Cellular Channel Assignment: a New Localized and Distributed Strategy

Roberto Battiti*† Alan A. Bertossi*† Mauro Brunato*†

September 14, 2000

Abstract

As the use of mobile communications systems grows, the need arises for new and more efficient *channel allocation* techniques. The total number of available channels on a real-world network is in fact a scarce resource, and many assignment heuristics suffer from a clear lack of flexibility (this is the case of Fixed Channel Allocation), or from high computational and communication complexity (as with channel borrowing techniques). Performance can be improved by representing the system with an objective function whose minimum is associated with a good configuration; the various constraints appear as penalty terms in the function. The problem is thus reduced to the search for a minimum, that is often performed via heuristic algorithms like Hopfield neural networks, simulated annealing or reinforcement learning. These strategies usually require a central process to have global information and decide for all cells.

We consider an objective-function formulation of the channel assignment problem that has been previously solved by search heuristics; we prove that the search time for the global minimum of the objective function is $O(n \log n)$, and therefore there is no need for search techniques.

Finally we show that the algorithm that arises from this formulation can be modified so that global knowledge and synchronization are no longer required, and we give its distributed version. By simulating a cellular network with mobile hosts on a hexagonal cell pattern with uniform call distribution, we show that our technique actually performs better than the best known algorithms.

1 Introduction

Consider a geographic area in which a number of multi-channel transceiver server stations are placed at a suitable mutual distance. Stations are connected to each other by means of a wired network, which we are not concerned about in the present paper. Each station acts as a network entry point for all the mobile radio hosts that can communicate with it at the highest signal-to-noise ratio among all stations. As a result, the geographic area is divided into *cells* whose borders are not well defined, as they depend on the ever-changing radio signal levels, but could be roughly sketched as in Fig. 1, which still depicts a *good* situation where every cell is a simply connected domain: reflections, interference and terrain configurations, as well as handoff policies, might substantially alter it. However, a more regular setting such as the common hexagonal pattern will be considered in experiments, since it is still the most common test ground. Modeling real-world instances is in fact a very hard and delicate work, and the situation is made worse by the policies of many cellular service providers who keep their data reserved, or provide them in a strictly confidential way, so that the scientific community cannot share them.

Usually a server station can be received by server stations in other cells. In this case, mutually interfering stations must employ different communication channels (i. e. frequency bands, time slices or codes from an orthogonal set), in order to avoid *co-channel interference* (interference caused by transmissions on the *same* channel). In its simplest (and most unrealistic) form, the channel assignment problem is equivalent to the Euclidean graph coloring problem, hence it is NP-hard. This problem

*Università di Trento, dipartimento di Matematica, via Sommarive 14, 38050 Povo (TN) — ITALY.
†E-mail: battiti/bertossi/brunato@science.unitn.it

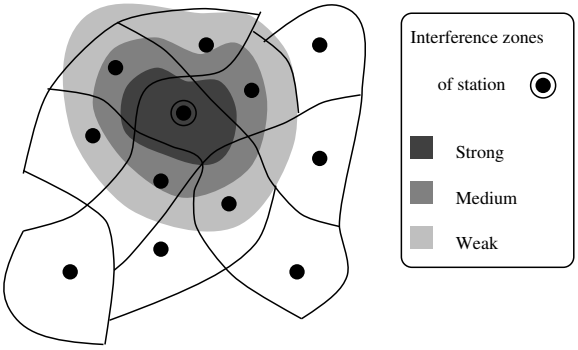

Figure 1: A simple geographic cellular network and the interference area of a server

can be treated with simple greedy heuristics [5] [3] [1]. Because a server station must communicate with several mobile hosts at once, however, we must assign more than one channel to each server. When this problem is handled with graph-coloring heuristics, we need to substitute every node with a clique of cardinality equal to the required number of channels, in order to give the appropriate number of channels to each transceiver while respecting the interference constraints. This approach causes, however, increment in the number of nodes and edges; in particular, the number of nodes is multiplied by the average traffic per cell, while the number of edges is multiplied by the square of this quantity.

Moreover, radio interference is *additive*, and simple adjacency restrictions (like those in the graph or list coloring problems) are not sufficient to catch the complexity of the real-world issues. If the interference phenomena are strong enough, even stations that use different channels may interfere, provided that they operate on adjacent frequency bands or on subsequent time slices (propagation delays may cause a time slice to partially invade another one). This problem becomes significant when the overall frequency spectrum has to be minimized; indeed, the strong request of radio bands for several purposes makes the reserved bandwidth for cellular communications rather small ($\Delta f \approx 60$MHz in the 900MHz band for the GSM system [2]), and hardware techniques[1] can't do all the job by themselves.

