TABLE 1.1. Comparison of Combinatorial Techniques Related to The Refactoring Problem

Authors	Journal/Venue	Google Scholar Citation Count	Optimization Technique	Objective(s)	Type of solution	Dimension Analysis Evaluation Method	Analyzed Systems	Supported Refactorings	Behavior Preservation	Reproducibility
Seng & Stammel (2006)	GECCO 2006	232	Single Objective EA (unclear technique).	Maximize features such as complexity, stability, coupling and cohesion by means of a weighted sum of seven metrics values. The input is the source code.	A list of model refactorings. The list is ordered according to established pre/post conditions of refactorings.	Fitness convergence analysis and an example code development.	JHotDraw	Move Method	Domain specific preconditions based on the object oriented structure and design patterns.	Medium
Keeffe, Mark O Cinnéide, Mel Ó (2007)	Journal of Software Maintenance and Evolution	101	Multiple-ascent hill climbing, simulated annealing and genetic algorithms.	An implementation of the unserstandability function from QMOOD model. The input is an Abstract Syntax Tree (AST).	The refactorings are applied to the AST, then the outputs are the refactored input code (unclear solution example).	Mean analysis on fitness values and execution time for each search technique.	SpecCheck, Beaver, EAOP, Mango and Grammatica.	14 Refactorings from Fowler's Catalog.	Static program analysis on an unknown pre/post-conditions. There is no evidence of how the refactoring is executed on the AST.	Hard
Harman & Tratt (2007)	GECCO 2007	173	Variant of hill climbing (unclear).	Two objectives: metric coupling between classes (CBO) and the standard deviation of methods per class (SDMPC).	Sequences of refactorings, although, it is unclear how the solution is configured.	Pareto front analysis. However, without concrete type solutions or outcome examples, the analysis is hard to interpret.	JHotDraw, Maven and Xom	Move Method	NA	Hard
Jensen & Cheng (2010)	GECCO 2010	52	Genetic programming.	A proposed fitness function composed of the QMOOD model, specific penalties,and number of modifications.	The individual is composed of a pair that includes a transformation tree and design graph (UML class diagram).	Exploratory analysis on fitness values. However, the analysis are very informal and no concrete solution is described.	The use of REMODEL on published models and real-world case study.	Gamma design patterns and mini-transformations.	NA	Hard
Ouni & Kessentini (2013)	GECCO 2013	24	NSGA-II (unclear how the refactorings were executed).	Maximize design quality, semantic coherence and the re-use of the history of changes.	Vector-based representation. A set of refactorings is configured and sorted using dominance principle and a comparison operator based on crowding distance. The order of the operations inside the vector is important.	DCR (defect correction ratio) and RP (refactoring precision). It is unclear how they reproduce previous approaches. The use of RefFinder (Eclipse Tool) for refactoring comparison.	Xerces-j, JFreeChart.	9 Refactorings from Fowler's Catalog.	Pre and post conditions described by Opdyke (not clear how authors performed behavior preservation).	Hard
Mkaouer & Kessentini (2014)	ASE 2014	13	NSGA-II (unclear how authors execute the 23 refactorings).	Improve software quality, reduce the number of refactorings and increase semantic coherence.	A set of Non dominated refactoring solutions. The set of refactorings are ranked.	Execution time analysis and a manual validation of the refactorings. There are comparison with other reported refactorings.	GanttProject, JHotDraw, JFreeChart, JDI-Ford.	23 Refactorings from Fowler's Catalog.	NA	Hard
Ouni & Kessentini (2015)	Software Quality Journal	6	Chemical Reaction Optimization. Although, there is no evidence or concrete example of how the refactorings are executed on the code.	One objective that minimize the number of bad smells.	A sequence of refactorings in a vector representation (unclear how it is the appearance of the suggested refactoring).	Execution time analysis and comparison to the number of bad-code smells with an old version of the systems.	Xerces-J, GanttProject, JHotDraw, JFreeChart.	11 Refactorings from Fowler's Catalog.	Opdyke's functions. However, these conditions are unclear in the implementation of the approach.	Hard
Mkaouer & Kessentini (2016)	Empirical Software Engineering	NA	NSGA-II (unclear how the refactorings are executed).	Maximize quality improvements and the respectively uncertainties associated with severity and importance of refactorings opportunities.	Vector-based representation of refactoring operations. The order depends on the position of the refactoring inside the array.	Hypervolume, Inverse Generational Distance, Number of Fixed Code-Smells, Severity of Fixed Code-Smells, Correctness of the suggested Refactorings and Computational Time.	Xerces-J, JFreeChart, GanttProject, ApacheAnt, JHotDraw, Rhino, Log4J, Nutch and JDI-Ford.	10 Refactorings from Fowler's Catalog.	Opdyke's functions (defining pre/post conditions). However, these conditions are unclear in the implementation of the approach.	Hard