NATIONAL CONFERENCE ON METHODS AND MODELS IN COMPUTING (NCM2C-2007), School of Computer & Systems Sciences, Jawaharlal Nehru University, New Delhi, December 13-14,2007

A DNA computing approach to solve Task Assignment problem in Real Time Distributed computing System

Aser Avinash Ekka
Department of Computer Science & Engineering
NIT Rourkela, ORISSA, INDIA

Bibhudatta Sahoo
Department of Computer Science & Engineering
NIT Rourkela, ORISSA, INDIA

Abstract

Early attention has focused on DNA because its properties are extremely attractive as a basis for a computational system. In this paper, we have proposed a new framework for assigning task in heterogeneous Real time distributed system(RTDS) including the cost of path connecting the nodes. The proposed approach is based upon a DNA replication technique using fixed length coding to select the computing node to which task to be assigned in RTDS.

1. Introduction

DNA computing is being established as a viable alternative to solve various NP complete problems. These alternative approaches to performing exhaustive search in solution space are found to be more efficient for various intractable problems. The computer and the DNA both use information embedded in simple coding, the binary software code and the quadruple genomic code, respectively, to support system operations. On top of the code, both systems display a modular, multi-layered architecture, which, in the case of a computer, arises from human engineering efforts through a combination of hardware implementation and software abstraction. A process represents just sequentially ordered actions by the CPU and only virtual parallelism can be implemented through CPU time-sharing. Whereas process management in a computer may simply mean job scheduling, coordinating pathway bandwidth through the gene expression machinery represents a major process management scheme in a DNA. In summary, a DNA can be viewed as a super-parallel computer, which computes through controlled hardware composition. A comparison between the architecture of a computer and a cell, within which DNA resides, has been given in Figure 1.

Nowadays the research effort in the area of DNA computing concentrates on four main problems: designing algorithms for some known combinatorial problems, designing new basic operations of ”DNA computers”, developing new ways of encoding information in DNA molecules and reduction of error in DNA based computations [1]. So, we are inspired to use DNA computing to solve the assignment and scheduling problem in distributed system, which is a NP-Hard problem. In this paper a DNA based algorithms has been proposed for solving the task assignment problem on real time distributed system. To our best knowledge it is the first attempt to solve this problem by molecular algorithms.

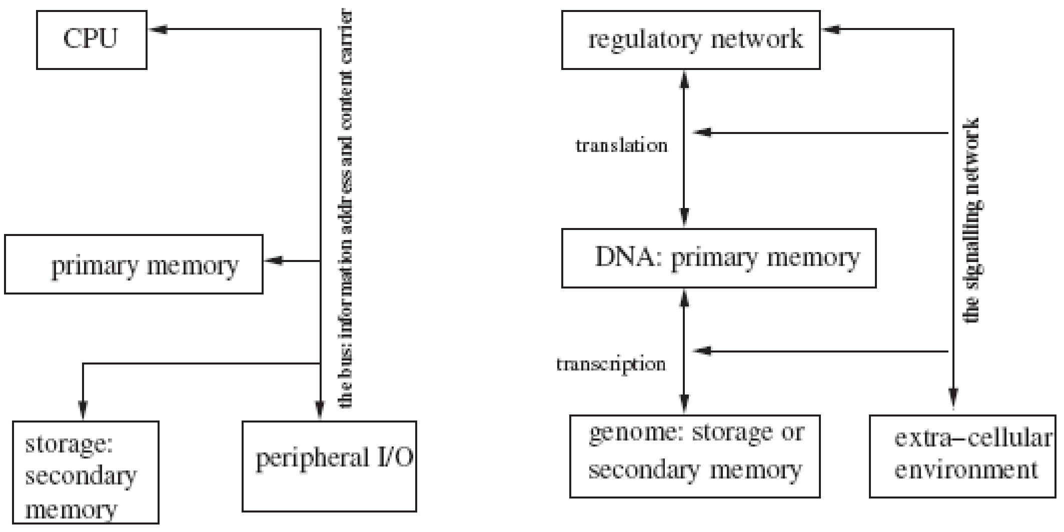
CPU
primary memory
storage: secondary memory
peripheral I/O
the bus: information address and content carrier
regulatory network
translation
DNA: primary memory
transcription
genome: storage or secondary memory
extra–cellular environment
the signalling network