In Section 2 we give a fast overview of the algorithms that can be found in the problem literature (Section 2.1); next (Section 2.2) we describe in greater detail the objective-function approach which is central to this paper and we show how the problem of minimizing our function, which leads to an approximate solution of the assignment problem, can be solved in polynomial time (Section 2.3).

In Section 3 we introduce some modifications to the algorithm in order to distribute it among the cells. To do so, we need a communication scheme to broadcast all status changes to a small local cluster (Section 3.1) and a local synchronization method to make sure that a cell allocates a channel only when it has enough knowledge of the status of its neighbors (Section 3.2).

Results of simulations are presented in Section 4, where the algorithm we describe is compared with those introduced in Section 2.1. In section 5 we draw some conclusions suggested by the experimental analysis.

2 Channel Assignment Algorithms

2.1 Combinatorial strategies

Let us briefly summarize the main channel assignment algorithms (for more details see [7] [6]). In the experimental Section 4 we shall consider the following techniques:

- The FCA (Fixed Channel Allocation) algorithm. Each cell is assigned a fixed pool of frequencies, so that no near cells can use the same channel. No communication is needed between cells; when all channels are in use, subsequent requests shall be rejected.

[1] Namely, Minimum Shift Keying and Gaussian Minimum Shift Keying to eliminate spectrum bumps around the channel, and frequency hopping schemes to eliminate constant interferences.

- The SBR (Simple Borrow from the Richest) algorithm. Each cell has an assigned pool of frequencies, but a channel can be borrowed from a richer neighbor, provided that its use does not cause interference. When more than one neighbor has a free channel, the cell chooses the richest one. It is more efficient than FCA with low traffic rates, but its performance deteriorates when the traffic rate increases, achieving the same performance of FCA at higher computational and communication costs.

- One of the many variations of DCA (Dynamic Channel Assignment) techniques, by which every cell can have access to every channel, as long as it does not cause interference; the cell chooses the channel which is most 'blocked' (due to the interference constraints) in the neighboring cells, so that it gives rise to the least blocking probability. It is better than FCA at low traffic rates, but worse at high traffic rates, because many cells might find no channels at all, due to non-optimal decisions at previous times.

- The BDCL (Borrow with Directional Channel Locking) algorithm. Like SBR, but the choice of the channel to borrow is done by the criterion of the above proposed DCA technique. It is the best combinatorial algorithm we know.

In Section 4 we shall compare these algorithms with the following technique, based on objective function minimization.

2.2 A penalty function heuristic

Many penalty-function heuristics have been applied to the channel assignment problem. Some of them require the rearrangement of the whole cellular system when a new channel is requested [9] [4]; unless the traffic is very low and service communications among cells are cheap and fast, this approach is of little practical use. Other penalty-function heuristics, such as the one we shall consider next, just rearrange the channel assignment inside the cell where the new communication request is issued, by choosing those channels that minimize the probability of a future channel refusal, trying to keep the system in a suitable status for future requests (this effort of maintaining a good configuration is clearly unnecessary in the former case).

In both cases, two heuristic steps must be taken. First, we must heuristically determine a good penalty function, where "good" means that its minimum should correspond to a suitable configuration, sufficiently close to the optimal assignment. Then, when this function is determined, we must actually locate its global minimum, or at least find some good local minimum; to do so, we need to apply a minimum-search heuristic technique to the penalty function.

A penalty function that has been used by previous researchers which empirically finds good approximations of the optimal assignment is shown in [8]. Following its notation, let n_{CE} be the number of cells and n_{CH} the total number of channels. Every cell i, $i = 1, \ldots, n_{\mathrm{CE}}$, has a traffic demand traf_i which changes with time. Let us denote with $d_{ii'}$ the Euclidean distance between the centers of cells i and i', and let $\mathrm{interf}_{ii'}$ be a $\{0,1\}$-valued function which states if the two cells interfere or not; the notation can nonetheless be extended to the case of various degrees of interference.

When a connection or termination request is issued in cell i^*, the frequency allocation in this cell must be optimized. The status of channel allocation is given by a $\{0,1\}$-valued matrix A_{ij} whose entry (i,j) is 1 if and only if channel j is currently in use in cell i. The new channel allocation for cell i^* is stored in vector $V_j^{(i^*)}$, $j = 1, \ldots, n_{\mathrm{CH}}$.

