Code Assignment for Hidden Terminal Interference Avoidance in Multihop Packet Radio Networks

Alan A. Bertossi and Maurizio A. Bonuccelli, *Member, IEEE*

Abstract— **Hidden terminal interference is caused by the (quasi-) simultaneous transmission of two stations that cannot hear each other, but are both received by the same destination station. This interference lowers the system throughput and increases the average packet delay. Some random access protocols that reduce this interference have been proposed, e.g., BTMA protocol. However, the hidden terminal interference can be totally avoided only by means of code division multiple access (CDMA) schemes. In this paper, we investigate the problem of assigning orthogonal codes to stations so as to eliminate the hidden terminal interference. Since the codes share the fixed channel capacity allocated to the network in the design stage, their number must not exceed a given bound. In this paper, we seek for assignments that minimize the number of codes used. We show that this problem is NP-complete, and thus computationally intractable, even for very restricted but very realistic network topologies. Then, we present optimal algorithms for further restricted topologies, as well as fast suboptimal centralized and distributed heuristic algorithms. The results of extensive simulation set up to derive the average performance of the proposed heuristics on realistic network topologies are presented.**

I. Introduction

MORE than two decades ago, the need to let remote computers exchange data arised. Such a need led to the design and development of the currently ubiquitous *packed switched computer networks*. The first computer network consisted of computers connected by point to point lines, which were usually telephone lines. This arrangement is not always suitable, and in certain cases even infeasible, for many applications, like close computers in local area networks, mobile computers, or computers displaced in wild areas where the telephone system is underdeveloped or not present at all. To overcome this difficulty, broadcast communication media were used, such as *busses* (only in local area networks) or *radio frequencies*.

Computers linked by radio frequencies are equipped with radio transmitters and receivers (*transceivers*) whose task is to broadcast outgoing packets and to listen for incoming packets. The arrangement *computer+transceiver* is often called *station*. In this case, the computer network is called *Packet Radio Network (PRN)*. PRN's were first displayed in 1969 at the University of Hawaii [1] and since then have greatly increased their presence and importance in the computer network scenario.

Sometimes all the stations of a PRN can directly receive each other transmissions. In this case, the network is called *single hop*, and is typical of fixed, relatively close stations, or of stations communicating by a satellite. More often this is not the case, and a packet must be received and later retransmitted by intermediate stations before reaching its final destination: the network is thus a *multihop* one.

Unconstrained transmission in broadcast media may lead to the time overlap of two or more packet receptions, called *collision* or *interference*, resulting in damaged useless packets at the destination. Collided packets must be retransmitted, thus increasing the *delay*, that is the time between the packet generation by the origin computer and its successful reception by the final destination, and the *bandwidth usage*, which in turn lowers the system throughput. There are two types of collisions: *direct (or primary) collision*, due to the transmission of stations which can hear each other, and *hidden terminal (or secondary) collision*, when stations outside the hearing range of each other transmit to the same receiving stations.

Several protocols have been devised to reduce or eliminate the collisions. Such protocols form the Medium Access Control (MAC) sublayer of the OSI model [14] and can be divided into two classes: *random access* protocols and *deterministic (code division multiple access, or CDMA, and time division multiple access, or TDMA)* ones. Under a random access protocol, a station starts the transmission whenever some locally testable conditions are met and there is a packet waiting for transmission. The most used and well-known MAC protocols fall into this class: ALOHA (no test is performed), slotted ALOHA (a time-out test is performed), CSMA (medium sensing for other transmissions), CSMA/CD (medium sensing and direct collision detection) [14], and BTMA (medium sensing for other transmissions or for a *busy tone* by the receiving station) [15]. These protocols are very simple, easy to implement, fast to run, truly distributed, and their features are well studied and understood. Unfortunalely, they only reduce the number of direct collisions, but do not eliminate all of them, which can sometimes happen. TDMA protocols, while avoiding interferences, are quite complex and time consuming since they require a slow gathering of transmission requests and a later large processing time before a packet is transmitted. To overcome these drawbacks, CDMA protocols

Manuscript received July 27, 1992; revised February 28, 1995; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor C.-L. I.

A. A. Bertossi is with the Department of Mathematics, University of Trento, 38050 Trento, Italy (e-mail: bertossi@science.unitn.it).

M. A. Bonuccelli is with the Dipartimento di Informatica, University of Pisa, 56100 Pisa, Italy (e-mail: bonucce@di.unipi.it).

IEEE Log Number 9412598.

1063–6692/95$04.00 © 1995 IEEE

have been recently introduced [8], [5], [9], [2], [10], [6]. These protocols allow a collision-free packet transmission without the time cost of TDMA protocols, but are more hardware demanding. The collision-free property is guaranteed by the use of spread spectrum communication techniques (e.g., hopping over different time slots or frequency bands, or use of direct-sequence pseudonoise) and proper assignment of *orthogonal codes*.

The only random access protocol designed to overcome the hidden terminal interferences is BTMA, which does only reduce but not eliminate such interferences. Under BTMA, when a receiving station senses a transmission bound for itself, it transmits a busy tone on another frequency. Transmitting stations test for the absence of the destination station busy tone, before broadcasting a packet. To see how the above interferences may arise, assume that the propagation delay between stations A and B is d_1, and that between stations B and C is d_2. Then, if A starts transmitting at time t, B will start receiving the A packet and broadcast the busy tone at time $t + d_1$. Station C will receive the busy tone at time $t + d_1 + d_2$. Thus, there is a time interval of length $d_1 + d_2$ during which C can transmit a packet without being aware of the A transmission, resulting in a collision at B. This drawback becomes more and more serious as $d_1 + d_2$ becomes larger and larger. So, when the PRN is widespread, or when the propagation delay is large with respect to the packet transmission time, BTMA protocol is of little help. In particular, when the propagation delay is larger than the packet transmission time, the busy tone is simply useless.

