MOBS: Multi-Operator Observation-Based Slicing using Lexical Approximation of Program Dependence

Seongmin Lee†, David Binkley‡, Nicolas Gold§, Syed Islam▽, Jens Krinke§, Shin Yoo†

†:KAIST, Republic of Korea ‡:Loyola University Baltimore, USA §:University College London, UK ▽:University of East London, UK

ORBS

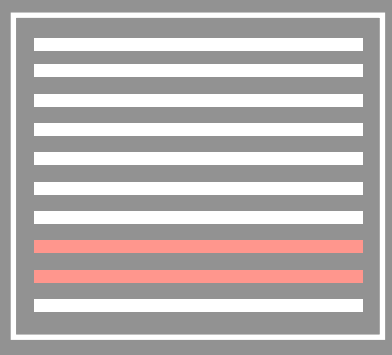
- Purely dynamic & Language Independent slicing
- Makes a series of deletions of code lines, which
  1) leaves the code (still) compilable, and
  2) preserves the trajectory of the slicing criterion.
- Approximates the program dependence via observations of test executions.

Scalability Issue

- Compilation & execution every time it attempts to delete

7200 sec / 220 lines * Guava escape package

Window Deletion



Code

- Deletion Candidate (p')

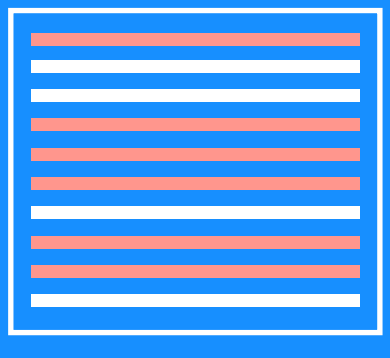
$$\left[\forall p' \ (\mathrm{Loc}_p - \mathrm{Loc}_{p'}) < \delta\right]$$

← Line to consider for deletion (p)

Lexical Similarity based ORBS

- Represents code lines as numeric vectors using IR-based methods.
  - Vector Space Model (VSM) and Latent Dirichlet Allocation (LDA)
- Attempts to delete lexically similar lines at once.

LS Deletion



Code

- Deletion Candidate (p')

$$\left[\forall p' \ \mathrm{sim}(p, p') < \gamma\right]$$

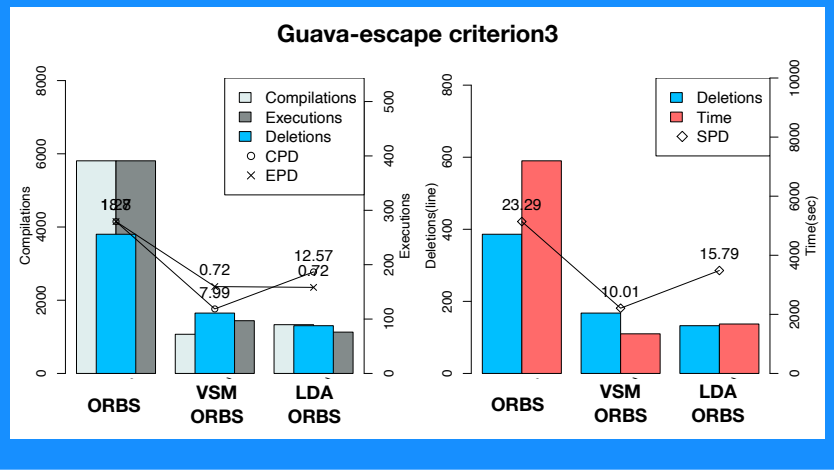
← Line to consider for deletion (p)

Advantages on Efficiency

- No limit to the number of lines that can be deleted simultaneously.
- Can delete non-consecutive lines.
- Makes only 1 deletion attempt at each code line in an iteration.

Experimental Results

- 12 slicing criteria from Java and C benchmarks
  - Java : Apache commons-cli, commons-csv, Guava
  - C : Siemens suite
  - NCLOC : 208 ~ 2,081



VSM: 45.9% comp/del 63.1% exec/del 49.5% time/del

LDA: 59.9% comp/del 65.9% exec/del 62.2% time/del

MOBS: Multi-Operator ORBS

- Window and LS deletion operators have different characteristics.
  - Window Deletion: more precise / LS Deletion: more efficient
- MOBS stochastically selects the next deletion operator to use from the pool of both window and LS deletion operators.

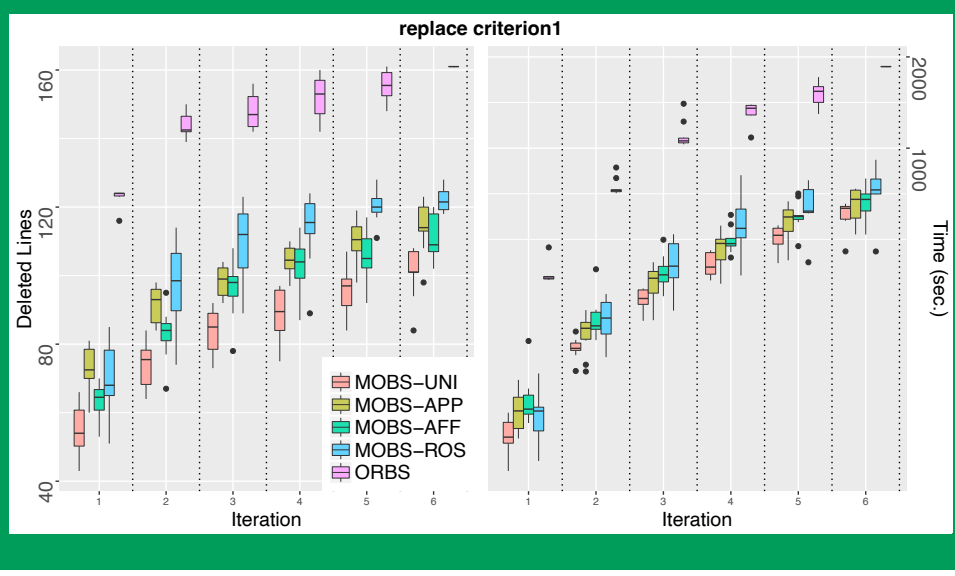
Selection Strategies

Fixed Operator Selection (FOS)

- Uniform
- Applicability (Success rate)
- Affect (# of deletable lines)

Rolling Operator Selection (ROS)

$$\left[P_{new}(D) = \omega \times P(D)\right]$$



Experimental Results

Achieves both effectiveness and efficiency

- MOBS with ROS selection strategy performs the best

Deletes 87% lines in 33% time w.r.t ORBS

Conclusion

- We present a generalization of ORBS that can use a wide range of deletion operators instead of the original deletion window only.
- We introduce lexical deletion operators that exploit lexical similarities between source code lines to improve the efficiency of ORBS.
- We propose MOBS that can significantly improve the efficiency by using multiple deletion operators.

ACKNOWLEDGEMENT - This research was supported by Next-Generation Information Computing Development Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT (No. 2017M3C4A7068179).