International Journal of Soft Computing 6 (5-6): 175-182, 2011
ISSN: 1816-9503

A Genetic Algorithm Based Scheduling of an Input Queued Switch

[1]D. Raghupathikumar and [2]K. Bommanna Raja
[1]K.L.N. College of Engineering, Sivagangai Dt, Tamil Nadu, India
[2]Excel College of Engineering, Salem Dt, Tamil Nadu, India

Abstract: This study is concerned with a genetic algorithm based scheduling an input-queued switch. This study uses binary encoding of real parameter vectors as chromosomes as genes and as alleles. In this study we investigate that a scheduling of IQ with GA researchers obtain feasible solutions. Initial population is generated by permutation method and fitness was calculated by weighted sum of the packet arrivals. The tournament method of parent's selection and single point crossover point is adopted. It is tested with 4×4 input queued switch. The result shows that the GA has been successfully applied to scheduling of IQ switches. Experimental results are also shown that GA based scheduling of an IQ switches have better performance in throughput and low latency.

Key words: Genetic algorithm, input-queued switch, tournament, crossbar-fabric, scheduling, India

INTRODUCTION

The popularity of the internet can be very much contributed to packet switching as it enables any device to communicate to any device with virtually at any time. The packet switching concept was first invented by Paul Baran in the early 1960's and later on some researchers build the ARPANET, the world's first packet switching network. Over the past few years, several technology trends have converged to internet and now it provides platform for high performance distributing computing which operates the link rate above gigabits per second. Because of increasing speed, the internet requires a switching/routing to handle an aggregate traffic. The switches and routers basically store, route and forward these packets before they reach the destination. One of the core functionality of these switches is to transfer the packets arriving at the input ports to the output ports. Although, it is challenging problem to solve this switching at faster line rates.

There are different switch architectures and algorithms have evolved. These switches performance was determined by the scheduling algorithms and queuing schemes. Since, time is slotted at most one packet can be transferred from input port to output port. The scheduler resolves the contention if more than one packet is contended for the same output. Not all packets can be switched and transmitted to unique output port as soon as they arrive; it is inevitable to buffer some packets. Depending on the position of the buffers, there are different kinds of switch architectures namely Input Queued (IQ), Output Queued (OQ) and Shared Queued (SQ), etc. Each architecture has its own pros and cons that will be discussed here.

LITERATURE SURVEY

Output queuing: When a packet arrives at an input port, without any delay, cross the switching fabric and immediately placed into the output buffer residing at the output port. At each time-slot at most one packet leaves each output port, conflicting packets are queued in a buffer at the output-port. Output Queuing (OQ) offers high performance for individual data flows or group of flows (Demers *et al*., 1989). OQ provides lowest average cell-delay, since cells are queued only when the output port is transmitting a cell. Therefore, an OQ switch architecture having queues of infinite size can always achieve better throughput for all kinds of traffic. The scheduling algorithms guarantee QoS requirements adopt OQ switches (Demers *et al*., 1989; Parkeh and Gallager, 1993; Shreedhar and Varghese, 1995). However, OQ suffers with the internal speedup problem of the switch. Because packets destined for the same output port may arrive simultaneously from many input ports, the output buffers needs to enqueue the packets at a much higher rate. In other words, the switching fabric and buffer needs to operate at N times faster than link rate. Unfortunately, this OQ is not suitable for architecture which needs scalable in increasing link rates and impractical for high speed switches with large number of ports. Some of the examples of OQ switch architecture of buffered crossbar switch, knock-out output queued switch, etc.

