Test data generation in software testing is the process of identifying a set of test cases to satisfy a selected test data adequacy criterion. Bertolino (Bertolino, 2007) points out that the most promising results towards automated test data generation have come from three approaches: model-based, random and search-based approaches. Search-based test data generation is a part of the much broader research area of search-based software engineering (Harman and Mansouri, 2010). Search-based test data generation consists of exploring the input domain of a program under test for test data to satisfy a selected test data adequacy criterion. By using meta-heuristic techniques - high-level frameworks which utilize heuristics in order to find solutions - search can be directed towards the most promising areas of the domain (McMinn, 2004). These techniques can be used for specification-based criteria as well as program-based criteria.

Metaheuristic techniques such as simulated annealing, tabu search, genetic algorithms, particle swarm optimization, quantum particle swarm optimization, scatter search, ant colony optimization, memetic algorithms, clonal selection algorithm and immune genetic algorithm have been applied to the problem of automated test data generation and provide evidence of their successful application. Amongst these, several have addressed the issue of test data generation with program-based criteria and in particular the branch coverage criterion. In program based test data generation using metaheuristic techniques, the basic approach is as follows. The source code of the program under test is instrumented to collect information about the program as it executes. The resulting information, collected during each execution of the program, is used to heuristically determine how close the input is to satisfying the test requirement as specified by the selected test criterion. This allows the test generator to modify the program's inputs gradually, moving them ever closer to values that actually do satisfy the requirement. In other words, the problem of generating test data reduces to the well understood problem of function optimization. Furthermore, such test generation methods can handle arrays and pointer references because the values of array indices and pointers are known throughout the generation process. Wegener et al. (Wegener et al., 2002) point out that because of the non-linearity of software (conditional statements, loops, flags, switch-case,
break) the conversion of test problems into optimization tasks usually results in complex, discontinuous and non-linear search spaces for which search methods such as hill climbing are not suitable, but metaheuristic search methods can be employed. This dissertation addresses important issues in the application of metaheuristic techniques to the test data generation problem for procedural programs.

In the thesis, chapter I broadly introduces the subject areas of research and points out the focus of research. Chapter II presents an in depth literature review on the application of metaheuristic techniques to the problem of program test data generation and outlines the major issues which this dissertation has addressed. Chapters III through VII present the main research contribution which is detailed in the subsequent paragraphs. Finally, chapter VIII concludes the thesis with suggestions for future extensions.

Chapter III introduces and presents an experimental evaluation of an improved approach for search-based test data generation with genetic algorithms and quantum inspired particle swarm optimization using branch ordering, memory and elitism. One of the problems faced in generating test data for branch coverage using a metaheuristic technique is that the population may not contain an 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, three approaches for ordering branches for selection as targets for coverage are introduced, namely, depth first strategy, breadth first strategy and the path prefix strategy. Further, elitism and memory are also considered as additional features to improve test data generation capability. Extensive experiments to measure performance with these improvements were carried out on standard benchmark programs and statistically evaluated for a genetic algorithm (GA) and a quantum inspired particle swarm optimization (QPSO) algorithm. Pilot experiments were carried out to fine tune GA and QPSO parameters which were then used in the final experiments. Further the performance of GA and QPSO was also compared. These experiments are described in detail in the chapter along with the statistical analysis that was carried out. Results of the final experiments indicate that using branch ordering, specially the path prefix strategy, and the strategies of elitism and memory may have contributed significantly to improving the performance of both GA and QPSO.

Chapter IV describes a modification of a clonal selection algorithm and its experimental evaluation for search-based test data generation for branch coverage. The proposed modified clonal selection makes use of the following ideas: test data generation problem is cast as
maximization problem; path prefix strategy is used to order branches for selection for coverage; memory is used to augment the population and improve performance; and local search is used as a part of the hypermutation process. Pilot experiments were carried out to choose from among four different hypermutation operators. This was followed by extensive experiments on standard benchmark programs to evaluate performance and to statistically compare performance with that of a genetic algorithm implementation. For most benchmark programs 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 and the mean percentage coverage achieved.

Chapter V introduces and presents the experimental evaluation of a dynamic strategy for search-based test data generation with genetic algorithm for branch coverage with dependencies. The program fragment below illustrates the problem:

```c
(1) int flag = 0;
(2) if (a == 0)
(3)   flag = 1;
(4) if (b != 0)
(5)   flag = 0;
(6) if (flag)
     {
(7)   // target
     }
```

In the code fragment, the true branch (6,7) is traversed only if node (3) is traversed first, i.e., there is a def-use dependency that must be covered for the target to be reached. The strategy described in the thesis involves computing a target path from the def-use elements to be covered and the program path actually traced by the test data. The target path is dynamically computed since it is different for each individual (test data) generated through the genetic algorithm. The target path is then used to compute fitness of the individual (test data) that uses path similarity (Lin and Yeh, 2001) and information at the first branch that leads the actual execution path away from the target. Experiments suggest significant improvement in performance with the proposed scheme over well-known path coverage and branch coverage proposals.

Chapter VI presents and experimentally evaluates an MPI based parallel approach for search-based test data generation with genetic algorithms using the path prefix strategy. Parallelisation, in general, offers significant performance advantages. However, the problem
in the case of search-based test data generation is not only in deciding which component of the process to parallelise, but also integrating it into the GA generations. In this research, branches are selected for coverage in parallel by the genetic algorithm using the path prefix strategy. The strategy is developed, implemented and also experimentally evaluated on a 24 blades, 288 core, Dell™ cluster for different benchmark programs. Results show significant improvement in test data generation performance over serial implementations.

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 order to explore alternate formulations, in Chapter VII a maximization approach is introduces and its experimental comparison with the minimization formulation is presented. A genetic algorithm and binary particle swarm optimization is used as the search technique and in addition to the usual operators, the path prefix strategy as a branch ordering strategy, memory and elitism, which were introduced in Chapter III, were also used. Results indicate that there is no significant difference in the performance or the coverage obtained through the two approaches and that there is no significant reason to use the minimization approach specially if coupled with the path prefix strategy, memory and elitism.