Particle Swarm and Quantum Particle Swarm Optimization Applied to DS/CDMA Multiuser Detection in Flat Rayleigh Channels

Leonardo D. de Oliveira,
Fernando Ciriaco and Taufik Abrão
DEEL-UEL State University of de Londrina
Londrina, PR, 86051-990, Brazil
Email: daguileo@yahoo.com.br, taufik@uel.br

Paul Jean E. Jeszensky
Departamento de Engenharia de Telecomunicações e Controle
Escola Politécnica of University of São Paulo (LCS-PTC-EPUSP)
São Paulo, 05508-900, Brazil,
Email: pjj@lcs.poli.usp.br

***Abstract*— The particle swarm and quantum particle swarm optimization (PSO and QPSO) techniques applied to Direct Sequence/Code Division Multiple Access systems (DS/CDMA) with multiuser detection (MuD) are analyzed, evaluated and compared. The swarm techniques efficiency when applied to the DS-CDMA MuD (PSO-MuD and QPSO-MuD) in Flat Rayleigh channels is compared through the trade-off between performance and computational complexity. With the same simulation scenario, the comparison is accomplished among the PSO-MuD, QPSO-MuD, genetic algorithm (GA) and evolutionary programming with cloning (EP-C) algorithms. The complexity of these four heuristics-MuD (Heur-MuD) is compared, expressing it through the number of computational operations necessary in order to reach the performance obtained through the maximum likelihood detector (ML).**

I. Introduction

This work analyzes two heuristic algorithms based on the swarm combinatorial optimization, originally proposed by Kennedy and Eberhart [1]. This technique is based in the individual-society interaction which learning suffers influence from the group behavior/learning as well from the individual. The swarm technique shows to be promising in its discrete version [2], [3], [4], [5], and consequently adequate for the MuD problem treatment.

Trying to optimize the PSO-MuD and QPSO-MuD input parameters the analysis will characterize the algorithms performances in flat Rayleigh channels as a function of some of its parameters. The MuD problem is treated under the perspective of search heuristic techniques applying, besides the swarm, two other evolutionary techniques, the GA-MuD [6] and EP-C-MuD [7].

II. System Model

With BPSK modulation and synchronous channel shared by K users, the i-th bit of received baseband DS/CDMA signal is given by:

$$r(t)=\sum_{k=1}^{K} A_k\cdot b_k(t)\cdot\zeta_k(t)\cdot s_k(t-iT_b)+n(t) \qquad (1)$$

where A_k is the signal amplitude, $b_k(t)$ the transmitted bit and $s_k(t)$ the spreading sequence contained in the interval $[0,T_b)$, where T_b is the information bit period; the additive white Gaussian noise (AWGN), $n(t)$, has zero mean and variance $\sigma_n^2=\frac{N_0}{2}$, where N_0 is the unilateral power spectral density; the spreading sequence for the k-th user is defined by:

$$s_k(t)=\sum_{n=0}^{N-1} p_T(t-nT_c)c_{k,n} \qquad (2)$$

where $p_T(\cdot)$ is the rectangular shaping pulse and $c_{k,n}\in[\pm 1]$ is the chip sequence with duration T_c. Considering short codes the processing gain coincides with the sequence length, i. e., $N=\frac{T_b}{T_c}$.

The channel attenuation and distortion will be represented by a flat Rayleigh channel, characterizing a non-light of sight (NLOS) communication defined by [8]:

$$\zeta_k(t)=\gamma_k(t)\cdot e^{j\theta_k(t)} \qquad (3)$$

where the complex coefficient envelope, $\gamma_k(\cdot)$, is a random variable (RV) with Rayleigh distribution and the phase, $\theta_k(\cdot)$, is a uniform RV in the interval $[0;2\pi)$. Considering a slowly varying channel, i. e. $T_b << (\Delta t)_c$, where $(\Delta t)_c$ is the coherence time of the channel, $\gamma_k(t)$ and $\theta_k(t)$ are admitted constants during the bit interval T_b.

