Cellular Channel Assignment: Comparing and Simplifying Heuristics

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

September 3, 1997

Abstract

As the use of mobile communications systems grows, the need arises for new and more efficient *channel allocation* techniques. In the model problem a geographic area is partitioned into a number of *cells*, each one served by a fixed station. When a mobile host wants to establish a connection to the network, it asks for a *free channel* to the server of the cell. The total number of available channels on a real-world network is a scarce resource, and many assignment heuristics suffer from a clear lack of flexibility (FCA), or from high computational and communication complexity (BCO, BDCL). Performance can be improved by representing the system with an objective function whose minimum is associated to a good configuration; the various constraints appear as penalty terms in the function. The problem is thus reduced to the search for a global minimum, that is often performed via heuristic algorithms like Hopfield neural networks, simulated annealing, reinforcement learning. These strategies require a central process to have global information and decide for all cells.

We consider a problem that has been previously solved by heuristics, we demonstrate that the search time for the global minimum is $O(n \log n)$, and therefore there is no need for search techniques, and that the algorithm can be distributed. We compare the main algorithms by simulating a cellular network with mobile hosts on the well-established hexagonal-cell pattern with a uniform call arrival distribution.

1 Cellular Networks

Consider a geographic area partitioned into zones (*cells*). In each cell a multi-channel transceiver server station is placed in a convenient place with the purpose to serve all the mobile hosts which are found in its cell. Of course, the area reached by the signal of the station is larger than the cell itself. As in figure 1, the cells need not be regular or of equal size; however, most of the times we shall refer to a more regular setting, like the hexagonal cell pattern used in practice.

Usually a server station can be received by server stations in other cells. In this case, mutually interferring 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 form, the channel assignment problem is equivalent to the Euclidean graph coloring problem, hence it is NP-hard. This problem can be treated with simple greedy heuristics [8] [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 treated with graph-coloring heuristics, the substitution of every node with a clique of cardinality equal to the required number of channels causes the combinatorial explosion of the problem.

The general problem, however, can be made harder. For instance, 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 subsequential time slices (propagation delays may cause a time slice to partially invade another one). This problem becomes significant when the overall frequency

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

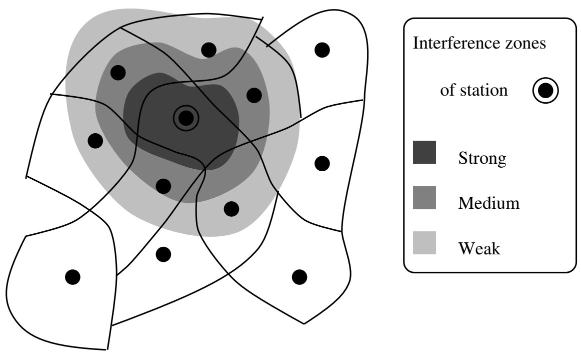

Figure 1: A geographic cellular network and its interference area

spectrum has to be minimized; indeed, the strong request of radio bands for several purposes makes the reserved bandwidth for cellular communications rather small ($\approx$ 60MHz in the 900MHz band for the GSM system [2]), and hardware techniques[1] can't do all the job by themselves.

2 Channel Assignment Algorithms

2.1 Combinatorial strategies

The state of the art about channel assignment algorithms is presented in [6]. Many algorithms are analyzed in [5] In particular, we 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.
- 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 it equals its performances when the traffic rate increases.
- A DCA (Dynamic Channel Assignment) technique, 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 themselves with no channels at all.
- 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 maybe the best known combinatorial algorithm.

Later, we shall compare these algorithms with the following technique based on objective function minimization [7].

2.2 A penalty function heuristic

Many heuristics have been applied to the channel assignment problem ([9], [4], [7]). Let us consider the case [7]. We have n_{CE} cells and a total number n_{CH} of channels. Every cell i, $i = 1, \ldots, n_{\mathrm{CE}}$, has

[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.

a traffic demand traf_i. 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 interfer or not. When a connection or termination request is issued in cell i^*, we must optimize the frequency allocation in this cell. 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, $j = 1, \ldots, n_{\mathrm{CH}}$.

