A Randomized Saturation Degree Heuristic for Channel Assignment in Cellular Radio Networks

Roberto Battiti, *Member, IEEE*, Alan Bertossi, and Daniela Cavallaro

Abstract—In this paper, we investigate the channel assignment problem, that is, the problem of assigning channels (codes) to the cells of a cellular radio network so as to avoid interference and minimize the number of channels used. The problem is formulated as a generalization of the *graph coloring problem*. We consider the *saturation degree* heuristic, first proposed as a technique for solving the *graph coloring problem,* which was already successfully used for code assignment in packet radio networks. We give a new version of this heuristic technique for cellular radio networks, called *randomized saturation degree* (RSD), based on node ordering and randomization. Furthermore, we improve the solution given by RSD by means of a local search technique. Experimental results show the effectiveness of the heuristic both in terms of solution quality and computing times.

Index Terms—Cellular network, channel assignment problem (CAP), graph coloring, heuristic, local search.

I. INTRODUCTION

IN THE last few years, the demand for services that allow communication in a mobile station setting is growing rapidly. The first cellular system, known as Advanced Mobile Phone Service (AMPS), appeared in Chicago, IL, in 1979. A cellular system was introduced in Europe in 1981 in the Scandinavian countries and was called Nordic Mobile Telephone (NMT). The channel assignment problem (CAP) is fairly well studied by researchers because it aims at optimizing the radio spectrum, a crucial resource for communication.

In a cellular mobile network, the covered area is divided into a discrete number of cells. A set of channels is assigned to each cell of the network, in order to meet the traffic demand generated in the cell. Due to the nature of radio transmissions, calls generated in a cell i may cause interference with calls generated in a cell j. We assume that the power of radio transmission (and thus the geographical distance) is the only factor that can cause interference. In addition, we assume all cells to be hexagonal, a situation corresponding to the case of a fairly flat area with no radio obstacles, such as the presence of large buildings.

The cellular network can be represented by means of a graph, and the channel assignment problem can thus be formulated as a graph coloring problem [9]. Because the graph coloring problem in its classical formulation is NP-complete [8], the CAP is also NP-complete, and therefore an optimal assignment cannot be found in reasonable (polynomial) time.

The CAP problem has been investigated by many researchers using graph theory and heuristic approaches. Many heuristics try to find a "good" ordering for the calls. This ordering, together with a given assignment criterion, usually leads to good results. Box [3] proposes a simple iterative technique based on classification of channel requirements in decreasing order of their "assignment difficulty." Hale [9] presents a wide collection of different versions of channel assignment problems in radio and television fields. Gamst [6] develops a theory on the optimal distance (in frequency terms) among adjacent channels for a homogeneous system of hexagonal cells and derives some lower bounds for a class of channel assignment problems [7]. Jordan and Schabe [10] propose some new metrics for measuring the performance of various channel assignment techniques. Tcha *et al.* [18] propose new lower bounds for the channel assignment problem, thus improving previous results given by Gamst. Recently, *simulated annealing* [4], [13] and *neural networks* [5], [12] have been used to solve the CAP problem, but they tend to require excessively long computing times. Funabiki and Takefuji [5] develop a parallel neural network algorithm. In their bidimensional parallel network model, nm processors are used to model a problem of n cells and m channels. Sivarajan and McEliece [16] present eight algorithms that result from the combinations of two cell ordering criteria, two call ordering criteria, and two different channel assignment techniques. Kim and Kim [11] propose a two-phase optimization technique based on frequency reuse patterns *(clusters)*. Using the same formulation of [16], Wang and Rushforth [19] present a local search algorithm for the coloring problem (CAP3), which is also used with an algorithm based on structure partitioning of the network, carried out by means of clusters (SPCAP). Sen [14] presents new lower and upper bounds in a particular homogeneous environment, with the same number of calls in each cell.

If one considers both the solutions found and the computing times, the best existing algorithms, at least for certain graphs, are those using local search, such as CAP3 and SPCAP. In this paper, we propose a new heuristic, called *randomized saturation degree* (RSD), which is a generalization of *saturation degree* (SD) for graph coloring [2]. It is based on node ordering and on randomization of choices. Unlike other existing heuristics, ordering and coloring are carried out simultaneously: the first nodes to be colored are those with the greatest number of colors in the neighborhood. The RSD performance is experimentally tested on the benchmark problems proposed in [5], [11], [16], and [19]. On these benchmarks, RSD often performs better than local search and provides good starting solutions for local search techniques. By combining RSD with a version of local search, obtained by giving more diversification to the CAP3 choices, we obtain the best results on most benchmark graphs.

This paper is organized as follows. In Section II, the CAP problem is formally defined and formulated as a version of the graph coloring problem. In Section III, the benchmark problems are proposed and local search is briefly recalled. In Section IV, the adaptation of the RSD heuristic to the CAP problem is proposed. Experimental results on the benchmark problems are presented in Section V, together with some modifications introduced to CAP3.

II. PROBLEM FORMULATION

A cellular network is illustrated in Fig. 1. Because interferences are caused by the power of radio transmission, if the geographical distance between two cells is larger than a fixed value, the same frequency channel can be "reused" in both cells at the same time without any interference (*cochannel cells*). As an example, suppose that channel Z is assigned to cell A (see Fig. 1). Suppose also that the interference extends up to the second ring of neighboring cells from the cell originating the call. Then, channel A cannot be reused in all gray cells shown in Fig. 1. In the same figure, the shaded cells denote the possible cochannel cells for channel A.

According to the model considered in [11], [16], and [19], we define a cellular network by means of the following five components:

1) a set of n distinct *cells*;
2) a *demand vector* $\mathbf{m} = (m_i)$, $1 \leq i \leq n$;
3) a *frequency separation matrix* or *interference matrix* $\mathbf{C} = (c_{ij})_{n \times n}$;
4) a *frequency assignment* f_{ik}, $1 \leq i \leq n$, $1 \leq k \leq m_i$, where each frequency f_{ik} is represented by a positive integer (code);
5) a set of *frequency separation constraints*

$$|f_{ik} - f_{jl}| \geq c_{ij} \qquad \forall i, k, j, l.$$

Each entry $c_{ij} \in \mathbf{C}$ represents the required frequency separation between each pair of system channels. If, for example, $c_{ij} = 0$, then no frequency separation is needed between f_{ik} and f_{jl}: cells i and j are *cochannel cells* and f_{ik} may be *reused* in cell j.

This paper considers the following frequency separation constraints.

