Observation on Task Partitioning Using Genetic Algorithm in Parallel/Distributed System

Dissertation submitted to the Jawaharlal Nehru University
in partial fulfillment of the requirements
for the award of the degree of

MASTER OF TECHNOLOGY
IN
COMPUTER SCIENCE AND TECHNOLOGY

By

ARUN KUMAR

UNDER THE SUPERVISION OF

Dr. D. P. Vidyarthi

SCHOOL OF COMPUTER AND SYSTEMS SCIENCES
JAWAHARLAL NEHRU UNIVERSITY
NEW DELHI-110067, INDIA

JULY, 2007

जवाहरलाल नेहरू विश्वविद्यालय

SCHOOL OF COMPUTER AND SYSTEMS SCIENCES

JAWAHARLAL NEHRU UNIVERSITY

NEW DELHI-110067, INDIA

CERTIFICATE

This is to certify that the dissertation titled **"Observation on Task Partitioning Using Genetic Algorithm in Parallel/Distributed System"**, which is being submitted by **Mr. ARUN KUMAR** to the **School of Computer & Systems Sciences, Jawaharlal Nehru University, New Delhi,** in partial fulfillment of the requirements for the award of **Master of Technology in Computer Science & Technology** is a bonafide work carried out by his under the supervision of **Dr. D. P. Vidyarthi.** The matter embodied in the dissertation has not been submitted for the award of any other degree or diploma.

Arun Kumar
Arun Kumar
(Student)

[illegible] 24/9
Prof Parimala N:
Dean
School of Computer & Systems Sciences
JAWAHARLAL NEHRU UNIVERSITY
NEW DELHI-110067
Dean
SC & SS
Jawaharlal Nehru University
New Delhi-110067

[illegible] 17/7/07
Dr. D. P. Vidyarthi
SC & SS
Jawaharlal Nehru University
New Delhi-110067

Dedicated to

My Dear Grandfather…..

ACKNOWLEDGEMENT

I am thankful to all who supports me to do this work. I want to thanks **Dr. D. P Vidyarthi**, my supervisor, who guided me excellently, provided support, guidance, encouragement and raw materials, which are needed to me during the dissertation in past one year. For fruitful discussion and helpful idea I would like to express my gratitude to him.

I am also thankful to Dean, Faculty members and staff of SC&SS for their support and help for project.

I am also thankful to my friend Akhilesh, Manish, Pratibha and Sushil and colleges for their support, help and encouragement during dissertation and difficulties in life in past one year.

Finally I am also grateful to my parents, my lovely younger brother Ashish and sister Suman and family members, for their love, inspiration and encouragements during each phase of dissertation and life.

Arunkumar

Arun Kumar

ABSTRACT

Parallel/distributed system allows the concurrent execution of tasks. Both Parallel and Distributed system are same in execution scenario. The difference between them is whereas parallel systems are single system with multiprocessor and distributed systems give appearance of single system having collection of autonomous computer systems. The main advantages of theses systems are resource sharing, openness, higher throughput and shorter response time, etc.

Task scheduling in parallel and distributed system is done by partitioning the task into sub-tasks and these sub-tasks are executed on different processors or systems. The groups of sub-tasks are allocated to the system and execute. Allocation methods of these task modules are an NP-hard problem.

Genetic algorithm (GA) is used for solving the task allocation problem. GA is adaptive heuristic search algorithm based on the evolutionary ideas of natural selection and genetics. GAs simulates the survival of the fittest among individuals over consecutive generation for solving a problem. Genetic algorithms operators partial matched crossover (PMX) and ordered crossover (OX) are used to solve task allocation problem in parallel and distributed system.

In this dissertation, we focus on partitioning and grouping the task modules in such a manner that these groups of the task can allocated in its entirety to the nodes of the system. The grouping will be done in such a manner so that the execution time difference amongst the modules of the group is minimum. We have tested two crossover operators for this i.e. PMX and OX. The C language is used for the implementation of the whole model.

In first chapter, parallel and distributed system, models of distributed system and execution scenario of parallel and distributed system are discussed.

Second chapter contains scheduling of parallel and distributed system, task preprocessing, and task partitioning and general idea about genetic algorithms.

Third chapter of dissertation contains the proposed model and genetic algorithm operators PMX and OX, which are uses to solved the problem.

Fourths chapter contains the observation of PMX and OX operator and comparisons between them.

Fifth chapter has the conclusion and the future work.

ABBREVIATIONS

ALU	Arithmetic and Logic Unit
BSF	Breadth First Search
CPU	Central Processing Unit
CU	Control Unit
CX	Circular Crossover
DCS	Distributed Computing System
DFS	Depth First Search
GA	Genetic Algorithm
IO	Input/Output
ISO	International Standards Organization
MIMD	Multiple Instruction stream and Multiple Data Stream
MISD	Multiple Instruction stream and Single Data Stream
OX	Ordered Crossover
PE	Processing Unit
PMX	Partial Matched Crossover
RM-ODP	Reference Model for Open Distributed Processing
SISD	Single Instruction stream and Single Data Stream
SIMD	Single Instruction stream and Multiple Data Stream
TSP	Traveling Sales Man Problem

TABLE OF CONTENTS

Chapter 1

Introduction

Parallel and distributed systems are similar in nature as far as execution is concerned. The difference lies in the hardware architecture. A distributed system is a collection of autonomous computers and parallel system is a single computer with many processors. Both have common goal to achieve i.e. the parallelism in system at the different levels. Parallel systems are having three memory architecture; shared memory, distributed memory and hybrid shared distributed memory. The main advantages of these systems are higher throughput and sharing of resources. In Parallel system communication is through memory whereas in the distributed system the communication is done by message passing. [6, 11, 12]

Task scheduling in parallel and distributed system is a challenging area in computer science research. The task scheduling on these system is done in such a manner that the execution time of the task is minimized. Scheduling deals with the allocation of group of tasks (or modules) to the system (CPU or resources). Task is divided into a number of modules (subtask). The modules are allocated to the system for concurrent execution. In the parallel and distributed system, allocation methods of these task modules are the NP-hard problem that is solved using various heuristic algorithms. [13, 14, 26]

Module formation and grouping is an important activity in task allocation. Better grouping of modules will lead to better allocation. We are using genetic algorithms to solve this problem [19]. Genetic algorithms are better search approach to solve an NP-hard problem. Genetic Algorithms works as a natural process of evaluation and is applied to solve problem based on the principle of "survival of the fittest". GA uses reproduction, mutation and recombination to generate solution for the problem. Genetic algorithms generate population with the group of individuals. The individual in population is evaluated; that evaluated individual gives fitness score in the population, higher fitness score gives higher chance to select individual. Mutation operation is also applied on the individual occasionally.

There are many genetic operators for crossover operation. We have used genetic operators partial-matched crossover (PMX) and ordered crossover (OX) to observe the task module formation. Applying these operators we find the better execution order of the module. The method of partitioning the task into number of subtasks is the preprocessing step of the task scheduling in Parallel/Distributed System.

Genetic Algorithms (GA) has been used for the task scheduling in parallel and distributed system since quite some time. We use particularly have observed genetic algorithms crossover operator PMX and OX for the task partitioning and execution of modules in parallel and distributed system environment. PMX and OX operators are based on the exchange operation. [15, 16]

1.1 Parallel System

This is higher speed computing era in which much more fast execution of task is needed. Parallel System is single computer with multiple processors which are used for executing more than one instruction simultaneously. Execution of more than one instruction on multiprocessor is known as parallel processing. In parallel processing task is divided into number of modules (sub tasks) and execute on the processors of the system. The Advantage of parallel processing is higher throughput, computation power and better performance than the single-processor system. These parallel systems are mainly classified into tow categories General purpose and Special purpose computer systems. General purpose computer system is Pipeline computers, Asynchronous multiprocessors and Data flow computers. Special purpose computers are Synchronous multiprocessors (Array Processors), Systolic Array and Neural Network. The main advantages of developing parallel system are to improve throughput, flexibility, reliability and availability. [6, 7, 21]

1.1.1 General Purpose Computers

The general purpose computers are based on Pipeline computers, Asynchronous Multiprocessors and Data flow computers. [6, 21]

