Solving Large Graph Problems Using Tree Decompositions: A Computational Study

Presented by

Blair D. Sullivan

Complex Systems Group
Center for Engineering Science Advanced Research

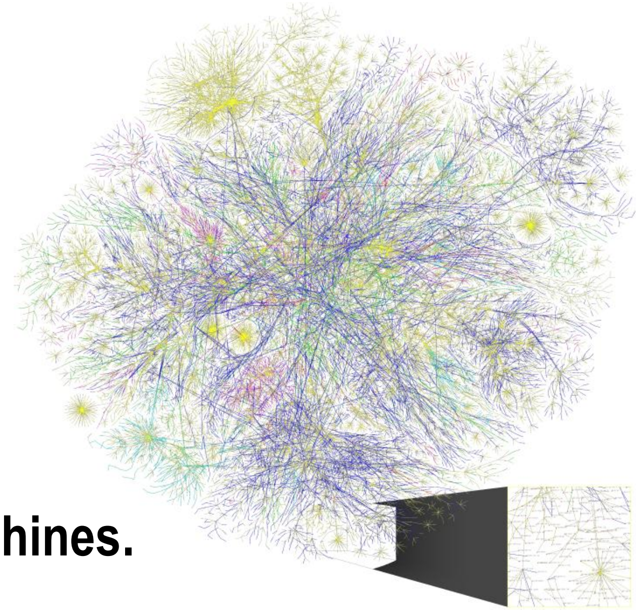
Chris Groër

Computational Mathematics Group

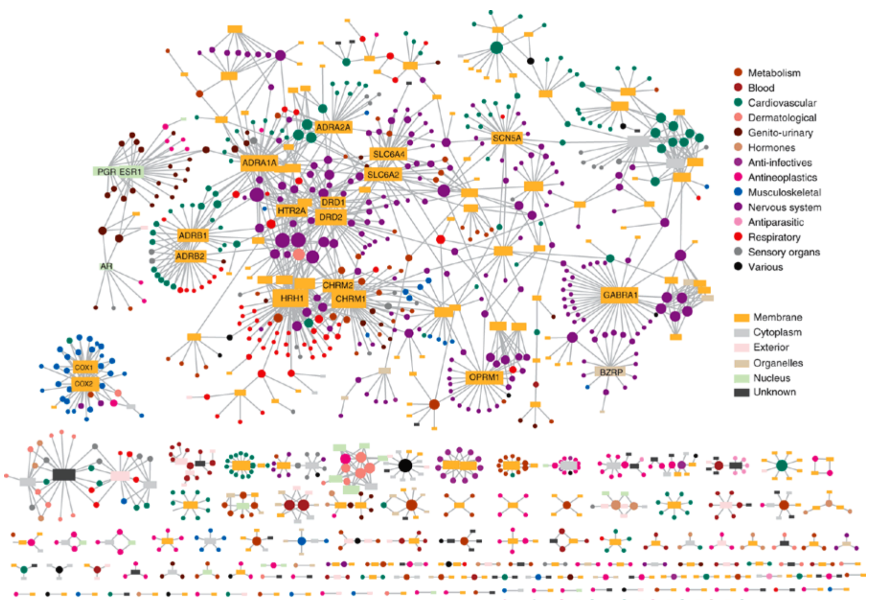
Research supported by the Department of Energy's Office of Science
Office of Advanced Scientific Computing Research
Applied Mathematics Program

