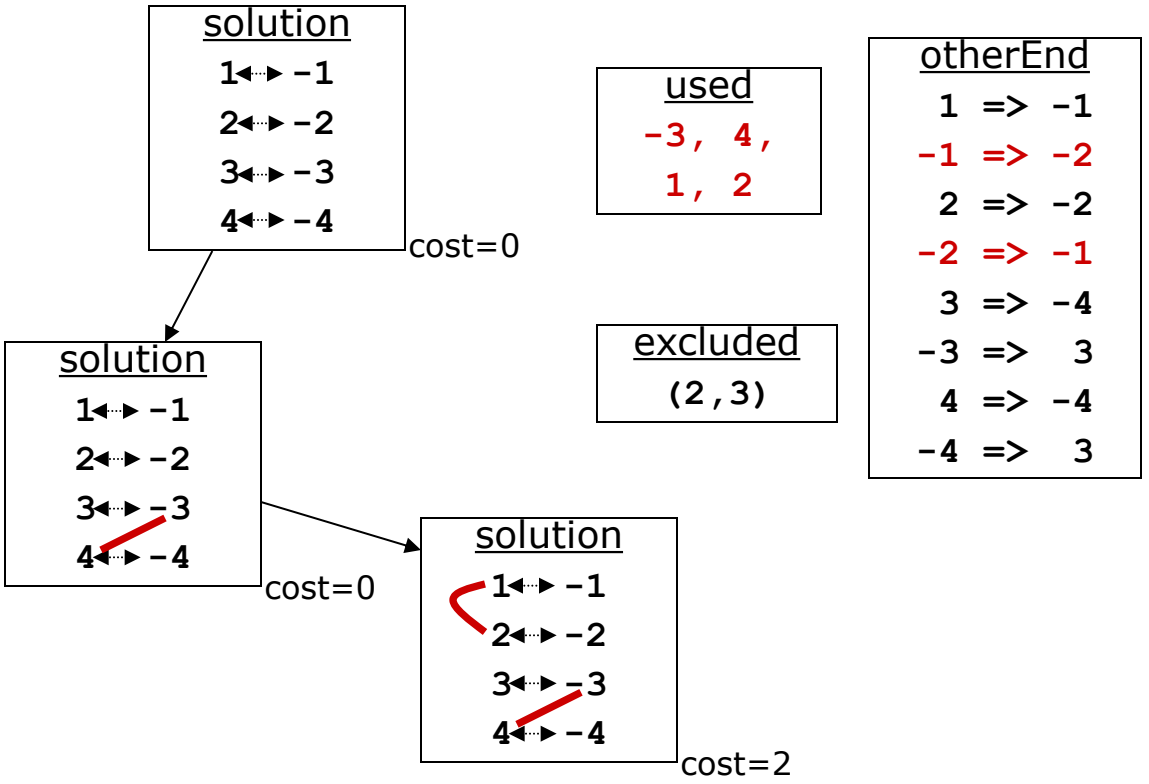


Breakpoint Median Algorithm

Edge list:

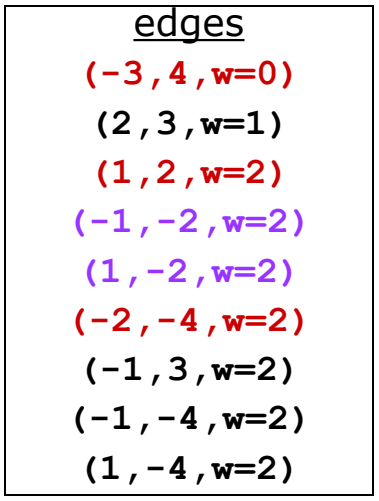


Search state:

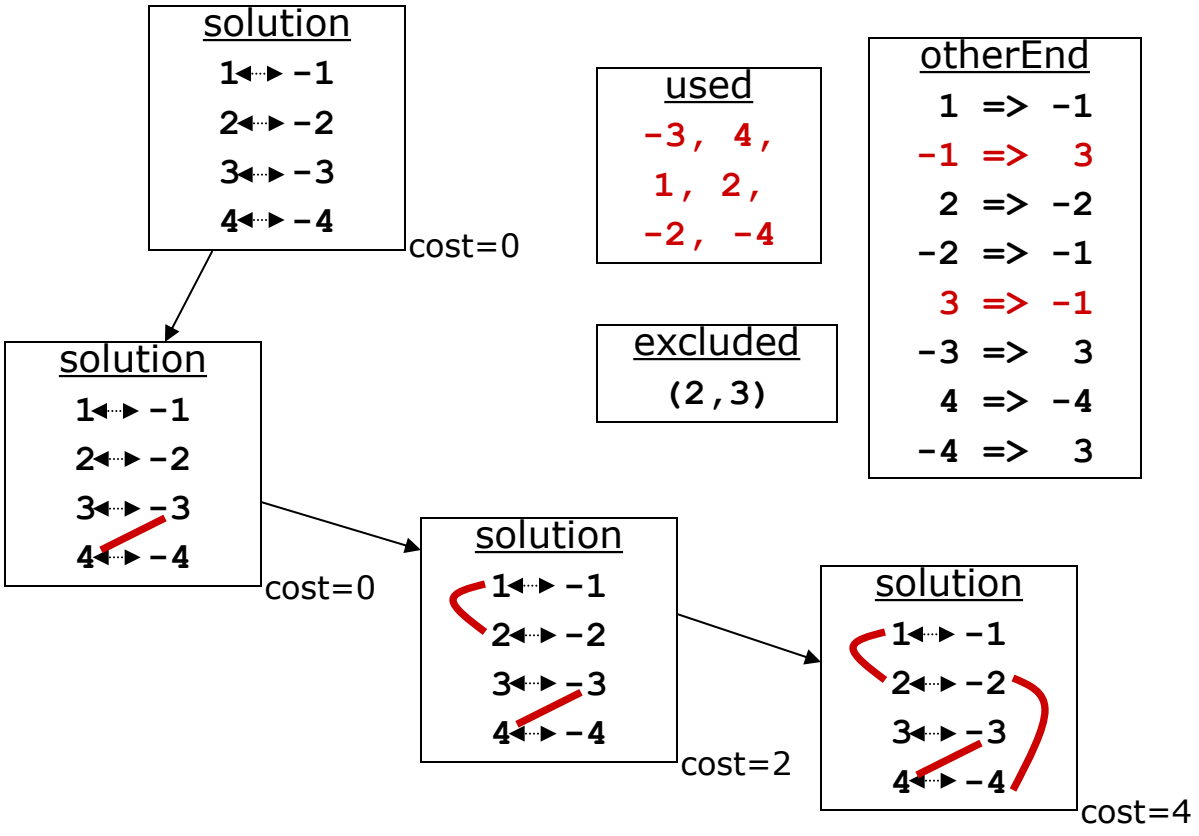


Breakpoint Median Algorithm

Edge list:

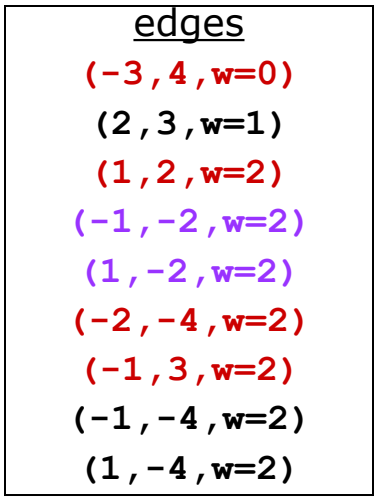


Search state:

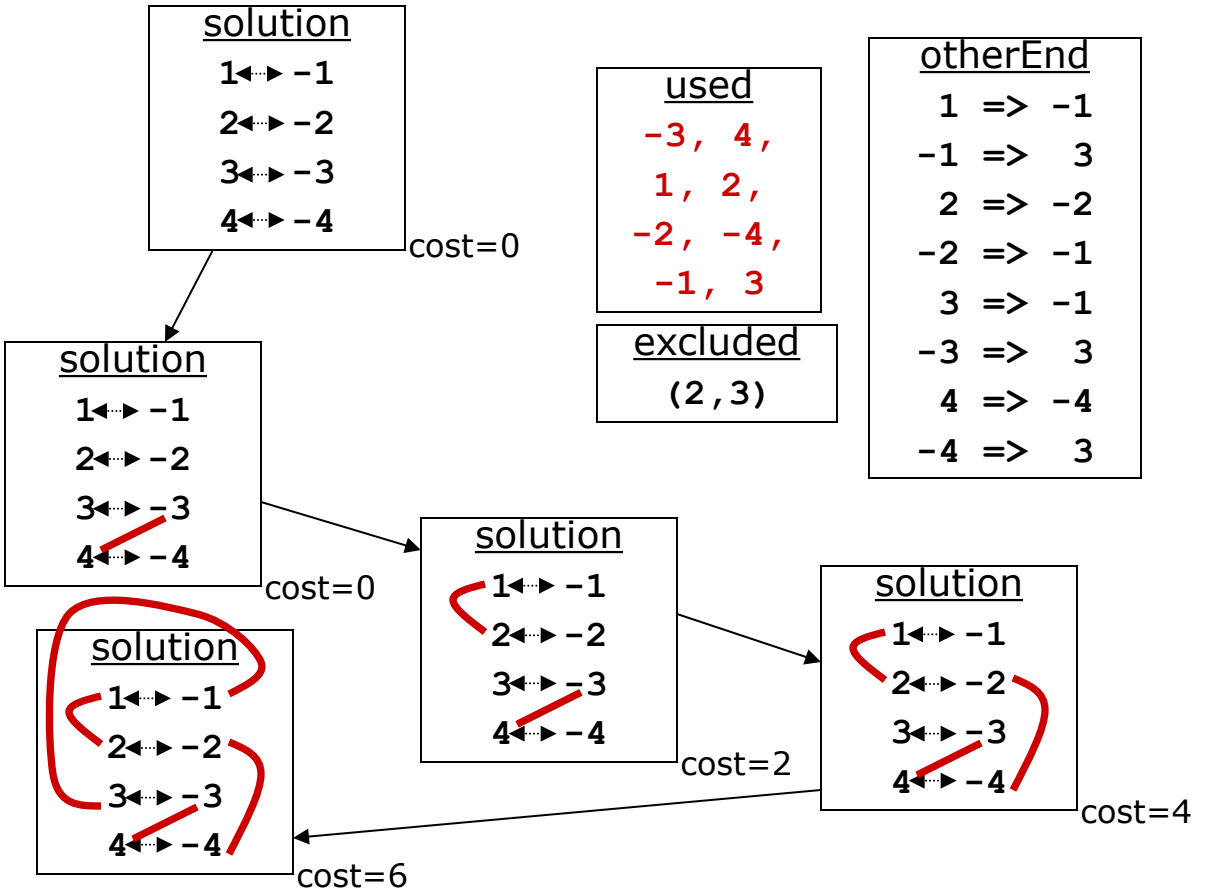


Breakpoint Median Algorithm

Edge list:



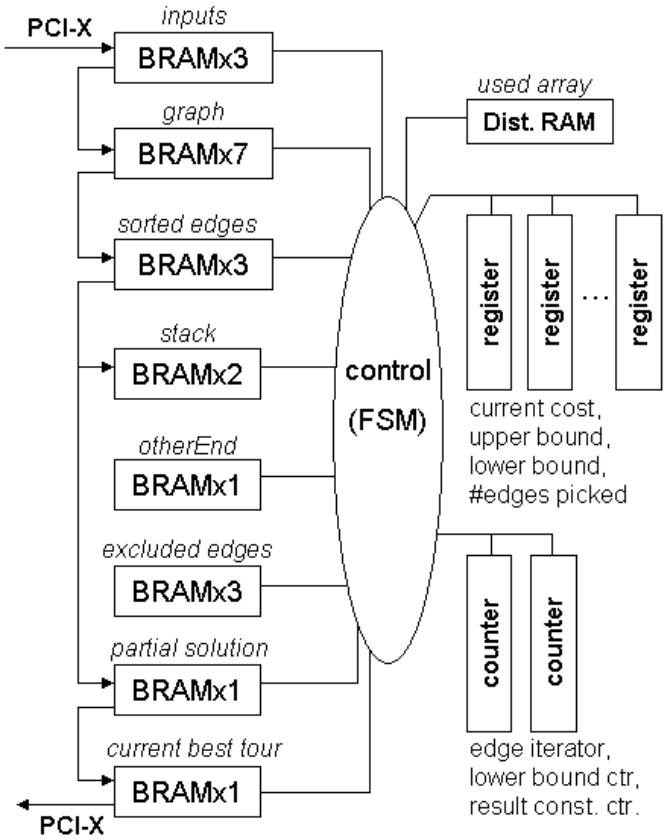
Search state:



tour is -1, 1, 2, -2, -4, 4, -3, 3
median is -1, 2, -4, -3

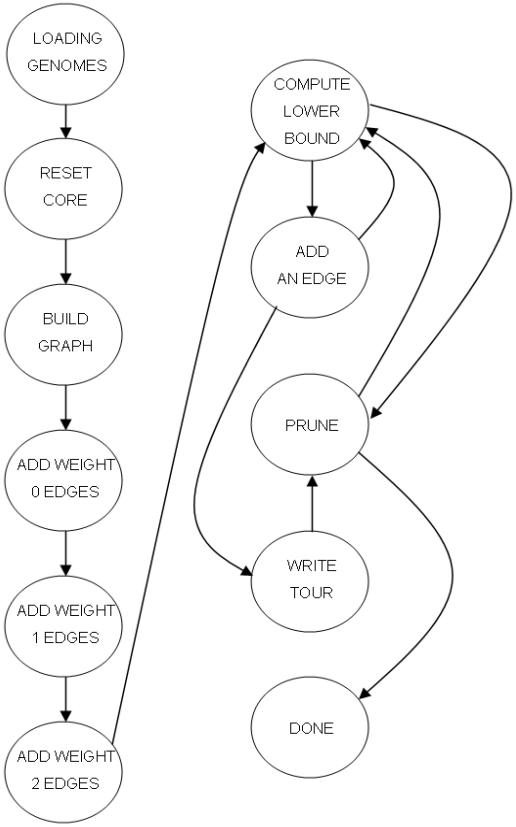
Median Core Architecture

top level



21/444 BRAMs/core

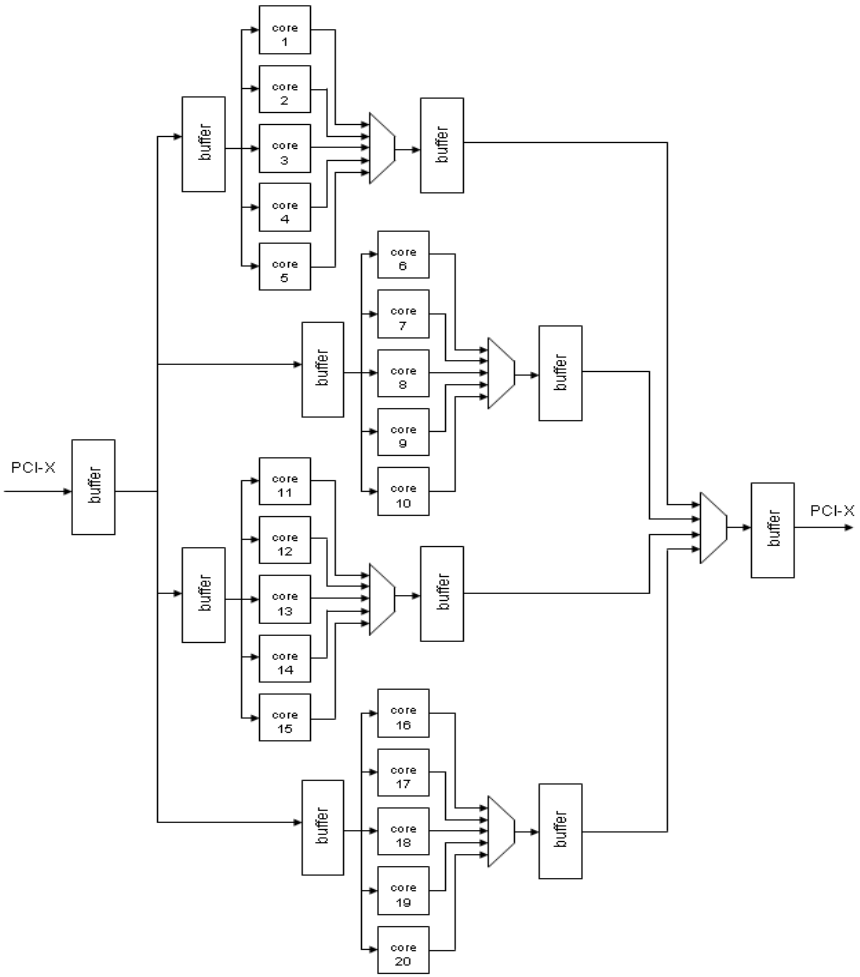
controller



1% slices/core

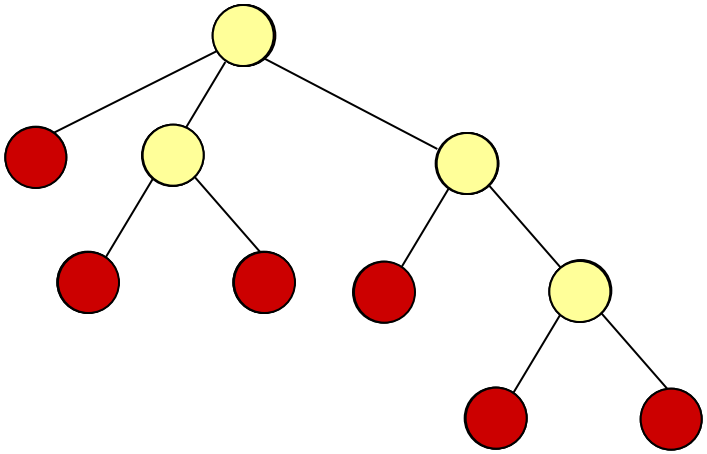
- Computation is sequential
- add edge
  7 cycles
- prune edge
  2 or 10 cycles
- compute lower bound
  2n+2 cycles
- 56 MHz

