# Synthetising Changes in XML Documents as PULs Extended Experiments

Federico Cavalieri, Alessandro Solimando, Giovanna Guerrini
Università di Genova, Italy - {name.surname}@unige.it

## 1. EXPERIMENTS

Since no suitable versioned XML document collection is available to use in our experiments we relied on synthetic documents, generated as follows. The source document is either produced by means of the XMark document generator [1] or selecting a random subset of the first-level children of the DBLP XML document[2] (1.16GB at the time of writing). These two kinds of source documents presents different structural characteristics. XMark documents have a complex organization of XML subtrees representing several different entities, whereas DBLP-based documents are a simple lists of subtrees describing scientific publications. To produce the target document we randomly generate a PUL to be applied on the source document. During the PUL generation we take extreme care to ensure that the generated PUL is also one of the minimal cost edit-script that transforms the source document in the target one. These PULs contain randomly generated `del`, `ins`$^d$, `repN`, `ren`, `repV` operations. Inserted and replaced subtrees can be randomly generated, similar or equal to another subtree of the source document, or randomly selected from another XMark document (or from the complete DBLP document). New names and values can be randomly generated, or be randomly selected from the set of names and values in another XMark document (or from the complete DBLP document). We also simulate subtree moves through pairs of deletion and insertion operations. The number of each operation type is roughly the same. Altough we run the experiments with many different distributions of operation types, for brevity we only report the results obtained with this distribution. Moreover, we repeated each experiment twice, once with XMark documents and once employing DBLP-based documents, and no significant difference was found. To easily combine and contrast the results obtained considering documents of different sizes, we do not reason directly on the tree edit-distance, rather we consider the *change ratio*, that is, the edit-distance divided by the source document weight.

In the experiments we considered $P = 2$, $Q = 3$ in *pq*-grams and *pql*-grams generations, $k = 5$, with no limits on the approximated matching window size.

The tests has been performed on a PC equipped with an *Intel Core i7-2670QM* CPU, 16GB of RAM and running *Kubuntu Linux 12.10 64-bit*. For increasing statistical significancy, every time measurement is the average of at least 50 samples.

In the remainder of the section, we first experimentally evaluate the quality of the PUL edit-distance estimation obtained by means of tree-grams, then we empirically test the soundness of our window-based approach used in top-down refinement stage, and we finally provide a time and edit-script cost comparison with other state of the art XML differencing algorithms.

### 1.1 PUL Edit-distance Estimation Using Tree-Grams

This section is devoted to the experimental evaluation of the quality of the PUL edit-script distance estimation using *pq*-grams and *pql*-grams. Specifically, given two trees $S$ and $T$, we want to verify that, independently from the kind of operations which are necessary to transform $S$ into $T$: (i) the estimated change ratio increases as the real change ratio increases, (ii) for any given real change ratio, the estimated change ratio has a low variance.

We stress that, given a real change ratio $r$, we are not aiming to estimate it as $r$, rather it is sufficient to have a low-variance estimation, even if the average estimated value is distant from $r$. For instance we can observe that the estimated change ratio tend to be over-estimated for small change ratios and small documents. These anomalies could be trivially reduced multiplying the obtained measure with non-constant coefficients. Similarly, over/under estimations with specific kind of operations could be corrected, at least in part, employing a more sophisticated tree-gram comparison function. However, especially because we are reasoning with syntethic documents we want to avoid the risk to overfit the estimation to our data. For this reason the *pq*-gram and *pql*-gram estimations are obtained computing the Jaccard distance between the tree-grams of the source document and those of the target document and we consider the fine tuning of our estimations as a future work. Moreover, we stress that tree-grams distance will only be used to select in a sequence of trees the $k$ most similar trees to a given one. Even if the $k$-best matches, according to the estimation, are sub-optimal, the application of the HIPS algorithm will often choose a perfect or near-perfect set of matches.

We considered not only the non-homogeneous random PULs discussed in Section 1, but also homogeneous PULs composed by `del`, `ins`$^d$, `repN`, `ren`, `repV` or move operations. For each operation we investigated also additional features: (i) the deleted subtrees weight range, for `del` operations, (ii) the number of inserted subtress, their weight range and generation algorithm for

[1] http://www.xml-benchmark.org/
[2] http://www.informatik.uni-trier.de/~ley/db/

`ins`$^d$ and `repN` operations, (iii) the number of adjacent sibling moved subtrees, for move operations, (iv) the name generation algorithm for `ren` PULs, (v) the value generation algorithm for `repV` PULs. We considered the following inserted tree generation algorithms: (i) *random trees*, (ii) trees which have a *similar structure* (10% of the nodes are renamed, 70% of the text values are changed) to either one of the inserted subtree sibling or to a subtree at the same nesting depth, (iii) trees which have a *similar size* to that of their new sibling/replaced node, but random structure, names and values, (iv) *real trees*, randomly extracted from another XMark/DBLP document, among the subtrees which usually occur in the considered document, at the considered position (*e.g.*, a new `article/person` can be inserted next to another `article/person`). Random trees are roughly generated as follows: (i) trees of weight 1–2 consist of a single element node eventually with a single child text node, (ii) trees of weight 3–20 consist of a single element node containing one or more trees at point (i), (iii) trees of weight 21–80 consist of a single element node containing two to ten trees at point (i or ii), and so on. Tree height is limited to 6 and the structure roughly resembles that of an XMark document.

Figures 1, 2, 3 and 4 contrast the real and estimated change ratio for PULs with different characteristics, among those which we consider more common in XML databases. Each measurement is repeated 300 times on xmark and dblp random documents whose size ranges from 1KB to 2MB. For the sake of conciseness we do not report the results for every investegated PUL kind in these figures. The interested reader can refer to Table 1, 2, 3 and 4 for a larger selection of PUL peculiarities. In the Tables we report both the average estimated value and its standard deviation. As can be observed from both the Figures and the Tables, the estimated change ratio increases as the real change ratio increases, and for any given real change ratio $r$, the estimated change ratio has a low variance. Moreover, *pql*-grams do not over-estimate change ratios when `ren` or `repV` operations are required to transform the two analyzed documents.

Finally, we generated roughly 1 million source documents and PULs, with different corresponding change ratios, uniformly distributed among the following real change ratios (0.01, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6 and 0.7), the following kinds (random, homogeneous `del`, `ins`$^d$, `repN`, `ren`, `repV` and move). Moreover, for each considered kind the PUL are uniformly distributed among the types investigated in Tables Table 1, 2, 3 and 4. We report in Table 5, the estimated change ratios with both *pq*-grams and *pql*-grams, and the estimations standard deviation, which experimentally proves that *pql*-grams are a more precise estimation than *pq*-grams for PUL edit-distance.

## 1.2 Window-based Sequence Test

In the top-down refinement stage a sequence of source subtrees is contrasted with a sequence of target subtrees aiming at finding the heaviest consistent set of matches between them.

The algorithm employs an approximated approach: each target subtree is compared employing *pql*-grams with all the source subtrees in its search window. The best $k$ matches for each target subtree are inserted into a set $M$. The HIPS algorithm is then invoked on $M$ identifying the heaviest consistent subset of $M$.

In this experiment we aim at replicating the window-based match settings producing subtree sequences, where the size of each subtree ranges from 20 to 60 nodes. Target documents are generated employing random non-homogeneous PULs as discussed in Section 1. The quality of the chosen matches for entire sequences of children can be desumed from the experiments in 1.3. In this experiment we consider the worst-case scenario: in each sequence all subtrees are very similar to each other (*e.g.*, all DBLP `article` elements or all `auction` XMark elements), modifications are roughly uniformly distributed among subtrees.

For each generated pair of sequences we contrast the cost difference between the edit-script generated by PUL-Diff ignoring all perfect matches (to simulate matching two sequences of unmatched subtrees), the edit-script generated by PUL-Diff and an optimal edit-script.

Results for different change ratios, sequence length (from 10 to 500) and different values for $k$ are reported in Figures 5, 6 and 7.

In each plot we devote the left (resp. right) side to represent the result obtained by without (resp. with) perfect matches.

As can be observed in the Figures, independently from the change ratio, sequence length, and value of $k$ the selected matches are on average very precise. Indeed, in no cases the cost difference w.r.t. the optimal solution is greater than 5%. Increasing the value of $k$ above 3 when perfect matches are employed has negligible impact. When perfect matches are not employed, increasing the value of $k$ to about half the length of the sequence yields to edit-scripts as expensive as those identified with perfect matches. However, the average distance between the edit-script obtained without perfect matches and the one obtained with perfect matches is extremely small (1-2% w.r.t. the optimal edit-script). Surprisingly, increasing the length of the sequence *reduces* the distance between the computed edit-scripts and the optimal one, thanks to the HIPS algorithm.

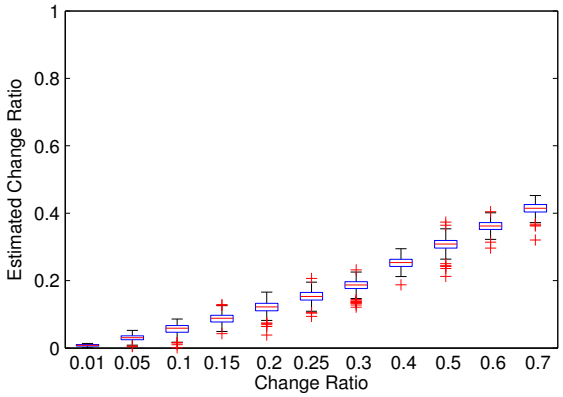
Results for extremely long sequences (from 1000 to 5000) are instead reported in Figure 8. When using perfect matches, the distance of obtained edit-scripts from the optimum still decreases as the sequence length increases and values of $k$ above 3 still have a neglegible impact. When perfect matches are not used, the distance of obtained edit-scripts from the optimum increases with sequences of 1000 subtrees to decrease again with sequences of 5000 subtrees. We can observe that the *pql*-gram-based match detection is less sensitive with change ratios near 0.3. In any case, even a moderate value of $k$ is still sufficient to keep the average distance well below 5%. Therefore, the proposed *pql*-gram-based distance metric, when paired with an HIPS-based top $k$ matches pruning algorithm is very reliable. Moreover, we remark that the synthesis operator benefits from perfect matches and thus the lenght of the sequences compared using this metric are much shorter.

## 1.3 Edit-script Cost and Differencing Time

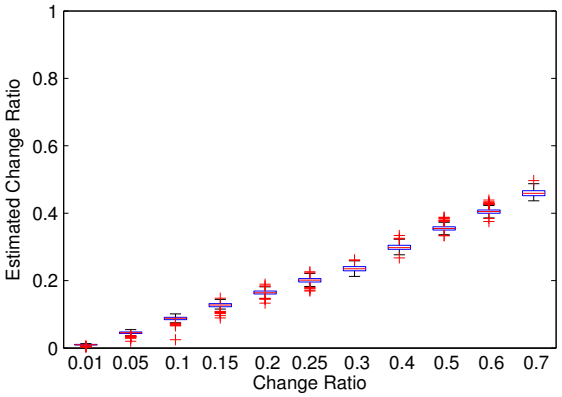
This test aims at empirically determining the time complexity of PUL-Diff for different document sizes and change ratios, as well as constrasting the edit-script cost and computational time of PUL-Diff with the state of the art. The PULs used to generate the target documents contains randomly generated `del`, `ins`$^d$, `repN`, `ren` and `repV` operations.

From the surveyed literature we identified several interesting XML differencing tools for ordered trees, including XyDiff [3], DeltaXML [4], MMDiff and XMDiff [1]. As can be observed in [2], MMDiff and XMDiff are not good candidates for our comparison as their objective is to identify optimal edit-script even at the expense of time complexity. For instance, in [2], the authors show that on 100KB documents XMDiff roughly requires one hour, MMDiff (main-memory version of XMDiff, which has quadratic memory complexity), roughly one minute, whereas both DeltaXML and XyDiff require less than a second.