Shared queuing: Shared memory switches are another variant of high speed switches where memory/buffer shared by all input and output lines. Although, shared memory implementation reduces the aggregate bandwidth compared with OQ. This type of switches is more flexible

Corresponding Author: D. Raghupathikumar, K.L.N. College of Engineering, Sivagangai Dt, Tamil Nadu, India

than dedicated memory and smaller buffer size than OQ. At each time slot, the packets arriving at the input ports are written to the shared memory which in due time are read out and sent to the output lines. The shared memory should operate in the aggregate rate of both input ports and the output ports. Because, packets destined for the same output port arrive simultaneously from many input ports, the output port needs to read packets at much higher rate than a single input port may write it which places string end limits on the switch size. Also, if line rates increase, then memory bandwidth of the switch should also increase linearly. Another draw back is memory has to be accessed twice (for both read and write) in every time slot which automatically increases access time and memory size increases linearly with line rate. Hence, like OQ switches, shared memory architecture are also not practical for high speed large size switches. Examples for shared memory switches include Hitachi switch, etc.

Input queuing: This makes Input Queuing (IQ) becomes very appealing for switches with fast line rates or with a large number of ports. The Input Queued (IQ) switch provides a low-cost architecture for cross bar-based switches designing and it is attractive for very high bandwidth. This is due to that both the memories and the switch fabric need only to operate at the same speed as the line rate which is independent of N. In IQ, packet switch architectures, packets arriving at input lines are queued at input ports maintained by line card. Then, packets are dequeued from the input port according to the crossbar constrained imposed by the scheduler i.e., it resolves contention by solving the packets contending for the same output port. During a switching cycle only one packet is dequeued from the input port to the output port. Additionally, the selection policy must be designed such that fairness among inputs and outputs is guaranteed. IQ does not have the scaling limitations of OQ or shared queuing. The input-queued switches are impractical because of poor performance if FIFO queues are used to queue cells at each input only the first cell in each queue is eligible to be forwarded. As a result, FIFO input queues suffer from Head of Line (HoL) blocking. When the packet at the head of the FIFO queue is blocked, all the packets behind it are prevented from being transmitted even if the output port they are destined to it is idle. HOL blocking limits the throughput of each input port to a max of 58.6% under uniform traffic and is much lower than that of burst traffic (Chuang *et al*., 1999) .To overcome this problem, each input queue maintains FIFO queue for each output hence, a total of N x N = N^2 queues are present. This separation of queues eliminates performance degradation due to HOL blocking and the

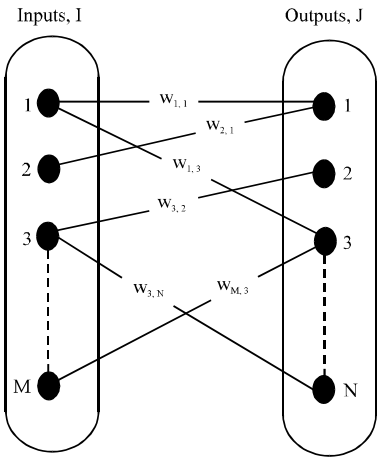

Fig. 1: Request graph G

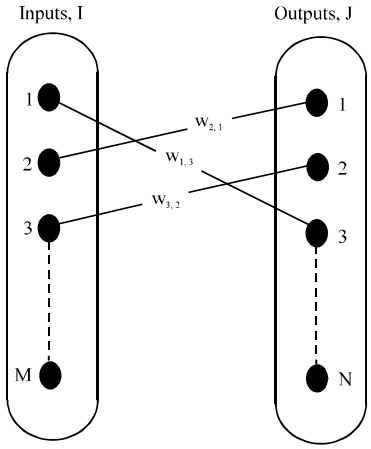

Fig. 2: Matching M

queue is said to be Virtual Output Queue (VOQ) or Destination Queue (DQ). Throughput and delay are the two main quantities with regards to the performance of a switch scheduling algorithm. A scheduling algorithm is said to be stable algorithm if it achieves 100% throughput. A scheduling algorithm selects a match (or) matching M. In the technical literature scheduling algorithm faced by cross-based IQ cell switch architectures are described as solutions of the matching problem in bipartite graphs. A Bipartite graph G = (V, E) consists of 2N vertices, partitioned into two sets namely N on the left, N on the right and set of V edges connecting vertices on both sides as shown in Fig. 1. A matching problem of an IQ switch can be matched to a bipartite graph G. IQ switch with input ports i corresponds to the left side vertices and output ports j corresponds to the right side vertices of a bipartite graph. These edges are connected by edges E. A weight metric is associated with an all the edges E of the graph G. A match M as shown in Fig. 2 on this graph is a selection of any admissible subset of edges E. A subset