1) *Cochannel constraint:* $c_{ij} = 1$, no frequency reuse is possible in cells i and j.
2) *Adjacent channel constraint:* $c_{ij} \geq 2$, no two adjacent channels may be assigned to cells i and j.
3) *Cosite constraint:* c_{ii} represents the required frequency separation between two channels assigned to the same cell i.

Let us observe that the required frequency separation is inversely proportional to the distance between two cells, that is, the largest entries in $\mathbf{C}$ are those lying on the diagonal.

The CAP problem consists of finding a channel assignment, i.e., the f_{ik}s, for the cellular network such that the system *bandwidth,* that is
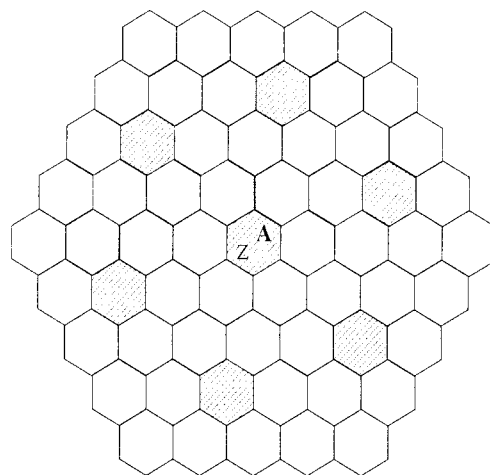
$$\max_{ik} f_{ik}$$

Fig. 1. A cellular network with interferences extending up to the second ring of neighboring cells.
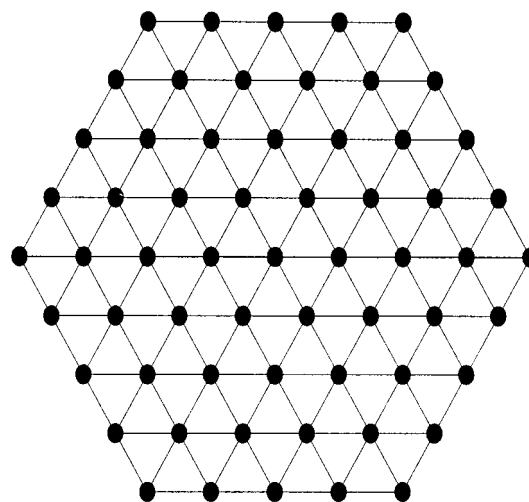

Fig. 2. A *cellular graph* modeling the network of Fig. 1.

is minimized.

Considering the above model, the CAP problem can be formulated as the generalization of a graph coloring problem. Several ways to model the CAP problem as a graph coloring problem have been proposed [14], [15], [19]. We consider the *adjacency graph* (or *cellular graph*) formulation [14], defined as follows. Each node represents one cell, and there is an edge between two nodes if the corresponding cells are adjacent in the network (i.e., they share a common cell boundary). Fig. 2 shows the cellular graph that models the network of Fig. 1.

According to this formulation, the CAP problem reduces to the problem of finding an assignment for the nodes of the cellular graph such that:

1) exactly m_i codes are assigned to each node i;
2) $|f_{ik} - f_{jl}| \geq c_{ij}$ for all i, k, j, l;
3) $\max_{ik} f_{ik}$ is minimized.

Clearly, the cellular graph coloring problem is the "generalization" of the classical graph coloring problem. More precisely, the latter problem is the CAP problem, where all entries of the demand vector are equal to one (only one code for each node) and where the interference matrix is a binary matrix (with

meaningless diagonal entries). Therefore, the CAP problem is NP-complete.

The authors of [3], [16], and [19] show different criteria for finding a good ordering of the calls to solve the CAP problem. It is clear that if the number of system calls is m, then there are $m!$ available calls orderings. Because the exchange of two calls in the same cell does not change the system bandwidth (calls in a cell are undistinguishable), all orderings can be partitioned into $m!/(m_1!m_2!\cdots m_n!)$ distinguishable possibilities.

The assignment strategy followed in this paper consists of assigning to the ith call the smallest "legal" code, that is, the smallest code that meets all frequency separation constraints imposed by the $i-1$ previously assigned calls. This assignment strategy is known as the *frequency exhaustive strategy* [16].

III. BENCHMARK INSTANCES

In this section, the benchmark problems are introduced and the local search technique that achieves the best results on the considered benchmarks is summarized.

A. Details of Ten CAP Benchmarks

We consider ten benchmark CAP instances, taken from [5], [16], and [19], which will be denoted as $A1, \ldots, A10$. Problems $A1, \ldots, A9$ are all formulated on the 21-cell system of Fig. 3. Two channel requirements for these problems are defined in Fig. 4 (Cases 1 and 2, respectively). Fig. 5 shows the problem specifications for $A1, \ldots, A9$.

$A10$ is formulated on a 25-cell system, whose frequency separation matrix and demand vector are shown in Fig. 6.

Note that all $\mathbf{C}$ entries in Fig. 6 are $c_{ii} = 2$ for all i, and $c_{ij} = 0$ or 1 for all $i \neq j$.

Problems $A1, A2, A3, A4, A6, A8, A9$ are taken from [16], while $A5$ and $A7$ come from [5]. Problem $A10$ is a practical assignment problem from Helsinki, Finland [12].

Fig. 7 reports the best bandwidths for the above ten benchmarks, obtained by the heuristic algorithms proposed in [3], [5], [12], [16], and [19]. The first line of Fig. 7 shows the lower bounds obtained by Gamst [7]. The other lines show the bandwidths obtained by the cited channel assignment algorithms. By observing Fig. 7, it is clear that the best results are those obtained by the local search algorithm CAP3 [19].

B. Larger Benchmark Instances

We consider also benchmark instances from [11], involving cellular networks larger than those considered in Section III-A. The instances are formulated on the 7×7 network shown in Fig. 8 and are denoted as $K1, \ldots, K9$.

The interferences extend up to the second ring of neighboring cells. The frequency separation between each pair of non-cochannel cells is 1 (absence of adjacent constraint), 2 or 3, and 3 or 4. The demand vector is generated by means of a distribution function $U(X, Y)$, uniform over the interval $[X, Y]$. More precisely, the demand vector entries are generated by the uniform distribution $U(10, 15)$ for $K1$, $K2$ and $K3$; by $U(10, 20)$ for $K4$, $K5$ and $K6$; and by $U(10, 30)$ for $K7$, $K8$ and $K9$.
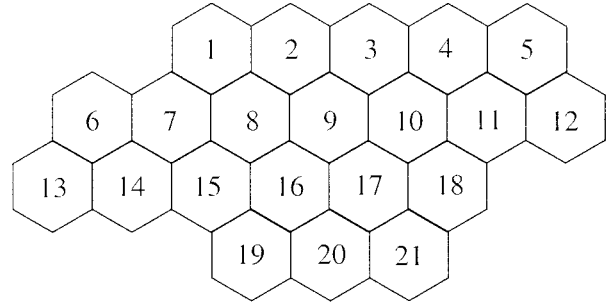

Fig. 3.   A 21-cell system (cell number is inside the cell).

