6

A Hybrid GA-TS Technique with Dynamic Operators and its Application to Channel Equalization and Fiber Tracking

Luis M. San José-Revuelta
University of Valladolid
Spain

1. Chapter description

This chapter is based on the author's research activities during the last decade on the fields of Evolutionary Computation (EC)-based techniques applied to digital communications and medical image processing. Specifically, the chapter is organized in three main sections:

- The first section (section 2) presents a concise introduction to metaheuristic EC-based strategies, mainly genetic algorithms (GA) and tabu search (TS) - for the sake of comparison, brief comments to simulated annealing (SA) are included, as well. Section 2.1 shows a general description of the standard GA while section 2.2 focuses on the basic TS algorithm.
- Next, section 3 develops the proposed hybrid GA-TS method. It begins with the description of a genetic algorithm with notable reduced complexity (known as a *micro genetic algorithm*, μGA) that uses a modification of the standard genetic operators in order to improve its convergence rate and computational load. Such features are achieved by on-line tuning up the probabilities of mutation and crossover by means of analysing the population individuals' fitness entropy. This way, a new method to control and adjust the diversity of the population is obtained. The μGA here described was partially developed in (San José, 2005). Once the GA is obtained, it is then modified and improved using the main distinctive concepts of TS. Specifically, we introduce a systematic use of memory in order to keep information on the last visited solutions as well as on the concrete parts of the chromosomes, or population's individuals, that have experimented alterations that have positively or negatively affected the fitness function. Besides, memory keeps track of the genes affected by the genetic operators and the tabu tenure depends on the explorative or exploitative sense of the search, which is estimated from the mean population fitness entropy previously described. This way, the TS main ideas will help to avoid both cycling and processing of non-interesting regions of the solutions' space. The hybrid algorithm thus developed will be denoted as "GA-TS".
- The last part of the chapter is devoted to the description of two application examples:
  (1) Application of the GA-TS algorithm to symbol detection in synchronous wireless communications. Numerical results will analyze the performance in comparison to traditional methods such as the matched filter detector, the minimum mean square

Source: Local Search Techniques: Focus on Tabu Search, Book edited by: Wassim Jaziri, ISBN 978-3-902613-34-9, pp. 278, October 2008, I-Tech, Vienna, Austria

error algorithm, the standard GA-based detector as well as some radial basis function (RBF)-based methods.
(2) Application of the developed GA-TS method to estimate the parameters of a recently developed algorithm (San José, 2007) for fiber tracking in diffusion tensor (DT) fields acquired via magnetic resonance imaging (MRI). This algorithm is successfully tested with both intricate synthetic images and real white matter DT-MR images. Numerical experiments will show the performance gain over previous approaches, especially with respect to convergence and computational load. Tracking of white matter fibers in the human brain becomes an issue of essential importance nowadays since it will improve the diagnosis and treatment of many neuronal diseases.

2. Introduction to metaheuristic EC-based strategies

2.1 Genetic algorithms

2.1.1 GA fundamentals

Genetic algorithms are a part of evolutionary computation (EC), which is a rapidly growing area of artificial intelligence. GAs are inspired by Darwin's theory about evolution. Simply said, these algorithms encode a potential solution to a specific problem on a simple data structure (known as *chromosome* or *individual*) and apply genetic operators to a selected group of the whole set of potential solutions (known as *population*) so as to preserve critical information. This is motivated by the hope that the new population will be better than the old one. Those chromosomes selected to form new individuals (*offspring*) are selected according to their *fitness* - the more suitable they are, the more chances they have to reproduce. Following the analogy with natural systems, the complete set of chromosomes is called *genome*. A particular set of genes in genome is called *genotype* (Mitchell, 1996).
Genetic algorithms are often viewed as function optimizers, although the range of problems to which they have been applied is quite broad. An implementation of a GA begins with a population of typically randomly generated chromosomes. These individuals are then evaluated and assigned reproductive opportunities so that those chromosomes representing a better solution to the target problem are given more chances to reproduce than those chromosomes which are poorer solutions.
Under this particular description of a GA, the term "genetic algorithm" has two meanings: In a strict interpretation, the GA refers to a model introduced and investigated by John Holland and his colleagues (Holland, 1975). Most of the existing theory for GAs applies to this model as well as variations on what is frequently referred to as the "standard genetic algorithm". In a broader usage of the term, a genetic algorithm is any population-based model that uses selection and recombination operators to generate new sample points in a multidimensional search space.
Considering an application oriented GA implementation the size of the search space is related to the number of bits used in the problem encoding, i.e. for a bit string encoding of length L, the size of the search space would be 2^L and it can be viewed as a hypercube. In this situation, the genetic algorithm can be considered as a method for sampling the corners of this L-dimensional hypercube. These type of problems are also known as "*NP-complete problems*", where *NP* stands for *nondeterministic polynomial* and it means that it is possible to "guess" the solution (by some nondeterministic algorithm) and then check it, both in polynomial time. A well-known example of NP problem is the *travelling salesman problem*.

Since the number of good solutions to a problem is sparse with respect to the size of the search space, then random search or search by enumeration becomes an impractical form of problem solving. Besides, any search other than random search imposes some bias in terms of how it looks for better solutions and where it looks in the search space. Though genetic algorithms indeed introduce a particular bias in terms of what new points in the space will be sampled, they make relatively few assumptions about the problem that is being solved. As a weak method, GAs are robust but very general. If there exists a good specialized optimization method for a specific problem, then genetic algorithm may not be the best optimization tool for that application. On the other hand, some researchers work with hybrid algorithms that combine existing methods with genetic algorithms. In this chapter we propose to enhance the performance of a particular GA (San José, 2005) with specific concepts representative of tabu search.
The theoretical concepts that explain how genetic algorithms work, i.e. how chromosomes evolve toward the optimum solution, are partially based on the *Schema Theorem*. For the sake of brevity, its description lies beyond the scope of this chapter.

2.1.2 GA cycle

In summary, a GA starts with a *population* or set of possible solutions represented by *chromosomes*. Solutions from one population are taken and used to form a new population. Solutions which are selected to form new solutions (*offspring*) are selected according to their fitness - the more suitable they are, the more chances they have to reproduce. This is repeated until some condition (for example number of populations or improvement of the best solution) is satisfied. Figure 1 outlines the main steps of the basic genetic algorithm.
This outline of basic GA is very general and there exist many things that can be implemented differently in various problems, for instance: how to create chromosomes, what type of encoding choose, how to define the two basic operators of GA (*crossover* and *mutation*), how to select parents for crossover, and many other implementation issues. Some of the concerning questions will be discussed in the specific applications later described.

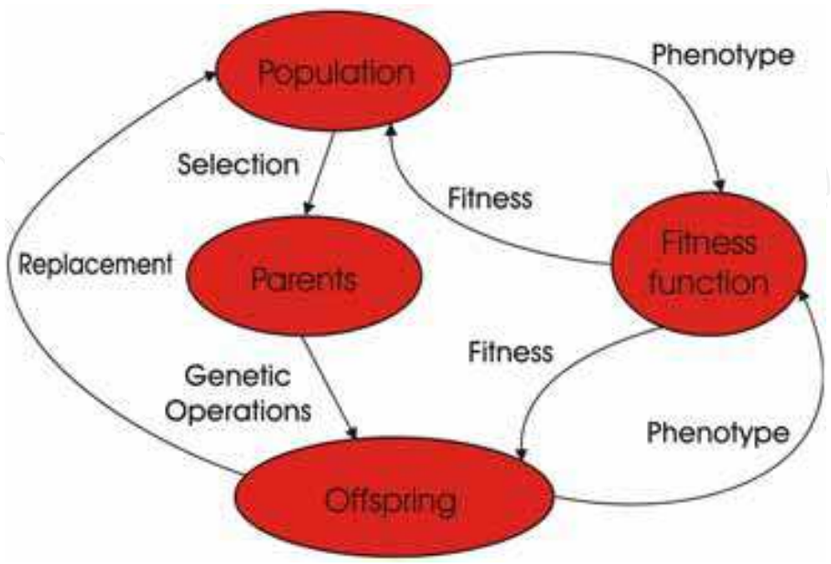


Fig. 1. Outline of the basic genetic algorithm.

2.1.3 Encoding and genetic operators

Crossover and mutation are the most important parts of a genetic algorithm. Performance is influenced by these two operators. However, the description of the chromosome encoding must be first commented.

The chromosome should contain information about the solution that it represents. The most used way of encoding is a binary string, for instance: $\mathbf{u}_1$=[1101100100110110], $\mathbf{u}_2$=[1101111000011110]. Each bit in this string can represent some characteristic of the solution or the whole string can represent a number. Other ways of encoding include directly integer or real numbers encoding.

Once encoding has been decided, the genetic operators (crossover and mutation) must be defined. Crossover selects genes from parent chromosomes and creates a new offspring. The simplest way to achieve this task is to choose randomly a few crossover points and interchange the genetic material separated by these points as illustrated in Figure 2.

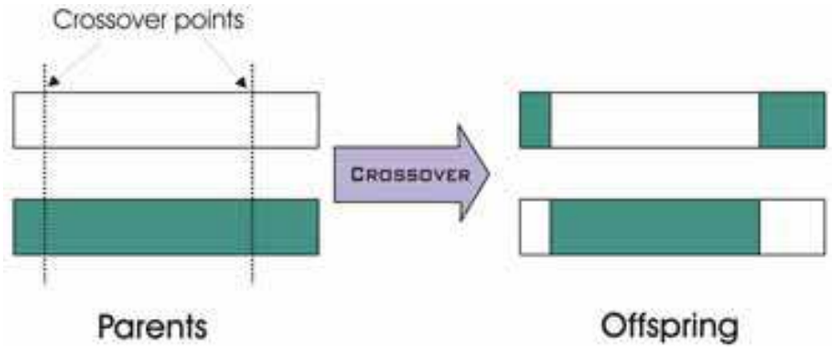


Fig. 2. Crossover example with two crossover points.

Notice that it is also possible to define a crossover operator with only one or even with multiple crossover points. Crossover can be rather complicated and depends on the encoding of the chromosome. Specific crossover made for a specific problem can improve performance of the genetic algorithm. For instance, in our applications, the *uniform crossover* and the *partially matched crossover* (PMX) operators will be used.

The crossover operator requires the definition of a selection procedure in order to select the parents (two chromosomes) from the population. According to Darwin's evolution theory, the best ones should survive and create new offspring. There are many methods for selecting the best chromosomes, for example roulette wheel selection, Boltzman selection, tournament selection, rank selection, steady state selection and some others. In our specific applications, the *roulette wheel selection* scheme will be used. Using this selection procedure, the better the chromosomes are, the more chances to be selected they have.

Once crossover is performed, mutation takes place. Mutation changes randomly the new offspring. For binary encoding we can switch a few randomly chosen bits from 1 to 0 or from 0 to 1. Obviously, both mutation and crossover depend on the encoding. An example of mutation is shown in Figure 3.

Consequently, there are two basic parameters of GA - the crossover probability and the mutation probability. Since we will propose a novel strategy for on-line adjusting these probabilities based on the entropy of the population individuals' fitness, a detailed description of these parameters as well as their influence on global convergence will be explained in sections 3.6-3.7.

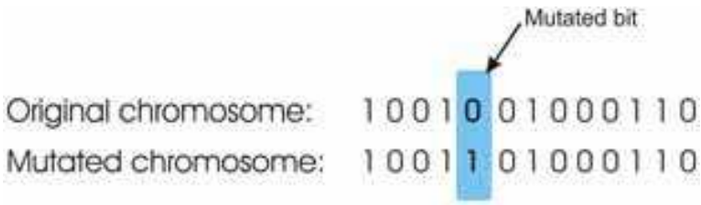


Fig. 3. Mutation example.

Thus, the GA works repeating a cycle: every iteration, those chromosomes with highest fitness are selected for creating a new offspring. Then, those ones with the lowest fitness are removed and the new offspring is placed in their place. The remaining chromosomes of the population survive to next generation.

2.1.4 Elitism and final remarks

Finally, we will comment the concept of *elitism* which is implemented in many GAs. This strategy was developed with the aim of avoiding the possibility of loosing the best chromosome in the current population when the genetic operators are applied. When elitism is used, the best chromosome (or a few best chromosomes) is directly copied into the new population. The rest is done in the same way. Elitism can very rapidly increase performance of GA, because it prevents losing the best found solution.

In summary, among the main advantages of GA we find: (i) their parallelism. GAs search the solutions' space with more individuals (and with genotype rather than phenotype) so they are less likely to get stuck in a suboptimal or local solution like some other methods. (ii) they are easy to implement. Once you have some GA, you just have to write the new chromosome (just one object) to solve a different problem. On the other hand, their main drawback is their computational load. If not properly designed, they can be slower than some other methods.

2.2 The Tabu search strategy

During the last decade, the tabu search (TS) technique has shown a remarkable efficiency on many problems. Tabu search was originally presented in its present form by Glover (Glover, 1986); the basic ideas have also been sketched by Hansen (Hansen, 1986). Additional efforts of formalization are reported in (Glover, 1989; de Werra & Hertz, 1989; Glover, 1990). Up to now, there is no formal explanation of this good behaviour, though, theoretical aspects of tabu search have been investigated (Faigle & Kern, 1992; Glover, 1992; Fox, 1993).

Tabu search belongs to the iterative techniques group, a type of optimization procedures. The general step of an iterative procedure consists in constructing from a current solution i a next solution j and in checking whether one should stop there or perform another iteration. Neighbourhood search methods are iterative procedures in which a neighbourhood $N(i)$ is defined for each feasible solution i, and the next solution j is searched among the solutions in $N(i)$. Simulated annealing (SA) and tabu search can be considered as neighbourhood search methods which are more elaborate than the classical descent method. The basic ingredients of tabu search are next described.