Background

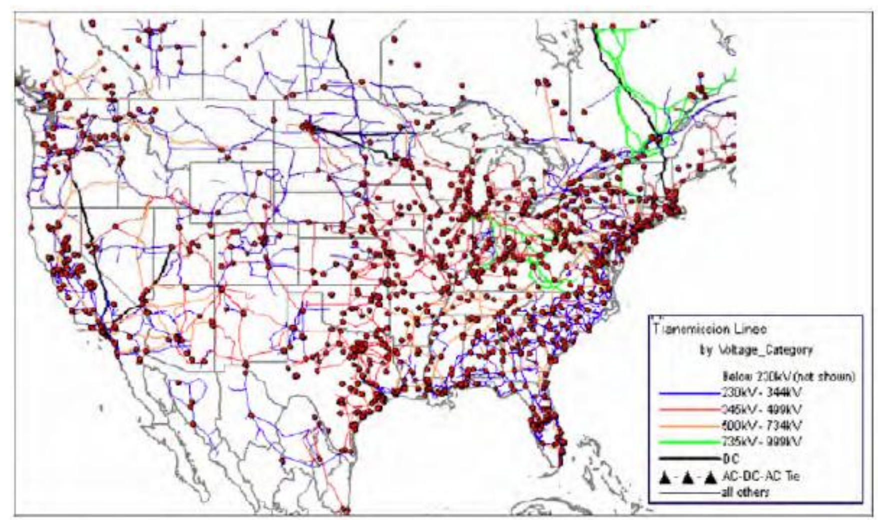
A partial map of the Internet, January 15 2005



- **Large networks are becoming ubiquitous in many domains – biology, physics, chemistry, infrastructure, communications, and sociology**
- **Graph problems have high computational complexity and require excessive computation for large networks**
- **Hard to solve efficiently on distributed memory machines.**



Drug-Target Network.
Nature Biotechnology 25(10), October 2007



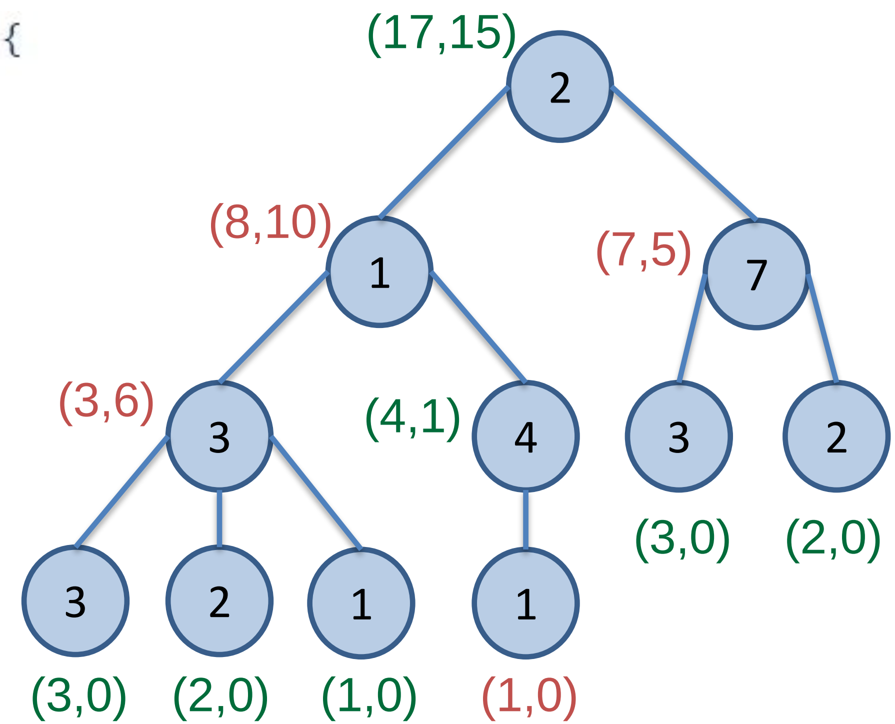
The US electric transmission system.
Courtesy North American Reliability Corporation.

Motivation: graph problems are easier on trees

- **Many NP-hard problems can be solved in polynomial time on trees (graphs with no cycles)**

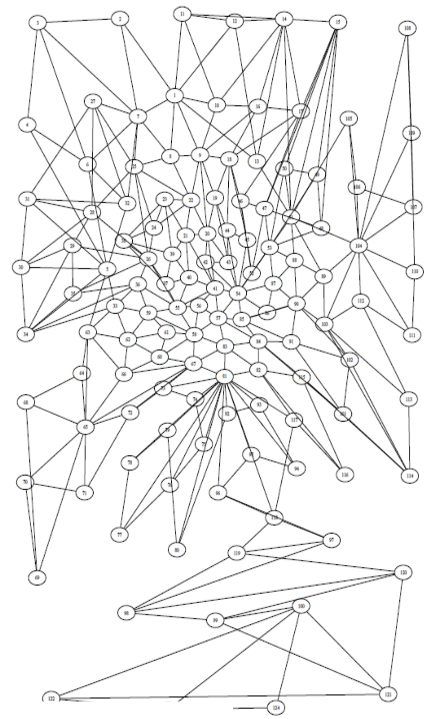
Example: Maximum Weighted Independent Set: Complexity O(|V|)