For a synchronous flat fading channel the k-th DS/CDMA signal at the matched filter to the k-th spreading sequence is:

$$y_k[i]=A_k b_k[i]\zeta_k[i]+\sum_{j\neq k}^{K} A_j b_j[i]\zeta_j[i]\lambda_{k,j}+n_k[i] \qquad (4)$$

where $n_k[i]$ is the i-th sample of filtered AWGN for the k-th user and $\lambda_{k,j}$ denotes the k,j-th element of the normalized correlation matrix $\mathbf{R}$ given by [9]:

$$\lambda_{k,j}=\frac{1}{N}\int_0^{T_b} s_k(t)\,s_j(t)dt, \qquad k,j=1,2,\ldots,K \qquad (5)$$

III. Heuristics MuD Detectors

The maximum likelihood (ML) detector is based in the maximum likelihood function in order to obtain the optimum detection (OMuD) [9], [10]. The ML simultaneous estimate for the detection of K synchronous users is given by:

$$\hat{\mathbf{b}}=\arg\left\{\max_{\mathbf{b}\in\{\pm1\}^K} 2Re\{\mathbf{y}^T\mathbf{C}^H\mathbf{A}\mathbf{b}\}-\mathbf{b}^T\mathbf{C}\mathbf{A}\mathbf{R}\mathbf{A}\mathbf{C}^H\mathbf{b}\right\} \qquad (6)$$

where $\mathbf{b} \in \{\pm 1\}^K$ are all possible column-vector, solutions for the MuD problem; $\mathbf{C}$ is a diagonal matrix with the channel complex coefficients; $\mathbf{A}$ is the diagonal matrix with the DS/CDMA signal amplitudes; and the vector $\mathbf{y}$ is the baseband received signal with dimension $K \times 1$. The solution vector $\hat{\mathbf{b}}$ will be one that maximizes (6).

The inconvenient of ML detector is the exponential increase of complexity with the number of users, $\mathcal{O}(2^K)$. Trying to improve the trade-off complexity $\times$ performance, heuristic algorithms minimize the computational cost of (6) reducing the search space for possible solutions but reaching a performance still close to the OMuD solution.

Heuristic algorithms are optimization methods based on the progressive approximation for a given problem. One objective function (cost) evaluates the possible solutions for the problem.

By brevity the EP-C and GA algorithms, both based on the Darwin evolutionary theory, are not described here. For details see [6], [7]. Next the PSO and QPSO algorithms are described. In the MuD context, particle (or individual) is a vector with bits belonging to the set of all possible solutions, being population (p) a set of particles present in a generation (the actual iteration in the search of optimization process); the number of particles in the initial population can be determined by [6], [11]:

$$p = 10 \cdot \left\lfloor 0.3454 \left(\sqrt{\pi(K-1)} + 2 \right) \right\rfloor \tag{7}$$

where the operator $\lfloor \cdot \rfloor$ returns the greatest integer not larger than the argument.

A. Swarm Algorithm

The algorithm is based on the movement of a group of particles, randomly distributed in the space, each one with an own position and velocity. This velocity is responsible by the movement imposed to the particle, changing its spatial location in a search of a better performance. The d-th dimension of the i-th particle position at time t is given by:

$$x_{id}(t) = x_{id}(t-1) + v_{id}(t-1) \tag{8}$$

where $v_{id}(t-1)$ is the i-th particle velocity in its d-th dimension and time $(t-1)$. For a kind of problems, like MuD, we should adapt the algorithm to a discrete model, adopting some probabilistic methods. Firstly, the number of dimensions should be considered as being the number of users of the system; each particle i is represented by a $K \times 1$ vector:

$$\mathbf{x}_i[t] = [x_{i1}[t] \; x_{i2}[t] \; \ldots \; x_{iK}[t]]^T, \quad i = 1, 2, ..., p \tag{9}$$

