A Linear Downlink Power Control Algorithm for Wireless Networks

Yisroel Mirsky, Yoram Haddad
Jerusalem College of Technology, Computer Science and Networks Department, Jerusalem, Israel
ymirsky1@gmail.com, haddad@jct.ac.il

Abstract—**In order to optimize its capacity, a cellular radio system can use a power control algorithm to provide the best overall carrier-to-interference ratio to all of its links. Unfortunately, the optimum algorithm has an impractical exponential complexity of $O(2^n)$. However, an approach to the problem has been overlooked. By taking advantage of propagation effects it is possible to split up a large problem into overlapping smaller ones. Doing so can achieve virtually identical results to that of the optimum algorithm in $O(n)$ time (having a stable system). Moreover, this proposed algorithm is suitable as a distributed power control algorithm, whereas the optimum algorithm is a centralized one. This makes the proposed algorithm more suitable for today's cellular network architectures. Furthermore, it is also very easy to parallelize the proposed algorithm over multiple threads and cores offering a great added hardware advantage. In this paper we introduce this algorithm, prove its linear complexity and provide numerical results from simulations.**

I. INTRODUCTION

It is in the best interest of mobile operators to plan and maintain their networks to maximize their capacity. This fact is obvious even more so today since data rate demands and subscriber rates are growing exponentially [1]. Unfortunately, the capacity of a radio network is restricted by the quality of its links and the number of frequencies available to it. Frequency reuse has long been the accepted solution to the latter restriction. However, since this method uses the same frequency multiple times in the same region, co-channel interference occurs which affects the quality of the network's links. Unfortunately, this situation is unavoidable since the licensed frequencies that are available to cellular networks are expensive commodities. For this reason and others, different methods for minimizing the interference in mobile networks have been researched extensively [2]. One of these methods is to place a power control algorithm (PCA) in charge of monitoring the link gains and managing the transmission power levels accordingly.

There are two types of PCAs; centralized and decentralized (distributed) [3]. Centralized PCAs have a single control center and calculate the best transmission power levels for an entire network. This means that although a centralized PCA can offer excellent results, the algorithm must have access to information on all of the network's links in real-time. This poses a great difficulty for large networks due to the large number of control links required. Decentralized PCAs have independent distributed control centers which are each in charge of a small region. Each of these control centers calculate the best transmission power levels for their region based on the live status of their local network links and sometimes their neighbors' as well. While decentralized PCAs generally provide less accurate results [4], they are preferred today over centralized PCAs due to the fewer control links required and their real-time management speeds.

When it comes to the theory of centralized PCAs, it has been proposed in [5] and shown in [6] that a balanced CIR throughout all the link gains of a network substantially improves the overall network capacity. In other words, a network maximizes its overall capacity if every receiver in that network experiences the same CIR level. The transmission power assignment (for the entire network) which achieves this result can be viewed as the optimum solution for the network. However, in certain cases, a balanced CIR scheme may have negative results when one or more receivers experience a great deal of interference. Zander in [7] proposed a solution to this problem. With the usage of an instantaneous normalized link gain matrix, and the properties of positive matrices, he was able to find the optimum power levels needed to achieve this balanced CIR scheme. Then he developed a PCA which finds that optimum solution given a minimum system CIR, taking into account the problematic cells.

Although this brute force algorithm (BFA) which he developed provides optimum solutions, it has an impractical complexity of $O(2^n)$ (see section IV). For this reason, Zander proposed an approximation algorithm called the stepwise removal algorithm (SRA) in [7]. This algorithm has a polynomial complexity of $O(n^3)$ [8] taking into account the complexity of the eigenvalue problem [9]. Other researchers have studied Zander's findings and have also developed approximation algorithms which have polynomial complexities as well. Some of these algorithms include the Stepwise Maximum-Interference Removal Algorithm (SMIRA), the Stepwise Maximum Received Interference Removal Algorithm (SMRIRA), and the Stepwise Maximum Transmitted Interference Removal Algorithm (SMTIRA) which are described in [8] and [10].

Although most of the aforementioned approximation algorithms provide improved results to those generated by the SRA, they are all centralized PCAs (like the BFA) which are generally not used in today's networks.

In this paper, we propose a decentralized PCA which takes advantage of the optimality of the centralized PCA, the BFA, without its extensive run-time. The algorithm achieves this by exploiting the natural propagation of radio waves in a populated environment. It basically operates in a *merge sort* fashion; taking a large problem and splitting it into smaller overlapping ones, solving them with the BFA and then merging them back together. This merging brute force algorithm (MBFA) provides extremely accurate approximations to the BFA with a linear complexity. More importantly, due to the MBFA's modularity, it is a decentralized PCA which makes it suitable for today's network architectures, unlike the BFA or the other approximation algorithms mentioned above.