An objective function is built whose minimum is likely to be a good solution of the new allocation for cell i^*. First, a term to privilege those solutions without interference (all terms depend on $V^{(i^*)}$, our unknown solution) is introduced[2]:

$$a(V) = \sum_{j=1}^{n_{\mathrm{CH}}} \sum_{\substack{i=1 \\ i \neq i^*}}^{n_{\mathrm{CE}}} V_j A_{ij} \, \mathrm{interf}_{ii^*} \, .$$

[2]From now on, we shall drop the (i^*) superscript from vector V, since all calculations are local to cell i^*.

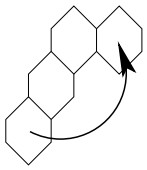

Figure 2: Building a reuse scheme: the basic move

This term adds 1 for each cell interfering with i^* which uses a channel in use in i^*. Second, the requests of the cell i^* should be respected as much as possible:

$$b(V) = \left(\mathrm{traf}_{i^*} - \sum_{j=1}^{n_{\mathrm{CH}}} V_j \right)^2 .$$

The only reason to make this term quadratic is that it must be nonnegative: (an absolute value would also work). Third, a *packing condition* is added: a channel should be reused as near as possible (outside the interference zone), to restrict the blocking probability in other cells.

$$c(V) = -\sum_{j=1}^{n_{\mathrm{CH}}} \sum_{\substack{i=1 \\ i \neq i^*}}^{n_{\mathrm{CE}}} V_j A_{ij} \frac{1 - \mathrm{interf}_{ii^*}}{d_{ii^*}}.$$

This subtracts a positive term for each cell outside the interference zone which reuses a channel employed in cell i^*; the larger the distance, the smaller the subtracted term. Next, changes in the present allocation of the cell i^* should be minimized:

$$d(V) = -\sum_{j=1}^{n_{\mathrm{CH}}} V_j A_{i^* j}.$$

This subtracts 1 every time a channel currently used by cell i^* is chosen for the next configuration (this means that a mobile host needs to change its channel as rarely as possible). If some frequency hopping technique is used, however, this requirement does not make much sense, as the mobile host is equipped for frequent configuration changes. Last, experimental evidence shows that to achieve a good performance the channel reuse should follow a regular scheme (for example, a compact pattern [10]). This is achieved by introducing the $\{0,1\}$-valued matrix $\mathrm{res}_{ii'}$ whose entry (i,i') is 1 if and only if cells i and i' belong to the same reuse scheme (i.e. should use the same channels if possible). Common reuse schemes follow some sort of "knight" move (for instance, the one shown in Fig. 2).

$$e(V) = \sum_{j=1}^{n_{\mathrm{CH}}} \sum_{\substack{i=1 \\ i \neq i^*}}^{n_{\mathrm{CE}}} V_j A_{ij} (1 - \mathrm{res}_{ii^*}).$$

Note that all terms are arranged to go towards a lower value when the constraints are satisfied. Let us combine them in a single objective function to minimize:

$$J(V) = A \cdot a(V) + B \cdot b(V) + C \cdot c(V) + D \cdot d(V) + E \cdot e(V),$$

where A, B, C, D and E give different importance to the various constraints.

2.3 The polynomial algorithm BBB

So far a possible penalty function $J(V)$ has been heuristically determined. To minimize $J(V)$, [8] employs Hopfield neural networks, but actually the minimization of this function is straightforward

and does not require any heuristic search technique. In fact, we can rewrite $J(V)$ as a quadratic function in which the quadratic term *depends only on the number of channels*, and not on the single channels used. Let us rewrite

$$a(V) = \sum_{j=1}^{n_{\mathrm{CH}}} V_j a_j, \qquad \text{where} \qquad a_j = \sum_{\substack{i=1 \\ i \neq i^*}}^{n_{\mathrm{CE}}} A_{ij} \, \mathrm{interf}_{ii^*};$$

the term a_j simply counts the number of cells in the interference zone of i^* which use the channel j. The term $b(V)$ can be rewritten as

$$b(V) = \left(\sum_{j=1}^{n_{\mathrm{CH}}} V_j \right)^2 - 2 \, \mathrm{traf}_{i^*} \sum_{j=1}^{n_{\mathrm{CH}}} V_j.$$

The $\mathrm{traf}_{i^*}^2$ term is constant and can be omitted, while the quadratic term is the square of the number of channels reserved for the cell i^* (the number of 1's in vector V). Let us rewrite it in a way similar to the other ones:

$$b(V) = \left(\sum_{j=1}^{n_{\mathrm{CH}}} V_j \right)^2 + \sum_{j=1}^{n_{\mathrm{CH}}} V_j b, \qquad \text{where} \qquad b = -2 \, \mathrm{traf}_{i^*} \, .$$