Pipeline computing is a technique that is based on the sequential execution of instructions into sub operations. These operations are performed into a specified segment concurrently. There are several pipelines of various purposes. Arithmetic pipelining: the ALU of a computer can be segmented for pipeline operations in a variety of data formats. Instruction Pipelining: the execution of set of instructions can be pipelined by overlapping the execution. Processor pipelining: the same data can be processed by cascade of processors. Depending upon functionality there are two type of pipelining Unifunctional Pipelining: A pipeline with set and committed function is unifunctional pipeline. Multifunctional pipeline: may perform different types of functions. Some other categories are Linear pipeline: linearly performed with fixed operations over a instruction stream flowing to one end point to another; Dynamic pipeline: designed to perform different type of variable function and it allows to feed forward and feed backward connections so that known as Nonlinear pipeline; Scalar pipeline: designed for operating scalar operations; Vector pipeline: It handles vector instructions. Pipeline throughput is defined by the average number of task instructions generated in per clock cycle and pipeline efficiency is related to throughput of pipeline which is the collection rate of all stage utilization of pipeline. Stage utilization is the percentage time used to perform a task. [6, 21]

In **Asynchronous Multiprocessor system** all CPU works independently and task execution is done by partitioning the task into number of subtasks and allocated to different CPUs. These sub-tasks are having low communication and synchronization requirements. In asynchronous system communication and synchronization is done by shared memory and message passing. Shared Memory Multiprocessors, Message Based

Multiprocessor and Hybrid Approach are the possible asynchronous multiprocessors. [6, 21]

In **Data Flow Computers,** basically execution is done if data is available. This method is derived by data-driven. In data flow computers, the execution data not follow any order rather depends on the availability of data operands. Data are seized in instruction instead of storing in shared memory. [6, 21]

1.1.2 Special Purpose Computers

The special purpose computers are Synchronous multiprocessors (Array processors), Systolic Arrays and Neural Networks. [21]

Synchronous Multiprocessors (Array Processors) are collection of multiple ALU's. These ALU's are called processing elements (PE), which can work in parallel in lock-step fashion. An ALU, register and a local memory is part of a processing element (PE) and each PE connected with data routing network. All PE's are connected to control unit (CU), which control all operation of PE's. [6]

Systolic Arrays are special purpose computing based on multidimensional pipeline and mapped on fixed architecture. This is specially designed for matrix multiplication. [21]

Neural Network is different from conventional computer. It is like human brain and is highly complex, nonlinear and parallel information processing. The main advantages of neural network are nonlinearity, adaptivity, IO mapping, fault tolerance and uniformity analysis and design, etc. [21]

Parallel computers are based on three type of memory architecture: Shared Memory, Distributed Memory and Hybrid Distributed Shared Memory.

The key term in parallel processing is efficiency of parallel processing which is derived by speedup(S). Suppose that in parallel system there is N number of processor then speedup of parallel system is [6]

$$S = \frac{\text{execution time for one processor}}{\text{execution time for N processor}}$$

1.2 Distributed System

Distributed Computing System (DCS) is the collection of autonomous computers which give the appearance of a single computer system. Distributed system is collection of multiple systems but components are not shared among systems. In computer network a component which communicates and coordinates their action by passing messages is called a distributed system. This defines following characteristics of distributed systems; concurrency of component, lack of global clock and independent failure. Concurrency of component means concurrent execution of any application in the system if there is any need to share any resource for execution then it is allowed. Distributed systems synchronize their clock when communication and coordination is needed. Then there is no need for global clock. In the Distributed system, there is independent machines connected to each other and they do not affect functionality of any other when any application is executed. [11, 12, 26]

The goals of distributed system are resources sharing, heterogeneity, openness, security, scalability, concurrency, fault tolerance and transparency. [11, 12]

Resources Sharing is main goal of distributed system which means shares any resource of system which is hardware, software or any data which resides where any in the system. Resources sharing also provide interaction among users of the system. Hardware resources such as CPU, printers and data resource are files, database. The sharing of resource is also economically advantageous due to reduction of cost of devices. A computer access physically encapsulated resource of other system by message passing. A resource manager provides a reliable and consistent access and update of resource by enabling communication interface. Resource manager is a program having responsibility of sharing resources and provides proper communication among recourses. The client-server and object based model of resource sharing give guidelines for how to provide and use resources. [11]

Heterogeneity defines the distributed system computers are having different type of architecture, application program, operating system, run time environment, network and network protocols, but it is masked by middleware.[11]

Openness is the characteristics of distributed worried about the ways of re-implementing and extend the system. The system is governed by a set of protocol which illustrate the syntax and semantic of system service. Flexibility of system is most important which shows the components of system are from different configuration and also developed by distinct developers, when they are added to system or replaced to old one that is simply adapted by the system, that component is a hardware or software or any file system.[11]

Security is an important in distributed system, because in distributed system data is transmitted among different systems, so that there is need to keep some data secret when it is transmitted. Encryption can be used for the securing the data from intruders. [12]

Scalability is term which is used for defining the system adaptability means if there are many users and component of are increased in system then it is also work successfully. Controlling the cost of physical resource maintains the performance of system and prevents software resources running out are the major challenges arises to design a scalable system. [12]

Concurrency defines the concurrent execution of process in system, in distributed system resources are shard among the computers, they are requested for the resource some time it may be happen two or more computer are request for same resource, to avoid such circumstances algorithm are developed and it ensures that execution is in concurrent environment.[12]

Fault Tolerance means to avoid the failure of any system. Distributed system contains number of components (which are hardware or software) there is interrupt of fault cause by any component, to avoid such failure of component some techniques can be used. These techniques are detecting failure, masking failure and other one is tolerating failure. Recovery from failure and redundancy are the techniques which are used to accomplish the fault tolerance in distributed system. [12]

Transparency is defined to distributed system seem as a single system rather than collection of number of cooperative autonomous system. The ANSA Reference Manual [ANSA 1989] and the International Standards Organization's Reference Model for Open Distributed Processing (RM-ODP) [ISO 1992] gives following transparency: Access transparency, Location transparency, Migration transparency, Relocation transparency, Replication transparency, Concurrency transparency, Failure transparency and Persistence transparency. [11]

1.2.1 Distributed System Models

The basic models of distributed system are:

- Minicomputer Model- In minicomputer model of distributed system number of minicomputers are connected to a network, each with some terminals.
- Workstation Model- Many work station are connected to network and make distributed system to useful when user uses remote workstation.
- Workstation server Model- Its is very advantageous due to sharing of several servers like file server, printer server etc.,
- Processor Pool Model- Pool of processors connected to network. [11, 12]

1.3 Execution Scenario in Parallel and Distributed System

In the parallel and distributed system the task execution is done by partitioning the task into number of sub-tasks and these sub-tasks are allocated and executed on the nodes of the system. The crucial issue of partitioning is based on the grain size. [6]

1.3.1 Grain Size

For any task execution there is some software computation involved. This computation contains basic segment of parallel processing and are measured by grain size (granularity). Grain size defines fine, medium and course grain.

Fine Grain is the lowest level grain size which contains less than 20 instructions in the grain. Some times fine grain size varies up to thousand which depends upon the program.

Medium Grain contains procedures and subprograms up to 2000 instructions. In MIMD execution mode medium grain size is involved. **Coarse Grain** corresponds at parallel execution of job and it contains ten thousand or more instruction in a single program. Message passing in system uses medium and coarse grain. [6, 22]

1.3.2 Latency

For any program execution in parallel and distributed system there is the communication between computers. The time taken is measured by latency. **Memory Latency** is time taken to access memory by processor and **Synchronization Latency** is time taken to synchronize to two processors with each other. [6]

1.4 Organization of the Dissertation

The thesis is organized in five chapters.

First chapter of our dissertation describes parallel system, distributed system, and execution scenario of parallel and distribute system.

Second chapter gives the idea of task preprocessing, task portioning and genetic algorithms.

Third chapter define proposed work and genetic algorithm operators PMX and OX.

Fourth Chapter contains observation using PMX and OX operators.

Conclusion and future work is given in Fifth Chapter.