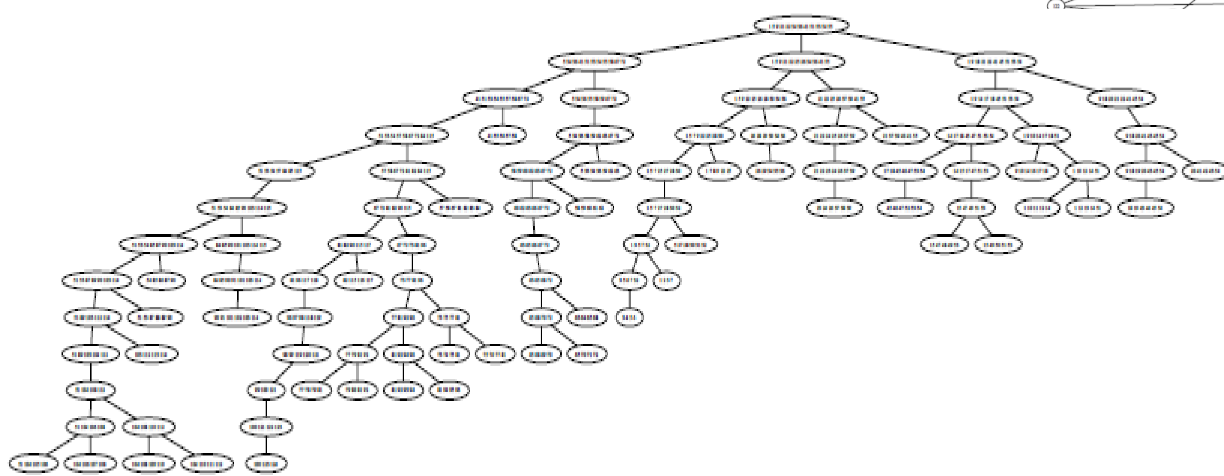
Maximum-weight independent set (Tree t, Node root) {
    For all nodes u of T in post-order {
        if u is a leaf {
            $M_{out}[u] = 0$
            $M_{in}[u] = w_u$
        }
        else {
            $M_{out}[u] = \sum_{v \in children(u)} \max(M_{out}[v], M_{in}[v])$
            $M_{in}[u] = w_u + \sum_{v \in children(u)} M_{out}[v]$
        }
    }
    return max(M_{out}[root], M_{in}[root])
}



Idea: generalize to “tree-like” graphs

- Tree decompositions are specialized maps of graphs onto trees, with subsets of V assigned to nodes of T.
- A graph is more “tree-like” if the subsets are all small.
- Many NP-hard decision/optimization problems are fixed-parameter tractable w.r.t. max subset size in map.
- This includes all problems expressible in second order monadic logic, including coloring, partial constraint satisfaction, maximum clique/independent set.