Figure 1: A simplistic schematic comparison of the architecture of a computer and a cell. Note that RNA acts as a messenger between DNA

Major area of evolutionary computing research focuses on a set of techniques inspired by the biological sciences, because biological organisms often exhibit properties that would be desirable in computer systems. DNA computing is one of the evolutionary computation techniques that, uses iterative process based upon various techniques inspired by biological mechanism of evolution. A DNA algorithm is based on the general scheme of an evolutionary algorithm as outlined in Algorithm 1 in the pseudo-code fashion [2].

Algorithm 1. The general scheme of an evolutionary algorithm in pseudo-code

```
BEGIN
    INITIALISE population with random candidate solutions;
    EVALUATE each candidate;
    REPEAT UNTIL (TERMINATION CONDITION is satisfied) DO
        1. SELECT parents;
        2. RECOMBINE pairs of parents;
        3. MUTATE the resulting offspring;
        4. EVALUATE new candidates;
        5. SELECT individuals for the next generation;
    OD
END
```

The proposed algorithm uses this general scheme of an evolutionary algorithm to solve Task Assignment problem in Real Time Distributed computing System

The rest of the paper is organized as follows. The next section discusses basics of DNA computing. *Section 3* describes Real time distributed computing system (HDCS) structure and task assignment problem. DNA Computational Model for task assignment is presented in *Section 4*. *Section 5* includes the proposed algorithm for task assignment based upon the computational model as discussed in previous section. Finally, conclusions and directions for future research are discussed in *Section 5.*

2. DNA Computing

DNA computing is to use DNA as the platform to compute by means of molecular biology techniques for encoding information, generating potential solutions, and selecting and identifying correct solutions [3]. DNA computing, also known as molecular computing, is a new approach to massively parallel computation based on groundbreaking work by Adleman.

DNA (deoxyribonucleic acid) is a double stranded sequence of four nucleotides; the four nucleotides that compose a strand of DNA are as follows: adenine (A), guanine (G), cytosine (C), and thiamine (T); they are often called bases. James Watson and Francis Crick discovered the famous double helix chemical structure of DNA in 1953. It consists of a particular bond of two linear sequences of bases. This bond follows a property of complementarily: adenine bonds with thiamine (A-T) and vice versa (T-A), cytosine bonds with guanine (CG) and vice versa (G-C). This is known as Watson-Crick complementarily. Each DNA strand has two different ends that determine its polarity: the 3.’end, and the 5.’end. The double helix is an anti-parallel (two strands of opposite polarity) bonding of two complementary strands.

DNA computers work by encoding the problem to be solved in the language of DNA: the base-four values A, T, C and G. Using this base four number system, the solution to any conceivable problem can be encoded along a DNA strand like in a Turing machine tape. Every possible sequence can be chemically created in a test tube on trillions of different DNA strands, and the correct sequences can be filtered out using genetic engineering tools. There are four reasons for using molecular biology to solve computational problems.

1. **The information density of DNA is much greater than that of silicon:** 1 bit can be stored in approximately one cubic nanometer. Others storage media, such as videotapes, can store 1 bit in 1,000,000,000,000 cubic nanometer.
2. **Operations on DNA are massively parallel:** a test tube of DNA can contain trillions of strands. Each operation on a test tube of DNA is carried out on all strands in the tube in parallel
3. **DNA computing is an interdisciplinary field where:** biologists, computer scientists, physics, mathematicians, chemists, etc. find a lot of interesting problems which can be applied to both theoretical and practical areas of DNA computing.
4. **Error tolerance capability:** the DNA exhibits much better error tolerance capability, resulting in improved system robustness. Redundancy plays a major role. The DNA have multiple mechanisms to process the same information. If one fails, there are other ways to ensure that crucial cellular processes, such as programmed cell death, are completed [4].