Chapter 2

Scheduling in Parallel and Distributed System

In parallel computing environment non-preemptive scheduling is a NP-Hard problem and gives good throughput. But most distributed/parallel processing use static scheduling but dynamic scheduling is very important and natural for such system. [23, 25]

Three cases arise in the scheduling in the parallel/distributed system. Let us suppose that there are m processing nodes and n tasks. If the numbers of tasks are larger than the number of processing nodes then, for optimal execution, the task execution time is known in advanced. Though, practically it is not possible for all the cases. The second algorithm is Round Robin algorithm also known as distributed, fault tolerance round ,robin algorithms. Here the number of processors are less than the number of tasks, and the execution is done according to there time quantum. For each time quantum a task is scheduled according to its priority. Third algorithm is Preemptive Task Bunching. In this n tasks are bunched and assign to all m processors. For coarse-grained and fine-grained task, preemptive task bunching algorithm works well. [1, 10, 24, 26]

2.1 Static scheduling

In static scheduling, grain determines optimize scheduling. To schedule a task following steps are required [6].

- Construct a graph according to grain value.
- Fine grain computation is scheduled in processor.
- Coarse grain is produced by grain packing.
- On the basis of grain packed graph, generate parallel scheduling.

For static scheduling most of the scheduling parameters are known in advance i.e. prior to make the scheduling decision.

2.2 Dynamic Scheduling

In real scheduling scenario, most of the parameters are not known and the decision is differed at the time of execution. Dynamic scheduling takes place for such cases. Tasks used to enter and exit the system dynamically. At any point of time, the decision can be taken considering the availability at that particular time.[6]

In parallel and distributed system some other scheduling algorithms are Adapted First Come First Serve (AFCFS), AFCFS-Blocking of Sequential Jobs (AFCFS-BS), Largest Gang First Served / Shortest Sequential Job First Served (LG-SS) and .LG-SS-Blocking of Sequential Jobs (LG-SS-BS).[1, 10, 24]

2.3 Task Preprocessing

Preparation of the raw data for taking further decision is known as task preprocessing. There are many approaches to preprocess data. Task preprocessing is required to extract the information before allocating a task on parallel/distributed system. The extracted information helps to take further decision for allocation. In the task preprocessing there are two observations: task creation and task synchronization. [2, 22]

Preprocessing is a method to analyze the task for mapping to processing nodes and specify an order to follow to execute these tasks in a synchronized way. There are mainly two types of preprocessing compile time and run time. [22]

In compile time preprocessing, the analysis and information collection done at compile time. In runtime preprocessing approach two steps are involved. First to analyze the data, optimize it, and generate information for second step. In second step actual computation is performed. There are four phases in preprocessing. It is sensing, recognition, decision and acting as depicted in fig. 2.1. [7]

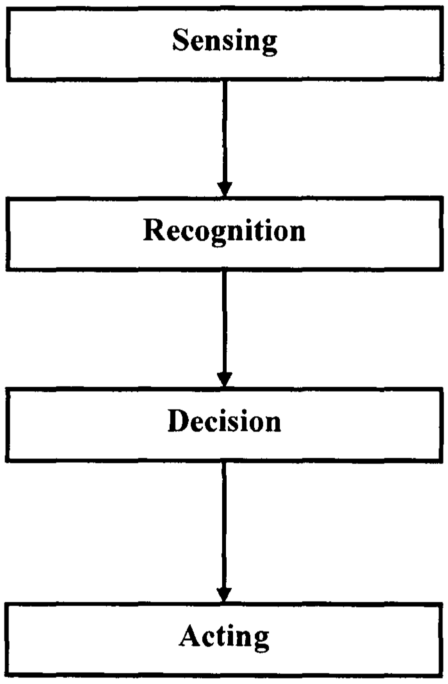

Fig 2.1 The Preprocessing Phase

2.3.1 Component of Preprocessing

There are three components of preprocessing given as follows:

Controller is the task data manager and their responsibility is to collecting the information from different components and distribute among the different components. [7]

Authenticator is responsible for unauthorized access. It checks the policies and protocols used to process the data. Correct data transportations is also the responsibility of the authenticator. [7]

Obfuscator work is to store information. How to store the data and in which manner it transfer. [7]

2.4 Task Partitioning

In parallel/distributed system environment, partitioning of the task into number of sub-modules (sub-tasks) is required. It is so, so that not the task as a whole, rather the modules of the task becomes the execution entity. Decomposition, mapping and tuning is three required steps to program execution in parallel/distributed computing system. [5, 6]

2.4.1 Perfect Decomposition

In parallel processing the task is divided into number of sub-tasks (modules) and these modules do not require a communication or a little communication among each other. This is called Perfect decomposition. In this decomposition technique, least effort of programmer is required and it balanced execution gives almost 100% efficiency. Perfect decomposition condition not created when dependency between processes exist. The π composition is a good example of a perfect decomposition in which only the partial sums need to be communicated. [6]

2.4.2 Domain Decomposition

The regularity of data structure is the main feature of domain decomposition. Static Data Structure, for example, matrix factorization for solving a large finite difference problem on a system with a regular network topology. Dynamic Data Structure tied to a single entity, for example, in a many body problem, subsets of bodies can be distributed to different nodes. Fixed Domain with Dynamic Computations with various regions of the domain, for example, a program those modules fluid vertices, where the domain stays fixed but which pools move around; are three types of problems recognized for domain decomposition. [6]

Three major steps are specified below to decompose the domain of a given applications:-

- At various nodes sub domain of data are distributed.
- Restrict the computation so that each node updates its own sub domains of data.
- Put the communication in node programs.

2.4.3 Control Decomposition

When the domain of data is not suitable for domain decomposition and irregular behavior of data structure and domain is found, then control decomposition is used. Control decomposition technique is used for artificial intelligence and symbolic processing problem. Functional decomposition and manager-worker approach gives idea about control decomposition. [6]

- Functional Decomposition: -Computation and data structure is the key for the control decomposition. Due to dependency on data structure and computation, interface between different functional modules is needed.

- A Manager- Worker approach: - It is based on divide and conquers method. Divide the application task into number of sub tasks which are not of the same size, and one sub task behaves as manager and others like worker. Manager performs dynamic load balancing to assign tasks to workers to improve the performance of the system.

For the distributed computing system the combination of object oriented programming with message passing technique gives an object decomposition technique. This technique is beneficial for the parallel computers also and provide higher granularity and avoids use of global variables. There is a layered decomposition technique, which uses different decomposition at different layers for parallelism. [6]

2.5 Genetic Algorithms

Genetic algorithms (GAs) are evolutionary algorithm in which there is best solution is chosen from the number of solutions. There are many search techniques available for problem solving; these search techniques are [3, 4, 15, 16, 17].

- Calculus Base Techniques (Fibonacci, Sorting)
- Enumerative Techniques (DFS, BFS, etc)
- Guided Random Search Techniques (Hill Climbing, Simulated Annealing, Evolutionary Algorithms, etc.)

Evolutionary Algorithm is basically Genetic Programming and Genetic Algorithms. Genetic algorithms are generating number of population of solution and evaluate best solution for the problem. GA is introduced by John Holland (1975) and his student DeJong (1975) at the University of Michigan. The goal of this development is to give details of natural system in thorough and preserve the mechanism of natural system by a software designed for artificial intelligence system. GA uses the biological concept of "Natural Selection" and "Genetic Inheritance" which is given by Darwin in 1859. "Survivals of Fittest" is the best definition of GA. GA is beneficial where we have little knowledge about problem solving approach, mostly for NP-Hard problems (e.g. TSP). A genetic algorithm is based on regeneration and selection operations. These operations help to generate new search population and apply evolution function on generated initial population to find new final population. GA is different than some other search techniques described as follows [15].

- GA mechanism applies on the coding of the parameter sets not only on the parameters.
- Populations of points are used in searching in GA. It is not a method to use single point for searching.
- The basic primary (or original) information is used in GA, no other information like secondary knowledge and consequential information is used.
- GA use probabilistic transition operator instead of deterministic transition operator.

