15

A Hybrid Genetic Approach for Channel Reuse in Multiple Access Telecommunication Networks

Ioannis E. Kassotakis, Maria E. Markaki, and Athanasios V. Vasilakos

CONTENTS

ABSTRACT The evolving Broadband Integrated Services Digital Network (B-ISDN) is reinforcing the demand for high-speed and high-performance multiple access networks. The number of channels available to support the isochronous traffic in these networks is limited by technology, due to implementation costs. We introduce a method using channel sharing/reusing in an effort to provide efficient management of isochronous traffic under this limitation. The proposed method is based on a hybrid genetic algorithm and aims to accomplish the establishment of a maximal number of connections with the minimal number of isochronous channels. Experimental results are provided and they are compared with those of a deterministic graph coloring algorithm. The performance of the proposed algorithm in all simulation runs reveals the robustness, the flexibility, and the efficiency of using evolutionary approaches to complex real-world problems.

KEY WORDS: *Hybrid Genetic Algorithm, NP-Complete, Isochronous Service, Channel Reuse.*

15.1 Introduction

A common design issue to all modern telecommunication networks is the efficient use of the available bandwidth (b/w). In all cases, the network stations have access to a multiple access physical medium of finite b/w required to support all the communication services. The physical medium can either be the air (e.g., wireless, radio, mobile networks) or a hardwired common medium (e.g., cable, optical fiber). In both cases, the concept of *channel* is used to describe and model a finite amount of b/w, which is adequate to satisfy the needs of the basic service provided to the network users. The b/w of a channel may be a frequency band for Frequency Division Multiple Access (FDMA) networks or a *time slot* for the Time Division Multiple Access (TDMA) networks.

The network services are divided into two major categories: The connectionless services that do not have strict time constraints (file transfer, e-mail, etc.) and the time critical connection-oriented services (video conferencing, telephony etc.) that require the continuous availability of guarantee b/w to achieve the desired quality of service (QoS). The connection-oriented services, when implemented at high speed multiple access networks are usually referred as isochronous services and the b/w shares (Virtual Channels) that they use when they employ TDMA schemes, are called Isochronous Channels (usually 64 Kbit/sec).

Channel management is an important aspect for high-performance networks to guarantee the QoS of critical and real-time traffic. It is evident that if channel assignment is improperly implemented, it can be b/w abusive. Hence, *Channel sharing/reusing* is one approach toward efficient management of isochronous traffic, since it increases the network's throughput and

decreases the fraction of blocked messages. Typical applications where channel sharing/reusing is critical are the following:

- file transfers in companies, organizations with hierarchical structure, where multi-person connections are established between neighboring bureaus;
- video telephony and conferencing;
- collaborative systems with electronic libraries, databases, etc., located in stations between the active stations;
- LAN interconnection of heterogeneous types, i.e., Ethernet, Token ring, FDDI, Fast Ethernet by a high speed, distributed dual bus, where traffic is concentrated mainly between the stations of each LAN.

In this study, we are interested in improving the provision of isochronous service and particularly in the problem of reusing isochronous channels, referred as *Isochronous Channel Reuse Problem* (*ICRP*), in topologies that involve a medium of multiple access (DQDB, FDDI, Ethernet, etc.). The task for the solution of ICRP is to assign isochronous channels to newcoming requests with the twofold issue of serving the maximum number of isochronous connections and employing the least number of channels.

Beyond its application to isochronous traffic in ISDN networks, the Isochronous Channel Reuse Problem (ICRP) applies to all types of communication networks. In satellite communication systems, the cochannel interference has become a major factor of determining system designs and operations to secure the communications quality.[1] Moreover, with the dramatic increase of geostationary satellites, two or more adjacent satellites can often cause the intersystem interference by using the same frequency. In order to cope with the cochannel interference problem, the rearrangement of frequency assignments has been considered as an effective countermeasure in practical situations. Similar channel assignment considerations apply for the TDMA and FDMA cellular networks.[2,3]

Mathematically speaking, channel assignment appears as a combinatorial optimization problem that is closely related to graph coloring, and as such known to be NP-complete.[4] Thus, an exact search for the best solution in real-time applications is infeasible, due to an exponentially growing calculation time. The fact that ICRP can be modeled as a graph coloring problem has led researchers to use a wide variety of algorithms for its solution. These algorithms range from deterministic graph coloring algorithms to more intelligent schemes involving neural networks, simulated annealing, etc.

The use of intelligent techniques is becoming an increasingly attractive practice in an effort to overcome the complex combinational problems that arise during the design and development of High Speed Networks. The classic combinational and analytical methods fail to produce solutions within the strict time constraints that these networks are bounded to operate.[5,6,7]

In this chapter, we propose a hybrid genetic algorithm (HGA) for the ICRP solution. A hybrid genetic algorithm is a genetic algorithm coupled with a

local improvement operator. The reason for combining a genetic algorithm with a local search algorithm is that they complement each other and therefore the reliability properties of the genetic algorithms are combined with the accuracy of hill-climbing methods. Genetic algorithms and hybrid genetic algorithms were found to be well suited for similar NP-complete problems.[8, 9, 10, 11] since they adopt a global strategy, rely on intelligent randomization, and explore a number of possible alternatives simultaneously. The performance of the HGA is compared via simulation to that of a graph coloring algorithm (GCA) proposed in Reference 12. A Dual Bus Distributed Queue (DQDB) network has been used as a testbed for the evaluation of the algorithms' performance.

The rest of the paper is organized as follows: In Section 15.2, we provide the definition of ICRP and a brief description of the DQDB testbed network. Section 15.3 presents the deterministic graph coloring algorithm (GCA) and discusses the genetic algorithms. The customization of hybrid genetic algorithm (HGA) for the ICRP solution is addressed in Section 15.4. Section 15.5 introduces the simulation models and assumptions and Section 15.6 presents and comments on the simulation results. Section 15.7 concludes the paper.

15.2 Problem Definition

In this section, we overview the DQDB network, used as a test-bed in our simulation experiments, we state the ICRP, and formulate the ICRP as a graph coloring problem.