Let us build an objective function whose minimum is likely to be a good solution of the new allocation for cell i^*. First, we need a term to privilege those solutions without interference (all terms depend on V, our unknown solution):

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

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 we want it nonnegative. Third, we add a *packing condition*: we want to reuse a channel 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 the frequency hopping technique is used, however, this requirement does not make much sense, as the host is equipped for 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]). We introduce the $\{0,1\}$-valued matrix $\mathrm{res}_{ii'}$ whose entry (i,i') is 1 if and only if the cells i and i' belong to the same reuse scheme (should use the same channels if possible). Common reuse schemes follow some sort of "knight" move (for instance, the one shown in figure 2).

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

Notice 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:

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

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

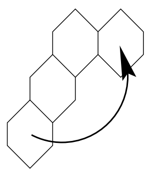

Figure 2: Building a reuse scheme: the basic move

2.3 A polynomial algorithm

To minimize $J(V)$, [7] employs Hopfield neural networks, but actually the minimization of this function is straightforward and does not require any 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

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

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

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, reusals and packings. 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

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

Because we do not use any heuristic search algorithm, we are not restricted to those values for A, B, C, D and E which ensure stability of the search, but we may let them variate over all the nonnegative real range.

3 A distributed polynomial algorithm

We first note that the previous technique 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 $O(n \log n)$ sorting time can thus be cut down to a simple $O(n)$ update of the array at each call.

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 preliminary tests shown in section 4 will suggest that it is ininfluent 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 figure 3. The message contains a flag bit; if the bit is set, we shall call it a "corner" message. The source sends six "corner messages" to

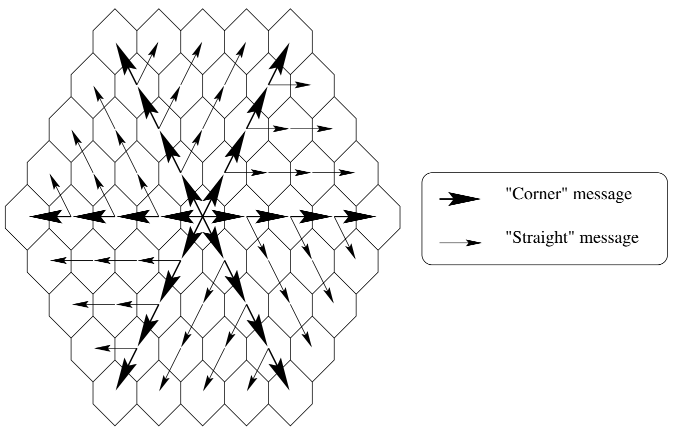

Figure 3: A local broadcasting scheme

```
source:   for $d$ in $0\dots5$ do
              send $(1,4,1,m)$ along direction $d$;
relay:    upon receipt of $(c,r,h,m)$ from direction $d$ do begin
              if $h < r$ then begin
                  send $(c,r,h+1,m)$ along direction $d+3 \pmod 6$;
                  if $c$ then
                      send $(0,r,h+1,m)$ along direction $d+4 \pmod 6$
              end;
              Act according to the received message $(c,r,h,m)$
          end;
```

Figure 4: The local broadcasting algorithm

its neighbors; when a cell receives a message, if it is not far enough, it must relay it to the cell on the opposite side; if it is a corner message, it must also propagate a non-corner copy of the message to the next direction (clockwise).

Let us now consider in greater detail the implementation. Let the message be structured as a tuple (c,r,h,m), where c is the "corner" flag, indicating if the message must be duplicated, r is the maximum distance 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 $d+3 \pmod 6$. For the case of two rings of interferring cells and of propagation distance up to 4, the algorithm is shown in figure 4.