GA is based on the resemblance of genetic structure and behavior of chromosomes within population of single units. A Chromosome structure (genotype) is sequence of genes and genes are the fundamental instructions for building an organism. The complete chromosome derived the solution of problem with the help of particular chromosome (phenotype). Phenotype is an organism made by genotype which contains all the information and take part to generate fitness function. [4]

2.5.1 Search Space

The feasible solution of population of individuals in GA is maintained in search space. In search space each individual is represented by finite length vectors or variables. These are any alphabets but we use binary number alphabet {0, 1}.An individual of population participate to generate fitness score of each solution. The most favorable fitness score is required. GA maintains population of 'n' chromosomes with there fitness score. The parents are able to select their mate on the basis of their fitness score and use reproductive map to produce offspring. In reproduction highly fit fitness score solutions are given more opportunity and they arrive to replace old one in population which is least fit among reproduced solution. [9, 16]

New generation of solution during reproduction gives better genes than previous solutions and more successive generation produce better partial solution than previous generation gives.

2.5.2 Genetic Algorithms Components

Genetic algorithms have following four basic components [17]

- **Selection:** For the reproduction, the mechanism of selecting an individual according to their fitness value.
- **Crossover:** represents mating between individuals, so that two parents produce good children.
- **Mutation:** the random modification of genetic population.
- **Sampling:** Generation of new offspring from old offspring.

2.5.2.1 Selection

The Key idea behind selection operator is to give preference to better individuals and integrity of individuals is dependent on there fitness score. Generation of next genes is dependent on their fitness score. Suppose that there are four populations A, B, C and D having their fitness score 4, 6, 2 and 8 respectively. The roulette wheel for selection these populations are as given in Fig. 2.2. [3, 16]

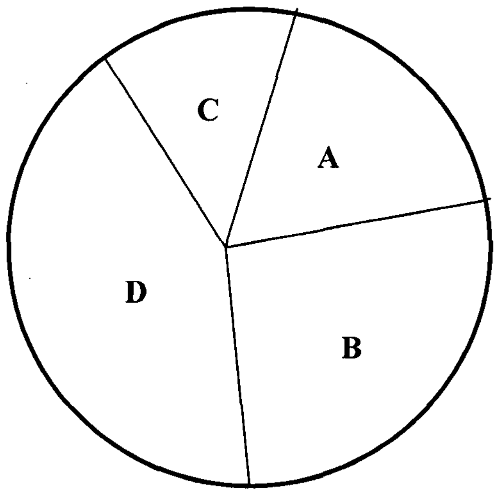

Fig2.2 Roulette Wheel representation of Selection Operation

2.5.2.2 Crossover

The crossover mechanism is like real world sexual reproduction when genetics of two parents are mixed. Then the chromosomes is split and merged and the genes of child are mixture of genes of parents. Crossover operators are categorized in Fig. 2.3. [8, 9, 14]

Parent1:	**11111111111111**
Parent2:	**00000000000000**
Child1:	**11111110000000**
Child2:	**00000001111111**

Fig2.3 Crossover Operator

1-point crossover is the methods in which the chromosomes are cut at a selection cut point and exchange the genes on that point. **2-point crossover (Multi-point crossover)** crossover exchange genes like 1-point crossover but here two selection cut points (Multi selection cut points). **Uniform crossover** is process of masking, where random crossover masks are chosen and copy genes of parents to exchange in parents to generate child.

2.5.2.3 Mutation

Mutation is important through applied with low probability. It provides search space when locality of population is very large. Mutation is responsible for reproducing genes and preventing them. Mutation combine with selection gives a hill climbing search algorithm. [9]

2.5.2.4 Effect of Operators

The selection operator generates the better population from the old population and combined with crossover operator it produced well but sub-optimal solution to population. The combination of selection and mutation operator creates parallel, noise-tolerant and hill climbing algorithms for solve the problem. Hill climbing algorithms produce set of solutions and then optimize solution again and again then produce good solution. Lonely mutation operator produces a random walk search space. [8,14]

2.5.2.5 Fitness Function

A fitness function is the objective function, which is to be optimized using the GA.

Genetic Algorithm ( )

1. Randomly generate the population of solutions
2. Evaluate the fitness of each individual against the fitness function.
3. While termination condition does not hold
    - Replicate individuals based on their fitness
    - Transform the individuals in the population

i. Randomly pick two parents from the population
ii. Crossover the parents (pick a random point in the strings and exchange their top parts) to produce offspring
iii. Mutate each offspring (randomly decide whether to flip each bit in the string) optionally flip each bit)

- Evaluate the fitness of each new individual

TH-14051

Chapter 3

The Proposed Model

The model proposed in this work partitions the task into modules and groups it so that it become such an execution entity that groups the modules making it almost the equal load. It is assumed that execution times of module and communication among the modules are given. The task partitioning problem takes the task graph and its different modules with their execution time as input. Different modules are grouped such that the differences of the sum of the execution times of groups are minimized. [19, 27]

In the proposed problem for the n modules; each is represented by a distinct integer number from the range {1, 2, ---- n}. These are partitioned into k groups of equal or unequal length. The idea, borrowed from the traveling salesman problem (TSP), is applied. Two types of crossover PMX (Partially Matched Crossover) and OX is applied. As observed PMX is most suited crossover for this type of problem although order crossover (OX) is also applied. The Evaluation function is as follow [27]

Let us assume that there are 'm' modules in a group and e_j is the execution time of jth modules. S_i is the sum of the execution time of this ith group then

$$S_i = \sum_{j=1}^{m} e_j$$

Difference in sum of the execution times of ith and jth group is

$D_{ij} = S_i - S_j$

Where i= 1, 2, -----------k

And j = 1, 2, 3--------k - j

These differences should approach to zero. The operators are applied until the evaluation function reaches almost to zero.

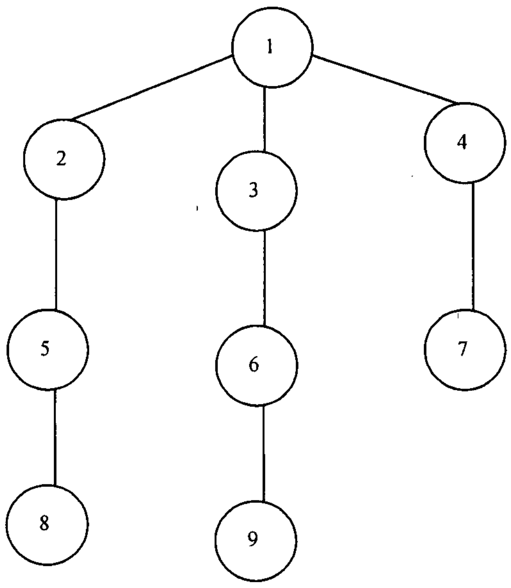

Fig. 3.1 Task Graph

3.1 Partitioning

As the task partitioning is an NP-class of problem, we use GA for task partitioning. Often the problem is given in a form of a task graph shown Fig. 3.1. Depending on the length of the problem there may be number of modules in a task graph. At one end each instruction of the problem is treated as one module whereas at the other end the whole task may be assumed to be one module. For the latter, which is also known as coarse grain, the inherent parallelism of the Parallel / Distributed system can not be utilized. In the former (fine grain) the number of modules may be, uselessly, quite large. Moreover different modules in the fine grain may require each other for the execution i.e. there may be the precedence among the modules of the task. [18, 20] Partitioning involves grouping of these fine grain modules.

3.2 Partial Matched Crossover (PMX) Operator

The PMX (Partial Matched Crossover) is developed by Goldberg, Davis Lingle and Smith to describe the construction of reordering operators that combine features of inversion and crossover into a single operator. In PMX there is two crossover points which define matching section for the population and position by position exchange operation of genes (string site unit) completed. [15]
Let us assume for the two initial populations

P1 = (1234 / 567 / 89)

P2 = (1258 / 369 / 47)

PMX would produce offspring in the following way. First the sequence between the end points is swapped.

O1 = (* * * * / 369 / **)

O2 = (* * * * / 567 / **)

At present * is unknown (represent holes) and there is defines a series of mappings

$3 \leftrightarrow 5, 6 \leftrightarrow 6, 9 \leftrightarrow 7$

We can further fill other objects from the original parent for which there is no conflict.

O1 = (12 * 4 / 369 / 8*)

O2 = (12 * 8 / 567 / 4 *)

Finally the third object in O1 which should be 3 but due to conflict will be replaced by 5. It is because of mapping 3 $\leftrightarrow$ 5. Similarly the last object 9 will be replaced by 7.