The remainder of the paper is organized as follows: in section II we present the model of our system with the notations needed to present the problem and the solution. In section III we define the optimization problem along with the algorithm used to solve it. In section IV analysis of the algorithm's complexity is provided. In section V we derive some numerical results from simulations and compare results. Finally, in section VI both the conclusion and future work are presented.

II. SYSTEM MODEL

Since this paper is based on Zander's work in [7], we extend his system model as follows.

In this paper we assume that we are dealing with a very large yet finite cellular network of N independent cells having altogether M downlink channels at their disposal. We also assume that each cell is covered by exactly one base station (BS) having an omnidirectional antenna located in the center of that cell. For the purpose of simplicity, we assume that there is no adjacent channel interference and neglect the naturally occurring background noise. Therefore, we assume our system to be an interference limited system. Since we are interested in the co-channel interference, we will only focus on a system of BSs transmitting over the same channel denoted by u, where $u \in U$ and U is the set of all M downlink channels.

We model the CIR measured by the receiving user equipment (UE) in a particular cell i as [7,10]

$$CIR_i(p_i) = \frac{\frac{p_i}{d_{ii}^{\propto}}}{\sum_{j=1}^{N} \frac{p_j}{d_{ij}^{\propto}} - \frac{p_i}{d_{ii}^{\propto}}} \tag{1}$$

where p_j is the transmission power used by the BS in cell j Let P be the vector containing the transmission power of each BS such that

$$P = \begin{bmatrix} p_1 \\ \vdots \\ p_N \end{bmatrix}. \tag{2}$$

The link gain is modeled as

$$L_{ij} = \frac{A_{ij}}{d_{ij}^{\propto}} \tag{3}$$

where A_{ij} is the attenuation factor, d_{ij} is the distance between the UE in cell i and the BS in cell j, and $\propto$ is the path loss exponent parameter (typically between 2 and 6).

Let $Z_{ij} = \frac{L_{ij}}{L_{ii}}$ where L_{ij} is the link gain between the UE i and the BS in cell j. Let $Z = [Z_{ij}]$ be the normalized downlink gain matrix. It is useful to view Z as having each row i indicate the perspective UE and each column j indicate the intruding cell to that UE. Let CIR_{min} be the network's minimum acceptable CIR threshold. As mentioned earlier, a network maximizes its capacity when each of its links experience the same CIR. Zander proved that the optimum CIR is reached when CIR is equal to CIR^* which can be derived as

$$CIR^* = \frac{1}{\lambda^* - 1} \tag{4}$$

where λ^* is the largest (in magnitude) real eigenvalue of the network's normalized downlink gain matrix Z for a particular channel u. It should be noted that λ^* is computed from an instantaneously sampled Z. The optimum power allocation to each BS achieving CIR^* is given by the eigenvector P^* associated with λ^*. In other words, when P^* is used as the transmission power vector P of the system, then all mobiles in the network described by Z, will experience the same CIR^*.

Let T be a collection of cells using the same channel u. We shall assume that these cells are geographically arranged as a staggered square grid (illustrated in Fig. 1). Let Z_T be T's normalized downlink gain matrix at some instance in time. Let S_k $(k \in \mathbb{N})$ describe the partition of T into geographically square and disjoint clusters $(s_v \in S_k)$. We assume that the square clusters are homogenous in their sizes such that the constant k describes the dimensions of these clusters in terms of cells ($|s_v| = k \times k$). For example, $s_v \in S_2$ is a square arrangement of four cells (each side having two).

Let a_{v_x} be the square ring of cells surrounding a particular s_v with a thickness of x cells $(x > 0)$ (illustrated in Fig. 1). Let e_v ("e" stands for *extension*) be defined as $s_v \cup a_{v_x}$, i.e. the extension of s_v with its ring a_{v_x}. Let E_{k_x} be the set of all extensions of S_k defined as:

$$E_{k_x} = \{e_v | \forall\, v \in [1, |S_k|]\}. \tag{5}$$

Note that if $e_i, e_j \in E_{k_x}$ and e_i and e_j are adjacent, then $e_i \cap e_j \neq \emptyset$ since $x > 0$.

