The Cost of Stability

Evaluating Stable Tree Differencing with Gumtree and HyperDiff

1 Introduction

Abstract Syntax Tree (AST) Differencing

- How to find differences between two ASTs T_1 and T_2?
  1. **Create mappings between similar nodes;**
  2. Compute required actions to get from T_1 to T_2 (insert, delete, update, move);

Gumtree Greedy Algorithm

1. **Top-down phase:** matches equal subtrees;
2. **Bottom-up phase:** matches each node in T_1 to best candidate node in T_2;
3. **Recovery:** after every bottom-up match, runs optimal algorithm on matched subtrees up to $maxSize$ to match missed nodes;

Stability

- Requires **diff(x, y) = diff(y, x)**;
- Provides reversibility and consistency in tools like Git;
- **Stable bottom-up phase:** *only* matches nodes in T_1 to node in T_2 if *both* are best candidates of each other;

HyperDiff

- Diff framework that leverages **HyperAST**s: novel data structure that **deduplicates repeated subtrees** across and within versions;
- Allows for faster **lazy evaluation** of HyperASTs;
- Gumtree Greedy is implemented, Gumtree Stable is not;

2 Research Questions

1. What are the **trade-offs** between **stability** and **performance** when using **Gumtree Stable**?
2. How does **lazy evaluation** made possible by HyperDiff affect the **performance** of **Gumtree Stable** and **Greedy**?

3 Methodology

- **Implement Gumtree Stable & Lazy Stable** in HyperDiff using reference Java implementation;
- **Benchmark** Gumtree variants on 1000+ Java file pairs with various values of $maxSize$;
- **Compare results** using various **pairwise statistics**:
  - **Runtime Δ:** mean & median (B – A);
  - **RBC:** Effect size; -1 = A faster, +1 = B faster;
  - **Log-ratio:** Mean speedup (%); >0 = A faster;
  - **Wilcoxon signed-rank test:** significance;

4 Results

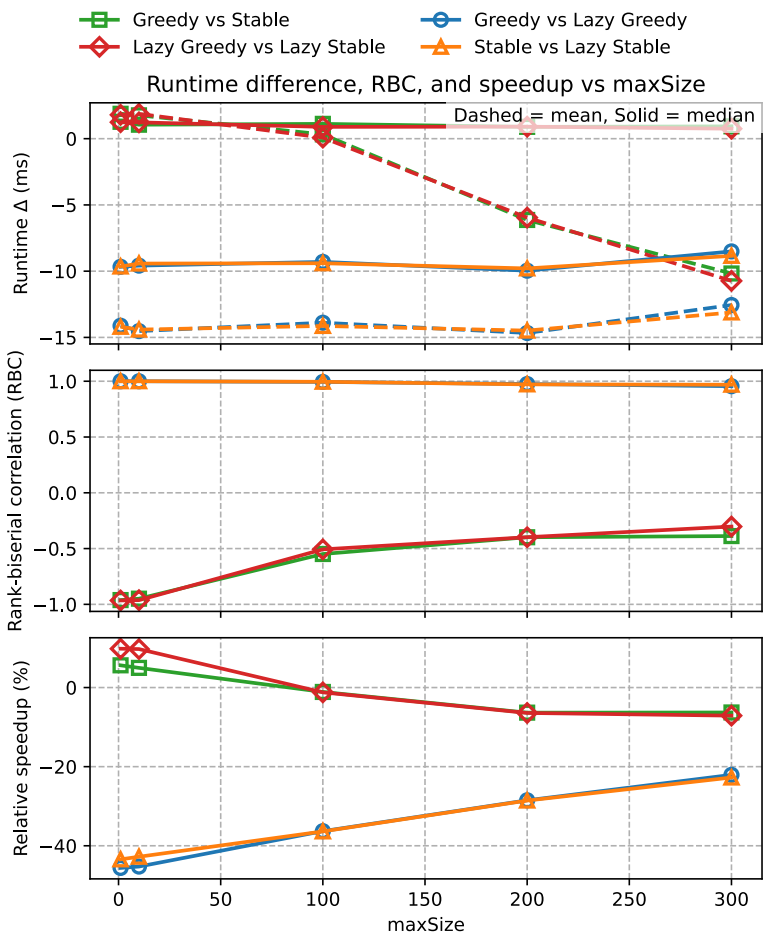
Results are statistically significant with $p \ll 10^{-16}$ (Wilcoxon)

Greedy vs Stable

- Greedy slightly outperforms Stable on most files (median > 0, RBC < 0);
- Stable becomes faster on average as $maxSize$ grows, implying it has **faster recovery**;

Lazy vs non-lazy

- Lazy variants consistently outperform non-lazy counterpart, both on average and on most files;



5 Conclusion

RQ1:

- **Greedy** is slightly faster on most file pairs, but **Stable** is faster on average, like on codebases;
- **Stable can avoid costly recovery** on some files using more restrictive mapping criteria;

RQ2:

- HyperDiff lazy evaluation optimizes both Greedy & Stable equally and is a **viable strategy for scaling**

Contact info

Author:	Elias Hoste
Email:	e.r.hoste@student.tudelft.nl
Supervisor:	Quentin Le Dilavrec
Responsible professor:	Carolin Brandt