15.2.1 Isochronous Traffic and DQDB Topology Definitions

Consider a DQDB network that interconnects N nodes labeled 1, 2, . . . , N (station index) from left to right as shown in Figure 15.1.[13] The underlying network of the DQDB consists of two unidirectional slotted buses operating in opposite directions. This allows full duplex communications between each pair of nodes, providing isochronous (circuit switched) and nonisochronous (packet switched) services. Every node receives and transmits on both buses, via read and write connections, so bus selection is based on the destination. Nodes use F-Bus to transmit to stations of higher indexes and R-Bus to transmit to stations of lower indexes. Transmissions on the two buses are independent; thus the effective data rate of a DQDB network is twice the data rate of the bus.

The data on each bus is formatted by the node at the head of each bus, called slot generator, into either DQDB layer management information octets or fixed-length units called *slots* with a length of 53 octets. Each slot contains an Access Control Field (ACF) that contains the protocol control information and a segment that forms the payload of the slot. The management

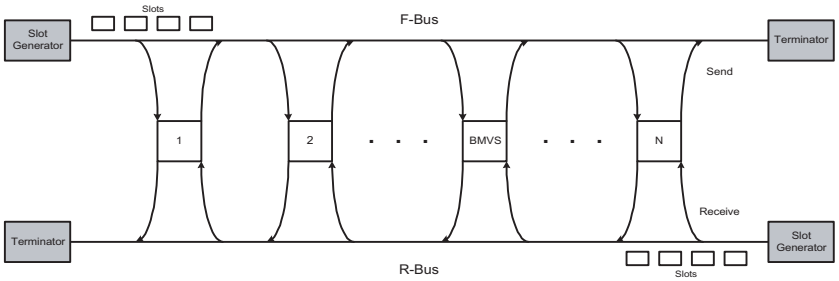

FIGURE 15.1
15.1 The DQDB network topology.

information octets are used to maintain the operational integrity of the network. The slots are used to carry data between the nodes. All data flow terminates at the end of each bus, at the slot sink.

The DQDB layer provides two modes of access control to the dual bus. These are Queued Arbitrated and Pre-Arbitrated, which use QA and PA slots for data transfer, respectively. Queued arbitrated access is controlled by the distributed queuing protocol[14, 15] and would be used typically to provide nonisochronous services, while pre-arbitrated access would be used typically to provide isochronous services. Next, we describe in detail only the access to PA slots since the PA slots are used to provide isochronous service.[14]

Access to PA slots: For each PA slot, there are 48 octets of payload, each of which can be occupied by a single station to form a 64 kbps isochronous channel. A sequence of coherent PA slots can either provide 48 distinct 64 kbps isochronous channels or a number of c_i*64 kbps isochronous channels with $\Sigma c_i = 48$. To maintain this b/w, the slot generator located at the head of each bus must generate at least one PA slot every 125 μsec for channels with a total b/w requirement of (48*64) kbps. To support larger b/w requirements, the slot generator may generate several PA slots in every 125 μsec. The header of a PA slot consists of a Virtual Channel Identifier (VCI), which is used to identify the PA slots that belong to the same isochronous channel. In order to identify which octet in a PA slot is occupied by a specific isochronous channel, an offset value is used. Thus, a VCI/Offset pair uniquely identifies a 64 kbps isochronous channel.

The source node that wants to establish an isochronous connection sends a channel request to the Bandwidth Manager and VCI Server (BMVS) via QA slots. The BMVS is an administrator node usually located at the center of the dual bus (Figure 15.1). The designation of slots to PA access and the marking of the PA slot header is controlled by BMVS. In particular, the BMVS must write the Virtual Channel Identifier (VCI) field into PA slots. The BMVS also ensures that the PA slots for each VCI are provided in a periodic manner on the bus to guarantee that sufficient b/w is available for isochronous service users.

In this approach, after receiving the channel request from a source node, the BMVS will accept the request if there is available b/w. If the request is accepted, the BMVS sends to the source node, via QA slots, a message containing the VCI(s) and the offset(s). After receiving this message, the source node uses QA slots to pass this information downstream to the destination node. Then, both the source and the destination nodes examine the VCI value on every PA slot, and use a local table to determine whether the VCI is intended to be used. If the node is intended to use the VCI, then the table indicates which offset octets within the slot should be used for reading and writing. The source node writes the octets in positions marked for writing and the destination node reads the octets in positions marked for reading. If the VCI is not in use by the node, then the PA slot is ignored. At the end of the session, the source node sends a message to BMVS, via QA slots, to indicate the end of the connection. Then, BMVS marks the particular offset(s) of the VCI(s) as available and is free to allocate them for another connection request.

15.2.2 Statement of ICRP

The terminology used in this paper is as in Reference 12. Let $X = \{x_1, x_2, \ldots, x_{n_c}\}$ denote a set of established isochronous connections, in which $x_i = \{s_i, d_i, O_i\}$ stands for an established isochronous connection between stations s_i and d_i with a set O_i of consecutive offsets (O_i represents the b/w in use). Let $Y = \{y_1, y_2, \ldots, y_{n_r}\}$, $y_i = (s_i, d_i, w_i)$ denote a set of isochronous requests; where y_i is the isochronous request issued from station s_i, and d_i, w_i are the destination station and the required b/w (number of 64 kbps isochronous channels), respectively. It is assumed that a station can issue several isochronous requests simultaneously to the same or different destinations. This means that for a pair of stations, the establishment of multiple isochronous connections is allowed. We say that a request y_i is *intersected* with another request y_j (or an established isochronous connection x_k) if $s_i = s_j$ or $s_i < s_j < d_i$ ($s_i = s_k$ or $s_i < s_k < d_i$). It is evident that two intersected isochronous requests/connections can not share the same isochronous channel(s).

Let the consecutive offsets to be used by request y_i be called range R_i. We say that $R(X, Y) = \{R_1, R_2, \ldots, R_{n_r}\}$ is a *ranges arrangement* of Y; where $R_i = (s_i, d_i, \{o_i\})$ and o_i is the arranged starting offset of request y_i, if for each pair of intersected requests in Y and/or established isochronous connections in X, their arranged ranges do not occupy an overlapped offset. Let $w(R(X, Y))$ denote the total number of assigned offsets of $R(X, Y)$. Let $R^*(X, Y)$ denote the optimal ranges arrangement of (X, Y) which has the minimum number of assigned offsets $w(R^*(X, Y))$. The problem considered can now be defined:

Isochronous Channel Reuse Problem (ICRP): *Given a set X of established isochronous connections a set Y of isochronous requests, find an optimal ranges arrangement $R^*(X, Y)$.*

In this chapter, in order to simplify the implementation, we assume that all the established isochronous connections as well as the isochronous requests have the same b/w, equal to that of a single isochronous channel (64 kbps).