2.2.1 Fundamentals

In order to improve the efficiency of the search, it is necessary to keep track not only of local information (like the current value of the objective function) but also of some information

related to the exploration process. The use of *memory* is one of the essential features of TS. While most exploration methods keep in memory essentially the value $f(i^*)$ of the best solution i^* visited so far, TS also keeps information on the *itinerary* through the last solutions visited. Such information will be used to guide the next move. Memory restricts the possibilities to some subset of $N(i)$ by forbidding for instance moves to some neighbour solutions. Specifically, the structure of the neighbourhood $N(i)$ of a solution i will vary from iteration to iteration. This is why TS is included in a class of procedures called *dynamic neighbourhood search techniques.*

The formal description is as follows (Hertz, 1995): Let us consider an optimization problem in the following way: given a set S of feasible solutions and a function f: $S \rightarrow \Re$, find some solution i^* in S such that $f(i^*)$ is acceptable with respect to some criterion (or criteria). Generally a criterion of acceptability for a solution i^* would be to have $f(i^*) \leq f(i)$ for every I in S. In this situation, TS would be an exact minimization algorithm provided the exploration process would guarantee that after a finite number of steps such an i^* would be reached.

However, in most contexts, no guarantee can be given that such an i^* will be obtained. Therefore, TS could simply be viewed as a general heuristic method. Since TS includes in its own operating rules some heuristic techniques, it would be more appropriate to characterize TS as a *metaheuristic*. Frequently, its role will be to guide and to orient the search of another (more local) search procedure. As a first step towards the description of TS, the classical descent method can be formulated as follows:

Step 1. Choose an initial solution i in S.

Step 2. Generate a subset V^* of solution in neighbourhood $N(i)$.

Step 3. Find the best j in V* ($f(j) \leq f(k)\ \forall k \in V^*$) and set $i=j$.

Step 4. If $f(j) \geq f(i)$ then stop. Else go to Step 2.

A descent method would generally take $V^*=N(i)$. Since this approach may often be too time-consuming, an appropriate choice of V^* may be a notable improvement.

The opposite case would be to take $|V^*|=1$; this would drop the phase of choice of a best j. Simulated annealing could be integrated within such a framework. A solution j would be accepted if $f(j) \leq f(i)$, otherwise it would be accepted with a certain probability depending upon the values of f at i and j as well as upon a parameter called *temperature* (there is no temperature in TS). Memory is used to choose V^* so as to exploit knowledge extending beyond the function f and the neighbourhood $N(i)$.

Descent procedures often get trapped in a local minimum. So any iterative exploration process should in some instances accept also non-improving moves from i to j in V^* (i.e. with $f(j) > f(i)$) if one would like to escape from a local minimum. Simulated annealing does this also, but it does not guide the choice of j, TS in contrast chooses a best j in V^*. However, this strategy increases the risk of cycling and re-visiting solutions. This is the point where memory is helpful to forbid moves which might lead to recently visited solutions. This way, the structure of $N(i)$ will depend upon the itinerary and hence upon the iteration k, so we may refer to $N(i,k)$ instead of $N(i)$. Taking this idea into account, the descent algorithm can be reformulated in a way which will bring it closer to the general TS procedure. It can be stated as follows (Hertz, 1995):

Step 1. Choose an initial solution i in S. Set $i^*=i$ and $k=0$.

Step 2. Set $k=k+1$ and generate a subset V^* of solution in $N(i,k)$

Step 3.Choose a best j in V^* (with respect to f or to some modified function $\tilde{f}$ and set $i=j$.

Step 4. If $f(i) < f(i^*)$ then set $i^*=i$.

Step 5. If a stopping condition is met, then stop. Else, go to Step 2.

Notice that we may consider the use of a modified $\tilde{f}$ instead of f in some circumstances. Some of the stopping conditions are the following: (i) $N(i,k+1)$ is empty, (ii) k is larger than the maximum number of iterations allowed, (iii) the number of iterations since the last improvement of i^* is larger than a specified threshold, and (iv) there exists evidence than an optimum solution has been obtained.

The definition of $N(i,k)$ at each iteration k and the choice of V^* are crucial for the final search result. The first one implies that some recently visited solutions are removed from $N(i)$. They are considered as *tabu solutions* which should be avoided in the next iteration. Such a memory based on recency will partially prevent cycling. Specifically, keeping at iteration k a list T (tabu list) of the last $|T|$ solutions visited will prevent cycles of size at most $|T|$. In such a case $N(i,k)=N(i)-T$. Since this list T may be extremely impractical to use, we will describe the exploration process in terms of moves from one solution to the next. For each solution i in S, we define $M(i)$ as the set of moves which can be applied to i in order to obtain a new solution j (notation: $j=i\oplus m$). Then $N(i)=\{j \;/\; \exists\; m\in M(i)$ with $j=i\oplus m\}$. So, instead of keeping a list T of the last $|T|$ solutions visited, we may simply keep track of the last $|T|$ moves. Obviously, this restriction losses information and that it does not guarantee that no cycle of length at most $|T|$ will occur. For efficiency purposes, it may be convenient to use several lists T_r at a time. Then some constituents t_r (of i or of m) will be given a tabu status to indicate that these constituents are currently not allowed. Generally, the tabu status of a move is a function of the tabu status of its constituents which may change at each iteration. A collection of tabu conditions can be formulated as

$$t_r(i,m) \in T_r(r = 1,...,t). \tag{1}$$

A move m will be a tabu move if *all* conditions are satisfied. A drawback of the replacement of solutions by moves is that unvisited solutions can be given a tabu status. We will then overrule the tabu status when some tabu solutions will look attractive. This is performed by means of *aspiration level conditions*.

A tabu move m applied to a current solution i may appear attractive because it gives, for example, a solution better than the best found so far. We would like to accept m in spite of its status; we shall do so if it has an *aspiration level* $a(i,m)$ which is better than a threshold value $A(i,m)$. Parameter $A(i,m)$ can be viewed as a set of preferred values for a function $a(i,m)$. Thus, conditions of aspiration can be written in the form

$$a_r(i,m) \in A_r(i,m)\ (r=1,...,a). \tag{2}$$

If at least one of these conditions is satisfied, then m will be accepted.

Finally, in some situations, it can be of interest to replace the process f by another function $\tilde{f}$ so as to introduce some intensification and diversification of the search. In the search process it is sometimes fruitful to intensify the search in some region of S because it may contain some acceptable solutions (*exploitative* search). Such intensification can be carried out by giving a high priority to the solutions which have common features with the current

solution. This can be done with the introduction of an additional term in the objective function that penalizes solutions far from the present one. This should be done during a few iterations and after this it may be useful to explore another region of S. On the other hand, diversification (*explorative* search) can be forced by introducing an additional term in the objective function in order to penalize solutions that are close to the present one. The modified objective function is the function $\tilde{f}$ that was mentioned earlier in the algorithm:

$\tilde{f} = f + Intensification + Diversification$.

Gathering together all these concepts, the TS procedure can be finally stated as follows (Hertz, 1995):

Step 1. Choose an initial solution i in S. Set $i^*=i$ and $k=0$.

Step 2. Set $k=k+1$ and generate a subset V^* of solution in $N(i,k)$ such that either one of the tabu conditions $t_r(i,m) \in T_r$ is violated ($r=1,...,t$) or at least one of the aspiration conditions $a_r(i,m) \in A_r(i,m)$ holds ($r=1,...,a$).

Step 3. Choose best $j=i \oplus m$ in V^* (with respect to f or to the function $\tilde{f}$) and set $i=j$.

Step 4. If $f(i) < f(i^*)$ then set $i^*=i$.

Step 5. Update tabu and aspiration conditions.

Step 6. If a stopping condition is met, then stop. Else, go to Step 2.

3. Proposed hybrid GA-TS algorithm

This section describes a hybrid GA-TS algorithm whose main novelty comes from its very low computational load (similar to a μGA), the introduction of an on-line procedure to adjust the GA's parameters in order to achieve and maintain a good population diversity, and the incorporation of memory concepts proper of TS that improve convergence speed and diversity, avoiding cycling and reducing computational load. This diversity is a key issue in the performance of any evolutionary algorithm, including GAs and TS methods (Ursem, 2002). For this reason, we first briefly comment the importance of diversity in relation to convergence properties, and, afterwards, we will present the specific GA-TS structure together with the diversity control algorithm.

3.1 Diversity and suboptimal convergence in evolutionary computation

As mentioned in section 2, both genetic algorithms and tabu search constitute robust global search and optimization strategies that can strike an attractive balance between the *exploitation* - vertical search in the proximities of an specified point or area in the solutions space - and the *exploration* - horizontal search throughout all the totality of the solutions space, allowing the test of new potential solutions - of possible problem solutions. However, simple GAs have a tendency to converge prematurely to local optima, mainly due to selection pressure and too high gene flow between population members (Ursem, 2002). First, a high selection pressure will fill the population with clones of the best fit individuals, since they have the highest survival probability. Diversity declines after a short while, and, because the population consists of similar individuals, the algorithm will have difficulties escaping the local optimum. Nonetheless, lowering the selection pressure will often lead to an unacceptable slow convergence speed. On the other hand, high gene flow is often determined by the population structure. In simple GAs any individual can mate with any

other individual. Therefore, genes spread fast throughout the population and the diversity drops quickly with fitness stagnation as a prevalent outcome.
All these facts point out the key role of maintaining a suitable diversity in the population in order to appropriately converge to the optimal solution, thus avoiding, the inefficient presence of duplicated or very similar individuals, as well as the possibility of getting trapped into suboptimal solutions. Some authors have already dealt with the problem of diversity monitoring and control: For instance, the Diversity-Control-Oriented GA (Shimodaira, 1999) calculates a survival probability by means of a diversity measure based on the Hamming distance between the individual and the current best individual. Hence, diversity is preserved through the selection procedure. Another approach is the Shifting-Balance GA (Oppacher, 1999), where a containment factor between two subpopulations - based on Hamming distances between all members of the two populations - determines the ratio between individuals selected on fitness and individuals selected to increase the distance between the two populations. A third, and more distantly related, approach is the Forking GA (Tsutsui, 1997), which uses specialized diversity measures to turn a subset of the population into a subpopulation. Two variants of the Forking GA exist. The first one operates on the genotype, whereas the second type bases the division on distances in the search space (on the phenotype). More recently, Ghosh et al. (Ghosh, 2003) proposed to vary the mutation probability in n equally-length intervals along the n_g iterations of the algorithm, while the probability of crossover was kept constant (see Fig. 3 in (Ghosh, 2003), p. 863, for an example). The authors remark that this scheme increases the diversity of the population when the probability of mutation is increased and, conversely, as the optimal string is approached, the probability of mutation is reduced.

3.2 Fundamentals and encoding

The principle of GAs consists in representing a set of potential solutions (*population*) with a predetermined encoding rule. Each potential solution (*chromosome*) is associated to a figure of merit, or *fitness* value, in accordance to its proximity to the optimal solution, i.e., each chromosome is evaluated for its fitness in solving a given optimization task. When no prior knowledge of the solution is available, this initial set of potential solutions, $P[0]$, which forms the whole population for each new signaling interval (iteration k=0), is randomly generated. We will denote $P[k] = \{\mathbf{u}_i\}_{i=0}^{n_p}$ to the population at iteration k, with n_p being the number of individuals $\mathbf{u}_i$ per generation. For instance, the specific encoding scheme used for the first application later considered is explained in section 4.3.

3.3 Genetic operators

Once individuals are generated and given a fitness value, the next step consists in applying the *genetic operators*, mainly *mutation* and *crossover*. The first one modifies specific individuals with probability p_m, changing (antipodal bit) the value of some concrete position/s in the encoding of $\mathbf{u}_i$. Both the position and the new value are randomly generated - see Fig. 2. A low level of mutation serves to prevent any element in the chromosome from remaining fixed to a single value in the entire population. However, a high level of mutation will essentially result in a random search. Hence, the value of p_m must be chosen carefully in order to avoid excessive mutation. To maintain a satisfactory balance between such extremes a good initial value for p_m in our applications is between 0.01 - 0.05.

Note that mutation promotes exploration by creating an opportunity to explore different areas of the solution space. A good element can even turn into a bad one. This is definitely undesirable, especially if the offspring, which constitutes the actual global optimum, is mutated to a suboptimum solution. Hence, the value of p_m must be carefully chosen in order to prevent excessive mutation.

On the other hand, the crossover operator requires two operands (*parents*) to produce two new individuals (*descendants* or *offspring*). These new individuals are created when merging parents by crossing them at specific internal points - see Fig. 3. This operation is performed as follows: parent individuals $\mathbf{u}_i$ and $\mathbf{u}_j$ are exchanged using the *uniform crossover* process (Mitchell, 1996) in order to produce two offspring individuals. The process of uniform crossover uses a so-called *crossover mask*, which is a sequence of randomly generated 0's and 1's. The elements in $\mathbf{u}_i$ and $\mathbf{u}_j$ are exchanged at bit locations corresponding to a "1" in the crossover mask with probability p_c.

3.4 Elitism and termination criteria

Since parents are much more likely to be selected from those individuals having a higher fitness, the small variations introduced within these individuals are intended to also generate high fit individuals. Using Markov chain modelling, it has been proved that Gas are guaranteed to asymptotically converge to the global optimum - with any choice of the initial population - if an elitist strategy is used, where at least the best chromosome at each generation is always maintained in the population (Bhandari, 1996). However, Bhandari et al. (Bhandari, 1996) provided the proof that no finite stopping time can guarantee the optimal solution, though, in practice, the GA process must terminate after a finite number of iterations with a high probability that the process has achieved the global optimal solution.