Fig. 9 reports the bandwidths obtained by the heuristic algorithms proposed in [11], [16], and [19]. By observing Fig. 9, one notes that the best results are those obtained by the SPCAP algorithm [19], which is based on a partitioning of the network into a "minimum" network and a "difference" network, and then on separately coloring the two networks by means of a local search technique.

C. Local Search (LS) Algorithms

Until now, the best bandwidths for the previously presented benchmarks were achieved by local search algorithms.

Local search is based on a search space $\mathcal{R}$ and an objective function $\mathcal{F}$. For each point $x_p \in \mathcal{R}$, a set of neighbors $N(x_p) \subset \mathcal{R}$ is defined. A local search algorithm "explores" $N(x_p)$ by looking for an $x_{p+1} \in N(x_p)$ that is "better" than x_p, that is, an x_{p+1} that improves the value of $\mathcal{F}$.

The CAP problem is solved by a local search technique in [19]. Let us consider a cellular network with n cells, a demand vector $\mathbf{m}$, and an interference matrix $\mathbf{C}$. Then $\mathcal{R}$, $\mathcal{F}$, $x_p \in \mathcal{R}$, and $N(x_p)$ can be defined as follows.

1) $\mathcal{R}$ is the set of all possible ordered lists of calls, which fulfill the interference constraints.
2) $\mathcal{F}$ is the objective function to be minimized, that is, the system bandwidth.
3) x_p is the current solution, namely, an ordered list of calls.
4) $N(x_p)$ is the neighborhood of x_p, defined as

$$N(x_p) = \{x_p' | H(x_p, x_p') \leq d\}$$

where $H(x_p, x_p')$ is the number of components in which x_p and x_p' differ. As in [19], we consider $d = 2$.

A local search algorithm for the CAP problem tries to find an $x_p' \in N(x_p)$, which decreases the system bandwidth [i.e., with $\mathcal{F}(x_p') < \mathcal{F}(x_p)$]. If this configuration exists, then x_p' becomes the new current solution (x_{p+1}) and the search is iterated; otherwise, x_p is a local optimum in $N(x_p)$ and the search ends.

The local-search based CAP3 algorithm, introduced in [19], consists of three phases: an initialization phase and two search phases that explore the configurations in $N(x_p)$. Given x_p, two calls are chosen: the call a_{ik} with maximum frequency and a random call a_{jl} $(j = 1, \ldots, i-1, i+1, \ldots, n)$. $x_p' \in N(x_p)$ is the ordered list of calls obtained from x_p by swapping a_{ik} and a_{jl}. The two search phases differ in the way they accept x_p' as the new current solution. The search within $N(x_p)$ is stopped

Fig. 4. Channel requirements for problems $A1, \ldots, A9$.

	A1	A2	A3	A4	A5	A6	A7	A8	A9
c_{ii}	7	7	5	5	7	7	5	5	12
c_{ij} ($d_{ij} = 1$)	1	2	1	2	1	2	1	2	2
c_{ij} ($d_{ij} = 2$)	1	1	1	1	1	1	1	1	1
c_{ij} ($d_{ij} = 3$)	0	0	0	0	0	0	0	0	1
m	Case 1	Case 1	Case 1	Case 1	Case 2	Case 2	Case 2	Case 2	Case 2

Fig. 5. Specification details for $A1, \ldots, A9$.

Fig. 6. $A10$: frequency separation matrix (**C**) and demand vector (**m**).

as soon as an x'_p "better" than x_p is found. More precisely, we have the following.

1) *First search phase (greedy local search):* x'_p becomes the new current solution (x_{p+1}) if $\mathcal{F}(x'_p) < \mathcal{F}(x_p)$ and the search is iterated.
2) *Second search phase (monotonic local search):* x'_p becomes the new current solution (x_{p+1}) if $\mathcal{F}(x'_p) \leq \mathcal{F}(x_p)$ and the search is iterated.

Each phase terminates either if the lower bound $\mathcal{F}_{\min}$ is reached or if the number of iterations exceeds the bound $limit = m_{\min} * (n-1)$, where $m_{\min}$ is the smallest entry of the demand vector.

IV. THE RANDOMIZED SATURATION DEGREE (RSD) HEURISTIC

We now present a new CAP heuristic, which is a generalization of a technique proposed by Brelasz [2] for graph coloring. The technique was used in [2] for choosing the "branching" node in a *branch and bound* algorithm, DSATUR, and is called

saturation degree. SD has been successfully applied by Battiti *et al.* [1] to solve the channel assignment problem for packet radio networks.

The basic idea of the SD heuristic is to color first the node having the largest number of colors already assigned to the neighbors. In the case of ties, the node with the largest number of colored neighbors is colored. The idea is that these nodes have a more limited choice for choosing colors, and thus a higher risk that all available colors will be assigned to the neighbors in future steps.

The coloring technique just mentioned is somewhat restrictive: if there are many ties, the first node encountered is colored first. We introduce more flexibility and *fairness* through the use of randomization. If there is a tie between two or more nodes, the winning nodes are inserted into a set of *candidates*. The node v^* to be colored is then chosen randomly among the nodes in the set of candidates. By iterating this randomization technique (with different random number sequences), many legal colorings of the same graph can be found. Obviously, the assignment that achieves the smallest bandwidth is then chosen.

To adapt SD to cellular networks, some additional changes have been introduced. First, each node i has to be assigned exactly m_i codes. Second, codes (colors) must satisfy the constraints imposed by the interference matrix (*cosite constraint, adjacent channel constraint,* and *cochannel constraint*).

To assign many colors to the same node, two ways can be followed. The first is to order the nodes of the cellular graph (i.e., the cells of the network) and, once v^* is chosen, to assign it m_{v^*} codes. The second is to assign to v^* only one code, and then to continue considering it as uncolored until m_{v^*} codes have been assigned to it.

It is worth observing that in this heuristic, ordering and coloring of the calls are carried out simultaneously. The RSD heuristic, which follows the first of the two techniques described above, is shown in Fig. 10.