15.2.3 A Graph Coloring Modeling of ICRP

In this section we briefly state the graph coloring problem[16] and show how the ICRP can be formulated as a graph coloring problem. A *graph* is a set of points called *vertices* with connections called *edges* between pairs of vertices. Given a graph $G = (V, E)$ with vertex set V and edge set E and given a positive integer k, one has to find a *k-coloring* of G, i.e., a partition of V into k independent sets $V_1, \ldots, V_k$ (a set V_i of vertices is called independent if no two vertices in V_i are linked by an edge in G). If such a partition exists, G is called *k-colorable*.

This problem has received much attention for several reasons. First it belongs to the class of problems known as NP-complete. In addition, a large number of practical problems can be formulated in terms of coloring a graph, including clustering and scheduling problems. Additionally, the ICRP can be restated as Graph Coloring Problem if we assume the following:

- The concept of an isochronous channel corresponds to that of a color.
- The established connections are represented as colored vertices.
- The isochronous requests are represented as uncolored vertices.
- Two vertices are connected with an edge if they are intersected.

Let us consider a set X of connections ($x_1 = (1, 7, \{1\})$, $x_2 = (4, 6, \{2\})$, $x_3 = (5, 6, \{3\})$, $x_4 = (7, 8, \{3\})$) and a set Y of isochronous requests ($y_1 = (7, 8, 1)$, $y_2 = (6, 8, 1)$, $y_3 = (3, 8, 1)$, $y_4 = (4, 8, 1)$, $y_5 = (5, 6, 1)$). The graph G with $V = X \cup Y$ is depicted in Figure 15.2, where the intersected pairs of vertices can be easily distinguished. For instance, it can be easily seen that y_1 is intersected with y_2, y_3, y_4, and x_4.

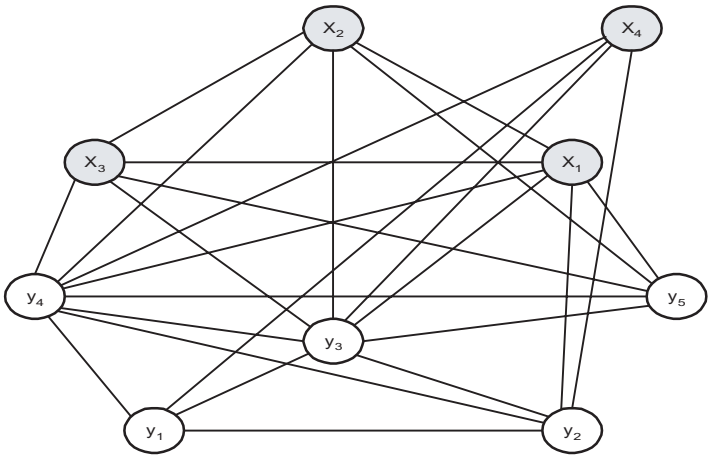

FIGURE 15.2
Example graph.

The ICRP can be restated as follows: Given a graph $G = (V, E)$ with vertex set $V = X \cup Y$ and edge set E, assign any of the uncolored vertices (requests) a color from the set of available colors, with the condition that no pair of uncolored (requests) and colored vertices (connections) connected with an edge shares the same color.

15.3 Algorithms for the ICRP Solution

15.3.1 Review of Graph Coloring Algorithms

In Reference 12 the authors propose an $O(n^3)$ graph coloring algorithm (GCA) to solve the ICRP, where n is the sum of the number of established isochronous connections and the number of isochronous requests. The following steps can describe the channel assignment method of GCA:

1. For every isochronous request find the number of intersected requests and impose a descending order on them.
2. Traversing the requests in this order, perform the following actions for every request:
   a. Find the colors used by the connections that are intersected with the request (unpreferred colors). From the set of available colors remove the unpreferred colors.
   b. If at least one color is found, assign the color (channel) with the lower VCI value to the request.
3. If there are still requests (uncolored vertices):
   a. Arrange them at a linear order. The linear order is defined under the following rules: first come the requests that have the lower source node IDs and if two requests have the same source, then the one with the lower destination node ID will proceed in the linear order.
   b. Color each of the pending requests with the smallest available color, provided there are no intersected connections that share the same color. If there are no available colors left, the request will be discarded.

15.3.2 Overview of Genetic Algorithms

Genetic Algorithms (GAs)[17, 18, 19] first specified by John Holland in the early seventies, are principal "search procedures" based on principles derived from the dynamics of natural population genetics. They have been successfully applied to numerous large space problems where no efficient polynomial-time algorithm is known, such as NP-complete problems.

The problem to be solved in Genetic Algorithms should be represented as one-dimensional or multi-dimensional structures, which represent a search point in the search space. This means that the problem should be encoded, that is, to find a pattern to represent each solution, called *chromosome*, as a chain of characters taken in a finite alphabet. The Simple Genetic Algorithm (SGA)[20] relies on a binary alphabet, and thus solutions appear as chains of "0" and "1". The GAs operate on chromosomes grouped into a set called *population*. Successive populations are called *generations*. Each chromosome is evaluated by the *fitness function*, which reflects its merit and its chances to survive in the next generation.

The operation of the Simple Genetic Algorithm (SGA) can be summarized as follows. The initial solutions, chosen randomly, are encoded as binary strings (chromosomes) and they form the initial population. Each chromosome is associated with a fitness value. The current population is evolved creating a new one with higher fitness value, by using three operators:

a. Selection—This operator selects a chromosome from the current population to survive in the next one. The probability of a chromosome to be selected is proportional to its fitness value. This operator is an artificial version of nature's Darwinian survival of the fittest chromosomes.
b. Crossover—This operator exchanges portions between selected chromosomes, as in nature it recombines the genetic material of two parent chromosomes to produce two children. The chromosomes are broken at the same (random) place and combined together creating two new children chromosomes.
c. Mutation—This operator causes sporadic and random alteration of the binary bits (changing a 0 to 1 and vice versa) of the chromosomes, as in nature it plays the role of regenerating lost genetic material.

