An Optimization Solution for Packet Scheduling: A Pipeline-Based Genetic Algorithm Accelerator

Shiann-Tsong Sheu, Yue-Ru Chuang, Yu-Hung Chen, and Eugene Lai

Department of Electrical Engineering, Tamkang University,
Tamsui, Taipei, Taiwan 25137, R.O.C.
stsheu@ee.tku.edu.tw, g8350021@tkgis.tku.edu.tw

Abstract. The dense wavelength division multiplexing (DWDM) technique has been developed to provide a tremendous number of wavelengths/channels in an optical fiber. In the multi-channel networks, it has been a challenge to effectively schedule a given number of wavelengths and variable-length packets into different wavelengths in order to achieve a maximal network throughput. This optimization process has been considered as difficult as the job scheduling in multiprocessor scenario, which is well known as a NP-hard problem. In current research, a heuristic method, genetic algorithms (GAs), is often employed to obtain the near-optimal solution because of its convergent property. Unfortunately, the convergent speed of conventional GAs cannot meet the speed requirement in high-speed networks. In this paper, we propose a novel hyper-generation GAs (HG-GA) concept to approach the fast convergence. By the HG-GA, a pipelined mechanism can be adopted to speed up the chromosome generating process. Due to the fast convergent property of HG-GA, which becomes possible to provide an efficient scheduler for switching variable-length packets in high-speed and multi-channel optical networks.

1 Introduction

The fast explosion of Internet traffic demands more and more network bandwidth day by day. It is evident that the optical network has become the Internet backbone because it offers sufficient bandwidth and acceptable link quality for delivering multimedia data. With the dense wavelength division multiplexing (DWDM) technique, an optical fiber can easily provide a set of parallel channels, each operating at different wavelengths [1], [2]. In each channel, the *statistical multiplexing* technique is used to transport data packets from different sources to enhance the bandwidth utilization. However, this technique incurs the complicated packet scheduling and channel assignment problem in each switching node underlying the tremendous wavelengths. Hence, it is desired to design a faster and more efficient scheduling algorithm for transporting variable-length packets in high-speed and multi-channel optical networks.

So far, many scheduling algorithms for multi-channel networks have been proposed and they are basically designed under two different network topologies: the star-based WDM network and the optical interconnected network. The

© Springer-Verlag Berlin Heidelberg 2003

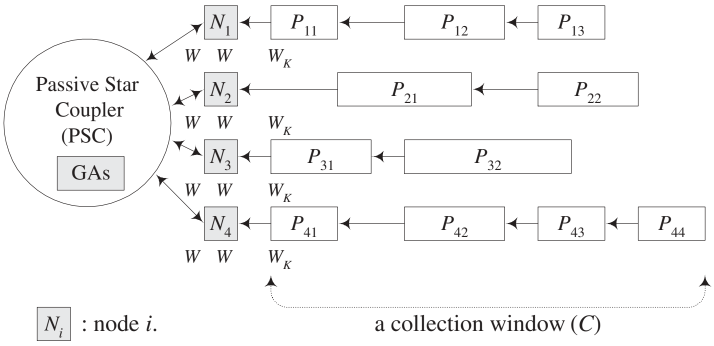

Passive Star Coupler (PSC)
GAs
N_1
N_2
N_3
N_4
P_{11} P_{12} P_{13}
P_{21} P_{22}
P_{31} P_{32}
P_{41} P_{42} P_{43} P_{44}
W W W_K
N_i : node i.
a collection window (C)

Fig. 1. An example illustrates the packet scheduling problem in a star-based network.

star-based network consists a passive start coupler (PSC), which is in charge of coupling packets/messages from different wavelengths and broadcasting all wavelengths to every connected node. The star-based network is often built for local area network due to its centralized control [3], [4], [5]. On the contrary, the other switching component, optical cross connect (OXC), performs efficiently in the space, timing and wavelength switching/converting, and thus, is often used in the optical backbone network [6]. An example shown in Fig. 1 is given to illustrate the scheduling problem in a star-based network while a number of packets with variable lengths from four nodes (N) arriving at PSC, in which there are K parallel channels per fiber (in usual, the number of nodes is smaller than wavelengths.). In this figure, notation P_{ij} is denoted as the j-th packet from node i. In order to minimize the total packet switching delay and maximize the channel utilization, these packets should be well scheduled in K available channels. In the literatures, the scheduling of sequencing tasks for multiprocessor has been addressed extensively and proved as an NP-hard problem [7]. Similarly, the packet scheduling and wavelength assignment problem under constraint of sequence maintenance is also well known as a difficult-to-solve issue. We believe that it is hard to design a real-time scheduling algorithm to resolve the NP-hard problem by general heuristic schemes.