	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10
Lower Bound [7]	533	533	381	414	309	309	221	229	529	73
CAP3 [19]	533	533	381	433	309	309	221	263	529	73
Sivarajan [16]	533	533	381	447	-	310	-	270	529	-
Funabiki [5]	533	533	381	-	309	309	221	-	-	73
Box [3]	-	-	-	445	-	-	-	-	-	-
Kunz [12]	-	-	-	-	-	-	-	-	-	73

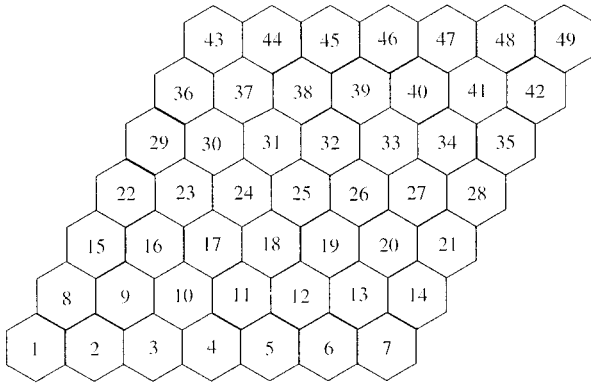Fig. 7.   Previous results for $A1, \ldots, A10$.

Fig. 8.   A 49-cell system (each cell number is inside the cell itself).

$ToBeAssigned$ is the set of uncolored nodes. A set $Neigh\ Codes\ [i]$ is associated to each node i, which contains the codes assigned to the neighbors that are at most k far from i, where k is the greatest distance for which an interference occurs. $NAssigned\ Neighbors\ [i]$ is the number of neighbors of i, up to distance k, which have already been assigned a code. This means that v^* is chosen among those nodes having the greatest number of codes already assigned in the neighborhood (up to distance k), lines 11–21. $Neigh\ (v^*)[h]$ is defined as the set of all nodes whose distance from node v^* is h.

At the beginning, $ToBeAssigned$ contains all nodes (line 1). $NAssigned\ Neighbors\ [i]$ and $Neigh\ Codes\ [i]$, for each i, are initialized to zero and the empty set, respectively (lines 2–6). At each step, a node v^* is chosen (lines 9–21). If there are ties (same coloring priority), the nodes are stored in a set of $Candidates$ (lines 12–20), and then a random node is chosen (line 21).

Successively, the node v^* is colored (line 22). Procedure ASSIGN-CODES assigns m_{v^*} colors to v^*, according to the constraints imposed by all nodes that are at most k edges far from v^* and according to the *frequency exhaustive strategy*. $ToBeAssigned$ is updated at line 23. Then $NAssigned\ Neighbors\ [i]$ and $Neigh\ Codes\ [i]$ are updated, for each node $j \in Neigh(v^*)[h]$ not yet colored, for each h, $1 \le h \le k$ (lines 24–30).

Procedure ASSIGN-CODES (lines 2–8 of Fig. 11) deletes from $Avail\_Codes$ all colors that cannot be assigned to v^* because they do not meet the channel constraints imposed by codes assigned to the neighbors of v^*. In the second part (lines 9–21),

v^* is assigned m_{v^*} codes, that is, one code to each call, according to the entries of the interference matrix.

A. Time Complexity

The initialization is carried out in $O(n)$ time (lines 2–6). Then a *while* loop begins (lines 7–31), which selects one node per iteration and is therefore repeated n times. A *for* loop, lines 11–20, carried out for each node in $ToBeAssigned$, is repeated $O(n)$ times in the worst case, so that these two nested loops together require $O(n^2)$ time.

Procedure ASSIGN-CODES assigns one code to each call in cell v^*. By implementing $Avail\_Codes$ with binary strings, $O(1)$ time is taken to delete one entry, while searching for the minimum (function MIN) requires also constant time because it depends on the constant MAX_CODE. The cardinality of $Neigh(v^*)[j]$ is $O(k)$. Therefore, the *for* loop in lines 2–8 costs $O(k^2 m_{max})$, while the *for* loop in lines 11–21 costs $O(m_{max})$, because the *while* loop (lines 14–18) is repeated for a fixed number of iterations.

Consequently, the *for* loop of RANDOMIZED-SATURATION-DEGREE (lines 24–30) requires $O(k^2 m_{max})$ time. Because this *for* loop is nested within the *while* loop of lines 7–31, the time complexity of the heuristic is $O(\max(n^2, nk^2 m_{max}))$, where k in the considered benchmarks is a small integer (two or three).

V. NEW EXPERIMENTAL RESULTS

In this section, the new RSD heuristic is tested on the benchmark problems previously introduced. RSD has been implemented in C++ and executed on an AlphaServer 2100.

A. First Results for $A1, \ldots, A10$

The execution of RSD obtains optimal solutions for eight out of the ten problems described in Section III-A. For two "harder" problems, neither the lower bound nor the CAP3 upper bound was reached. The results are shown in Fig. 12 (third line).

Problems $A1$, $A3$, $A5$, $A7$, $A9$, $A10$ are perhaps the easiest problems, and their lower bound is reached many times during 100 iterations. For the $A2$ problem, 210 iterations are necessary to reach its lower bound. For $A6$, the optimum is reached after 670 iterations. In contrast, the optimum is never reached for problems $A4$ and $A8$, even increasing the number of iterations. Problems $A2$, $A4$, $A6$, and $A8$ are the most difficult ones

	K1	K2	K3	K4	K5	K6	K7	K8	K9
c_{ij}	1	2,3	3,4	1	2,3	3,4	1	2,3	3,4
c_{ii}	3	5	7	3	5	7	3	5	7
m	U(10,15)			U(10,20)			U(10,30)		
SPCAP [19]	96	241	337	121	331	415	166	455	604
Kim [11]	98	279	368	127	366	488	173	546	672
CRF [16]	127	437	583	168	470	624	234	769	837
CRR ″	127	319	457	168	381	520	234	610	705
CCF ″	110	325	445	143	393	545	194	580	724
CCR ″	156	332	437	176	432	556	255	594	702
DRF ″	134	400	531	161	495	635	236	764	846
DRR ″	134	303	448	161	384	540	236	586	692
DCF ″	110	318	445	133	391	548	180	560	727
DCR ″	134	313	440	193	391	558	237	582	704