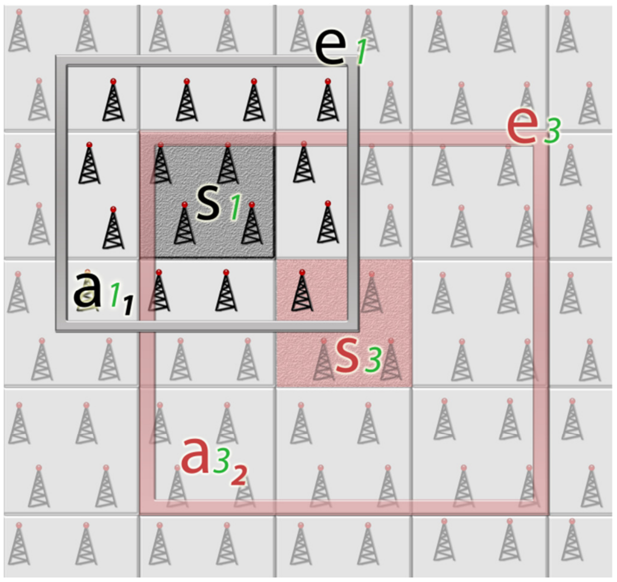

Figure 1. The illustration of $e_1 \in E_{2_1}$ and $e_3 \in E_{2_2}$ such that e_1 and e_3 are expanded-partitions of the partitions $s_1, s_3 \in S_2$ with different widths. a_{1_1} and a_{3_1} are outer rings of s_1 and s_3 such that $e_v - s_v = a_{v_x}$ respectively.

III. DECENTRALIZED POWER CONTROL ALGORITHM

In this section the optimization problem is formalized. Then the difficulties of finding the optimum solution are discussed. Finally, two algorithms for finding the optimum solution are described.

The CIR optimization problem which Zander solved by computing λ^* can be described as follows:

$$\begin{array}{lll} \textit{maximize} & \sum_i CIR_i(p_i) & (6.1) \\ \textit{subject to} & CIR_i(p_i) = CIR_j(p_i) \textit{ and } CIR_i(p_i) \geq CIR_{min}\ \forall i,j & \\ \textit{variables} & p & i,j \in \{1,2,3 \ldots |T|\} \end{array}$$

As mentioned earlier, the steps to finding P^* from Z provides a fast and analytical solution to (6.1). However, in the case where the constraint $CIR_i(p_i) \geq CIR_{min}$ is impossible to achieve, then the cells which are generating too much interference will have to be excluded from the system. This forms a new optimization problem:

$$\begin{array}{lll} \textit{maximize} & \sum_i CIR_i(p_i) - |T'| & (6.2) \\ \textit{subject to} & CIR_i(p_i) = CIR_j(p_i) \textit{ and } CIR_i(p_i) \geq CIR_{min}\ \forall i,j & \\ \textit{variables} & p, T' & i,j \in \{1,2,3 \ldots |T'|\} \end{array}$$

Here in (6.2) the objective is the same as it was in (6.1); to find the optimum power vector P^* for the system. However, the difference is that in (6.2) we intend to satisfy the CIR_{min} constraint when it fails. In order to do that we must find the smallest selection of cells T' from T such that their exclusion from the problem will maximize the objective function (6.1) while upholding the constraint $CIR_i(p_i) \geq CIR_{min}$. Furthermore, the T' chosen to be excluded should offer the largest CIR amongst all possible selections of that size ($|T'|$). This is because if a number of cells must be excluded from the system then the most effective combination of that number of cells should be removed. The reason $|T'|$ is being minimized in (6.2) is because it is undesirable to drop any cells from an existing system.

It is possible to simplify the process of solving the problem in (6.2) by reducing (6.2) to (6.1). The way to perform this reduction is by satisfying the problematic constraint. If there is no issue with the constraint $CIR_i(p_i) \geq CIR_{min}$ in (6.2) then the problem is equivalent to (6.1) and can be solved easily by following Zander's method (calculating P^* via λ^* from Z_T). Therefore, when it comes to computing the solution to (6.2), we must first find the optimum T' to exclude which will make the constraint feasible. Afterwards Zander's method can be applied to acquire a quick analytical solution.

The focus of this paper is to present an algorithm which will efficiently and effectively find the optimum T' to exclude from the problem in (6.2). This is because finding the optimum set of cells T' to be excluded is the most computationally difficult part of solving (6.2). Therefore, let us define C^* to be the optimum solution used to help solve (6.2), where C^* is the T' which if excluded from the system reduces the problem to (6.1). We shall also define C to be an approximated solution to the optimum solution C^*.

The BFA can be run in order to find C^*, however its run-time is unacceptable. A better solution proposed in this paper is the MBFA, since it is a linearly complex approximation algorithm which yields accurate results. In order to explain how the MBFA finds C, we will first explain how the BFA finds C^* since the BFA is used in the MBFA.