NP-complete problems have an exponential number of solutions, and while it is easy to verify that an instance is a solution, it is hard to find a correct solution. To solve this, a nature inspired technique is helpful that uses biology and chemistry to generate all possible solutions and filter them quickly. This is possible through DNA coding of the problem as it is possible to store 2^{60} bits of information in 1 mL of DNA. This massive parallelism property of DNA can be used to generate and test the possible combinations, that can be the optimal solutions for a NP-complete problem.

3. Real Time Distributed Computing System

A real time distributed computing system (DCS) utilizes a distributed suite of different high-performance machines, interconnected with high-speed real time communication network, to perform different computationally intensive applications that have diverse computational requirements and dead line. Real time distributed computing provides the capability for the utilization of remote computing resources and allows for increased levels of flexibility, reliability, and modularity. Resource management sub systems of the DCS are designated to schedule the execution of the tasks that arrive for the service to meet the specific dead line. A RDCS environments are well suited to meet the computational demands of large, diverse groups of real time tasks.

We have considered the DCS where, the real time tasks are assumed to be independent, i.e., no communications between the tasks are needed. The individual users of the systems are independently submitting their jobs to the central scheduler. The central scheduler operates using a dynamic scheme, because the arrival times of the tasks may be random and some machines in the suite may go off-line and new machines may come on-line.

3.1 Real Time Distributed System Architecture

The real time distributed system architecture is modeled by a graph G=(V, E) where vertices are nodes, and edges are bi-directional point-to-point communication links [8,9].

A node N_i is made of one processor PE_i, one local memory LM_i, and at least one communicator CC_i. A processor PE_i executes Tasks T_i sequentially, reads from and writes data into its local memory MM_i. A communicator CC_i cooperates with another communicator CC_j in order to execute sequentially transfers of data stored in the memory (send or receive) between processors through a link.

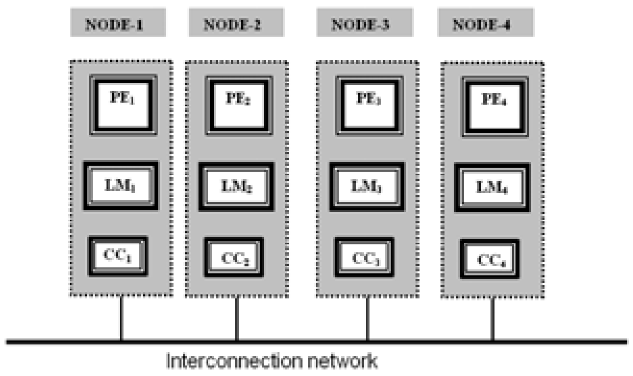

Figure 2. A Real Time Distributed Computing Systems model

Figure 2 gives an example of G, with four nodes Node 1, Node 2, Node 3 and Node 4, and an interconnection network where each node is made of one processor PE, one local memory LM, and a communicator CC. To each processor PE we associate a list of tasks (D_i, C_i=PE_k), where D_i is the deadline and C_i is the worst-case execution time (WCET) of the task T_i on processor PE_i. Since the target architecture is heterogeneous, the WCET for a given processor can be distinct on each processor. Nodes have their own RTOS [10].