Fig. 9. Previous results for $K1, \ldots, K9$.

```
RANDOMIZED-SATURATION-DEGREE
1          ToBeAssigned ← {1, 2, ..., n}
2          for i ← 1 to n do
3            begin
4                NAssignedNeighbors[i] ← 0
5                NeighCodes[i] ← ∅
6            end
7          while ToBeAssigned ≠ ∅ do
8          begin
9                MaxNeighCodes ← −1
10               MaxAssignedNeighbors ← −1
11               for each i ∈ ToBeAssigned do
12                   if |NeighCodes[i]| > MaxNeighCodes then
13                   begin
14                       MaxNeighCodes ← |NeighCodes[i]|
15                       MaxAssignedNeighbors ← NAssignedNeighbors[i]
16                       Candidates ← {i}
17                   end
18                   else if |NeighCodes[i]| = MaxNeighCodes then
19                       if NAssignedNeighbors[i] ≥ MaxAssignedNeighbors then
20                           Candidates ← Candidates ∪ {i}
21               v* ← random(Candidates)
22               ASSIGN-CODES(m_{v*}, v*)
23               ToBeAssigned ← ToBeAssigned \ {v*}
24               for h ← 1 to k do
25                   for each j ∈ Neigh(v*)[h] ∩ ToBeAssigned do
26                   begin
27                       for s ← 1 to m_{v*} do
28                           NeighCodes[j] ← NeighCodes[j] ∪ {f_{v* s}}
29                           NAssignedNeighbors[j] ← NAssignedNeighbors[j] + m_{v*}
30                   end
31         end
```

Fig. 10. Randomized saturation degree.

because they also take into account the adjacency constraint ($c_{ij} = 2$ if $d_{ij} = 1$), which is a restrictive constraint to be satisfied; thus the optimal assignment is harder to be found. The computing times, however, are always less than 1 s, for all problems. The CAP3 algorithm requires about 1 second for all problems, except for $A4$ and $A8$, which take 110–170 s [19]. These values have been calculated using the UNIX `time` function. To fairly compare computing times, the SPECInt95 of our machine and of that used in [19] are considered to normalize the two different machine speeds.

```
ASSIGN-CODES ( m_v* , v* )
1    Avail_Codes ← {1.2, ... ,MAX_CODE}
2    for j ← 1 to k do
3        for each pos ∈ Neigh(v*)[j] do
4            if (pos ∉ ToBeAssigned) then
6                for s ← 1 to m_pos do
                 /* delete all codes that cannot be assigned */
7                    for each h such that | f_pos,s − h | < c_v*pos do
8                        Avail_Codes ← Avail_Codes \ {h}
9    f_v*,1 ← MIN(Avail_Codes)
10   Avail_Codes ← Avail_Codes \ {f_v*,1}
11   for r ← 1 to (m_v*-1) do
12   begin
13       p ← MIN(Avail_Codes)
         /* search for the smallest elegible code */
14       while (| p − f_v*,r | < c_v*v*) do
15       begin
16           Avail_Codes ← Avail_Codes \ {p}
17           p ← MIN(Avail_Codes)
18       end
19       f_v*,r+1 ← p
20       Avail_Codes ← Avail_Codes \ {p}
21   end
```

Fig. 11. Procedure ASSIGN-CODES.

B. RSD Plus Local Search

To improve the solutions found for $A4$ and $A8$, a local search technique can be combined with the RSD heuristic. Starting from the ordered list of calls given by RSD, the idea is to search for an ordered list of calls that leads to a smaller bandwidth by introducing local search (LS). To do this, the local search CAP3 algorithm, described in Section III-C, can be modified as follows.

1) Initialization: The starting calls list (x_0) is that given by RSD.

2) Search: In the search phase, we observed that the criterion (given in Section III-C), which selects the two calls (a_{ik} and a_{jl}) to be swapped, induces a negligible diversification for the found solutions. Indeed, CAP3 selects the call a_{ik}, which is assigned the maximum frequency channel in the current solution x_p. It is clear that a_{ik} will remain the same until a new current configuration x_{p+1} will be found. This choice forces the algorithm to explore a "small" subset of $N(x_p)$. Thus a *better* configuration cannot be found for many steps, until a random call a_{jl} is found that leads to a better solution when it is swapped with a_{ik}.

To introduce more diversification, the search was modified by changing the criterion for choosing the calls to be swapped. The a_{jl} call is still chosen in a random way, while the a_{ik} call is that to which the maximum frequency is assigned in the coloring associated to the last ordering visited $N(x_p)$. A new neighbor $x'_p \in N(x_p)$ is obtained from x_p by swapping a_{ik} and a_{jl}. In this way, the search phase has more diversification in its choices because it "exploits" the neighbors x_p while they are visited.

3) Further Results for $A4$ and $A8$: The last line in Fig. 8 shows the bandwidths obtained by executing the local search technique described above, starting from the solutions found by RSD. For $A4$ and $A8$, not only is the bandwidth given by RSD improved but also that given by CAP3 is considerably lowered in much shorter computing times.

C. Results for $K1, \ldots, K9$

Because the demand vector for the CAP benchmarks $K1, \ldots, K9$ introduced in Section III-B is not fixed, RSD was tested on the cellular network of Fig. 8 by considering different values drawn from proper uniform distributions. The uniform distribution function U was implemented by means of the UNIX `drand48()` function.

The bandwidths obtained by RSD are illustrated in Fig. 13. For each problem, the best solution obtained is shown together with the average over 30 iterations. The four mentioned criteria consist of four different ways of breaking ties during the assignment phase.

Criterion 1)   The node (cell) with the greatest number of neighboring colors is chosen and codes are assigned to *all its calls*. Ties among nodes are broken choosing the node with the greatest number of colored neighbors.

Criterion 2)   Like Criterion 1), but ties among nodes are broken preferring that node with the greatest number of calls (greatest demand vector entry).

Criterion 3)   The node (cell) with the greatest number of neighboring colors, including colors inside the node itself, is chosen and is assigned *only one color*. Ties among calls are broken choosing the call with the greatest number of *colored calls* that could interfere with it.

Criterion 4)   Like Criterion 3), but ties among calls are broken coloring that call that belongs to the cell with the greatest number of *not yet colored calls*.

By observing Fig. 13, one notes that the results of the SPCAP algorithm [19] are improved for six out of nine problems. No tie-breaking criterion, however, is significantly better that the other ones on the considered benchmarks. Criteria 3) and 4) are the slowest to execute because calls have to be colored one at each step, and they take on the average 2 min for computing 30 iterations for each problem. Criteria 1) and 2) are faster to execute because they need only 1 min to compute 30 iterations. In contrast, the SPCAP algorithm takes 70–100 s to execute only one iteration.