The interaction among particles, which is the base of the swarm algorithm, is inserted in the calculation of particles velocity. The velocity of the i-th particle, with dimension $K \times 1$, is defined by:

$$\begin{aligned} \mathbf{v}_i[t+1] &= \omega \cdot \mathbf{v}_i[t] + \phi_1 \cdot \mathbf{U}_{i_1}[t](\mathbf{x}_i^{best}[t] - \mathbf{x}_i[t]) + \\ &+ \phi_2 \cdot \mathbf{U}_{i_2}[t](\mathbf{x}_g^{best}[t] - \mathbf{x}_i[t]) \end{aligned} \tag{10}$$

where ω is the weight of the previous velocity in the present speed calculation; $\mathbf{U}_{i_1}[t]$ and $\mathbf{U}_{i_2}[t]$ are diagonal matrices with dimension K, and elements are RV modeled through uniform distribution $\mathcal{U}(0,1)$, generated for the i-th particle; $\mathbf{x}_g^{best}[t]$ is the $K \times 1$ vector with the best global position found until that iteration and $\mathbf{x}_i^{best}[t]$ is the best individual position for the i-th particle until that iteration, with length $K \times 1$; ϕ_1 and ϕ_2 are weight factors regarding the best individual position and the best global position influences in the velocity calculation, respectively.

For the MuD problem, each element x_{id} in (9) just assumes the "0" or "1" values. This implies in a discrete mode for the position choice. That is carried out inserting in the algorithm a command of choice, dependent of the velocity. However, the velocity needs to be adjusted to a probabilistic mode. Several functions possess this characteristic, being adopted here the sigmoid function:

$$S(a) = \frac{1}{1 + e^{-a}} \tag{11}$$

This function is limited in the interval $[0, 1]$. $S(a)$ tends to 0 when $a \to -\infty$ and to 1 when $a \to \infty$. The selection of the future particle position is obtained through the statement:

$$\text{If } S(v_{id}[t]) > \rho_{id}[t], \text{ then } x_{id}[t+1] = 1, \text{ else } x_{id}[t+1] = 0, \tag{12}$$

where $\rho_{id}[t]$ is a RV modeled through $\mathcal{U}(0,1)$.

Starting from the necessity of larger diversification for the search universe (in order to escape from a local maximum), a factor (V_{max}) is added to the algorithm model, which will be responsible for limiting the velocity in the range $[\pm V_{max}]$. This factor inserts in the velocity calculation a minimum probability that the bit change, making possible that the algorithm escapes from eventual local maximum and consequently the performance improvement occurs.

Table I shows this probability as a function of the V_{max} value. The bit change is more probable every time that the particle velocity crosses the limit established by $[\pm V_{max}]$.

TABLE I
MINIMUM PROBABILITY OF BIT CHANGE AS A FUNCTION OF V_{max}.

$V_{\max}$	1	2	3	4	5
$100[1 - S(V_{max})][\%]$	26.90	11.92	4.74	1.80	0.67

The steps for the PSO-MuD are:

1) Input variables definitions: $\omega, \phi_1, \phi_2, V_{max}$.
2) Generation of the initial p particles. The initial particle of the population is adopted as being the conventional detector (CD) output, and the others are generated in random mode.
3) Each particle (position) is evaluated through the cost function (6):

$$f(\mathbf{x}_i) = 2Re\{\mathbf{y}^T \mathbf{C}^H \mathbf{A} \mathbf{x}_i\} - \mathbf{x}_i^T \mathbf{CARAC}^H \mathbf{x}_i \tag{13}$$

if there are better positions, the best global performance is updated in $\mathbf{x}_g^{best}$ and the best performance of each particle in $\mathbf{x}_i^{best}$.

4) The velocity of each particle is calculated through equation (10) and the position of the particle is updated through (12).
5) Return to step 3 until to reach the previously established number of iterations.
6) The output vector is $\mathbf{x}_g^{best}$.

