**Research Methodology:**

Computer Science as a research discipline has always struggled with its identity. On the one hand, it is a field deeply rooted in mathematics which resulted in strong theories. For example, there is computational complexity theory like turingmachines, the halting problem, database theory (the relational model), formal language theory (the chomsky hierarchy, well-formed, formal semantics). On the other hand, it is a field deeply rooted in engineering which resulted in machines that have completely warped our society: the von Neumann architecture (the basis for digital computers), parallel processors (the new generation of multi-core machines), distributed computers (a prerequisite for the success of the internet and recent phenomena like grid computing). Consequently, computer science has inherited its research methods from the same disciplines: on the one hand, the mathematical approach with axioms, postulates and proofs; on the other hand the engineering approach with quantification, measurements and comparison.

Software Engineering research in particular has suffered from this identity crisis, and several authors have argued the need for stronger research methods. Moreover, software engineering research - with its emphasis on processes and team work - must also take into account group dynamics and cognitive factors, hence borrows research methods from sociology and psychology as well. Certainly, with innovations like distributed development and open source release, software engineering is at the forefront of introducing new communication paradigms, hence is itself a test-bed for experiments in social sciences.

Computer Science draws its foundations from a wide variety of disciplines. Study of Computer Science requires utilizing concepts from many different fields. Computer Science integrates theory and practice, abstraction and design. The historical development has led to emergence of a big number of sciences that communicate more and more not only because the means of communication are getting very convenient and effective, but also because a need increases for getting a holistic view of our world.

Following research methodologies have been accepted for computer science and software engineering.

1. Theoretical Methodology
2. Experimental Methodology
3. Computer Simulation or Modeling

**Theoretical Computer Science**

Concerning Theoretical Computer Science, which connects to the traditions of Logic and Mathematics, we can conclude that it follows the very classical methodology of building theories as logical systems with stringent definitions of objects (axioms) and operations (rules) for deriving/proving theorems. The key recurring concepts fundamental for computing are:
- Conceptual and formal models
- Levels of abstraction
- Efficiency

Data models are used to formulate different mathematical concepts. In CS a data model has two aspects: the values that data objects can assume and the operations on the data.
Some of the central methodological themes in theoretical Computer Science (inherited from Mathematics) are **iteration, induction and recursion**.

**Iteration:** The simplest way to perform a sequence of operations repeatedly is to use an iterative construct such as `for`- or `while`-statement.

**Recursion:** Recursive procedures call themselves either directly or indirectly. This is self-definition, in which a concept is defined in terms of itself. There is no circularity involved in properly used self-definition, because the self-defined subparts are always “smaller” than the object being defined. Further, after a finite number of steps, we arrive at the basis case at which the self-definition ends.

**Induction:** Inductive definitions and proofs use basis and inductive step to encompass all possible cases.

To summarize theoretical computer science seeks largely to understand the limits on computation and the power of computational paradigms. Theoreticians also develop general approaches to problem solving. One of theoretical computer science’s most important functions is the distillation of knowledge acquired through conceptualization, modeling and analysis. Knowledge is accumulating so rapidly that it must be collected, condensed and structured in order to get useful.

**Experimental Computer Science**

Experimental computer science is most effective on problems that require complex software solutions such as the creation of software development environments, the organization of data that is not tabular, or the construction of tools to solve constrained optimization problems. The approach is largely to identify concepts that facilitate solutions to a problem and then evaluate the solutions through construction of prototype systems. Experiment in different fields such as search, automatic theorem proving, planning, NP-complete problems, natural language, vision, games, neural nets/connectionism, machine learning, is also used in CS, and is described by methodology.

**Computer Simulation or Modeling**

In recent years, computation, which comprises computer-based modeling and simulation, has become the third research methodology, complementing theory and experiment. Today, computing environments and methods for using them have become powerful enough to tackle problems of great complexity. Mastery of Computational Science tools, such as 3D visualization and computer simulation, efficient handling of large data sets, ability to access a variety of distributed resources and collaborate with other experts over the Internet, etc. are now expected of university graduates, not necessarily Computer Science majors. Those skills are becoming a part of scientific culture.