In the proposed GA-TS algorithm, the elite E_k for $P[k+1]$ is formed by selecting those individuals from both the elite of $P[k]$ and the mutated elite of $P[k]$ having the highest objective value in the population. The mutation of the elite is performed with a probability $p_{m,e}$ considerably smaller than p_m (so as to avoid the destruction of good solution guesses). In our simulations we used $p_{m,e} = x\ p_m$, with x=0.2 (heuristical trials show little differences for values of x in the range [0.1,0.5]). No crossover is performed on the elite, since it implies, in most of cases, abandoning the exploitation of these good potential solutions.

This procedure, whose flowchart is shown in Fig. 4, is iterated a predefined number of consecutive generations, n_g. At the end, the string $\mathbf{u}_i$ corresponding to the best fit individual is finally chosen as the problem solution.

3.5 Diversity and entropy-dependent genetic operators

Standard GAs suffer from an excessive computational load: the application of the genetic operators is often costly and the evaluation of fitness can also be a very time-consuming task. Populations sizes n_p normally are 100, 200 or even much higher - for instance, (Uursem, 2002) uses populations with 400 individuals.

Hence, we propose an algorithm works with much smaller population sizes (in the order of 15 to 35 individuals). An elite of 2 to 5 individuals is kept and the crossover and mutation probabilities depend on the Shannon entropy of the population (excluding the elite) fitness which is calculated as

$$H(P[k]) = -\sum_{i=1}^{n_p} \lambda_i\ (k) \log \lambda_i\ (k) \tag{3}$$

with λ_i being the normalized fitness of individual $\mathbf{u}_i$. When all the fitness values are very similar, with small dispersion, $H(P[k])$ becomes high and p_c is decreased (it is not worthwhile wasting time merging very similar individuals). This way, the exploration character of the search is boosted, while, conversely, exploitation decreases. On the other hand, when this entropy is small, there exists a high diversity within the population, a fact that can be exploited in order to increase the horizontal sense of search. Following a similar reasoning, the probability of mutation is increased when the entropy is high, so as to augment the diversity of the population and escape from local suboptimal solutions (exploration decreases, exploitation becomes higher).

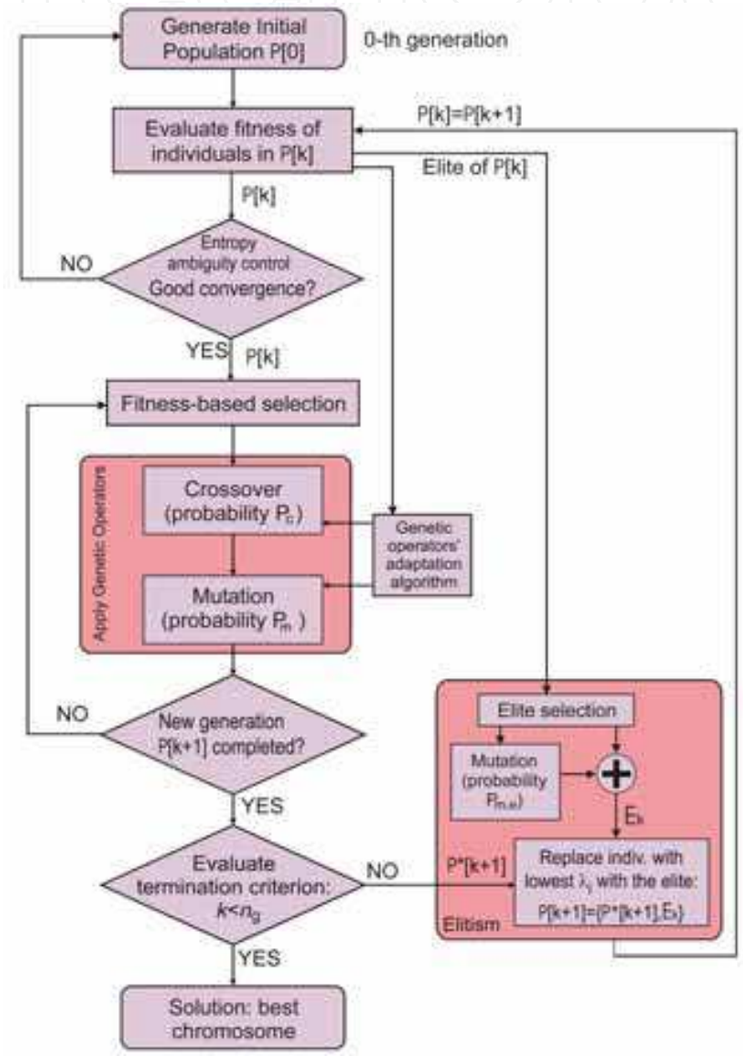


Fig. 4. Schematic representation of the GA structure and the genetic operators.

Therefore, we have that probabilities p_m and p_c are directly/inverselly proportional to the population fitness entropy, respectively[1],

[1] Symbol "α " denotes proportionality.

$$p_c(k) \propto [H(P[k])]^{-1} \tag{4}$$

$$p_m(k) \propto H(P[k]) \tag{5}$$

Some exponentially dependence on time k is also included in the model - making use of exponential functions - in order to relax (decrease), along time, the degree of dependence of the genetic operators' probabilities with the dispersion measure[2]. This avoids abandoning good solution estimates when very noisy sporadic samples are received or when the whole population has converged to the global optimum.
Hence, the proposed algorithm uses a much smaller population size than standard GAs, calculates crossover with a very low probability (and only on individuals not belonging to the elite), and the exploration/exploitation sense of the search is on-line tuned up by measuring the diversity of the population by means of its fitness entropy.
Next section describes how the diversity of the population affects the genetic operators' probabilities.

3.6 Genetic operators and convergence cycle

A typical estimation convergence process is depicted in Fig. 5. Notice that this is just one - the most frequently found and representative - of the possible situations, which will greatly depend on the randomly generated initial population $P[0]$ and the specific application. This is also a mere schematic representation and the vertical axis should have a different scale for each represented parameter.

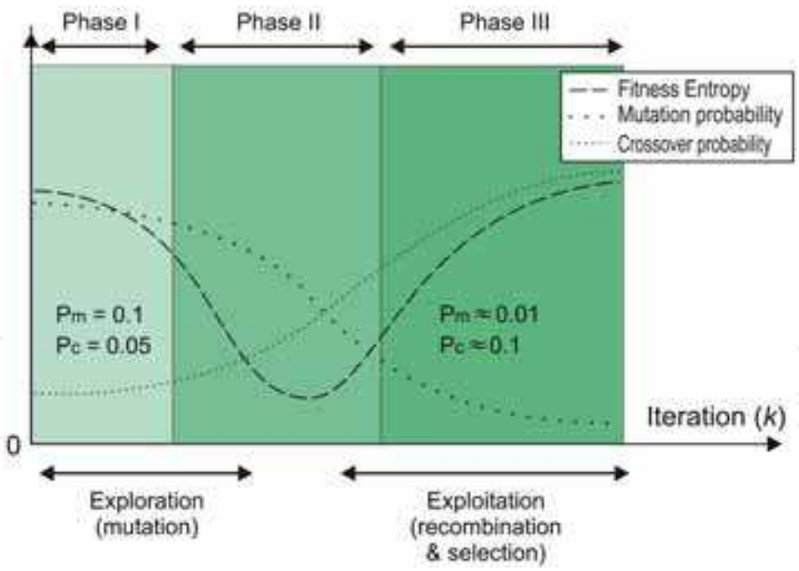


Fig. 5. Schematic representation of the alternating exploring and exploiting behavior in the diversity-guided GA-TS. The proportionality between different curves is not represented; only the increasing/decreasing tendency of each parameter is representative.

[2] See, also, complementary discussion on the form of these two functions, p_c and p_m, in section 3.7 *Genetic operators and convergence cycle*.

Three different phases can be appreciated:

(I) Population $P[0]$ has been randomly generated. Individuals $\mathbf{u}_i$ share a very few bits I common. Their fitness values are very similar since all of them are, probably, wrong solution estimates and, thus, the entropy is high. The probability of mutation is kept high so as to explore new potential solutions and the probability of crossover is small.

(II) Some individuals begin to converge to good solution estimates. Their fitness increase while the remaining individuals of the population continue presenting low values. The entropy of the fitness considerably decreases. The probability of mutation is decreased gradually while the one associated to crossover becomes higher so as to exploit the genetic information of the good estimates.

(III) Most of the population individuals have converged to good solutions. The fitness of all the individuals is very close (all of them tend to be equiprobable solutions). The entropy is again high and the exploitative behavior prevails over the explorative one.

As an example, since concrete values greatly depend on the randomly generated initial population, Fig. 5 also shows the approximate range in which p_c and p_m evolve along the execution of a satisfactory-convergence typical cycle: $p_m(I)=0.1 \rightarrow p_m(III)\approx 0.01$ and $p_c(I) = 0.05 \rightarrow p_c(III)\approx 0.1$.

Keeping these three different phases in mind, the determination of the form of functions p_c and p_m in Eqs. (4) and (5) is achieved as follows:

- **Probability of mutation**: during phase I, where most of the individuals are wrong estimates and the entropy is high, p_c is required to be high to as to increase the explorative sense of the search. Thus, p_c is directly proportional to the entropy H and to another function, which is denoted as g_m, and has the form $g_m(k) = \exp(-\beta_m k)$, which, for low values of k and $\beta_m < 1$, is close to 1. Thus, we can write

$$p_m(k) = \xi_m \cdot g_m(k) \cdot H(P[k]) \tag{6}$$

  where ξ_m stands for a normalization parameter, which ensures that $p_m \in [0,1]$, and $g_m(k) = \exp(-\beta_m k)$ with $0 < \beta_m < 1$.

  Once the GA-TS algorithm has converged to some good solution estimates, H becomes high again (see Phase III in Fig. 5), but now p_m must be lowered so as to switch from exploration to the exploitation of new individuals. This is achieved selecting g_m such that its decreasing character prevails over the entropy values. In our particular system, exponential functions $g_m(k) = \exp(-\beta_m k)$, with $\beta_m \in [0.1 , 0.05]$, were used.

- **Probability of crossover**: in this case, p_c should maintain low values in Phase I in order to allow an explorative search. In this phase H is high and p_c is chosen to be inversely proportional:

$$p_c(k) = \frac{\xi_c g_c(k)}{H(P[k])} \tag{7}$$

  If function g_c has the form of an inverse exponential (logarithmic) function, durin Phase I, the product of g_c and the inverse of H adopts low values (see Phase I in Fig. 5), while in the third phase the g_c prevails over the low values of the inverse of the entropy, leading to high probabilities (Phase III), in accordance to the desired exploitative sense of the search.

In both cases, probabilities must be properly normalized in the range [0,1] making use of the parameters ξ_m and ξ_c. The weight of this updating of both probabilities also decreases with time, so that when Phase II is widely surpassed, the mutation and crossover probabilities remain mainly constant.

3.7 Eliminating the entropy ambiguity

The use of the entropy function as defined in Eq. (3) with the aim of classifying populations, can lead, if not properly corrected, to ambiguous situations since a population with all its individuals being very similar and having high fitness values will show the same high entropy as another population with very similar individuals taking on, all of them, very low fitness values. Obviously, the dispersion in both cases is low, and the associated entropy will show a high value. In order to detect and differentiate these situations, the following simple strategy has been devised.

Once a few generations have been processed (about 10–15), the following parameters are compared:

$$\Gamma_1^{\alpha}(k) = \frac{1}{\alpha}\sum_{j=0}^{\alpha-1}\left[\frac{1}{n_p}\sum_{i=1}^{n_p}\lambda_i(k-j)\right] \tag{8}$$

$$\Gamma_0^{\alpha}(k) = \frac{1}{\alpha}\sum_{j=0}^{\alpha-1}\left[\frac{1}{n_p}\sum_{i=1}^{n_p}\lambda_i(k-(j+\zeta))\right] \tag{9}$$

The first one, $\Gamma_1^{\alpha}(k)$, represents the averaged mean value of the individuals' fitness in a successive generations –the current one, k, and the α −1 generations before. The second parameter, $\Gamma_0^{\alpha}(k)$, represents this same value but averaging the mean fitness values in a group of α iterations, ζ generations before. For instance, the simulations for the first application proposed in this chapter were carried out with α =3 and ζ=8, thus the mean value of the fitness values averaged over the last 3 successive iterations is compared to this value 8 generations before.

For example, at generation k=11, the algorithm compares the averaged value of the mean fitness in generations 9, 10 and 11, with the averaged value of the fitness in the generations 1, 2 and 3.

In cases of good convergence (or, at least, when the GA-TS algorithm has begun to converge), the difference between $\Gamma_1^{\alpha}(k)$ and $\Gamma_0^{\alpha}(k)$, will be higher than a predetermined threshold Γ_{th} (heuristically determined). On the other hand, when the population contains low fitness individuals after $2\alpha+\zeta$ generations, it is a clear indicator of unsatisfactory convergence. In this case the whole population is randomly re-initialized.

According to this, the difference between both averaged values, is compared to a threshold Γ_{th}. Whenever condition

$$\Gamma_1^{\alpha}(k) - \Gamma_0^{\alpha}(k) < \Gamma_{th} \tag{10}$$

is satisfied, the whole population will be randomly re-initialized.

3.8 GA improvement with TS concepts

The search algorithm described before has been improved by incorporating some of the most important and characteristic concepts of TS. Specifically, *memory* has been implemented so as to avoid revisiting solutions. Depending on the application, *tabu conditions* t_r have been defined representing non-possible or incongruent solutions. This mechanism, together with an appropriate *diversity control* based on the entropy dependent operators leads to a very efficient hybrid GA-TS method that avoids cycling search, improves convergence (as a consequence of improving diversity) and can be implemented in a computationally efficient algorithm.