Clearly, b does not depend on j. The other terms can be rewritten as follows:

$$c(V) = \sum_{j=1}^{n_{\mathrm{CH}}} V_j c_j, \quad d(V) = \sum_{j=1}^{n_{\mathrm{CH}}} V_j d_j, \quad e(V) = \sum_{j=1}^{n_{\mathrm{CH}}} V_j e_j$$

where

$$c_j = - \sum_{\substack{i=1 \\ i \neq i^*}}^{n_{\mathrm{CE}}} A_{ij} \frac{1 - \mathrm{interf}_{ii^*}}{d_{ii^*}},$$

$$d_j = -A_{i^* j}, \qquad e_j = \sum_{\substack{i=1 \\ i \neq i^*}}^{n_{\mathrm{CE}}} A_{ij} (1 - \mathrm{res}_{ii^*}).$$

The term c_j evaluates the packing condition for channel j; the term d_j rewards the choice of channel j if it was already in use; the term e_j penalizes the use of a channel outside the reuse scheme.

We can collect the single coefficients into global ones:

$$w_j = A \cdot a_j + B \cdot b + C \cdot c_j + D \cdot d_j + E \cdot e_j;$$

the global objective function is then

$$J(V) = \left(\sum_{j=1}^{n_{\mathrm{CH}}} V_j \right)^2 + \sum_{j=1}^{n_{\mathrm{CH}}} w_j V_j,$$

where, as we have already pointed out, the square term is just the square of the number of assigned channels.

To minimize $J(V)$ we calculate the weights w_j for each channel; each calculation requires at most n_{CE} steps to test interferences, reuses and packing. Globally, the calculation of the weights w_j requires time $O(n_{\mathrm{CE}} n_{\mathrm{CH}})$. If we had fixed the number n of channels that we want to assign, the minimization would be achieved by taking the channels j whose w_j are the least (the quadratic term is constant among the solutions with the same number of channels). To take advantage of this, we calculate a permutation σ_j, $j = 1, \ldots, n_{\mathrm{CH}}$, such that the vector $(w_{\sigma_j})_{j=1,\ldots,n_{\mathrm{CH}}}$ is sorted in increasing order.

The sort requires time $O(n_{\mathrm{CH}} \log n_{\mathrm{CH}})$. At last, let us call J_n the minimum of the objective function restricted to n-channel solutions. Its value is

$$J_n = n^2 + \sum_{j=1}^{n} w_{\sigma_j}, \qquad n = 0, \ldots, n_{\mathrm{CH}},$$

and the difference between the minima for successive values of n is

$$J_n - J_{n-1} = 2n - 1 + w_{\sigma_n}, \qquad n = 1, \ldots, n_{\mathrm{CH}}.$$

So, a simple scan of the vector w_{σ_j} is enough to find the minimum for all n, that is the global minimum, in time $O(n_{\mathrm{CH}})$.

Hence, the global minimum of the objective function,

$$\min_{V \in \{0,1\}^{n_{\mathrm{CH}}}} J(V) = \min_{n=0,\ldots,n_{\mathrm{CH}}} J_n,$$

can be found in total time $O\big(n_{\mathrm{CH}}(n_{\mathrm{CE}} + \log n_{\mathrm{CH}})\big)$. The procedure returns also the number of channels in the optimal solution, say n^*, therefore the channels to be assigned to cell i^* are

$$n_{\sigma_1}, n_{\sigma_2}, \ldots, n_{\sigma_{n^*}}.$$

Let us call this penalty-function minimization heuristic the BBB algorithm. The first advantage of using BBB is that, of course, we find the true global minimum of $J(V)$. In addition, consider that Hopfield networks, like many other techniques, require our function coefficients to vary only in a certain range in order to ensure stability and convergence of the search; in other words, coefficients are critical not only in weighting the various constraints (which is precisely what they are introduced for), but also in making the minimum-search procedure succeed. Algorithm BBB does not use any heuristic search algorithm, so it is not restricted to those coefficient values that ensure convergence, and we may let them vary over all the nonnegative real range, thus having more freedom in tuning them.

3 The distributed polynomial algorithm dBBB

We first note that algorithm BBB can be improved by storing at each cell a permanently sorted array of weights to be updated at each change of state in the nearby cells. The sorting time can thus be cut down to a simple update of the sorted array at each call. Moreover, by considering only local interference the calculation of the weights does not depend on the total number n_{CE} of cells, but it can be performed in $O(1)$ time.

We need, however, to simplify our objective function by eliminating non-locality. There are only two global terms in the function:

- The "reuse scheme" given by the array $\mathrm{res}_{ii'}$; some tests (Fig. 8, Section 4) show that it is not influential on the overall system performance.
- The "packing condition", whose weight decreases at increasing distances; the same preliminary tests cited above prove that we can restrict the "packing condition" to the $2r$-th ring of neighboring cells (where r is the interference radius).