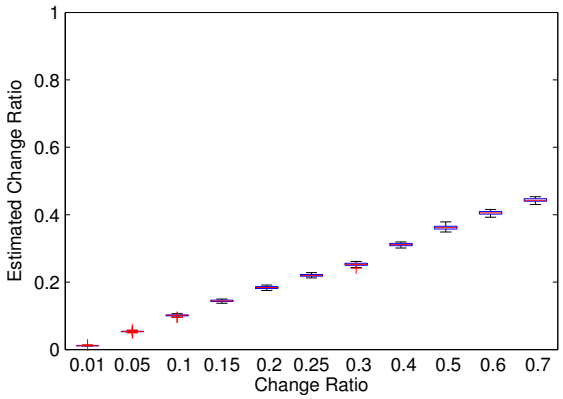
For this reason, we considered only XyDiff (C++ version) and DeltaXML 6.4.1 (Linux 14-days trial version, with a processing limit of 1 million nodes, around 7MB). The edit-script produced



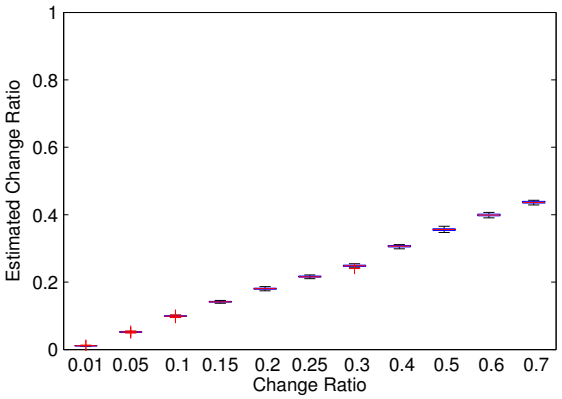

(a) Moves (1-3 siblings, tot. weight 1-500) (*pq*-grams)



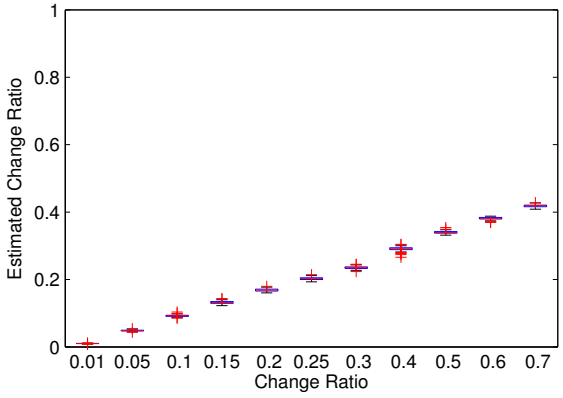

(b) Moves (1-3 siblings, tot. weight 1-500) (*pql*-grams)



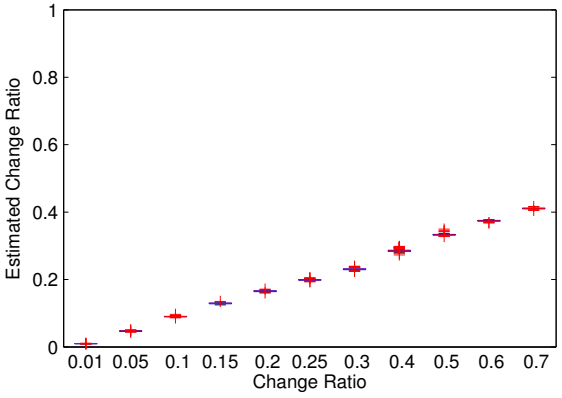

(c) Insertions (1-3 random trees, tot. weight 1-200) (*pq*-grams)




(d) Insertions (1-3 random trees, tot. weight 1-200) (*pql*-grams)

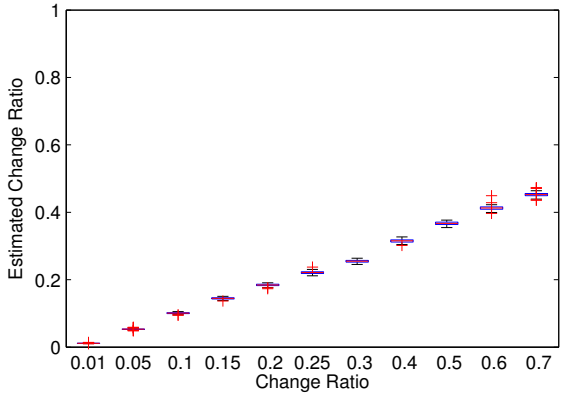



(e) Insertions (1-3 real trees, tot. weight 1-200) (*pq*-grams)

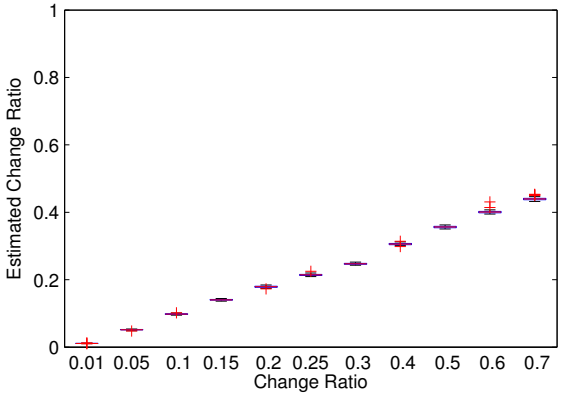



(f) Insertions (1-3 real trees, tot. weight 1-200) (*pql*-grams)

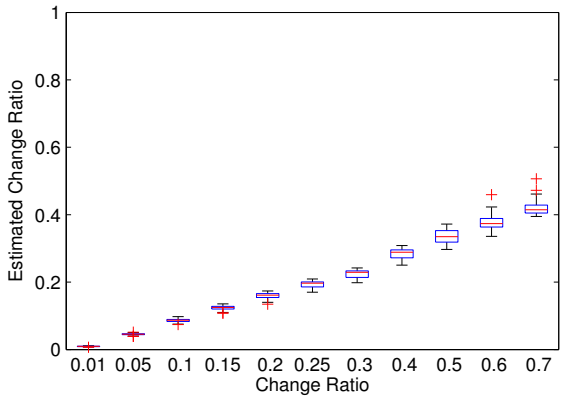
**Figure 1: Tree PUL edit-distance estimation with *pq*-grams and *pql*-grams (Part 1)**



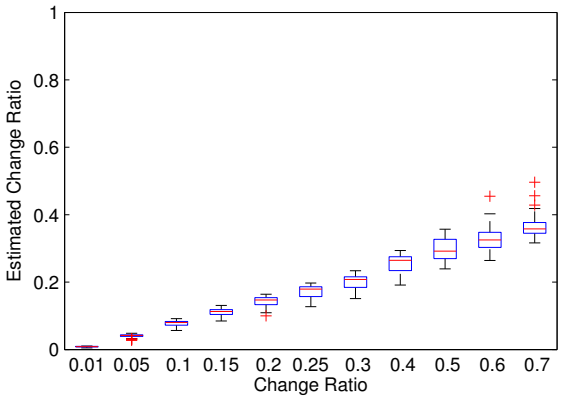

(a) Replacements (of nodes weighting 1-500, with 1-3 random trees with tot. weight 1-200) (*pq*-grams)



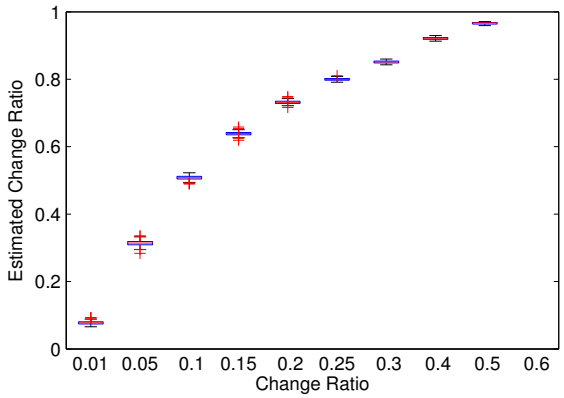

(b) Replacements (of nodes weighting 1-500, with 1-3 random trees with tot. weight 1-200) (*pql*-grams)



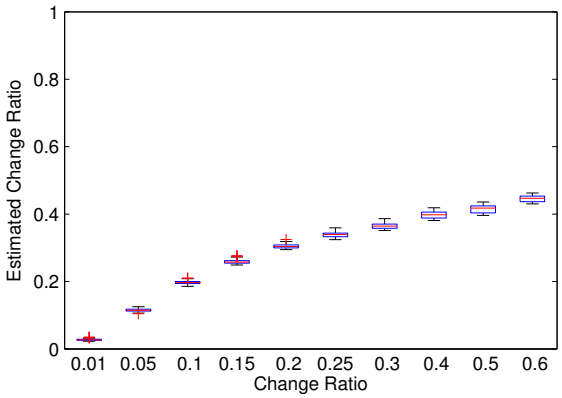

(c) Replacements (of nodes weighting 1-500, with 1-3 real trees with tot. weight 1-200) (*pq*-grams)




(d) Replacements (of nodes weighting 1-500, with 1-3 real trees with tot. weight 1-200) (*pql*-grams)

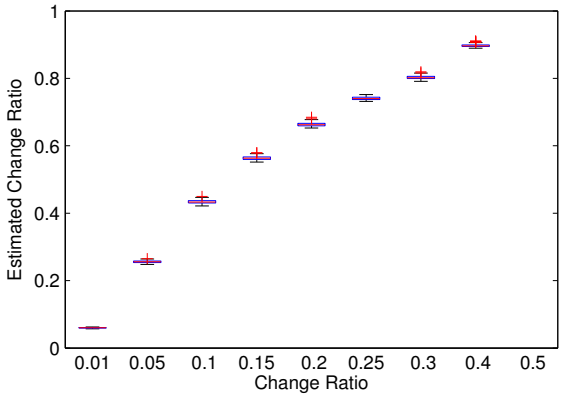



(e) Renames (to a name used by a node at the same nesting depth in the original document) (*pq*-grams)

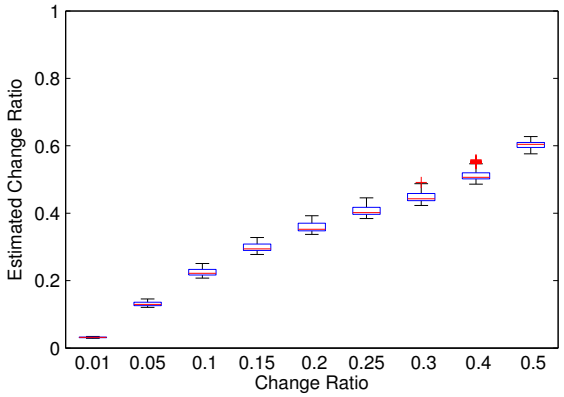



(f) Renames (to a name used by a node at the same nesting depth in the original document) (*pql*-grams)

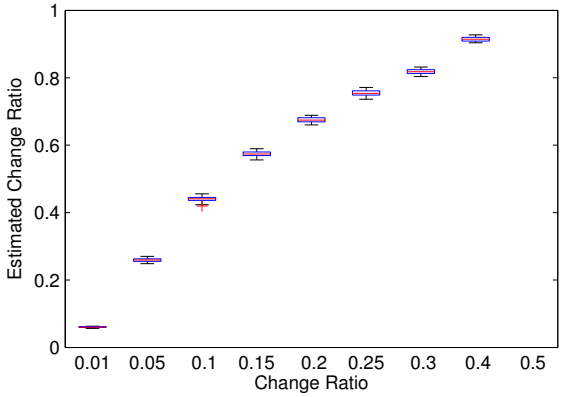
**Figure 2: Tree PUL edit-distance estimation with *pq*-grams and *pql*-grams (Part 2)**



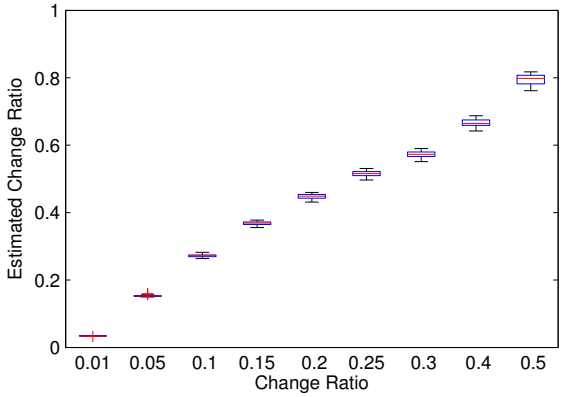

(a) Value replacements (with a value used in the original document) (*pq*-grams)