A. Brute Force Algorithm (BFA)

First the BFA receives Z_T and the system's CIR_{min}. Then the BFA finds CIR^* from Z_T. If CIR^* is larger than or equal to the CIR_{min}, then the BFA returns an empty C^* and ends. Else, the BFA finds the single cell, which by dropping it, achieves the largest CIR^*. If this new CIR^* is greater than the CIR_{min} then the BFA returns that selected cell as C^* and ends. Otherwise, the BFA tries again by finding the two cells, which by dropping them, achieves the largest CIR^*, and then performs the same check as before. The BFA continues this way until it surpasses the CIR_{min} or otherwise fails.

It is already clear why the BFA has a complexity of $O(2^n)$, however we will go into this in greater detail later in section IV

Below, the BFA is described in pseudo-code followed with a detailed explanation. This pseudo-code has not been presented in [7]. It should be noted that each column in Z_T describes a cell in the system. Therefore, cells are referred to by their corresponding column index values within Z_T. Furthermore, cells are removed from Z_T by removing their respective rows and columns from the original Z_T.

For the rest of the paper, we shall use arrays indexed with a value of 1. Below, pseudo-code for the BFA is presented, followed by a detailed explanation.

```
$BFA\ (Z_T, CIR_{min})$
1  $MaxAchvbleCIR^* \leftarrow 0$
2  $C^* \leftarrow \emptyset$
3  $numCell \leftarrow dim(Z_T)$
4  for $dropnum \leftarrow 0$ to $numCell$ do
5      $dropIndexes \leftarrow CplmntCmbNum(numBTS, dropnum)$
6      for each $indexvector\ v$ in $dropIndexs$ do
7          $m \leftarrow FindMaxCIR^*(SubMatrix(Z_T, v))$
8          if $m > MaxAchvbleCIR^*$ then
9              $MaxAchvbleCIR^* \leftarrow m$
10             $C^* \leftarrow v$
11     if $MaxAchvbleCIR^* > CIR_{min}$
12         return $C^*$
13 return $\{-1\}$
```

Figure 2. Pseudo-code for the BFA

In line 3, the function $dim(A)$ takes a $n \times n$ matrix and returns the value n. In line 5, the function $CplmntCmbNum(i,j)$ takes in two numbers$i,j \in \mathbb{N}$, finds all possible unique combinations of the numbers 1 through i with a set size of j and then returns the complement of the result. In other words, the function returns $\binom{i}{j}$ results, each of which are the indicated indexes to drop (and then test). In line 7 $FindMaxCIR^*(Z_T)$, takes in a normalized co-channel downlink gain matrix and computes the CIR^* according to the definition (4). The function in line 7 $SubMatrix(A,v)$, takes in a square matrix A and returns a square sub matrix A' such that A' equals A but with the rows and columns $i \in v$ dropped. The value returned on line 13 indicates that there is no solution for the given parameters.

B. Merging Brute Force Algorithm (MBFA)

The MBFA takes advantage of the fact that there is an exponential loss of signal power over distance. At some distance d_{ij}, the ratio $\frac{L_{ij}}{L_{ii}}$ becomes so small that it becomes insignificant to consider it in the matrix Z_T. Due to this property, cells which are not close to a perspective cell hardly interfere with that at all. Therefore, we are able to ignore all cells which are distant to those within some cluster $s_v \in S_k$. This allows us to run the BFA on one small s_v at a time and then merge the results. This is opposed to naively running the BFA on the entire set of co-channel cells T at once.

However, a question arises about the cells bordering the edges of each cluster in S_k. Zander pointed out in [**5**] that in all finite networks there will be boundary effects, slightly favoring boundary cells. This is because they neighbor fewer cells and therefore interfere with fewer cells than those which are fully surrounded. This means that favored cells will almost never be dropped in the BFA since they affect other cells' CIR levels the least. In our case this effect would cripple our accuracies since the sum of all the bordering cells throughout all the clusters is much larger than those of a single system.

The way the MBFA avoids this problem is by using the extended collection E_{k_x} as opposed to S_k. If there is a cell which is not fully surrounded then it won't be computed accurately. To avoid this phenomenon, every cell in every cluster s_v must be completely surrounded by other cells. The collection of extended clusters E_{k_x} insures this by surrounding every cluster with a ring of existing cells. This results in giving the cells we care about (s_v) fair consideration, while ignoring the added cells (a_{v_x}) which are not part of that sub-problem (see Fig. 1 for a visualization). It should be noted that the added ring of cells a_{v_x} causes each extended cluster $e_v \in E_{k_x}$ to overlap its neighboring extended clusters.