Accelerator Architecture

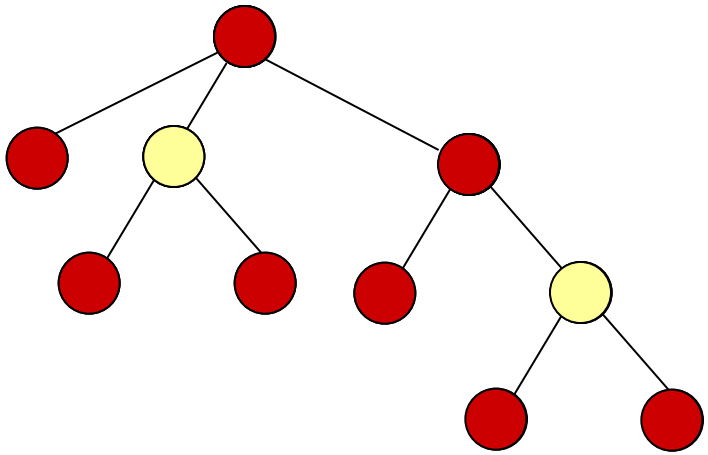


- Fill FPGAs with median cores
- Fan-outs and fan-ins are pipelined to meet PCI-X timing
- Platform:
  - Annapolis Wild-Star II Pro
  - Virtex-2 Pro 100 -5
- I/O
  - Programmed I/O
  - Hosts polls each core for state
  - Comm. overhead is significant for easy medians

Phylogeny Scoring Steps

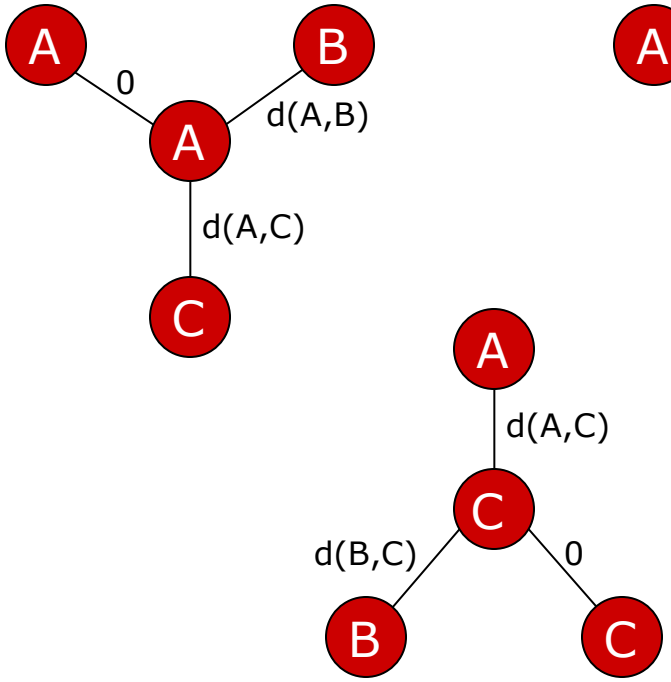


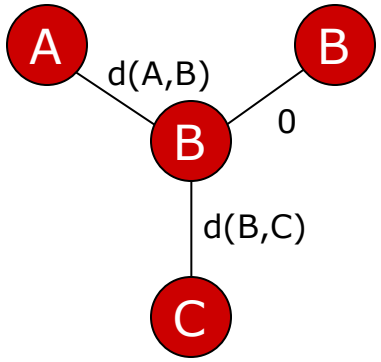
1. Initialize unlabeled tree
   - Use 3 nearest labels
   - Initialize upper bound from inputs



2. Iteratively refine tree to convergence
   - Use 3 immediate neighbors
   - Initialize upper bound using score of previous label