3.2 Workload Model

A Directed Acyclic Graph DAG is used to model a real time tasks with dependent subtasks. The basic task model $\prod$ is modeled by a set of N periodic tasks T_i:

$$\Pi = \{T_i = (P_i, D_i, C_i) \mid 1 \le i \le N\} \qquad (1)$$

where P_i is the period of the task. Each task is released after every P_i time units. For non-periodic tasks, P_i is represented by the minimum (or average) separation time between two consecutive releases. The difference in time between the arrivals of two consecutive instances of a periodic task is always fixed and is referred to as period of that task. Although the inter arrival times of instances of a periodic task are fixed, the inter release time may not be. D_i is the relative deadline, the period of time after the release time within which the task has to finish. Tasks can have an arbitrary deadline. C_i is the worst-case execution time of the task at each release $C_i \geq D_i$, i.e. the maximum time span between release of a task and end of the response of that release.

In real time systems it is often necessary to determine an upper bound of time in that the program block is executed. Operations of T_i can be only a computation operation. The task T_i is invoked repeatedly at each period from the sensors. Equation 1 gives an example of T_i. We consider each Task T_i to consist of a set of subtasks, which execute "serially". For convenience, we denote the set of subtasks of task T_i as shown in
We are considering tasks as $C_i \geq D_i \geq P_i$. As we consider on-line real time distribution scheduling heuristics, the arrival, distribution, the execution of operations and communications are time-triggered, each task arrives after a given period.

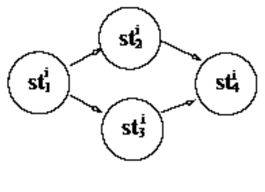

Figure 3. Task Graph for Ti with four subtasks

We consider each task T_i to consists of set of subtasks each having timing and precedence constraint [11] as shown in Figure 3. The subtask in a precedence constraint task st_4^i cannot be executed until the subtasks preceded by it have completed their execution i.e. st_2^i and st_3^i. We have considered different number of subtask for each task and the dependency is different for each task at each invocation at each period.

4. DNA Computational Model

HDCS environments are well suited to meet the computational demands of large, diverse groups of tasks. The problem of optimally mapping (defined as matching and scheduling) these tasks onto the machines of a distributed HC environment has been shown, in general, to be NP-complete, hence DNA replication techniques can be used to find a solution. This approach is based on Adleman's finding to solve the TSP problem using DNA solution technique in 1994 [5]. The technique uses the possible combinations of valid routes among the cities by Marge and anneals the two-solution set. One of the solutions is the cities DNA molecule and the other as edge DNA molecule.

We have use a DNA replication technique using fixed length coding to select the computing node to which task is to be assigned in RTDS as shown in Figure 4. The designated computing node executes the task generated with real time constraints. The approach to this work starts by relating each task on the source computing node as salesman [5] and that after the destination node is selected the task is transferred to that site. Table 1 shows the city and cost fixed length code for five nodes and costs in RTDS [6].

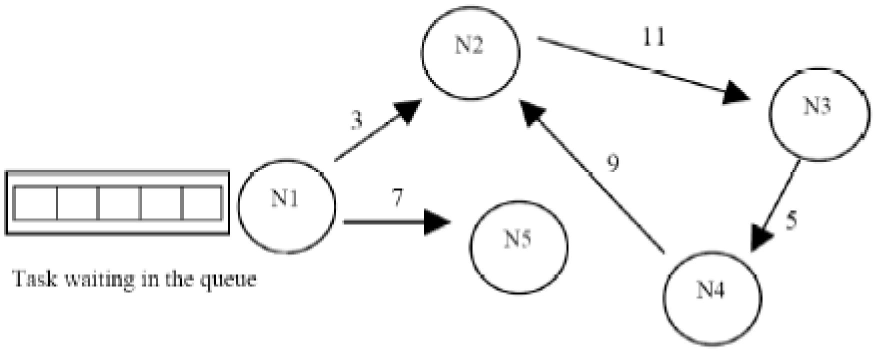

Figure 4: Schematic structure of RTDS

4.1 Encoding

The encoding scheme, which includes the communication cost between a given pair of computing nodes i.e., source and destination is presented in [6,7]. Table 1 shows the encoded values for the given RTDS as shown in Figure 4.

