Chapter VIII

Conclusion

8.1. Conclusion

Search-based software engineering (SBSE) has become an important area of research with many conferences and tracks devoted to the subject. This thesis has addressed some fundamental issues in search-based software test data generation for branch testing, an important sub-field of SBSE. The issues addressed in the context of branch coverage are as follows:

1. Branch ordering schemes for coverage.
2. Exploiting and comparing different metaheuristic techniques for branch coverage using branch ordering and other enhancements.
3. Combinational approaches to fitness function design.
4. Enhanced parallel approaches to test data generation.
5. Using the maximization approach to define the test data generation problem and its comparison with the minimization approach.

The conclusions of the work done to address these issues are described in subsequent paragraphs.

Metaheuristic techniques, and in particular the genetic algorithm, have now been applied extensively to the problem of automated test data generation. However, as research points out, the application of metaheuristic techniques poses new challenges. One of the problems faced in generating test data for branch coverage using a metaheuristic technique is that the population may not contain any individual that encodes test data for which the execution path reaches the predicate node of the target branch. In order to deal with this problem, in Chapter III, we (a) have introduced three approaches for ordering branches for selection as targets for coverage; namely, depth first strategy, breadth first strategy and the path prefix strategy and (b) have considered elitism and memory along with branch ordering to improve test data
generation capability. Extensive experiments have been carried out on standard benchmark programs for a genetic algorithm (GA) and quantum particle swarm optimization algorithm (QPSO) whose implementations have also been described in detail in the chapter. Pilot experiments were carried out to fine tune GA and QPSO parameters which were then used in the final experiments. The performance of GA and QPSO has also been compared and the results described in the chapter.

Results indicate that

- A scheme in which the population is initialized once at the beginning of the GA and QPSO run, the fitness is computed using approach level and normalized branch distance and is maximized, the branch ordering strategy is the path prefix strategy, memory and elitism are used and for the genetic algorithm, binary tournament selection with two point crossover, $P_m$-0.01 and $P_c$-1.0, gives the best performance in terms of number of generations and coverage.

- On considering the results of the random strategy for branch selection (RAN) one may conclude that the strategies of elitism, memory and branch ordering may have significantly contributed to improving performance of both GA and QPSO.

- Better performance may be possible for programs with integer inputs as compared to real inputs and large population sizes.

A comparison between GA and QPSO to determine which gives better performance is inconclusive. Further, analysis needs to be carried out to identify program features that may allow one to choose a particular metaheuristic approach.

In Chapter IV we have described a modification of CLONALG and applied it to the program test data generation problem. The chosen test data adequacy criterion was the branch coverage criterion. Four different hypermutation operators were introduced and compared in a pilot experiment. Final experiments were carried out on eight benchmark programs and the performance of modified clonal selection was statistically compared with genetic algorithm. In most cases it was found that clonal selection performed significantly better than the genetic algorithm in terms of the mean number of generations required to achieve coverage. Further, it was found that modified clonal selection achieved full branch coverage in all the cases whereas genetic algorithm could not for small population sizes for some benchmarks. Modified clonal selection made use of newer ideas as follows. The test data generation problem was cast as maximization problem; the path prefix strategy was used to order
branches for selection for coverage so as to ensure that the population of antibodies had a member for which the execution path traversed the sibling branch; memory was explicitly used to infuse the population with fitter antibodies; and local search was used as a part of the hypermutation process. Further experiments with variants of CLONALG and hybrids need to be carried out to determine if search performance can be improved further.

In Chapter V, we describe a novel approach for fitness computation for metaheuristic search. Fitness computation is a two-step process. In the first step a target node sequence is determined and in the second step the actual execution path is compared with the target node sequence to compute fitness. Fitness computation is based on a combination of two criteria

1. Branch distance and approach level. This is based on the maximization proposal described in Chapter V.

2. Path similarity. This is based on the description by Lin and Yeh (Lin and Yeh, 2000).

Comparison of the described fitness technique with approaches that use (1) and (2) independently shows significant improvement in search performance.
In Chapter V we present a proposal for an approach to test data generation for branch coverage with a structured (parallel) genetic algorithm (GA) using the extended path prefix strategy. The structured GA implements a master-slave distributed model in which each slave implements an elitist panmictic GA. Branches to be covered are selected by the master using the extended path prefix strategy and then dispatched to slaves. The slaves then conduct search for test data to cover the assigned target branch. The master keeps track of all the branches that have been covered and also store a specific number of individuals (test data) that traverse them in memory (referred to as test-data-memory here). The extended path prefix strategy ensures that each time a branch is selected for coverage, the sibling branch is already covered and that individuals are available in the test-data-memory that traverse the sibling. Thus along with the target branch, individuals are also dispatched to slaves that cover the sibling branch. Slaves keep track of additional branches that are traversed in the course of search and return these branches along with the individuals that traverse them to the master. The extended path prefix strategy permits a variable number of slaves to be used which can help speed up the test data generation process. Experiments on two programs with real inputs indicate that significant improvements are achieved over a simple panmictic GA in terms of number of generations and the coverage achieved.

In search based test data generation, the problem of test data generation is reduced to that of function minimization or maximization. Traditionally, for branch testing, the problem of test data generation has been formulated as a minimization problem. In Chapter VI we have defined an alternate maximization formulation and experimentally compared it with the minimization formulation. We have used a genetic algorithm and binary particle swarm optimization as the search technique and in addition to the usual operators we have also employed the path prefix strategy as a branch ordering strategy and memory and elitism. Results indicate that there is no significant difference in the performance or the coverage obtained through the two approaches and either could be used in test data generation if coupled with the path prefix strategy, memory and elitism.

As further work to be carried out, we need to focus on

1. Incorporating local improvement strategies. During search, performance may be improved if a local neighbourhood search is incorporated in each generation.
2. Testing the strategies on a suite of larger programs. This will give further results on the efficacy of the suggested strategies.
3. Multiobjective approaches to fitness function design. Although in Chapter V, we do present a form of multiobjective fitness computation, but we need to identify many conflicting objectives and work on their optimizations.