B. Quantum Swarm Algorithm

The QPSO was presented in [5] as an algorithm with good performance-complexity trade-off. Using the social and biological principles of the swarm algorithm, the QPSO is based on the behavior of group of various animals, associated to the quantum mechanics physical principle. In the quantum theory a *qubit* is defined as the smaller unity that holds information, assuming any value in the range $[0, 1]$. The i-th particle with quantum energy is defined by:

$$\mathbf{q}_i[t] \quad = \quad [q_{i1}[t] \;\; q_{i2}[t] \;\; \ldots \;\; q_{iK}[t]]^T \tag{14}$$

where K is the particle length (equal to the number of users for the MuD case), $i = 1, 2, \ldots p$ and $q_{id}[t] \in (0, 1)$.

For the MuD problem the term $q_{id}[t]$ means the probability of a bit to be “0”, and has to be transformed to a discrete form. In the original swarm algorithm a sigmoid function is used in order to adapt the velocity in the range $[0, 1]$. In the QPSO case the energies are already contained in this interval. Therefore, the discretized particle $\mathbf{x}_i[t] = [x_{i1}[t] \;\; x_{i2}[t] \;\; \ldots \;\; x_{iK}[t]]^T$ is obtained through the statement:

$$\text{If } \rho_{id}[t] > q_{id}[t], \text{ then } x_{id}[t+1] = 1, \text{ else } x_{id}[t+1] = 0, \tag{15}$$

Like the PSO the QPSO algorithm has memory in order to store the best position values already found for each particle ($\mathbf{x}_i^{best}[t]$) and the best global position ($\mathbf{x}_g^{best}[t]$). From these positions, the best global and individual quantum energy values are calculated in order to generate changes in the particle positions.

$$\mathbf{q}_g^{best}[t] = \alpha \cdot \mathbf{x}_g^{best}[t] + \beta \cdot (1 - \mathbf{x}_g^{best}[t]) \tag{16}$$

$$\mathbf{q}_i^{best}[t] = \alpha \cdot \mathbf{x}_i^{best}[t] + \beta \cdot (1 - \mathbf{x}_i^{best}[t]) \tag{17}$$

where the parameters α and β control the step for the $\mathbf{q}$ function, with $\alpha + \beta = 1.0$. The i-th particle energy is updated by:

$$\mathbf{q}_i[t+1] = c_1 \cdot \mathbf{q}_i[t] + c_2 \cdot \mathbf{q}_i^{best}[t] + c_3 \cdot \mathbf{q}_g^{best}[t] \tag{18}$$

where c_1, c_2 and c_3 represent the weight for each component of the energy, with $c_1 + c_2 + c_3 = 1.0$.

The algorithm can be implemented following the steps:

1) Input variables definitions: $\alpha, \beta, c_1, c_2, c_3$, with $\alpha, \beta, c_1, c_2, c_3 \in \mathbb{R}_+$.
2) Generation of the initial p particles, each i-th with energy $\mathbf{q}_i[t]$ generated in a random form and $\mathbf{x}_i[t]$ obtained through (15);
3) Evaluation of $\mathbf{x}_i[t]$ through the cost function (13). If there are better positions, the best individual and/or global positions ($\mathbf{x}_i^{best}[t]$ and $\mathbf{x}_g^{best}[t]$) will be updated.
4) Change the energy of each particle, $\mathbf{q}_i[t]$, according (18), following by discretization, through (15), in order to obtain $\mathbf{x}_i[t]$.
5) Return to step 3 until to reach the previously established number of iterations.
6) The output vector is $\mathbf{x}_g^{best}$.

IV. Numerical Results

Tests for the PSO-MuD and QPSO-MuD parameters optimization in the synchronous flat Rayleigh channels were carried out (not shown here), in the attempt to find the optimum values (or almost optimum). For all simulations spreading sequences with processing gain $N = 32$ was adopted.

For comparison purpose the conventional detector (CD) and the single user bound (SuB) were included [8].