The MBFA finds the collection of cells C (an approximation to the optimum C^*) which, if excluded, reduces (6.2) to (6.1). It achieves this by first obtaining the C^*s from each of the $e_v \in E_{k_x}$ by using sub-matrices from the original Z_T. Afterwards, it merges all of the obtained C^*s to form a final solution C. By following this process, the MBFA receives a collection of cells which, if excluded, gives virtually identical results compared to those of the BFA used on the entire Z_T. Furthermore, the MBFA has a linear time complexity; unlike the BFA. Below, pseudo-code for the MBFA is presented, followed by a detailed explanation.

```
$MBFA(Z_T, E_{k_x}, CIR_{min})$
1  $N \longleftarrow \emptyset$
2  $FinalDrpIndxs \longleftarrow \emptyset$
3  for each element $e_v$ in $E_{k_x}$ do
4          $N \longleftarrow \{ N \cup BFA(CplmntSubMatrix(Z_T, e_v), CIR_{min}) \}$
5  $C \longleftarrow SubMatrix(Z_T, Unique(N))$
6  return $C$
```

Figure 3. Pseudo-code for the MBFA

In line 4 the function $CplmntSubMatrix(A,v)$ operates like the function $SubMatrix(A,v)$ except that it complements the selection in vector v first. In line 5, the function $Unique(V)$ takes in a collection of vectors and returns a single sorted vector with unique elements. Should the function come across a negative value, it will return a vector containing a single negative value to indicate that a solution was not found. The set N in the MBFA is essentially the collection of all the drop indexes indicated by the BFA on each of the subgroups of E_{k_x}.

The inputs of the MBFA are: a Z_T, the collection E_{k_x} (described by the column indexes in Z_T) and the CIR_{min}. It should be noted that E_{k_x} is a predetermined assignment for the network which does not change (like a configuration), and that all of the widths x are uniform. It is recommended that this assignment be limited to the inequality:

$$k + 2x \leq 7 \tag{7}$$

for stable systems and

$$k + 2x \leq 5 \tag{8}$$

for unstable systems. The reasons are computational and are explained later in section IV.

It is clear that not every assignment of k and x will provide accurate results. The assignment of these constants must be tested on the scenario before implementation. We found that the assignment E_{5_1} yielded excellent results in a stable system as will be further investigated in section V.

The MBFA can be a centralized PCA if the Z_T it uses is that of the entire network. If the network is geographically separated into overlapping clusters, and if each cluster is managed by its own controller, then the MBFA operates as a decentralized PCA. This is accomplished when the MBFA controller of each region communicates strictly with its neighboring region's MBFA controllers to determine whether an overlapping cell in its overlapping region should be dropped. For a visual reference, see Fig. 4.

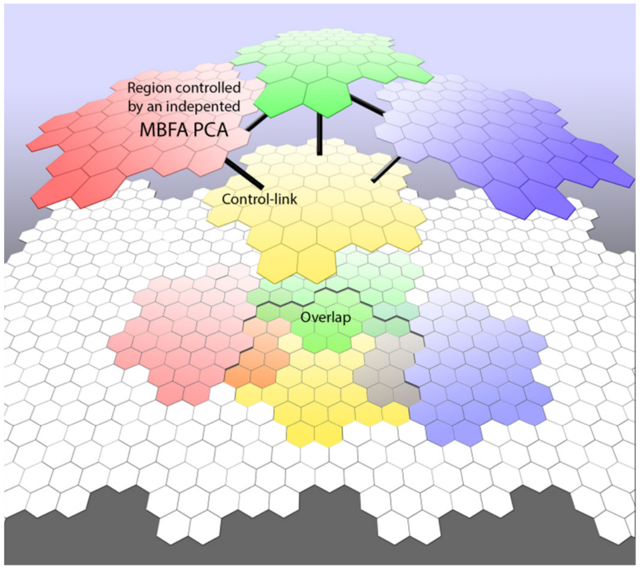

Figure 4. The MBFA operating as a decentralized PCA. Each region (containing a number of co-channel BSs) is identified by a color and overlaps other regions. Since the regions overlap each other, they must talk to their neighbors when deciding upon dropping shared cells.

IV. ALGORITHM COMPLEXITY ANALYSIS

In this section we analyze the run-time complexity of both the BFA and MBFA.

It is not hard to see why the BFA has an exponential complexity of $O(2^n)$. In lines 4-12 the BFA attempts to find the highest achievable CIR for the system while dropping the fewest number of cells as possible. In the worst case scenario, the algorithm has to run the $FindMaxCIR^*$ function

$$\sum_{i=0}^{n} n \, \mathrm{C} \, i = 2^n \tag{9}$$

times, where $n = |T|$.