With the dramatic changes in computing, the need for dynamic and flexible Computational Science becomes ever more obvious. Computational Science has emerged, at the intersection of Computer Science, applied Mathematics, and science disciplines in both theoretical investigation and experimentation. Computer simulation makes it possible to investigate regimes that are beyond current experimental capabilities and to study phenomena that cannot be replicated in laboratories, such as the evolution of the universe. In the realm of science,
computer simulations are guided by theory as well as experimental results, while the computational results often suggest new experiments and theoretical models.

**Proposed Design**

Proposed project will be based on parallelism in the solution techniques to Transportation problem. Algorithm will be developed to apply methods to find feasible and optimal solution to Transportation problem. Performance of algorithm will be measured in conventional Big-O notation. This will help differentiate solution using parallelism and solution using sequential methods. As specified, there are many ways parallelism can be applied to conventional linear system.

First approach is to apply all solution methods to the given data simultaneously to find first feasible solution and then again repeat to find next feasible/ optimal solution. Use of multithreading concept is made here.

Another method is to restructure the algorithmic steps, preserving the meaning of algorithm intact. For instance, a simple bubble sort algorithm’s performance can be improved by changing its way of comparing the elements. Instead of comparing two adjacent elements of an array, all odd and then all even indexed elements can be compared with each other to maintain order.

Finally, one more technique can be thought to speed up the processing, pipelining approach. While one method’s feasible solution is being processed another method’s optimal solution can be calculated. This also helps speed up the computations and solves the Transportation problem in parallel with multiple methods.

**Algorithm**

a. Formulate balanced Transport table of size m*n.

b. Obtain initial basic feasible solution using:
   
   i. North west corner rule
   
   j. Lowest cost entry
   
   k. Vogel’s approximation

c. Test acceptability of solution; number of non-zero entries will be m+n-1.

d. Find the optimal solution using:
   
   i. Stepping stone method
   
   j. Modified distribution method
Inception of parallelism in system

As specified earlier parallelism lies in various forms in the system and it may take place at any instance of the system. The concerned system may exhibit parallelism in the method one has opted for. An individual can run all three methods in parallel on same data to check the accuracy of the feasible solution, viz.

1. North-West corner rule
2. Lowest cost entry method
3. Vogel’s approximation method

The result of each of these methods can be compared against most feasible solution. The most feasible solution can be taken for optimality test. These methods basically differ in the approach they take.

After determination of feasible solution, it has to be checked whether the solution is optimal or not. To find optimal solution following methods can be used;

1. Stepping stone method
2. Modified distribution method

Optimality test can also be run in parallel, which means both of these methods can be used simultaneously to get optimal solution.

Although the methods differ in their approaches but their performance varies with data being processed. For example Vogel’s approximation method works little better than that of other two methods. Here also one can observe parallelism in the system. The concept of parallelism is not only limited to do the things faster by introducing new techniques or
tools in multithreaded execution environment. Parallelism can also be achieved by modifying the existing algorithm or method in order to make the things faster. Vogel’s approximation does exactly that. It reduces the number of comparisons by taking difference between two successive cells. This is the significant improvement comparatively over speed at which other two methods work. That’s another way parallelism comes in the system.

System requirement

Proposed project can be deployed in a live system for better results and some important conclusions can be drawn from it. For its real time implementation following are the minimum requirements:

Hardware requirements

Project should be implemented on at least Pentium dual core processor with minimum 64MB RAM.

Software requirements

As project is based on parallel processing and parallel processing can be better implemented using multithreading concept, Java programming language is the best option for implementation.

Application development language: Java
Platform: JDK 1.3 or higher
Integrated development environment: Netbeans 7.0 or higher
Operating system: Any 32 bit.

Research Instrument:

1. A normal Pentium Dual Core processor with 64 MB RAM and L2 Cache of 1 MB
2. Java Interpreter
3. JDK 1.3 or higher
4. Netbeans 7.0 or higher
5. Any 32 bit OS
Work Plan

1. Review of more literature for next two month.
2. To design of algorithm for serial / sequential approach next two months.
3. To develop the program using Java for serial / sequential approach for next two months.
4. To design of algorithm for parallelism approach next two months.
5. To develop the program using Java for parallelism approach for next two months.
6. To test the programs for different data values for next two month.
7. Start writing the thesis to finish it next three months.