of admissible edges such that no two edges in M have a common vertex i.e., it never happens that two cells are transferred from input port i to output j thus satisfying the bandwidth restrictions imposed by the crossbar. A match can be maximum matching and maximal matching. A maximum matching is the largest size matching made on a given graph. A maximal matching is one no further edge can be added without modifying an already matched edge. Therefore, a maximum matching is maximal matching but reverse is not true. The scheduling algorithm of an input queued switch can be classified as:

- Maximum Size Matching (MSM) algorithm
- Maximum Weight Matching (MWM) algorithm
- Randomization matching algorithm

Maximum size matching: A MSM is one that finds the maximum numbers of matches between inputs and outputs. In other words, a MSM on the graph G is one that maximizes the number of edges in M. This would provide the highest possible instantaneous throughput in a given time slot. The complexity of solving MSM is $O(N^{5/2})$. A maximal matching can be achieved by adding edges incrementally without modifying/removing the existing edges made earlier. Maximal matching converges for multiple iterations. It provides fairness, QoS support and good throughput under uniform and identically distributed (i.i.d) Bernoulli traffic's. It can lead to instability under inadmissible traffic and they can even lead to starvation. There are several algorithms like PIM (Nong *et al*., 1999), iSLIP (Anderson *et al*., 1993), iFAIR (McKeown, 1999), FIRM (Kumar *et al*., 2004) were proposed in the literature. PIM is 3 phase iterative algorithm which consists of request, grant and accept phase. During grant and accept if any unmatched output/input receives grant/accept then they will be chosen by random selection. The PIM algorithm is difficult to implement at very high speed due to its randomness. iSLIP was developed by McKeown and is analyzed extensively by Chuang *et al*. (1999). Each input port and output port has a distinct priority wheel. Input ports accept/grant based on priority wheels and advance their wheels. However, under non uniform traffic, the pointers are not necessarily desynchronized and the performance potentially degrades. FIRM (Duan *et al*., 1998) is fairer than iSLIP in terms of FCFS service, i.e., it approximates FCFS closer than iSLIP with the use of the round-robin pointers. FIRM provides a basis for a class of distributed scheduling algorithms.In addition to it, there are also scheduling algorithms like MUCS (Shah *et al*., 2002; Marsan *et al*., 1999) uses matrix method for scheduling, RPA (Duan *et al*., 1998) gives priority to urgent cells were proposed in the literature.

Maximum weight matching: A MWM is one that finds a matching M which maximizes the total weight $W(t) = \sum W_{ij}(t)$ at time slot t provided a weights $W_{ij}(t)$ is attached to the edges of graph G. A switch is stable (for $N \geq 2$) if the MWM algorithm is used (Demers *et al*., 1989; Shreedhar and Varghese, 1995). On the other hand, it is known that with the MSM algorithm, a switch can be unstable for $N \geq 3$ (Demers *et al*., 1989) (if ties are broken randomly). A MSM is a special case of MWM with all weights $W_{ij}(t) = 1$ at time slot t. The complexity of solving MWM is $O(N^3)$ which infeasible to implement at high speed ports. Some examples are LQF (Serpanos and Antoniadis, 2000), OCF (Mekkittikul and McKeown, 1996), LPF (McKeown, 1995). These algorithms adopt queue occupancies as weights, i.e., $W_{ij}(t) = Q_{ij}(t)$ uses the age of the HoL packets as weights to find the maximum weights. MWM have been used to show that IQ switches with 100% throughput under admissible traffic (Demers *et al*., 1989). However, MWM have intrinsically high computation complexity that is translated into long resolution time and high hardware complexity. The complexity makes these schemes prohibitively expensive for a practical implementation with currently available technologies. However, the hardware and time complexity of these schemes can be considered still high for the ever increasing data rates. In addition, a large number of iterations may be needed to achieve satisfactory matching results, driving the complexity even higher. LQF gives preferences to VOQ with higher occupancy and considers this as weight. It does not consider the waiting time of the cells and leads to starvation. LQF is very difficult to implement in hardware at high speed. OCF gives preference to cells that have been waiting for the longest time. The weight of the LPF algorithm (Tassisulas, 1998) is port occupancy which is the total number of cells queued at the input and output ports.