In the past few years, Genetic Algorithms (GAs) have received considerable attention regarding their potential as an optimization technique for complex problems and have been successfully applied in the areas of scheduling, matching, routing, and so on [7], [8], [9], [10]. The GAs mimic the natural genetic ideas to provide excellent evolutionary processes including *crossover*, *mutation* and *selection*. Although GAs have been already applied in many scheduling and sequencing problems, the slow convergence speed of typical GAs limits the possibility of applying them in real-time systems (e.g., network optimization problems often require short response time for each decision.). It has been a major drawback for this mechanism. To overcome the drawback, in this paper, we proposed a pipeline-based hyper-generation GA (HG-GA) mechanism for solving this tough packet schedule problem. The proposed HG-GA mechanism adopts a hyper-generation concept to break the thumb rule of performing crossover on two chromosomes of the same generation. By generating more 'better' chromosomes than the general GA (G-GA) mechanisms within a limited time interval, the HG-GA mechanism can improve significantly in convergence speed, as compared with the G-GA mechanisms. Therefore, the proposed HG-GA mechanism makes the possibility of employing GAs to solve the complicated optimization problem in any real-time environment.

The rest of paper is organized as following. The general GAs packet scheduler (G-GAPS) is introduced in Section 2. In Section 3, we describe the proposed hyper-generation GAs packet scheduler (HG-GAPS) and analyze the precisely generating time of each offspring chromosome. Section 4 provides the simulation comparison to compare the performance difference between these two technologies. Finally, some conclusion remarks are given in Section 5.

2 General Genetic Algorithms Packet Scheduler (G-GAPS)

The G-GA mechanisms applied in industrial engineering contain three main components: the Crossover Component (XOC), Mutation Component (MTC) and Selection Component (SLC), as show in Fig. 2. A process to apply the G-GA mechanisms in solving the optimization problem of packet scheduling and wavelength assignment in networks is named as the general GAs packet scheduler (G-GAPS) [11], [12]. Basically, the G-GAPS needs a collection window (C) for collecting packets. As soon as the scheduling process is executed, a new collection window will be started. This workflow will smooth the traffic flow if the window size is properly selected.

2.1 Definition

In G-GAPS, packets destined to the same output port are collected and permutated for all available wavelengths to form a *chromosome* (i.e., a chromosome presents a kind of permutations), in which each packet is referred to a *gene* [11], [12]. The example showed in Fig. 3 demonstrates that a set of collected packets (P) with different lengths (l) and time stamps (T) are permutated for two available wavelengths (W_1 and W_2) to form a chromosome. A number of chromosomes, denoted as N, will be first generated to form the *base generation* (also called the *first generation*). Following the sequencing, each arrival packet is associated with a time stamp and all permutations will follow these time stamps of packets to be executed as in the scheduling principle. Therefore, the problem becomes how to decide the switch timing and the associated wavelength of a packet so that the precedence relations of the same connection can be maintained and the total required switching time (TRST) of schedule can be minimized. More definitely, the TRST presents the maximum scheduled queue length of packets assigned into different wavelengths. Therefore, we can define the TRST(j) by the formula:

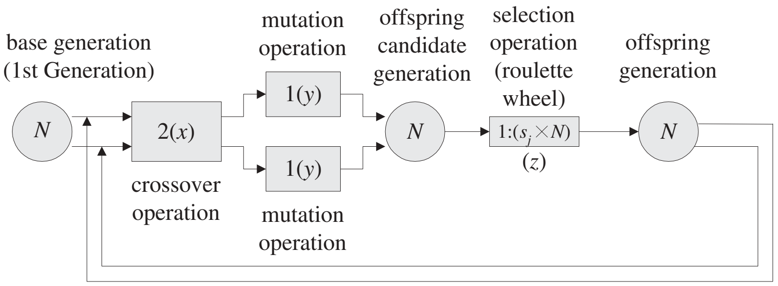


Fig. 2. The flow block diagram of the G-GAPS.