The hybrid GA-TS thus implemented can be viewed as an advanced TS method with multiple hypotheses i evolving in parallel and with a powerful procedure to evolve to the next iteration using the dynamic operators here developed. This method can perfectly be classified as a hybrid evolutionary algorithm combining the advantages of both GAs and TS.

4. Application I: multiuser detection for DS/CDMA communications

This section shows the application of the previously described GA-TS algorithm to the problem of symbol detection in DS/CDMA multiuser communications. The thus obtained detector has an extremely low computational load and offers an interesting alternative to previous suboptimal algorithms whose performance is frequently subject to the near-far problem and multiple access interference degradations. Its performance is compared to that of standard GA-based detectors, as well as traditional multiuser detectors, such as the matched filter, the decorrelator and the MMSE detectors. This section is mainly based on (San José, 2005).

4.1 Problem description

During the last decade, the utilization of wireless communications has shown growth rates of 20-50% per year in various parts of the world. The use of Code-Division Multiple Access (CDMA) communications has received a considerable amount of attention as mobile cellular telephone providers look for schemes of transmission which can exploit the capacity of the available spectrum to the maximum (Glisic, 1997; Proakis, 1995; Verdú, 1998)

Since CDMA systems give users access to very high data rates, intersymbol interference (ISI) cannot be neglected and, together with multi-access interference (MAI), constitutes the major drawback to the overall system performance (Proakis, 1995). Both effects, if not appropriately controlled, can seriously deteriorate the quality of reception.

Numerous methods have been proposed for reducing the amount of MAI present in the received signal, such as power control, optimization of signature sequences or sectorized antennas. Conventional receivers, which rely on a filter matched to the signature of the user of interest, are only optimum when the set of received signatures is perfectly orthogonal, which is not the most common case found in practice. Thus, their performance is notably degraded, especially when near-far effects exist. In 1986, Verdú showed that this problem could be solved by jointly-extracting the information sequences of all the users (Verdú, 1998). Unfortunately, the complexity of the optimum multiuser detector (MUD) based on the maximum likelihood (ML) rule increases exponentially with the number of active users, making it impractical for realistic environments. Consequently, the development of suboptimal detectors has deserved a considerable amount of attention during last years.

Many of these suboptimal schemes make use of Natural Computation techniques, such as genetic algorithms and neural networks (Ergun, 2000; Juntti, 1997; Shayesteh, 2001; Shayesteh, 2003; Wang, 1998; Yen, 2000).
A GA-based MUD was first proposed by Juntti et al. in (Juntti, 1997). The analysis is based on a synchronous CDMA system and requires good initial guesses concerning the possible user symbol sequences obtained from the other detectors. However, Yen et al. (Yen, 2000) showed that, by incorporating an element of local search prior to invoking the GA, the performance approaches the single-user performance bound. The proposal by Ergün et al. (Ergun, 2000) uses a multistage MUD as part of the GA-aided detection procedure, in order to improve the convergence rate. Other interesting approaches briefly described in (San José, 2005) include those by Yen and Hanzo (Yen, 2001) and Shayesteh et al. (Shayesteh, 2001; Shayesteh, 2003).
In the following sections we propose to use a multiuser detector based on the GA-TS algorithm described in section 3. This detector offers an extremely low computational load as well as a remarkable diversity control based on the on-line adjustment of its internal parameters. This task is achieved by making use of an individuals' fitness dispersion measure based on the Shannon entropy. Based on the ML rule, we develop a GA-TS based multiuser detector that estimates both the channel impulse response (CIR) and the transmitted bit sequences on the basis of the statistics provided by the bank of matched filters at the receiver.

4.2 DS/CDMA system description

Consider a symbol-synchronous binary communication system with K active users, employing normalized modulation waveforms $\left\{ s_i(t) \right\}_{i=1}^{K}$, and signaling through a dispersive channel with AWG noise. User i transmits a sequence of statistically-independent symbols, $b_i(n)$, which modulates the PN sequence, $s_i(n)$, so that the spectrum is spread by a factor N (processing gain) (Glisic, 1997). Thus, the signal transmitted by the ith user is

$$x_i(t) = \sum_{n=0}^{M-1} b_i(n) s_i(t - nT) \tag{11}$$

where T is the signalling interval, M is the number of data bits in a frame transmitted by each user, and the signature waveforms are given by

$$s_i(t) = \sum_{\ell=0}^{N-1} s_{i,\ell} \psi(t - \ell T_c) \tag{12}$$

where $\mathbf{s}_i=(s_{i,0},...,s_{i,N-1})^T$ is the signature vector of user i, $T_c=T/N$ is the chip interval and $\Psi(t)$ represents the energy-normalized chip pulse.
Due to the consideration of synchronous transmission and assuming that users transmit data packets over a single-path frequency-nonselective slowly Rayleigh fading channel, the received baseband signal is given by

$$r(t) = \sum_{i=1}^{K} r_i(t) + g(t) \qquad 0 \le t \le T_f \tag{13}$$

with

$$r_i(t) = \sqrt{E_i} \sum_{n=0}^{M-1} h_i(n) b_i(n) s_i(t - nT) \tag{14}$$

where T_f is the frame duration, $g(t)$ is a zero-mean complex additive white Gaussian noise uncorrelated with $b_i(n)$, $h_i(n)$ denotes the complex CIR coefficient of the ith user and E_i represents the bit energy of user i. We consider, also, the time variation model proposed in (Yen, 2001), where $h_i(n)$ varies over the frame duration according to a specified Doppler frequency, f_d. The unknown variables in Eq. (14) are $b_i(n) \in \{+1,-1\}$ and $h_i(n)$, which denote the nth bit and the corresponding complex CIR coefficient of the ith user, respectively.
At the receiver, a bank of filters matched to the set of users' signature sequences takes samples at every bit interval (see Fig. 6).

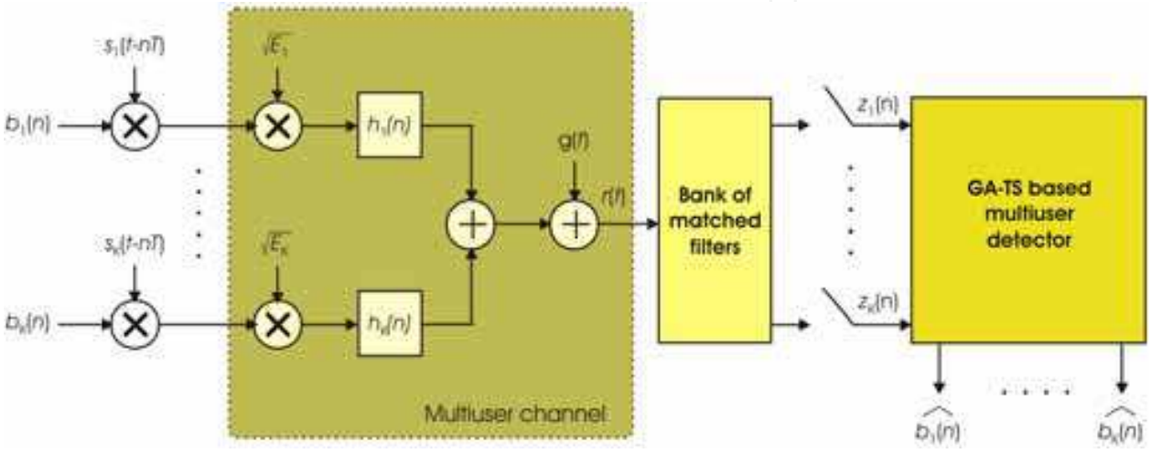


Fig. 6. DS/CDMA multiuser communications system model.

The developed GA-TS-based multiuser detector estimates symbol vector $\boldsymbol{b}(n)=[b_1(n),\ldots,b_K(n)]^T$, containing the symbols transmitted in the nth symbol period. Using the maximization problem as stated in (Yen, 2001), the output of the matched filters can be written as

$$\mathbf{z}(n) = [z_1(n),\ldots,z_K(n)]^T = \mathbf{RH}(n)\mathbf{Eb}(n) + \mathbf{g} \tag{15}$$

where $\mathbf{R}$ is the $K\times K$ user signature sequence cross-correlation matrix, $\mathbf{H}(n) = \mathrm{diag}\left(h_1(n),\ldots,h_K(n)\right)$, $\mathbf{E} = \mathrm{diag}\left(\sqrt{E_1},\ldots,\sqrt{E_K}\right)$, $\mathbf{b}(n) = \left[b_1(n),\ldots,b_K(n)\right]$ and $\mathbf{g} = \left[g_1(n),\ldots,g_K(n)\right]$.

Based on the observation of vector $\mathbf{z}$, it can be shown that the log-likelihood function (LLF conditioned on both the channel matrix $\mathbf{H}(n)$ and the users' data vector $\mathbf{b}(n)$, is given by (Fawer, 1995)

$$\begin{aligned} L\left(\mathbf{H}(n),\mathbf{b}(n)\right) &= 2\Re\left\{\mathbf{b}(n)^T\,\mathbf{E}[\mathbf{H}(n)]^*\,\mathbf{z}(n)\right\} \\ &\quad -\mathbf{b}(n)^T\,\mathbf{EH}(n)\mathbf{R}[\mathbf{H}(n)]^*\,\mathbf{Eb}(n) \end{aligned} \tag{16}$$

where symbol "*" denotes the conjugate operation. Thus, the optimal estimates of the diagonal channel gain matrix $\mathbf{H}(n)$ and the vector of transmitted symbols $\mathbf{b}(n)$ are given by

$$(\widehat{\mathbf{H}(n)},\widehat{\mathbf{b}(n)}) = \arg \max_{\mathbf{H}(n),\mathbf{b}(n)} \left\{ L\left(\mathbf{H}(n),\mathbf{b}(n)\right) \right\} \tag{17}$$

subject to a constraint given by the specific model chosen for the time variation of the channel coefficients matrix **H**.

In this work the channel fading is assumed to slow such that coefficients $h_i(n)$ may be considered constant over one signalling interval and the fading is independent for all users. Several models have been used in order to describe the channel characteristics. For example, the Jakes model (Verdú, 1998) or a first-order Gauss-Markov model as given by

$$h_i(n+1) = \alpha h_i(n) + \nu_i(n) \tag{18}$$

where $\alpha = \exp(2\pi f_d T)$ and $v_i(n)$ is a zero mean white Gaussian variable. Alternatively, this relation between $h_i(n)$ and $h_i(n+1)$ can be expressed as

$$h_i(n+1) = h_i(n) + \Delta_i(n) \tag{19}$$

with $\Delta_i(n)$ being a random variable whose value depends on α and $\nu_i(n)$. Thus, Eq. (19) constitutes the constraint required for solving the maximization problem given by Eq. (17), and represents the channel model that will be here used.

4.3 Specific fitness calculation and encoding scheme

In the DS/CDMA symbol detection problem the fitness value is computed by substituting the corresponding elements of $\mathbf{u}_i$ into the log-likelihood function L defined in Eq. (16), since both the channel coefficients and the users' transmitted symbols are encoded in $\mathbf{u}_i$:

$$\begin{aligned} \mathbf{u}_i &= [\mathbf{H}_i(n,k) \,|\, \mathbf{b}_i(n,k)] \\ &= [(h_{i,1}(n,k), h_{i,2}(n,k), \ldots, h_{i,K}(n,k)), (b_{i,1}(n,k), b_{i,2}(n,k), \ldots, b_{i,K}(n,k))] \end{aligned} \tag{20}$$

with parameters i, n, k and K denoting the ith chromosome of the population, the nth signaling interval, the kth generation and the number of active users K, respectively. When the performance of the algorithm is evaluated by means of simulations, the number of users, K, is initially fixed, and it is not allowed to change with respect to the rest of the parameters.

Vector $\mathbf{b}_i(n,k)$ represents a bit string of length K, which is directly binary encoded wit {+1,−1} elements. On the other hand, each complex CIR coefficient in encoded into a binary string of length 22 as depicted in Fig. 7.

Each complex coefficient $h_{i,j}(n,k)$ has both a real and an imaginary part. Each of them, in turn, is represented by a string of 1+10 bits. The first bit (on the left) indicates the sign of the coefficient (with "1" meaning positive sign) and the remaining 10 bits encode the magnitude of the coefficients - with three decimal positions - as shown in the example. Since 10 bits allow numbers between 0 and 1023, a normalization step is used in order to work in the range [0,1].

On the other hand, due to our particular application and encoding scheme, K+1 different positions are randomly generated for possible mutation: one position within the 22 bits of the binary encoding of each channel coefficient $h_{i,j}(n,k)$, $1 \le j \le K$, in Eq. (20), and another one in the part that encodes the users' bits, i.e. $\mathbf{b}_i(n,k)$ in Eq. (20). In each of these K+1 positions, mutation is performed with probability p_m.