Random matching algorithm: The basic idea of randomized scheduling is to select the best matching from a set of random matches. Randomized scheduling algorithms have been proposed for input queued switches in an attempt to simplify the scheduling problem and provide fairness. TASS (Mekkittikul and McKeown, 1998) is the basic randomized algorithm proposed by Tassiulus. TASS achieves stability under any admissible traffic. It chooses a match uniformly at random from the set of all N! Possible matches. Under non-uniform traffic it does not have good delay performance. A group of randomized algorithms including APSARA, LAURA and SERENA were proposed in the literature (Tassisulas, 1998). APSARA performs matching by using neighbors match,

previous time slot match and randomly chosen match. LAURA (Tassisulas, 1998) uses a non-uniform random sampling.

In LAURA a match with increased weight is obtained by merging randomly chosen with the previous slot. Genetic algorithm have been used for solving job shop scheduling (Gomes *et al*., 2005; Murovec and Suhel, 2004) and broadcast scheduling in packet radio networks (Chakraborty and Hirano, 1998). In addition, GA was implement to schedule the input queued switch based on the matrix decomposition method (Jin *et al*., 2005).

IQ SWITCH-PROPOSED WORK

A large class of ATM switches is represented by architectures using a non-blocking interconnection network. In principle, a non-blocking interconnection is a crossbar structure that guarantees absence of switching conflicts (internal conflicts) between packets addressing different switch output ports.

The input queued packet switch consists of M inputs to N outputs, crossbar switch fabric and scheduler as shown in Fig. 3. The fixed sized packets or cells that arrive at input i are represented by $A_{(i,\ j)}$ (t), $1 \le i \le N$. At the beginning of each time slot, either zero or one cell arrive at each input i and is destined to output j, $1 \le j \le N$. The cell is placed in the FIFO queue $VOQ_{(i,\ j)}$ which has queue occupancy L_{ij} (n). The arrival process A_{ij} (n) is at a rate λ_{ij} and the set of all arrival processes as:

$$A(n) = \{A_i(n); 1 \le i \le N\}$$

A (n) is considered admissible if $\sum \lambda_{ij} < 1$, otherwise it is inadmissible. A scheduler selects a match (or)

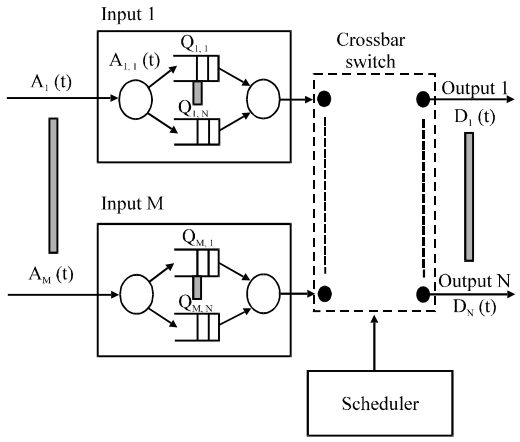

Fig. 3: Schematic diagram of an input-queued switch

smatching; M between the inputs and outputs defined as a collection of edges from the set of nonempty input queues to set of outputs such the each nonempty input is connected to atmost one output and each nonempty output is connected to atmost one input. At the end of the time slot if input i is connected to output j, one cell is removed from $VOQ_{(i,\ j)}$ and sent via crossbar fabric to output j.