CDMA protocols require that either transmitters or receivers are able to communicate over a multitude of codes. Such codes share the fixed channel capacity allocated to the network in the design stage. Thus, their number must no exceed a given bound, and their use has to be minimized. This is done by properly assigning to the stations the minimum number of different orthogonal codes needed to eliminate collisions. When the transmitters are code-agile, namely, able to communicate over several codes, we are in presence of a *receiver-oriented code assignment (ROCA)* scheme. Alternatively, the receivers are code-agile, in which case the scheme is a *transmitter-oriented code assignment (TOCA)* one [7]. Recently, the *pairwise-oriented code assignment (POCA)* scheme has been proposed [6]. Under this scheme, both receivers and transmitters are code-agile, and codes are assigned to each single-hop receiver-transmitter pair. ROCA schemes are cheaper and simpler, but yield a lower throughput than TOCA ones. Moreover, hidden terminal interferences cannot be completely avoided by ROCA schemes, while they can be totally eliminated by properly assigning orthogonal codes in TOCA schemes [7]. POCA schemes retain the same interference avoidance properties of TOCA ones, while requiring a more expensive harware and (in some cases) a smaller number of codes, and yielding a slightly worse performance [6].

In this paper, we investigate the problem of minimizing the codes needed to eliminate hidden terminal interferences in a Packet Radio Network with TOCA MAC protocols. In [7] such a problem is introduced, some heuristic procedures for code assignment in special network topologies are proposed, and a performance evaluation of the mixed protocol is presented. However, a graph-theoretic investigation of such problem is left as a relevant open question: in this paper, we present the results of such an investigation.

Briefly, the remainder of the paper is structured in the following fashion. In Section II we give a formal statement of the problem and establish its NP-completeness, even for very special and realistic network topologies, thus proving its computational intractability. Section III presents optimal algorithms for some special network topologies, and fast suboptimal heuristic algorithms for general topologies, both centralized and distributed. The average performance of such heuristics is evaluated by means of extensive simulation experiments. Finally, conclusions terminate the paper in Section IV.

II. Problem Formulation and Computational Complexity

We begin the present section with a formal problem statement. Such a statement is then used to establish the problem complexity.

A. Problem Formulation

A PRN can be modeled as an undirected graph $G = (V, E)$, where the set of *vertices* (or *nodes*) $V = \{1, \cdots, n\}$ represents the set of stations, and the set of *edges* E the common channel property between pairs of stations. More precisely, there is a one-to-one mapping of the stations onto the vertices in V, and two vertices i and j in V are joined by an undirected edge $[i, j] \in E$ if and only if their corresponding stations can hear each other transmission. In such a case, the vertices (or, equivalently, the stations) are called *adjacent*. Thus, the graph G represents the network topology. A *path* between the vertices i and j is a sequence $i = v_1, v_2, ..., v_h = j$ of vertices such that $[v_k, v_{k+1}] \in E$ for $k = 1, 2, ..., h - 1$; its *length* is $h - 1$, namely the number of edges appearing in it. The *distance* $d_{i,j}$ between two vertices i and j of G is the length of the shortest path between i and j; it equals the minimum number of hops that a packet must undergo in a communication between stations i and j. Two vertices (stations) i and j can generate a hidden terminal interference if and only if they are two hops away, namely, when $d_{i,j} = 2$. Such an interference can be eliminated if i and j transmit on different orthogonal codes. Thus, the hidden terminal interference avoidance problem can be formulated as follows: assign codes to vertices in V so that every pair of vertices at distance two is assigned a couple of different codes and the minimum number of different codes is used.

B. Computational Complexity

By equating codes with colors, the problem can be graph-theoretically formulated as that of coloring the vertices of the graph with the minimum number of colors in such a way that *vertices at distance two are colored with different colors*.

In [6], it has been observed that this problem is identical to the ROCA code assignment one, which only avoids direct interferences.

The above problem is a variation of the classical VERTEX COLORING problem, where vertices at distance one have to be assigned different colors, which is known to be strong NP-complete even when restricted to very special classes of graphs [4]. It is easy to derive the NP-completeness result for our problem, when the graph G can assume a general topology, giving a reduction from the VERTEX COLORING problem. However, this does not establish the problem complexity for PRN topologies, which are not general graphs since they must meet some practical constraints. Therefore, we shall propose in the following a proof of the strong NP-completeness of our problem when restricted to special but very realistic network topologies.

First, let us assume that the stations are located on a plane with no (radio) obstacle. Physically, this means that the zone in which the transmission of a station can be heard is a disk. Such situation can be commonly met, e.g., when the stations are displayed in a relatively flat land. In this case, the network topology can be depicted by the station coordinates in the Euclidean plane representing the land, and their transmission ranges. The network with the above property will be called *Euclidean (Disk) Network*. We further restrict our attention to such networks in which all the transmission ranges are identical (e.g., say because the stations have equal power) and each station can be heard by at most three other stations. Let us call such subclass of networks *3-Euclidean* networks. In the following, we shall show that the problem of deciding whether three orthogonal codes are sufficient or not to eliminate the hidden terminal interference in 3-Euclidean networks is NP-complete.

In order to prove the above NP-completeness result, we need another problem Q, already known NP-complete, and we have to give a polynomial time transformation of any instance of the problem Q into a particular instance of our problem, so that any solution for Q can be quickly transformed in a solution of our problem, and vice-versa. We shall reduce the NP-complete problem 3-VERTEX COLORING of straight line planar graphs [13] to our problem.