A. PSO-MuD parameters optimization

$\underline{V_{max}}$: a slow convergence was observed for values $V_{max} < 3.5$. This delay is more significant increasing the number of the users and the E_b/N_0 values. For $V_{max} > 10$ a short delay at the end of convergence process was observed indicating a lack of diversity. Therefore, $V_{max} = 4$ was adopted.

$\underline{\phi_1}$: as the ϕ_1 value increases, occurs a slow start in the PSO-MuD algorithm, but the algorithm tends to reach a slightly superior performance; this gain is not expressive. The delay effect is more pronounced as the number of users increases. A compromise value of $\phi_1 = 2$ was adopted.

$\underline{\phi_2}$: the convergence is faster increasing ϕ_2, due to the intensification of the best global position search (as a consequence, smaller diversification of the space search); however, the BER performance improvement is insignificant for a large range of tested values for ϕ_2. In the subsequent simulations the value $\phi_2 = 10$ was adopted.

$\underline{\omega}$: smaller number of iterations and better performance were obtained with $\omega = 1$. An increase in ω implies in a reduction in the diversification, resulting in a low efficiency to escape from local maximums. A decrease in the ω value implies in slow convergence. The value $\omega = 1$ was adopted.

B. QPSO-MuD parameters optimization

$\underline{\alpha \text{ and } \beta}$: The algorithm has better performance for $\alpha < 0.1$. High values imply in slow convergence and low ones imply in convergence to non-optimal performance values. Thus, $\alpha = 0.05$ and $\beta = 0.95$.

$\underline{c_1,\ c_2, \text{ and } c_3}$: In a similar way as obtained for the PSO, the weight of best global position factor (c_3) is more important than the others. Low values for c_3 imply in slow convergence and high values result in lack of diversity. A good trade-off is obtained for the values $c_1 = c_2 = 0.2$ and $c_3 = 0.6$.

C. Convergence and BER for 4 Heur-MuD algorithms

System parameters:

- $E_b/N_0 \in [0; 30]dB$;
- Near-far effect (NFR) with perfect power control scenario ($NFR = 0$) or half of users with $NFR = 0dB$ and half with $NFR = +8dB$;

- PN spreading sequences with length $N = 32$;
- Number of users $K = 16$ (load $L = 0.5$) or $K = 32$ ($L = 1.0$);
- Number of errors in the Monte Carlo Simulation (MCS) equal or greater than 30.

Heur-MuD parameters:

- Population given by equation (7);
- PSO: $V_{max} = 4$, $\phi_1 = 2$, $\phi_2 = 10$ and $\omega = 1$;
- QPSO: $c_1 = c_2 = 0.2$, $c_3 = 0.6$, $\alpha = 0.05$ and $\beta = 0.95$;
- GA: $T = p/10$ (number of better selected particles), $i_c = 50\%$ and $p_m = 100/K\%$;
- EP-C: $i_{\%} = 10$ and $I_C = p\frac{i_{\%}}{100}$.

Fig. 1 shows the Heur-MuD, the conventional detector (CD) and the bound limit (SuB) mean performance considering $E_b/N_0 = 12dB$. The GA algorithm has a good start, reaching the OMuD performance (not shown in the figure) after only 9 iterations. The EP-C was shown to be slow at beginning but also reaching the same BER after 20 iterations. The PSO and QPSO have a initial delay in order to start the convergence but both reach the OMuD performance after 2 iterations of GA convergence, showing their efficiency in those conditions. This delay is due to the null initial velocity condition for the PSO and the random initial energy condition for the QPSO.