The fitness values of the new chromosomes are evaluated by the fitness function, before the next selection process. From that point the algorithm will be repeated. The algorithm stops after a fixed number of generations, or when a chosen criterion reaches a predefined threshold. When optimization heuristics that can provide some domain-based guidance to the search process are incorporated into the GA, the resulting algorithm is called hybrid genetic algorithm. It is expected that if knowledge of the problem is embodied into the GA, then the convergence speed of the hybrid GA will increase.

15.4 Proposed Hybrid GA for the ICRP Solution

The proposed algorithm[21] has the typical components of a hybrid genetic algorithm and along with its specific characteristics it is going to be described next in detail. Figure 15.3 shows the algorithm's layout.

1. Create initial population of *pop_size* chromosomes
2. Evaluate (*pop_size*)

   repeat *pop_size*/2 times
3. Select 2 parents P_1 ,P_2 from the population *pop_size*
4. Offsprings C_1,C_2 $\leftarrow$ Crossover (P_1,P_2)
5. Offsprings C_1',C_2' $\leftarrow$ Mutate (C_1,C_2)
6. Replace if better $P_1 \leftarrow C_1'$

   $P_2 \leftarrow C_2'$

   end;
7. Find the best chromosome of the population, λ
8. Optimize (λ)
9. Output (λ)
10. Update stopping condition
11. Repeat steps 2 to 10 until the stopping condition is met

FIGURE 15.3
The outline of the proposed algorithm.

15.4.1 Representation and Population Initialization

Denote *chan* and n_r^j as the isochronous channels and the requests of chromosome *j*, respectively. Consider w_i^j as the isochronous channel assigned to *i* request of *j* chromosome. For solving the ICRP a binary encoding is used to represent the *j* chromosome, i.e., $V^j = (w_1^j, w_2^j, \ldots, w_{nr}^j)$. The binary matrix representation of *j* chromosome is formed as follows: The matrix is [*chan* $\times$ n_r^j] and if the channel w^j is assigned to the request *i*, the entry (w^j, *i*) of the matrix is set to 1. The remaining entries of the matrix for request *n* are set to 0. The binary encoding of V^j = (2, 2, 1, 4, 3) with *chan* = 4 and n^j = 5 and a more general chromosome representation are given below:

$$\text{chrom}^j = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

,

$$\begin{array}{c} \\ 1 \\ \\ \\ \\ w_i^j \\ \\ \\ chan \end{array} \begin{array}{c} \begin{array}{ccccccc} 1 & . & . & . & i & . & . & . & n_r^j \end{array} \\ \begin{bmatrix} & & & \cdot & & \\ & & & \cdot & & \\ & & & \cdot & & \\ & & & \cdot & & \\ \cdots & & & \cdot & & \\ & & & & & \\ & & & & & \\ & & & & & \end{bmatrix} \end{array}$$

We define as *pop_size* the number of chromosomes. The *pop_size* chromosomes of the initial population are randomly initialized, by assigning to each of the n^j chromosome's requests a randomly generated channel such that

$$1 \leq w_i^j \leq chan, \quad \forall(i = 1, 2, \ldots, n_r^j,\ j = 1, 2, \ldots, pop\_size) \qquad (15.1)$$

15.4.2 Evaluation

The evaluation of a chromosome takes place in order to test its fitness as a solution and is achieved by making use of a fitness function. Let n_ℓ^j denote the set of established connections of j chromosome. Consider z_i^j as the isochronous channel assigned to i connection of j chromosome. In our approach, the fitness function of j chromosome, F^j, is defined as the number of intersected pairs of n^j requests and n^j connections that employ the same isochronous channel. Hence,

$$F^j = \sum_{i=1}^{n_r} f_i^j \qquad (15.2)$$

where f_i^j is defined as the fitness function for the i request of j chromosome and is calculated as shown in the following code.

```
for (i=1; i ≤ n^j; i++)
{
    f_i^j: = 0;
    for (k = 1; k ≤ n^j; k++)
        {
            if ((i request intersected with k connection) and (w_i^j = z_k^j)) f_i^j: = f_i^j + 1;
        }
}
```

15.4.3 Selection

A variety of selection strategies are used, some of which use the fitness value directly (tournament with tournament size of two, roulette-wheel, FPS) and others order the population and allocate parents by rank with a scaling factor of two (linear scaling, sigma truncation). With the exception of tournament and roulette-wheel selections, which are commonly used in GAs, the rest of the selection strategies are briefly described. More details can be found in References 20, 22. FPS (Fitness Proportionate Selection) selects the first parent in proportion to its fitness, while the second parent is picked at random. Linear scaling requires a linear relationship between the scaled fitness, F_s^j, and the raw fitness, F^j, of j chromosome as follows:

$$F_s^j = a\ F^j + b \qquad (15.3)$$

where the coefficients a and b are chosen to ensure that the average population member will receive one offspring on average and the best will receive copies equal to the scaling factor. In sigma truncation the scaled fitness, F_s^j, is given by:

$$F_s^j = F^j - (\text{aver}\{F^j, j = 1, 2, \ldots, pop\_size\} - c\sigma) \tag{15.4}$$

where c is the scaling factor taken equal to two and σ is the standard deviation of the fitness values of the *pop_size* chromosomes.

In the phase of selection, a new fitness function is adopted for each of the *pop_size* chromosomes. It is apparent that during the GA's operation the hard constraint is the requirement that the requests intersected with established connections cannot share the same channel. However, in the phase of selection we are also interested in the number of currently active connections. Hence, due to the soft constraint of maximizing the active connections, the new evaluation function for the j chromosome, F^{j*} becomes:

$$F^{j*} = \max\{F^i, i = 1, 2, \ldots, pop\_size\} - F^j + n_c^j \tag{15.5}$$

Considering the new fitness function, the fittest chromosome is the one that has the highest fitness value of Equation (15.5) determined by the number of intersected pairs of requests and connections that share the same channel and the number of active isochronous connections.

15.4.4 Crossover

The crossover probability *pcross* recombines the genetic material of two parent chromosomes to generate two new ones. We use a one-point crossover; thus the part of the two binary chromosomes i and j after the crossover point is swapped to generate two children. The crossover point stands for column position and is a random number between 1 and min (n^i, n^j).

15.4.5 Mutation

The mutation operator, applied after the crossover operator on each of the offspring chromosomes independently, offers the opportunity for new genetic material to be introduced into the population. If no crossover is to be performed, the mutation operator is applied on the parent chromosomes. In both cases, each of the n^j isochronous requests of j chromosome is assigned a random channel from [1, *chan*]. The operator's probability, *pmutation* is proportional to the contribution the operator has made to the chromosome's fitness. If the use of the randomly selected channel improves the fitness value of the chromosome, we set *pmutation* to 0.4, or else to 0.1.

15.4.6 Replacement

The children chromosomes replace their parent chromosomes in the population to maintain a fixed-population size *pop_size*, if their fitness values are lower than the fitness values of their parents. Since the steady-state GA does not guarantee that the best member in the current population will be present in the next, we include the elitist strategy.[17] The elitism ensures that the chromosome with the best performance always survives intact into the next generation (replaces the chromosome with the worst performance) and we introduce the constraint that the successive replacements with the same best member of the population will not be more than [N/5], in order to avoid premature convergence.

15.4.7 Local Improvement Operator

The motivation for combining the genetic algorithm with a local search algorithm to form a hybrid algorithm is that the genetic algorithm will try to optimize globally, while the local optimizer will try to optimize locally.[23] We introduce a new local improvement operator, which is called Heuristic Algorithm and it works as follows. The algorithm is described in Figure 15.4.

```
for (i=1; i≤ n_r^λ ; i++)
   {
        init_ch := w_i^λ ;  {store the initial channel}
        init_fit := f_i^λ ;   {store the fitness value of init_ch channel}
        vch :=0;
        repeat
                vch :=vch+1;
                w_i^λ :=vch;       {assign vch channel to the i request}
                next_fit := f_i^λ ;   {compute the fitness value of vch channel}
        until ((4*next_fit<=init_fit) or (vch>=chan));
        if (4*next_fit>init_fit) w_i^λ :=init_ch;   {assign the initial channel}
    }
```

FIGURE 15.4
A pseudo-code of the heuristic algorithm.

Our goal is to improve the channel assignment of the pending requests of the best chromosome $\lambda(F^{\lambda} \leq F^{i}, \forall_i\ 1, 2, \ldots, pop\_size)$, by minimizing the value of its fitness function. We examine separately every n_r^{λ} request of the λ chromosome. For each request we assign initially the channel 1 and compute the value of the fitness function. If the computed fitness value is less than 25% of the initial fitness value, the newly assigned channel replaces the initial one. Otherwise, we try the second channel and so on until we find a channel that satisfies the above requirement. In case none of the *chan* channels fulfills the condition, the initial channel is reassigned and we proceed with the next request until all n_r^{λ} requests are inspected. The output of the algorithm will be the solution λ optimized according to the previously described procedure.

15.4.8 Stop Criterion

The algorithm is halted when the generation counter exceeds the maximum number of generations *maxgen* or when convergence is indicated. A good indication of convergence is that a performance index has not changed in a given number of generations, which in our experiments is set to maxgen/2 generations. The performance index is the throughput, which is defined as the total number of connections made by all nodes divided by the total number of requests issued by all nodes. Additionally, the algorithm terminates in case we find the "optimal" solution, i.e., a chromosome with no pending requests, during the procedures of crossover, mutation or local improvement.

15.4.9 Time Complexity

It is easy to check that the time complexity of the parent selection is $O(pop\_size)$. For j chromosome, the crossover operator takes $n^j * chan$. For $n = n^j + n^j\ \forall_j = 1, 2, \ldots, pop\_size$, since $n^j * chan \leq n * chan$ the crossover operator takes $O(n)$ time in the worst case. Similarly, we find that the mutation operator takes $O(n^2)$ time and the time complexity of the local improvement is $O(n^2)$. Therefore, the time complexity for each generation (select 2 parents, apply the crossover and the mutation operator $pop\_size/2$ times, next find the best of the population and finally locally improve it) is:

$$\text{Time Complexity} < O(n^2 * pop\_size^2) \tag{15.6}$$

The GA's setup time (steps 1 and 2 in Figure 15.3) is bounded by $O(n^2 * pop\_size)$.[24]

15.5 Simulation Models and Assumptions

In order to evaluate the performance of the HGA introduced in the previous section, two experimental scenarios were considered. In the first scenario

referred to as *static load*, HGA was tested under static conditions, i.e., a predefined number of connections and requests is generated. In the second referred to as *dynamic load*, a dynamic environment is simulated, where both isochronous requests issued by the DQDB nodes and releases of the established connections are of Poisson distribution. The experimental performance of HGA is also compared with that of GCA.

15.5.1 Static Load Simulations

For the first experimental scenario the ICRP was formulated as follows:

A number of nodes (10 in our experiments) are arranged on a unidirectional DQDB bus of (chan) isochronous channel capacity. At the initial state there are (Con) = (chan) randomly established connections among the nodes and a pool of (Req) requests awaiting channel allocation. Both algorithms are tasked to serve the maximum number of (Req) requests by reusing the (chan) available channels.

The number of (*Req*) requests is normalized and expressed as a function of the available channels (*Req/chan*), thus representing the network's load. The higher the load, the harder the allocation problem becomes.

Two performance measures are used for the evaluation of the ICRP solution:

a. The established connections per available channel (*Conn/chan*) <*Efficiency*>.
b. The percentage of the served requests (*Conn/Req*) <*Throughput*>.
c. The time that each algorithm requires to reach its optimal allocation solution was also monitored. Both algorithms were simulated on the same (630 MIPS) machine using the same resources at all simulation experiments (runs). The allocation time measured in msec corresponds to the time complexity of each algorithm.

15.5.2 Dynamic Load Simulations

Both algorithms were tested on a DQDB network of ten nodes. The isochronous requests generated by each node are transmitted to the BMVS administration node. The BMVS node concentrates the requests from all nodes and periodically employs the allocation algorithm to select the isochronous channel that will serve each of the accumulated requests. The number of requests that are served at every allocation is set to 100. The traffic model that is considered for the generation of the channel requests is a uniform distribution. Thus, every node has equal probability to request a connection with any of the other nodes.

The following assumptions have been made for the DQDB network simulation:

- Network size is set to *Nodes* = 10;
- The stations are equally spaced along the dual bus;
- The load of the network is equally distributed among the stations;
- The arrival rate of the isochronous requests (r_i) in each station is a Poisson process;
- The departure rate of the established connections in each station (r_o) is a Poisson process.

The network load is expressed by the pair of the parameters (r_i, r_o). The quotient r_i/r_o, named *Load Quotient (LQ),* is a measure of the load intensity. Higher *LQ* values indicate heavier network load.

For the evaluation of the algorithm performance in the DQDB network environment, the following two performance measures have been considered:

- The total number of established *connections* after all stations have issued a number of 10000 requests.
- The number of *active connections* maintained by all stations during the simulation run.

15.5.3 HGA Parameters

HGA requires a number of tuning parameters to be preset. After many experimental runs for tuning the genetic algorithm the following parameter values were determined:

- Population size is set to 2* *Nodes* = 20;
- Number of generations is set to 10 for the dynamic load simulation runs and to 1 for the static load simulation runs. It was concluded that for the static load runs, no more than one generation was required to converge to the optimal solution.
- Crossover probability is set to 0.8 (see Reference 19);
- Mutation probability is set to either 0.1 or 0.4 as discussed in previous section.
- Several different selection schemes (linear, tournament, FPS, sigma, roulette) were tested and their results are presented.

The HGA operators have been extensively presented in Section 15.5. The different selection schemes have been simulated in an effort to study their effect on the HGA performance. The selection of the proper GA parameters is a process based both on extensive experimentation and the experience gained by the authors during previous work at the same field. At a real application, there should be a predefined set of parameters that will automatically be applied according to the traffic model of the user requirements.

15.6 Simulation Results

The simulation results presented in this section are derived from 20 simulation runs for each experiment. Experience has showed that at least 20 runs are needed to achieve statistically reasonable results (confidence interval of at least 0.9), regardless of the network size, since the variation of the on-line node number is automatically compensated by the proportional adjustment of the GA population size (*pop_size* = 2* *Nodes*).

15.6.1 Static Load Simulations

In Figure 15.5, we investigate the efficiency of each algorithm at different loads (*Req/chan*). At the lower side of the load curve (2.8 & 3), the GCA and the HGA(s) appear to have similar performance. However, as the number of the requests per channel increases, the HGA(s) continue to accommodate an increasing number of requests using the limited number of network resources (isochronous channels), while GCA diverts to lower number of connections per channel. It is evident that among the HGA(s) using different selection schemes, no significant difference can be noted. For the particular case of *chan* = *Con* = 50, Table 15.1 presents the number of established connections for different request pool size (*Req*). In Figure 15.6 we investigate the throughput of the algorithms by measuring the percentage of the served requests at different load. It is evident that as the algorithm load becomes increasingly heavier, the HGA(s) successfully maintain their QoS by preserving the percentage of the served requests. On the other hand, GCA fails to maintain a constant percentage of served requests.

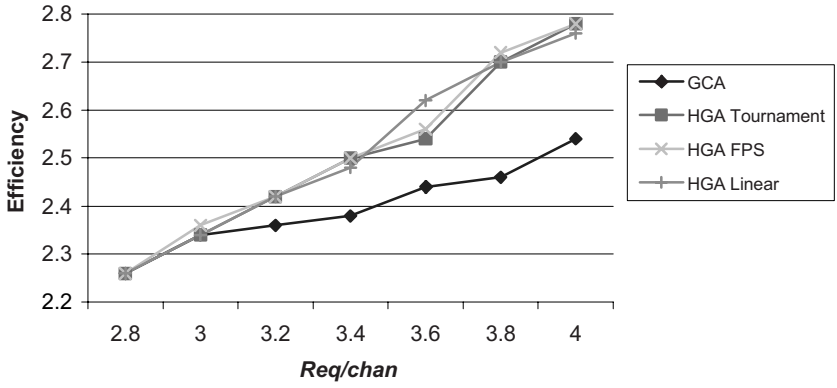

FIGURE 15.5
Channel allocation efficiency at static load.

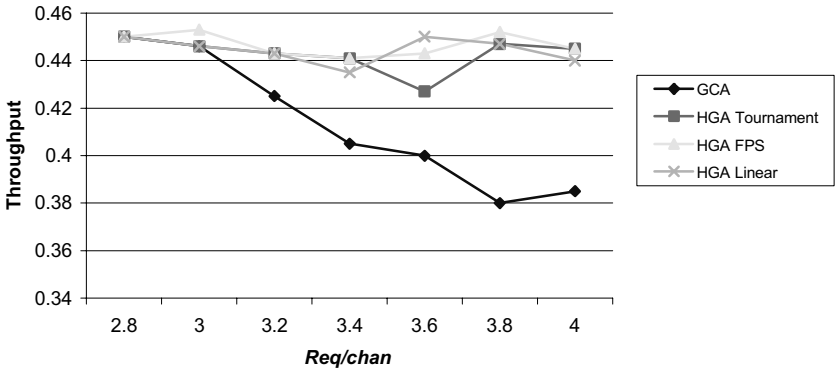

FIGURE 15.6
Channel allocation throughput at static load.

TABLE 15.1

Established Connections for Different Values of *Req* with *Chan* = *Con* = 50

Req	GCA	GA Tournament	GA Roulette	HGA FPS	HGA Sigma	HGA Linear
90	63	63	63	63	63	63
100	67	67	63	**68**	**68**	67
110	68	71	**72**	71	**72**	71
120	69	**75**	**75**	**75**	73	74
130	72	77	79	78	80	**81**
140	73	85	86	**86**	84	85
150	77	**89**	87	**89**	87	88
Average	69.857	75.285	75	**75.714**	75.285	75.571

At both experiments we must note that if excessive numbers of requests are considered, neither the number of connections per channel will continue to increase at increasingly heavier loads, nor will the percentage of served requests be maintained. It is obvious that eventually all available channels will be saturated and no more requests would be served. This case is beyond the scope of our work as it corresponds to extreme traffic conditions.

