6.1 Conclusions

The stochastic simulator based on GA_FA (Genetic Algorithm for Fragmentation and Access Strategies) was run a number of times on various test databases and benchmark queries by varying the number of transaction attributes, transaction frequencies and up scaling the size of the query problems. Results analysis from experiments helped in drawing following conclusions.

- GA_FA proved to be an innovative heuristical approach for integrating access strategies with fragmentation scheme at a local site. It contributes significantly in reducing the overall total cost of a distributed query by reducing the local access costs and help find a better subquery allocation scheme for various network sites.
- The number of disk accesses and hence query costs showed a reduction of 61 percent as compared to an un-partitioned scheme. Reduction of disk access was to the tune of 33 percent in comparison to that of earlier popular deterministic techniques of [Cornell & Yu (1990)], and by 23% of the genetic solution by [March, (1983)]. GA_FA’s disk access cost showed a 10% decrease in access costs and communication costs as well, as compared to [Barker & Jun (2006)]’s approach.

In case of GA_SA (Genetic Algorithm for Subquery Allocation) the parameters varied were number of joins & number of sites, for a set of Wisconsin Benchmark Database queries. Various existing techniques like deterministic Exhaustive procedures, Dynamic Programming, Branch & Bound, Simulated Annealing technique’s run time and costs are compared with GA_SA.

- GA_SA’s findings are that when number of joins and sites individually grow more than seven, deterministic and heuristic procedures tend to go exponential and computationally intractable quickly. Stochastic Solutions like GA_SA Still take time around a minute or less, because they are independent from query tree search.
space size. Exhaustive Enumeration Procedures choke very quickly for very large problem sizes and computing time to find a solution goes into units of hours, which is not permissible in any interactive query system. Whereas, Genetic Algorithm GA_SA’s run time rises very slowly and is virtually independent of the problem complexity. Though, GA’s does not guarantee the best or optimal solution, but give a reasonably optimal solution in a very small computing time.

- Large scale Stochastic Testing was done to study the effect of different implementations of various Genetic Operators (SELECTION, CROSSOVER, and MUTATION) and their Parameter Value Variations, on the Quality of a Genetic Solution. Results are analyzed and verified empirically. Role of general genetic parameters like population size, crossover rate, mutation rate on the performance of GA_SA(Genetic Algorithm for SubQuery Allocation) was examined. A few drawn conclusions are , Population size should be kept fixed and if it’s a small number N (between 10 and 40) then minimum number of generations to reach a good quality solution is high. But if it’s a big number N (between 100 and 200) then time taken by the GA_SA to reach a good solution (Computation Time) increases. Minimum number of generations to reach a good quality solution is no better than the case if we keep N to a moderate size of (70-100). Optimal number for size of Population was observed to be 100, which did not result in increase of computation time and needed lesser number of generations to reach a solution.

- Crossover Rate for GA_SA is varied from a low level of 0.2 to high of 1.0. If Crossover Rate is kept very low it takes lot many generations to reach a good solution and probability of finding a good solution reduces considerably. If it is kept high like 1.o where whole population crosses, quality of solution is poorer than we get with moderate values like 0.6.

- Next observation was that Mutation should not occur very often, because then GA will in fact change to a random search. It is should be applied to less than 1% of the population of a generation. Mutation rate of 0.2 was found to be most suitable for genetic Sub Query Allocation Algorithm (GA_SA). Crossover rates are varied from 0.2 to 0.9 and Mutation rates are varied from 0.005 to 0.2 on various test
queries. For example (0.7, 0.2) combination of Crossover Percentage & Mutation percentage provided the best quality of solution for medium size sub query allocation problems for SA_GA.

- Experiments were performed for determining the optimal parameter values for LAN/WAN to produce a good quality genetic solution for these environments. Total Query Cost’s breakups into CPU, I/O and Communication costs were studied for LAN/WAN environments by varying the number of joins involved in a query. When ‘number of joins & sites’ were increased more than 12, Communication costs started dominating all other costs for a WAN but surprisingly I/O costs remained dominating costs in case of a LAN.

Future Directions
- Research Contributions highlighted in this thesis are more applicable to OLTP (On Line Transaction Processing queries), which prefer Total Time Optimization. Total time optimization concentrates on minimum use of resources rather than minimizing the response time of a query. All query steps are linear and sequential in nature. Whereas DSS (Decision Support System) Queries vouch for Response Time Optimization and employ use of parallel processing techniques. This work can be extended to DSS Queries by incorporating parallel processing for various sub operations that are independent of each other and may occur simultaneously at different sites.
- Adding the parallel processing component would necessitate the load balancing of various processors at various sites. Interposes as well as intraprocess parallelization can greatly improve the response time of a distributed query.
- All four major components of database design: Fragmentation, Data Allocation, Subquery Allocation and Load Balancing needs to be integrated in one composite genetic solution.
- Update Transactions, Concurrency and Security issues should be included in future improvements.
- More innovative Complex Relational Operators and heuristics are needed to be explored to help optimize complex update queries.