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 distance is two and the network is a regular hexagonal grid, let us refer to figure 5. The grey cells are possessing 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 possible blocking cells which are possibly using the token at the same time in its neighborhood. To take account of border effects, however, the two leftmost

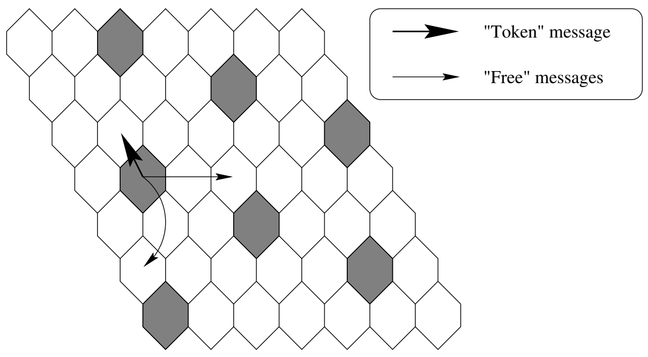

Figure 5: A multiple token-passing scheme

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 whenever they get enough “free” messages.

The actual algorithm for a 7×7 grid with a reuse distance of 2 is presented in figure 6 (the directions are numbered from 0 to 5 going clockwise and starting from west). All cells must initially call the *Init* procedure, except those initially possessing the token (the grey ones in figure 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, the “Free” messages have an integer part.

4 Experiments and tests

We consider seven rows of seven hexagonal cells, disposed as in figure fig:tokens, 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.

A program has been written in C++ to execute simulations of the type described in [7]. Figure 7 shows the preliminary results of a series of simulations of five algorithms (FCA, SBR, BDCL, DCA with local optimization and the penalty optimization heuristic described in section 2.2).

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 figure 2, which gives the same pattern as the token placement in figure 5. The same scheme has been used to distribute the seven channel groups among the cells for the FCA algorithm.

The mean duration of a connection is exponential with an average of 180sec; the simulations have been run for 10000 seconds (of simulated time) and connection rates of 160, 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 results are shown in figure 7, where we have tested the original asset of [7]. The elimination of the reuse scheme and the localisation of the channel packing condition as in section 3 finally lead us to the data in figure 8, where we compare the original and the distributed scheme. As it is apparent, all differences lie within the standard error. This enables us to transform the algorithm into a local one with no performance loss.

```
Procedure Have_token_at_the_beginning:
        Token_flag ← true;
        Free_count ← 0;
        do Check
end procedure Have_token_at_the_beginning;
Procedure Init:
        if (node is in the first row) then
            Token_flag ← true
        else
            Token_flag ← false;
        Free_count ← 2;
        if node is in the two upper rows then
            decrease Free_count;
        if node is in the two leftmost columns then
            decrease Free_count;
        Critical_section_flag ← false;
end procedure Init;
Procedure Enter_critical_section:
        Critical_section_flag ← true;
        wait until Critical_section_flag = false;
end procedure Enter_critical_section;
procedure Check:
        if Token_flag and Free_count = 0 then begin
            if Critical_section_flag then
                Run the channel assignment procedure;
            Send Token along direction 1;
            Send Free(0) along directions 3 and 4;
            do Init
        end
end procedure Check;
Main polling loop:
        Upon receipt of Token do begin
            Token_flag ← true;
            do Check
        end;
        Upon receipt of Free(0) from direction 1 do
            send Free(1) along direction 5;
        Upon receipt of Free(0) from direction 0 do
            send Free(1) along direction 3;
        Upon receipt of Free(1) do begin
            decrease Free_count;
            do Check
        end;
End main loop.
```

Figure 6: The token-passing algorithm

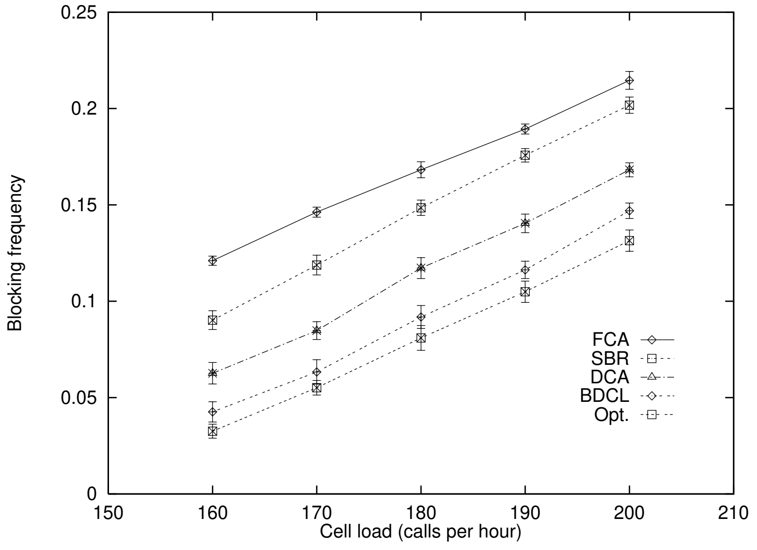

Figure 7: Comparison among algorithms

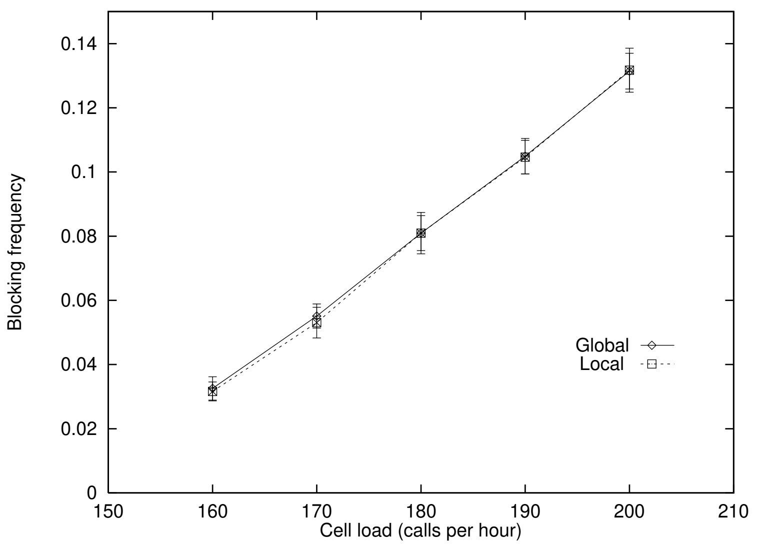

Figure 8: Global vs. local optimization

References

[1] R. Battiti, A.A. Bertossi, and M.A. Bonuccelli. Assigning codes in multihop packet radio networks: Heuristics and exact algorithms. *Preprint Università di Trento*, UTM 496, 1996.

[2] O. Bertazioli and L. Favalli. *GSM — Il Sistema Europeo di Comunicazione Mobile: Tecniche, Architettura e Procedure*. Advanced Technology Education Services. Hoepli Editore, Milano, 1996.

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

[4] M. Duque-Antón, D. Kunz, and B. Rüber. Channel assignment for cellular radio using simulated annealing. *IEEE Transactions on Vehicular Technology*, 42:14–21, 1993.

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

[6] I. Katzela and M. Naghshineh. Channel assignment schemes for cellular mobile telecommunication systems: A comprehensive survey. *IEEE Personal Communications*, pages 10–31, june 1996.

[7] E. Del Re, R. Fantacci, and L. Ronga. A dynamic channel allocation technique based on hopfield neural networks. *IEEE Transactions on Vehicular Technology*, 45:26–32, 1996.

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

[9] S. Singh and D. Bertsekas. Reinforcement learning for dynamic channel allocation in cellular telephone systems. *submitted to NIPS96*, 1996.

[10] K.L. Yung and T.P. Yum. Compact pattern based dynamic channel assignment for cellular mobile systems. *IEEE Transactions on Vehicular Technology*, 43:892–896, november 1994.