A *planar* graph is a graph having a nonintersecting edges planar layout. A planar graph is called *straight line* whenever its edges can be represented by straight line segments. Any straight line planar graph can be represented by a diagram of nonintersecting vertical and horizontal straight line segments, corresponding, respectively, to the graph vertices and edges. Any horizontal segment, e.g., representing the edge $[i, j]$, must join the vertical segments representing its endpoint vertices, i.e., i and j, but cannot intersect any other segment. An example of a straight line planar graph, as well as its associated segment diagram, is shown in Fig. 1. Given a straight line planar graph G, its associated segment diagram can be built in polynomial time [3]. Thus, the graph and its diagram are equivalent for the NP-completeness proof, and we shall use the latter. In a segment diagram, the horizontal (vertical) distance between two vertical (horizontal) segments, if not zero, is irrelevant. Thus, we can horizontally and vertically stretch or shrink the diagram at will, without a change in the corresponding graph.

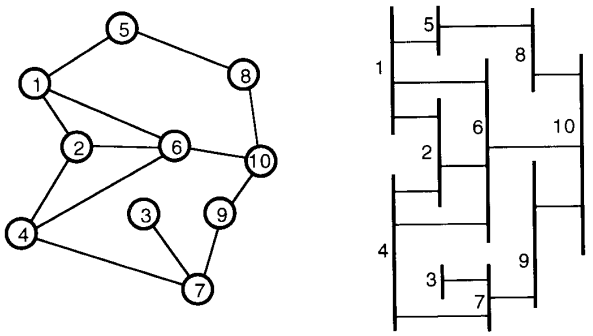

Fig. 1. A straight-line planar graph and its representation by a segment diagram.

Theorem 1: The question of deciding whether three codes are sufficient for hidden terminal avoidance in 3-Euclidean networks is NP-complete.

Proof: See the Appendix. □

The problem reduction used in the above theorem proof allows us to transfer some complexity properties from the VERTEX COLORING problem to the code assignment one, besides the NP-completeness. For instance, no polynomial time algorithm exists which is guaranteed to always produce a code assignment with no more than twice the minimum number of codes for general network topologies. However, in special cases, optimal code assignments can be found quickly, as we shall see in the next section. There, we also present fast suboptimal algorithms for general network topologies, assess their average performance, and give a distributed algorithm for the code assignment problem studied in this paper.

III. ALGORITHMS

In this section, optimal algorithms for code assignment in special networks, as well as both centralized and distributed suboptimal algorithms for general topologies, will be presented.

A. Optimal Algorithms for Special Networks

Makansi [7] has shown that the minimum number of codes needed to eliminate the hidden terminal interference cannot be smaller than the maximum number of vertices which are mutually at distance 2. This statement (a trivial one in our model, since all vertices at distance two must have different codes), allowed him to derive optimum code assignments for some special kinds of regular network topologies, such as busses, hexagonal, and grid topologies. We show here how to derive optimum code assignments for two other regular network topologies, namely, *rings* and *trees*.

Let us first consider a ring network whose number n of stations is a multiple of four, namely, for which $n = 4k$ for some integer $k \geq 1$. In such a case, an optimum assignment using only two codes can be easily derived by replicating k times the code assignment pattern shown in Fig. 2(a). An example for $n = 4k$ stations is shown in Fig 2(b).

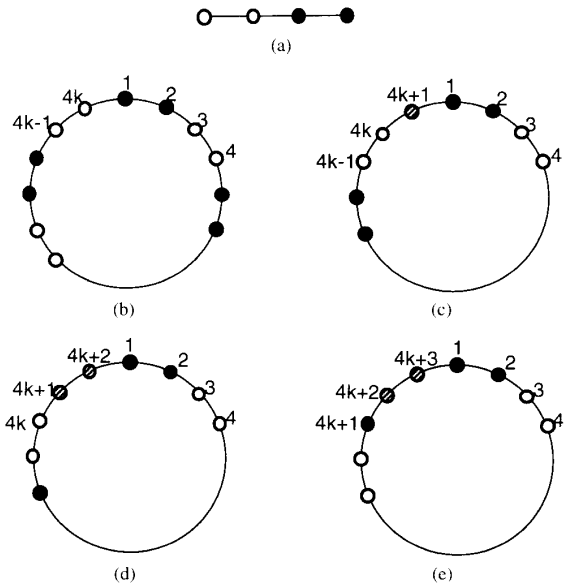

Fig. 2. Optimal code assignment for n station ring networks. (a) Basic pattern. (b) $n = 4k$. (c) $n = 4k + 1$. (d) $n = 4k + 2$. (e) $n = 4k + 3$.

When n is not a multiple of four, however, three codes are needed. As an example, consider a ring network with $n = 4k + 1$ stations. By arbitrarily choosing an origin station on the ring and replicating k times the basic code assignment pattern of Fig 2(a) to the first $4k$ stations, the situation shown in Fig. 2(c) arises, where a third code is needed for station n. A similar situation comes up when $n = 4k + 2$: as shown in Fig 2(d), the last two nodes are assigned a third code. Finally, the case $n = 4k + 3$ is similar to the previous one; as shown in Fig. 2(e), the first code can be assigned to station $4k + 1$, while the third code has to be assigned to the last two stations $4k + 2$ and $4k + 3$.

Now consider complete binary tree topologies. In such networks, any station must be assigned a code different from that of its brother, grandfather, and grandsons, since they are the only stations which are two hops away from it. Three codes suffice. Let T be a complete binary tree, and let the *level* of a station be its distance from the root, namely, the length of the shortest path between it and the root (the root is at level 0). An optimum code assignment for T can be found as follows. Assign first codes 1, 2, and 3, respectively, to the root of T, its left son, and its right son. Then consider the stations in T by increasing levels: if a station has been assigned a code C, then assign the remaining two codes to its grandsons, but giving different codes to brother grandsons. An example is shown in Fig. 3(b), where assignment of codes to grand-sons is done by using the three basic assignment patterns shown in Fig. 3(a).