Figure 15.7 depicts the allocation time of each algorithm. We notice that the allocation time of HGA(s) is only moderately affected by the increasing number of requests compared against the GCA, which consumes exponentially longer times. Comparing the experimental results for the allocation time and the theoretical time resulting from the time complexity formulae of Section 15.4, we conclude that they match in the following manner. As it has been described the HGA's time complexity is proportional to n^2, where $n = n_r + n_c$, while the complexity of GCA is proportional to n^3 (Section 15.3.1). In Table 15.2, the algorithms' allocation time is presented analytically for different values of n. It is evident that the increase of run time curve is steeper for GCA corresponding to the $O(n^3)$ time complexity against the $O(n^2 * pop\_size^2)$ of HGA.

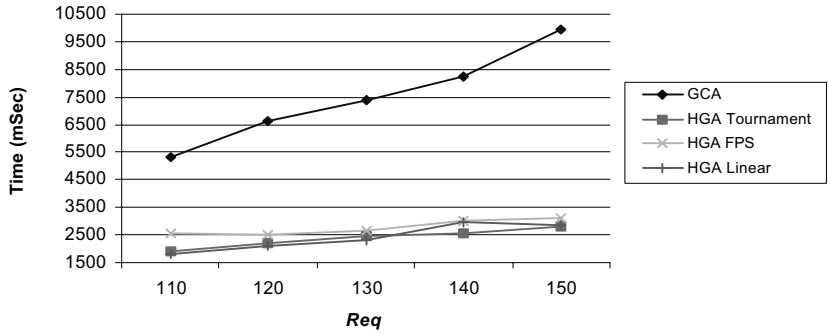

FIGURE 15.7
Channel allocation time at static load.

TABLE 15.2

Channel Allocation Time (mSec) for different values of *Req* with *Chan* = *Con* = 50

REQs	GCA	HGA Tournament	HGA Roulette	HGA FPS	HGA Sigma	HGA Linear
110	5330	1920	2310	2580	2030	**1820**
120	6650	2200	3070	2520	2360	**2090**
130	7360	2480	2420	2640	2530	**2310**
140	8240	**2580**	2800	3020	2800	2970
150	9950	**2800**	3190	3130	3180	2850
Average	7506	**2396**	2758	2778	2580	2408

15.6.2 Dynamic Load Simulations

In Figure 15.8, we present the total number of established connections at different values of LQ. At the heavier load (LQ = 0.5/0.01) the HGA(s) achieve considerably higher number of connections compared to GCA. As the load becomes lighter, i.e., the release rate r_o increases, the total number of connections grows. At these conditions, all algorithms tend to achieve similar numbers of connections with HGA(s) always in favor.

In Figure 15.9, we monitor the number of active connections along the network simulation run. It is obvious that when all nodes have generated approximately 10,000 requests, the network has reached its steady state, since the number of *active connections* is stabilized. The HGA(s) achieve a considerably higher number of active connections than GCA given the same load. Considering the convergence to the steady state, the GAs appear to seize faster the available b/w (channels) and accommodate at any given instant the maximum number of requests. No safe conclusion can be derived that a specific selection scheme performs better than the others in all cases. We can only note that the linear scaling scheme achieves slightly better steady state performance, since it converges to the higher number of active connections.

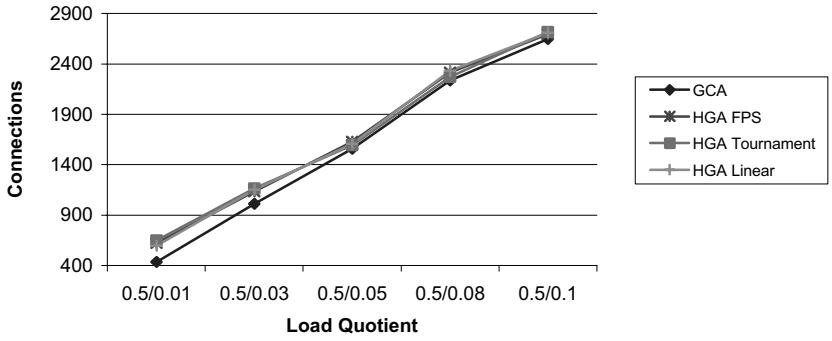

FIGURE 15.8
Established connections at dynamic load.

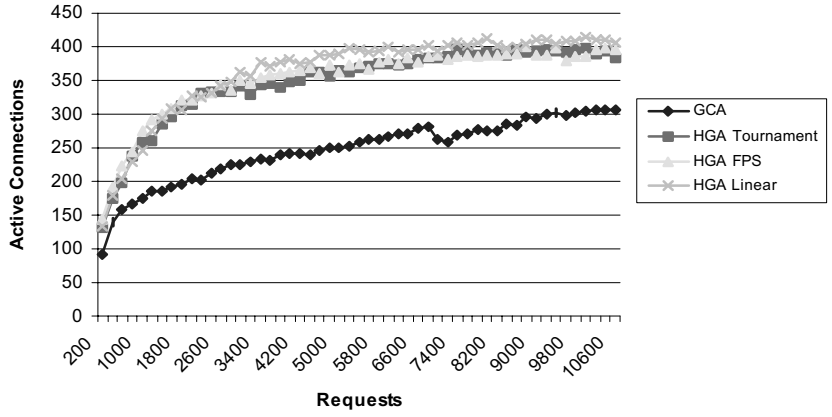

FIGURE 15.9
Active connections at dynamic load.

15.7 Concluding Remarks

Motivated by the limitation on the number of channels that can be supported and the increasing demand for giving access to a huge number of users, we have considered the use of channel sharing/reusing in multiple access networks. A hybrid genetic algorithm (HGA) has been presented for solving the NP-complete problem in-hand (ICRP) and is compared to a representative deterministic graph coloring algorithm (GCA), via simulation on a DQDB testbed network.

The results have shown that HGA is an interesting novel approach to ICRP. We concluded that HGA maintains its capability to reach optimal solutions for the ICRP at a wide range of network loads. A study of the simulation

results reveals that the GCA performance is comparable to that of HGA at light/medium load, while HGA solutions massively outperform the GCA ones at heavy network load. This happens due to the fact that deterministic algorithms are more efficient when applied in small-size problems. On the contrary, as the problem size increases, robust and general procedures are favored. Deterministic algorithms may be trapped in local optimum, in contrast to genetic algorithms. In hybrid genetic algorithms, the combination of a local optimization algorithm with mechanisms as selection, crossover, mutation, the population-wide search, the capability of moving from one solution to another significantly different, permit to scan a bigger neighborhood of solutions and therefore to move towards increasingly better optima.