Therefore, strictly due to the combinatoric properties of the BFA, running it on a large system is highly impractical. However, in [11] it states that running a complex algorithm with small inputs can be completely satisfactory. This is how the MBFA remains in the category of linear complexity.

Before analyzing the MBFA, it is crucial to point out that the assignment of the expanded clusters E_{k_x} should be limited to the inequality (7) if the system is stable or (8) if the system is unstable. Let γ be a selected constant based on the computing power of the CPU running the MBFA. A stable system is defined as one that statistically drops between $0 - \gamma$ cells and an unstable system is defined as one that statistically drops more than γ cells. The reason for the constraints (7, 8) is to limit the maximum number of iterations of the BFA's lines 4-12 to a safe constant value. If these recommended inequalities are violated, then it will likely result in undesirably long run-time computations for each cluster.

In the worst case scenario the MBFA in Fig. 3 repeats line 4- $\left\lceil \frac{n}{k^2} \right\rceil$ times where $k^2 = |s_v|$, which stands for the total number of different clusters in the considered system. Furthermore, line 4 has a maximum of $2^{C_1} \cdot {C_1}^{C_2} \cdot C_3$ computations, where C_1 is the number of cells left at any given iteration, C_2 is the polynomial power for solving an eigen-system and C_3 is the number of cells to be dropped. Since $C_1, C_2, C_3 < |e_i|$, where $e_i \in E_{k_x}$, and since all C_i are constants, line 4 has a maximum of a constant C_0 computations independent to the total number of cells in the system $|T|$. Therefore, the complexity of lines 1-4 is

$$O\left(C_0 \cdot \left\lceil \frac{n}{k^2} \right\rceil\right) = O(n), \tag{10}$$

where n is the total number of cells in the system.

Line 5 of the MBFA runs the $Unique$ function which has to sort the sets of drop indexes gathered from each of the overlapping subgroups in order to remove any duplicates. This has a run-time of $O\big(m\log(m)\big)$ where m is the size of the vector to be sorted. This would mean the total run-time of the MBFA is $O(m\log(m) + n)$. However, the typical size of the vector to be sorted m is significantly less than the number of the same co-channel cells n in that system (when dealing with stable systems). Therefore the run-time for MBFA in a stable system is $O(n)$ but for an unstable system is $O\big(m\log(m)\big)$. The unstable system's complexity is still not as exponential as the regular BFA; however it is no longer linear.

It should be noted that a very useful property of the MBFA is its ability to be parallelized. Since each iteration of line 4 is completely independent from all the others, it is easy to run this algorithm on multiple threads and cores. This gives the MBFA a great advantage over the polynomial complex PCA algorithms; such as the SRA, SMIRA, SMTIRA, SMRIRA and of course the exponentially complex BFA.

V. NUMERICAL RESULTS

For the simulations in this section, we used the scenario depicted in Fig. 5. The system was a large system containing 225 co-channel cells. The Z_T matrices were generated using a Monte Carlo approach as each respective UE was placed randomly each time within a 400 meter radius of its respective cell. There were no UEs placed in-between cells since such UEs, in reality, would have been attached to other BSs operating on different frequencies. Since the MBFA is a co-channel optimization algorithm, other cells operating with different channels were ignored in the simulation. In addition, we took $\propto = 3$ and $A_{ij} = 1$. We also decided on the assignment $E_{5,1}$ since the results proved to be the most accurate for our stable system. Since the BFA provides optimum results, we shall compare all results to it as a measure of accuracy.

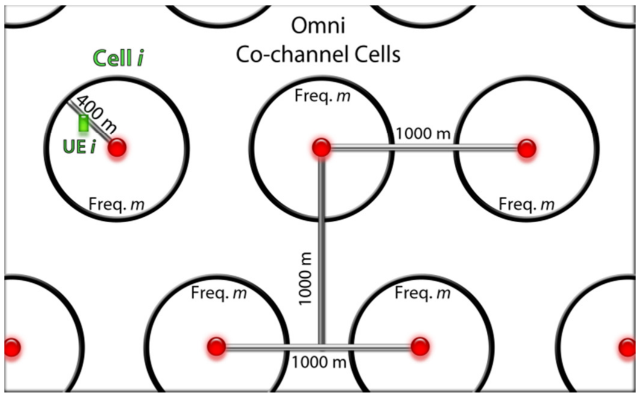

Figure 5. The scenario used in the simulations, showing the *co-channels* with their distance from one another, and the random UE placement near its respective cell.