It is easy to realize that the above procedure can be generalized to find optimum code assignments for $k - ary$ tree topologies, namely for trees in which every node has at most k sons. An optimum code assignment requires in such a case $k + 1$ codes (for $k = 2$, i.e., for binary trees, 3 codes are needed as seen before, while for $k = 1$, i.e., for busses, 2 codes are enough; notice that the code assignment for busses given in [7] is a special case of that given here for trees).

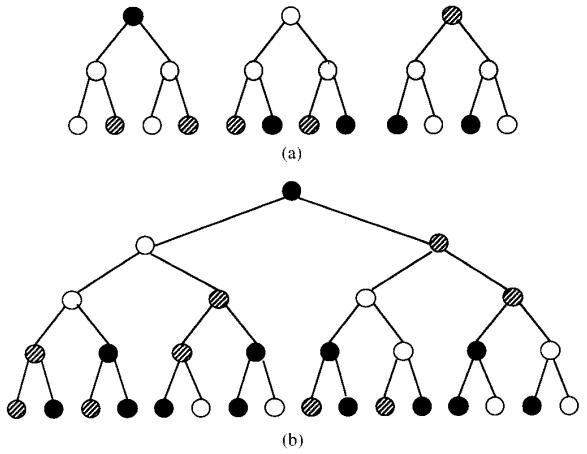

Fig. 3. Optimal code assignment for binary tree networks. (a) Three basic assignmants for a station and its grandsons. (b) An optimal assignment for a binary tree with 32 stations.

B. Centralized Heuristics for General Networks

In this section, we describe simple centralized heuristic algorithms for general network topologies. Since the problem of finding the optimal code assignment is computationally intractable, the proposed heuristics generate quickly (in polynomial time) code assignments which eliminate hidden terminal interferences but do not use the minimum number of codes. The extreme simplicity of the proposed heuristics, however, makes them attractive for actual utilization. In [6], a centralized algorithm similar to those presented here has been proposed. Our algorithm is based on a naive greedy graph coloring heuristic paradigm. The new part of it is the node ordering, and the choice of the best such ordering. Observe that there is no better practical approach.

Centralized algorithms are performed by one selected station. Such station can be selected, e.g., by using distributed election algorithms [11], and must know the network topology. This requires the exchange of many control messages, and can severely lower the system performance. However, centralized algorithms are usually faster and produce better results. So, they are a good choice when the network topology changes rarely, namely for (relatively) static networks. Besides, in a TOCA setting, the system cost of a distributed protocol implementation can be very high.

Let the n stations be named $1, 2, \ldots, n$ according to any specified criterion (e.g., by sorted station nick names, by increasing number of neighbors at distance 1, by increasing number of neighbors at distance 2, or by a random ordering). Moreover, let $H2(i)$ be the set of stations j which are at distance 2 (i.e., two hops away) from station i and such that $j < i$, for $i = 1, 2, \ldots, n$. A simple code assignment algorithm considers the stations $1, 2, \ldots, n$ sequentially, one at a time, and assigns code k to station i if it is the smallest index code not assigned to the stations in $H2(i)$. Let $code[i]$ be the code

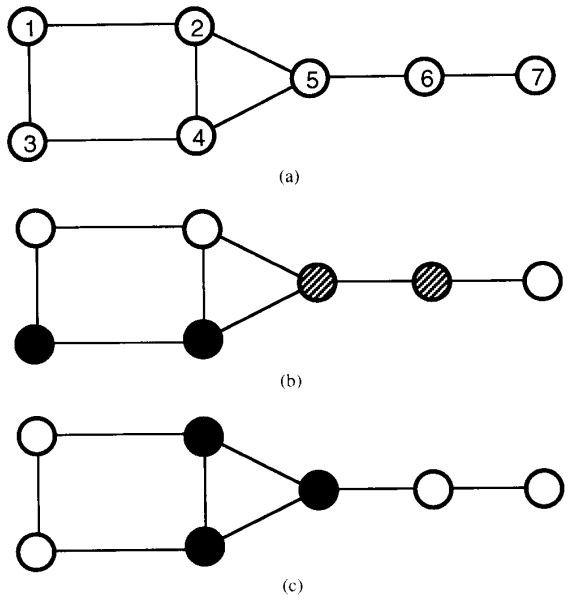

Fig. 4. Example of execution of algorithm CentralizedCodeAssignment. (a) A network with seven stations. (b) Code assignment found by the algorithm. (c) Optimal code assignment.

number assigned to node i, and let $assignedcodeset$ be the set of codes assigned to the stations in $H2(i)$. The above algorithm can be described in Pascal-like fashion as follows.

Algorithm CentralizedCodeAssignment:

```
code[1] := 1;
for i := 2 to n do
  assignedcodeset := ∅ ;
  for j := 1 to i − 1 do
    if j ∈ H2 (i) then add code [j] to
      assignedcodeset ;
  enddo;
  k := 1;
  while k ∈ assignedcodeset do
    k := k + 1;
  enddo;
  code[i] := k ;
enddo.
```

As an example, consider the seven station network depicted in Fig. 4(a). As one can easily check, the code assignment found by the above algorithm is that exhibited in Fig. 4(b). Such an assignment uses 3 codes, and is not optimal, since an assignment using only 2 codes exists, as shown in Fig. 4(c). It is worth noting that the optimal code assignment for this example would be found by the algorithm if the stations were renumbered in such a way that the numbers of stations 2 and 3 were swapped. However, this is not always true for every network. A quick algorithm inspection is sufficient to establish its correctness, namely that the produced code assignments are legal.