O1 = (1254 / 369/ 87)

O2 = (1238 / 567 / 49)

3.2.1 Algorithms for PMX

1. *Take initial populations and two crossover points.*
2. *Replace bits with holes which do not reside between crossover points.*
3. *Swap the bits of offspring residing between crossover points.*
4. *Fill the holes created in step 2 with the help of initial population offspring (parent) with no conflict.*
5. *Fill remaining holes with the help of exchange operation.*
6. *Find new populations.*

3.2.2 Flow Chart for PMX

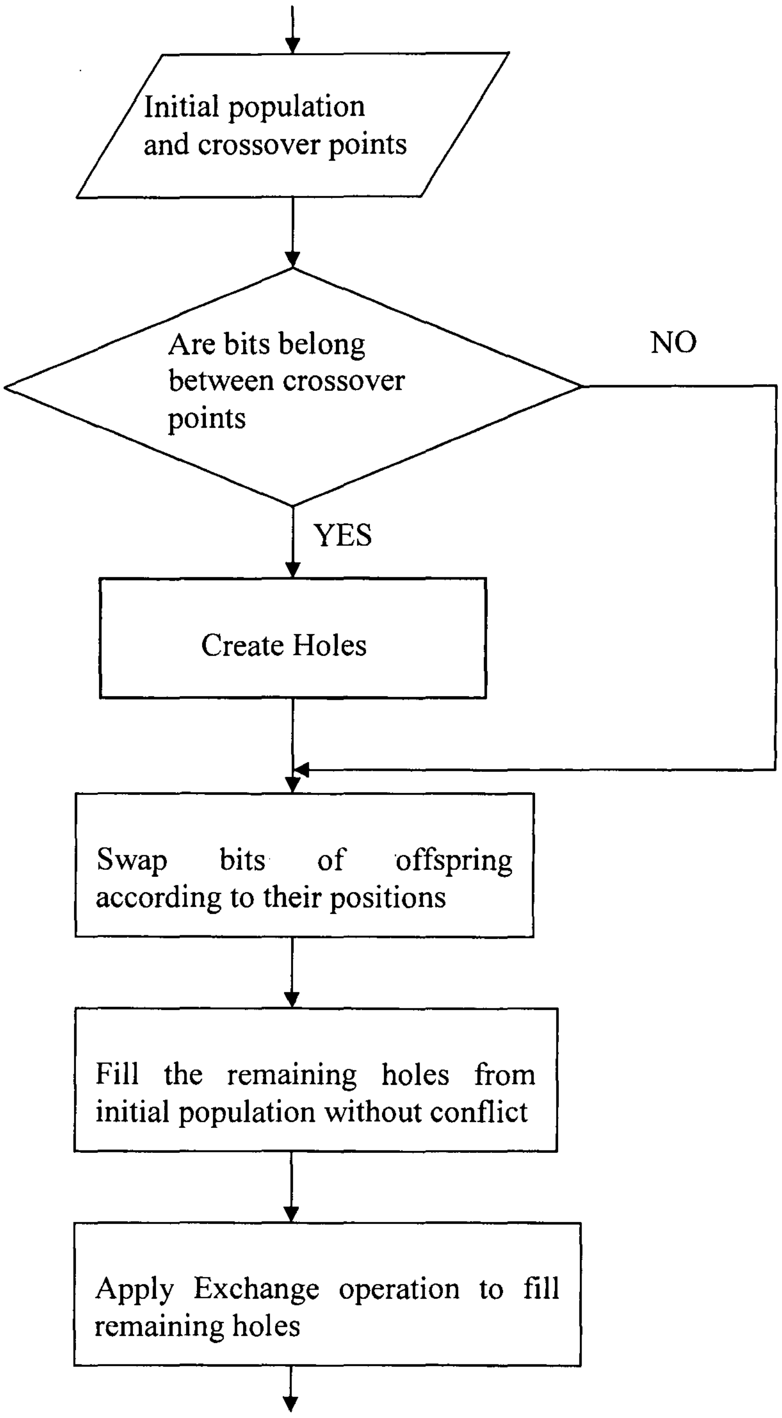
Initial population and crossover points
Are bits belong between crossover points
NO
YES
Create Holes
Swap bits of offspring according to their positions
Fill the remaining holes from initial population without conflict
Apply Exchange operation to fill remaining holes

Fig. 3.2 Flow Chart of PMX Operator

3.3 Ordered Crossover (OX) Operator

The order crossover operator starts of in a manner similar to PMX. Starting with the example strings p1 and p2 used to illustrate PMX; we select a matching section [15]

P1 = (1234 / 567 / 89)

P2 = (1258 / 369 / 47)

Like PMX, each string maps to constituent of the matching section of its mate. Order crossover uses a sliding motion to fill the holes left by transferring the mapped positions. When P2 maps to P1, the points 5, 6 and 7 leave holes in the offspring.

O1 = (1234 / *** / 89)

O2 = (12*8 / 3*9 / 4*)

Start with first value of second string 1 fills hole after 4 by sliding motion in circular order and then hole created at place of 1 that hole is fill by 2 to slid.

O1 = (1234 / *** / 89)

O2 = (2**8 / 3*9 / 41)

The holes of string are filled by sliding motion of individuals.

O1 = (1234 / *** / 89)

O2 = (2389 / *** / 41)

The holes are then filled with the matching section from mate. Performing this operation and completing the complementary cross we obtain the offspring O1 and O2 as follows:

O1 = (1254 / 369 / 87)

O2 = (2389 / 567 / 41)

3.3.1 Algorithms for OX

1. *Take initial populations and two crossover points*
2. *Choose bits which reside between crossover points of one offspring and then replace matching bits by holes of offspring (populations).*
3. *Create holes between crossovers points in offspring and with help of circular left shift operation fill holes of offspring which not belong between crossover points.*
4. *Repeat process 3 until all holes of offspring not belong between crossovers points are not fill.*
5. *Fill holes between crossover points with reaming unused bits without conflict.*
6. *Find new populations.*

3.3.2 Flow Chart for OX

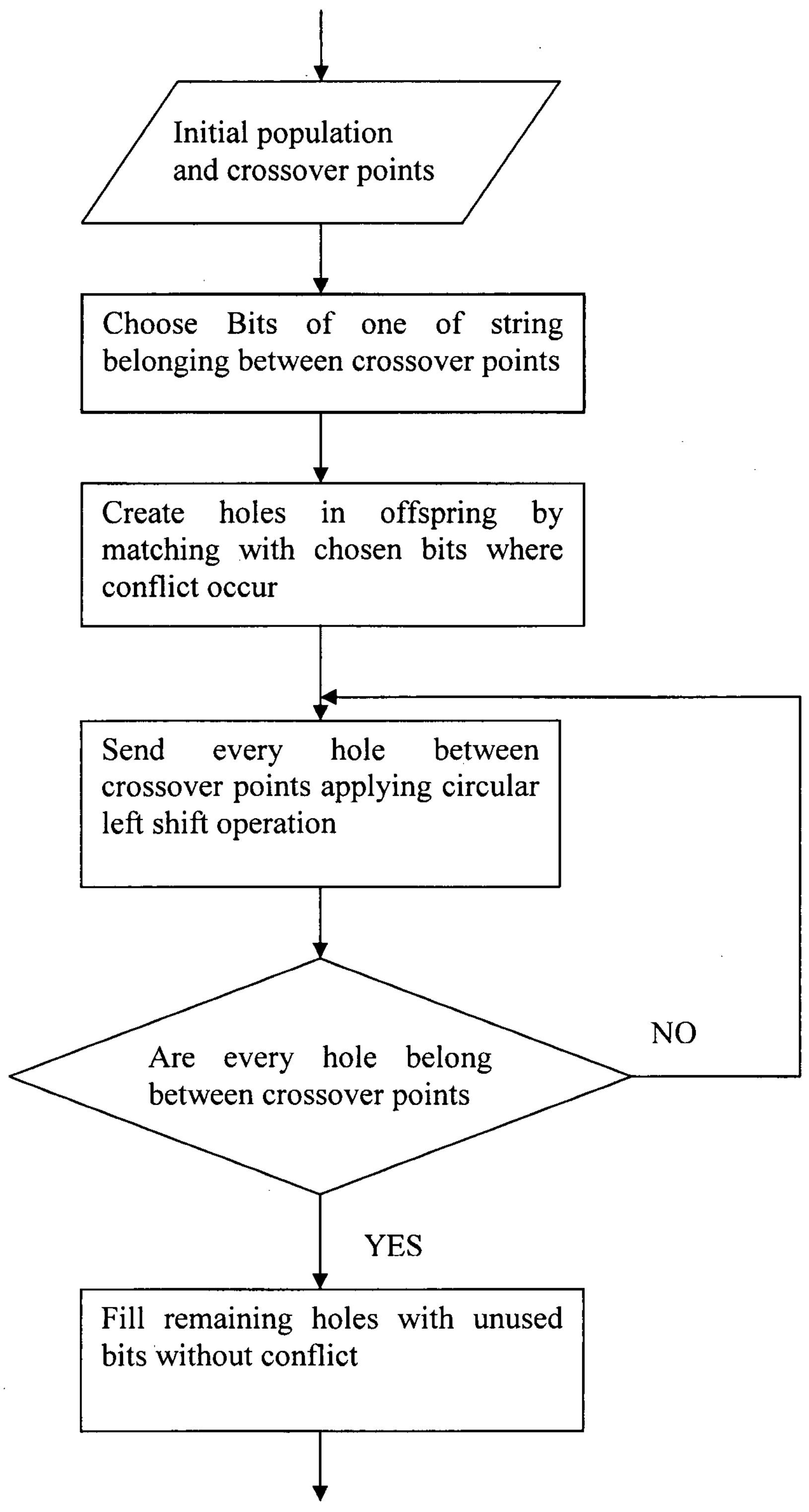
Initial population and crossover points
Choose Bits of one of string belonging between crossover points
Create holes in offspring by matching with chosen bits where conflict occur
Send every hole between crossover points applying circular left shift operation
Are every hole belong between crossover points
NO
YES
Fill remaining holes with unused bits without conflict