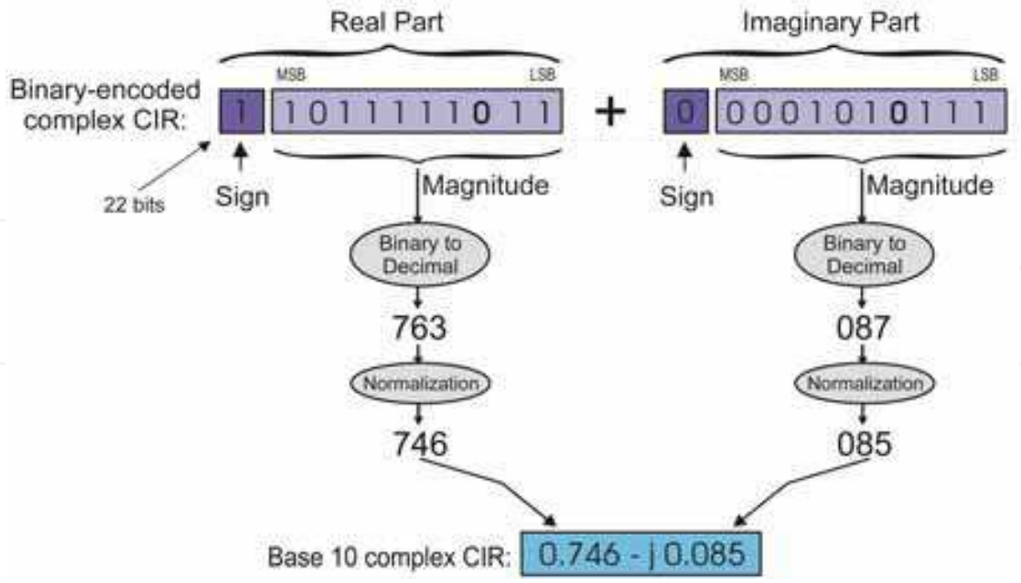


Figure 7. Example of the binary encoding of the channel impulse response coefficients.

Note that, due to the consideration of synchronous transmission and perfect time coordination between the transmitter and the receiver, the detection is achieved by considering separate signal intervals, and, consequently, one GA-TS is executed per interval. As a consequence, the knowledge about the channel impulse response is maintained from one symbol period to another.

4.4 Numerical results

For the simulations, a rectangular chip pulse $\Psi(t)$, AWG noise, Gold-type signatures and a Doppler frequency of 1000 rad/sec are considered. A perfect synchronization is assumed as well. Experiments are performed using binary encoding, K=10, n_p=16, β_m= 0.1 and N=31, if nothing different is specified. Also, binary tournament selection, uniform crossover and an elitism of 3 individuals are used. Memory and simple control of tabu moves are also implemented. The initial probabilities were: p_m=0.1 and p_c=0.05, which proved to be good settings in preliminary experiments.

- **BER performance of the user of interest for different multiuser detection schemes**

Figure 8 shows the BER performance of the user of interest versus $SNR=E_1/N_0$, with a near-far effect of 4 dB (i.e. $E_k/E_1 = 4$ dB, $2 \le k \le K$), for different types of detectors.

First, the proposed GA-TS algorithm is compared to a standard GA in terms of the BER of the user of interest (user 1) versus the signal to noise ratio of this user (E_1/N_0). It can be seen how the GA-TS has a performance close to the limit of the optimal single user detector - a lower bound of the multiuser detector can be found by simulating the BER of the single user receiver in the absence of the other interfering users (Proakis, 1995) - while the standard GA saturates at high SNRs even with higher population sizes and number of iterations – this fact has also been pointed out by other authors, such as Shayesteh in (Shayesteh, 2003).

Simulations show that the GA-TS method needs about 23% of the population needed for the standard GA, and a number of iterations about 15-20%, to get a similar performance, i.e., saving about 70% of time. This value is similar to that reported in (Shayesteh, 2003), where a GA is used to estimate only the vector of users' symbols (in contrast to our approach that jointly estimates both the CIR and the symbols (see section 4.2).

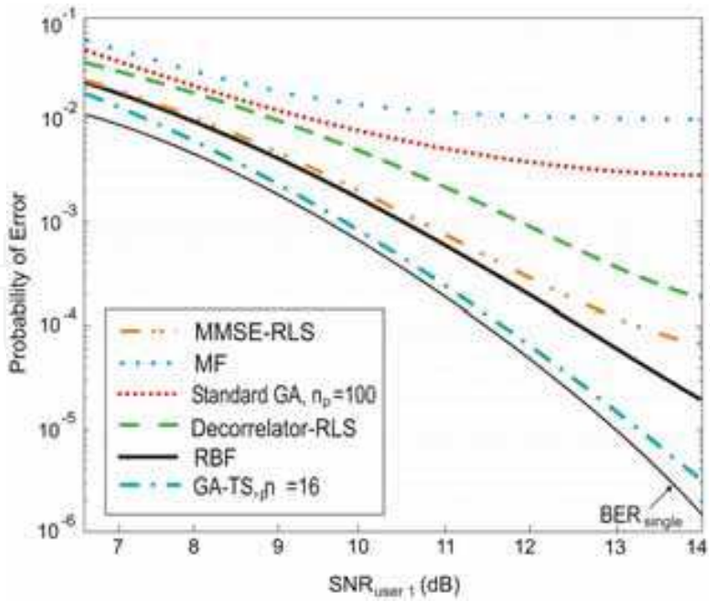


Fig. 8. Probability of error for different types of multiuser detectors. E_k/E_1=4 dB for $k \neq 1$.

The error probabilities of those traditional detectors based on matched filters (MF), the MMSE detector, the decorrelator and an RBF network are also shown for comparison in Fig. 8. The RBF-based MUD chosen for comparison is described in (San José, 2003). As kernel function, the basis function based on the Mahalanobis distance measure is used - its explicit expression can be found in (San José, 2003). The simulations carried out with both the MMSE and the decorrelator detectors, where implemented using a CIR estimation algorithm whose main details and equations can be found in (Cañibano, 2004). For the simulations carried out in this chapter, the RLS version of the equations there obtained was used.

	Standard GA	(Shayesteh, 2003) method	(Yen, 2001) method	Proposed GA-TS
Population size, n_p	60–80	25	40	15
No. generations, n_g	150–250	20	10	20-25
Fitness evaluations	5000	520	400	260

Table 1. Quantitative comparison between various evolutionary multiuser detectors (K=10).

Analyzing the BER plots shown in Figure 8, it can be observed that the proposed GA-TS shows better performance, specially when the SNR is high, than the other compared MUDs. Table 1 compares quantitatively the proposed GA-TS with (Shayesteh, 2003) (for this algorithm the channel is considered to be known), (Yen, 2001) and the standard GA. In case the channel response is known, (Shayesteh, 2003) reports that its algorithm needs about 21% of the population size required in a standard GA (also implemented with known channel response), but using a similar number of iterations. In our simulations, the standard GA used for comparison needs about 150-250 iterations for convergence, while the proposed GA-TS uses only about 20-25. Fig. 8 shows that only a small degradation is observed with respect to the single-user bound when the proposed GA-TS multiuser detector is used.

- **Population size vs number of active users**

Analyzing how the number of active users affects the proposed detector, we find that the BER performance gradually degrades upon increasing the number of users, due to the limited population size n_p, which becomes too small for adequately exploring a larger space. Furthermore, in comparison with the "brute-force" optimum ML MUD requiring $2^K=2^{10}=1024$ fitness function evaluations, the proposed detector is substantially less complex, requiring only $n_p \, x \, n_g$ = 15 x 20 (or 23) = 300 (or 375) evaluations, yet performing close to the optimum ML MUD. This fact clearly manifests that the computational complexity of the GA-TS algorithm does not depend exponentially on K. In fact, with 20 users, a population size of 125 - keeping n_g constant - allows the detector to attain a near-optimal performance. This means an increase by a factor of 150/15=8.33 in the computational load. Furthermore, in contrast to the reduced tree-search type algorithms, our detector does not require any memory capacity, since all the information related to previous generations can be erased.

It should be noticed that this increment in the number of active users could also be addressed by increasing the number of iterations n_g while keeping constant the size of the population n_p. These two solutions could be considered also if the SNR decreases. This interplay between n_g and n_p has also been pointed out in (Shayesteh, 2003). However, it is difficult to find a general rule that could easily quantify this equivalence between n_p and n_g since it greatly depends on factors like the number of users K, the ratio E_k/N_0, etc. In the aforementioned example, with (K=20, n_p=125, n_g=20) another configuration of the GA-TS with (K=20, n_p=75, n_g=40) obtains similar results.

Next, the BER performance for K=15 users is analyzed. Because of the higher number of users to be optimized, we increased the population size. With n_p = 35, a significant degradation was observed. This situation was due to the limited population size, which was too small for optimizing the number of involved variables. However, when increasing n_p up to 45, the performance became near optimum, though the increase in complexity is really low in comparison with the computational load of the conventional optimum MUD, whose complexity will now be increased by a factor of 32.

No. of users (K)	Population size (n_p)	No. of generations (n_g)
10	15 / **40**	20 / **10**
15	45 / **75**	20 / **10**
20	125 / **160**	20 / **10**

Table 2: Population size (n_p) and number of generations (n_g) required for different number of active system users (K) for the proposed GA-TS algorithm. Bold values correspond to the (Yen, 2001) method.

These results are summarized in Table 2. Values in bold typesetting correspond to the GA-based detector developed by Yen and Hanzo (Yen, 2001). It can be seen that the GA-TS scheme requires a smaller population size at the expense of a higher number of iterations.

- **BER vs number of active users**

Now, in Figure 9, the BER for different receiver structures (GA-TS, RBF, MF and MMSE) as a function of the number of users is compared. For this simulation, a perfect power control has been considered: $E_i = E_j$, $1 \le i,j \le K$, $i \ne j$.

The MMSE structure and the linear MF detector are outperformed by the nonlinear RBF and GA-TS receivers. The MF suffers from MAI and the MMSE lacks stability to perform a

nonlinear decision boundary. The choice of the Mahalanobis distance in the RBF basis function allows some advantages over the Euclidean one. The Euclidean-RBF performance tends towards the MMSE receiver performance due to the non-spherical shape of the multidimensional clusters.

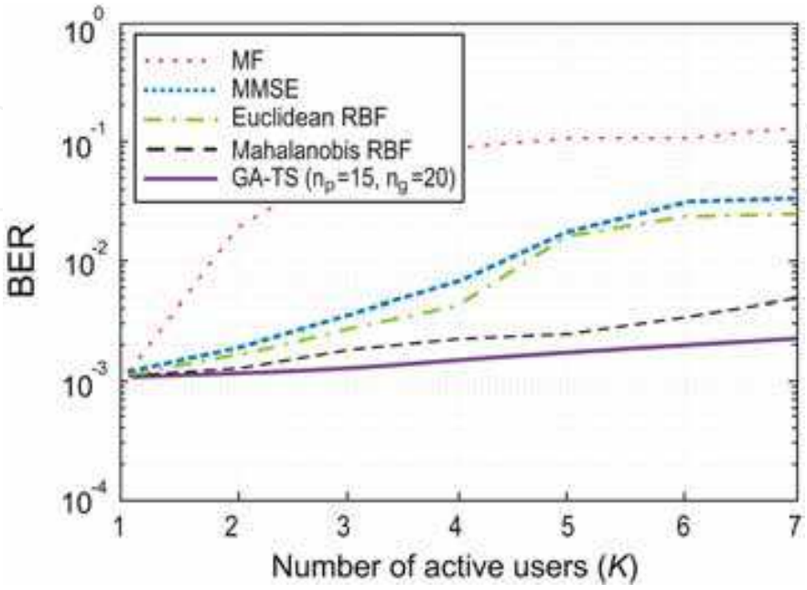


Fig. 9. Probability of error vs number of active users.

Figure 9 shows that the proposed GA-TS method and the Mahalanobis RBF detector have similar performance. However, it must be pointed out that the complexity of the RBF detector used for comparison is much higher. This RBF-based detector - described in (San José, 2003) - needs a training period if the channel response is unknown and its complexity (number of nodes) increases exponentially with the number of active users.

Quantitatively, the computational requirements (in terms of floating point operations in MATLAB) have been experimentally shown to be at the same level when just a few users are present ($K \leq 3$). However, for higher values of K, the proposed GA-TS receiver clearly outperforms the RBF-based detector in terms of computational complexity.

- **Joint channel estimation and symbol detection performance**

In this section we compare the performance of the proposed GA-TS-based multiuser detector when (i) the channel impulse response is estimated using the GA-TS method (joint estimation of the users' symbols and the CIR) and (ii) the CIR is perfectly known, i.e. the true channel coefficients are substituted in Eq. (16).

Figure 10 shows the different BERs obtained with and without estimation of the channel impulse response coefficients. The single-user bound shown is computed as $P_e = 0.5(1-(\gamma/(1-\gamma))^{0.5})$, where $\gamma = E_k/N_0$ (valid for BPSK).

Two comments apply to this figure: (i) the higher n_g and n_p are chosen, the closer are the BER curves for both cases (with and without CIR estimation), and (ii) both BERs are similar up to a threshold value of the E_k/N_0 parameter that depends on the election of n_p and n_g. For instance, in the case with n_p=35 and n_g=15, BERs are very similar whenever $E_k/N_0 \leq 26$ dB. In the other case, with n_p=25 and n_g=10, this threshold is located around 14 dB.

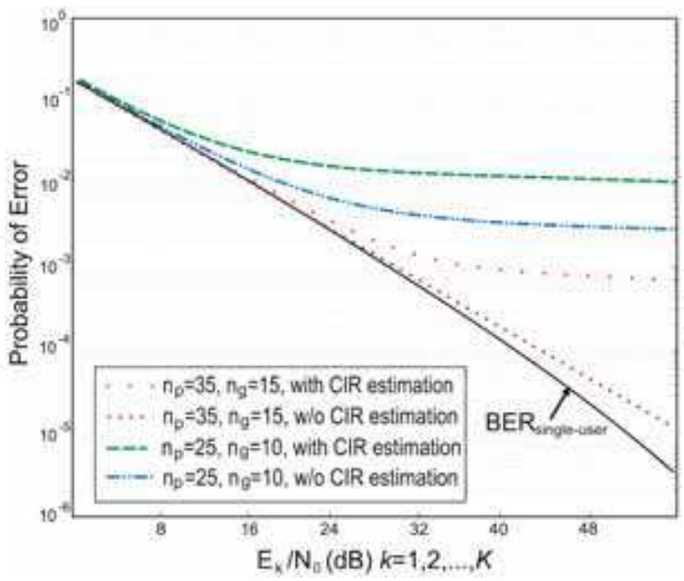


Fig. 10. BER performance comparison between the GA-TS algorithm in which both the channel impulse response and the users' symbols are jointly estimated, and that in which only the users' symbols are unknown. (CIR: channel impulse response).