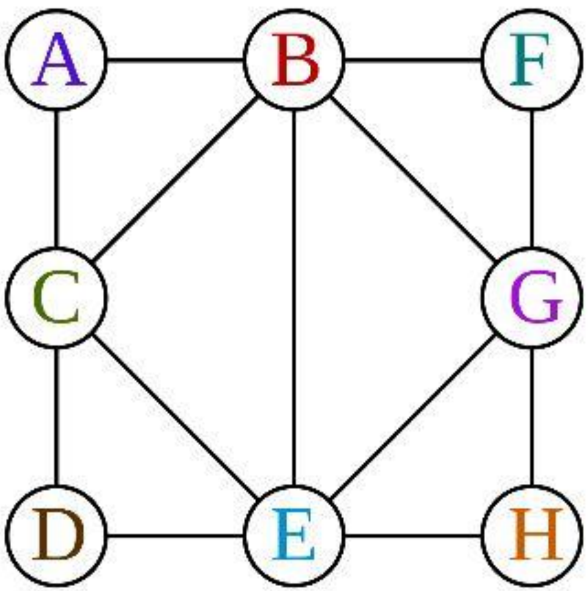


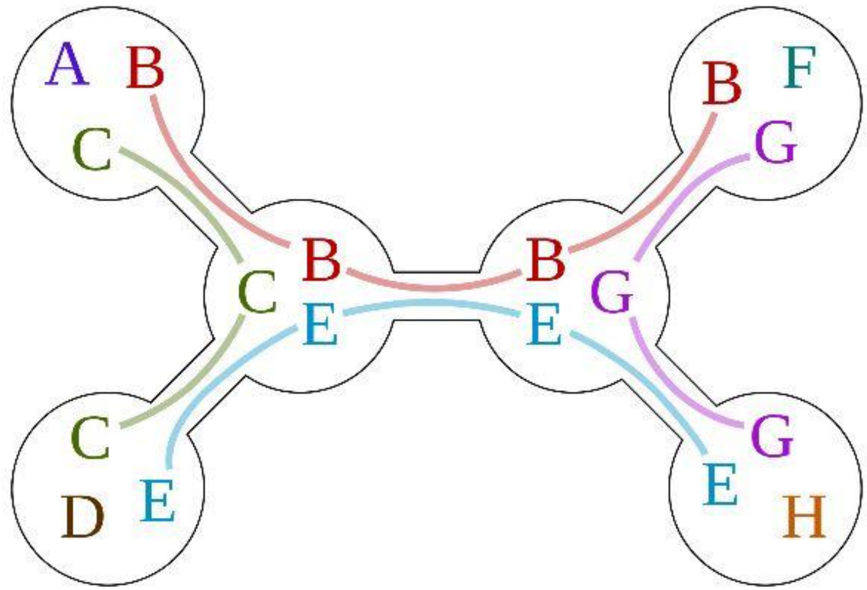


Tree Decompositions

Formally, a *tree decomposition* of a graph $G = (V, E)$ is a pair (X, T), where $X = \{X_1, \ldots, X_n\}$ is a collection of subsets of V and $T = (U, F)$ is a tree with $U = \{1, \ldots, n\}$, satisfying three conditions:

1. the union of the subsets X_i is equal to the vertex set V $(1 \le i \le n)$,
2. for every edge uv in G, $\{u, v\} \subseteq X_i$ for some $i \in [1, n]$, and
3. for every $v \in V$, if X_i and X_j contain v for some $i, j \in [1, n]$, then X_k also contains v for all k on the (unique) path in T connecting i and j. In other words, the set of nodes whose subsets contain v form a connected sub-tree of T.