Node	Sequence
1	A—G—T—C—G—G
2	G—T—G—G—A—C
3	C—A—G—T—A—A
4	T—A—T—G—G—G
5	A—A—G—G—C—C
Cost	
3	A—T—G—A—T—A
5	G—G—A—T—A—A

7	T—A—C—T—A—A
9	C—C—T—G—C—A
11	T—T—A—C—C—A

Table 1: Node and cost sequences for the five node RTDS

4.2 Operations

The basic operations associated with DNA algorithm are usually designed for selecting which satisfy some particular conditions. On the other hand there may be different sets of such basic operations. The set of operations used in this problem are:

1. MERGE: given test tube TU_1 and TU_2, create a new tube T containing all strands form TU_1 and TU_2.
2. AMPLIFY: given tube TU create copy of them.
3. DETECT: given tube TU return true if TU contains at least one DNA strand, otherwise return false.
4. SEPARATE: given tube TU and word w over alphabet $\sum_{DNA}$ create two tubes +(TU;w) and -(TU;w), where +(TU;w) consists of all strand from TU containing w as a substring and -(TU;w) consists of the remaining strands.
5. LENGTH-SEPARATE: given tube TU and positive integer n create tube (TU;$\leq$ n) containing all strands from TU which are of length n or less.
6. POSITION-SEPARATE: given tub TU and word w over alphabet $\sum_{DNA}$ create tube B(TU;w) containing all strands from TU which have w as prefix and tube E(TU;w) containing all strands from TU which have w as suffix.

Each of the above operations is a result of some standard biochemical procedure [1,3].

4. Proposed Algorithm for Task Assignment

Tasks in real time systems can be either periodic or aperiodic in nature. In real time systems the functions not just to be logically correct but also to be within deadline. Tasks are said to be periodic when they arrive within a fixed interval of time. The periodic real time tasks can be described by its period, deadline and the worst-case execution time. The proposed task assignment/allocation scheme for RTDS is consists of following four basic steps:

Algorithm: *Task Assignment using DNA*

1. ***Solution Space Generation Phase.*** *Encode each computing node and the cost of communication with 6 base strands and the edge between each node with complementary base. Create copies of them.*

2. ***Select itineraries that start and end with the correct nodes.*** *Select strands, which have start node as the centralized scheduler and end nodes as the possible connected destination node.*
3. ***Select itineraries that contain the least cost of communication.*** *Check the cost of the DNA strand by decoding the code sequence of the cost between the source and destination node.*
4. ***Select itineraries that give feasible schedule.*** *Remove the last six codes of the top strand i.e. node N2. Check the feasibility of sending the task if assigned to it depending on available resources such as available CPU time, deadline constraint of the task which may or may not depend on the period of the task. If the choice is feasible then the assignment is made else go for the next available strand.*

This proposed algorithm is used to design a fault tolerant real-time scheduling scheme for hard real time task. This scheme results multiple feasible allocation schemes, with out computational overhead in comparisons to other iterative evolutionary techniques. The proposed scheme is outlined in Figure 5 (darkened block) as shown in Figure 5. Figure 6 depicts the proposed computation result for transferring tasks from node N1 to node N2.

5. Conclusion

The research in DNA computing is in a primary level. High information density of DNA molecules and massive parallelism involved in the DNA reactions make DNA computing a powerful tool. Tackling problems with DNA computing would be more appropriate when the problems are computationally intractable in nature. The most challenging task is concern with efficient representation of the problem using four nucleotides of DNA. This is always a open problem for the researchers, to represent a problem so that, each iteration yield results solutions for the said problem. Due to its high degree of parallelism in DNA Computing, that can overcome the difficulties arise to solve intractable problem on silicon computers. The proposed ideas and methods show promising results that DNA computing approach can be well suited for solving such real-world application in the near future.

