Total number of tools found: 39
Total number of potentially suitable tools found: 11

Frameworks that implement multiple techniques:

1. **Arja** (2020): https://github.com/yyxhdy/arja
2. **ASTOR** (2016): https://github.com/SpoonLabs/astor:

GenProg reimplementations:

3. **GenProg**- Arja's implementation for GenProg
4. **KGenProg** (2018): https://github.com/kusumotolab/kGenProg
5. **jGenProg** (2014)- Astor's implementation for GenProg

6. Genprog4java **JarFly** (2020) Squareslab's implementation for GenProg
   https://github.com/squaresLab/genprog4java/

7. **RSRepair**-A Arja's implementation for RSRepair
   Variation of GenProg
8. **jMutRepair** (2016)
   "a mutation-based repair approach implementation for Java with a 3 built-in mutation operators and an easy way to add new ones."

9. **HistoricalFix** (2016): https://github.com/xuanbachle/bugfixes
   a. Mines bug fix patterns from history of many projects
   b. Use mutation operators + generate fix candidates
   c. Give priority to candidates that match frequently occurring historical bug fixes
      Too specific of an implementation might not generalize

Involve machine-learning:

10. **LIANA** (2022) https://ieeexplore.ieee.org/document/9749899
    Uses a model that is initially trained offline to learn features of fixes. Gets repeatedly updated online during the fix generation process

---

Sources of Information-: How tools were found + what was eliminated and why:

Color key:

- 🟥 Not an option for us. Justification provided.
- 🟨 Preferably not (might need first hand investigation). Justification provided.
- 🟩 To consider

1. Paper: a 2020 systematic assessment of Java APR Tools (showed up in my lit review):
   Kim, T. F. Bissyand´e, D. Kim, P. Wu, J. Klein, X. Mao, and Y. L. Traon, “On the efficiency of test suite based program repair: A systematic assessment of 16 automated repair systems for Java programs”
   https://arxiv.org/abs/2008.00914

   Includes 31 tools!

   1. **Arja (2020): https://github.com/yyxhdy/arja**
   2. **RSRepair-A Arja’s implementation for RSRepair**
   3. **GenProg- Arja’s implementation for GenProg**
   4. **jGenProg (2014)- Astor’s implementation for GenProg**
   5. **jMutRepair (2016)**

   6. **PraPR https://github.com/prapr/prapr**
      Bytecode-level. Not what we were looking for but could be interesting to discuss
   7. **HDRepair**
      Breaks the criterion of APR tools that they should only have the source code + test suite as input with no extra assumption
   8. **JAID (2017): https://bitbucket.org/maxpei/jaid/wiki/Home**
      Breaks the criterion of APR tools that they should only have the source code + test suite as input with no extra assumption
   9. **SketchFix (2018): https://github.com/SketchFix/SketchFix**
      Breaks the criterion of APR tools that they should only have the source code + test suite as input with no extra assumption

   10. **Kali-A Arja’s implementation for GenProg**

   11. **jKali (2016)- Astor’s implementation for Kali**
       Mutation is just deletion

   12. **JFix/s3 (2017)**
       - Seems like source code is available but installation and tutorial “coming soon”
       - Also semantics based/symex… not genetic improvement
   13. **DeepRepair:**
       Excluded in paper because it didn’t run
   14. **CapGen (2018): https://github.com/justinwm/CapGen**
       Excluded in paper from bullet 3 because it didn’t run
   15. **NPEFix (2017): https://github.com/SpoonLabs/npefix**
       Does not use a fault localization technique
   16. **ssFix (2017): https://github.com/qixin5/ssFix**
       Both review papers had issues running it and excluded it
   17. **Par**
       Not public
   18. **xPar**

    Not public

19. **Elixir**

    Not public

20. **Hercules**

    Not public

21. **SOFix**

    Not public

22. **Cardumen:**

    Constraint-based repair approach: “Dedicated to repairing buggy if cibsutuibs abd to adding missing if preconditions”