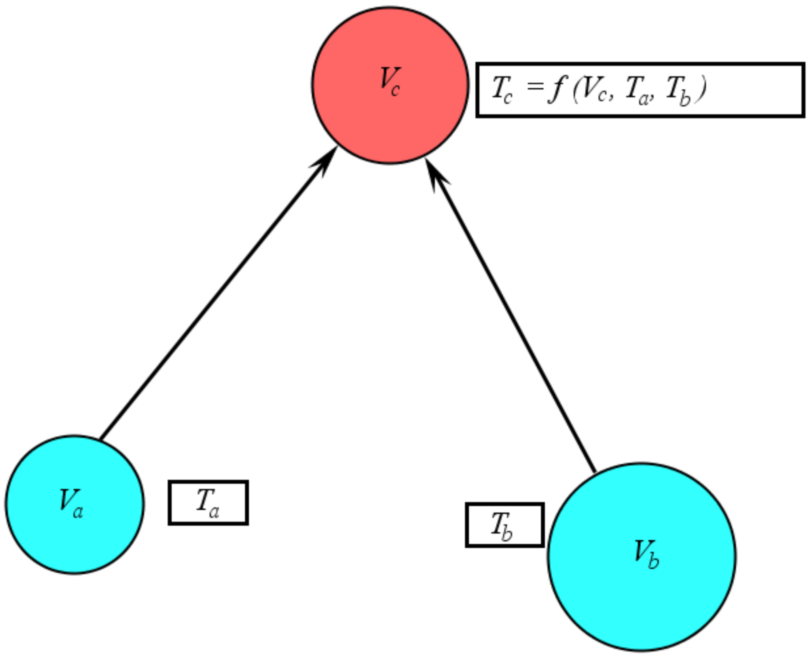




The *width* of a tree decomposition is max($|X_i|$-1), and the *treewidth* of a graph is the minimum width over all valid tree decompositions.

Dynamic Programming

- Solving decision/optimization problems uses DP on the tree decomposition.
- The general strategy is to root the tree and then work “up” from the leaves, solving sub-problems & storing partial solutions (tables) along the way, as in MWIS on a tree.



In a tree decomposition, computing the dynamic programming table at node *c* requires information about the vertices in the bag V_c and the children’s tables, T_a and T_b. The complexity of this computation can be exponential in $|V_c|$.

- Solving the sub-problems requires information about only a small part of the original graph, represented by the child nodes lower in the tree.
- The complexity of processing a specific node can be exponential in its bag size

Progress

"Tree decomposition based algorithms are a valuable alternative whenever the underlying graphs have small treewidth. As a rule of thumb, the typical border of practical feasibility lies somewhere below a treewidth of 20 for the underlying graph"
-Huffner, Niedermeier, Wernicke (2007)

- **Sequential code improvements include:**
  - **bitwise representation of vertex subsets that enable fast unions, intersections, quick elimination of families of non-independent subsets**
  - **storage of reduced dynamic programming tables using parent-child intersection properties (5-10% memory reduction on average test graph, more for sparser examples)**
  - **two-stage refinement technique for the decomposition to solve optimization problem with solution reconstruction makes two dynamic programming sweeps (demonstrated storage savings of up to an additional 80-85%)**
- **These improvements enabled the computation of MWIS on a 2 million node graph with a decomposition of over 1.8 million tree nodes, and on graphs with widths over 400.**

Some Implementation Details

- We represent a subset *S* in X_t as a single 128-bit word (type uint128_t) where a 1-bit in position *i* indicates that the *i*-th entry in X_t is in *S*
- The required union/intersection operations can be done via AND's and OR's
- The binary representation is convenient to rule out many of the 2^k possibilities in a single stroke (where $|X_t| = k$):
  - Suppose the edge *(7,11)* exists in *G*, so that any set *S* containing both *7* and *11* is not independent.
  - Let *{13,11,7,5,3,2,1}* be a bag, and consider what happens as we process subsets. We will encounter the set *{7,11}* as:

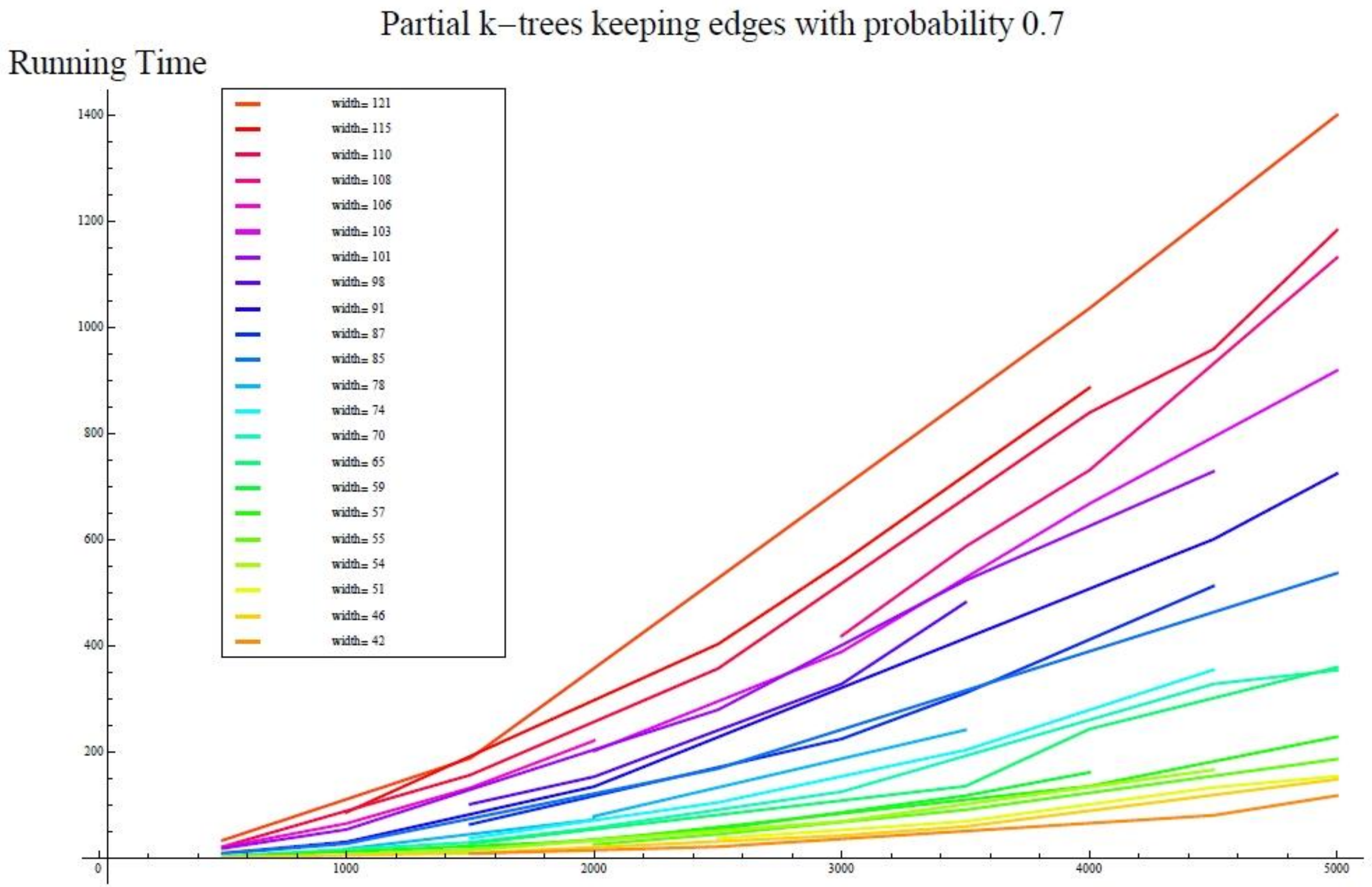
Bag	13	11	7	5	3	2	1
Mask	0	1	1	0	0	0	0

  - Now we know any mask of the form *011***** cannot represent an independent set and we eliminate $2^4 - 1$ additional possible subsets at once

– Several other similar tricks are used to speed up the computations

Running Time Analysis

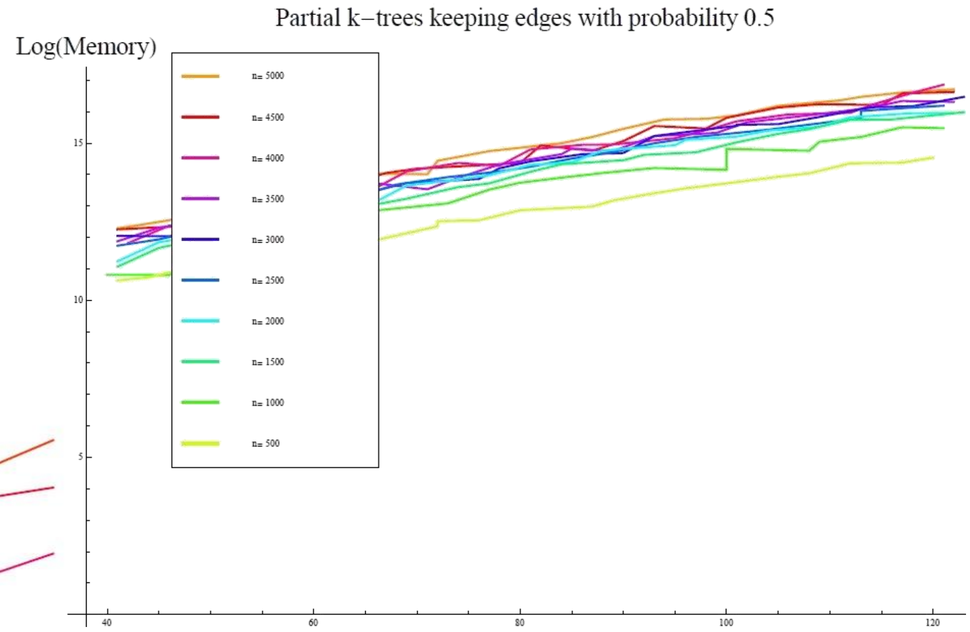
Demonstrated (approx) linear growth in running time with graph size for fixed width