Our simulation randomly generated a Z_T and then tested the same five CIR_{min}s on it using the BFA, MBFA, SRA, SMIRA, SMRIRA and SMTIRA. This was repeated 2341 times. All simulations were run with MATLAB® and the results found in Table 1 were run on a single 2.7 GHz CPU with less than 1 GB of memory available.

TABLE 1
SIMULATION RESULTS

	BFA (optimum)	MBFA
Avrg. CIR^* [dB] $(\overline{\boldsymbol{CIR}}^*)$	5.6954	5.6577
Std. Dev. of CIR^* $(\boldsymbol{CIR^*\sigma})$	0.6079	0.6289
Avrg. Cells Dropped $(\overline{\boldsymbol{C}})$	0.3457	0.3107
Std. Dev. of C $(\boldsymbol{C\sigma})$	0.5964	0.5978
Avrg. Time Elapsed [sec] $(\overline{\boldsymbol{t}})$	297.6218	0.3526
Std. Dev. of t [sec] $(\boldsymbol{t\sigma})$	3492.4135	0.7131

The results in Table 1 and Figures 6 and 7 certainly speak for themselves; however, it is important to analyze them. According to the definition of an optimum solution (the BFA) there should be as few cells dropped as possible in order to exceed the CIR_{min} of the system. Should there be any need to drop a number of cells then the particular selection of those cells should provide the maximum CIR possible. Therefore, it is worthwhile to note that the percentage of average cells dropped by the MBFA relative to the BFA is

$$\frac{|\overline{C}_{BFA} - \overline{C}_{MBFA}|}{\overline{C}_{BFA}} = \frac{0.035}{0.3457} \approx 10\% \qquad (11)$$

where $\overline{C}$ is the average number of cells dropped and the standard deviation of the cell-drops relative to the BFA is

$$\frac{|C\sigma_{BFA} - C\sigma_{MBFA}|}{C\sigma_{BFA}} = \frac{0.0014}{0.5964} \approx 0.002\%, \qquad (12)$$

where $C\sigma$ is the standard deviation of the dropped cells. Additionally, we can rate the accuracy of the MBFA further by noting that the average achieved CIR^* by MBFA relative to BFA is

$$\frac{\overline{CIR^*}_{MBFA}}{\overline{CIR^*}_{BFA}} = \frac{5.6577}{5.6954} \approx 99.34\%\,, \qquad (13)$$

having an error percentage of

$$\frac{|\overline{CIR^*}_{BFA} - \overline{CIR^*}_{MBFA}|}{\overline{CIR^*}_{MBFA}} = \frac{0.0377}{5.6954} \approx 0.007\%, \qquad (14)$$

where $\overline{CIR}^*$ is the average CIR^* received. The percentage (13) with its error percentage (14) clearly indicates the high accuracy of the MBFA's results. Lastly, it is worthwhile to note that in Table 1 the BFA has an average run-time of 297.6218 seconds with an enormous standard deviation of 3492.4135 (58 min.) while the MBFA has a small average run-time of 0.3526 seconds with a standard deviation 0.7131. This was expected due to their complexities outlined in section IV.

Figures 6 and 7 show the MBFA's accuracy compared to the other approximation algorithms mentioned in section I. It is important to note again that although the MBFA is a decentralized PCA, it gets relatively the same level of accuracy as the centralized PCAs: SMIRA, SMRIRA and SMTIRA.

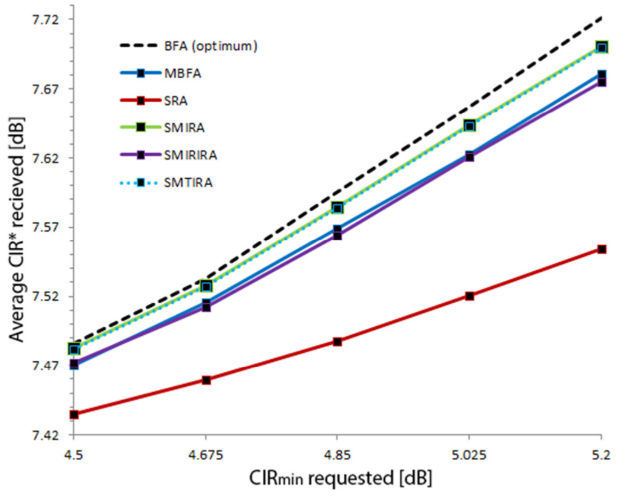

Figure 6. The CIR^* averages of each of the five CIR_{min} values collected from the simulation.