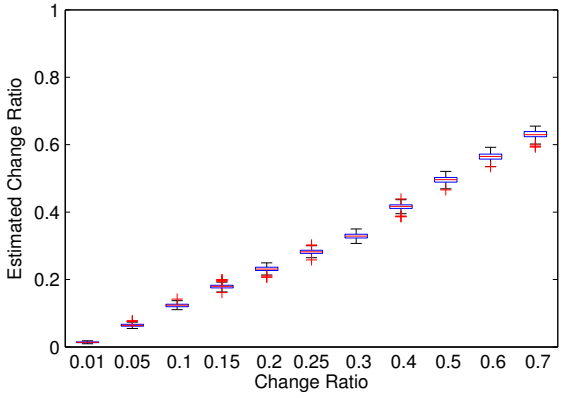

(b) Value replacements (with a value used in the original document) (*pql*-grams)



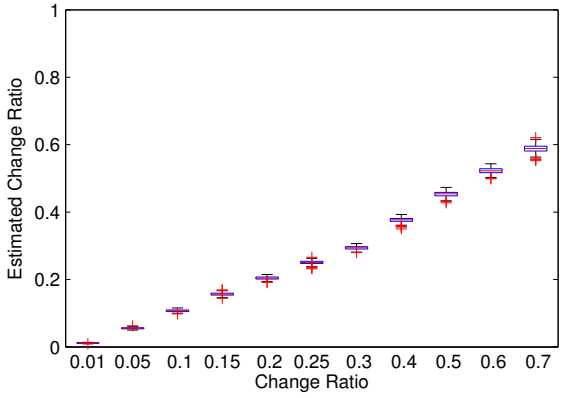

(c) Value replacements (with a random value) (*pq*-grams)




(d) Value replacements (with a random value) (*pql*-grams)




(e) Deletions (removed subtree weight 1-500) (*pq*-grams)




(f) Deletions (removed subtree weight 1-500) (*pql*-grams)

**Figure 3: Tree PUL edit-distance estimation with *pq*-grams and *pql*-grams (Part 3)**

| Operation type | **0.01** | **0.05** | **0.1** | **0.15** | **0.2** |
|---|---|---|---|---|---|
| `ren` to any existing name | 0.07995(5.85e-03) | 0.32298(1.36e-02) | 0.52075(1.54e-02) | 0.65080(1.28e-02) | 0.74279(1.04e-02) |
| `ren` to a random name | 0.08002(5.94e-03) | 0.32379(1.38e-02) | 0.52131(1.61e-02) | 0.65244(1.37e-02) | 0.74453(1.11e-02) |
| `ren` to a name of a node at the same level | 0.07975(5.86e-03) | 0.31904(1.17e-02) | 0.51359(9.70e-03) | 0.64291(7.18e-03) | 0.73323(5.47e-03) |
| `ren` to a name of a node with the same parent name | 0.07926(5.51e-03) | 0.31959(1.04e-02) | 0.51473(1.05e-02) | 0.64307(7.26e-03) | 0.73370(5.65e-03) |
| `ren` to a name of a node at the same level +- 1 | 0.07954(5.85e-03) | 0.31956(9.98e-03) | 0.51469(9.59e-03) | 0.64364(6.74e-03) | 0.73366(5.51e-03) |
| Move 1 tree (total weight 0–500) | 0.00916(3.29e-03) | 0.03824(1.23e-02) | 0.07428(1.93e-02) | 0.11179(2.78e-02) | 0.14926(3.44e-02) |
| Move 2 trees (total weight 0–500) | 0.00692(2.67e-03) | 0.02466(9.98e-03) | 0.04386(1.63e-02) | 0.06433(2.37e-02) | 0.08279(2.78e-02) |
| Move 3 trees (total weight 0–500) | 0.00503(2.41e-03) | 0.01606(7.02e-03) | 0.02828(1.06e-02) | 0.03953(1.43e-02) | 0.05150(1.90e-02) |
| Move 4 trees (total weight 0–500) | 0.00415(2.15e-03) | 0.01104(5.35e-03) | 0.01778(7.26e-03) | 0.02592(8.38e-03) | 0.03418(1.17e-02) |
| `repN` with a random tree of a similar weight | 0.01551(1.18e-03) | 0.07260(4.67e-03) | 0.13833(7.77e-03) | 0.19788(1.04e-02) | 0.25219(1.22e-02) |
| `repN` with 1 random tree weighting 1–10 | 0.01563(1.40e-03) | 0.07460(5.50e-03) | 0.14119(8.40e-03) | 0.20223(1.12e-02) | 0.25708(1.34e-02) |
| `repN` with 1 random tree weighting 10–25 | 0.01598(1.22e-03) | 0.07471(5.06e-03) | 0.14014(8.87e-03) | 0.19811(1.15e-02) | 0.25017(1.32e-02) |
| `repN` with 1 random tree weighting 25–50 | 0.01663(1.27e-03) | 0.07654(5.21e-03) | 0.14313(8.76e-03) | 0.20137(1.15e-02) | 0.25254(1.38e-02) |
| `repN` with 1 random tree weighting 50–500 | 0.01664(1.21e-03) | 0.07711(5.18e-03) | 0.14327(8.92e-03) | 0.20066(1.11e-02) | 0.25091(1.34e-02) |
| `repN` with 2 random trees (total weight 50–100) | 0.01666(1.14e-03) | 0.07693(5.16e-03) | 0.14317(9.04e-03) | 0.20091(1.13e-02) | 0.25155(1.34e-02) |
| `repN` with 3 random trees (total weight 75–150) | 0.01673(1.09e-03) | 0.07708(5.12e-03) | 0.14347(8.82e-03) | 0.20086(1.14e-02) | 0.25179(1.32e-02) |
| `repN` with a tree similar to one at the same level | 0.01227(1.00e-03) | 0.05696(3.52e-03) | 0.10741(6.74e-03) | 0.15434(1.04e-02) | 0.19753(1.28e-02) |
| `repN` with a tree similar to one of its siblings | 0.01183(1.29e-03) | 0.05449(5.00e-03) | 0.10310(1.00e-02) | 0.14977(1.27e-02) | 0.19097(1.52e-02) |
| `repN` with 1 real tree | 0.00985(7.98e-04) | 0.04600(2.21e-03) | 0.08709(4.20e-03) | 0.12468(6.46e-03) | 0.16000(7.88e-03) |
| `repN` with 2 real trees | 0.00988(5.15e-04) | 0.04637(1.85e-03) | 0.08700(3.78e-03) | 0.12541(4.66e-03) | 0.15911(6.12e-03) |
| `repN` with 3 real trees | 0.00994(4.28e-04) | 0.04653(1.84e-03) | 0.08780(3.39e-03) | 0.12595(4.17e-03) | 0.16071(5.06e-03) |
| $\texttt{ins}^{\rightarrow}$ 1 random tree (weight similar to a sibling) | 0.01855(1.36e-03) | 0.08565(6.43e-03) | 0.15892(1.10e-02) | 0.22223(1.51e-02) | 0.27831(1.74e-02) |
| $\texttt{ins}^{\rightarrow}$ 1 node weighting 1–10 | 0.01867(1.39e-03) | 0.08759(6.00e-03) | 0.16210(9.99e-03) | 0.22673(1.31e-02) | 0.28315(1.52e-02) |
| $\texttt{ins}^{\rightarrow}$ 1 node weighting 10–25 | 0.01701(1.32e-03) | 0.07970(5.46e-03) | 0.14806(9.25e-03) | 0.20734(1.19e-02) | 0.25940(1.39e-02) |
| $\texttt{ins}^{\rightarrow}$ 1 node weighting 25–50 | 0.01702(1.29e-03) | 0.07941(5.25e-03) | 0.14724(9.10e-03) | 0.20615(1.21e-02) | 0.25736(1.38e-02) |
| $\texttt{ins}^{\rightarrow}$ 1 node weighting 50–500 | 0.01693(1.19e-03) | 0.07768(5.44e-03) | 0.14381(8.77e-03) | 0.20140(1.15e-02) | 0.25192(1.32e-02) |
| $\texttt{ins}^{\rightarrow}$ 2 nodes (total weight 50-100) | 0.01683(1.26e-03) | 0.07827(5.30e-03) | 0.14523(8.91e-03) | 0.20324(1.16e-02) | 0.25385(1.35e-02) |
| $\texttt{ins}^{\rightarrow}$ 3 nodes (total weight 75-150) | 0.01699(1.07e-03) | 0.07798(5.22e-03) | 0.14506(8.86e-03) | 0.20250(1.13e-02) | 0.25353(1.30e-02) |
| $\texttt{ins}^{\rightarrow}$ 1 tree similar to one at the same level | 0.01437(1.33e-03) | 0.06574(5.23e-03) | 0.12262(9.04e-03) | 0.17205(1.18e-02) | 0.21649(1.42e-02) |
| $\texttt{ins}^{\rightarrow}$ 1 tree similar to one of its siblings | 0.01392(1.57e-03) | 0.06414(5.94e-03) | 0.11942(1.06e-02) | 0.16847(1.35e-02) | 0.21258(1.80e-02) |
| $\texttt{ins}^{\rightarrow}$ 1 real tree | 0.01079(7.05e-04) | 0.04985(1.60e-03) | 0.09382(2.55e-03) | 0.13340(3.08e-03) | 0.16942(3.33e-03) |
| $\texttt{ins}^{\rightarrow}$ 2 real trees | 0.01036(5.25e-04) | 0.04909(1.33e-03) | 0.09305(1.96e-03) | 0.13257(2.26e-03) | 0.16931(2.67e-03) |
| $\texttt{ins}^{\rightarrow}$ 3 real trees | 0.01032(5.06e-04) | 0.04882(1.38e-03) | 0.09276(1.70e-03) | 0.13211(2.46e-03) | 0.16874(2.37e-03) |
| `del` a node weighting 0-500 | 0.01440(1.65e-03) | 0.06771(5.41e-03) | 0.12969(8.92e-03) | 0.18679(1.24e-02) | 0.24169(1.48e-02) |
| `del` a node weighting 0-10 | 0.01603(1.77e-03) | 0.07667(7.49e-03) | 0.14556(1.39e-02) | 0.20980(1.78e-02) | 0.26751(2.14e-02) |
| `del` a node weighting 10-25 | 0.01155(1.48e-03) | 0.05554(4.60e-03) | 0.10808(7.97e-03) | 0.15778(8.80e-03) | 0.20647(1.00e-02) |
| `del` a node weighting 25-500 | 0.01021(1.01e-03) | 0.05030(2.24e-03) | 0.09951(2.43e-03) | 0.14765(3.63e-03) | 0.19675(6.05e-03) |
| Random | 0.01859(3.47e-03) | 0.07947(6.93e-03) | 0.14489(1.17e-02) | 0.20319(1.21e-02) | 0.25369(1.51e-02) |
| `repV` with an already used value | 0.06350(4.46e-03) | 0.26836(1.55e-02) | 0.45024(2.02e-02) | 0.58043(2.07e-02) | 0.67752(1.81e-02) |
| `repV` with a random value | 0.06373(4.35e-03) | 0.27068(1.50e-02) | 0.45484(1.95e-02) | 0.58658(1.97e-02) | 0.68531(1.74e-02) |
| `repV` with that of a node with the same name | 0.06093(5.17e-03) | 0.24376(2.51e-02) | 0.39681(4.02e-02) | 0.50116(4.85e-02) | 0.57451(5.31e-02) |