23. **DynaMoth**

    Constraint-based repair approach: “Dedicated to repairing buggy if cibsutuibs abd to adding missing if preconditions”

24. **FixMiner:**

    Template-based repair approach (fix pattern based)

25. **ACS (2017): https://github.com/Adobee/ACS:**

    Constraint-based repair approach: “Dedicated to repairing buggy if cibsutuibs and to adding missing if preconditions”

26. **Avatar (2019): https://github.com/TruX-DTF/AVATAR:**

    Template-based repair approach (fix pattern based)

27. **LSRepair (2018): https://github.com/TruX-DTF/LSRepair**

    - Requires run-time code search over Github repositories

28. **Nopol (2014): https://github.com/SpoonLabs/nopol/**

    Constraint-based repair approach: “Dedicated to repairing buggy if cibsutuibs and to adding missing if preconditions”

29. **SimFix (2018): https://github.com/xgdsmileboy/SimFix**

    Only compatible with Defects4J

30. **TBar (2019): https://github.com/TruX-DTF/TBar**

    Template-based repair approach (fix pattern based)

31. **kPAR (2019): https://github.com/TruX-DTF/FL-VS-APR/tree/master/kPAR**

    Template-based repair approach (fix pattern based)

2. Website (found it through the paper in bullet 1): *http://program-repair.org/tools.html*

   Tools that were not in the previous paper but were on the website:

32. **ASTOR (2016): https://github.com/SpoonLabs/astor:**

    Includes the following

    a. jGenProg
    b. Cardumen
    c. Jkali
    d. Jmutrepair
    e. Deeprepair
    f. 3sfix

33. **ConFix (2019): https://github.com/thwak/ConFix**

“Currently, ConFix is fitted to execute for Defects4j bugs”

34. **GenPat (2019): https://github.com/xgdsmileboy/GenPat**
    Uses an inference algorithm to fix bugs- not a genetic approach
35. **Genesis (2017): https://github.com/monperrus/genesis**
    Doesn’t use failed test cases. Using successful human patches to infer patches for unseen bugs
36. **HistoricalFix (2016): https://github.com/xuanbachle/bugfixes**
37. **QACrashFix (2015):**
    Queries stackoverflow
38. **Repairnator (2018): https://github.com/eclipse/repairnator**
    Repairs build failures on Travix CI
39. **KGenProg (2018): https://github.com/kusumotolab/kGenProg**

3. Used this paper to also to help me categorize the tools (2019):
Empirical review of Java program repair tools: a large-scale experiment on 2,141 bugs and 23,551 repair attempts
https://dl.acm.org/doi/abs/10.1145/3338906.3338911?casa_token=nfSCn3NRdE4AAAAA:7-PjSJvWg_Fdm20KBXOTAnGlJPwe6RaJQkYJkDodmgphcwm4v5mAE6DPtLwuYFCd4mn3GGL6ymWcZsU

4. Since the two review papers were form 2019 and 2020, there was a gap. It was also clear by the missing JarFly tool in my search so far. Therefore I conducted a search for “java repair” in IEEE Xplore with the “All Metadata” filter and set the date range to 2020-2022.
I got **47** results. New tools discovered:

- DifFuzzAR: works on timing side-channel vulnerabilities in Java code, not general bugs
- RCSRepair https://ieeexplore.ieee.org/document/9742203
  Uses random search instead of APR based on genetic search
- JarFly
- DRONE “a framework to automatically detect and repair defects from API documents”
- Phoenix: “automatically generating high-quality patches for static analysis violations by learning from previous repair examples”. Doesn’t meet out test case input and genetic approach.
- ARJANMT (2022) not public yet https://ieeexplore.ieee.org/document/9749095
- ReFixar: for regression bugs
- Sorald: for SonarQube static analysis violations
- LIANA https://ieeexplore.ieee.org/document/9749899