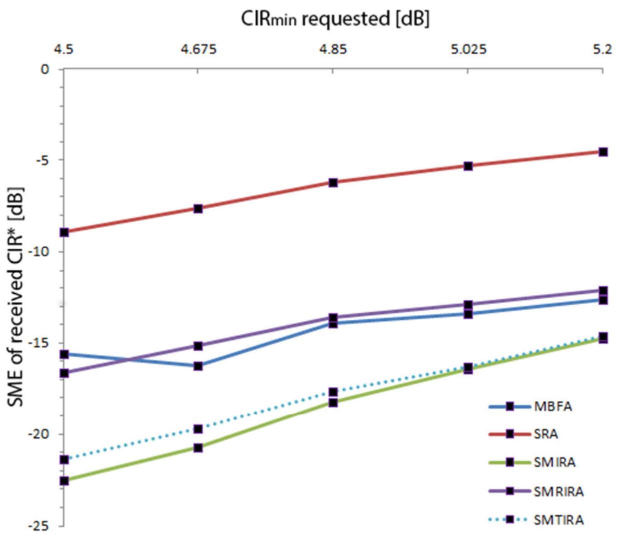

Figure 7. The squared-mean-errors of each algorithm versus the results from the BFA (the optimum solution) collected from the simulation.

VI. CONCLUSION

There exists a great interest to expand the capacity of the cellular network today. However, in order to do so, an accurate and efficient PCA must be implemented in order to guarantee the best possible CIR across a large network in real time. While a centralized PCA can offer the best results, it is impractical in today's large networks due to the sheer number of control links required. Therefore, it is preferred to use a decentralized PCA which can approximate to the optimum solution, as close as possible, without knowledge of the entire network. An example of such an algorithm is the merging brute force algorithm (MBFA) presented in this paper. The MBFA, in certain scenarios, can provide surprisingly high accuracy even though it is an approximation algorithm to the optimum algorithm; the BFA (a centralized PCA). Accuracies of 99.34% are achievable when the algorithm is configured correctly for each particular scenario. Further research is required to find out how these different arrangements of the overlapping clusters E_{k_x} affect the results. The MBFA also has a run-time complexity of $O(n)$ with stable systems and $O(m\log(m))$ with unstable systems. This is a great improvement to the $O(n^3)$ complexity attributed to the SRA and others. It should also be noted that the MBFA is also an excellent tool for radio network planning as well (due to its accuracy and linearity). Lastly, the MBFA has a lot of room for modifications, therefore further research and development of the MBFA and its application to other fields of science can prove to be most beneficial.

ACKNOWLEDGEMENT

Dr. Yoram Haddad acknowledges the support of ESF-COST Action IntelliCIS.

REFERENCES

[1] Cisco, Cisco Visual Networking Index: Global Mobile Data, 2011, Traffic Forecast Update 2010-2015.

[2] Dimitrie C. Popescu and Christopher Rose, Interference Avoidance Methods for Wireless Systems, 1st ed.: Springer, 2004.

[3] L. Won-Yeol and M. C. Vuran, S. Mohanty, “A survey on spectrum management in cognitive radio networks”, IEEE Communications Magazine, Vol. 46, Issue 4, pp. 40-48, April 2008.

[4] L. Qian, X. Li, D. R. Vaman and Z. Gajic, “Joint Power Control and Proportional Fair Scheduling with Minimum Rate Constraints in Cluster Based MANET”, in MSN 2006 Proceedings, pp. 197-208, 2006.

[5] J. M. Aein, "Power balancing in systems employing frequency reuse," COMSAT Tech. Rev., vol. 3, no. 2, Fall 1973.

[6] R. W. Nettleton and H. Alavi, "Downstream power control for spread spectrum cellular mobile radio system," in Globecom '82, Miami, pp. 84-88, 1982.

[7] Jens Zander, "Performance of Optimum Transmitter Power," IEEE Transactions on Vehicular Technology , vol. 41, no. 1, pp. 57-62, February 1992.

[8] H. Jin and Wenbin Jiang, Handbook of Research on Developments and Trends in Wireless Sensor Networks: From Principle to Practice, 1st ed.: Information Science Reference, 2010.

[9] V. Y. Pan and Z. Q. Chen, "The complexity of the matrix eigen problem," in STOC '99, pp. 507-516, 1999.

[10] L. Tsern-Huei, L. Jen-Cheng, and Yu T. S., "Downlink Power Control Algorithms for Cellular Radio Systems," IEEE Transactions on Vehicular Technology, vol. 44, no. 1, pp. 89-94, February 1995.

[11] T. H. Cormen, C. E. Leiserson, L., R. Rivest, and Stein C., Introduction to Algorithms, 2nd ed.: MIT Press, 2001.