**Table 1: *pq*-gram based change ratio estimation (Part 1)**

| Operation type | **0.3** | **0.4** | **0.5** | **0.6** | **0.7** |
|---|---|---|---|---|---|
| `ren` to any existing name | 0.85890(6.39e-03) | 0.92782(4.25e-03) | 0.97396(4.44e-03) | 0.99123(2.11e-03) | |
| `ren` to a random name | 0.86100(6.80e-03) | 0.93019(4.60e-03) | 0.97652(5.06e-03) | 0.99253(2.34e-03) | |
| `ren` to a name of a node at the same level | 0.84835(5.87e-03) | 0.91734(7.55e-03) | 0.96378(6.52e-03) | 0.98851(3.31e-03) | |
| `ren` to a name of a node with the same parent name | 0.84876(5.99e-03) | 0.91752(7.61e-03) | 0.96375(6.45e-03) | 0.98848(3.25e-03) | |
| `ren` to a name of a node at the same level +- 1 | 0.84874(5.88e-03) | 0.91749(7.42e-03) | 0.96420(6.41e-03) | 0.98856(3.23e-03) | |
| Move 1 tree (total weight 0–500) | 0.22297(4.42e-02) | 0.29618(4.99e-02) | 0.35883(5.11e-02) | 0.41165(5.38e-02) | 0.45939(4.81e-02) |
| Move 2 trees (total weight 0–500) | 0.12542(4.03e-02) | 0.16922(4.55e-02) | 0.21187(5.53e-02) | 0.25717(5.46e-02) | 0.29564(5.91e-02) |
| Move 3 trees (total weight 0–500) | 0.07630(2.49e-02) | 0.10425(2.93e-02) | 0.12864(3.42e-02) | 0.15775(3.36e-02) | 0.18557(3.31e-02) |
| Move 4 trees (total weight 0–500) | 0.04970(1.33e-02) | 0.06672(1.66e-02) | 0.08891(1.86e-02) | 0.10928(2.24e-02) | 0.12919(2.47e-02) |
| `repN` with a random tree of a similar weight | 0.34815(1.45e-02) | 0.42905(1.59e-02) | 0.49813(1.61e-02) | 0.55772(1.60e-02) | 0.60926(1.54e-02) |
| `repN` with 1 random tree weighting 1–10 | 0.35298(1.48e-02) | 0.43261(1.54e-02) | 0.50020(1.50e-02) | 0.55768(1.49e-02) | 0.60622(1.42e-02) |
| `repN` with 1 random tree weighting 10–25 | 0.33867(1.52e-02) | 0.41159(1.61e-02) | 0.47201(1.64e-02) | 0.52354(1.59e-02) | 0.56751(1.56e-02) |
| `repN` with 1 random tree weighting 25–50 | 0.33911(1.56e-02) | 0.40956(1.66e-02) | 0.46754(1.65e-02) | 0.51616(1.65e-02) | 0.55774(1.60e-02) |
| `repN` with 1 random tree weighting 50–500 | 0.33457(1.57e-02) | 0.40207(1.67e-02) | 0.45731(1.75e-02) | 0.50321(1.71e-02) | 0.54160(1.72e-02) |
| `repN` with 2 random trees (total weight 50–100) | 0.33675(1.55e-02) | 0.40540(1.65e-02) | 0.46160(1.68e-02) | 0.50868(1.72e-02) | 0.54855(1.70e-02) |
| `repN` with 3 random trees (total weight 75–150) | 0.33640(1.53e-02) | 0.40413(1.67e-02) | 0.46019(1.71e-02) | 0.50673(1.72e-02) | 0.54615(1.67e-02) |
| `repN` with a tree similar to one at the same level | 0.27282(1.83e-02) | 0.34084(1.95e-02) | 0.39432(2.49e-02) | 0.44392(2.73e-02) | 0.48466(3.21e-02) |
| `repN` with a tree similar to one of its siblings | 0.26829(2.08e-02) | 0.33420(2.36e-02) | 0.39096(2.79e-02) | 0.44133(2.94e-02) | 0.48443(3.03e-02) |
| `repN` with 1 real tree | 0.22367(1.10e-02) | 0.28181(1.33e-02) | 0.33399(1.48e-02) | 0.38269(1.76e-02) | 0.42749(1.78e-02) |
| `repN` with 2 real trees | 0.22189(8.29e-03) | 0.27708(1.01e-02) | 0.32588(1.13e-02) | 0.37037(1.23e-02) | 0.41073(1.38e-02) |
| `repN` with 3 real trees | 0.22271(6.83e-03) | 0.27781(8.36e-03) | 0.32637(8.54e-03) | 0.36910(9.40e-03) | 0.40876(9.78e-03) |
| $\texttt{ins}^{\rightarrow}$ 1 random tree (weight similar to a sibling) | 0.37017(2.20e-02) | 0.44441(2.39e-02) | 0.50467(2.52e-02) | 0.55494(2.57e-02) | 0.59686(2.65e-02) |
| $\texttt{ins}^{\rightarrow}$ 1 node weighting 1–10 | 0.37623(1.80e-02) | 0.45076(1.93e-02) | 0.51136(1.95e-02) | 0.56147(1.96e-02) | 0.60377(1.95e-02) |
| $\texttt{ins}^{\rightarrow}$ 1 node weighting 10–25 | 0.34611(1.67e-02) | 0.41571(1.80e-02) | 0.47244(1.88e-02) | 0.52000(1.88e-02) | 0.56017(1.88e-02) |
| $\texttt{ins}^{\rightarrow}$ 1 node weighting 25–50 | 0.34282(1.66e-02) | 0.41113(1.78e-02) | 0.46704(1.82e-02) | 0.51336(1.83e-02) | 0.55300(1.82e-02) |
| $\texttt{ins}^{\rightarrow}$ 1 node weighting 50–500 | 0.33493(1.59e-02) | 0.40191(1.70e-02) | 0.45675(1.73e-02) | 0.50251(1.76e-02) | 0.54083(1.77e-02) |
| $\texttt{ins}^{\rightarrow}$ 2 nodes (total weight 50-100) | 0.33833(1.58e-02) | 0.40598(1.71e-02) | 0.46103(1.78e-02) | 0.50679(1.80e-02) | 0.54596(1.77e-02) |
| $\texttt{ins}^{\rightarrow}$ 3 nodes (total weight 75-150) | 0.33763(1.58e-02) | 0.40477(1.73e-02) | 0.45957(1.73e-02) | 0.50537(1.74e-02) | 0.54415(1.75e-02) |
| $\texttt{ins}^{\rightarrow}$ 1 tree similar to one at the same level | 0.29135(1.94e-02) | 0.35328(2.04e-02) | 0.40411(2.32e-02) | 0.44722(2.57e-02) | 0.48604(2.45e-02) |
| $\texttt{ins}^{\rightarrow}$ 1 tree similar to one of its siblings | 0.28829(2.02e-02) | 0.34982(2.32e-02) | 0.40118(2.60e-02) | 0.44405(2.73e-02) | 0.48178(2.85e-02) |
| $\texttt{ins}^{\rightarrow}$ 1 real tree | 0.23280(4.28e-03) | 0.28743(5.46e-03) | 0.33456(6.37e-03) | 0.37515(6.82e-03) | 0.41153(7.48e-03) |
| $\texttt{ins}^{\rightarrow}$ 2 real trees | 0.23305(3.16e-03) | 0.28786(3.77e-03) | 0.33536(4.38e-03) | 0.37658(4.69e-03) | 0.41358(5.22e-03) |
| $\texttt{ins}^{\rightarrow}$ 3 real trees | 0.23298(2.87e-03) | 0.28777(3.35e-03) | 0.33521(3.68e-03) | 0.37681(3.86e-03) | 0.41353(4.01e-03) |
| `del` a node weighting 0-500 | 0.33967(1.70e-02) | 0.42840(1.80e-02) | 0.50627(1.97e-02) | 0.57778(1.79e-02) | 0.63974(1.92e-02) |
| `del` a node weighting 0-10 | 0.36996(2.56e-02) | 0.45844(2.77e-02) | 0.53499(2.77e-02) | 0.60248(2.69e-02) | 0.66210(2.49e-02) |
| `del` a node weighting 10-25 | 0.29869(8.60e-03) | 0.38670(7.32e-03) | 0.47362(9.14e-03) | 0.56707(1.25e-02) | 0.66594(1.96e-03) |
| `del` a node weighting 25-500 | 0.29348(8.27e-03) | 0.38246(1.26e-02) | 0.47224(1.52e-02) | 0.55574(2.07e-02) | 0.64196(2.36e-02) |
| Random | 0.33736(1.72e-02) | 0.40553(1.82e-02) | 0.46098(1.97e-02) | 0.50724(1.87e-02) | 0.54687(1.91e-02) |
| `repV` with an already used value | 0.81090(1.12e-02) | 0.89777(4.25e-03) | 1.02913(3.41e-02) | | |
| `repV` with a random value | 0.82164(1.03e-02) | 0.91058(4.61e-03) | 1.04513(3.64e-02) | | |
| `repV` with that of a node with the same name | 0.66485(5.60e-02) | 0.71073(5.50e-02) | 0.79642(3.97e-02) | | |

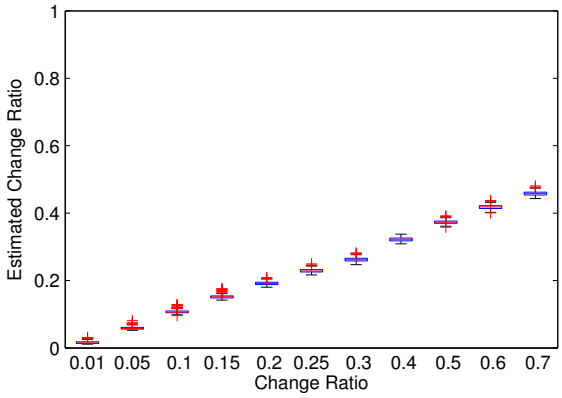
**Table 2: *pq*-gram based change ratio estimation (Part 2)**