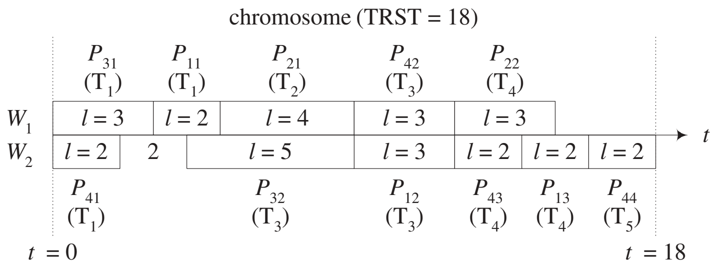


Fig. 3. An example presents a permutation to form a chromosome.

$$TRST(j) = \max\{trst(W_j(1)), trst(W_j(2)), ..., trst(W_j(K))\}. \tag{1}$$

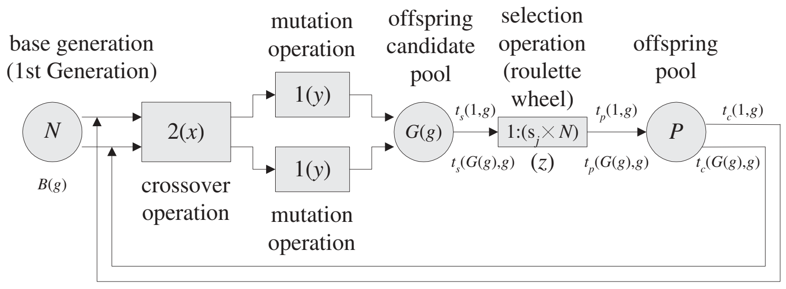
where j is the j-th chromosome, and $trst(W_j(k))$ is the TRST for these packets scheduled in the k-th wavelength of the j-th chromosome. Here we assume an optical fiber carries K wavelengths.

2.2 The Fitness Function

The *fitness function*, denoted as Ω in the G-GAPS is defined as the *objective function* that we want to optimize. It is used to evaluate chromosomes during selection operation to determine which offspring should be remained as the parents for the next generation. The objective function in the scheduling is the TRST and it is often converted into maximization form. Thus, the fitness value of the j-th chromosome, denoted as $\Omega(j)$, is calculated as following:

$$\Omega(j) = \Psi_1^{worst} - TRST(j). \tag{2}$$

where $\Psi_1^{worst} = \sum_u \sum_v l_{uv}$ represents the worst TRST in the first generation (i.e., all packets are scheduled in one wavelength.). Therefore, the optimal schedule will be the chromosome with the largest fitness value denoted as Ω^{opt}.



P : The accumulated number of offspring chromosomes generated from selection operation.

Fig. 4. The flow block diagram of the HG-GAPS.

2.3 Implementation of the Genetic Algorithms

In G-GAPS, each crossover operation selects two chromosomes from the same generation and generates two new offspring chromosomes as the candidates for the next generation. These candidate offspring will involve in mutation operation and selection operation according to their mutation probabilities (P_m) and fitness values, respectively [11], [12].

In the implementation, we simply assume the number of chromosomes in the base generation, say N, is even. Let P_c and P_m denote as the crossover and mutation probabilities, respectively. According to roulette wheel method, the selected probability of the j-th chromosome is $s_j = \Omega(j)/\sum_{r=1}^{r=N} \Omega(r)$.

3 Hyper-generation GAs Packet Scheduler (HG-GAPS)

Basically, the G-GAPS is a generation-based scheme, which processes chromosomes generation by generation. In this scheme, the population size in each generation is kept to N. It means that the selection operation is only triggered when all crossovers and mutations on the chromosomes in a generation are completed, and, also all fitness values of N chromosomes are calculated to support the roulette wheel method. These restraints cause considerable waiting time for propagating good chromosomes to the next generation. For general optimization problems, they do not require quick response time, thus, such a batch behavior will work well to provide an acceptable solution. However, the features of long waiting time and slow convergence speed definitely block the G-GAPS to be a suitable solution for real-time systems. In this section, we will introduce a pipeline-based mechanism, named hyper-generation GAPS (HG-GAPS), to overcome the potential drawbacks of G-GAPS.