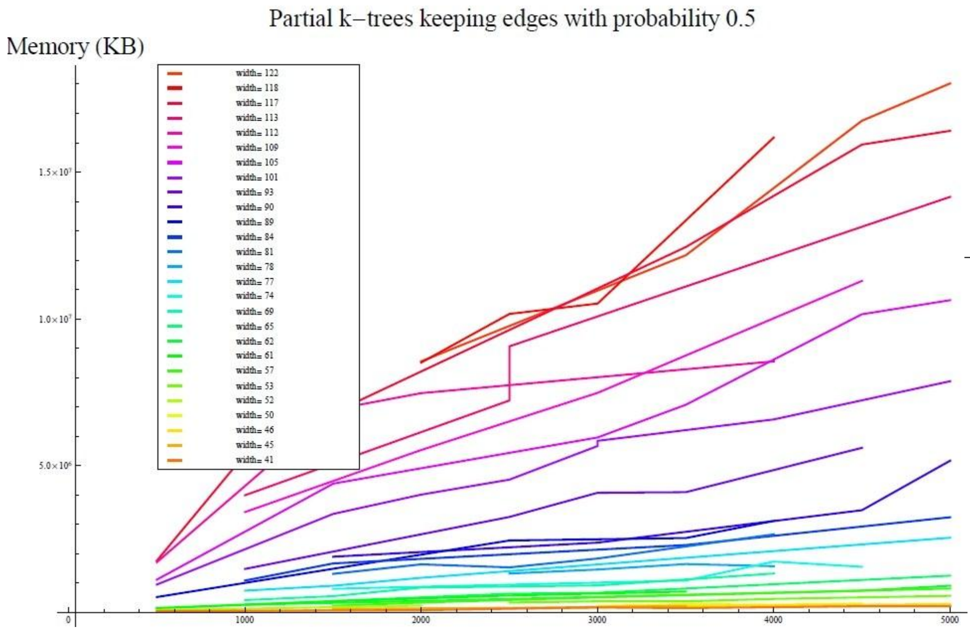


- Generated k-trees for k =50; 52; … ; 120 with n = 500; 1000; … 5000.
- Solved MWIS to optimality; recorded total time.
- Lines in the plot correspond to runs with a fixed decomposition width

Maximum Memory Usage

Memory usage also seems to grow linearly with graph size for fixed width, but exponentially with width for fixed graph size