| Operation type | **0.01** | **0.05** | **0.1** | **0.15** | **0.2** |
|---|---|---|---|---|---|
| `ren` to any existing name | 0.02703(2.21e-03) | 0.11585(7.70e-03) | 0.19953(1.40e-02) | 0.26053(2.12e-02) | 0.30689(2.74e-02) |
| `ren` to a random name | 0.02962(1.83e-03) | 0.13001(3.73e-03) | 0.22504(3.86e-03) | 0.29661(2.97e-03) | 0.35148(2.64e-03) |
| `ren` to a name of a node at the same level | 0.02552(2.73e-03) | 0.10607(9.85e-03) | 0.18145(1.84e-02) | 0.23693(2.57e-02) | 0.27769(3.19e-02) |
| `ren` to a name of a node with the same parent name | 0.02535(2.55e-03) | 0.10634(1.05e-02) | 0.18186(1.85e-02) | 0.23696(2.62e-02) | 0.27800(3.18e-02) |
| `ren` to a name of a node at the same level +- 1 | 0.02586(2.67e-03) | 0.10876(9.28e-03) | 0.18630(1.75e-02) | 0.24373(2.38e-02) | 0.28632(2.89e-02) |
| Move 1 tree (total weight 0–500) | 0.00972(1.52e-03) | 0.04567(4.01e-03) | 0.08798(6.43e-03) | 0.12969(6.37e-03) | 0.16750(6.98e-03) |
| Move 2 trees (total weight 0–500) | 0.00837(1.46e-03) | 0.04059(4.87e-03) | 0.07798(8.30e-03) | 0.11445(1.03e-02) | 0.14729(1.18e-02) |
| Move 3 trees (total weight 0–500) | 0.00736(2.21e-03) | 0.03718(7.27e-03) | 0.07522(1.17e-02) | 0.11132(1.10e-02) | 0.14386(1.50e-02) |
| Move 4 trees (total weight 0–500) | 0.00741(2.50e-03) | 0.03951(6.47e-03) | 0.07601(1.18e-02) | 0.11162(1.29e-02) | 0.14442(1.61e-02) |
| `repN` with a random tree of a similar weight | 0.01283(7.74e-04) | 0.06049(2.86e-03) | 0.11565(4.92e-03) | 0.16632(6.58e-03) | 0.21296(7.72e-03) |
| `repN` with 1 random tree weighting 1–10 | 0.01296(8.84e-04) | 0.06170(3.48e-03) | 0.11752(5.49e-03) | 0.16932(7.37e-03) | 0.21667(8.66e-03) |
| `repN` with 1 random tree weighting 10–25 | 0.01332(7.00e-04) | 0.06259(3.04e-03) | 0.11838(5.11e-03) | 0.16862(6.61e-03) | 0.21404(7.78e-03) |
| `repN` with 1 random tree weighting 25–50 | 0.01368(7.17e-04) | 0.06393(2.77e-03) | 0.12062(4.80e-03) | 0.17119(6.25e-03) | 0.21660(7.60e-03) |
| `repN` with 1 random tree weighting 50–500 | 0.01372(6.13e-04) | 0.06468(2.64e-03) | 0.12154(4.55e-03) | 0.17200(5.57e-03) | 0.21700(6.84e-03) |
| `repN` with 2 random trees (total weight 50–100) | 0.01374(5.94e-04) | 0.06442(2.58e-03) | 0.12116(4.69e-03) | 0.17158(5.80e-03) | 0.21669(6.98e-03) |
| `repN` with 3 random trees (total weight 75–150) | 0.01374(5.90e-04) | 0.06461(2.51e-03) | 0.12145(4.41e-03) | 0.17186(5.73e-03) | 0.21706(6.92e-03) |
| `repN` with a tree similar to one at the same level | 0.01016(9.03e-04) | 0.04487(2.55e-03) | 0.08287(5.06e-03) | 0.11792(6.97e-03) | 0.15017(9.00e-03) |
| `repN` with a tree similar to one of its siblings | 0.00952(1.13e-03) | 0.04154(3.36e-03) | 0.07743(7.04e-03) | 0.11200(8.36e-03) | 0.14193(9.64e-03) |
| `repN` with 1 real tree | 0.00822(9.79e-04) | 0.03657(4.41e-03) | 0.06778(9.24e-03) | 0.09607(1.44e-02) | 0.12228(1.80e-02) |
| `repN` with 2 real trees | 0.00837(8.06e-04) | 0.03850(3.84e-03) | 0.07079(6.32e-03) | 0.10313(1.04e-02) | 0.12854(1.17e-02) |
| `repN` with 3 real trees | 0.00872(5.65e-04) | 0.04038(3.23e-03) | 0.07516(5.68e-03) | 0.10802(8.20e-03) | 0.13691(9.67e-03) |
| $\texttt{ins}^{\rightarrow}$ 1 random tree (weight similar to a sibling) | 0.01463(7.42e-04) | 0.06859(3.47e-03) | 0.12891(6.03e-03) | 0.18249(8.39e-03) | 0.23062(1.00e-02) |
| $\texttt{ins}^{\rightarrow}$ 1 node weighting 1–10 | 0.01470(7.66e-04) | 0.06971(3.28e-03) | 0.13087(5.74e-03) | 0.18519(7.64e-03) | 0.23361(9.08e-03) |
| $\texttt{ins}^{\rightarrow}$ 1 node weighting 10–25 | 0.01395(6.80e-04) | 0.06602(2.88e-03) | 0.12414(4.87e-03) | 0.17567(6.50e-03) | 0.22176(7.63e-03) |
| $\texttt{ins}^{\rightarrow}$ 1 node weighting 25–50 | 0.01395(6.55e-04) | 0.06600(2.60e-03) | 0.12385(4.64e-03) | 0.17523(6.24e-03) | 0.22080(7.27e-03) |
| $\texttt{ins}^{\rightarrow}$ 1 node weighting 50–500 | 0.01392(5.96e-04) | 0.06514(2.68e-03) | 0.12212(4.29e-03) | 0.17280(5.74e-03) | 0.21803(6.73e-03) |
| $\texttt{ins}^{\rightarrow}$ 2 nodes (total weight 50-100) | 0.01384(6.42e-04) | 0.06540(2.60e-03) | 0.12289(4.43e-03) | 0.17374(5.83e-03) | 0.21900(6.91e-03) |
| $\texttt{ins}^{\rightarrow}$ 3 nodes (total weight 75-150) | 0.01393(5.43e-04) | 0.06529(2.53e-03) | 0.12283(4.37e-03) | 0.17336(5.62e-03) | 0.21894(6.49e-03) |
| $\texttt{ins}^{\rightarrow}$ 1 tree similar to one at the same level | 0.01201(8.83e-04) | 0.05402(2.57e-03) | 0.10068(4.23e-03) | 0.14195(5.33e-03) | 0.17957(6.29e-03) |
| $\texttt{ins}^{\rightarrow}$ 1 tree similar to one of its siblings | 0.01149(9.01e-04) | 0.05237(2.80e-03) | 0.09776(4.83e-03) | 0.13829(6.00e-03) | 0.17577(7.73e-03) |
| $\texttt{ins}^{\rightarrow}$ 1 real tree | 0.00990(2.51e-04) | 0.04716(6.80e-04) | 0.08979(1.35e-03) | 0.12853(1.85e-03) | 0.16400(2.26e-03) |
| $\texttt{ins}^{\rightarrow}$ 2 real trees | 0.00991(2.43e-04) | 0.04755(6.66e-04) | 0.09065(1.11e-03) | 0.12972(1.25e-03) | 0.16595(1.69e-03) |
| $\texttt{ins}^{\rightarrow}$ 3 real trees | 0.00996(2.27e-04) | 0.04768(7.58e-04) | 0.09098(9.95e-04) | 0.13017(1.43e-03) | 0.16637(1.44e-03) |
| `del` a node weighting 0-500 | 0.01182(8.47e-04) | 0.05653(2.51e-03) | 0.10930(3.81e-03) | 0.15914(5.43e-03) | 0.20735(6.52e-03) |
| `del` a node weighting 0-10 | 0.01271(9.34e-04) | 0.06082(3.66e-03) | 0.11657(6.54e-03) | 0.16925(8.55e-03) | 0.21842(1.01e-02) |
| `del` a node weighting 10-25 | 0.01040(7.90e-04) | 0.05084(2.51e-03) | 0.10000(4.06e-03) | 0.14778(4.12e-03) | 0.19485(4.60e-03) |
| `del` a node weighting 25-500 | 0.00989(5.63e-04) | 0.04908(1.38e-03) | 0.09771(1.99e-03) | 0.14565(3.13e-03) | 0.19442(5.23e-03) |
| Random | 0.01396(1.33e-03) | 0.06478(3.22e-03) | 0.12153(4.88e-03) | 0.17187(6.10e-03) | 0.21710(7.09e-03) |
| `repV` with an already used value | 0.03269(1.78e-03) | 0.13482(6.87e-03) | 0.22861(1.01e-02) | 0.29964(1.33e-02) | 0.35422(1.81e-02) |
| `repV` with a random value | 0.03500(1.26e-03) | 0.15482(3.66e-03) | 0.27196(6.28e-03) | 0.36428(1.03e-02) | 0.43948(1.53e-02) |
| `repV` with that of a node with the same name | 0.03253(2.19e-03) | 0.13137(1.22e-02) | 0.21927(2.00e-02) | 0.28385(2.52e-02) | 0.33217(2.86e-02) |

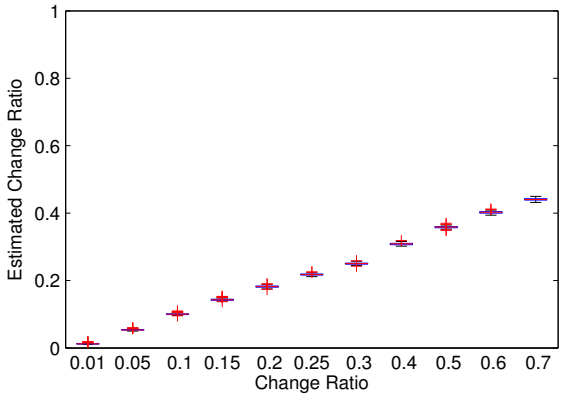
**Table 3: *pql*-gram based change ratio estimation (Part 1)**

| Operation type | **0.3** | **0.4** | **0.5** | **0.6** | **0.7** |
|---|---|---|---|---|---|
| `ren` to any existing name | 0.36634(3.90e-02) | 0.39975(4.91e-02) | 0.41873(5.70e-02) | 0.46175(4.97e-02) | |
| `ren` to a random name | 0.42699(4.78e-03) | 0.47521(5.96e-03) | 0.50900(5.40e-03) | 0.56874(1.21e-02) | |
| `ren` to a name of a node at the same level | 0.32971(4.18e-02) | 0.35754(4.93e-02) | 0.37213(5.43e-02) | 0.40802(4.62e-02) | |
| `ren` to a name of a node with the same parent name | 0.33005(4.21e-02) | 0.35764(4.96e-02) | 0.37209(5.43e-02) | 0.40840(4.61e-02) | |
| `ren` to a name of a node at the same level +- 1 | 0.34226(3.80e-02) | 0.37371(4.35e-02) | 0.39164(4.68e-02) | 0.43161(3.69e-02) | |
| Move 1 tree (total weight 0–500) | 0.23706(8.03e-03) | 0.29957(9.02e-03) | 0.35228(1.06e-02) | 0.39792(1.43e-02) | 0.44107(2.23e-02) |
| Move 2 trees (total weight 0–500) | 0.21178(1.23e-02) | 0.27167(1.17e-02) | 0.32201(1.53e-02) | 0.37107(1.55e-02) | 0.41539(1.68e-02) |
| Move 3 trees (total weight 0–500) | 0.20611(1.66e-02) | 0.25871(2.26e-02) | 0.31484(2.01e-02) | 0.36302(2.01e-02) | 0.40643(2.06e-02) |
| Move 4 trees (total weight 0–500) | 0.20879(2.11e-02) | 0.26571(2.13e-02) | 0.31650(2.16e-02) | 0.36709(1.99e-02) | 0.41515(2.09e-02) |
| `repN` with a random tree of a similar weight | 0.29697(9.03e-03) | 0.36985(9.51e-03) | 0.43328(9.36e-03) | 0.48912(8.74e-03) | 0.53827(8.04e-03) |
| `repN` with 1 random tree weighting 1–10 | 0.30137(9.32e-03) | 0.37368(9.35e-03) | 0.43628(8.95e-03) | 0.49091(8.56e-03) | 0.53811(7.79e-03) |
| `repN` with 1 random tree weighting 10–25 | 0.29349(8.93e-03) | 0.36038(9.55e-03) | 0.41740(9.49e-03) | 0.46659(9.32e-03) | 0.50931(9.21e-03) |
| `repN` with 1 random tree weighting 25–50 | 0.29483(8.67e-03) | 0.36021(9.43e-03) | 0.41517(9.69e-03) | 0.46226(9.65e-03) | 0.50312(9.45e-03) |
| `repN` with 1 random tree weighting 50–500 | 0.29377(8.10e-03) | 0.35719(8.89e-03) | 0.41021(9.57e-03) | 0.45536(9.36e-03) | 0.49376(9.58e-03) |
| `repN` with 2 random trees (total weight 50–100) | 0.29405(8.29e-03) | 0.35821(8.95e-03) | 0.41208(9.25e-03) | 0.45784(9.64e-03) | 0.49739(9.59e-03) |
| `repN` with 3 random trees (total weight 75–150) | 0.29445(8.00e-03) | 0.35790(8.96e-03) | 0.41153(9.29e-03) | 0.45704(9.46e-03) | 0.49607(9.23e-03) |
| `repN` with a tree similar to one at the same level | 0.20815(1.18e-02) | 0.25961(1.30e-02) | 0.30204(1.50e-02) | 0.34251(1.70e-02) | 0.37776(1.75e-02) |
| `repN` with a tree similar to one of its siblings | 0.19952(1.34e-02) | 0.24990(1.49e-02) | 0.29328(1.74e-02) | 0.33361(1.78e-02) | 0.36927(1.78e-02) |
| `repN` with 1 real tree | 0.16832(2.41e-02) | 0.21269(3.25e-02) | 0.25065(3.85e-02) | 0.28616(4.43e-02) | 0.32025(5.12e-02) |
| `repN` with 2 real trees | 0.17891(1.65e-02) | 0.22290(2.09e-02) | 0.26299(2.29e-02) | 0.29942(2.67e-02) | 0.33276(2.88e-02) |
| `repN` with 3 real trees | 0.18984(1.32e-02) | 0.23700(1.57e-02) | 0.27815(1.58e-02) | 0.31461(1.80e-02) | 0.34966(1.93e-02) |
| $\texttt{ins}^{\rightarrow}$ 1 random tree (weight similar to a sibling) | 0.31225(1.30e-02) | 0.37998(1.45e-02) | 0.43654(1.57e-02) | 0.48467(1.63e-02) | 0.52575(1.71e-02) |
| $\texttt{ins}^{\rightarrow}$ 1 node weighting 1–10 | 0.31615(1.11e-02) | 0.38418(1.23e-02) | 0.44108(1.28e-02) | 0.48930(1.32e-02) | 0.53062(1.34e-02) |
| $\texttt{ins}^{\rightarrow}$ 1 node weighting 10–25 | 0.30040(9.48e-03) | 0.36524(1.05e-02) | 0.41943(1.12e-02) | 0.46563(1.15e-02) | 0.50536(1.16e-02) |
| $\texttt{ins}^{\rightarrow}$ 1 node weighting 25–50 | 0.29878(9.03e-03) | 0.36281(9.91e-03) | 0.41647(1.03e-02) | 0.46189(1.06e-02) | 0.50114(1.08e-02) |
| $\texttt{ins}^{\rightarrow}$ 1 node weighting 50–500 | 0.29446(8.32e-03) | 0.35775(9.07e-03) | 0.41064(9.37e-03) | 0.45550(9.80e-03) | 0.49387(1.00e-02) |
| $\texttt{ins}^{\rightarrow}$ 2 nodes (total weight 50-100) | 0.29634(8.38e-03) | 0.36000(9.24e-03) | 0.41307(9.90e-03) | 0.45801(1.02e-02) | 0.49693(1.01e-02) |
| $\texttt{ins}^{\rightarrow}$ 3 nodes (total weight 75-150) | 0.29610(8.28e-03) | 0.35936(9.34e-03) | 0.41228(9.44e-03) | 0.45735(9.63e-03) | 0.49596(9.85e-03) |
| $\texttt{ins}^{\rightarrow}$ 1 tree similar to one at the same level | 0.24477(7.99e-03) | 0.29925(8.72e-03) | 0.34641(8.69e-03) | 0.38724(9.28e-03) | 0.42305(8.76e-03) |
| $\texttt{ins}^{\rightarrow}$ 1 tree similar to one of its siblings | 0.24058(8.20e-03) | 0.29555(8.98e-03) | 0.34243(9.75e-03) | 0.38279(9.73e-03) | 0.41901(9.70e-03) |
| $\texttt{ins}^{\rightarrow}$ 1 real tree | 0.22685(3.14e-03) | 0.28107(4.03e-03) | 0.32806(4.65e-03) | 0.36900(4.95e-03) | 0.40546(5.34e-03) |
| $\texttt{ins}^{\rightarrow}$ 2 real trees | 0.22938(2.09e-03) | 0.28395(2.51e-03) | 0.33130(2.91e-03) | 0.37264(3.06e-03) | 0.40945(3.45e-03) |
| $\texttt{ins}^{\rightarrow}$ 3 real trees | 0.23023(1.81e-03) | 0.28496(2.20e-03) | 0.33232(2.39e-03) | 0.37395(2.55e-03) | 0.41063(2.69e-03) |
| `del` a node weighting 0-500 | 0.29671(7.92e-03) | 0.37940(8.10e-03) | 0.45514(1.00e-02) | 0.52617(9.78e-03) | 0.59011(1.11e-02) |
| `del` a node weighting 0-10 | 0.30829(1.20e-02) | 0.38909(1.35e-02) | 0.46257(1.37e-02) | 0.52964(1.37e-02) | 0.59083(1.36e-02) |
| `del` a node weighting 10-25 | 0.28568(4.96e-03) | 0.37394(9.70e-03) | 0.46105(1.58e-02) | 0.55907(1.93e-02) | 0.66354(7.79e-04) |
| `del` a node weighting 25-500 | 0.29032(7.75e-03) | 0.37949(1.20e-02) | 0.46930(1.48e-02) | 0.55299(2.03e-02) | 0.63935(2.35e-02) |
| Random | 0.29402(8.51e-03) | 0.35766(9.33e-03) | 0.41103(9.81e-03) | 0.45633(9.86e-03) | 0.49566(9.84e-03) |
| `repV` with an already used value | 0.43111(3.02e-02) | 0.47940(4.54e-02) | 0.53043(6.53e-02) | | |
| `repV` with a random value | 0.55525(2.55e-02) | 0.64054(3.54e-02) | 0.73944(5.86e-02) | | |
| `repV` with that of a node with the same name | 0.39680(3.23e-02) | 0.43376(3.50e-02) | 0.47774(3.23e-02) | | |