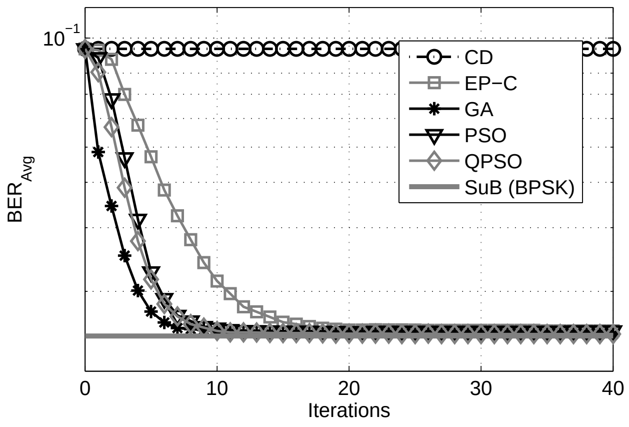

Fig. 1. Heur-MuD algorithms convergence; $L = 50\%$ and $E_b/N_0 = 12dB$.

Next, the Heur-MuD algorithms performances were analyzed with power disparities (8 users with $NFR = +8dB$). The average convergence curves for the 8 weaker users ($NFR = 0dB$), not shown here, have indicated that the two swarm algorithms are robust against the near far effect.

The performance at medium and high E_b/N_0 ($18dB$ and $25dB$) were also analyzed, considering $L = 50\%$. GA, PSO and QPSO keep their convergence. For these conditions the EP-C algorithm has a very low convergence, not shown here, indicating a weak diversification strategy.

With the same system conditions as in Fig. 1, except by increasing the number of users to $K = 32$, the Heur-MuD algorithms performance were analyzed. The results in Fig. 2 indicate that the performances are very similar to the $L = 50\%$ case, except for the increase of the iteration in that convergence happens.

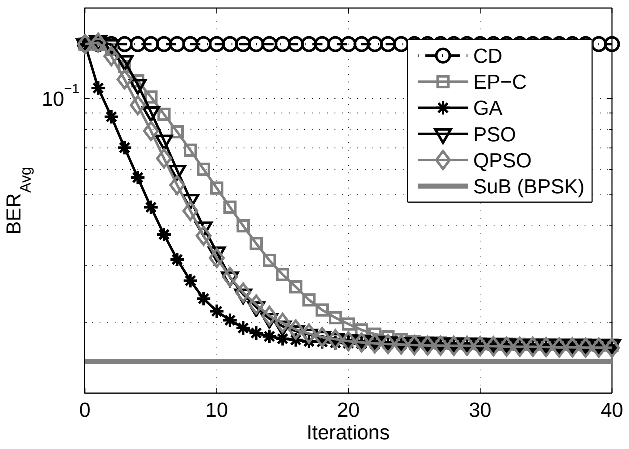

Fig. 2. Heur-MuD convergence, $L = 100\%$ and $E_b/N_0 = 12dB$.

The Heur-MuD performance degradation was also analyzed considering errors in the channel estimates. Errors were introduced separately and jointly being modeled through uniform distributions:

$$\widehat{\gamma}_k = \; \mathcal{U}\left(1 \pm \epsilon_\gamma\right) \times \gamma_k \;\; ; \;\; \widehat{\theta}_k = \; \mathcal{U}\left(1 \pm \epsilon_\theta\right) \times \theta_k$$

where ϵ_γ and ϵ_θ are the maximum module and phase channel coefficients errors, respectively. Figure 3 shows the BER of the four Heur-MuD algorithms, after convergence, as a function of E_b/N_0 for $K = 16$ users, with (10% and 25%) and without errors in the channel coefficients estimates. The four algorithms have similar behavior, once they optimize the same cost function. Without errors the performance is close to the SuB case. With errors the same degradation can be observed for the four algorithms. Note that phase errors cause more degradation than module errors.