When the sets $H2(i)$ and *assignedcodeset* are properly implemented, e.g., by means of boolean arrays, the time

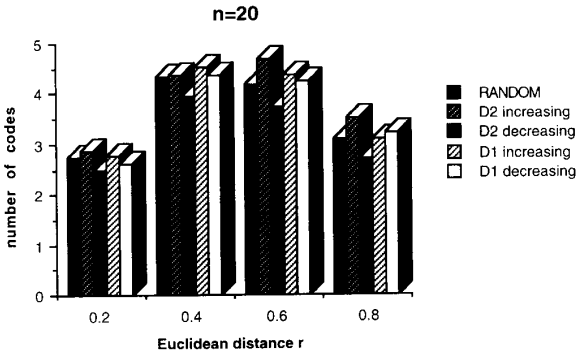

Fig. 5. $n = 20$.

required for testing membership and adding an element to the above sets are $O(1)$. Since the size of $H2$ and *assignedcodeset* are upper bounded by n, the number of stations, the overall running time of the algorithm is (On^2).

In order to evaluate the actual behavior of the above algorithm, in terms of number of codes employed, two approaches are possible: analytic and simulation. The former is more accurate but requires sophisticated mathematical tools. Unfortunately, the mathematical tools currently available for an average algorithm performance evaluation can be employed for very simple and unrealistic network topologies only, like the tandem topology. Thus, we relied on simulation experiments.

The algorithm was run on finite random networks with varying connectivity patterns and number of stations. We considered n station networks, with $n = 20$, 50, 100, and 200. The stations were represented by n randomly generated points in the unit square [0, 1] × [0, 1]. Each point was generated as a pair of random real numbers, each comprised between 0 and 1, which correspond to its Cartesian coordinates. For a given set of n points, the edges of the networks were chosen among all pairs of points whose Euclidean distance was not larger than r. We considered four values of r, namely 0.2, 0.4, 0.6, and 0.8. The $H2(i)$ sets, $i = 1, 2, \ldots, n$, were then constructed according to the network topology so generated. For each value of n and r, 500 networks were generated, and five variants of the proposed heuristic algorithms were run. Each variant differs from the others according to the chosen criterion for ordering the station numbers. Specifically, the considered ordering criteria were: random ordering ($RANDOM$), increasing number of neighbors at distance two ($D2$ *increasing*), decreasing number of neighbors at distance two ($D2$ *decreasing*), increasing number of neighbors at distance one ($D1$ *increasing*), and decreasing number of neighbors at distance one ($D1$ *decreasing*). Notice that the Random algorithm proposed here is identical to Makansi's Algorithm 1 [7], and very similar to Algorithm C1 of [6]. Figs. 5–8 report the average number of codes used by each algorithm for each pair of n and r. As the values reported in the figures show, the number of codes used by all the above criteria is quite small. In particular, the lower number of codes is always achieved by the D2 decreasing criterion, which

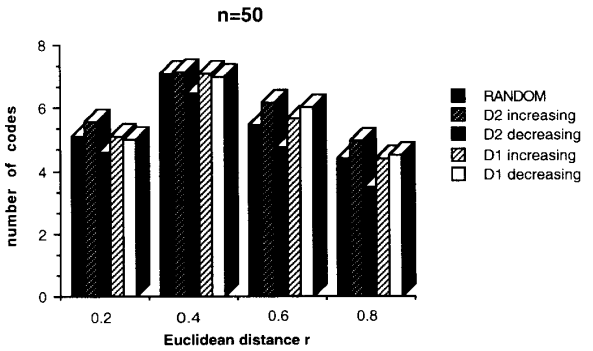

Fig. 6. $n = 50$.

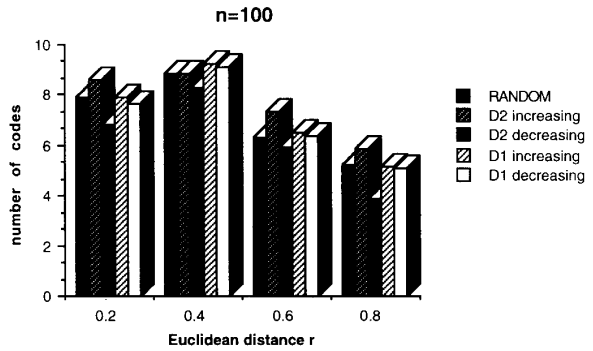

Fig. 7. $n = 100$.

improves on codes over Algorithm 1 given in [7] (named RANDOM in this paper) by 11%.

C. Distributed Heuristics for General Networks

We present here a distributed version of the above code assignment heuristics whereby all information upon the topology of the network is truly distributed through the entire network. Distributed algorithms are the natural choice for computer networks, and are particularly suitable for dynamic topologies, e.g., for mobile stations. Besides, they allow a more reliable computation since no single station failure can compromise the result, like a fault in the station performing the centralized algorithm.

Few distributed algorithms for optimization problems are known. In [6], a distributed algorithm similar to the one presented here is proposed, as well as other distributed algorithms. However, such algorithms have been devised for a different objective function, since they do not seek for minimum number of codes.