**Table 4: *pql*-gram based change ratio estimation (Part 2)**




(a) Random (*pq*-grams)




(b) Random (*pql*-grams)

**Figure 4: Tree PUL edit-distance estimation with *pq*-grams and *pql*-grams (Part 4)**

| | *pq*-gram estimation | | *pql*-gram estimation | |
|---|---|---|---|---|
| Change ratio | Mean | Standard dev. | Mean | Standard dev. |
| 0.01 | 0.029572 | 0.026947 | 0.016702 | 0.008945 |
| 0.05 | 0.124398 | 0.107564 | 0.074867 | 0.035163 |
| 0.10 | 0.213398 | 0.169629 | 0.135634 | 0.056661 |
| 0.20 | 0.338368 | 0.231525 | 0.231765 | 0.078262 |
| 0.30 | 0.425672 | 0.255584 | 0.305835 | 0.086130 |
| 0.40 | 0.492553 | 0.263406 | 0.365108 | 0.089523 |
| 0.50 | 0.556594 | 0.278110 | 0.418311 | 0.100024 |

**Table 5: PUL change ratio estimation summary, random PUL.**

by these algorithms can be contrasted with ours without unfair advantages. Specifically, w.r.t. our operation cost model, the only relevant differences are that the `repC` operation exist only in PUL-Diff and that a move operation exists only in XyDiff. Since in this test we disabled the generation of `repC` operations, and since XyDiff devotes a great effort to minimize the number of move operations, we just consider the cost of XyDiff move operations as the cost of the equivalent deletion and insertion operations. Moreover, we double checked that the XyDiff edit-script costs are fair by contrasting the presented results with the results obtained disabling the generation of moves in the PULs used to produce the target documents, and employing only randomly generated name, values and inserted/replacement subtrees to almost avoid the presence of move operations in the XyDiff edit-scripts. Since the results are extremely similar we consider the weight of XyDiff edit-scripts fair.

We report in Figure 9 the cost difference between a minimum-cost edit-script (the one used to generate the target document) and the cost of the edit-script generated by PUL-Diff and those generated by XyDiff and DeltaXML. We consider different change ratios and document sizes. As can be observed in the figure, independently from the change ratio, both PUL-Diff and DeltaXML produce almost minimal edit-scripts, with a slight advantage for PUL-Diff. XyDiff, instead, produces far worse results. With very small documents (up to 1MB) the quality is comparable with that of the other two algorithms, whereas usually the generated edit-script is far from optimal. We believe that this result does not depend on the XyDiff algorithm implementation as [3, 2] consider small documents found on the web and report that the cost of the edit-script identified by XyDiff can easily be 5 times more expensive than the considered reference (on average 2 times more expensive).

In Figure 10, we contrast also the time required for differencing two documents with different size and change ratios. We can observe that XyDiff is roughly 2 to 5 times slower than PUL-Diff and that both algorithms have an almost linear time complexity. As can be expected, the change ratio and the two document sizes influences the computational time. For what concerns PUL-Diff, as the change ratio increases, the number of perfectly matched subtrees decreases, thus increasing the time spent in the bottom-up and top-down refinement stages. Roughly we can observe that both XyDiff and PUL-Diff are three times slower when the change ratio is 0.9 w.r.t. their results with change ratio 0.01. The result also show that the excellent quality of the DeltaXML edit-scripts is counterbalanced by the computational time required by the algorithm, which is affected by the change ratio between the two documents. Indeed, with change ratio 0.01 DeltaXML is only marginally slower than PUL-Diff, whereas with change ratio 0.1 it is as slow as XyDiff. With higher change ratios the time required by DeltaXML becomes exponential in the size of the considered documents.

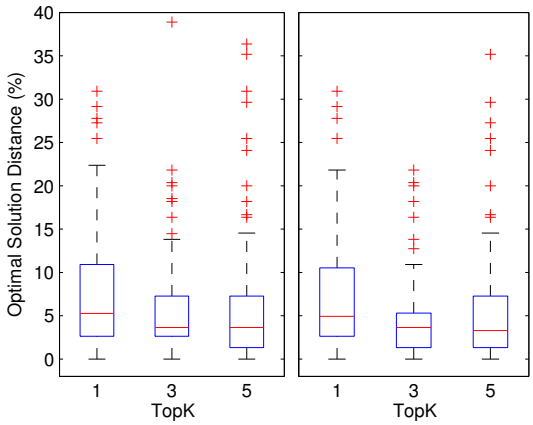
## 2. REFERENCES

[1] S. S. Chawathe. Comparing Hierarchical Data in External Memory. In *VLDB*, pages 90–101, 1999.

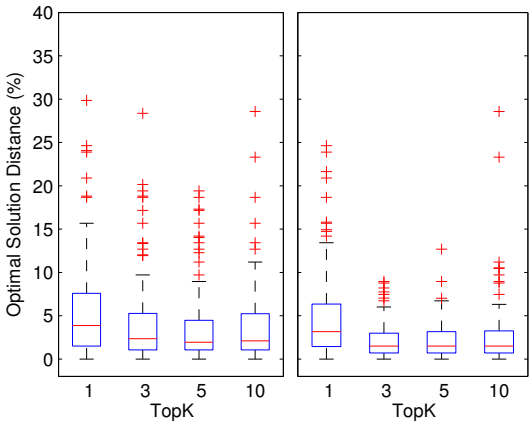
[2] G. Cobena, T. Abdessalem, and Y. Hinnach. A Comparative Study for XML Change Detection. In *BDA*, 2002.

[3] G. Cobena, S. Abiteboul, and A. Marian. Detecting Changes in XML Documents. In *ICDE*, pages 41–52, 2002.

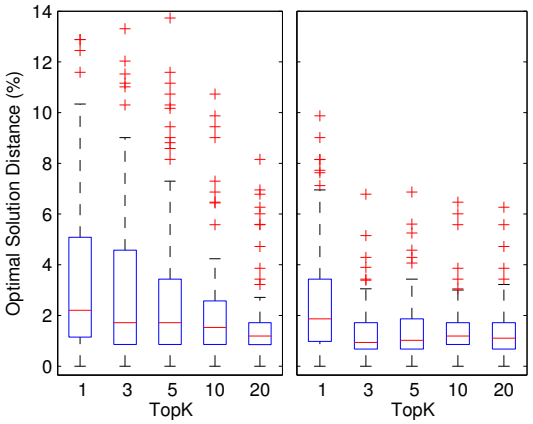
[4] R. L. Fontaine. A Delta Format for XML: Identifying Changes in XML Files and Representing the Changes in XML. XML Europe, 2001.