The local search, as described in Section V-B, was also applied to problems $K1, \ldots, K9$. The resulting bandwidths, shown in Fig. 14, are better than those of Fig. 13 for problems $K5, K8, K9$. For $K4$ and $K7$, all four criteria find the same results; this suggests that these bandwidths could be optimal. SPCAP still gives better results for only three problems: $K2, K3, K6$.

The computing times are slightly longer than before because local search is introduced: for Criteria 3) and 4), 15 min are needed to execute 30 iterations for each problem, while Criteria 1) and 2) require 7 min.

	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10
Lower Bound	533	533	381	414	309	309	221	229	529	73
CAP3	533	533	381	433	309	309	221	263	529	73
	<1s	<1s	<1s	110/170s	<1s	<1s	<1s	110/170s	<1s	<1s
RSD	533	533	381	463	309	309	221	275	529	73
	<1s	<1s	<1s	<1s	<1s	<1s	<1s	<1s	<1s	<1s
RSD + LS	533	533	381	(427)	309	309	221	(254)	529	73
				<30 s				<35 s		

Fig. 12. Performance of RSD for $A1, \ldots, A10$.

		K1	K2	K3	K4	K5	K6	K7	K8	K9
SPCAP		96	(241)	(337)	121	331	(415)	166	455	604
RSD criterium 1	best	91	255	356	(108)	(306)	436	148	421	(576)
	average	97.16	269.03	376.1	120.47	340	474.7	169.07	479.4	670.27
RSD criterium 2	best	91	255	355	109	309	432	150	423	584
	average	98.1	269.07	376.2	121.33	340.4	475.8	169.23	481.7	674.06
RSD criterium 3	best	(90)	264	361	109	313	438	149	418	587
	average	95.8	275.367	383.2	120.43	340.03	473.43	168.7	472.96	658
RSD criterium 4	best	91	262	362	111	312	435	(146)	(411)	579
	average	96.03	274.83	382.03	120.6	339.36	473.4	168.5	469.7	655.23

Fig. 13. Performance of RSD for $K1, \ldots, K9$.

		K1	K2	K3	K4	K5	K6	K7	K8	K9
SPCAP		96	(241)	(337)	121	331	(415)	166	455	604
RSD + LS criterium 1	best	(89)	249	352	(108)	(299)	419	(145)	405	563
	average	95.03	265.43	371.67	118.67	332.2	463.83	165.13	466.8	649.86
RSD + LS criterium 2	best	(89)	250	351	(108)	301	424	(145)	408	566
	average	95.2	265.4	371.83	118.6	332.77	466.86	165.33	466.2	651.9
RSD + LS criterium 3	best	90	255	351	(108)	301	421	(145)	405	572
	average	94.93	268.6	373.66	118.7	330.1	460.53	165.23	455.43	635.76
RSD + LS criterium 4	best	(89)	254	353	(108)	302	416	(145)	(392)	(558)
	average	95	267.36	373.73	118.7	329.7	458.83	165.2	454.06	634.13

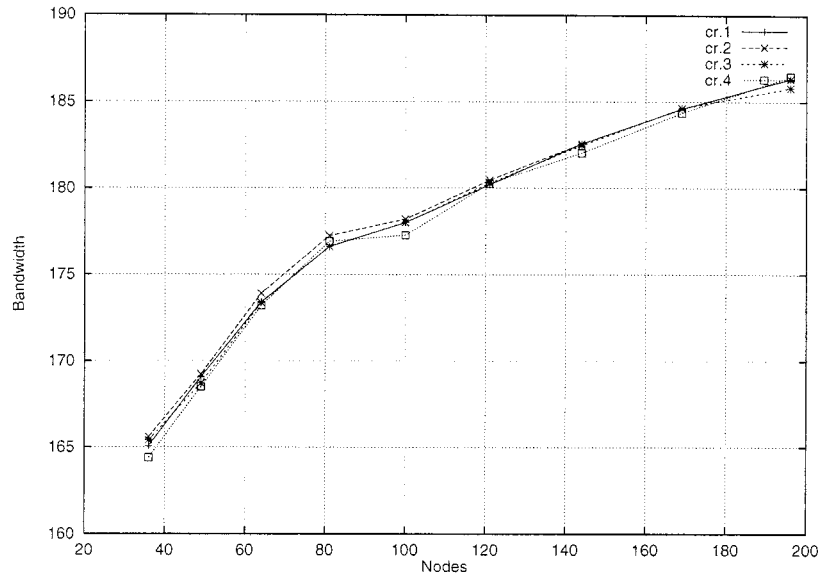Fig. 14. Performance of RSD and local search for $K1, \ldots, K9$.

Fig. 15.   Bandwidths for the four criteria of RSD. The channel constraints are $c_{ii} = 3$ and $c_{ij} = 1$ for $d_{ij} = 1, 2$.

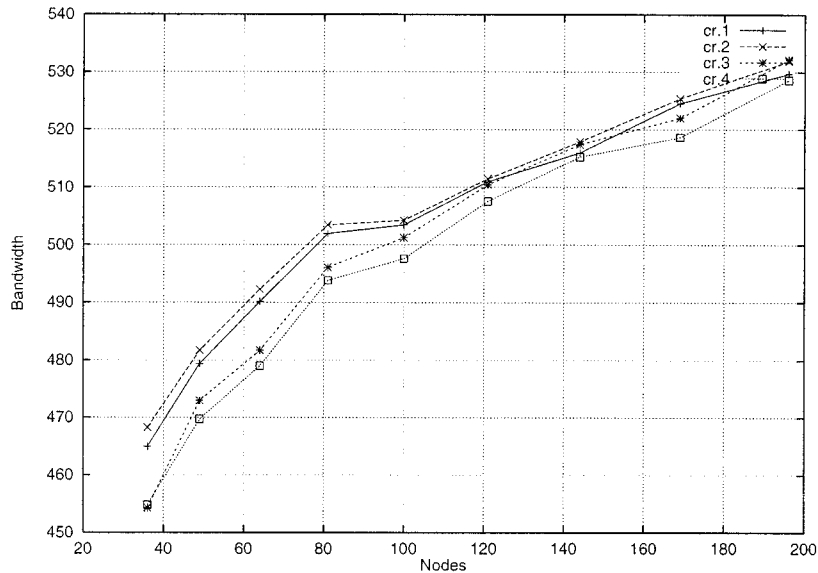
Fig. 16.   Bandwidths for the four criteria of RSD. The channel constraints are $c_{ii} = 5$, $c_{ij} = 3$ for $d_{ij} = 1$ and $c_{ij} = 2$ for $d_{ij} = 2$.