The CPU time that HGA consumes in order to reach its solution is proved to be considerably lower than the time required by the GCA. This conclusion matches with the theoretically computed time complexity in terms of magnitude. The nature of genetic algorithms to intuitively search for optimal solutions against the combinational methods employed by the deterministic graph coloring algorithms, ensures considerably lower search time for the HGA especially at heavy traffic conditions.

The application of HGA can be extended to a variety of resource allocation problems at multiple access networks (e.g., satellite, mobile networks), where channel sharing/reusing can be used to enhance the channel management.

Problems

1. Implement the classical GA algorithm for the ICRP problem. Keeping constant all but one of the GA parameters to be tuned, analyze the behavior of the simulation results by varying one parameter at a time. Compare the results with those found by the HGA approach.
2. Evaluate the advantages and disadvantages of using the HGA approach for the ICRP problem.
3. Apply the HGA approach to Channel assignment problem in cellular mobile networks.
4. Compare the HGA approach with Neural Network approach and other heuristic approaches (i.e., Simulated Annealing) for the channel assignment problem in cellular mobile networks.
   Design and implement a common simulative framework (CSF) and a common testbed framework (CTF) to evaluate deterministic, HGA, Neural Networks, and heuristic approaches for channel management in mobile networks.
5. Is it possible to combine two different approaches to solve the ICRP problem? (which is NP-Complete).

References

1. N. Funabiki and S. Nishikawa, "A Gradual Neural-Network Approach for Frequency Assignment in Satellite Communication Systems," *IEEE Trans. on Neural Networks*, vol. 8, No. 6, 1359–1370, 1997.
2. M. Duque-Anton, D. Kunz, and B. Ruber, "Channel Assignment for Cellular Radio Using Simulated Annealing," *IEEE Trans. on Vehicular Technology*, vol. 42, No. 1, 14–21, 1993.
3. G. Chakraborty and Y. Hirano, "Genetic Algorithm for Broadcast Scheduling in Packet Radio Networks," in *Proc. IEEE Int. Conf. Evolutionary Computation*, 183–188, 1998.
4. S. Olariu, "An Optimal Greedy Heuristic to Color Interval Graphs," *Information Processing Letters*, vol. 37, 21–25, 1991.
5. H.G. Sandalidis, P.P. Stavroulakis, and J. Rodriguez-Tellez, "An Efficient Evolutionary Algorithm for Channel Resource Management in Cellular Mobile Systems," *IEEE Trans. on Evolutionary Computation*, vol. 2, No. 4, 125–137, 1998.
6. V. Schnecke and O. Vornberger, "Hybrid Genetic Algorithms for Constrained Placement Problems," *IEEE Trans. on Evolutionary Computation*, vol. 1, No. 4, 226–277, 1997.
7. L.D. Chou and J.C. Wu, "Bandwidth Allocation in ATM Networks Using Genetic Algorithms and Neural Networks," in *Proc. IEEE GLOBECOM'97*, 962–966, 1997.
8. J. M. Renders and S.P. Flasse, Hybrid Methods using Genetic Algorithms for Global Optimization," *IEEE Trans. Syst., Man, Cybern. B*, vol. 26, No. 2, 243–258, 1996.
9. T. Bäck and S. Khuri, "An Evolutionary Heuristic for the Maximum Independent Set Problem," in *Proc. IEEE INFOCOM'94*, 531–535, 1994.
10. R. Elbaum and M. Sidi, "Topological Design of Local-Area Networks Using Genetic Algorithms," *IEEE/ACM Trans. Networking*, vol. 4, No. 5, 766–778, 1996.
11. H. Esbensen, "Finding (Near-) Optimal Steiner Trees in Large Graphs," in *Proc. Int. Conf. on Genetic Algorithms and Their Applications*, 485–491, 1995.
12. N.-F. Huang and H.-I. Liu, "A Study of Isochronous Channel Reuse in DQDB Metropolitan Area Networks," *IEEE/ACM Trans. Networking*, vol. 6. No. 4, 475–484, 1998.
13. W. Stallings, *Local and Metropolitan Area Networks*, 4th ed., New York: Macmillan, 1993.
14. IEEE Standard*: Distributed Queue Dual Bus(DQDB) Metropolitan Area Network(MAN)*, Media Access Control and Physical Layer Protocol Documents, P802.6/D12, 1990.
15. R.M. Newman, Z.L. Budrikis, and J.L. Hullett, "The QPSX Man," *IEEE Communications Magazine*, vol. 26, 20–28, 1988.
16. M.C. Golumbic, *Algorithmic Graph Theory and Perfect Graphs*, New York: Academic Press, 1980.
17. L. Davis, *Handbook of Genetic Algorithms*, New York: Van Nostrand Reinhold, 1991.
18. G. J.E. Rawlings, *Foundation of Genetic Algorithms*, San Mateo, California: Morgan Kaufmann Publishers, 1991.
19. T. Bäck, *Evolutionary Algorithms in Theory and Practice*, New York: Oxford University Press, 1996.

20. D.E. Goldberg, *Genetic Algorithm in Search, Optimization, and Machine Learning*, Reading, Massachusetts: Addison-Wesley, 1989.
21. M. Markaki, A. Vasilakos, and I. Kassotakis, "A Hybrid Genetic Algorithm for the Provision of Isochronous Service in High Speed Networks," in *Proc. IEEE Int. Conf. Evolutionary Computation*, 133–137, 1997.
22. L. Chambers, *Practical Handbook of Genetic Algorithms Volume II*, Florida: CRC Press, 1995.
23. J.A. Miller, W.D. Potter, R.V. Gandham, and C.N. Lapena, "An Evaluation of Local Improvement Operator for Genetic Algorithms," *IEEE Trans. Syst, Man, Cybern.*, vol. 23, No. 5, 1340–1351, 1993.
24. M.R. Garey and D. S. Johnson, *Computers and Intractability*, San Francisco: W.H. Freeman and Company, 4–11, 1979.