- **Near-far resistance**

Next, Fig. 11 characterizes the near-far resistance of the proposed detector in terms of the BER of the desired user. Simulations used n_p=30 and n_g=15. The user of interest is user 1 and the averaged received bit energies of all the other users were set to 0, 5, 10 and 15 dB higher. Simulations of the decorrelator detector were performed with RLS parameter estimation, while the GA-TS detector was implemented with the joint estimation of the channel and the symbols. It can be seen that for $E_k/E_1 \leq 10$ dB, the proposed detector is near-far resistant up to $E_1/N_0 \approx 15$ dB. Beyond that, a significant BER performance degradation is observed.

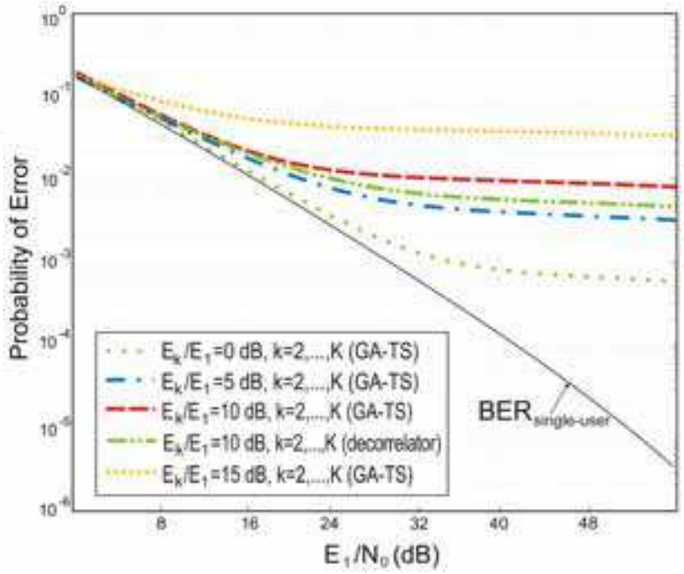


Fig. 11. BER performance for K=10 users with E_k/E_1=0, 5, 10 and 15 dB for k=2,…,K. User 1: user of interest.

For the sake of comparison, the BER performance corresponding to the decorrelator detector implemented using a RLS-type estimation algorithm, is also plotted for the case with E_k/E_1=10 dB. In this case, the decorrelator detector shows a little better performance at the expense of a much higher complexity.

- **Entropy-guided parameter tuning**

Finally, some experiments were carried out in order to study the influence of the entropy-based tuning up process of the GA-TS parameters. Figure 12 represents the evolution of the mean population fitness (mean value after 100 runs) of: (i) the individuals of the elite, (ii) all those individuals not belonging to the elite. For both cases, the evolution is drawn with and without entropy-guided adjustment for the first 50 iterations of the algorithm. For both sets of individuals, the mean fitness approaches the maximum before when the entropy is monitored and the probabilities p_c and p_m are dynamically adjusted. Furthermore, redundant individuals are more efficiently eliminated when the fitness entropy is used to adjust the genetic operators. This way, the diversity of the population increases, and, simultaneously, the probability of getting trapped into suboptimal solutions is reduced. This capability also affects the population size required to get a specific BER, since: (i) very similar individuals do not represent any advantage at the time of searching the solutions space, and (ii) almost no improvement (in terms of mean population fitness) occurs during the exploration phases. Thus, many fitness evaluations can be saved during these periods. Making use of this fact, the number of evaluations can be lowered around 25%.

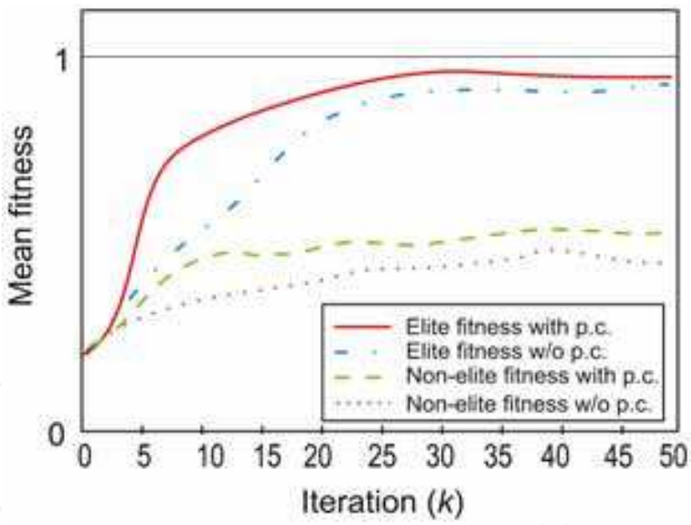


Fig. 12. Evolution of the population fitness with and without parameter control (p.c.).

4.5 Conclusions

A novel hybrid genetic-algorithm / tabu-search (GA-TS) algorithm with an extremely low complexity has been proposed for DS/CDMA multiuser detection. It requires no training period, applies the genetic operators only very few times and saves a substantial amount of fitness evaluations, which is a critical factor in real-world applications. The performance is close to the limit of the single user detector and its complexity remains much smaller than that of traditional approaches and those based on standard GAs. The entropy-guided

procedure to on-line adjust the probabilities of the genetic operators controls the diversity of the population; this leads to a reduction of the population size required to attain a certain performance and a high probability of achieving a reliable convergence result.

5. Application II: tuning-up of a DT-MRI tracking algorithm

5.1 Motivation and problem description

The Diffusion Tensor Magnetic Resonance Imaging (DT-MRI) technique measures the diffusion of hydrogen atoms within water molecules in 3D space. Since in cerebral white matter most random motion of water molecules are restricted by axonal membranes and myelin sheets, diffusion anisotropy allows depiction of directional anisotropy within neural fiber structures (Ehricke, 2006). The DT-MRI technique has raised great interest in the neuroscience community for a better understanding of the fiber tract anatomy of the human brain. Among the many applications that arise from tractography we find: brain surgery (knowing the extension of the fiber bundles could minimize the functional damage to the patient), white matter visualization using fiber traces (for a better understanding of brain anatomy) and inference of connectivity between different parts of the brain (useful for functional and morphological research of the brain).

Most of DT-MRI visualization techniques focuses on the integration of sample points along fiber trajectories (Mori, 2002), using only the principal eigenvector of the diffusion ellipsoid as an estimate of the predominant direction of water diffusion (Ehricke, 2006). However, these methods may depict some fiber tracts which do not exist in reality or miss to visualize important connectivity features, e.g. crossing or branching structures. In order to avoid misinterpretations, the viewer must be provided with some information on the uncertainty of every depicted fiber and of its presence in a certain location. In (San José, 2007) we proposed an estimation algorithm that takes into account the whole information provided by the diffusion matrix, i.e., it does not only consider the principal eigenvector direction but the complete 3D information about the certainty of continuing the path through every possible future direction. An improved version of this algorithm was developed in (San José, 2007b). This article included two main aspects: (i) a procedure for on-line adapting the number of *offspring paths* emerging from the actual voxel, to the degree of anisotropy observed in its proximity (this strategy was proved to enhance the estimation robustness in areas where multiple fibers cross while keeping complexity to a moderate level), and (ii) an initial version of a neural network (NN) for adjusting the parameters of the algorithm in a user-directed training stage. Subsequent work (San José, 2008) studied with more detailed the architecture of the neural network and numerically evaluated its tracking capability, robustness and computational load when used with both synthetic and real DT-MR images. This work showed that in many cases, such as real images with low SNR, a huge computational load was required.

Now we propose to use an evolutionary computation-based approach - the GA-TS algorithm - for tuning-up the parameters of the tracking algorithm instead of using the neural network. The main aim is to adjust the parameters with a less complex procedure and to obtain a robust and efficient tracking algorithm. The required human intervention time can also be reduced. Numerical results will prove that the GA-TS approach leads to similar and even better convergence results while offering much lower computational requirements.

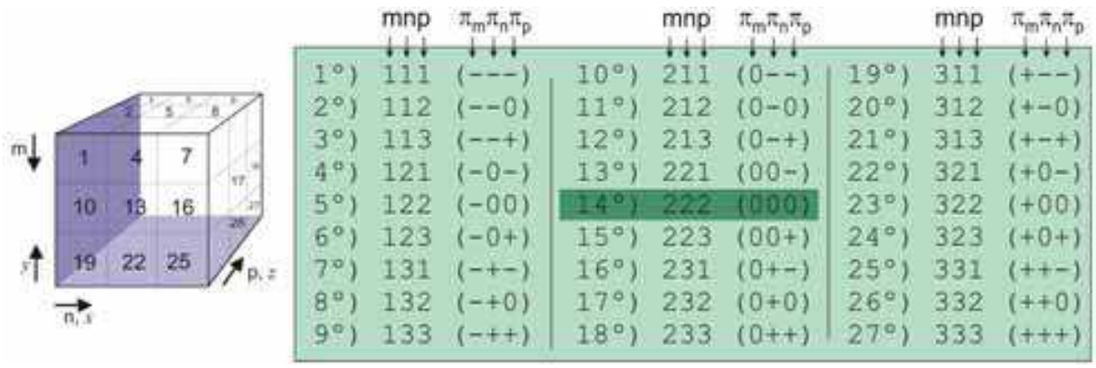
5.2 Brief tracking algorithm description

Since our main purpose is the development of an algorithm to adjust the parameters of a tracking algorithm, it is necessary to outline the head expressions of this algorithm - for the sake of brevity, the reader interested in a detailed development of this tracking algorithm can see the corresponding sections in (San José, 2007). Consequently, this section presents a brief summary of the method. The algorithm uses probabilistic criteria and iterates over several points in the analyzed volume (the points given by the highest probabilities in the previous iteration). The process starts in a user-selected seed voxel, V_0.

Every iteration, the method evaluates a set of parameters related to the central voxel of a cubic structure similar to that shown in Figure 13, left. The central point, V_c, (No. 14 in the figure) represents the last point of the tract being analyzed. In the first iteration, $V_c = V_0$.

- **Basic concepts**

First, a measure P_i, $i \in \{valid\ points\}$, is evaluated based on the probability of going from voxel V_c to voxel V_i. This probability takes into account the eigenvalues and eigenvectors available at point V_c from the DT-MR image diffusion matrix. In order to calculate this probability, the information shown in Fig. 13, right, is used.



	mnp	$\pi_m\pi_n\pi_p$		mnp	$\pi_m\pi_n\pi_p$		mnp	$\pi_m\pi_n\pi_p$
1°)	111	(---)	10°)	211	(0--)	19°)	311	(+--)
2°)	112	(--0)	11°)	212	(0-0)	20°)	312	(+-0)
3°)	113	(--+)	12°)	213	(0-+)	21°)	313	(+-+)
4°)	121	(-0-)	13°)	221	(00-)	22°)	321	(+0-)
5°)	122	(-00)	14°)	222	(000)	23°)	322	(+00)
6°)	123	(-0+)	15°)	223	(00+)	24°)	323	(+0+)
7°)	131	(-+-)	16°)	231	(0+-)	25°)	331	(++-)
8°)	132	(-+0)	17°)	232	(0+0)	26°)	332	(++0)
9°)	133	(-++)	18°)	233	(0++)	27°)	333	(+++)

Fig. 13. Modifications of indices (m,n,p) when moving from V_c to the neighbouring voxel V_i, $1 \le i \le 27$, $i \ne 14$.

The table shows, for every voxel shown in Fig. 13, left, the changes that must occur in indices (m,n,p), when a tract goes from voxel V_c to voxel V_i. For instance: when going from point No. 14 to point No. 17, coordinates m and n increase by 1, and p remains the same. This is represented in the table with "π_m,π_n,π_p = (++0)". With this information, the probability of each possible destination V_i can be calculated taking into account the projection of each of the eigenvectors to each of the directions defined in the triplet π_m, π_n, π_p. Besides, each projection is weighted by the corresponding eigenvalue λ. Thus, in the previous example, P_i should be calculated as $P_i = V_{1y}\,\lambda_1 + V_{2y}\,\lambda_2 + V_{3y}\,\lambda_3 + V_{1z}\,\lambda_1 + V_{2z}\,\lambda_2 + V_{3z}\,\lambda_3$, where $V_{j\alpha}$ represents the α-component of eigenvector j, $1 \le j \le 3$, $\alpha \in \{x,y,z\}$.

The axes reference criterion for the (x,y,z) vector components is also shown in Fig. 13. Note that, for this calculus, the sign "-" in the triplet is equivalent to sign "+". In order to properly calculate P_i, it must be weighed by 0.33 if there are no zeros in triplet i, and by 0.5 if there is one zero.

- **Anisotropy and local probability**

The following anisotropy index is used in the algorithm:

$$fa = \sqrt{\frac{(\lambda_1 - \lambda_2)^2 + (\lambda_2 - \lambda_3)^2 + (\lambda_1 - \lambda_3)^2}{2(\lambda_1^2 + \lambda_2^2 + \lambda_3^2)}}, \tag{21}$$

where $\lambda_1 \geq \lambda_2 \geq \lambda_3$. When both $fa(V_c)$ and $fa(V_i)$ do not exceed a certain threshold, then point V_i is eliminated as a possible destination point.