3.1 A communication scheme

Once the objective function is localized, we just need a good communication strategy to replace the central authority which took all the decisions in the previous algorithm. When a cell initiates or terminates a call, it must broadcast its new status to its neighbors (up to the $2r$-th ring). To do so, we need a simple broadcasting scheme, like the one presented in Fig. 3.

Let the message be structured as a tuple (c, r, h, m), where c is a flag indicating if the message must be duplicated: if c is set, the message shall be called a "corner" message; r is the maximum distance

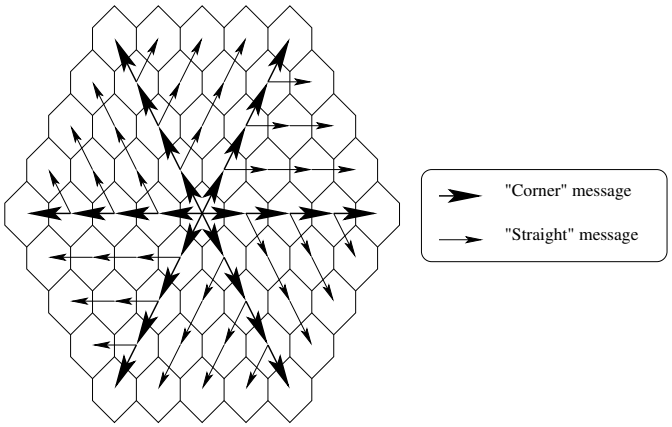

Figure 3: A local broadcasting scheme

```
1.  source:
2.      for d in 0...5 do
3.          send (1, 4, 1, m) along direction d;
4.  relay:
5.      upon receipt of (c, r, h, m) from direction d do
6.          if h < r then
7.              send (c, r, h + 1, m) along direction d + 3   (mod 6);
8.              if c then
9.                  send (0, r, h + 1, m) along direction d + 4   (mod 6);
10.         Act according to the received message (c, r, h, m);
```

Figure 4: The local broadcasting algorithm

from the source the message must reach, h is the number of steps the message has taken up to now and m is the message itself. Suppose that in each cell the directions of incoming and outgoing messages are numbered clockwise modulo 6. Then if a message arrives on direction d, the opposite direction will be $d + 3 \pmod 6$.

The algorithm can be implemented as in Fig. 4. The source sends six "corner messages" to its neighbors (lines 2–3); when a cell receives a message, if it is not far enough (line 6), it must relay it to the cell on the opposite side (line 7); if it is a corner message, the cell must also propagate a non-corner ($c = 0$) copy of the message to the clockwise-next direction (lines 8–9). This copy shall be subsequently propagated only in straight line. After the propagation of the message, of course, the relay cell must modify its record about the broadcasting cell according to the message content m (line 10).

3.2 A Mutual Exclusion technique

Last, to avoid conflicts in channel choice we must ensure that, when a cell is changing its configuration, none of its neighbors up to the reuse distance does the same thing simultaneously. For this we can implement a multiple token-passing protocol such that no two tokens are nearer than the reuse distance. If the reuse distance is two and the network is a regular hexagonal grid, let us refer to Fig. 5. The grey cells possess the token; when a cell is done with it, it sends a "token" message "upwards" (following the thick arrow) and two "free" messages along the thin arrows (this requires two other cells to act as relays). Before entering the critical state, a cell must wait for one "token" and two "free" messages. The "token" message ensures that all preceding cells in the token-passing chain are safe, while the two "free" messages declare the safety of the two potentially blocking cells which are possibly using

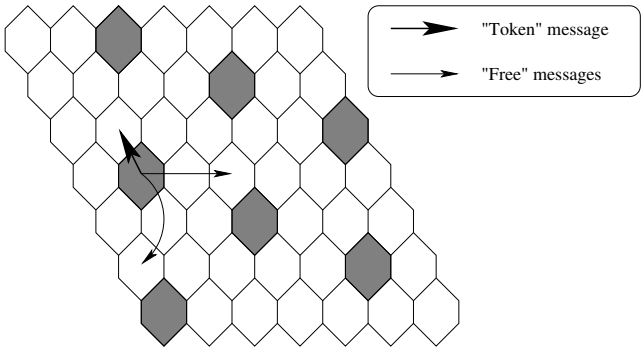

Figure 5: A multiple token-passing scheme

the token at the same time in its neighborhood. To take account of border effects, however, the two leftmost columns and the two uppermost rows should wait for just one "free" message, the cells in the upper left corner don't have to wait for any "free" message, while the lower row cells should generate a token (without waiting for one) whenever they get enough "free" messages.