Extracting Parallelism: Initialization



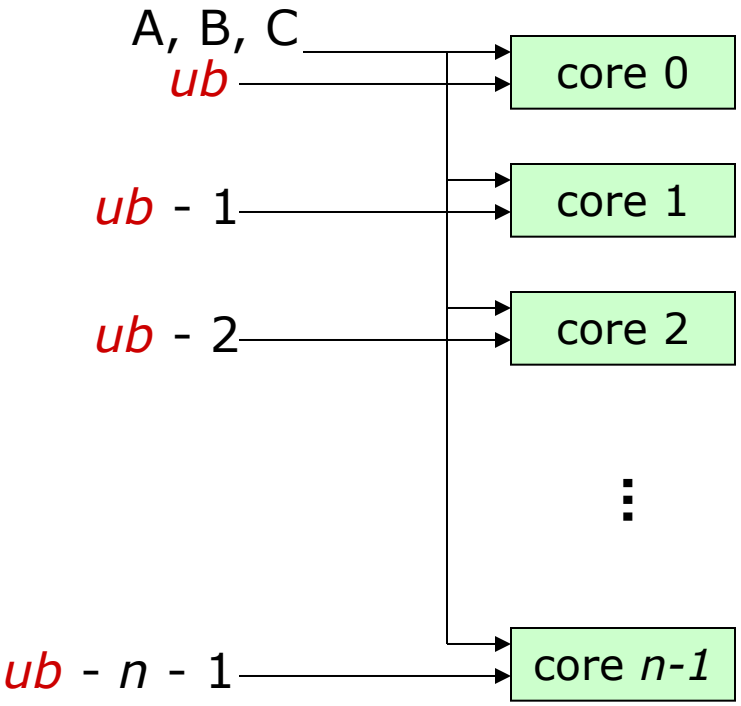


initial upper bound = *ub* =

d(**A**,B) + d(**A**,C)

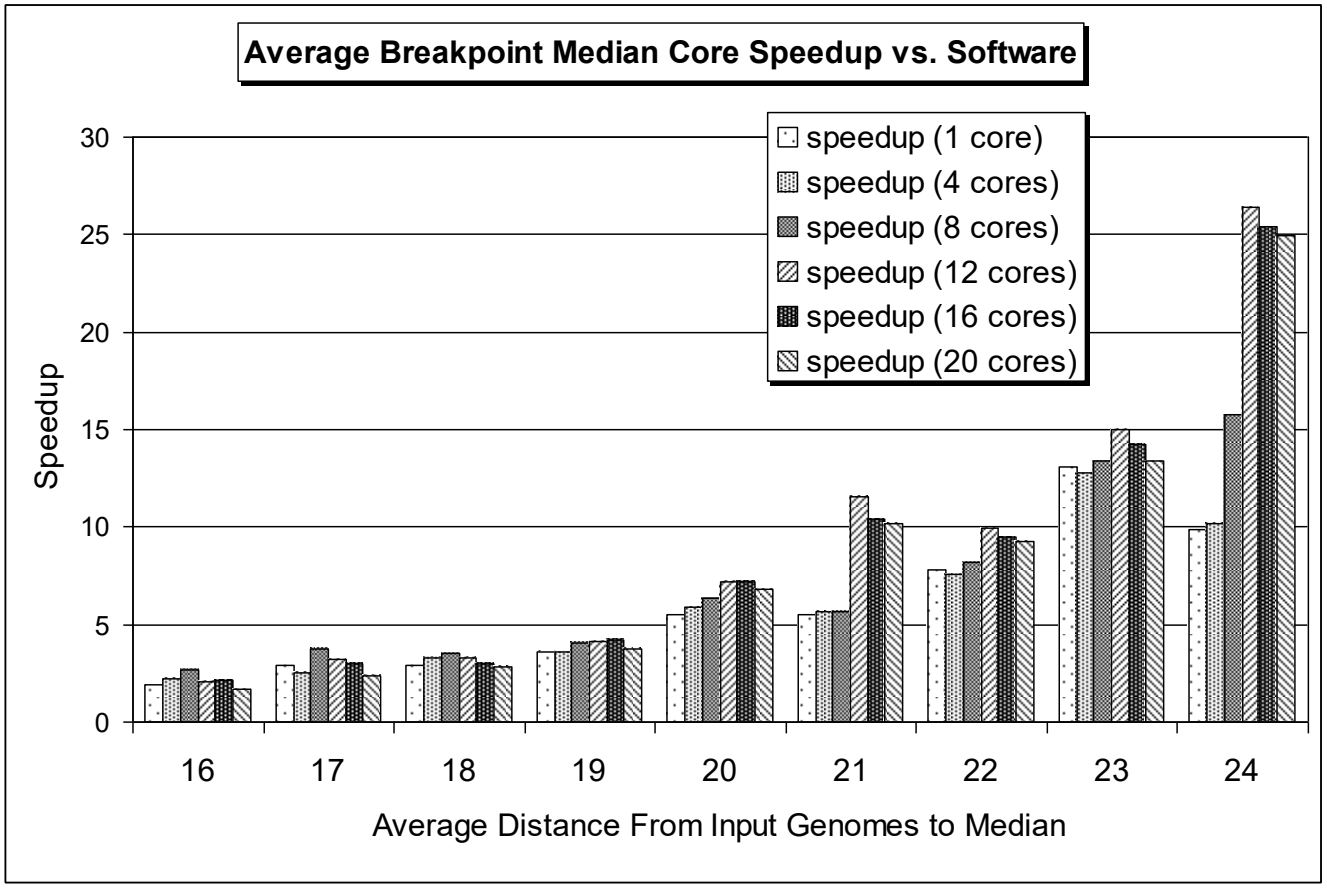
d(**B**,A) + d(**B**,C)

d(**C**,A) + d(**C**,B)



