Final Report for AI based SE

ABC – 2018xxxx Jeonggwan Lee – 20184400
GHI – 2018xxxx Nick Heppert – 20186505

December 15, 2018

Chapter 1

Introduction

The amount of information is exponentially increasing, the robust and delicate index structure also be needed. SQL, which is a language to manage database management system(DBMS) is widely used as tool which arranges effectively indices and relationship between tables, the area of application is very large, such as Electronic Health Record(EHR), customer management, and distribution management. Software developers can get necessary information through query statement in SQL. However, they need to verify their own query statement and check whether unexpected bugs exist. Otherwise, we can't believe the results of query and beyond lose reliability of SQL. After verifying, we can figure out this query statement is semantic or not. If the query statement is simple, developers can make test cases manually considering coverage targets, however query statement is complex, this manual making is very inefficient and hard to deal with.

To handle this problem, EvoSQL made test case generation tool covering whole coverage targets for SQL queries using genetic algorithm(GA), it shows astonishingly covering performance, 98.6%. Their contributions are, first suggesting the definition of test case generation problem for SQL and a "physical query plan", which is a series of step to solve in query. Through the physical query plan, they defined a fitness function and implement crossover and mutation as general GA. They presented seeding strategies, which the model has seeding pool to generate new individual, to fit the characteristic of SQL which normally covers string and integer(days). They compared three methods (Random search, Biased Random Search (supported by Seeding strategy), GA), and show how GA can cover coverage targets of various queries.

GA is much better than other two methods, but we thought any other improvements than GA, and wondered why they didn't use Multi-objective optimization(MOOP) methods. First, they didn't arrange the order of coverage targets to solve so that we should wait for solving easy coverage target if it is behind of hard coverage targets, and coverage targets don't share their semantic discovery to others because of limitation of single target strategy. Second, we thought there might be unnecessary time budget to solve to solve infeasible cov-

erage targets if the users didn't eliminate infeasible coverage targets. EvoSQL also removed those infeasible targets manually. We assumed that MOOP can handle them without manual elimination, since it simultaneously covers multiple coverage targets and detect them if it analyzes the tendency of non-decreasing fitness function.

Therefore, we expected that to apply MOOP, our model can be guided by similar solution from other coverage targets and reduce unnecessary time budget. Therefore, we first applied NSGA-II as basic MOOP method, changed crowding distance into "sort fronts by covered target" and added combine operator as minor technique to satisfy the coverage target easily.

The goal of our paper is implementation of MOOP version of EvoSQL, analysis and comparison of results, answer to suitability of MOOP for test case generation for SQL.

Our conclusion is, MOOP of EvoSQL is not applicable to, future work are ...1. add new fitness function 2. find a solution for a coverage target and save it, then remove that target from the pool of objectives

(need to add our contribution, invention, results) We organized contents as follows. Section 2 describes genetic algorithm in EvoSQL as baseline, and the representation of GA setting for SQL test case generation. Section 3 presents our representation of MOOP setting, our modified model based on NSGA-II. Section 4 we evaluate our model and analyze failure of it. We conclude the paper in section 5.

Chapter 2

Problem Formulation

Chapter 3

Methodology

Chapter 4

Evaluation