Definitions:

- Speed up: operating the switch fabric at S times faster than the input lines
- Matching: matching is a one to one active input and output mapping
- Switching cycle: time required to switch packets from input port to output port
- Internal blocking: two packets contend for same link at same time inside the switch fabric
- Throughput: probability that a packet received on an input port is successfully switched and transmitted by the addressed switch output port

Taxonomy:

- M x N: Number of input and output ports
- VOQ_{ij}: VOQ hold packet arriving to input i and Destined for output j
- λ_{ij}: The average packet arrival rate at input port i for output j
- W_{ij}: Weight assigned from input i to output j
- Q_{ij}: The packet at the head of the FIFO queue
- L_{ij} (t): The queue occupancy or length of the queue at time t
- C_q: Current capacity of the VOQ
- Q_{max}: Maximum capacity of VOQ eg 10000
- A_{ij} (t): Packets arrive at VOQ_{ij} at the beginning of time slot t

Assumption:

- Size of the switch M equals N
- Buffers are not shared
- Excess packets are dropped
- The internal speed of an IQ switch need not be larger than the highest line rate
- The switch operation is synchronous on a cell basis, i.e., time is slotted and at most one cell can be transferred from inputs to outputs in one time slot
- Switches that internally operate on fixed-size data units called cells from the ATM jargon
- Arriving traffic follows a Bernoulli i.i.d distribution with mean rate λ_{ij} arriving to VOQ_{ij}

GENETIC ALGORITHM

Genetic Algorithms (GA) are inspired by Darwin theory of the survival of the fittest. The GA was first

introduced by Holland in 1975. It is a stochastic heuristics which encompass semi-random search method whose mechanism is based on the simplifications of evolutionary process observed in nature. As opposed to many other optimization methods, GA works with a population of solutions instead of just a single solution. GA assigns a value to each individual in the population according to a problem-specific fitness function. A survival-of-the-fittest step selects individuals from the old population. A reproduction step applies operators such as crossover or mutation to those individuals to produce a new population that is fitter than the previous one. In applying GA, researchers have to analyze specific properties of problems and decide on a proper representation, an fitness function and a construction method of initial population, a genetic operator and a genetic parameter. The study describes in detail how the GA is developed to solve the Scheduling problem of an IQ switch.

Definitions:

- Chromosome: chromosome contains the solution in form of genes
- Gene: a part of chromosome. A gene contains a part of solution
- Individual: same as chromosome
  Population: no of individuals present with same length of chromosome
- Fitness: fitness is the value assigned to an individual. It is based on how far or close a individual is from the solution. Greater the fitness value better the solution it contains
- Crossover: taking two fit individuals and intermingling there chromosome to create new two individuals
- Mutation: changing a random gene in an individual
- Selection: selecting individuals for creating the next generation
- Search space: the space for all possible feasible solutions is called search space

Taxonomy:

- P: Population in the Search space
- Ω_i: Parent in the population P
- F_i: Fitness function for the parent Ω_i
- N_u: Frequency of selection
- p_c: Crossover probability
- p_m: Mutation probability
- p_e: Elitism rate

Outline of the basic genetic algorithm:

- Start: generate population of n chromosomes
- Fitness: evaluate the fitness F_i of each chromosome x in the population
- New population: create a new population by repeating following steps until the new population is complete
- Selection: select two parent chromosomes from a mating pool according to their fitness (the better fitness, the bigger chance to be selected)
- Crossover: with a crossover probability cross over the parents to form a new offspring
- Mutation: with a mutation probability mutate new offspring at each locus (position in chromosome)
- Accepting: place new offspring in a new population
- Replace: use new generated population for a further run of algorithm
- Test: if the end condition is satisfied, stop and return the best solution in current population
- Loop: go to step 2

SCHEDULING IQ WITH GA

In this study, researchers propose a genetic algorithm to schedule the packets arrived at HoL of the VOQ_{ij}. Let λ_{ij} be the arrival rate of the packet from input port i to the output j for M x N input-queued crossbar switch and Q_{ij} be the current packet. Let for an example we consider the following arrival rate at VOQ for an IQ (Table 1):

$$\sum_{i=1}^{n} W_{ij}\lambda_{ij}, \ j = 1...N \tag{1}$$

$$\sum_{j=1}^{n} W_{ij}\lambda_{ij}, \ i = 1...N \tag{2}$$

The two conditions in Eq. 1 and 2 called no overbooking or over subscribed conditions in (Chuang *et al*., 1999) as the simply state that neither the total rate to an input port can be >1. Otherwise it is referred to admissible.