The actual algorithm for a 7×7 grid with a reuse distance of 2 is presented in Fig. 6 (the directions are numbered clockwise from 0 to 5 starting from west). Three variables are used to store the status of the cell. The boolean *Token_flag* is **true** if and only if the cell is holding the token; the counter *Free_count* contains the number of "free" messages that still have to be received before the cell is able to use the token; the flag *Critical_section_flag* is **true** if and only if a channel request or release has been issued in the cell. The entry to the critical section is only allowed when *Token_flag* is **true** and *Free_count* is 0.

All cells must initially call the *Init* procedure to initialize their status, except those initially possessing the token (the grey ones in Fig. 5). The cells that initially possess the token must start calling the procedure *Have_token_at_the_beginning*. Cells that need to process a channel request (or release) must call the procedure *Enter_critical_section* that sets the *Critical_section_flag*, so that the algorithm runs the channel-assignment procedure when possible, and waits until that same bit is reset.

Of course the main loop is executed concurrently, and the *Check* procedure must end before any other message is received. For routing purposes, since they have to travel two cells, the "Free" messages have an integer part. When a cell receives a "free(0)" message, it just has to retransmit it as "free(1)" to a direction that depends on the incoming path (lines 30–33), while a "free(1)" message must be operated on place (lines 34–36) by procedure *Check*, that verifies if the cell has to enter the critical section, skip it or wait because it isn't ready.

4 Experimental analysis

We consider a 7×7 hexagonal grid, as the one shown in Fig. 5, like in most of the literature. A fixed server station is placed at the center of each cell, while a number of mobile hosts is free to move across the whole land. The total number of available channels is 70 and the co-channel interference is extended to the second ring of neighbors. The grid does not wrap like a torus.

A C++ program has been written to run a comparative simulation of five algorithms (FCA, SBR, DCA with local optimization, BDCL and BBB).

For FCA, the reuse scheme (a reuse distance of two cells has been considered) consists in seven partitions of ten channels each. The Euclidean distance between the centers of two neighbors is 1, and the reuse scheme given by the function $\text{res}_{ii'}$ of Section 2.2 is built by iterating the basic "knight" move of Fig. 2, which gives the same pattern as the token placement in Fig. 5. The same scheme has been used to distribute the seven channel groups among the cells for the FCA algorithm.

Coefficients for the function $J(V)$ are shown in table 1.

Each algorithm (FCA SBR, DCA, BDCL, BBB) has been simulated for connection rates of 160,

```
1.  Procedure Have_token_at_the_beginning:
2.      Token_flag ← true;
3.      Free_count ← 0;
4.      do Check

5.  Procedure Init:
6.      if node is in the first row then
7.          Token_flag ← true
8.      else
9.          Token_flag ← false;
10.     Free_count ← 2;
11.     if node is in the two upper rows then
12.         decrease Free_count;
13.     if node is in the two leftmost columns then
14.         decrease Free_count;
15.     Critical_section_flag ← false;

16. Procedure Enter_critical_section:
17.     Critical_section_flag ← true;
18.     wait until Critical_section_flag = false;

19. Procedure Check:
20.     if Token_flag and Free_count = 0 then
21.         if Critical_section_flag then
22.             Run the channel assignment procedure;
23.         Send Token along direction 1;
24.         Send Free(0) along directions 3 and 4;
25.         do Init

26. Main polling loop:
27.     Upon receipt of Token do
28.         Token_flag ← true;
29.         do Check
30.     Upon receipt of Free(0) from direction 1 do
31.         Send Free(1) along direction 5;
32.     Upon receipt of Free(0) from direction 0 do
33.         Send Free(1) along direction 3;
34.     Upon receipt of Free(1) do
35.         decrease Free_count;
36.         do Check
```