Let us assume that each station of the network only knows the names of its neighbors which are at distance at most two from it (this list of names can be obtained by obviously changing the usual topology-exchange distributed algorithms [14]). Let the stations be ordered according to the selected criterion (random, decreasing number of neighbors, etc.). This ordering can be deduced by a number (e.g., the cardinality of the neighbor set) associated to each station and not explicitly

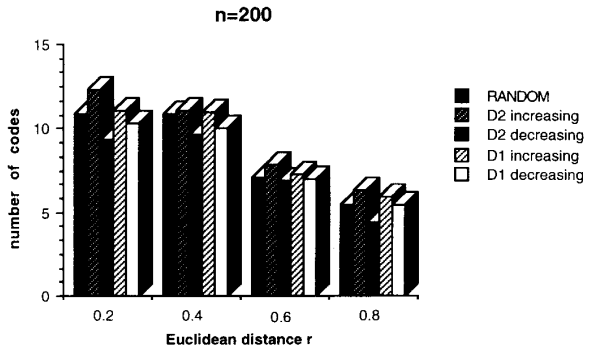

Fig. 8. $n = 200$.

transmitted. The stations are assumed to be asynchronous and can communicate by exchanging *control messages*.

There is one kind of control message: ASSIGN(i, k). This message is sent by station i to its two hops away neighbors to tell them that the code k has been $definitively$ assigned to it. All the control messages incoming into a station are buffered. When ASSIGN(j, k) comes to station i from a station h which is one hop away from i, two events may occur: if $j \in H2(i)$ (and so $j < i$), then the incoming control message is stored into the buffer; otherwise, it is either broadcast to the one hop neighbors of i, if $h = j$, or neglected, if $h \neq j$. As soon as ASSIGN(j, k) has been received from all $j \in H2(i)$, the buffer is inspected, and the smallest indexed code, say k', not in the buffer, is self-assigned to station i. Then, a message ASSIGN$(i, k$') is broadcast to i's neighbors. A Pascal-like description of such distributed algorithm for the station i is the following.

Algorithm DistributedCodeAssignment for Station i:

```
counter := |H2(i)|;
while counter > 0 do
  if ASSIGN(j,k) is received from node j
    then broadcast ASSIGN(j,k)
    else if ASSIGN(j,k) is received and j ∈ H2(i) then
           counter := counter −1;
           add k to assignedcodeset;
         endif
  endif
enddo
k := 1;
while j ∈assignedcodeset do
    k := k + 1
enddo
code[i] := k;
send ASSIGN(i,k).
```

As an example of execution of the distributed algorithm, consider again the seven station network depicted in Fig. 4(a). Initially, the seven sets $H2(i)$, $1 \leq i \leq 7$, are the following: $H2(1) = \emptyset$, $H2(2) = \emptyset$, $H2(3) = \{2\}$, $H2(4) = \{1\}$, $H2(5) = \{1, 3\}$, $H2(6) = \{2, 4\}$, and $H2(7) = \{5\}$. Here is a possible sequence of events that may occur during

the (asynchronous) execution of the distributed algorithm (broadcasting messages coming from one hop neighbors are not reported).

1) Station 2 starts execution; since $H2(2) = \emptyset$, it sets code[2] to 1, sends an ASSIGN(2,1) control message, and stops its execution;
2) Stations 4 and 5 start execution, also. Since $H2(4) = \{1\}$, and $H2(5) = \{1,3\}$, they wait for ASSIGN messages from stations 1 and 3;
3) Station 3 receives ASSIGN(2,1). Since this is the only ASSIGN message on wait , it assigns 2 to code[3], sends an ASSIGN(3,2) control message, and stops its execution;
4) Station 1 sets code[1] to 1 ($H2(1) = \emptyset$), sends ASSIGN(1,1), and terminates its execution;
5) Upon activating, Station 7 waits for a message from station 5;
6) Station 6 is the last station to activate. As $H2(6) = \{2,4\}$, it waits for two messages from stations 2 and 4;
7) Station 5 receives ASSIGN(3,2), and place it in the buffer. Once ASSIGN(1,1) is also received, code[5] is set to 3, ASSIGN(5,3) is broadcast, and the computation is halted;
8) Station 4, after receiving ASSIGN(1,1) sets its code to 2, sends ASSIGN(4,2), and stops;
9) Station 6 waits for messages. Then, it receives ASSIGN(2,1) and ASSIGN(4,2). Subsequently, it assigns 3 to code[6], sends ASSIGN(6,3), and terminates;
10) Finally, station 7 receives ASSIGN(5,3), and sets code[7] to 1, sends ASSIGN(7,1), and halts.

The code assignment found by the distributed algorithm for this example is exactly the same assignment found by its centralized counterpart (see Fig. 4(b)): stations 1, 2, and 7 are assigned code 1, stations 3 and 4 are assigned code 2, and stations 5 and 6 have code 3.

To validate the correctness of the proposed distributed code assignment algorithm, only note that:

a) Each station tries the codes for a strictly increasing sequence of values of k;
b) Each station waits for control messages coming from stations having smaller identifiers;
c) There is at least one station having an empty $H2$ set, which can thus be immediately assigned code 1;
d) At most n different codes (i.e, one different code per station) are needed in the worst possible assignment.

Conditions a)–d) together guarantee that no deadlock or livelock can occur, that the distributed algorithm converges within a finite amount of time, and that the code assignment is valid.

To evaluate the complexity of the proposed algorithm, let us count the number of control messages that are exchanged among the stations of the network in the worst case. Each time a generic station, say station i, is assigned a code, say k, it sends an ASSIGN(i,k) control message to its neighbors. Since this control message is broadcast by the one hop neighbors to the two hops neighbors of i, there are at most $O(d)$ copies of such message travelling on the network, where d is the maximum node degree in the network. Since there are n nodes, the overall number of control messages which are exchanged among the nodes of the network is upper bounded by $O(dn)$.

Finally, to evaluate the performance of the code assignment produced by the algorithm, only observe that such assignment is the same as that produced by its centralized version. Therefore, all the considerations on the number of codes which were made in Section III-B still hold true.