Core with a lower initial upper bound will converge on solution fastest

Performance Results: Median Computation



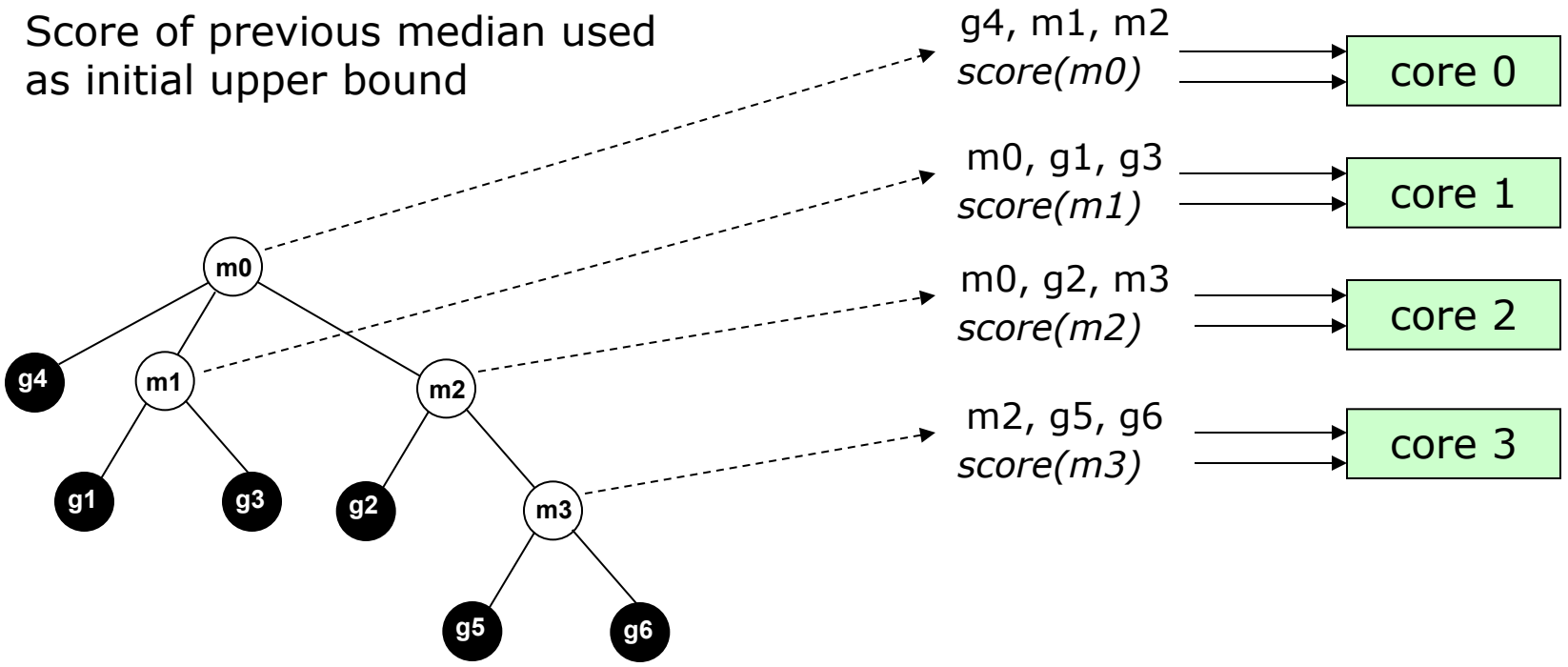
Average over 1000 median computations

12 cores =>

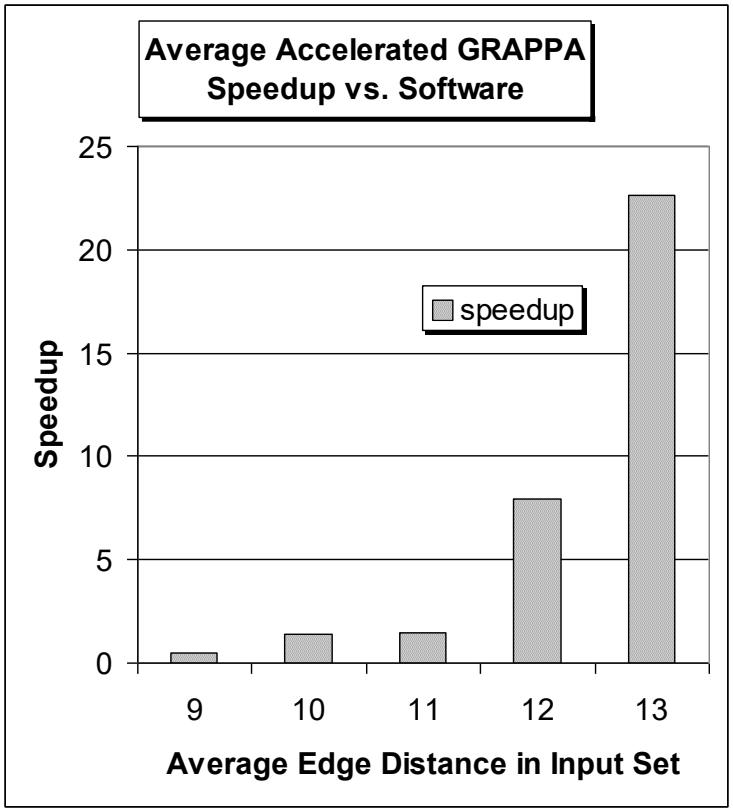
25X speedup

Extracting Parallelism: Re-Labeling

- All medians are dispatched in parallel for each iteration
- Score of previous median used as initial upper bound



Performance Results: Accelerated GRAPPA



- Replace software median with driver for FPGA card
- Initialization phase:
  - Use 12 median cores
- Re-labeling phase:
  - Parallel labeling
  - Use *n* - 2 median cores
- Average over 10 GRAPPA runs

Conclusions

- Breakpoint median computation: combinatorial optimization algorithm
  - Control-dependent, memory-intensive
- Median core consistently achieves speedup over software and only requires 4% of FPGA resources
- Initial results show that median-level parallelism can be achieved (take advantage of hardware)
- Up to 25X application speedup

Future Work

- More efficient technique to parallelize median computation
- Inversion median core
- Design tree generation and bounding cores
  - Bottleneck for larger input sets
  - Initial design requires only 2 BRAMs
- Develop and package library of phylogenetic cores
- Develop tool that generates architecture based on characteristics of the input set

Raw Data for Median Core

Distance	speedup (1 core)	stdev (1 core)	speedup (4 cores)	stdev (4 cores)	speedup (8 cores)	stdev (8 cores)	speedup (12 cores)	stdev (12 cores)	speedup (16 cores)	stdev (16 cores)	speedup (20 cores)	stdev (20 cores)