Figure 6: The token-passing algorithm

Table 1: Coefficients for function $J(V)$

Coefficient	Value
A	7000
B	45
C	1.2625
D	0.01
E	4.17625

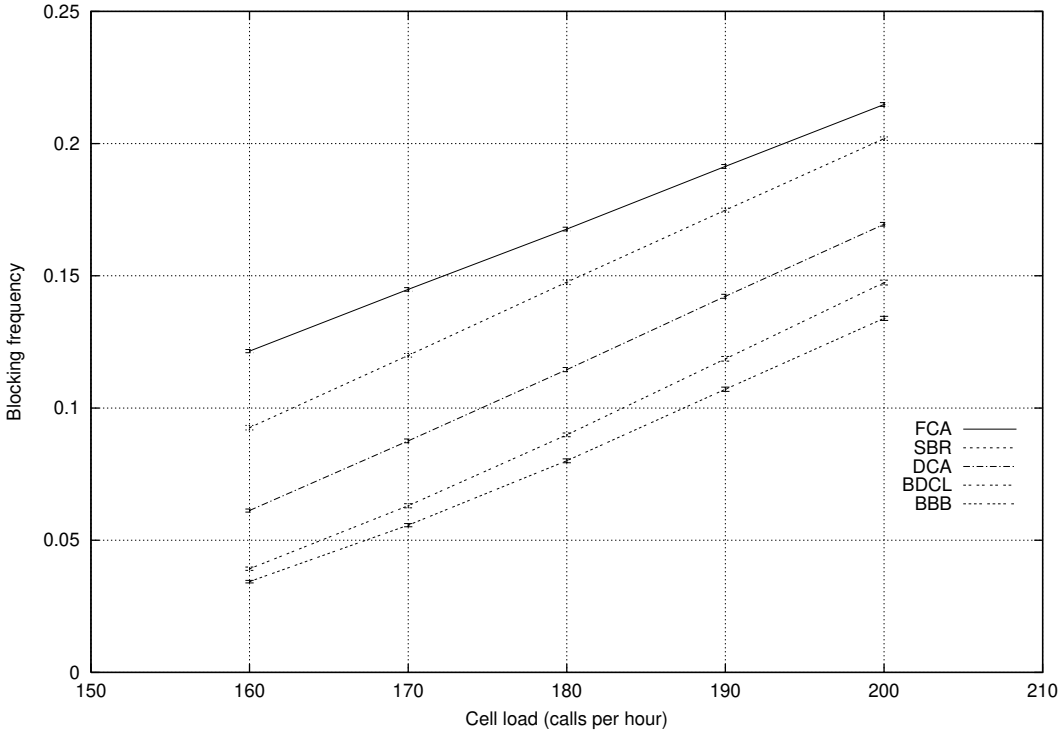

Figure 7: Comparison among algorithms

Table 2: Comparison among algorithms (see fig. 7)

Cell load (cph)	FCA	Err ($\times 10^{-3}$)	SBR	Err ($\times 10^{-3}$)	DCA	Err ($\times 10^{-3}$)	BDCL	Err ($\times 10^{-3}$)	BBB	Err ($\times 10^{-3}$)
160	.121	.60	.093	.85	.061	.57	.039	.61	.034	.48
170	.145	.69	.120	.69	.088	.70	.063	.79	.056	.62
180	.168	.70	.148	.89	.115	.77	.090	.72	.080	.73
190	.191	.75	.175	.75	.142	.76	.119	.88	.107	.79
200	.215	.69	.202	.64	.169	.71	.147	.93	.134	.77

Table 3: Global vs. local optimization (see fig. 8)

Cell load (cph)	BBB	Err ($\times 10^{-3}$)	dBBB	Err ($\times 10^{-3}$)
160	.034	.48	.033	.64
170	.056	.62	.055	.54
180	.080	.73	.080	.69
190	.107	.79	.106	.65
200	.134	.77	.134	.80