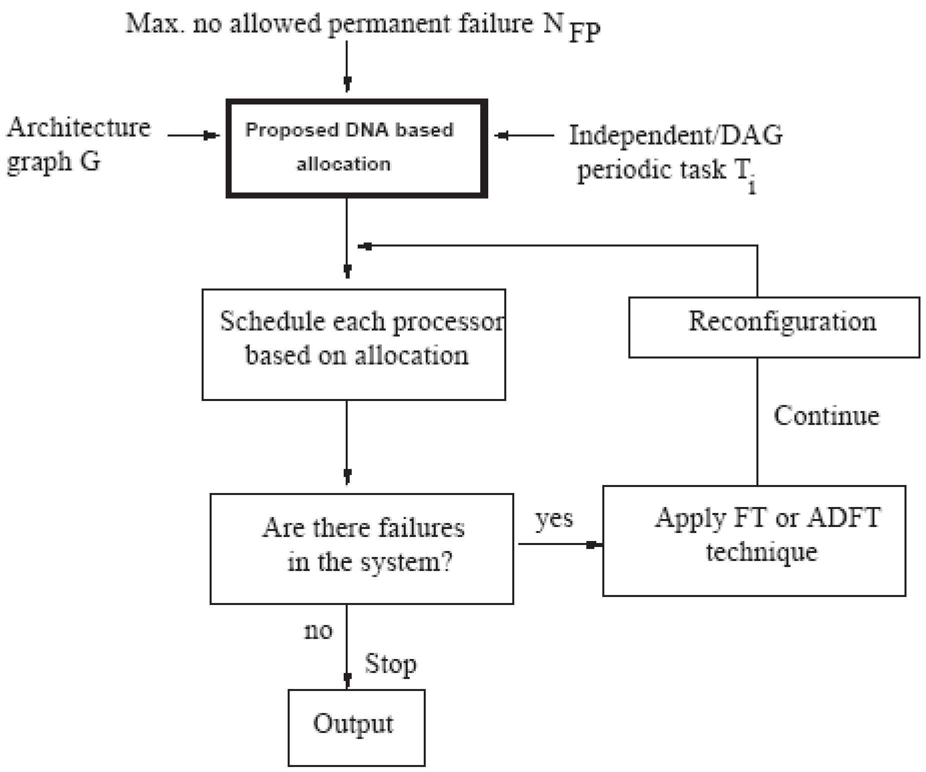
Max. no allowed permanent failure N FP
Architecture graph G
Proposed DNA based allocation
Independent/DAG periodic task Ti
Schedule each processor based on allocation
Reconfiguration
Continue
Are there failures in the system?
yes
Apply FT or ADFT technique
no
Stop
Output