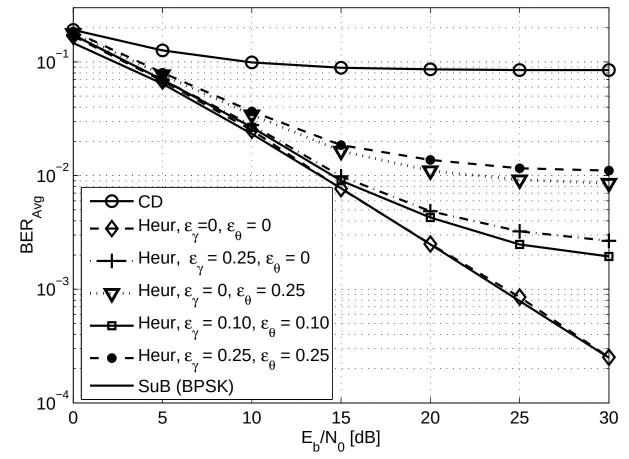

Fig. 3. Heur-MuD performance with errors in the channel coefficients estimates.

V. Computational Complexity

In spite of the fact that the four Heur-MuD algorithms have reached a similar performance, the number of iterations (g) in order to reach this point changes with E_b/N_0 and also with the particular algorithm in the form $g_{\text{GA}} < g_{\text{QPSO}} <$

$g_{\text{PSO}} < g_{\text{EP-C}}$. The total number of required operations defines the computational complexity through the computation of the number of multiplications, random number generations, comparisons and selections.

Note that the two terms of (13) can be calculated before the iterations loop:

$$\mathbf{f}_1 = 2\mathbf{y}^T\mathbf{C}^H\mathbf{A} \qquad \text{and} \qquad \mathbf{f}_2 = \mathbf{CARAC}$$

resulting in $4K^3 + 6K^2 + 3K$ operations and representing a marginal computational cost as K increases. Thus, cost function calculation can be obtained as:

$$f\left(\mathbf{x}_i\right) = Re\{\mathbf{f}_1\mathbf{x}_i\} - \mathbf{x}_i^T\mathbf{f}_2\mathbf{x}_i \tag{19}$$

These two factors result in a total of $pg(K^2+2K)$ operations.

OMuD. There are $2^K K$ generations of bits and $2^K\left(K^2+2K\right) + 4K^3 + 6K^2 + 3K$ products in the calculation of the cost function.

PSO-MuD. Random number generation occurs in the velocity calculation and discretization. The total generated numbers is given by $3pgK+pK$. There are $K(pg+p+2)$ transpositions. Also occur $pg(2K+2)+g+p$ comparisons, between the particle positions with the best global position, the best individual position and the velocities in each iteration with V_{max}; there are still $pg(K^2+6K)+K[4K^2+(6+p)K+2p+1]$ multiplications in order to compute the cost function, velocity and discretization.

QPSO-MuD. It should be considered $pg\left(K^2+6K\right) + K\left[4K^2+(6+p)K+3p+3g+1\right]$ multiplications for the cost function (19) and energy calculation. The number of comparisons is $2pg+p+g$, in the update of the best individual and global positions. The random number generation is a great advantage in confront to the PSO algorithm, being necessary only $pgK+pK$ generations in order to discretize the vector of bit energy. Also occur $K(pg+p+2)$ transpositions.

EP-C-MuD. For the EP-C algorithm the number of operations depends mainly on the operations of cost function [6], being necessary $K(pg+p-1)$ bit generations, $(pg+p)(K^2+2K)+4K(K^2+6K+3)$ operations for the computation of the cost function, $2Kpg$ ordinations, pgK/I_C selections, gI_C clonings and pgK comparisons.

GA-MuD. It is necessary $K(pg+p-1)$ bit generations, pgK selections, $(pg+p)(K^2+2K)+4K(K^2+6K+3)$ operations for the computation of the cost function, $2pgK$ ordinations and pgK comparisons.

Finally, the complexity analysis shown here is limited by the fact that some operations with distinct computational complexity (multiplication, selection, comparison) were considered with the same cost. Table II shows in literal and numerical forms (for the analyzed conditions) the number of required operations.