As shown in Fig. 4, the key feature of the HG-GAPS is to adopt the pipeline concept and to discard the generation restraint to accelerate convergence speed. From Fig. 4, at the candidate state after mutation operation, the number of

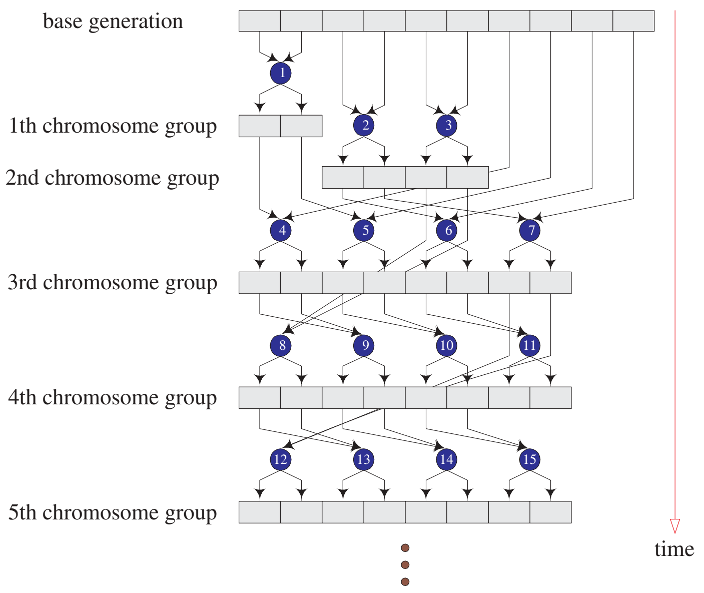


Fig. 5. An example demonstrates the concepts of the chromosome groups and the hyper-generation crossovers in the HG-GAPS when there are $N = 10$ chromosomes in the base generation.

offspring chromosomes is a function of g, which presents the number of '*chromosome group*', as shown in Fig. 5. HG-GAPS uses '*chromosome group*' concept instead of '*generation*' concept to break the crossover limitation in the same generation (i.e., batch operation). In other words, the member of a chromosome group may be generated from parent mating with parent, parent mating with offspring, or offspring mating with offspring.

3.1 Hardware Block Diagram of the HG-GAPS

The detailed HG-GAPS hardware block diagram is designed and shown in Fig. 6. As mentioned before, all arrival packets destined to the same outlet in a collection window are gathered and queued in a Shared Memory. Each of them is tagged with a global time stamp. In the Shared Memory, packets with the same time stamp are linked together. At the end of collection window, packets of the same link are concurrently assigned into K wavelengths through an $M \times K$ switch in a random manner to form a chromosome (where the number of the inlets (I) is M, and K presents the number of wavelengths (W) in a fiber.). This procedure is repeated in the Chromosome Generator until a number of N chromosomes are generated to form the base generation. To promote a more efficient scheduling process, the first two newborn chromosomes in the base generation will be immediately forwarded into the XOC once they are generated.

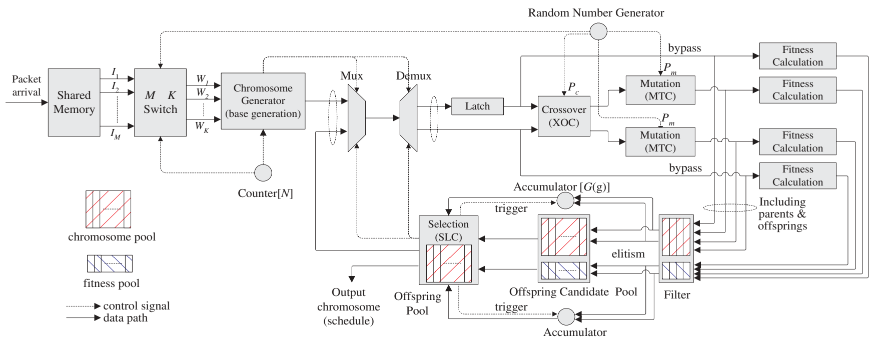


Fig. 6. The hardware architecture of HG-GAPS.

Before the system generates the first offspring chromosome, two chromosomes needed for the XOC are provided from the Chromosome Generator and this time period is named as the *start-up* phase. Then the system enters the *warm-up* phase as soon as the first offspring participates the crossover operation. And the system will switch the chromosomes from the base generation and the Offspring Pool (which is included in the SLC.) in a round robin manner. How fast of the timing for the system entering the *warm-up* phase. It depends on the processing speeds of GA components and the number of remainders in the base generation. Once the base generation runs out of its chromosomes, the system will enter the *saturation* phase, where both participators for crossover operation are provided from the Offspring Pool. Afterward, the HG-GAPS becomes a closed system, in which the cycle processing delay is constant. In the *warm-up* or the *saturation* phase, the chromosome first arriving at the XOC must be buffered in the Latch in order to synchronize the crossover operation with another chromosome.

In the *saturation* phase, the HG-GAPS behaves more like the conventional G-GAPS. Nevertheless, there are two significant differences between them: (1) The offspring generating procedure in the HG-GAPS is still faster than G-GAPS due to all components in the HG-GAPS system are executed in parallel. On the contrary, in the G-GAPS, the SLC cannot work unless the mutation operation has been completed. Afterward, when the SLC performs selection process, the other two components are also stalled. The *stop-and-go* behavior is the well-known drawback in most batch systems. (2) The number of the chromosomes circulating in both systems may differ from each other even when the population sizes in their base generations are set to equal. In the G-GAPS, the offspring chromosome will be selected and collected to form a new generation, and the population size of the new generation is the same as previous one. This feature has no longer been maintained in the HG-GAPS. Due to the limited pages, the analyses of the timings of generating chromosomes, the population size of each group and the convergent speed are not included in this paper.

A Random Number Generator is required for the XOC and the MTC to generate the desired crossover probability P_c and mutation probability P_m. In addition, it also provides a random number for some random processes in the GA operation. After the crossover operation, two mated chromosomes are separately forwarded into the MTCs. Meanwhile, they are also bypassed to Fitness components, one for each, to calculate their fitness values and then to queue in a temp pool (i.e., the Filter). As the pairs of the original parents and the produced offsprings are all stored in the temp pool, two chromosomes with better fitness values will be selected and pushed into the Offspring Candidate Pool for elitism, which is similar to the concept of enlarged sampling space [8].

Finally, the SLC equips two Accumulators: one is used to accumulate the fitness values of the current chromosome group and the other is used to count the number of chromosomes queued in this group. Both of them dedicate the necessary information for the roulette wheel method adopted in SLC. When the last chromosome queued in Offspring Pool is forwarded to the XOC, the Offspring Candidate Pool will pass whole group of chromosomes into SLC by selection and duplication. (That is why we use the '*chromosome group*' to replace '*generation*' as a set of chromosomes to calculate the selection probability of the chromosome in the HG-GAPS.) Thus these two Accumulators will be reset for the next group. As soon as the offspring is produced in the Offspring Pool, it can be as a new parent for the next GA cycle.

4 Simulation Model and Results

4.1 Simulation Model

In the simulation, we construct the GAPS simulation model with several realistic system parameters: the number of time units consumed by each XOC ($= x$), MTC ($= y$) and SLC ($= z$). Besides, there are N chromosomes in both base generations in the G-GAPS and the HG-GAPS. During simulating, we set the time units to be $N = 10$, $x = 2$, $y = 1$ and $z = 2$. (Here, we assume the crossover and the selection operations are more complicated than the mutation operation.) The simulation probabilities of the crossover (P_c) and the mutation (P_m) operations are 0.9 and 0.05, respectively. To simplify the model, we consider the deterministic service rate in each wavelength is measured in the preset time units. The traffic arrival rate of a wavelength in each input fiber is following a Poisson distribution with a mean λ. The packet length is following an exponential distribution with a mean L in the preset time units. The number of the wavelengths in each input or output fiber is K. Thus, the total traffic load Λ is equal to $K \times \lambda \times L$. Furthermore, in order to simulate a real-time system, we fix the scheduling time period to enforce the G-GAPS and the HG-GPAS to output its current optimal schedule within the due time.

4.2 Simulation Results

Fig. 7 shows the average TRSTs derived from the G-GAPS and the HG-GAPS under the variable collection windows (C) and fixed the scheduling time interval

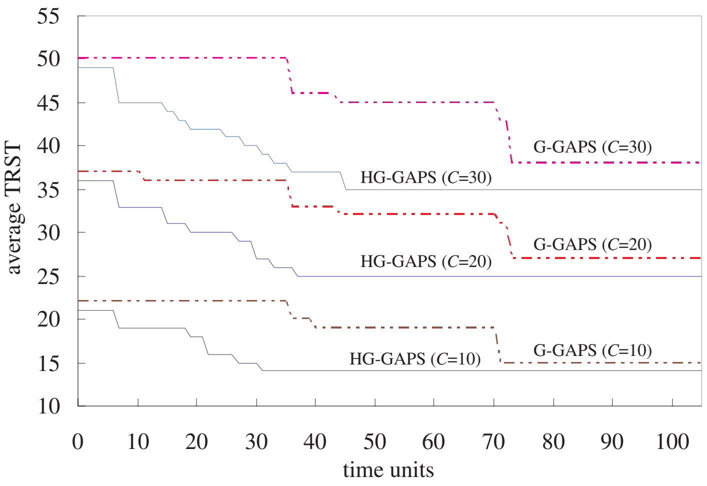


Fig. 7. The average TRSTs are simulated under the different collection window sizes (C) from 10 to 30 time units, when $K = 8$, $\Lambda = 8$ and $L = 5$.

at $K = 8$, $\Lambda = 8$ and $L = 5$ time units. Here, we consider the collection window varying from 10 to 30 time units and the scheduling time period is fixed at 105 time units. In Fig. 7, we can see that the G-GAPS will generate a schedule with a smaller TRST as soon as a generation is completed. That is, the improvements on TRST in the G-GAPS will occur at the time units of 35, 70 and 105. On the contrary, our HG-GAPS starts to minimize the TRST within a short period and obtains the near-optimal TRST at approximate 35 time units. In addition, we also note that under a larger collection window size, the more gain in the decrease TRST will be obtained by the HG-GAPS comparing to the G-GAPS.

Fig. 8 presents the difference in the accumulated chromosome generating rates between the G-GAPS and the HG-GAPS. During the same scheduling time period of 105 time units, the G-GAPS evolves three generations (including the first generation) and only generates 30 chromosomes. On the contrary, the HG-GAPS requires a shorter period to increase the generating rate than the G-GAPS due to its chromosome group and the pipeline concepts. HG-GAPS does not only have the advantage in continuously generating offsprings during a short period, but also keeps the advantage in having a large candidates space for selection operation. Therefore, HG-GAPS can evolve 56 chromosomes during 105 time units.

Fig. 9 shows the consecutively snap shops during a period of 500 time units with a randomly selection from the whole simulation run. We set both of the scheduling time period and the collection windows to be 50 time units. The other system parameters are set as following: $K = 8$, $\Lambda = 6.4$ and $L = 5$ time units. Within a limited scheduling window, the HG-GAPS always provides a

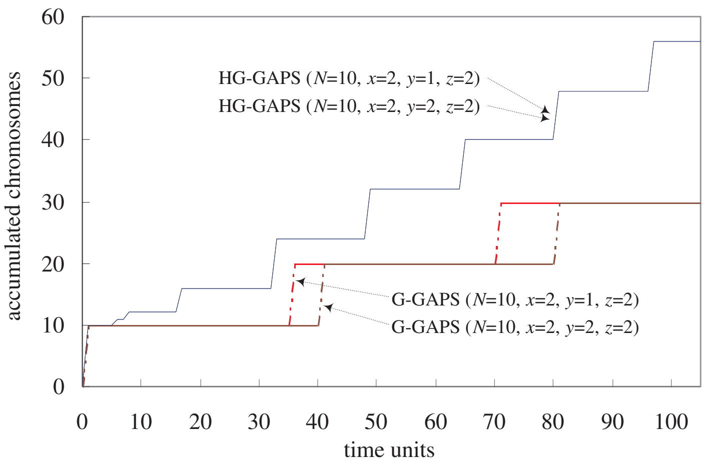


Fig. 8. An illustration to present the accumulated chromosomes generated by the G-GAPS and the HG-GAPS under the different system parameters.

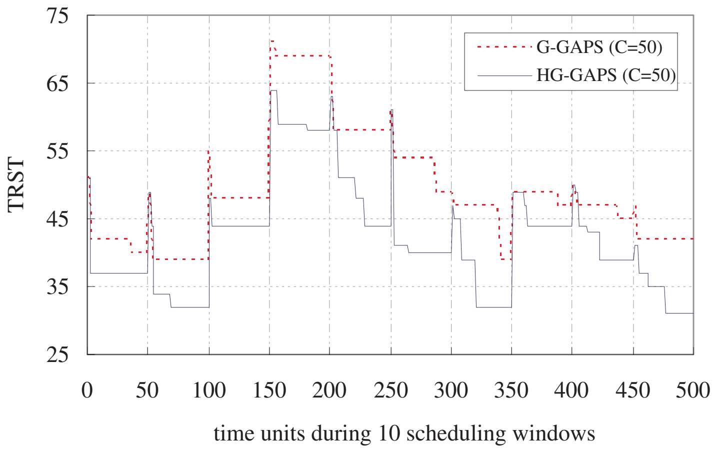


Fig. 9. A comparison between the G-GAPS and the HG-GAPS in the TRSTs of the chromosomes during 10 consecutively scheduling windows.

smaller TRST than the one from the G-GAPS. In fact, if we further shorten the scheduling window to conform a real-time situation, the performance difference between these two GAPSs will become more obvious. During a very short period, the HG-GAPS can generate a scheduling result to approach to a near-optimal solution, but the G-GAPS cannot. In a real continuous transmission environment, a larger TRST from the data transmission will defer the following scheduling tasks. Thus, the difference between the accumulated TRSTs from the G-GAPS and the HG-GAPS will become larger and larger as the time expired, and the packet loss is also getting larger due to the buffer overflow. Therefore, we conclude that the proposed HG-GAPS cannot only indeed provide a significant improvement in solving an optimization problem, but also further support more complicated real-time systems.

5 Conclusions

In this paper, a novel and faster convergent GAPS mechanism, the hyper-generation GAPS (HG-GAPS) mechanism, for scheduling variable-length packets in high-speed optical networks was proposed. It is a powerful mechanism to provide a near-optimal solution for scheduling an optimization problem within a limited response time. This proposed HG-GAPS utilizes the hyper-generation and pipeline concepts to speed up the way of generating chromosomes and to shorten the evolutional consuming time produced from traditional genetic algorithms. From the simulation results, we proved that the HG-GAPS is indeed more suitable for solving the complex optimization problems, such as the packets scheduling and the wavelength assignment problem in a real-time environment.

References

1. Charles A. Brackett: Dense Wavelength Division Multiplexing Networks: Principles and Applications. IEEE J. Select. Areas Communication, Vol. 8, No. 6, pp. 948–964, August (1990).
2. Paul Green: Progress in Optical Networking. IEEE Communications magazine, Vol. 39, No. 1, pp. 54–61, January (2001).
3. F. Jia, B. Mukherjee, J. Iness: Scheduling Variable-length Messages in A Single-hop Multichannel Local Lightwave Network. IEEE/ACM Trans. Networking, Vol. 3, pp. 477–487, August (1995).
4. J. H. Lee, C. K. Un: Dynamic Scheduling Protocol for Variable-sized Messages in A WDM-based Local Network. J. Lightwave Technol., pp. 1595–1600, July (1996).
5. Babak Hamidzadeh, Ma Maode, Mounir Hamdi: Efficient Sequencing Techniques for Variable-Length Messages in WDM Network. J. Lightwave Tech., Vol. 17, pp. 1309–1319, August (1999).
6. Sengupta S., Ramamurthy R.: From Network Design to Dynamic Provisioning and Restoration in Optical Cross-connect Mesh Networks: an Architectural and Algorithmic Overview. IEEE Network, Vol. 15, Issue 4, pp. 46–54, July-Aug (2001).
7. Edwin S.H. Hou, Nirwan Ansari, Hong Ren: A Genetic Algorithm for Multiprocessor Scheduling. IEEE Transaction on Parallel and Distributed Systems, Vol. 5, No. 2, February (1994).
8. Mitsuo Gen, Runwei Cheng: Genetic Algorithms and Engineering Design. Wiley Interscience Publication, (1997).
9. J. S.R. Jang, C. T. Sun, E. Mizutani: Neuro-Fuzzy Soft Computing. Prentice-Hall International, Inc., Chapter 7.
10. D. E. Goldberg: Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley, Reading, MA, (1989).

11. Shiann-Tsong Sheu, Yue-Ru Chuang, Yu-Jie Cheng, Hsuen-Wen Tseng: A Novel Optical IP Router Architecture for WDM Networks. In Proceedings of IEEE ICOIN-15, pp. 335–340, (2001).
12. Shiann-Tsong Sheu, Yue-Ru Chuang: Fast Convergent Genetic Algorithm for Scheduling Variable-Length Packets in High-speed Multi-Channel Optical Networks. Submitted to IEEE Transaction on Evolutionary Computation, (2002).