Fig 3.3 Flow Chart of Ox Operator

Chapter 4

Observations

In our observation we take number of modules as input apply crossover operators and produce output in terms of grouping and difference in execution time(among groups). The PMX and OX operator are used to generate the output and up to 50 numbers of modules are taken as input starting from 5 with the difference of 5. These inputs results in different outputs.

4.1 Observation Using PMX Operator

We have given the number of modules in a task as the input. Also the execution times of the modules have been generated randomly and are in the range of depending on the number of modules in that task. The observation of PMX operator is given in Table 4.1. The first column shows how many number of module are taken as the input and second column shows the output with two sub columns showing the module group formation and the difference of execution time (among groups).

INPUT	**OUTPUT**	
Number of Modules	**Groups of Modules**	**Difference of Execution Time (D) (among groups) (in msec)**
5	**1 / 5 2/3 4**	**0**
10	**1 10 5 4 /2 6 8/ 7 3 9**	**5**
15	**11 6 15 2 5 14/ 4 1 10 12 3/ 9 13 8 7**	**6**
20	**8 16 15 7 6 13 3 19 / 2 20 4 1 11 12 10/ 14 17 18 9 5**	**3**

25	10 16 11 2 1 15 19 22 17 18 23/ 24 12 14 5 20 13 6 21/ 25 8 3 4 9 7	2
30	22 29 9 26 27 5 12 6 18 20 16 23 7 25 /8 3 13 19 11 4 24 15 2 / 17 1 21 10 28 30 14	11
35	21 1 7 23 25 34 9 4 3 20 5 29 14 27 17 31 6 /12 33 15 16 8 19 28 30 11 32 2 24 / 35 22 13 26 18 10	9
40	33 4 10 39 2 27 1 17 12 3 11 34 13 15 32 37 26 18 23 /38 19 25 20 16 29 31 35 6 9 22 40 14 36/ 8 30 24 28 21 7 5	6
45	29 43 34 13 7 8 20 41 1 18 21 27 40 16 37 4 22 15 45 24 35 11 / 26 33 14 10 31 3 38 19 44 32 6 17 23 25 28 / 2 12 39 36 9 5 42 30	4
50	34 14 50 17 45 12 13 32 24 44 23 47 40 27 41 4 26 36 3 42 8 25 30 20 /2 19 10 15 35 22 43 49 18 7 5 16 31 33 39 38 21 / 11 28 29 6 1 9 37 46 48	13

Table 4.1 Observation Using PMX Operator

When input number of module is 5 then output difference of execution time among groups is 0, it is according to our model which gives the difference of execution time among groups of string. When the input string changes the output varies it may be increase and decrease also. We see that when input value is 10 and 15 output D is 5 and 6 respectively it shows increasing value. When input is 20 then value of output D reduces and gives output 3.

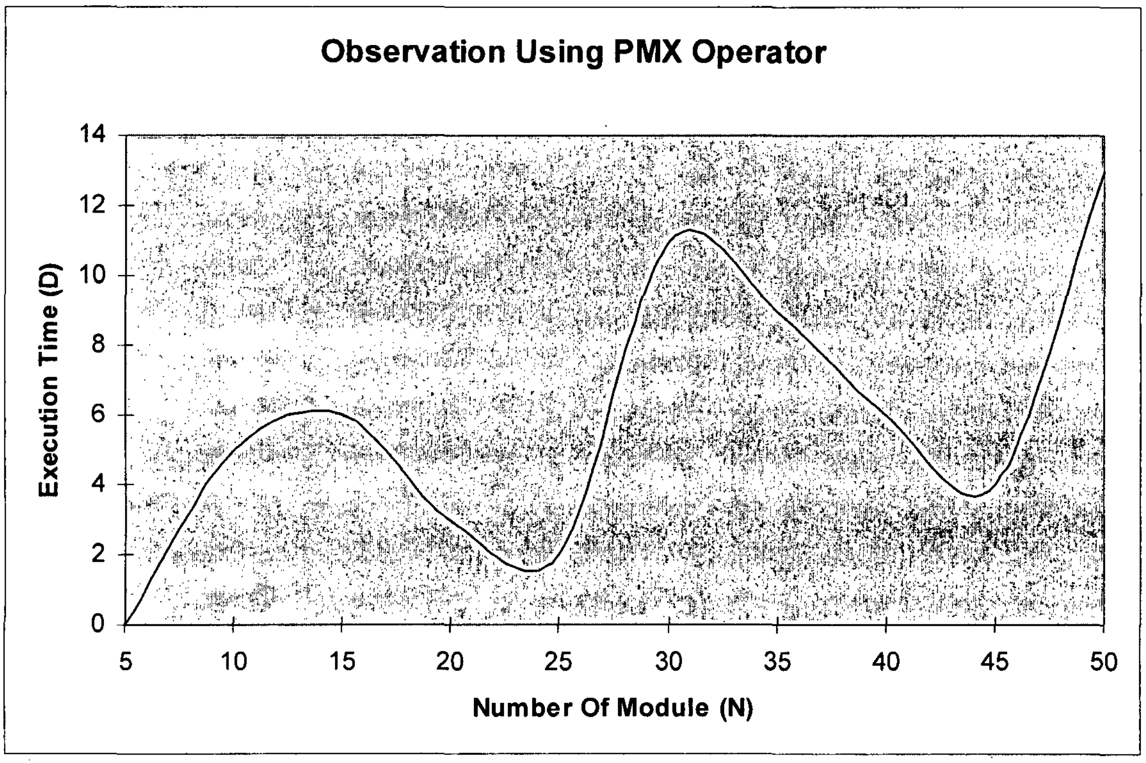

Fig. 4.1 Observation Using PMX Operator

In the graph of Fig. 4.1 graph X-axis represent number of module as input and Y-axis represent difference of execution time.

4.2 Observation Using OX Operator

We have given the same input as in 4.1 applying the OX operator. The observation of OX operator is given in Table 4.2. The first column shows how many numbers of modules are taken as the input and the second column shows the group formation of the modules and the difference in the execution times of the groups as the output.