Taking into account both P_i and the anisotropy given by Eq. (21), the local probability of voxel i is defined as

$$P_i^{'} = a \cdot \mu_1 \cdot fa(V_i) + (1-a) \cdot \mu_2 \cdot P_i, \qquad 0 < a < 1 \tag{22}$$

where parameter a allows the user to give a higher relative weight to either the anisotropy or the local probability, and μ_1 and μ_2 are scaling factors (normally, 1 and 1000, respectively). The set of values $P_i^{'}$ is properly normalized so that they can be interpreted as probabilities.

- **Eigenvectors and direction considerations**

Besides these considerations, the final probability of voxel i makes also use of the so-called *smoothness parameters* - described in (Kang, 2005) - which judge the coherence of fiber directions among the trajectories passing through voxel V_c. The mathematical expressions of these four parameters, $\{sp_i\}_{i=1}^{4}$, as well as their geometrical meaning, is explained in (San José, 2008). They measure the angles between the directions that join successive path points, as well as the angles between these directions and the eigenvectors associated to the largest eigenvalues found in those voxels. sp_2, sp_3 and sp_4 are used to maintain the local directional coherence of the estimated tract and avoid the trajectory to follow unlikely pathways. The threshold for sp_1 is set such that the tracking direction could be moved forward consistently and smoothly, preventing the computed path from sharp transitions.

Next, the following parameter is calculated for every valid point whose smoothness parameters satisfy the four corresponding threshold conditions,

$$P_i^{''} = b(\xi_1 sp_1 + \xi_2 sp_2 + \xi_3 sp_3 + \xi_4 sp_4) + (1-b)P_i \tag{23}$$

where, ξ_1, ξ_2, ξ_3 and ξ_4 are the corresponding weights of the smoothness parameters (normally, 0.25), and b stands for a weighting factor.

- **Path probabilities**

Probabilities $P_i^{''}$ can be recursively accumulated, yielding the probability of the path generated by the successive values of V_c,

$$P_i^{'''} = P_i^{''} / \sum_i P_i^{''} \tag{24}$$

with k being the iteration number, and $P_i^{'''} = P_i^{''} / \sum_i P_i^{''}$. At the end of the visualization stage, every estimated path is plotted with a colour depending on its probability P_p.

- **Final criterion and pool of "future seeds"**

A pool of voxels is formed by selecting, at the end of each iteration, the s best voxels according to Eq. (23). The first voxel of the pool becomes the central voxel V_c at next iteration, expanding, this way, the current pathway.

As proposed in (San José, 2008), the value of s is adjusted depending on the degree of anisotropy found in current voxel V_c and its surroundings. When this anisotropy is high, it

means that a high directivity exists in that zone, and the probability that V_c belongs to a region where fibers cross is really low. Consequently, s takes a small value (1, 2 or 3). On the other hand, if V_c is found to be situated in a region of high anisotropy, the probabilities of having fibers crossing or branching is higher. In this case, it is interesting to explore various paths starting in V_c. This can be achieved by setting parameter s to a higher value.

- **Parameters to be estimated using the GA-TS algorithm**

We propose to use the previously developed GA-TS procedure for adjusting the parameters of the algorithm $\Omega = (a, b, \mu_1, \mu_2, \xi_1, \xi_2, \xi_3, \xi_4)$, instead of using the complex and time consuming neural network proposed in (San José, 2007b; San José, 2008). This adjustment is useful when the algorithm is applied to a different part of the brain (fiber bundles) or even to the same portion but having been scanned with under different conditions. In these cases, the volume of interest will have a different smoothness and anisotropy characterization.

- **Estimation procedure**

The user-aided estimation procedure works as follows: first, the user is requested to manually draw a sample fiber path as well as to compare this fiber path to those estimated by the GA-TS method during its first stages. Specifically, the steps for the estimation of the parameters are: (i) the user manually draws a sample fiber path, $\mathbf{r}_u$, (ii) the GA-TS scheme starts with a randomly generated population of n_p=10 individuals $\{\mathbf{u}_i\}_{i=1}^{n_p}$, each of them being a possible binary representation of parameters Ω, (iii) the tracking algorithm of section 5.2 is applied n_p times, each of them with the set of parameters represented by each GA-TS's individual. This way, n_p different paths $\mathbf{r}_i$ are obtained, (iv) every path $\mathbf{r}_i$ is compared with $\mathbf{r}_u$ and given a fitness value λ_i, (v) iterate the GA-TS algorithm during n_g=25 generations and then go to step (ii).

Every time that the fiber paths are obtained at step (ii) the user must compare them to his sample $\mathbf{r}_u$ and, in case he finds that a tract $\mathbf{r}_j$, $1 \le j \le n_p$, is better than his first estimation $\mathbf{r}_u$, then $\mathbf{r}_j$ becomes the new reference path $\mathbf{r}_u$. At the end, solution Ω is obtained from the encoding of the fittest individual.

Though this scheme seems initially complicated, experiments show that a few iterations lead to sets Ω that allow to obtain good results when used in the tracking algorithm. The user will not have to assign too many fitness values or to perform many comparisons. The extremely reduced size of the population and the low number of generation per GA-TS execution, derive in moderately short training periods.

5.3 Numerical results

In order to evaluate the proposed algorithm (parameter tuning + tracking), both synthetic and real DT-MR images have been used. For the sake of comparison, we have used the same test images as in (San José, 2008).

- **Synthetic images**

Figure 14 shows four different synthetic DT-MRI data defined in a 50×50×50 grid (we will refer to them as "cross", "earth", "log" and "star"). To make the simulated field more realistic, Rician noise (Gudbjartsson, 1995) was added in the diffusion weighted images which were calculated from the Stejskal-Tanner equation using the gradient sequence in (Westin, 2002) and a b-value of 1000. The desired noisy synthetic diffusion tensor data was obtained using an analytic solution to the Stejskal-Tanner equation.

Satisfactory tracing results for the first three cases can be found in (San José, 2007), where a simpler algorithm was used. For the sake of brevity, in this paper we have worked with the

most complex case, the *star*. This image consists of six orthogonal sine half-waves, each of them with an arbitrary radius. Under this scenario the diffusion field experiments variations with the three coordinate axes and there exists a crossing region.

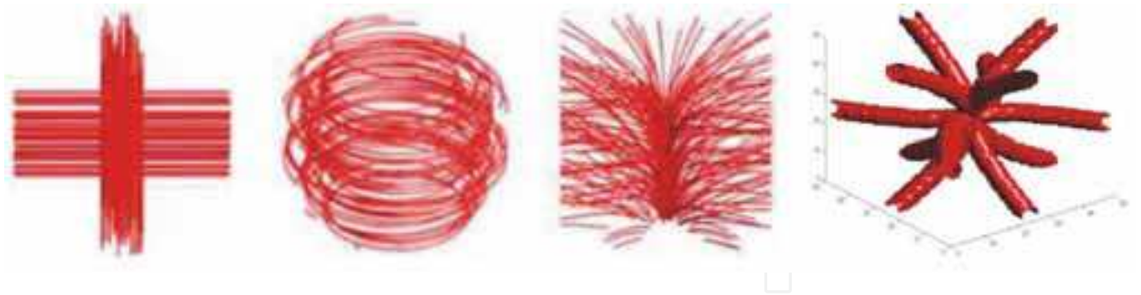


Fig. 14. Synthetic DT-MR images used for testing the proposed algorithm: "cross" (left), "earth", "log" and "star" (right).

Three different tracking results are shown in Fig. 15, each of them for a different seed V_0={S_1,S_2,S_3}. Blue tracts were obtained with an algorithm were parameters were estimated with a NN (San José, 2008), while green ones correspond to the estimation using the proposed GA-TS algorithm.

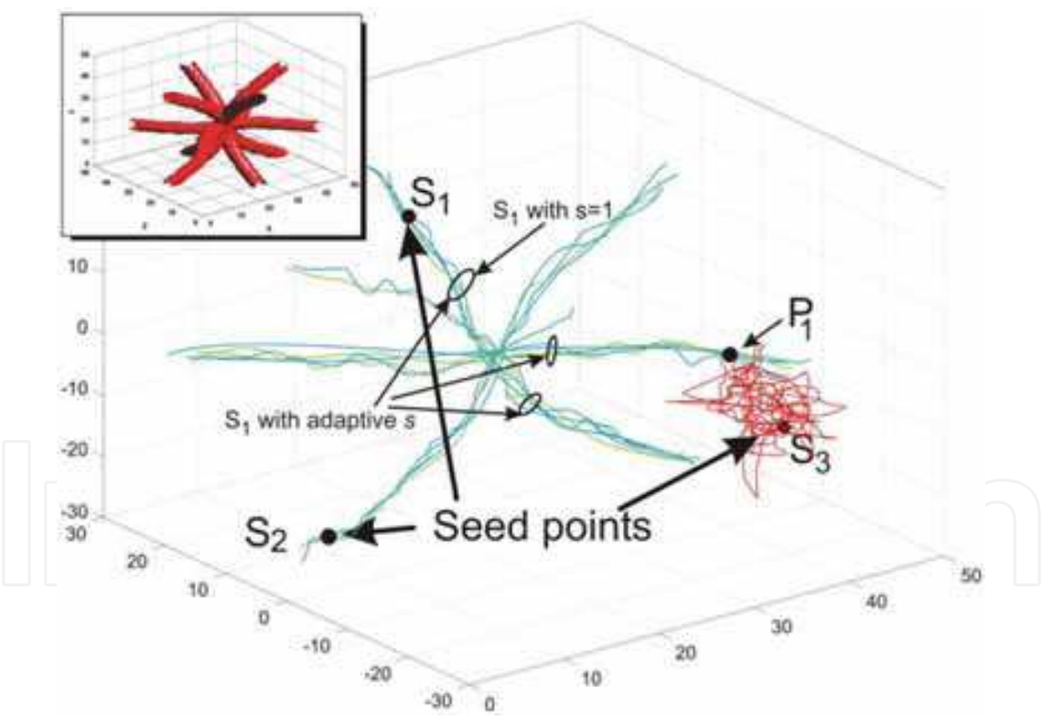


Fig. 15. Tracking results for the "star" synthetic DT-MR image. Black: seed points. Blue: fiber paths obtained using adjustment with NN, Green: paths using estimation with the proposed GA-TS. Red: extrinsic voxels. Initial seeds V_0={S_1,S_2,S_3}. Top left: original synthetic image.

It can be seen that in both cases the path estimates pass through isotropic zones where different fiber bundles cross. It is also appreciated how both methods differentiate between the totally isotropic zones extrinsic to the tracts and the fiber bundles.

The differentiation between voxels belonging to a fiber or to a very isotropic area, respectively, is attained by mapping the path probabilities given by Eq. (24) into a colour scale and classifying them according to some fixed thresholds. Notice that seeds S_1 and S_2 belong to the intrinsic volume (voxels with a very high anisotropy). In this case, both methods move through the most probable direction following the main direction of the star in each situation. When extrinsic point S_3 is selected as seed, the algorithms explore in the neighbouring voxels until they find a voxel with a high anisotropy value (point P_1). Once P_1 is found, the tracking algorithm proceeds as in the case of S_1 and S_2. Fig. 15 shows how the algorithm finds the proper fiber path whatever (extrinsic or intrinsic) seed voxel is chosen, for both methods of parameters' estimation.

			SNR (dB)					
			5	10	15	20	25	30
Image	Cross	NN	78.3/**82.8**	89.7/**93.6**	92.1/**94.3**	98.3/**98.7**	99.0/**99.0**	100/**100**
		AG	81.2/**85.7**	93.6/**94.5**	94.9/**96.1**	98.8/**98.7**	99.0/**100**	100/**100**
		[3]	76.8	89.0	90.7	97.0	100	100
	Earth	NN	77.7/**76.2**	88.6/**87.5**	89.9/**89.0**	98.2/**98.2**	99.0/**99.0**	100/**100**
		AG	82.1/**79.6**	89.6/**92.5**	93.5/**94.0**	98.8/**98.9**	99.0/**100**	100/**100**
		[3]	74.4	83.2	85.0	97.3	99.2	100
	Log	NN	71.0/**69.7**	82.1/**81.0**	86.1/**85.5**	96.0/**95.8**	98.0/**97.8**	100/**100**
		AG	75.0/**75.2**	85.2/**85.0**	89.9/**87.7**	97.0/**97.2**	98.2/**98.4**	100/**100**
		[3]	68.8	78.3	85.2	96.0	98.0	100

Table 3. Convergence performance for different SNRs values. Cell values represent percentage of right convergence for two configurations of the algorithm: s=1/$\boldsymbol{s}$ = **4**. Each cell shows: top: NN-estimation, middle: GA-TS estimation, bottom: Bayesian tracking (Friman, 2005).

Next, the robustness of the tracking algorithm with both parameter estimation methods is now studied. For the sake of brevity, these experiments were run with parameter s kept constant during the fiber tract estimation (see "*Final criterion and pool of future seeds*" in section 5.2).

The convergence performance for different SNRs is shown in Table 3. The first row in each cell corresponds to tracking results when parameters where estimated using the NN, the second contains the results when the proposed GA-TS is used for this estimation, and the third one shows the values obtained with a slightly modified version of the Bayesian method proposed in (Friman, 2005).

It can be seen that both algorithms (with NN- and GA-TS-based adjustment) converge properly within a wide range of SNRs, with the GA-TS version showing a convergence gain of about 3-6% in all cases. The percentage values for the "cross" and the "earth" test images are very close, while for the "log" case both algorithms exhibit a slightly lower convergence. Comparing our methods with the Bayesian approach, we see that the proposed tracking algorithm performs slightly better when the SNR is low, while the three methods tend to similar results with high SNRs.