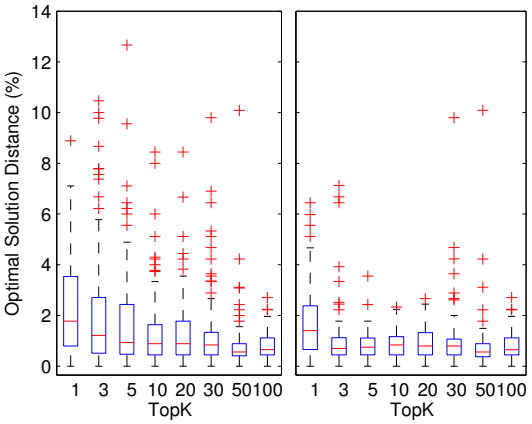

(a) Sequence length 10



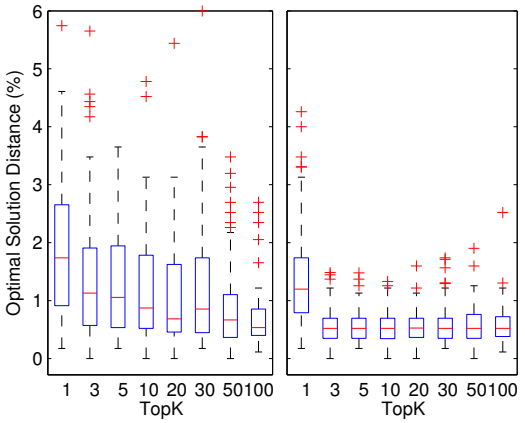

(b) Sequence length 25



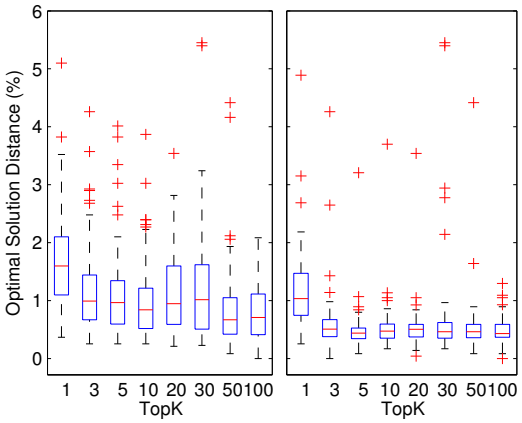

(c) Sequence length 50




(d) Sequence length 100

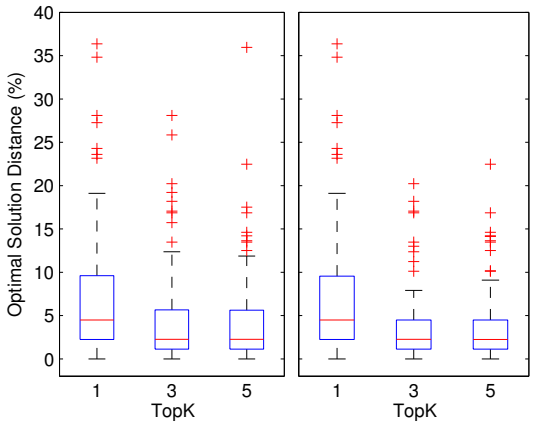



(e) Sequence length 250

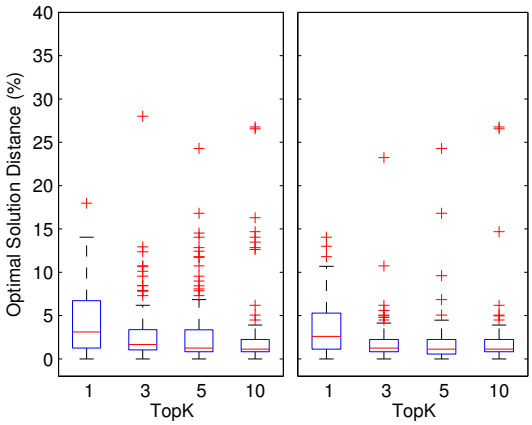



(f) Sequence length 500

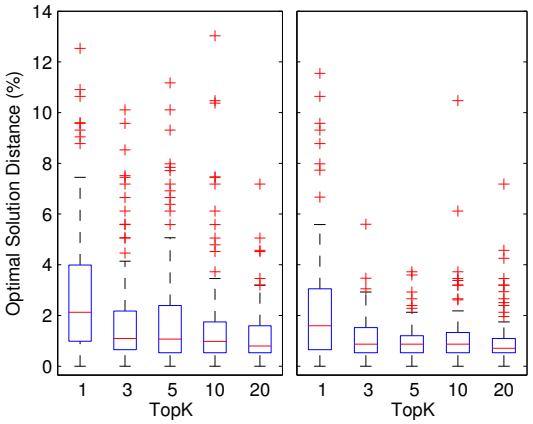
**Figure 5: Sequence matching with change ratio 0.3, cost distance from an optimal solution**



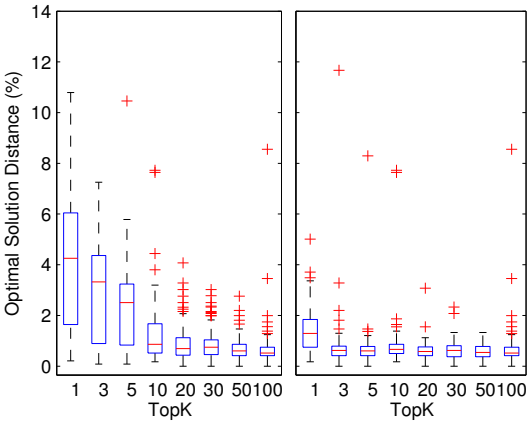

(a) Sequence length 10



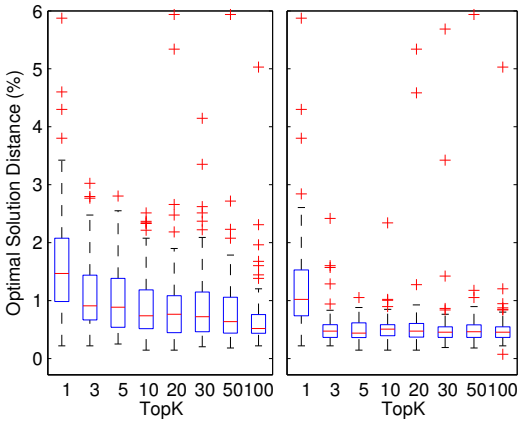

(b) Sequence length 25



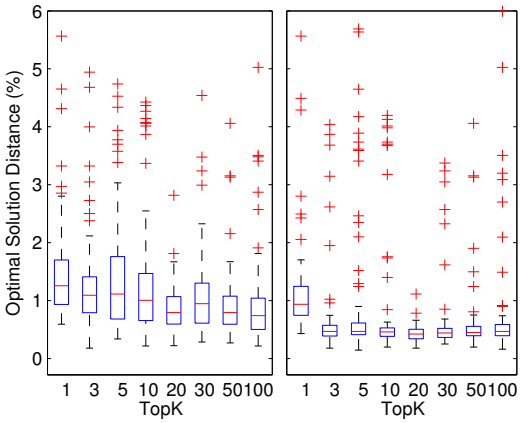

(c) Sequence length 50




(d) Sequence length 100

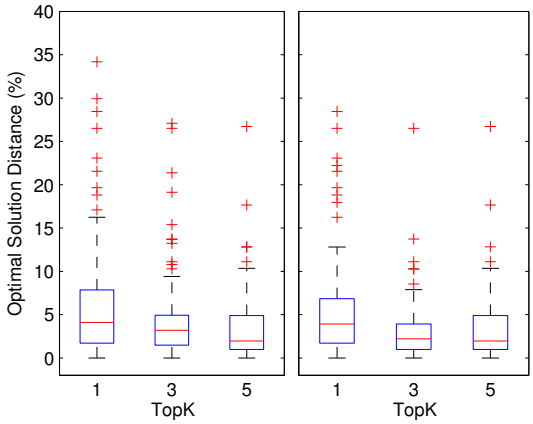



(e) Sequence length 250

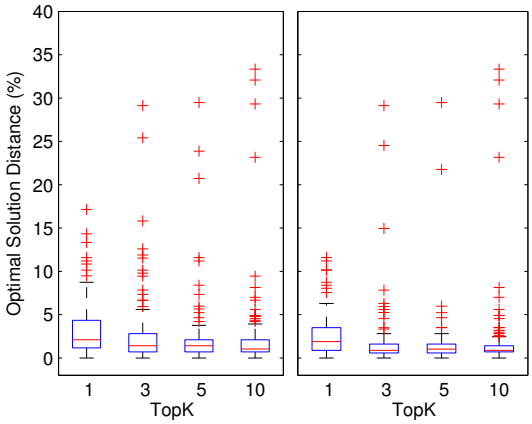



(f) Sequence length 500

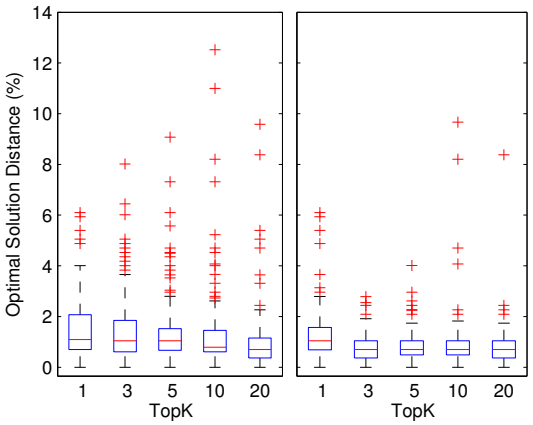
**Figure 6: Sequence matching with change ratio 0.5, cost distance from an optimal solution**



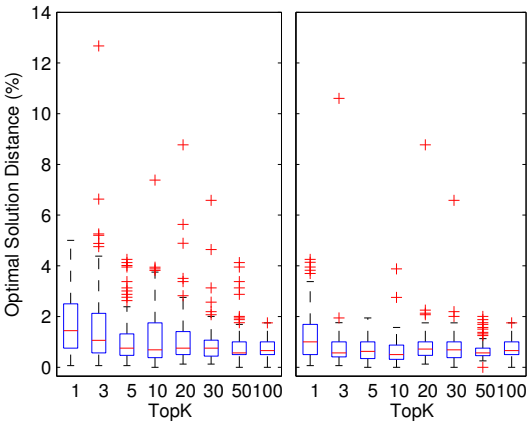

(a) Sequence length 10



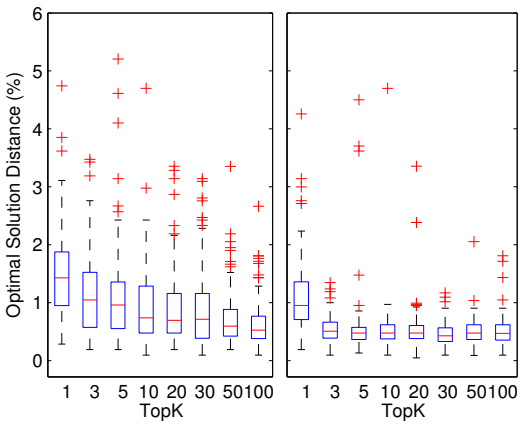

(b) Sequence length 25



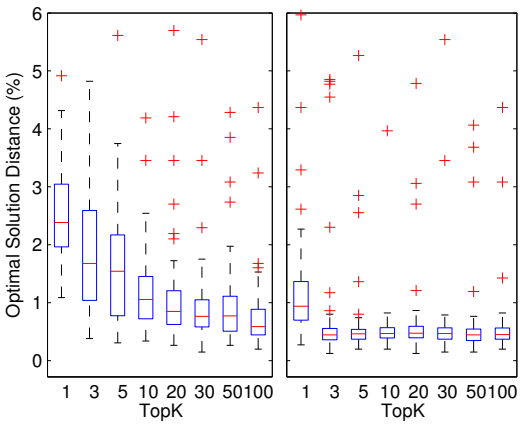

(c) Sequence length 50




(d) Sequence length 100

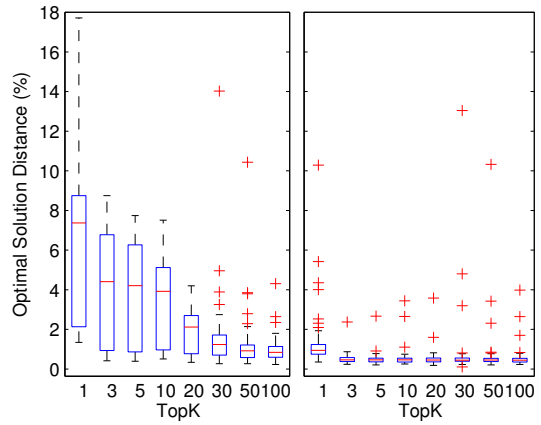



(e) Sequence length 250

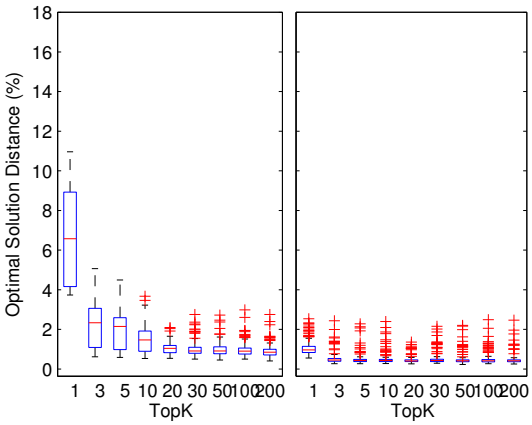



(f) Sequence length 500

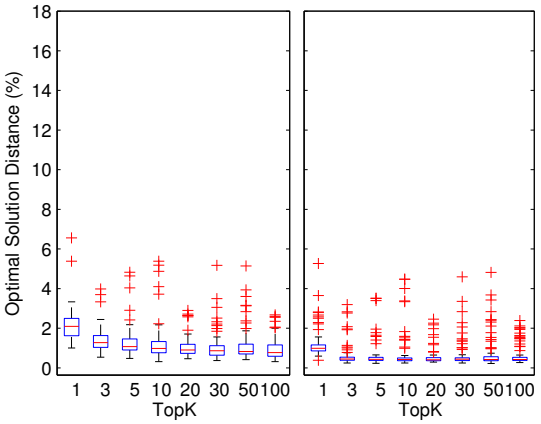
**Figure 7: Sequence matching with change ratio 0.7, cost distance from an optimal solution**



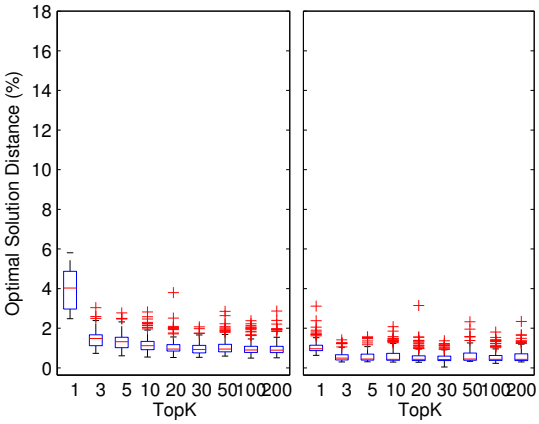

(a) Sequence length 1000, Change Ratio 0.3



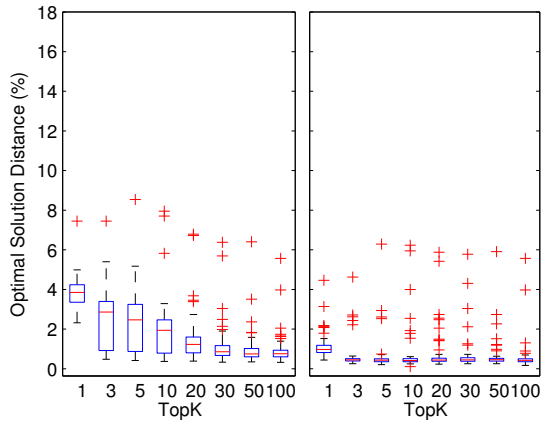

(b) Sequence length 5000, Change Ratio 0.3



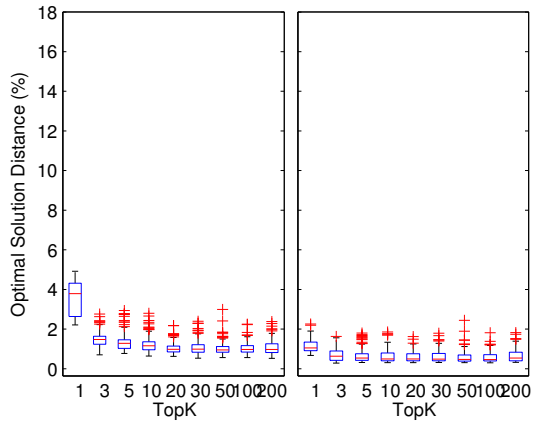

(c) Sequence length 1000, Change Ratio 0.5




(d) Sequence length 5000, Change Ratio 0.5

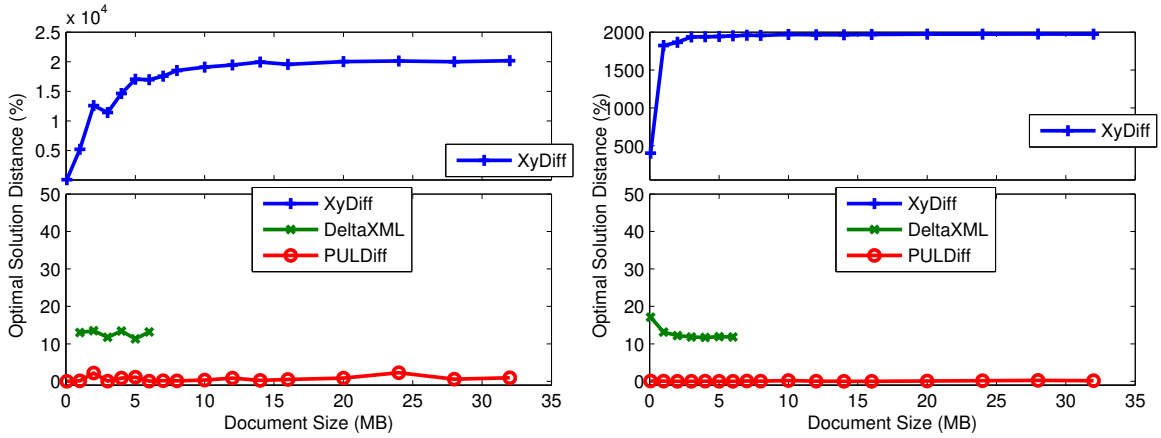



(e) Sequence length 1000, Change Ratio 0.7




(f) Sequence length 5000, Change Ratio 0.7

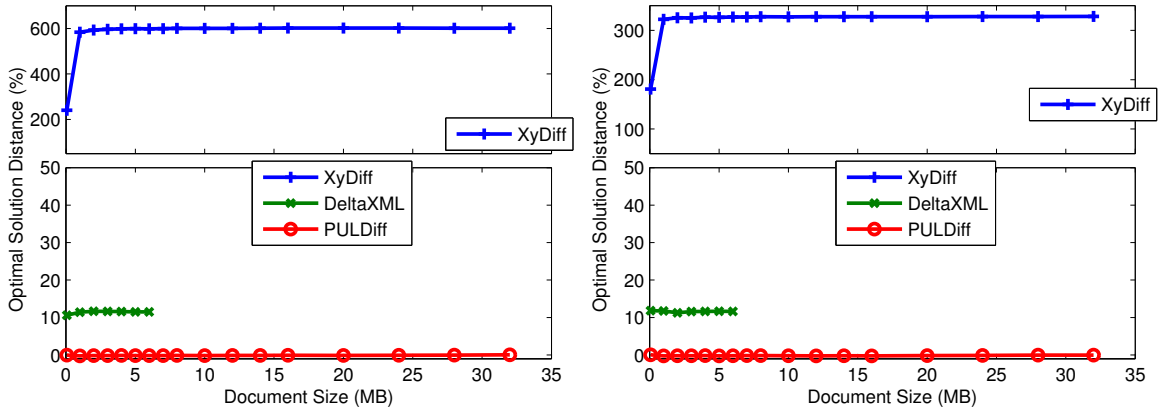
**Figure 8: Sequence matching for very long sequences. Cost distance from an optimal solution**




(a) Change ratio 0.01



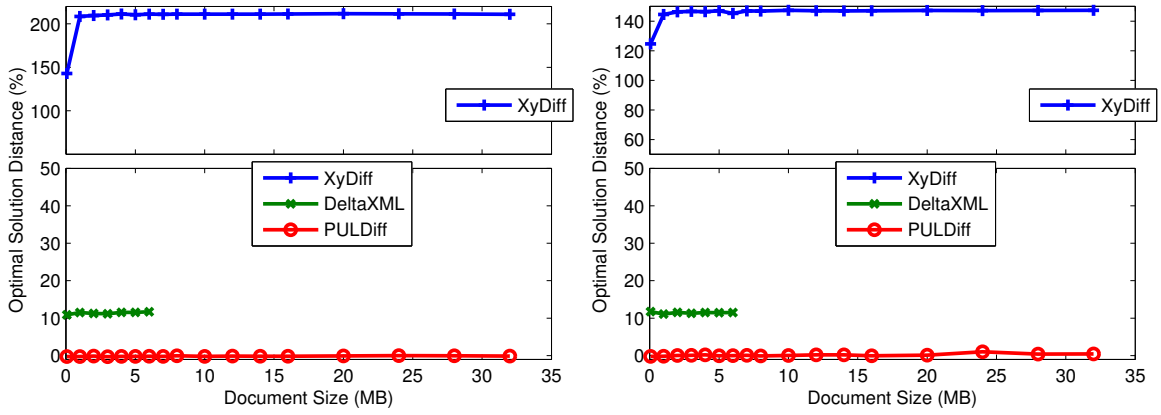

(b) Change ratio 0.1




(c) Change ratio 0.3




(d) Change ratio 0.5

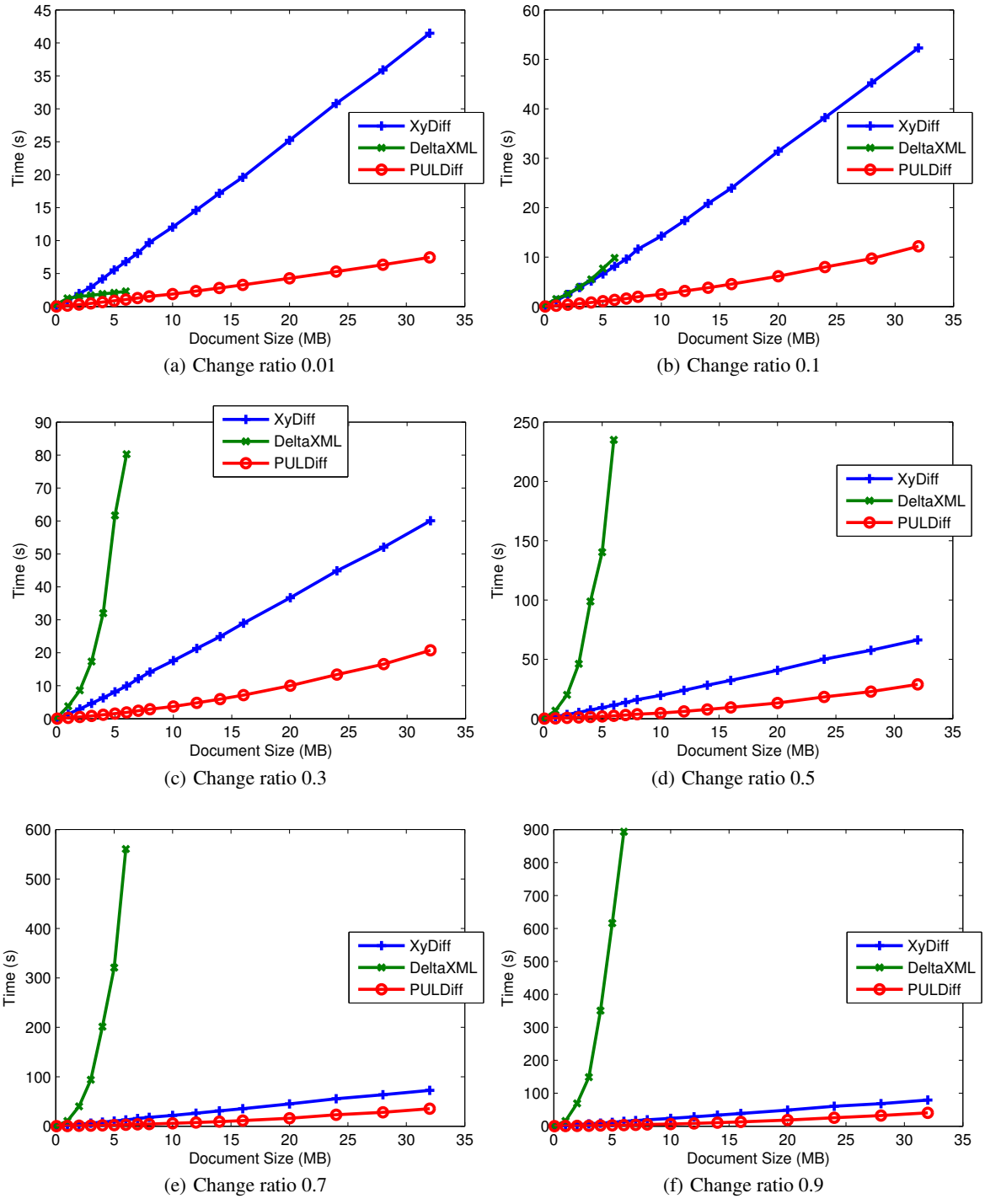



(e) Change ratio 0.7




(f) Change ratio 0.9

**Figure 9: Cost distance from an optimal solution**




(a) Change ratio 0.01

(b) Change ratio 0.1

(c) Change ratio 0.3

(d) Change ratio 0.5

(e) Change ratio 0.7

(f) Change ratio 0.9

**Figure 10: Differencing time and document size**