Meaningful Change Detection in Structured Data

Sudarshan S. Chawathe, Hector Garcia-Molina

Presented by Lawrence Lin

Focus

- Detecting meaningful changes in hierarchically structured data
- Use operations that move and copy entire subtrees to describe changes meaningfully with regard to semantic information
- Algorithm reduces problem to computing a minimum-cost edge cover of a bitartite graph.

Change detection examples

- Detecting changes in a competitor's website.
- System administrator detecting differences between mirrored file systems.
- Engineer comparing different but related chip designs.

Model

- Rooted, labeled trees for structured data.
- Each node n has a label $l(n)$.
- A tree T is defined by nodes N, parent function p, and labeling function l. $T=(N,p,l)$
- A cost model for edit operations is defined, so goal is to find a minimum-cost script transforming one tree to another.

Operations

- Insertion: places a new node with a given label at a given position in the tree
- Deletion: DEL(n) removes n from the tree and makes its children the new children of its parent node.
- Update: UPD(n,v) changes the label of the node n to v.

Operations (cont.)

- Move: MOV(n,p) moves the subtree rooted at n to another position in the tree specified by the new parent p.
- Copy: CPY(m,p) copies the subtree rooted at n to another position.
- Glue: inverse of copy, GLU(n1, n2) causes subtree rooted at n1 to disappear when n1 and n2 are isomorphic.

Edit Script

- An *edit script* is a sequence of zero or more edit operations that can be applied in the order in which they occur in the sequence.

Cost Model

- Each operation has a given cost, given by constants ci, cd, cm, cc, and cg.
- With certain operations being symmetric, ci = cd, cc = cg.
- Also, cm < cc.

The Graph

- We start with the initial tree T1 and the final tree T2
- The idea is to find, for each node in T1, its corresponding node in T2.
- We start with a graph containing dashed lines connecting nodes in T1 to nodes in T2, with all the possible operations that can make the transformation.
- We want to find a subset K of the edges of the graph B, telling us the correspondences.

Getting the Answer

- First, we use conservative pruning rules, removing edges of the graph which we are sure cannot be part of a minimum-cost edit script.
- Then the edges that are not needed to cover nodes (ie. choosing to eliminate an edge or subset of edges whose action is accomplished redundantly).

Getting the Answer (cont.)

- Once the cost is defined for each edge in the pruned induced graph, standard techniques are used to reduce the problem to a weighted matching problem, and then further to solve that.

CtoS

- Generates an edit script between two trees, given an edge cover of their induced graph.
- With the edge cover, edit operations are computed in several different phases to ensure simplicity (ie. INS phase after DEL phase).
- Order is DEL, CPY, UPD, MOV, GLU, INS.

DEL Phase

- In DEL phase, if a node m is connected to – (deletion node), a DEL operation is added to the edit script.
- Any node attached to – is absent from the final tree.

CPY Phase

- Algorithm searches for edges incident on a common node *m* in T1.
- It ignores nodes generated through a copy of some ancestor.
- Remaining edges found in this search are logged as CPY operations.

Remaining Phases

- UPD phase: straightforward, records a CPY operation when an edge connects nodes whose labels differ.
- MOV: also straightforward (not mentioned in paper)
- GLU, INS: analogous to CPY, DEL respectively

MH-DIFF

- MH-DIFF is the algorithm which finds a minimal edge cover of the induced graph.
- The goal is to find not just any minimal edge cover, but one that corresponds to a minimum-cost edit script, known as a target cover.

Choosing Edges

- The algorithm must decide for each edge whether it should be included in the cover.
- The actual cost would be useful, but it creates a “chicken and the egg problem.”
- Solution: compute upper and lower bounds to the cost.

Pruning Rules

Pruning Rule 1 Let $C_t = \max\{c_m, c_c, c_g\}$. If $c_{lb}(e_1) \geq c_{ub}(e_2) + c_{ub}(e_3) + 2C_t$ then prune e_1.

- Take an edge e1 which we are considering pruning. Let n1 be the node in T1 and n2 be the node in T2. If the lower bound cost of e1 is higher than the combined cost of another edge connected to n1 and another edge connected to n2, we can prune e1.

Pruning Rules (cont.)

Pruning Rule 2 If $c_{lb}(e_1) \geq c_d(m) + c_i(n)$ then prune e_1.

- Essentially, if it costs less to delete one node and insert another, then we can eliminate the edge matching the two nodes to each other.

Choosing a Minimal Edge Cover

- After pruning, there may still be several minimal edge covers possible for the pruned induced graph.
- Use the lower bound (or upper, or an average) to approximate the cost of every edge remaining.
- Given constant estimated costs, reduce the edge cover problem to a bipartite weighted matching problem, which has established solution methods.