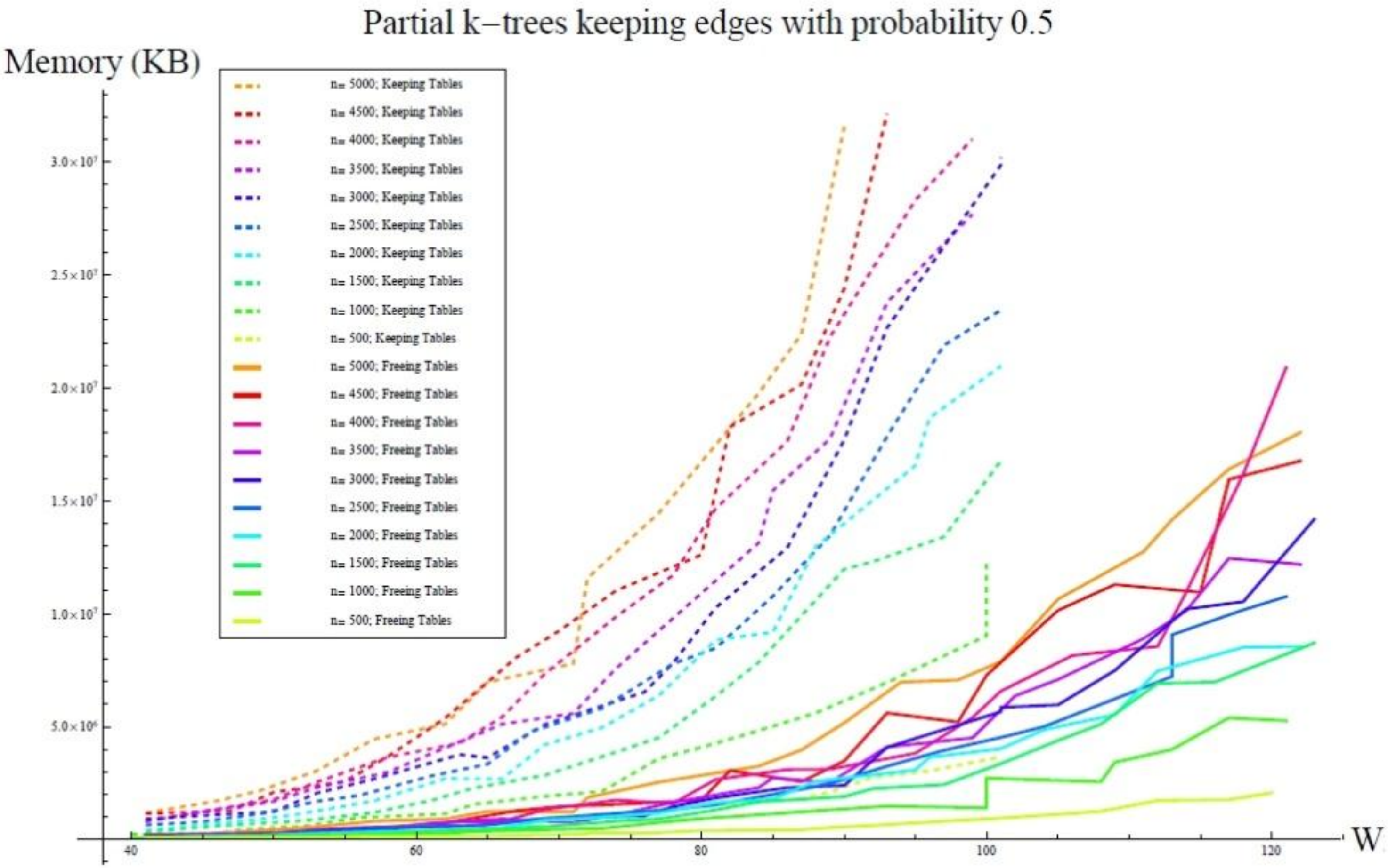




- Generated k-trees for k =50; 52; … ; 120 with n = 500; 1000; … 5000.
- Solved MWIS to optimality; recorded total time.
- Lines in the left plot correspond to fixed width, right plot correspond to fixed size

Memory Savings from Reconstruction Pruning

Second DP sweep on reduced decomposition reduces consumption drastically



- Generated k-trees for k =50; 52; … ; 120 with n = 500; 1000; … 5000.
- Solved MWIS to optimality; recorded memory high water mark.
- Negligible additional computation time required for second DP pass.

Contacts

Blair D. Sullivan

Complex Systems Group
Computer Science & Mathematics Division
Oak Ridge National Laboratory
sullivanb@ornl.gov

Chris Groër

Computational Mathematics Group
Computer Science & Mathematics Division
Oak Ridge National Laboratory
groercs@ornl.gov

Additional Project Members

Dinesh Weerapurage

Oak Ridge National Laboratory
weerapuraged@ornl.gov