INPUT	OUTPUT	
Number of Modules	Groups of Modules	Difference of Execution Time(D) (among groups) (in msec)
5	2/4 5/1 3	1
10	2 8 10 3/ 7 5 9/ 4 6 1	3
15	7 15 14 4 10 5 /11 6 13 1 2 /9 8 12 3	0
20	2 3 20 11 5 13 18 4 /1 19 8 10 12 7 14/ 16 6 17 9 15	3
25	7 24 5 22 25 2 23 4 3 14 17 / 15 20 11 9 19 8 13 21 / 6 16 18 12 10 1	4
30	3 14 23 13 6 4 24 16 7 25 9 5 20 11 /27 22 2 15 12 19 8 17 29 / 1 30 18 26 28 10 21	1
35	2 19 29 6 11 26 28 9 12 1 10 5 13 18 22 16 27 /32 3 14 31 21 30 24 20 34 25 4 15/ 33 8 15 7 23 35	6
40	1 21 33 36 5 9 27 38 13 39 7 34 26 4 25 2 35 19 20 /40 15 3 37 10 16 8 17 24 22 30 32 23 28/ 29 14 6 11 31 18 12	2
45	10 9 23 45 33 17 1 35 40 21 12 27 43 3 31 4 39 16 22 36 15 29 /14 28 37 5 38 18 11 41 30 24 42 2 34 44 8 / 26 25 7 32 6 19 20 13	2

50	37 14 28 46 4 23 36 3 6 48 29 32 9 40 25 11 2 10 15 22 7 5 16 31/ 34 33 50 17 45 12 13 24 27 18 49 43 21 42 19 35 20/ 1 38 39 47 8 26 30 44 41	5

Table 4.2 Observation Using OX Operator

In our proposed model we find the D difference between sums of execution time of groups of string, in a string number of modules make permutation. When input number of module is 5 then output D is 1, 1 is the difference of execution times among groups. When input value N increasing then the output value D may increases and decreases also, at input 15 it goes to 0. When input value number of modules increases the difference between execution times value increases and it also decreases to increasing number of modules.

In the graph of Fig. 4.2 graph X-axis represent number of module as input and Y-axis represent difference of execution time according to our model.

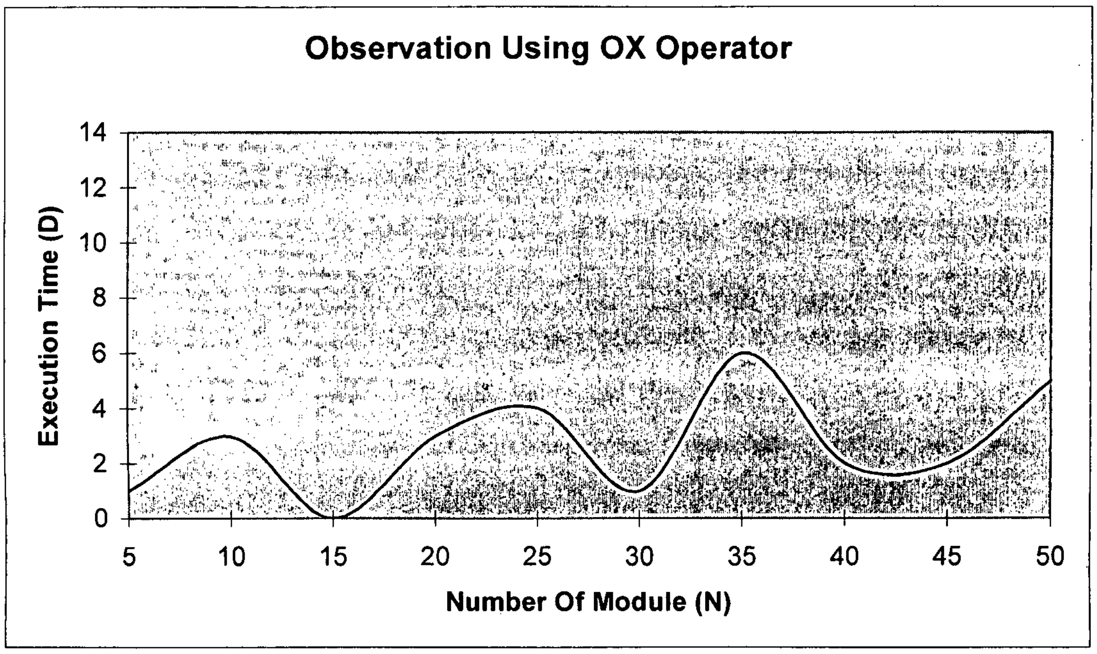

Fig. 4.2 Observation Using OX Operator

4.3 Comparison between Results of PMX and OX Operator

In the table 4.3 the differences of execution time of groups according to PMX and OX operator are given. Both Operators provide different set of values of execution time. OX operator gives less value than the PMX operator. Most of the value of execution time in OX operator is nearly to zero but in PMX operator not like OX operator.

Number of Modules	Difference of Execution Time(D) (among groups) (in msec) Using PMX	Difference of Execution Time(D) (among groups) (in msec) Using OX
5	0	1
10	5	3
15	6	0
20	3	3
25	2	4
30	11	1
35	9	6
40	6	2
45	4	2
50	13	5

Table 4.3 Comparison

The graph of Fig. 4.3 shows OX Operator provides better solution than the PMX operator. In graph of Fig. 4.3 dotted line represent OX operàtor and solid line represent PMX operator value. In the graph of Fig 4.3 initially OX operator takes greater value than the PMX but after that its goes bellow to PMX line. At input range 20-25 OX line again goes above to PMX line. OX line close to axis than the PMX line, in our proposed model we require the difference of execution time of groups goes towards zero gives good solution. According to proposed model OX operator gives better solution than the PMX operator.

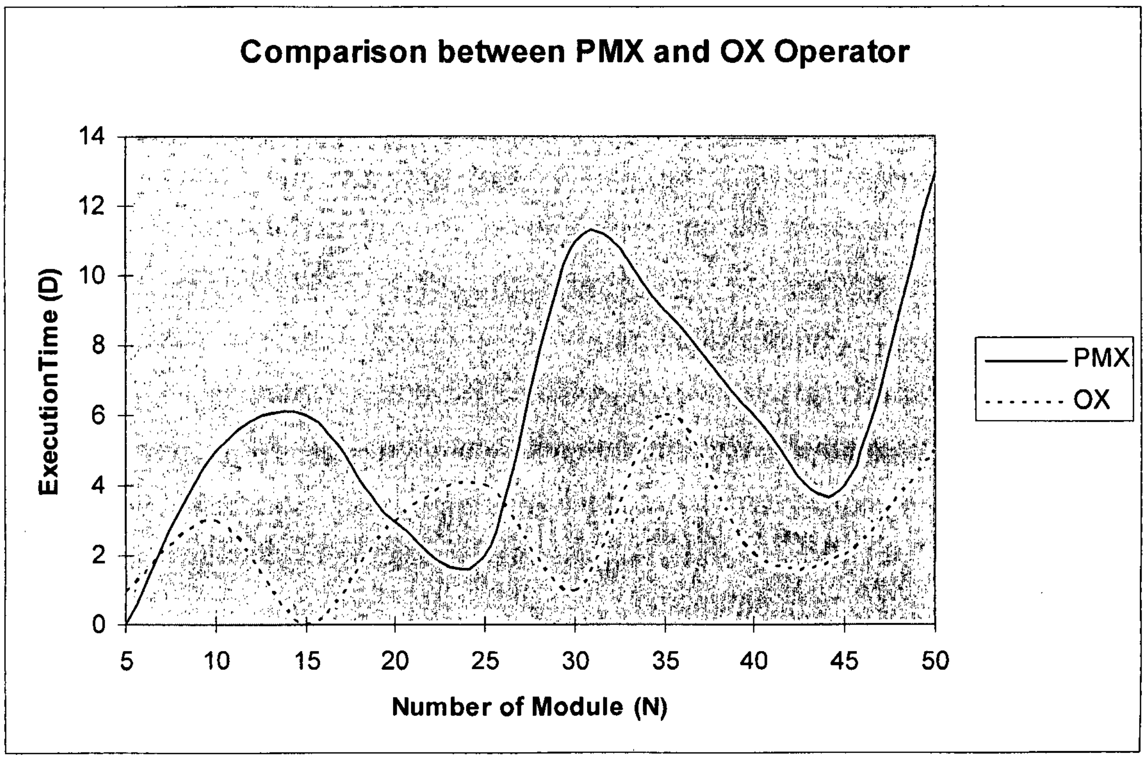

Fig. 4.3 Comparison between PMX and OX operator

Chapter 5

Conclusion and Future Work

5.1 Conclusion

In our dissertation task partitioning and grouping of the modules are supposed to provide better solutions if number of module are very large. PMX operator and OX operator both provides a better task sequence for execution in multiprocessor and multi-computer system. Our result show OX operator provides solution better than the PMX operator. We do not use CX (circular crossover) operator but it also provides a better solution for task partitioning. Observation of execution of task modules in parallel and distributed system provides help to develop a better operating system for parallel and distributed computing environment.

Graph of Fig. 4.1 represents Observation using PMX Operator. X-axis represent number of module as input and Y-axis represent difference of execution time among modules. Initially 5 numbers of modules taken as input and it gives D 0, at 10 numbers of modules the graphs takes value D 4 and for 15 D is 6 after that graph nature is decreasing up to 25 numbers of modules, and 30 its again goes to D value 11 and so on.

Graph of Fig. 4.2 represents Observation using OX Operator. X-axis represent number of module as input and Y-axis represent difference of execution time among modules. Initially 5 numbers of modules taken as input and it gives D 1, at 10 numbers of modules the graphs takes value D 3 and for 15 D is 0 after that graph nature is increasing up to 25 numbers of modules, and 30 its decreases. At the input 35 number of modules graph of Fig 4.2 takes D value 6 which is maximum, after that it decreases up to 2 for 40 and 45 and again increases for 50.

The comparison between PMX and OX operator observations are given in graph of Fig. 4.3. At initial value PMX operator produce better result means less value of D, after increasing the value of N OX operator gives less value of D than the PMX operator. At 25 value of N PMX operator again gives less value of D than OX operator, but again PMX operator gain value of D and shows higher value of D than the OX operator. In our observation OX operator gives better result then the PMX operator.

5.2 Future Work

In our dissertation, we observe task partitioning and there execution in parallel/ distributed system. The proposed work is based on the decomposition and the execution of task in parallel/distributed system. Our future work will be to generate modules by graph- theoretic approach for the solution so that we obtain consequent regrouping of the structure.

References

1) "A compile-Time Scheduling Heuristic for Interconnection-Constrained Heterogeneous Processor Architectures,", G.C. Sih and E.A. Lee, IEEE Trans. Parallel and Distributed Systems, vol. 4, no. 2, pp. 175-186, Feb. 1993.

2) " A comparison of heuristics for scheduling DAGS on multiprocessor" C. McCreary, A Khan, J. Thompson, and M. McArdle, 8th International Parallel Processing Symposium, 446-451, 1994

3) "A Genetic Algorithm for Multiprocessor Scheduling", Edwin S. H. Hou, Member, IEEE, Ninvan Ansari, Member, IEEE, and Hong Ren, IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS. VOL. 5, NO. 2, FEBRUARY 1994.

4) "A Genetic Algorithm Tutorial" by Darrell Whitley-Computer Science Department Colorado State University; Fort Collins, CO 80523

5) "Adaptive load sharing m homogeneous distributed systems"- EAGER, D. L., LAZOWSKA, E. D., AND ZAHORIAN, J. IEEE Trans. Software Engineering. 12 (1986), 662-675.

6) "Advance Computer Architecture" Kai Hwang, Tata McGraw Hill Ed, 2001.

7) An architecture for distributed agent-based data preprocessing Petteri Nurmi, Michael Przybilski, Greger Linden and Patrik Floreen Helsinki Institute for Information Technology HIIT, Basic Research Unit Department of Computer Science, P.O. Box 68.

8) "An introduction to genetic-based scheduling in parallel processor systems."- A. Y. Zomaya, F. Ercal, and S. Olariu, editors,Solutions to Parallel and

Distributed Computing Problems, chapter 5, pages 111–133. John Wiley and Sons, 2001.

9) "An Overview of Genetic Algorithms: Part 2, Research Topics" by David Beasley- Department of Computing Mathematics, University of Wales College of Cardiff, Cardiff, CF2 4YN, UK; David R. Bully-Department of Electrical and Electronic Engineering, University of Bristol, Bristol, BS8 1TR, UK; Ralph R. Martinz-Department of Computing Mathematics, University of Wales College of Cardiff, Cardiff, CF2 4YN, UK; University Computing, 1993, 15(4) 170-181.

10) "Communication Contention in Task Scheduling"- Oliver Sinnen and Leonel A. Sousa, Senior Member, IEEE, IEEE Transaction on Parallel and Distributed System Vol. 16, No-6, June 2005.

11) Distributed system Concept and Design – George Coulouris, Jean Dollimore and Tim Kindberg, Third Edition, Sixth Indian Reprint 2004, Publisher-Pearson Education (Singapore) Pte Ltd.

12) "Distributed System - Principles and Paradigms" By Andrew S. Tenenbaum, Maarten Van Steen, Pearson Education, Revised Edition 2006.

13) "Dynamic Scheduling of Computer Tasks Using Genetic Algorithms"- C.A Gonzalez Pico and R.L Wainwright; Proc. First IEEE Conf. Evolutionary Computation, IEEE World Congress Computational Intelligence, vol. 2,1994, pp. 829-833.

14) "EFFICIENT MULTIPROCESSOR SCHEDULING BASED ON GENETIC ALGORITHMS" E. S. H. Hou, R. Hong, and N. Ansari Department of

Electrical and Computer Engineering New Jersey Institute of Technology Newark, NJ 07102,087942-6004/90/1100-1239, 1990 IEEE.

15) "Genetic Algorithm - in search, optimization & machine learning", By David E. Goldberg, Pearson Education, 2006.

16) "Genetic Alorithms- simulating nature's methods of evolving the best design solution", Cezary Z. Janikow and Daniel St. Clair; Feb-March 1995 0278-6648/95- 1995 IEEE.

17) "Genetic Algorithms: Theory and Application", Lecture Notes, Third Edition –Winter 2003/2004 by Ulrich Bodenhofer.

18) "Genetic Scheduling for Parallel Processor Systems: Comparative Studies and Performance Issues", Albert Y. Zomaya, Senior Member, IEEE, Chris Ward, and Ben Macey; IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, VOL. 10, NO. 8, AUGUST 1999.

19) "Observation on Using Genetic Algorithms For Dynamic Load-Balancing", Albert Y. Zomaya, And Yee-Hwei, IEEE Transactions On Parallel And Distributed Systems, Vol. 12, No. 9, September 2001.

20) "Optimal scheduling strategies in a multiprocessor system" C. V. Ramamoorthy et al IEEE Trans. Comput., vol. C-21, Feb.1972, pp. 137-146.

21) "Parallel Computing" by Moreshwer R. Bhujade, New Age International Publishers, 2003.

22) "Parallelization of Fine-grained Irregular DAGs" by Frederic T. Chong, Shamik D. Sharmay, Eric A. Brewer and Joel Saltzx -Massachusetts Institute

of Technology ,University of Maryland, College Park, University of California, Berkeley.

23) “Practical Multiprocessor scheduling algorithms for efficient processing”- H Kasahara and S. Narita; IEEE Trans, Comput. Vol. C-33 no. 11, pp. 1023-1029, Nov 1984.

24) “Preemptive Scheduling for Distributed Systems” By - Donald McLaughlin-Department of Computer Science and Engineering. Arizona State University, and Shantanu Sardesai and Partha Dasgupta- Tandem Computers Inc. 19333 Vallco Parkway, Cupertino, CA.

25) “Scheduling a Job Mix in a Partitionable Parallel System” By - Helen D. Karatza, Department of Informatics, Aristotle University of Thessaloniki, 54006 Thessaloniki, Greece and Ralph C. Hilzer, Computer Science Department,California State University, Chico, Chico, California 95929-0410 USA; Proceedings of the 35thAnnual Simulation Symposium (SS.02)- 2002 IEEE.

26) “Task Assignment in a Distributed System: Improving Performance by Unbalancing Load”, Mark E. Crovella Department of Computer Science Boston University Boston, , Mor Harchol-Balter Laboratory for Computer Science MIT, NE43-340 Cambridge, MA 02139, Cristina D. Murtaz Department of Computer Science Boston University Boston, MA 02215; October 31, 1997 BUCS-TR-1997-018.

27) “Task partitioning using Genetic Algorithm” By Anil Kumar Tripathi, Deo Prakash Vidyarthi and A. N. Mantri, Proceeding of International Conference on Cognitive System, New Delhi, December 1997, pp. 248-254.