IV. Conclusions

In this paper, we considered the hidden terminal interference avoidance by means of Code Division Multiple Access. The problem of minimizing the number of orthogonal codes needed to eliminate the above interference was investigated. We established the problem NP-completeness even for very restricted but very realistic cases. Then, we proposed optimal algorithms for special network topologies, and presented centralized and distributed sub-optimal heuristic algorithms, together with the results of simulation experiments set up to derive the average algorithms performances.

Further work on this subject should be directed to devise new better heuristic algorithms, and to obtain mathematical tools for an analytical average performance evaluation of such heuristics.

Appendix

Proof of Theorem 1: Given a straight line planar graph $G = (V, E)$, we shall construct a 3-Euclidean Network N such that the vertices of G can be colored with three colors if and only if three orthogonal codes are sufficient to eliminate the hidden terminal interference in N. Without loss of generality, we shall consider a segment diagram for G such that the horizontal distance between any two vertical segments, and thus the length of any horizontal segment, is either zero or a multiple of 24 units, and the vertical distance between any two horizontal segments, i.e., the length of any vertical segment, is either zero or a multiple of 12 units, even if their endpoints share a common vertical segment but do not lie on the same side with respect to that vertical segment. Besides, we also assume that the segments endpoints have coordinates multiple of 12.

The technique employed in the reduction is the so called *component design* [4]. We firstly assume that the station transmission range r is greater than $\sqrt{13}$ and smaller than 4 units. We define four component types for the 3-Euclidean network N: *vertical, interior, leftmost, and rightmost*. These components are shown in Fig. 9.

The vertical and interior components are identical, but are 90° rotate. The vertices (i.e., stations) distances are depicted in the figure. Any code assignment for these components requires three codes at least (the vertices T, IL and IR need different codes, being mutually at distance 2), and in any three code assignment the vertices B and T must have the same code, since B must bear a code different from that of the vertices IL and IR, by symmetry. The leftmost component has the same code assignment features of the previous two components, since it differs from them only in vertex R, which is one unit

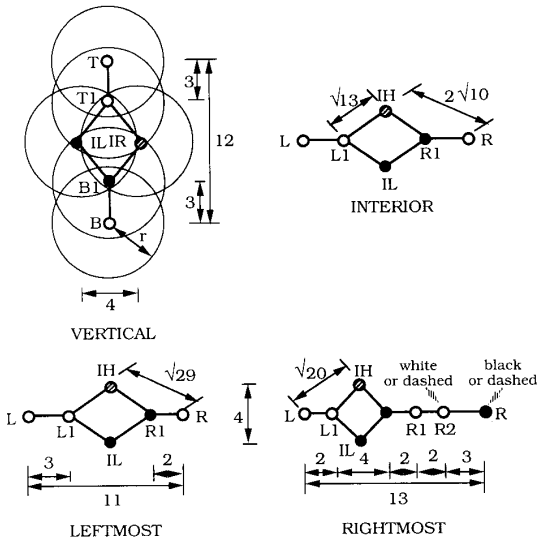

Fig. 9. The four components used in the NP-completeness proof.

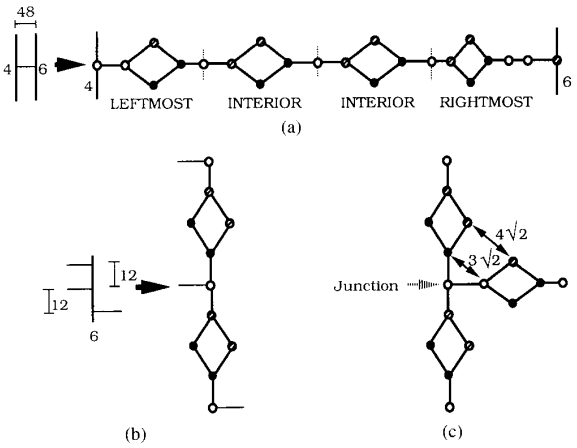

Fig. 10. Component substitutions in the segment diagram. (a) Horizontal substitution. (b) Vertical substitution. (c) Horizontal–vertical substitution.

closer to R1. This, however, does not alter the code assignment at all. Finally, the rightmost component needs three codes, also. Note that the portion of this component from vertex L to vertex R1 is isomorphic to the vertical component, with all horizontal distances reduced from 3 to 2. Such a reduction does not modify the adjacency relation between the vertices. Thus, L and R1 must be assigned the same code, and so vertex R must have a code different from that of vertices R1 and L. A three code assignment for the above components is shown in Fig. 9, where the codes are represented by black, white, and dashed vertices.

Given a segment diagram for G, we substitute each horizontal segment (whose length is $k \times 24$) with a chain consisting of one leftmost, $2(k-1)$ interior, and one rightmost components, which are connected in series, from left to right. The chain is such that the rightmost vertex of a component and the leftmost vertex of its right adjacent component are the same (see Fig. 10(a)). Besides, we substitute each vertical segment, of length $h \times 12$, with a chain of h vertical components, connected like the previous chain (see Fig. 10(b)). Note that the leftmost and the rightmost vertices of a horizontal chain must have different codes, while the top and the bottom vertices of a vertical chain must be assigned the same code.

Horizontal and vertical chains are joined at points with multiple of 12 coordinates, like the corresponding segments. Thus, in these junctions a leftmost or rightmost horizontal component vertex and a top or bottom vertical component vertex are superimposed. Such superimposed vertices are merged into a unique one. Fig. 10(c) shows such a junction. The junctions do not alter the adjacency relation between vertices since vertices in different joined components are at least $3\sqrt{2} > 4$ units apart.

We now show that the given segment diagram can be colored with 3 colors if and only if three codes are sufficient to eliminate the hidden terminal interference in the so built 3-Euclidean network N.