Choosing a Minimal Edge Cover (cont.)

- Weighted matching problem can be solved in O(ne) time, with n nodes and e edges.

Performance

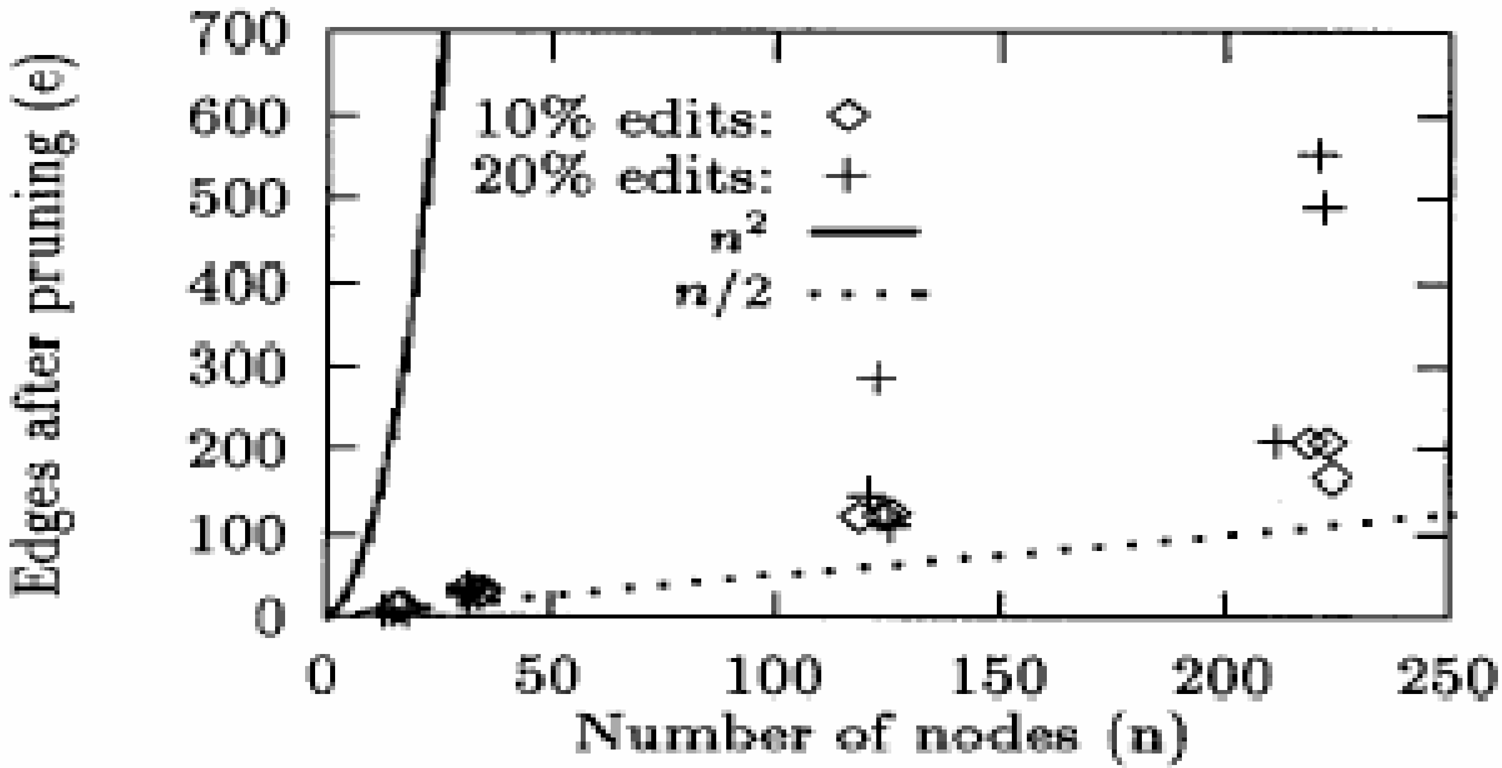

Figure 10: Effectiveness of pruning

Performance (cont.)

- 50 experiments were run comparing the result of MH-DIFF to the perfectly optimal edit script.
- In 48 (96%), MH-DIFF found the optimal edit script, and the script costs of the remaining 2 were about 15% above the minimum possible.

Summary

- The method presented compares data structures and determines the minimum edit script to transform the first into the second.
- Edit scripts contain a set of edit operations, arranged in a sequence.
- Trees are constructed with edges representing edit operations, and the minimum cost edge cover chosen by the algorithm presented.