Chromosome representation and encoding: The first step in constructing the GA is to define a genetic representation (coding). A good representation is crucial because it significantly affects all the subsequent steps of the GA. In this study, researchers consider 4×4 with the size of the chromosome being 16 (Table 2). In this study researchers proposed to encode the arrival rate in binary representation (Table 3).

Table 1: Packets arrival rate in IQ

i\j	1	2	3	4
VOQ_{1j}	0.24	0.36	0.12	0.28
VOQ_{2j}	0.51	0.07	0.34	0.08
VOQ_{3j}	0.04	0.42	0.09	0.45
VOQ_{4j}	0.21	0.15	0.45	0.19

Table 2: Chromoseome representation (size = 16)

Arrival rate	λ_{11}	λ_{12}	λ_{13}	λ_{14}	.	λ_{42}	λ_{43}	λ_{44}
Input port	3	1	4	2	.	3	1	2
Output port	2	1	2	4	.	1	4	2

Table 3: Binary encoding of the chromosome

VOQ_{ij}

Input i	Output j	Arrival rate λ_{ij}	Binary encoding
1	3	0.24	0010 0100
2	4	0.36	0011 0110
3	1	0.12	0001 0010
4	2	0.28	0010 1000
:	:	:	:

Table 4: Fitness calculation for chromosome

Population size Ω_i	----------Arrival rate λ_{ij}----------				Chromosome	Fitness value F_i
1	0.24	0.36	0.12	0.28	0010 0100..1000	0.64
2	0.36	0.28	0.24	0.12	0011 0110..0010	0.72
3	0.28	0.24	0.12	0.36	0010 1000..0110	0.61
4	0.24	0.28	0.36	0.12	0010 0100..0010	0.66
5	0.36	0.12	0.24	0.28	0011 0110..1000	0.64
6	0.28	0.12	0.36	0.24	0010 1000..0100	0.61
7	0.12	0.24	0.28	0.36	0001 0010..0100	0.53
8	0.28	0.36	0.24	0.12	0010 1000..0010	0.70
:	:	:	:	:	:	:

Fitness functions: Fitness function is the measure of the quality of the chromosome and its survival in the next generation. The greater the fitness of a chromosome is the greater the probability to survive. In this study, the fitness function is calculated as (Table 4):

$$F = \sum_{i=1}^{n} W_{ij}\lambda_{ij}, \quad j = 1...N \tag{3}$$

A weightage W_{ij} is assigned to arrival rate λ_{ij} of the VOQ_{ij} is HoL packets in decreasing order. The fitness function is based on their weightage.

Selection: GA uses a strategy to select the chromosomes from population and insert them into a mating pool and chromosomes from the mating pool are used to generate new offspring which are the basis for the next generation. In this study, researchers use tournament selection strategy for selecting parents from mating pool. The tournament strategy selects competitive parents among N_u chromosomes. We considered N_u as 2. The best chromosome (the winner) from the tournament is the one with highest fitness F_i which is the winner of N_u is inserted into mating pool (Table 5).