8	1.650	0.571	2.129	11.640	1.685	1.581	1.195	0.418	0.898	0.319	0.746	0.976
9	1.639	0.833	2.076	9.237	1.766	1.757	1.256	0.453	0.933	0.323	0.745	0.321
10	1.586	0.667	1.661	1.352	1.729	1.515	1.261	0.417	0.980	0.320	0.783	0.302
11	1.537	0.594	1.764	1.661	1.641	1.392	1.277	0.462	1.030	0.369	0.821	0.344
12	1.478	0.934	1.642	1.576	1.786	1.996	1.415	0.971	1.102	0.720	0.894	0.563
13	1.455	0.865	1.837	7.271	1.981	10.489	1.336	0.900	1.113	0.508	0.922	0.503
14	1.422	0.838	1.691	1.557	1.694	1.761	1.502	2.410	1.254	3.176	0.982	0.647
15	1.831	5.097	2.034	5.268	2.453	10.033	1.838	5.571	1.725	6.272	1.464	5.183
16	1.878	4.473	2.236	5.300	2.710	11.572	2.074	5.264	2.131	6.200	1.677	4.490
17	2.875	10.024	2.554	7.226	3.720	15.743	3.203	11.842	2.956	10.310	2.374	8.533
18	2.944	11.476	3.294	12.040	3.544	13.132	3.283	12.922	3.013	12.152	2.819	11.991
19	3.615	17.156	3.586	15.249	4.081	17.842	4.122	19.395	4.222	19.271	3.736	16.515
20	5.522	37.323	5.892	39.076	6.351	47.605	7.231	48.318	7.180	46.982	6.773	46.566
21	5.480	25.792	5.654	26.362	5.647	24.849	11.518	159.558	10.395	138.232	10.182	137.706
22	7.776	43.555	7.567	42.572	8.198	46.117	9.960	49.148	9.514	48.252	9.237	45.502
23	13.057	86.250	12.755	83.839	13.411	92.422	15.020	94.865	14.259	89.157	13.397	77.622
24	9.864	35.560	10.167	35.983	15.766	128.929	26.416	309.879	25.381	297.798	24.923	296.536

Raw Data for Accelerated-GRAPPA

distance	mean speedup	stddev speedup
9	0.506	0.136
10	1.412	1.926
11	1.440	1.251
12	7.942	12.513
13	22.648	49.622