Let us assume that colors and codes are represented by the same symbols, that the segment diagram can be colored with 3 colors, and that such a coloring be given. Then, if a vertical segment has been colored with color c, we assign the associated code c to the top vertex in the chain corresponding to that segment, and we assign the codes to the other vertices in that chain according to the scheme given in Fig. 9. Note that all the top and bottom vertices of the components of the chain get the same code c. Similarly, horizontal chains are code assigned according to the Fig. 9 scheme. Junction vertices, which are the merging of two extreme vertices, one of a vertical chain, and the other of a horizontal chain, will get the code of their vertical image, i.e., according to the vertical chain code assignment.

We claim that the above code assignment is legal, i.e., with no hidden terminal interference. It is easy to see that the only critical code assignment is that of junction vertices, since the others are derived from these according to the schemes given in Fig. 9. Junction nodes are assigned the code associated to their vertical segment color. Any horizontal chain connects (on junction vertices) two vertical chains representing a pair of horizontally connected vertical segments. These segments have been assigned different colors, and so the corresponding junction vertices have been assigned different codes. Remembering the shape of horizontal chains, i.e., one leftmost, some interior, and one rightmost components, it is easy to see that all the extreme vertices of these components have the same code of the left junction vertex, but the right junction vertex (the rightmost vertex of the rightmost component) has a different code, which is legal.

Conversely, let us assume now that the 3-Euclidean network N has been successfully assigned three codes. Then, we color a vertical segment with the color associated with the code assigned to the junction vertices in the vertical chain corresponding to this segment. An argument similar to the above can be easily used to prove that the coloring meets the coloring constraints, i.e., that horizontally connected vertical segments have different colors.

Finally, it is a simple task to verify that the transformation given above can be carried out in polynomial time, and that our 3-Euclidean network code assignment problem belongs to NP, thus completing the proof.

REFERENCES

[1] N. Abrahmson, "The ALOHA system—Another alternative for computer communications," in *Proc. FJCC*, 1970, pp. 281–285.

[2] I. Cidon and M. Sidi, "Distributed assignment algorithms for multihop packet radio networks," *IEEE Trans. Comput.*, vol. 38, pp. 1353–1361, Oct. 1989.

[3] P. Duchet, Y. Hamidoune, M. Las Vergnas, and H. Meyniel, "Representing a planar graph by vertical lines joining different levels," *Discrete Math.*, vol. 46, pp. 319-321, Oct. 1983.

[4] M. R. Garey and D. S. Johnson, *Computers and Intractability—A Guide to the Theory of NP-Completeness*. San Francisco, CA: Freeman, 1979.

[5] B. Hajek and G. Sasaki, "Link scheduling in polynomial time," IEEE *Trans. Inform. Theory*, vol. 34, pp. 910-917, Sept. 1988.

[6] L. Hu, "Distributed code assignments for CDMA packet radio networks," in *Proc. INFOCOM'91*, Apr. 1991, pp. 1500-1509.

[7] T. Makansi, "Transmitted-oriented code assignment for multihop packet radio," *IEEE Trans. Commun.*, vol. 35, pp. 1379-1382, Dec. 1987.

[8] R. Nelson and L. Kleinrock, "Spatial TDMA: A collision-free multihop channel access protocol," *IEEE Trans. Commun.*, vol. 33, pp. 934-944, Sept. 1985.

[9] M. J. Post, A. S. Kershenbaum, and P. E. Sarachik, "Scheduling multihop CDMA networks in the presence of secondary conflicts," *Algorithmica*, vol. 4, pp. 365-394, Dec. 1989.

[10] C. G. Prohazka, "Decoupling link scheduling constraints in multihop packet radio networks," *IEEE Trans. Comput.*, vol. 38, pp. 455-458, Mar. 1989.

[11] M. Raynal, *Distributed Algorithms and Protocols*. New York: Wiley, 1988.

[12] M. Schwartz, *Telecommunications Networks: Protocols, Modeling, and Analysis*. Reading, MA: Addison-Wesley, 1987.

[13] L. Stockmeyer, "Planar 3-colorability is polynomial complete," *SIGACT News*, vol. 3, pp. 19-25, July 1973.

[14] A. S. Tanenbaum, *Computer Networks*. Englewood Cliffs, NJ: Prentice-Hall, 1989, 2nd ed.

[15] F. A. Tobagi and L. Kleinrock, "Packet switching in radio channels, part II: The hidden terminal problem in CSMA and the busy tone solution," *IEEE Trans. Commun.*, vol. 23, pp. 1417-1433, Dec. 1975.

Alan A. Bertossi was born in London, England, on January 7, 1956. He received the Laurea degree in computer science from the University of Pisa, Italy, in 1979.

Afterwards, he worked as a System Programmer and Designer. From 1983 through 1994 he was with the Department of Computer Science of the University of Pisa. Since 1995 he has been with the Department of Mathematics of the University of Trento, as a Professor of Computer Science. His main research interest is the design and analysis of algorithms for parallel and distributed systems.

Maurizio A. Bonuccelli (M'82) received the Laurea degree in computer science from the University of Pisa, Pisa, Italy in 1975.

From 1976 until 1990 he has been associated with the Department of Computer Science, University of Pisa, as a Research Associate first, and later as an Associate Professor. From 1990 until 1994 he was a Professor of Computer Science at the Università di Roma "La Sapienza." In November 1994 he returned to the Università di Pisa, as a Professor of Computer Science. During 1981 he took sabbatical leave from IBM T. J. Watson Research Center, Yorktown Heights, NY, working in the computer communications group. Later, he spent September 1993 at the International Computer Science Institute, Berkeley, CA, associated with the TENET group. His field of interest is algorithmic aspects of communication, computer networks, and parallel processing systems.