Crossover: Crossover is more rapidly exploring a search space. Crossover selects genes from parent chromosomes

Table 5: Tournament method of chromosomes selection

Population size Ω_i	Chromosomes randomly selected	Winner (offspring)	Fitness value F_i
1	Ω_1 and Ω_4	Ω_4	0.66
2	Ω_2 and Ω_3	Ω_2	0.72
3	Ω_5 and Ω_7	Ω_5	0.64
4	Ω_6 and Ω_2	Ω_2	0.72
5	Ω_7 and Ω_4	Ω_4	0.66
:	:	:	:

Table 6: Single point crossover

Single point cross over	Parent 1	Parent 2
Before cross over (old)	0010 0100	0011 0110
After cross over	0010 0110	0011 0100

and creates a new offspring. The simplest way how to do this is to choose randomly some crossover point and everything before this point copy from a first parent and then everything after a crossover point copy from the second parent. The two offspring each inherit some genes from each parent. This is known as single point crossover. Crossover is not usually applied to all pairs of individuals selected for mating. A random choice is made and in this paper single-point crossover is applied with probability P_c is 0.5 (Table 6).

Mutation: Mutation provides a small amount of random search and helps ensure that no point in the search space has a zero probability of being examined. It randomly alerts each gene with a small probability $P_m = 0.001$.

SIMULATION AND RESULT

In this study simulation is carried by using Network Simulator (NS2). NS2 is a discrete event simulator targeted at networking research and it provides substantial support for simulation of TCP, routing over wired and wireless network. The simulations contain four input ports and output ports. A novel genetic algorithm was proposed to schedule the packets arrived at the VOQ of the input queued switch. The size of VOQ is 10000 to hold the incoming packets, excess packets will be dropped. The initial population is generated with permutation and the parents for new generation are selected by using Tournament method with frequency of selection is set to 2 (i.e., $N_u = 2$). The parents from mated were cross over by Single point crossover with probability $P_c = 0.5$ and mutated with $P_m = 0.001$ (Table 7 and 8). The best selected packets with highest arrival rate in the input port are then mapped to output port. A random choice was made for contending output port. The packets are then dequeued from the VOQ to the output link via output port. It is infer that scheduling IQ with GA has better throughput and low latency which is shown in Fig. 4.

Table 7: Parameters for genetic algorithm

Parameters	Notation	Values
Chromosome size	CS	16.000
Population size	Ω	32,000.000
Crossover probability	p_c	0.500
Mutation rate	P_m	0.001
Elitism rate	P_e	0.100

Table 8: Parameters for input queued switch

Parameters	Notation	Values
Input ports	M	4
Output ports	N	4
Switch size	M x N	16
VOQ size	Q_{max}	10,000
Arrival rate	λ_{ij}	0-1
Weightage	W_{ij}	0.25-1

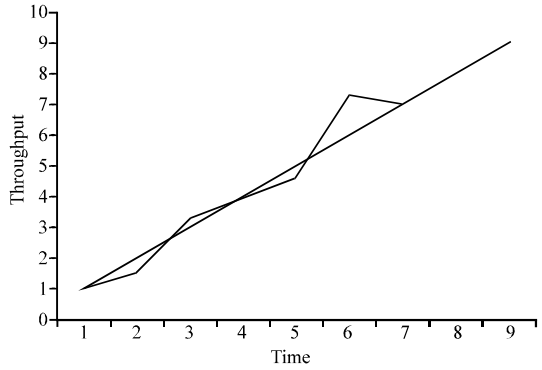

Fig. 4: Throughput in terms of time

CONCLUSION

This study proposed a GA for scheduling the 4×4 IQ switch to minimize the delay and to improve the throughput. Three-row chromosome structure is designed based on the scheduling problem. Parents were selected using Tournament strategy and crossed with single point crossover. Finally, a graph is obtained which shows that there is an improvement both in throughput and delay. The computational result shows that GA can obtain better solution for scheduling IQ switch.

REFERENCES

Anderson, T.E., S.S. Owicki, J.B. Saxe and C.P. Thacker, 1993. High speed switch scheduling for local area networks. ACM Trans. Comput. Syst., 11: 319-352.

Chakraborty, G. and Y. Hirano, 1998. Genetic algorithm for broadcast scheduling in packet radio networks. Proceedings of the 1998 IEEE International Conference on Evolutionary Computation IEEE World Congress on Computational Intelligence, May 4-9, 1998, Anchorage, AK., pp: 183-188.

Chuang, S.T., A. Goel, N. Mckeown and B. Prabhakar, 1999. Matchingoutput queueing with a comibed input/output-queued switch. IEEE J. Sel. Area Commun., 17: 1030-1039.

Demers, A., S. Keshan and S. Shenker, 1989. Analysis and simulation of a fair queueing algorithm. Proceedings of the SIGCOMM'89 Symposium on Communications Architecture and Protocols, September 1989, New York, USA., pp: 1-12.

Duan, H., J.W. Lockwood and S.M. Kang, 1998. Matrix unit cell scheduler mucs for input-buffered ATM switches. IEEE Commun. Lett., 2: 20-23.

Gomes, M.C., A.P. Barbosa-Povoa and A.Q. Novais, 2005. Optimal scheduling for flexible job shop operations. Int. J. Prod. Res., 43: 2323-2353.

Jin, Y., J. Zhang and W. Hu, 2005. A Genetic Algorithm of High-Throughput and Low-Jitter Scheduling for Input-Queued Switches. In: Advances in Natural Computation, Wang, L., K. Chen and Y.S. Ong (Eds.). LN CS 3612, Springer, Berlin, Heidelberg, ISBN-13: 978-3-540-28320-1, pp: 102-111.

Kumar, N., R. Pan and D. Shah, 2004. Fair scheduling in input-queued switches under inadmissible traffic. Proceedings of the IEEE Global Telecommunications Conference, November 29-December 3, 2004, IEEE Communications Society, USA., pp: 1713-1717.

Marsan, M., A. Bianco, E. Leonardi and L. Milia, 1999. RPA: A flexible scheduling algorithm for input buffered switches. J. Commun. IEEE Trans., 47: 1921-1933.

McKeown, N., 1995. Scheduling algorithms for input-queued cell switches. Ph.D. Thesis, UC Berkeley.

McKeown, N., 1999. The iSLIP scheduling algorithm for input-queued switches. Networking IEEE/ACM Trans., 7: 188-201.

Mekkittikul, A. and N. McKeown, 1996. A starvation free algorithm for achieving 100% throughput in an input queued switch. Proceedings of the 5th International Conference on Computer Communications and Networks, October 16-19, 1996, Rockville, MA., pp: 226-229.

Mekkittikul, A. and N. McKeown, 1998. A practical scheduling algorithm to achieve 100% throughput in input-queued switches. Proc. INFOCOM Annu. Joint Conf. IEEE Comput. Commun. Soc., 2: 792-799.

Murovec, B. and P. Suhel, 2004. A repairing technique for the local search of the job-shop problem. Eur. J. Operat. Res., 153: 220-238.

Nong, G., J.K. Muppala and M. Hamdi, 1999. Analysis of non blocking ATM switches with multiple input queues. Networking IEEE/ACM Trans., 7: 60-74.

Parkeh, A.K. and R.G. Gallager, 1993. A generalized processor sharing approach to flow control in integrated services networks: The single-node case. Networking IEEE/ACM Trans., 1: 344-357.

Serpanos, D.N. and P.I. Antoniadis, 2000. FIRM: A class of distributed scheduling algorithms for high-speed atm switches with multiple input queues. Proc. IEEE INFOCOM Annu. Joint Conf. IEEE Comput. Communi. Soc., 2: 548-555.

Shah, D., P. Giaccone and B. Prabhakar, 2002. An efficient randomized algorithms for input-queued switch scheduling. Micro IEEE, 22: 10-18.

Shreedhar, M. and G. Varghese, 1995. Efficient fair queueing using deficit round robin. Proceedings of the Conference on Appplications, Technologies, Architectures and Protocols for Computer Communication, October 1995, New York, USA., pp: 231-242.

Tassisulas, L., 1998. Linear complexity algorithms for maximum throughput in radio networks and input queued switches. INFOCOM 98. Seventeenth, 2: 533-539.