TABLE II

MUD COMPLEXITY IN TERMS OF THE NUMBER OF OPERATIONS

MuD	**Number of Operations**		
	Literal	16 users	32 users
OMuD	$2^K(K^2+3K)+K(4K^2+6K+3)$	2×10^7	4.8×10^{12}
EP-C	$pg\left(K^2+6K+K/I_C\right)$ $+K\left(4K^2+(6+p)K+4p+gI_C+2\right)$	242912	1557824
GA	$pg\left(K^2+7K\right)+$ $K\left(4K^2+(p+6)K+3p+2\right)$	126432	1180480
PSO	$pg\left(K^2+12K+2\right)+p+g+$ $K\left[4K^2+K(p+6)+(4p+3)\right]$	176109	1424254
QPSO	$pg\left(K^2+8K+2\right)+p+g+$ $K\left[4K^2+(p+6)K+(5p+3g+3)\right]$	155997	1315006
		$@E_bN_0 = 12dB$	

VI. CONCLUSIONS

The PSO-MuD and QPSO-MuD are promising algorithms. The parameters optimization in a synchronous flat Rayleigh channel shows that the algorithms reach very closely the optimum ML performance and are stable for many different system's operation point (load, near-far effect and E_b/N_0), considering the optimized parameters as constant. The two swarm algorithms are robust converging in a fast mode, even with high load condition, in spite of a slow initial convergence. The four Heur-MuD suffer similar performance degradation in the presence of errors in the channel coefficient estimates, being more sensitive to the phase errors.

The four Heur-MuD algorithms have a complexity of the same order, $\mathcal{O}(pgK^2)$, with the GA-MuD resulting the smaller number of operations for the two analyzed loading ($K=16$ and $K=32$ users).

For the MuD problem the swarm technique results in a good complexity $\times$ performance tradeoff, having the QPSO-MuD a smaller computational complexity than the PSO-MuD; however this complexity difference tends to be marginal as loading increases.

REFERENCES

[1] J. Kennedy and R. Eberhart, “Particle Swarm Optimization”, *IEEE International Conference on Neural Networks*, pp. 1942-1948, 1995.

[2] J. Kennedy and R. Eberhart, “A Discrete Binary Version of the Particle Swarm Algorithm”, *IEEE International Conference on Systems*, pp. 4104-4108, 1997.

[3] Z. Lu and S. Yan, “Multiuser Detector Based on Particle Swarm Algorithm”, *Proc. of the IEEE* 6^{th} *Circuits and Systems Symposium on Emerging Technologies*, vol.2, pp. 783-786, 2004.

[4] Y. Zhao and J. Zeng, “Particle Swarm Optimization Algorithm in Signal Detection and Blind Extraction”, 7^{th} *International Symposium on Parallel Architectures, Algorithms and Networks (ISPAN'04)*, 2004.

[5] S. Yang, M. Wang and L. Jiao, “A Quantum Particle Swarm Optimization”, *IEEE Congress on Evolutionary Computation, CEC'04*, vol. 1, pp. 320-324, June 2004.

[6] F. Ciriaco, T. Abrão and P. Jeszensky, “DS/CDMA Multiuser Detection with Evolutionary Algorithms”, *Journal of Universal Computer Science*, vol.12, no.4, pg. 450-480, May, 2006.

[7] T. Abrão, F. Ciriaco and P. J. Jeszensky. “Evolutionary Programming with Cloning and Adaptive Cost Function Applied To Multi-User DS-CDMA Systems”. *IEEE ISSSTA'04*, Sydney, Australia, pp. 160-163, Sep. 2004.

[8] J. Proakis. *Digital Communications*. McGraw-Hill, 1989.

[9] S. Verdú, *Multiuser Detection*, New York: Cambridge University Press, 1998.

[10] S. Verdu, “Optimum Multiuser Signal Detection”, PhD Thesis, *University of Illinois at Urbana, Champaign*,1984.

[11] C. W. Ahn and R. S. Ramakrishna, “A Genetic Algorithm for Shortest Path Routing Problem and the Sizing of Populations”, *IEEE Transactions on Evolutionary Computation*, vol. 6(6), pp. 566-578, 2002.