Analyzing the simulations of the synthetic images considered, it is seen that convergence results improve whenever the MR image contains branching or crossing areas - as it is the case in real DT MR images. This is the case of our "cross" image. For this image, the convergence results are improved ~ 5% when parameter s is modified according to the anisotropy. Besides, for these studied cases, we see that the influence of the procedure that adapts s is higher for low SNRs.

- **Real images**

The proposed tracking algorithm has also been applied to real DT-MR images. Specifically, we have selected the *corpus callosum* of the brain (see Fig. 16).
Simulation results show that whichever parameters tuning-up method is used, the algorithm is able to follow the main fiber bundle directions without getting out of the area of interest. Fig. 16 shows some bundles of properly estimated tracts. Red/green color indicates high/low certainty.

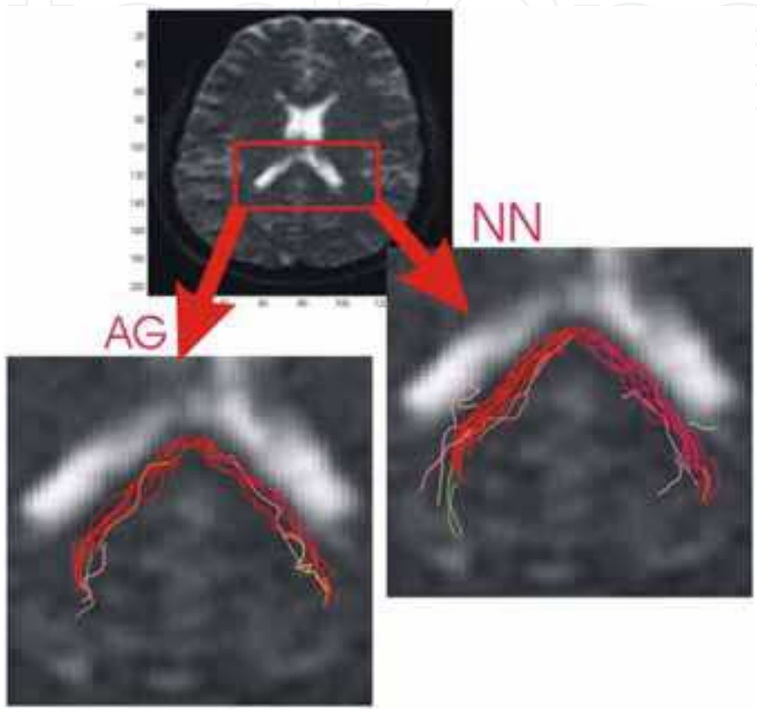


Fig. 16. Tracking results for the *corpus callosum* area of the human brain. Left: tracts obtained with the tracking algorithm tuned-up with the proposed GA-TS method, Right: parameter estimation with neural network.

- **Final remarks**

The proposed parameter estimation procedure is useful when the volume being varies. For instance, with just 5-10 training iterations (repetitions of the "training procedure"), in synthetic images, or 8-16, in real images, the parameters of the algorithm are fine-tuned so as to get satisfactory results. Note that these previous training times are: (i) always inferior to those required by the NN-based method proposed in (San José, 2007b; San José 2008), (ii) always much inferior to the time required to heuristically adjust the parameters, (iii) only required when the scanning conditions vary.

5.4 Conclusions

Numerical simulations have shown that the tracking algorithm that has been tuned-up using the proposed GA-TS-based method is capable of estimating fiber tracts both in synthetic and real images. The robustness and convergence have been studied for different image qualities (SNRs). Results show a convergence gain of about 3-6% with respect to our previous work (San José, 2007b; San José, 2008).

The experiments carried out show that an efficient parameter adjustment in conjunction with precise rules to manage and update both the pool of future seeds and the memory with the tabu list, lead to: (i) a better use of computational resources, (ii) a better performance in regions with crossing or branching fibers, and (iii) a minimization of the required human intervention time. The method has been tested with synthetic and real DT-MR images with satisfactory results, showing better computational and convergence properties even than already existing Bayesian methods such as (Friman, 2005).

6.Acknowledgements

The author would like the Spanish Education Ministry for Grant TEC2007-67073/TCM, that partially funded this work.

7. References

Bhandari, D., Murthy, C.A, Pal, S.K. (1996). Genetic algorithm with elitist model and its convergence, *International Journal on Pattern Recognition and Artificial Intelligence*, Vol. 10, No. 6 (1996) 731–747.

Cañibano, N. & San José-Revuelta, L.M. (2004). Analysis of a Bayesian multiuser detector for non-data-aided CDMA communications, *Proc. IEEE Workshop on Machine Learning and Signal Processing, MLSP'04*, 2004, Sao Lluis, Brazil.

Ergün, C., Hacioglu, K. (2000), Multiuser detection using a GA in CDMA communications systems, *IEEE Trans. on Comm.*, Vol. 48, No. 8 (2000) 1374–1383. ISSN:

Ehricke, H.H., Klose, U., Grodd, U. (2006). Visualizing MR diffusion tensor fields by dynamic fiber tracking and uncertainty mapping, *Computers & Graphics*, Vol. 30 (2006) 255–264.

Faigle, U. & Kern, W. (1992). Some Convergence Results, *Journal on Computing*, Vol. 4 (1992) 32-37.

Fawer, U., Aazhang, B. (1995). A multiuser receiver for CDMA communications over multipath channels, *IEEE Trans. on Comm.*, Vol. 43, (1995) 1556–1565.

Fox, B.L. (1993). Integrating and accelerating tabu search, simulated annealing and genetic algorithms, *Annals of Operations Research*, Vol. 41 (1993) 47-67.

Friman, O. & Westin, C.-F. (2005). Uncertainty in white matter fiber tractography, *Proceedings of the MICCAI 2005, LNCS 3749*, 107-114, 2005.

Ghosh, S.C., Sinha, B.P., Das, N. (2003). Channel assignment using genetic algorithm based on geometric symmetry, *IEEE Transactions on Vehicular Technology*, Vol. 52, No. 4 (2003) 860–875.

Glisic, S., Vucetic, B. (1997). *Spread Spectrum CDMA Systems for Wireless Communications*, Artech House Publishers, ISBN, UK.

Glover, F. (1986). Future paths for integer programming and links to artificial intelligence. *Computers and Operations Research*, Vol. 13 (1986) pp. 533-549.

Glover, F. (1989). Tabu Search, Part I, *ORSA Journal on Computing*, Vol. 1 (1989) 190-206.

Glover, F. (1990). Tabu Search, Part II, *ORSA Journal on Computing*, Vol. 2 (1990) 4-32.

Glover, F. (1992) Private communication.

Gudbjartsson, H. & Patz, S. (1995). The Rician distribution of noisy MRI data, *Magnetic Resonance in Medicine*, Vol. 34 (1995) 910–914.

Hansen, P. (1986). The steepest ascent mildest descent heuristic for combinatorial programming, *Proc. Congress on Numerical Methods in Combinatorial Optimization*, Capri, Italy, 1986.

Hertz, A. (1995). A tutorial on tabu search, *Proc. of Giornate di Lavoro AIRO'95 (Enterprise Systems: Management of Technological and Organizational Changes)*, pp. 13-24, Italy, 1995.

Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems*, University of Michigan Press (Ann Arbor).

Juntti, M.J., Schlosser, T., Lilleberg, J.O. (1997). GAs for multiuser detection in synchronous CDMA, *Proc. IEEE Int'l Symp. on Inf. Theory*, p. 492, 1997.

Kang, N., Zhang, J., Carlson, E.S., Gembris, D. (2005), White matter fiber tractography via anisotropic diffusion simulation in the human brain, *IEEE Transactions on Medical Imaging*, Vol. 24 (2005) 1127–1137.

Mitchell, M. (1996). *An Introduction to Genetic Algorithms*, The MIT Press, Cambridge, MA.

Mori, S. & van Zijl, P.C.M. (2002). Fiber tracking: principles and strategies – A technical review, *Nuclear Magnetic Resonance in Biomedicine*, Vol. 15 (2002) 468–480.

Oppacher F. & Wineberg, M. (1999). The Shifting Balance Genetic Algorithm: Improving the GA in dynamic environment, *Proc. Genetic and Evol. Comput. Conf.*, W. Banzhaf et al. (Eds.) Vol. 1, pp. 504–510.

Proakis, J.G. (1995). *Digital Communications*, McGrawHill Int'l Editions, New York.

San José-Revuelta, L.M. & Cid-Sueiro, J. (2003). Bayesian and RBF structures for wireless communications detection, *Proc. IEEE Workshop on Neural Networks and Signal Processing, NNSP'03*, pp. 749–758, Toulouse, France.

San José-Revuelta, L.M. (2005). Entropy-guided micro-genetic algorithm for multiuser detection in CDMA communications, *Signal Processing*, Vol. 85, No. 8, (August 2005) 1572-1587, ISSN 0165-1684.

San-José-Revuelta, L.M., Martín-Fernández, M., Alberola-López, C. (2007). A new proposal for 3D fiber tracking in synthetic diffusion tensor magnetic resonance images, *Proceedings of the IEEE Int'l Symp. on Signal Processing and its Applications, ISSPA'07*, Sharjah, United Arab Emirates, 2007.

San-José-Revuelta, L.M., Martín-Fernández, M., Alberola-López, C. (2007b). Neural-network assisted fiber tracking of synthetic and white matter DT-MR images, *Proceedings of the World Congress on Engineering 2007, WEC 2007*, pp. 618-623, London, United Kingdom, July 2007.

San-José-Revuelta, L.M., Martín-Fernández, M., Alberola-López, C. (2008). Efficient tracking of MR tensor fields using a multilayer neural network, *IAENG International Journal of Computer Science*, Vol. 35, No. 1 (2008) 129–139.

Shayesteh, M.G., Menhaj, M.B., Nobary, B.G. (2001). A new modified GA for multiuser detection in DS/CDMA sytems, *Proc. Comput. Intell., Theory and Appl., 7th Fuzzy Days*, 608-618, Dortmund, Germany, 2001.

Shayesteh, M.G., Menhaj, M.B., Nobary, B.G. (2003). A modified Genetic Algorithm for multiuser detection in DS/CDMA systems, *IEICE Trans. on Comm.*, Vol. E86-B, No. 8 (2003) 2377–2388.

Shimodaira, H. (1999). A Diversity Control Oriented Genetic Algorithm (DCGA): Development and experimental results, *Proc. Genetic and Evol. Comput. Conf.*, W. Banzhaf et al. (Eds.) Vol. 1 (1999) 603–611.

Tsutsui, S., Fujimoto, Y., Ghosh, A. (1997). Forking Genetic Algorithms: GAs with search space division schemes, *Evolutionary Computation*, Vol. 5 (1997) 61–80.

Ursem, R.K. (2002). Diversity-guided Evolutionary Algorithms, *Proc. Int'l Conf. on Parallel Problem Solving from Nature VII, PPSN VII*, pp. 462–471, Granada, Spain, 2002.

Verdú, S. (1998). *Multiuser Detection*, Cambridge University Press, Cambridge, UK.

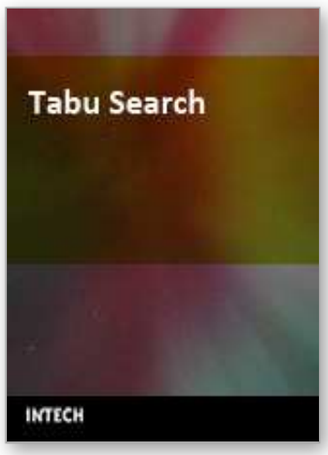
Wang, X.F., Lu, W.S., Antoniou, A. (1998). A Genetic Algorithm-based multiuser detector for multiple access communications, *Proc. IEEE Int'l Symp. on Circuits and Systems, ISCAS'98*, pp. 534-537, 1998.

de Werra, D. & Hertz, A. (1989). Tabu Search Techniques: networks, *OR Spektrum*, (1989) 131-141.

Westin, C.-F., Mainer, S.E., Mamata, H., Nabavi, A., Jolesz, F.A., Kikinis, R. (2002). Processing and visualization for diffusion tensor MRI, *Medical Image Analysis*, Vol. 6 (2002) 93–108.

Yen, K. & Hanzo, L. (2000). Hybrid GA-based multiuser detection schemes for synchronous CDMA systems, *Proc. Vehicular Technology Conf.*, pp. 1400-1404, Tokyo, Japan, 2000.

Yen, K. & Hanzo, L. (2001). Genetic Algorithm assisted joint multiuser symbol detection and fading channel estimation for synchronous CDMA systems, *IEEE J. on Sel. Areas in Comm.*, Vol. 19, No. 6 (2001) 985–997.



Tabu Search

Edited by Wassim Jaziri

ISBN 978-3-902613-34-9
Hard cover, 278 pages
Publisher I-Tech Education and Publishing
Published online 01, September, 2008
Published in print edition September, 2008

The goal of this book is to report original researches on algorithms and applications of Tabu Search to real-world problems as well as recent improvements and extensions on its concepts and algorithms. The book' Chapters identify useful new implementations and ways to integrate and apply the principles of Tabu Search, to hybrid it with others optimization methods, to prove new theoretical results, and to describe the successful application of optimization methods to real world problems. Chapters were selected after a careful review process by reviewers, based on the originality, relevance and their contribution to local search techniques and more precisely to Tabu Search.

How to reference

In order to correctly reference this scholarly work, feel free to copy and paste the following:

Luis M. San José-Revuelta (2008). A Hybrid GA-TS Technique with Dynamic Operators and its Application to Channel Equalization and Fiber Tracking, Tabu Search, Wassim Jaziri (Ed.), ISBN: 978-3-902613-34-9, InTech, Available from: http://www.intechopen.com/books/tabu_search/a_hybrid_ga-ts_technique_with_dynamic_operators_and_its_application_to_channel_equalization_and_fibe

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen