Search-Based Parallel Refactoring Using Population-Based Direct Approaches

Kilic, H (Kilic, Hurevren)[1]; Koc, E (Koc, Ekin); Cereci, I (Cereci, Ibrahim)

Abstract

Automated software refactoring is known to be one of the "hard" combinatorial optimization problems of the search-based software engineering field. The difficulty is mainly due to candidate solution representation, objective function description and necessity of functional behavior preservation of software. The problem is formulated as a combinatorial optimization problem whose objective function is characterized by an aggregate of object-oriented metrics or pareto-front solution description. In our recent empirical study, we have reported the results of a comparison among alternative search algorithms applied for the same problem: pure random, steepest descent, multiple first descent, simulated annealing, multiple steepest descent and artificial bee colony searches. The main goal of the study was to investigate potential of alternative multiple and population-based search techniques. The results showed that multiple steepest descent and artificial bee colony algorithms were most suitable two approaches for an efficient solution of the problem. An important observation was either with depth-oriented multiple steepest descent or breadth-oriented population-based artficial bee colony searches, better results could be obtained through higher number of executions supported by a lightweight solution representation. On the other hand different from multiple steepest descent search, population-based, scalable and being suitable for parallel execution characteristics of artificial bee colony search made the population-based choices to be the topic of this empirical study. I In this study, we report the search-based parallel refactoring results of an empirical comparative study among three population-based search techniques namely, artificial bee colony search, local beam search and stochastic beam search and a non-populated technique multiple steepest descent as the baseline. For our purpose, we used parallel features of our prototype automated refactoring tool A-CMA written in Java language. A-CMA accepts bytecode compiled Java codes as its input. It supports 20 different refactoring actions that realize searches on design landscape defined by an adhoc quality model being an aggregation of 24 object-oriented software metrics. We experimented 6 input programs written in Java where 5 of them being open source codes and one student project code. The empirical results showed that for almost all of the considered input programs with different run parameter settings, local beam search is the most suitable population-based search technique for the efficient solution of the search-based parallel refactoring problem in terms of mean and maximum normalized quality gain. However, we observed that the computational time requirement for local beam search becomes rather high when the beam size exceeds 60. On the other hand, even though it is not able to identify high quality designs for less populated search setups, time-efficiency and scalability properties of artificial bee colony search makes it a good choice for population sizes >= 200.