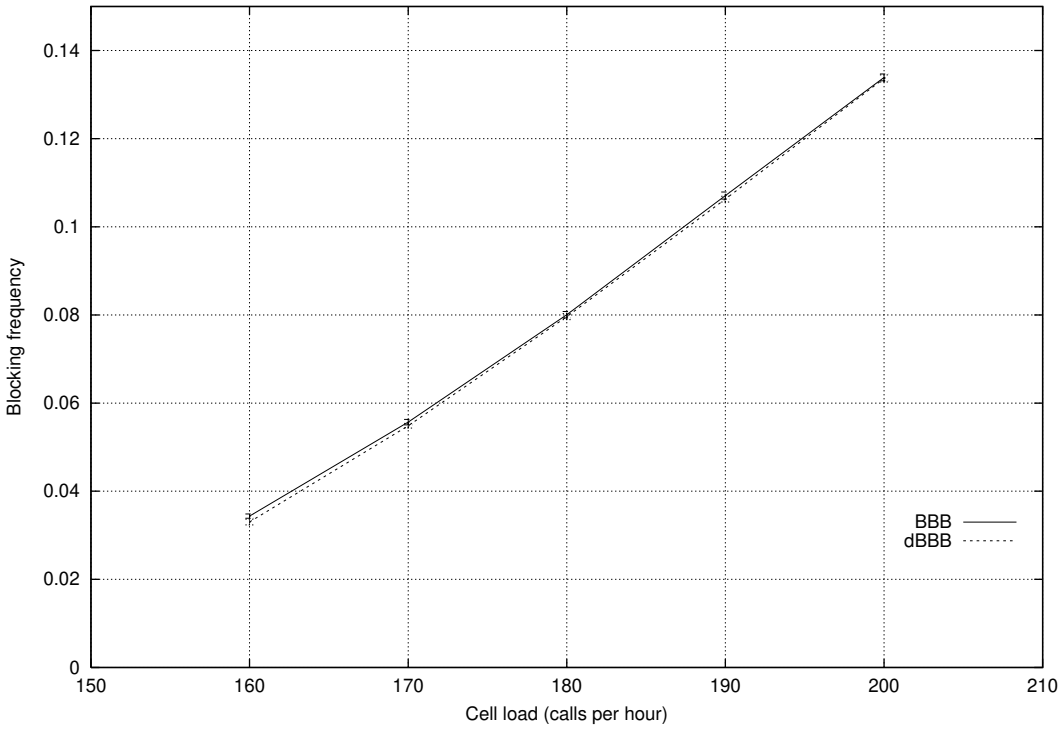

Figure 8: Global vs. local optimization

170, 180, 190 and 200 calls per hour with Poissonian distribution (and hence exponential interarrival time), corresponding to a traffic of 8, 8.5, 9, 9.5 and 10 erlangs. The mean duration of a connection is exponential with an average of 180sec. Each simulation consisted of 50 runs of 100000 seconds of simulated time each.

Fig. 7 shows the results of this simulation. Every error bar represents the 95% confidence interval calculated over 50 independent runs. It is clear that algorithm BBB outperforms all these algorithms.

The elimination of the reuse scheme and the localization of the channel packing condition as in Section 3 finally lead us to the data in Fig. 8, where we compare the original BBB algorithm with its distributed version dBBB. Because the error bars are not visible in the figures, they are also reported in tables 2 and 3. As it is apparent, all differences lie within the confidence intervals. This enables us to transform the algorithm into a local one with no performance loss.

5 Conclusions

After the description of some well known algorithms, we have introduced the penalty-function algorithm BBB, we have described its distributed version dBBB and simulated it over a large number of runs. As we have seen, our heuristic behaves better than the others we tried, at least when compared on regular hexagonal patterns.

Due to their nature, most algorithms have only been studied with such restrictive assumptions as "boolean" (non-additive) interference, and so our tests had to restrict to that case. However, our method can easily be applied to a large variety of channel assignment problems with additive interference constraints, adjacent-channel interference and so on. In fact, all that is required for the minimization to be polynomial is that every channel provides a *linear* contribution to the total penalty function.

References

[1] Roberto Battiti, Alan A. Bertossi, and Maurizio A. Bonuccelli. Assigning codes in wireless networks: Bounds and scaling properties. *Wireless Networks*, 5:195–209, 1999.

[2] Onelio Bertazioli and Lorenzo Favalli. *GSM — Il Sistema Europeo di Comunicazione Mobile: Tecniche, Architettura e Procedure*. ATES. Hoepli, Milan, 1996.

[3] Alan A. Bertossi and Maurizio A. Bonuccelli. Code assignment for hidden terminal interference avoidance in multihop packet radio networks. *IEEE/ACM Transactions on Networking*, 3:441–449, 1995.

[4] Manuel Duque-Antón, Dietmar Kunk, and Bernhard Rüber. Channel assignment for cellular radio networks using simulated annealing. *IEEE Transactions on Vehicular Technology*, 42(1):14–21, February 1993.

[5] Eli Upfal Eli Shamir. Sequential and distributed graph coloring algorithms with performance analysis in random graph spaces. *Journal of Algorithms*, 5:488–501, 1984.

[6] Scott Jordan and Eric J. Schwabe. Worst-case performance of cellular channel assignment policies. *Wireless Networks*, 2:265–275, 1996.

[7] Irene Katzela and Mahmoud Nagshineh. Channel assignment schemes for cellular mobile telecommunication systems: A comprehensive survey. *IEEE Personal Communications*, pages 10–31, June 1996.

[8] Enrico Del Re, Romano Fantacci, and Luca Ronga. A dynamic channel allocation technique based on hopfield neural networks. *IEEE Transactions on Vehicular Technology*, 45(1):26–32, February 1996.

[9] Satinder Singh and Dimitri Bertsekas. Reinforcement learning for dynamic channel allocation in cellular telephone systems. *Submitted to NIPS96*, 1996.

[10] Kwan Lawrence Yeung and Tak-Shing Peter Yum. Compact pattern based dynamic channel assignment for cellular mobile systems. *IEEE Transactions on Vehicular Technology*, 43(4):892–896, November 1994.