D. Performance of the Four RSD Criteria

As we can see from Figs. 13 and 11, there is not a criterion that performs better than the others. To appreciate the difference among the four criteria, we shall consider more CAP instances.

Let U be a uniform distribution over the interval [10, 30]; and let the cellular network be a system where the interferences extend up to the second ring of neighboring cells. Consider a network of $n \times n$ cells, $n = 6, \ldots, 13$. Fig. 15 represents the performance of the four RSD criteria on the networks described above, where the channel constraints for frequency separation are $c_{ii} = 3$ and $c_{ij} = 1$ for $d_{ij} = 1, 2$. The results shown are the average among 30 graph instances. It is clear that there is no best performing criterion yet.

Figs. 16–18 represent the results obtained for the same networks when the channel constraints become $c_{ii} = 5$, $c_{ij} = 3$ for $d_{ij} = 1$ and $c_{ij} = 2$ for $d_{ij} = 2$ (Fig. 16); $c_{ii} = 7$, $c_{ij} = 4$ for $d_{ij} = 1$ and $c_{ij} = 3$ for $d_{ij} = 2$ (Fig. 17); and $c_{ii} = 9$, $c_{ij} = 5$ for $d_{ij} = 1$ and $c_{ij} = 4$ for $d_{ij} = 2$ (Fig. 18).

Two different behaviors of the system bandwidth can be distinguished. A gap results between the bandwidths found by Criteria 1) and 2) and those given by Criteria 3) and 4). Indeed, when the channel constraints are not very restrictive (i.e., see Fig. 15), all criteria perform in the same way, while as the constraints become more and more restrictive Criteria 3) and 4) are better than Criteria 1) and 2).

Note that the four criteria tend to behave in the same way if the channel interference extends up to the third, fourth, etc., ring of neighboring cells. More precisely, in these cases, the difference among the four criteria is bigger when the number of cells increases: for instance, if the interference extends up to the third ring of neighboring cells, the gap between Criteria
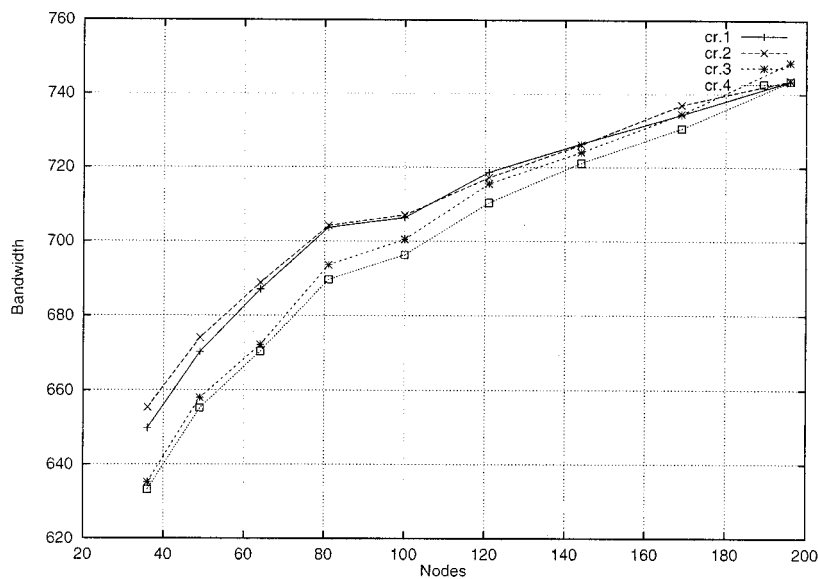
Fig. 17. Bandwidths for the four criteria of RSD. The channel constraints are $c_{ii} = 7$, $c_{ij} = 4$ for $d_{ij} = 1$ and $c_{ij} = 3$ for $d_{ij} = 2$.
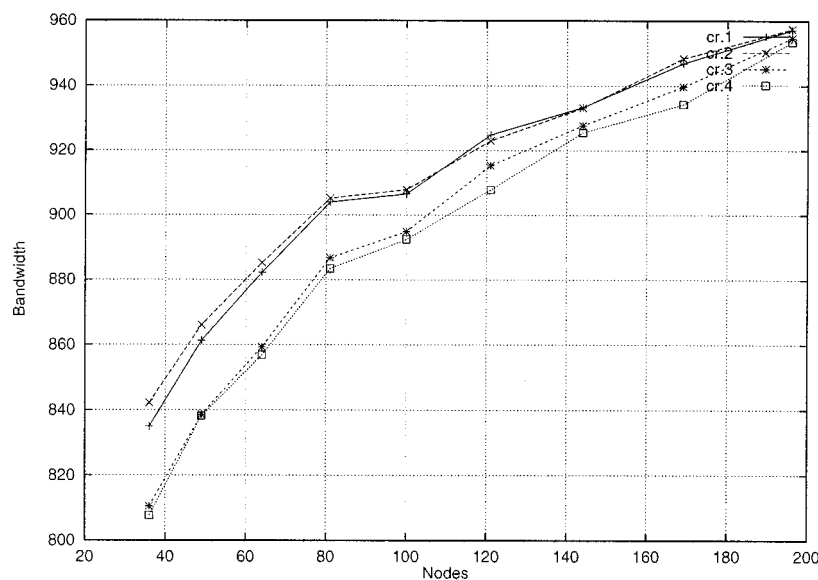

Fig. 18. Bandwidths for the four criteria of RSD. The channel constraints are $c_{ii} = 9$, $c_{ij} = 5$ for $d_{ij} = 1$ and $c_{ij} = 4$ for $d_{ij} = 2$.

1) or 2) and Criteria 3) or 4) becomes larger when the cellular network is a 12×12 or a 13×13 cell system.

The reason for the two different behaviors is found in the way that the two colorings are carried out: Criteria 1) and 2) assign at each step colors to all calls of the nodes, while Criteria 3) and 4) color at each step just one call of the node. That is, Criteria 3) and 4) manage to arrange colors in a smoother and finer way.

VI. CONCLUSION

In this paper, the *randomized saturation degree* heuristic has been introduced to solve the CAP problem on cellular networks. Like other state-of-the-art algorithms, RSD is a *greedy* algorithm, where objects are ordered according to a specific criterion. Indeed, RSD is a heuristic that aims at finding a "good ordering" of calls. The main difference is that, while in [11], [16],

and [19] the ordering is found according to the characteristics of the graph and then maintained during the coloring procedure, in RSD the ordering is carried out simultaneously with the assignment, that is, the ith node to be colored is not known until the $(i-1)$th node is colored. In this manner, properties of the current partial coloring can be taken into account.

The experimental results show that the node ordering given by RSD manages to achieve competitive results. RSD is a simple and fast algorithm, which achieves very low bandwidths. Furthermore, its performance is improved when it is combined with a local search technique. Unlike most heuristics presented in the literature, RSD is flexible and easy to adapt to any kind of network. For example, the algorithms in [11] and [19] use the *cluster* technique, which is not suited for some physical networks because it cannot take into account detailed information about the area to be covered. The simple structure of RSD, in-

stead, is based only on *node ordering and randomization*. Therefore, it can be adapted to actual cellular networks, where cells can be of any shape and not necessarily hexagonal.

Further developments of interest are to test RSD either on cellular networks with an arbitrary topology or on different CAP formulations, like formulations that do not aim at minimizing the channel bandwidth, but try to optimize the use of a given fixed bandwidth while minimizing the number of interferences.

ACKNOWLEDGMENT

REFERENCES

[1] R. Battiti, A. A. Bertossi, and M. A. Bonuccelli, "Assigning codes in wireless networks: Bounds and scaling properties," *Wireless Networks*, vol. 5, pp. 195–209, 1999.

[2] D. Brélasz, "New methods to color the vertices of a graph," *Commun. ACM*, vol. 22, pp. 251–256, 1979.

[3] F. Box, "A heuristic technique for assigning frequencies to mobile radio nets.," *IEEE Trans. Veh. Technol.*, vol. VT-27, pp. 57–64, May 1978.

[4] M. Duque-Anton, D. Kunz, and B. Ruber, "Channel assignment for cellular radio using simulated annealing," *IEEE Trans. Veh. Technol.*, vol. 42, pp. 14–21, Feb. 1993.

[5] N. Funabiki and Y. Takefuji, "A neural network algorithm for channel assignments in cellular radio networks," *IEEE Trans. Veh. Technol.*, vol. 40, pp. 14–21, Feb. 1991.

[6] A. Gamst, "Homogeneous distribution of frequencies in a regular hexagonal cell system," *IEEE Trans. Veh. Technol.*, vol. VT-31, pp. 132–144, Aug. 1982.

[7] ——, "Some lower bounds for a class of frequency assignment problems," *IEEE Trans. Veh. Technol.*, vol. VT-35, pp. 8–14, Feb. 1986.

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

[9] W. K. Hale, "Frequency assignment: Theory and applications," *Proc. IEEE*, vol. 68, pp. 1497–1514, Dec. 1980.

[10] S. Jordan and E. J. Schwabe, "Worst-case performance of cellular channel assignment policies," *Wireless Networks*, vol. 2, pp. 265–275, 1996.

[11] S. Kim and S. L. Kim, "A two-phase algorithm for frequency assignment in cellular mobile systems," *IEEE Trans. Veh. Technol.*, vol. 43, pp. 542–548, Aug. 1994.

[12] D. Kunz, "Channel assignment for cellular radio using neural networks," *IEEE Trans. Veh. Technol.*, vol. 40, pp. 188–193, Feb. 1991.

[13] R. Mathar and J. Mattfeld, "Channel assignment in cellular radio networks," *IEEE Trans. Veh. Technol.*, vol. 42, pp. 647–656, Nov. 1993.

[14] A. Sen, T. Roxborough, and S. Medidi, "Upper and lower bounds of a class of channel assignment problems in cellular networks," Tech. Rep. Arizona State Univ., http://www.eas.asu.edu/~csedept/people/faculty/sen.html, 1997.

[15] M. Sengoku, H. Tamura, S. Shinoda, T. Abe, and Y. Kajitani, "Graph teoretical consierations of channel offset systems in a cellular mobile system," *IEEE Trans. Veh. Technol.*, vol. 40, no. 2, pp. 405–411, May 1991.

[16] K. N. Sivarajan, R. J. McEliece, and J. W. Ketchum, "Channel assignment in cellular radio," in *Proc. 39th IEEE Vehicular Technology Conf.*, May 1989, pp. 846–850.

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

[18] D. Tcha, Y. Chung, and T. Choi, "A new lower bound for the frequency assignment problem," *IEEE/ACM Trans. Networking*, vol. 5, no. 1, Feb. 1997.

[19] W. Wang and C. K. Rushforth, "Local search for channel assignment in cellular mobile networks," in *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, 1997, vol. 35, pp. 689–709.

Roberto Battiti (M'00) received the Laurea degree in physics from the University of Trento, Italy, in 1985 and the Ph.D. degree in computation and neural systems from the California Institute of Technology, Pasadena, in 1990.

He has been a Consultant in the area of parallel computing and pattern recognition. Since 1991, he has been with the University of Trento, where he is now full Professor of computer science in the Faculty of Science. He is the Dean of the new international Ph.D. program in information and communication technology and Director of the Alpine Research and Development Laboratory for Networks and Telematics (ARDENT). He is the author or coauthor of about 50 refereed papers for international journals, conferences, and encyclopedias. His main research interests are heuristic algorithms for combinatorial problems, in particular reactive algorithms for maximum clique, satisfiability, coloring, algorithms for massively parallel architectures, code assignment in wireless networks, optical networks protocols, and architectures.

Prof. Battiti is a member of ACM.

Alan Bertossi was born in London, U.K., in 1956. He received the Laurea degree *(summa cum laude)* 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, University of Pisa, as a Research Associate and later as an Associate Professor. Since 1995, he has been with the Department of Mathematics, University of Trento, Italy, as a Professor of computer science. His main research interests are the computational aspects of high-performance, parallel, VLSI, distributed, fault-tolerant, and real-time systems. He has published about 50 refereed papers (two invited) in international journals, conferences, and encyclopedias. He is a Guest Co-Editor for two special issues of *Algorithmica* and *Discrete Applied Mathematics,* both on experimental algorithmics. He is a member of the editorial board of *Information Processing Letters.*

Prof. Bertossi is included in the 1999 edition of *Who's Who in the World* and in the 2000 edition of *Who's Who in Science and Engineering.*

Daniela Cavallaro received the master's degree in applied mathematics *(summa cum laude)* from Trento University, Italy, in 1998.

Since 1999, she has been a Software Developer with HiT Internet Technologies SpA, Italy.