Figure 5: Developing a DNA based RTDS schedule

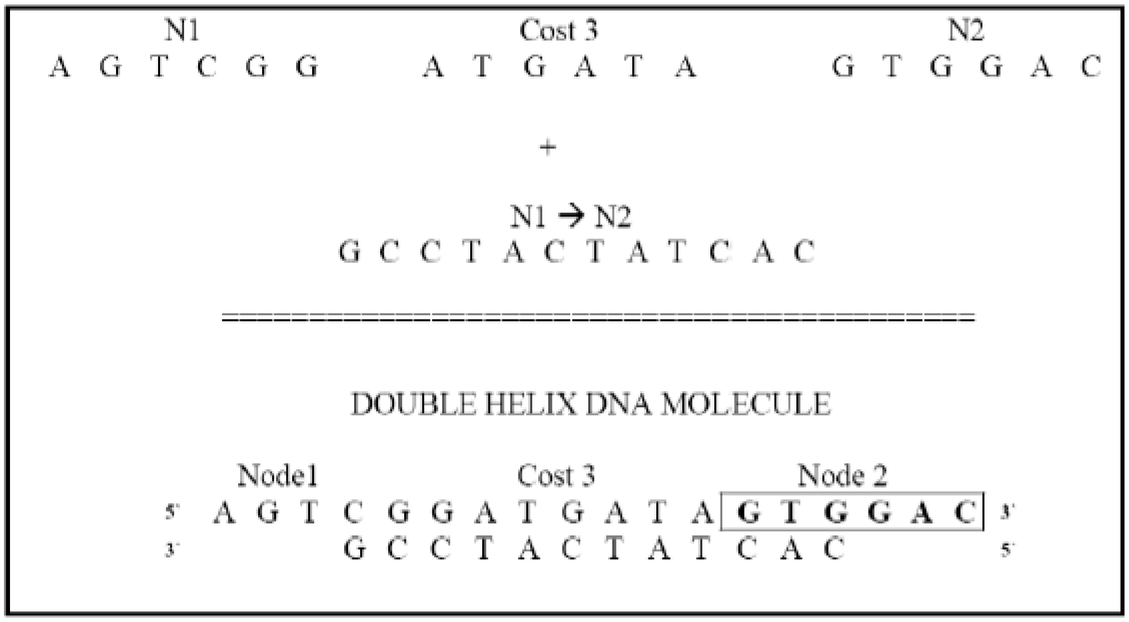
N1
A G T C G G
Cost 3
A T G A T A
N2
G T G G A C
+
N1 → N2
G C C T A C T A T C A C
DOUBLE HELIX DNA MOLECULE
Node1
Cost 3
Node 2
5' A G T C G G A T G A T A G T G G A C 3'
3' G C C T A C T A T C A C 5'

Figure 6: DNA Computation for N1→N2 including the cost.

References:

[1] Piotr Formanowicz. DNA computing. *Computational Methods in Science and Technology*, 11(1):11–20, 2005.

[2] A.E. Eiben and J.E. Smith. *Introduction to Evolutionary Computing*, chapter What is an Evolutionary Algorithm? Natural Computing Series. Springer ISBN 3-540-40184-9, 2003.

[3] Yang ren Rau and Huey jenn Chiang. DNA computing on the minimum spanning tree problem. *International Symposium on Nanoelectronic Circuits and Giga-scale Systems (ISNCGS 2004)*, pages 101–105, 2004

[4] Degeng Wang and Michael Gribskov. Examining the architecture of cellular computing through a comparative study with a computer. *Journal of The Royal Society Interface*, 2(3), June 2005.

[5] Leonard M. Adleman. Molecular computation of solutions to combinatorial problems. *Science*, 266:1021–1024, November 1994.

[6] S.Y. Shin, B.T. Zhang, and S.S. Jun. Solving traveling salesman problems using molecular programming. *Proceedings of the Congress on Evolutionary Computation*, 2:994–1000, 1999.

[7] Ji Youn Lee, Soo-Yong Shin, Tai Hyun Park, and Byoung-Tak Zhang. Solving traveling salesman problems with DNA molecules encoding numerical values. *BioSystems*, 78:39–47, 2004.

[8] Xiao Qin; Hong Jiang and Swanson, D.R, "An efficient fault-tolerant scheduling algorithm for real-time tasks with precedence constraints in heterogeneous systems", Proceedings of International Conference on Parallel Processing, Page(s):360-368, 18-21 Aug. 2002.

[9] Yongbing Zhang; Hakozaki, K.; Kameda, H.; Shimizu, K., "A performance comparison of adaptive and static load balancing in heterogeneous distributed systems", Proceedings of the 28th Annual Simulation Symposium, April 1995 pp. 332 – 340

[10] Rajib Mall, Real-Time Systems, Pearson Education'07.

[11] I. Gupta, G. Manimaran, and C. Siva Ram Murthy, "Primary-Backup based fault-tolerant dynamic scheduling of object-based tasks in multiprocessor real-time systems," Dependable Network Computing, Kluwer Academic Publishers,