The Importance of the Individual Encoding in Memetic Algorithms with Diversity Control Applied to Large Sudoku Puzzles

Carlos Segura*, Eduardo Segredo† and Gara Miranda‡
*Centro de Investigación en Matemáticas, A.C. (CIMAT), Área de Computación
Jalisco S/N, Col. Valenciana, Guanajuato, 36023, México
Email: carlos.segura@cimat.mx
†School of Computing
Edinburgh Napier University
Edinburgh, Scotland, United Kingdom
Email: e.segredo@napier.ac.uk
‡Departamento de Ingeniería Informática y de Sistemas
Universidad de La Laguna
San Cristóbal de La Laguna, Spain
Email: gmiranda@ull.edu.es

***Abstract*—In recent years, several memetic algorithms with explicit mechanisms to delay convergence have shown great promise when solving 9x9 Sudoku puzzles. This paper analyzes and extends state-of-the-art schemes for dealing with Sudoku puzzles of larger dimensionality. Two interesting aspects are analyzed: the importance of the encoding and its relation with the way of managing the diversity. Specifically, three different ways of encoding the individuals and six different methods, including four that control the diversity in a special way, are studied. Computational results are shown with twenty 16x16 Sudoku puzzles. Contrary to the low-dimensional case, important differences appear among the several ways of controlling diversity. Specifically, a method that incorporates multi-objective concepts in the replacement phase to deal with the diversity, resulted in the most promising method. Results show that both the encoding and the way of managing diversity are crucial to attain high success probabilities in large Sudoku puzzles. They also show that, while the analyzed encodings induce different search space sizes, this feature is not enough to justify the differences in the performance attained by them.**

I. Introduction

Sudoku is a very popular and widespread pastime today. The origins of this number-based logic puzzle can be established within the works of the mathematician Leonard Euler who, in 1783, reported on the idea of *Latin Squares*: grids of equal dimensions in which every symbol occurs exactly once in every row and every column. The rules for Sudoku are simple: given an $N^2 \times N^2$ board which is divided into N^2 blocks of size $N \times N$, the aim is to fill each empty square so that the following criteria are met: some squares have already an assigned number which must be kept, each row of squares contains the integers 1 through to N^2 exactly once, each column of squares contains the integers 1 through to N^2 exactly once, and each $N \times N$ block contains the integers 1 through to N^2 exactly once.

When solving Sudoku problems, N is considered the order of the problem. The most popular variant of Sudoku is that with order $N = 3$, thus involving a grid of size 9×9. Since this problem is NP-Complete [1], larger grids might be too complicated to be fun, and as a result, most works analyze problems with $N = 2$ (4×4) or $N = 3$ (9×9), but only a few analyze instances of $N = 4$ (16×16) or $N = 5$ (25×25) [2]. Anyway, even with a 9×9 grid there are enough different Sudoku boards to provide plenty of fun: 6,670,903,752,021,072,936,960 to be exact [3]. Even if we were interested in counting only essentially different, non-equivalent Sudoku boards, the number is also rather large, namely 5,472,730,538 [4].

There are several commonly known techniques for solving Sudoku by hand. However, all puzzles are not that simple to be solved by hand, so various algorithms have been developed. Some of them mimic the way a human would solve the puzzle, but for harder puzzles, where guessing is required, those algorithms do not suit well. Some algorithms perform exhaustive searches but for harder puzzles they will take longer to solve. Due to the constraint nature of Sudoku, it is very likely to find a solution that is a local optimum, under typically used neighborhoods. When stochastic techniques are used, the solution space is still searched even though a local optimum has been found, allowing for the global optimum to be detected in some cases, so a solution to this problem could lie in using stochastic optimization techniques [5]. In fact, most popular meta-heuristics have been adapted to tackle the Sudoku problem [6], [7]. In the particular case of Evolutionary Algorithms (EAs), its basic variants can solve simple Sudoku puzzles [8], although they encounter significant difficulties when tackling more difficult instances [9], [10], [11]. In order to alleviate these drawbacks, hybrid variants and ad-hoc

genetic operators have been proposed [12], so incorporating problem-specific knowledge seems to be quite important for designing successful EAs for Sudoku.

Adapting EAs to new problems is not an easy task, as this usually involves many difficult design decisions [13]. Particularly, premature convergence has been recognized as one of their recurrent drawbacks [14], so when applying EAs to new problems, special attention must be paid to this issue. Maintaining a diverse population appears as an alternative to alleviate this problem. In [15] six mature methods that were designed to control the diversity of the population [16] were successfully incorporated into a simple *Memetic Algorithm* (MA) that uses some of the first genetic operators ever designed. In addition, in [15], a more recent method—proposed in [17], [18]—was also used. It combines the idea of transforming a single-objective problem into a multi-objective one by considering diversity as an explicit objective, with the idea of adapting the balance induced between exploration and exploitation to the various optimization stages. In this work, the four different methods of controlling the diversity that attained the most promising results are analyzed together with three different individual encodings with larger Sudoku puzzles. Additionally, two commonly used methods that do not control diversity in a special way are also taken into account. The individual encoding is somehow related to how the search space will be explored, so it is mandatory to choose a codification which can ensure a proper space exploration. In this work two interesting aspects are analyzed: the importance of the encoding and its relation with the way of managing the diversity. For the study, 16x16 Sudoku puzzles are taken into account rather than the most common and human-affordable 9x9 puzzles.

The rest of the paper is organized as follows. A discussion of the relevant background, including EAs for Sudoku and some issues related to the encoding of individuals, is given in Section II. In Section III, the solver designed for Sudoku is described. Not only the general operation of the MA is presented, but also the different proposals regarding the individual encoding and diversity management. Then, our experimental validation is presented in Section IV. Finally, conclusions and some lines of future work are given in Section V.

II. Literature Review

A. Meta-heuristics for Sudoku

Since the popularization of Sudoku, several different stochastic optimization algorithms have been applied to solve these types of puzzles [6], [7]. From among all different meta-heuristics, EAs are likely to be the most popular approaches. In the initial proposals the genetic operators applied were quite straightforward [8], [19], and therefore, good results were achieved only for simple Sudoku puzzles. However, in order to afford harder puzzles an important research line in this field has focused on the design of more complex crossover and mutation operators [9], [20], [21], [10]. Due to these weaknesses of general operators, genetic operators that take into account specific knowledge of the problem have been developed [22], [11], [23], [24].

Another research line in this field involves hybridizing EAs with other population-based meta-heuristics. For instance, in [25], some principles of *Particle Swarm Optimization* are incorporated into an EA, while in [12], EAs and *Ant Colony Optimization* are applied together, which is a well-known stochastic optimizer for Sudoku. In addition, it is very typical to integrate local searches or individual learning procedures into EAs. In most cases, simple definitions of neighborhoods are used. One of the most typical approaches is to generate each neighbor by simply swapping two positions of a subchromosome [26], [24]. Incorporating a local search greatly reduces the amount of time required to converge. In this work, we incorporate a simple stochastic hill climbing method.

Finally, it has been shown that using some of the traditional schemes for avoiding premature convergence by controlling the diversity of the population provide important benefits in the solving of small Sudoku puzzles [15]. In fact, schemes that do not include any mechanisms to alleviate premature convergence yield quite disappointing results [6], [27]. whereas some of the most effective solvers do in fact include mechanisms to avoid premature convergence. For instance, in [12], three different mechanisms to alleviate premature convergence are included: aging, a restarting mechanism, and a special sorting method that promotes the selection of complementary individuals. Furthermore, the advantages of imposing mating restrictions, which seek to decelerate the loss of diversity, are shown in [10]. More evidence regarding the above is provided in [28], where the *parallel independent runs model* [29] was successfully applied to the Sudoku problem. This model is more successful when the approach is highly dependent on the initial population, which is highly related to the appearance of premature convergence.

B. Importance of Individual Encoding

Since different problem instances induce different landscapes and complexities, whilst different individual encodings may lead to different solution spaces, the choice of a suitable encoding or representation is a challenging task. For the Sudoku problem different alternatives have been proposed. In most cases, the search space for the Sudoku problem is defined by all those $N^2 \times N^2$ grids which fulfill two of the four given constraints: prefixed square constraint and any one of the other three constraints (rows, columns or blocks). The degree of fulfillment of the remaining constraints defines the fitness value of the grid. For instance, in [8], a representation using the prefixed squares and the divisions by blocks, so that the only constraints that must be taken into account are those associated with rows and columns, is devised. Note that the generation of solutions that take into account three constraints simultaneously is NP-Complete [9], and consequently, the encodings that take into account two kinds of constraints are the most popular ones. The theoretical and empirical advantages of this encoding compared to those where no constraints are considered is shown in [9].

Some direct or grid-based encodings make use of a group table [30], [31]. In [31], preset squares and blocks criteria are made the hard constraints to be satisfied. Moreover, an additional condition was imposed in order to characterize the search space: any empty square has only limited probable candidates. In this way, for each empty square there is a group of possible values which do not enter in conflict with the given constraints. The individuals produced by the algorithm will only be those whose empty squares are filled in with one of the probable numbers that can be placed there, thus reducing the search space. In [30], a multistage genetic algorithm also applied a group table to create an initial random population. The said table got updated in each cycle, since it was also used to perform mutation.

Other possibility consists of representing the Sudoku board as a graph with $N^2 \times N^2$ vertices (one for each square of the board), together with edges. Two vertices are connected by an edge if the squares that they correspond to are in the same column, row or block. Then, each number from 1 to N^2 is assigned a color, so that vertices containing a given number (prefixed squares) will be colored with the corresponding color of the number. Afterwards, completing the Sudoku puzzle without violating the constraints is equivalent to coloring the vertices of the corresponding graph, while ensuring that two adjacent vertices are not assigned the same color. A Sudoku puzzle can be thus presented as a graph coloring problem [32], [33]. Related to this graph-based representation, the Sudoku can be represented as a system of polynomial equations, by considering pairs of squares that share a region rather than by considering entire regions at a time [34]. After the transformation, any of the encodings that have been used for solving these related problems might be used to tackle the Sudoku.

Another important issue for the performance of EAs is the definition of the fitness function. Such definition highly depends on the individual representation being used. For instance, when using grid-based representations, the sum of each column, row and grid must be equal to $\sum_{i=1}^{N^2} i$ and its product must be equal to $(N^2)!$. One possible fitness function implements these arithmetic operations to ensure that the constraints are met, although the non-repetition of an integer in the same column, row or grid is not guaranteed. This could cause the algorithm to converge to a solution that does not meet all the constraints. For this reason, the non-compliance with the restrictions must be somehow penalized. In [5], a fitness value is assigned to a possible solution based on the number of repeated or non-present integers. In [8] a weighted sum of several conditions that must be fulfilled in a valid Sudoku solution is considered. Subsequently, it was shown that such complexity was not justified and that by simply penalizing the repetitions of numbers in rows, columns and blocks, similar or even better results might be obtained [22]. In these initial definitions, the starting numbers given in the board were just ignored because the encoding of solutions did not allow altering them. In [35], the results of a best-first search could be improved by additionally penalizing the conflicts with the given values. The advantages of these types of penalties have been confirmed in several variants of EAs [36], [23].

Algorithm 1 Diversity-based Memetic Algorithm with Lamarckian Individual Learning Procedure

1: **Initialization**: Assign $t = 0$. The initial population P_0 is filled with N individuals produced at random.
2: **Individual learning**: For each individual in P_0, the learning procedure is applied.
3: **while** (stopping criterion is not satisfied) **do**
4: **Evaluation**: Every individual in P_t is evaluated.
5: **Mating selection**: The mating pool is filled by performing binary tournament selection on P_t.
6: **Variation**: The offspring population CP is obtained through the application of the variation operators to the mating pool.
7: **Individual learning**: Every offspring in CP undergoes the learning procedure.
8: **Survivor selection**: P_t and CP are combined and the surviving individuals selected by the replacement scheme MULTI_DYN define P_{t+1}.
9: $t = t + 1$
10: **end while**

III. Algorithmic Proposals

This section is devoted to describe the algorithmic schemes considered in the current work. Firstly, in Section III-A, the MA used as our main evolutionary engine is detailed. Then, Section III-B introduces the different encodings used to represent Sudoku solutions. Since the variation operators and the individual learning procedure depend on the particular individual encoding, they are also presented at this point. Finally, different approaches to deal with premature convergence, including those applied together with the aforementioned MA, are described in Section III-C.

A. Memetic algorithm

The MA considered in this work, which was proposed in [15], consists of an EA and a stochastic hill-climbing individual learning procedure. Novel variation operators and neighborhood definitions to address Sudoku puzzles are not proposed herein. Instead, some simple mechanisms, which can be already found in the related literature, are taken into account. We selected those simple approaches because they have not usually been able to deal with hard Sudoku instances. One of the aims is thus demonstrating that by using a suitable diversity preservation mechanism, general and simple operators, rather than knowledge-based and complex ones, can be applied, even when using large Sudoku puzzles.

The pseudocode of the particular MA considered herein is shown in Algorithm 1. It is a standard EA combined with a *Lamarckian* individual learning procedure (steps 2 and 7). Moreover, it makes use of a diversity-based survivor selection strategy (step 8), which will be referred to as MULTI_DYN in the rest of the paper. The operation of that strategy will be detailed in Section III-C.

In order to evaluate a particular individual (step 4 of Algorithm 1), the objective function takes into consideration constraint violations at block, row, and column levels. In order to simplify their description, only constraint violations at blocks are explained in the following lines, but we note that those constraint violations may also occur considering rows and columns. Particularly, constraint violations might

be classified in two different types. In the first type, which arises when one of the empty squares of a particular block is assigned a number that was initially set on that block, i.e., a number originally given by the instance, the value 100 is added to the objective function. The second type of constraint violation consists of the number of repetitions at blocks that are not classified into the first type, which is also added to the objective function.

B. Individual Encoding, Variation Operators, and Individual Learning Procedure

Since in the current paper the main goal is to analyze the importance of the individual encoding and its relation with diversity preservation, this section is devoted to describe the different representations we take into account. Particularly, three different encodings are applied herein.

In the first representation [8], which is likely to be the most popular one, η sub-chromosomes, with η being the size of the Sudoku board, define a given individual. Hence, each sub-chromosome represents a particular block of the board. At the same time, each sub-chromosome consists of a permutation of integer numbers whose size is equal to the number of empty squares in the corresponding block. Integer values are in the range $[1, \eta]$. We note that values that are preset in a particular block of the board, i.e., those values originally given by the Sudoku instance, can not be used in the corresponding permutation of that block. In the case of this work, 16x16 Sudoku instances are addressed, and consequently, an individual consists of $\eta = 16$ sub-chromosomes. Finally, it is important to mention that for a given instance, the different sub-chromosomes might have different sizes, since the number of empty squares varies depending on the particular block.

The second and third encodings only differ from the previous one in what each sub-chromosome represents. In the second encoding, each sub-chromosome consists of a permutation representing the numbers assigned to the empty squares of a particular row, while in the third one, permutations define the values assigned to the empty squares of a given column.

Fig. 1 depicts the encoding of a particular solution through the above three approaches. Red values are those assigned to empty squares, while black values are those preset by the instance. A 9x9 board is considered due to space restrictions. In this particular case, $\eta = 9$ different sub-chromosomes would define a complete individual. Due to the same aforementioned reason however, only encodings for two of them are illustrated.

One important observation is that the size of the search space changes depending on the individual encoding selected. The size of the search space in terms of the board size η and the size of each sub-chromosome s_i, with $i = 1 \dots \eta$, can be calculated by means of (1). For example, if we consider the instance detailed in Fig. 1, the size of the search space regarding the block-based encoding is equal to $7.8640006e + 22$. In the case of the row-based and column-based encodings, sizes are equal to $1.9266801e + 23$ and $7.8640006e + 22$, respectively.

Algorithm 2 The replacement scheme MULTI_DYN

1: CurrentMembers = Population ∪ Offspring
2: Best = The fittest individual in CurrentMembers in terms of the original objective function
3: NewPop = { Best }
4: CurrentMembers = CurrentMembers - { Best }
5: **while** (|NewPop| < N) **do**
6: Considering NewPop as the reference set, the *Distance to the Closest Neighbor* (DCN) is calculated for each individual in CurrentMembers.
7: $D = D_I - D_I * \frac{T_{Elapsed}}{T_{End}}$
8: Penalize(CurrentMembers, D)
9: ND = Non-dominated individuals in CurrentMembers (without repetitions)
10: Selected = Select an individual from ND at random
11: NewPop = NewPop ∪ Selected
12: CurrentMembers = CurrentMembers - {Selected}
13: **end while**
14: Population = NewPop

$$\prod_{i=1}^{\eta} s_i! \tag{1}$$

With regard to the variation scheme, only a crossover operator is applied at step 6 of Algorithm 1. This MA does not apply a mutation operator because, besides the fact that mutation is not always considered for the variation scheme of MAs, a preliminary analysis showed that its usage did not provide any advantage, and therefore, it was removed to keep our approach as simple as possible. The crossover operator, which is applied in every generation, exchanges complete blocks/rows/columns—depending on the particular encoding—between solutions uniformly, i.e., a typical uniform crossover is considered, but instead of operating at the gene level, it operates at the sub-chromosome level.

Finally, the individual learning procedure is very straightforward. A neighbor of a candidate solution is generated by swapping two elements of a block/row/column. Then, using that definition of neighborhood, a stochastic hill-climbing is applied, i.e., neighbors are considered in random order and only movements resulting in an improvement of the objective function are accepted. The above steps are repeated until a local optimum is reached.

C. Diversity Management Mechanisms

A novel survivor selection mechanism termed as MULTI_DYN, which considers diversity in an explicit way, was recently proposed [17], [37]. Furthermore, it was integrated into the MA introduced in previous sections, with the aim of solving 9x9 Sudoku instances [15]. This section aims to detail the operation of the replacement scheme MULTI_DYN, whose pseudocode is shown in Algorithm 2.

First of all, the fittest individual—regarding the original objective function—from among parents and offspring (*CurrentMembers*) is selected as one of the survivors (steps 1–4). Afterwards, while the surviving population *NewPop* is not filled with N individuals, being N the population size, the following steps are repeated (step 5). Basically, the non-dominated set that considers the original objective function, and at the same time, the diversity contribution of each individual in *CurrentMembers* is determined (step 9). The

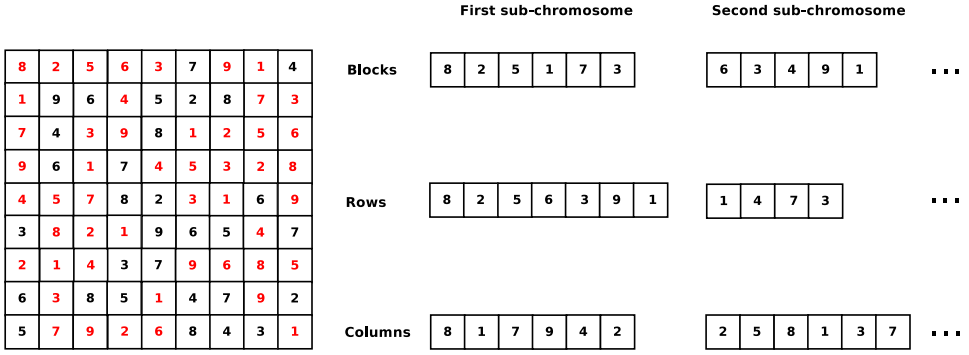

Fig. 1. Encoding of a Sudoku solution by means of three different representations

diversity contribution of a given individual is calculated as its *Distance to the Closest Neighbor* (DCN) belonging to the set of individuals that have already been selected to survive *NewPop* (step 6). Considering our different individual encodings, we selected the *Hamming distance*. The distance between two individuals is thus given by the number of differences between their corresponding chromosomes, i.e., in the decision variable space. Finally, an individual uniformly chosen at random from the non-dominated set becomes a new survivor (steps 10–12).

It is worth explaining at this point the operation of steps 7 and 8 of Algorithm 2. As it can be observed, the stopping criterion T_{End} and the elapsed time $T_{Elapsed}$ are used by the replacement scheme (step 7) with the aim of balancing its diversification and intensification abilities. For doing that, a mechanism that penalizes individuals which do not significantly contribute in terms of diversity, regardless of their original objective value, is applied (step 8). The minimum DCN value that a given individual must satisfy is represented by D. If that DCN value is lower than D then the individual is penalized. The penalty consists of assigning a quite high value to its original objective value. By means of the said penalty, its non-domination rank may be increased, and in that case, its probability to survive would be decreased.

We note that an initial value D_I has to be fixed, in such a way D is then decreased linearly until $D = 0$. Bearing the above in mind, at the beginning of a run, MULTI_DYN promotes diversification. At the end of the run, the balance is moved towards intensification.

In addition to the replacement scheme MULTI_DYN, other methods are also taken into account herein for comparison purposes. In particular, we apply two replacement schemes that do not include any mechanism to delay convergence. The first one is a *generational approach with elitism* (GEN_ELIT), where the best individual in the current generation survives to the next one, while remaining individuals are created by means of the variation operators. The second one is the *replace-worst* strategy (RW), which is a more elitist version. In this case, the offspring and parent populations are joined in a set with $2 \cdot N$ individuals. Afterwards, the N fittest individuals, from among parents and offspring, are selected to survive.

At the same time, three approaches that incorporate specific mechanisms to manage diversity are also used as comparison methods. They are some of the approaches providing the best results in [15] with smaller Sudoku puzzles: *Restricted Tournament Selection* (RTS) [38], COMB [39], and the *Saw-Tooth Genetic Algorithm* (Saw-Tooth-GA) [40]. In RTS, which is a *crowding* method, an individual C is produced whereas CF individuals are selected at random from the current population. Then, C and its most similar individual included in CF compete to survive trough a traditional binary tournament. In COMB, where maximizing diversity is considered as an aim of the optimization process, individuals are sorted, first, by their original objective function, and second, by their contribution to diversity. Afterwards, both rankings are combined to calculate the final fitness value of a particular individual via two different parameters: N_{Close} and N_{Elit}. The individual with the lowest fitness is erased and rankings are re-calculated at each step of the replacement strategy. Finally, the *Saw-Tooth* GA, includes a diversity preservation mechanism in the form of a restarting scheme. Specifically, it considers a population of variable size, which is initialized periodically, by following a saw-tooth function. The said function is parameterized by means of its period P and amplitude D.

IV. Experimental Validation

In this section we describe the set of experiments that have been carried out to analyze the importance of the individual encoding and its relation with the diversity management in the solving of large Sudoku puzzles. The *Meta-heuristic-based Extensible Tool for Cooperative Optimization* (METCO) [41] was used to implement our proposals. Experiments were run on bi-processor machines with 32 Gb RAM. Each processor is an Intel(R) Xeon(TM) CPU E5-2620 at 2.10 GHz. The analyses were performed with 20 different Sudoku puzzles. Since most 9x9 Sudoku puzzles are solvable with several MAS [15], larger puzzles with sizes equal to 16x16 have been used. Specifically, a set with 20 puzzles was used. They were generated with two of the most popular websites dedicated to Sudoku [42], [43][1]. In the case of the Sudoku puzzles generated in [42] (tagged as ES), the *Diabolique level* was

[1]The set of instances can be downloaded in http://www.cimat.mx/~carlos.segura/Sudoku/16x16SudokuPuzzles.tar.gz

selected, whereas for the ones generated through [43] (tagged as TS), the Expert level was used. In both cases, the level selected was the most difficult from among those available.

Since stochastic algorithms were considered, each execution was repeated 100 times. In this kind of problem, the objective is to completely solve the puzzle. Thus, when the puzzle is not solved, there is no point in comparing the resulting fitness values. Consequently, the analyses shown in this paper were done in base of the success probability in achieving a solution of the puzzle. The test described in [44] was applied to statistically compare the success probabilities attained by the different approaches.

The experiments are presented in two different subsections. The first one is devoted to analyze the importance of the encoding in the application of MAs to the solving of Sudoku. Since the MULTI_DYN scheme presented the best performance with the most difficult 9x9 Sudoku puzzles in [15], this scheme is used to carry out this analysis. The second subsection is aimed to study the performance attained by several well-known diversity-management schemes and to analyze the relations between the encoding and the management of diversity.

A. Importance of Encoding

In order to check the importance of the representation of individuals for the Sudoku, the MULTI_DYN scheme was used by incorporating the three previously discussed representations: block permutations (BP), row permutations (RP), and column permutations (CP). The MULTI_DYN scheme requires two parameters: the population size (N) and the initial distance used in the penalty approach (D_I). The population size was set to 100 because it is a very typical value and because it has reported a promising behavior in smaller Sudoku puzzles [15]. In the case of D_I, five different values were tested: 0, 5, 10, 20, and 30. Additionally, in every case the stopping criterion T_{End} was set to 5 minutes. Every D_I value reports quite a similar performance when taking into account the mean success probability obtained with all the instances and representations. Fig. 2 shows the mean of the success probability with the different values of D_I. The most adequate value reported a mean success probability equal to 69.33%, whereas the success probability attained with the worst parameterization was 65.95%. Thus, the MULTI_DYN scheme is not very sensible to its parameterization. In fact, with D_I equal to 0, promising results are attained, meaning that most of the benefits arise from the use of a multi-objective replacement, whereas the activation of the penalty approach (D_I values different to 0) just refines the method. In any case, since the value 10 reported the highest mean success probability, it was selected to report our remaining results.

Table I shows, for the 20 Sudoku puzzles, the success probability (column SP) attained by MULTI_DYN with each encoding. In each instance, the largest success probability is shown in **bold face**. In addition, the success probabilities of the schemes whose differences, when compared to the best one, were not statistically significant are also shown in **bold face**. It is quite clear that, selecting a proper encoding is a very important step to properly solve the puzzles. For instance, when taking into account the TS5 puzzle, MULTI_DYN with the BP representation attained a success probability equal to 83%; however, if the encoding applied is RP, then the success probability decreases to 9%, which is a quite remarkable difference in performance. Moreover, for each kind of encoding, there is at least one instance where the application of such an encoding provokes an important decrease in performance when compared to the best encoding. In fact, no one of the schemes appears in **bold face** in every instance. Thus, selecting always the same encoding does not seem the best way to proceed. In any case, it is remarkable that the BP encoding attains quite a promising performance. In fact, it obtained the largest success probability in 17 out of the 20 instances, and only in one of the instances (ES9) where the success probability was not the largest one, there were statistically significant differences. Thus, it seems that there is a bias that favors the application of the BP encoding. However, the existence of ES9 confirms that this is not a completely general pattern. Since the puzzles were created by taking into account only two generators, it is unknown for us if this bias comes from the way in which the instances are generated or if it is a feature of the problem itself. Obtaining more general conclusions would require the generation of many puzzles from several sources, as well as knowing the way in which they are created, which is usually not reported. As a result, that study is out of the scope of the current work.

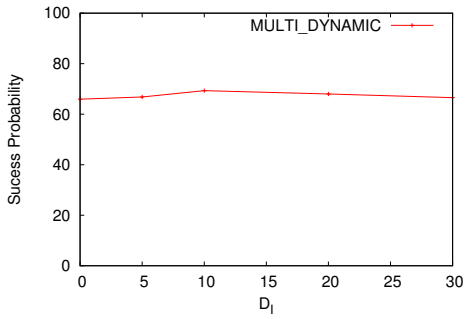

Fig. 2. Mean Success Probability attained with different values of D_I

Another interesting fact is that the different encodings induce different search space sizes. Initially, we hypothesized that these sizes might be related to the performance obtained by the different encodings. In order to check this hypothesis, the sizes induced by each encoding in the different puzzles are also reported in Table I. The lowest search space sizes are shown in **bold face** for each instance. First, we could find out that differences in the sizes are noticeable in several cases. For instance, in the ES1 instance, the size induced by the RP encoding is 57.06 times larger than the one induced by the CP encoding. In 11 out of the 20 instances, the encoding that reported the best performance is the one that induced the lowest search space. However, this means that in 9 out of the 20 instances, this is not the case. Thus, it is clear that using only the search space size to predict the overall performance

TABLE I
SUMMARY OF THE SEARCH SPACE SIZE AND RESULTS OBTAINED BY MULTI_DYN WITH DIFFERENT REPRESENTATIONS

	Blocks Permutations		Rows Permutations		Columns Permutations	
Inst.	Size	SP	Size	SP	Size	SP
ES1	4.53e+93	**95%**	2.14e+95	27%	**3.75e+93**	**98%**
ES2	**3.36e+89**	**100%**	1.23e+91	54%	1.16e+90	**100%**
ES3	**2.01e+95**	**98%**	3.58e+95	93%	2.83e+97	31%
ES4	**2.10e+85**	**100%**	1.52e+87	**100%**	1.52e+87	23%
ES5	4.63e+91	**100%**	**2.60e+91**	**100%**	**2.60e+91**	95%
ES6	**1.66e+93**	**100%**	8.75e+93	61%	1.02e+96	35%
ES7	**4.90e+85**	**99%**	2.72e+86	**97%**	2.72e+86	90%
ES8	**3.28e+87**	**100%**	5.31e+87	84%	5.31e+87	**100%**
ES9	**3.36e+89**	59%	5.42e+89	**97%**	1.61e+95	51%
ES10	**7.50e+89**	**100%**	9.23e+91	45%	3.39e+91	74%
TS1	8.84e+106	**98%**	**4.93e+106**	57%	2.29e+107	90%
TS2	3.02e+105	**52%**	1.40e+106	3%	**2.28e+105**	33%
TS3	3.89e+102	**100%**	1.95e+103	90%	**3.59e+102**	81%
TS4	8.95e+109	**92%**	**4.88e+109**	33%	1.18e+110	52%
TS5	**4.20e+104**	**83%**	1.62e+105	9%	1.07e+105	50%
TS6	**2.44e+106**	**45%**	6.71e+106	**54%**	8.05e+106	17%
TS7	8.88e+105	**27%**	4.11e+105	15%	**3.86e+105**	**21%**
TS8	**2.81e+108**	**76%**	2.01e+109	47%	1.18e+109	**70%**
TS9	5.13e+99	**100%**	**4.06e+99**	83%	1.71e+100	64%
TS10	3.79e+103	**93%**	4.20e+103	30%	**3.62e+103**	**89%**

of each encoding is not appropriate. Note that, the relation between search space size and performance seems to be clearer in the ES puzzles. In this first group, the best encoding was the one with lowest search space size in 9 out of 10 cases. However, this means that in the TS puzzles, this only happened in 2 out of 10 cases. Moreover, there are cases, such as the TS4 instance, where the degradation of the performance when using the encoding that induces the lowest search space is quite large. In fact, in such a case, the RP encoding attains a success probability equal to 33%, whereas the BP encoding, which induces a search space that is 1.83 times larger, achieves a success probability equal to 92%. Additionally, in the case of the TS1 puzzle, the encoding with the lowest search space size is the one that reports the worst results among the three encodings. Thus, properly selecting the encoding is a quite important step. However, our results show that calculating the search space size to select the encoding is just a heuristic that fails in several cases.

B. Diversity Management

Previous analyses [15] have shown that several state-of-the-art MAs with diversity management strategies are capable of properly solving 9x9 Sudoku puzzles. Since the problem is NP-Complete, it is expected that with puzzles of larger sizes, the performance of them might degrade in different ways. The main aim of this section is two-fold. First, to analyze the performance of the diversity management strategies that reported the best results with 9x9 Sudoku puzzles, when applied to larger puzzles. Second, to analyze the relative importance between the diversity management strategies and the encodings.

In order to carry out this analysis, all the schemes described in Section III-C were taken into account. Since all of them require the setting of additional parameters, different parameterizations were tested. The set of parameters used in the preliminary studies are reported in Table II. In each case, the parameterization that is shown in **bold face** is the one that attained the largest success probability, when taking into account the whole set of instances.

TABLE II
PARAMETRIZATION OF THE METHODS APPLIED TO SUDOKU

Method	Parametrization
COMB	$N_{Close} = \mathbf{1}, 3, 8; N_{Elit} = \mathbf{1}, 3, 8$
GEN_ELIT	No parametrization required
MULTI_DYN	$D_I = 0, 5, \mathbf{10}, 20, 30$
RTS	$CF = 25, 50, 75, \mathbf{95}, 100$
RW	No parametrization required
SAW-TOOTH-GA	$D = 50, 75, \mathbf{99}, P = \mathbf{25}, 50, 200, 500$

Table III shows the success probabilities obtained with each scheme and individual encoding, when taking into account the best found parameterization. As in previous tables, for each instance the largest success probability is shown in **bold face**. In addition, the data obtained by schemes whose differences were not statistically significant when compared to the best approach are also highlighted. Taking into account all these results several conclusions can be drawn. In order to facilitate the analyses, the last two rows summarizes the results. The second-to-last row shows for each scheme and encoding, the mean success probability. This shows a clear superiority of the MULTI_DYN scheme when compared against the other methods. MULTI_DYN with the BP encoding attained a success probability equal to 85.85%, whereas the second best scheme obtained a success probability equal to 75.05%. Moreover, it is clear that in order to obtain promising results, a diversity-preservation method is required. The first four listed methods, which incorporate diversity-preservation, obtain mean success probabilities larger than 50% with a proper encoding, whereas RW and GEN_ELIT attained much lower mean success probabilities regardless of the encoding. However, in some way, once that any diversity preservation scheme is selected, the election of a proper encoding becomes crucial. In fact, the performance of MULTI_DYN, which is the best-performing method, degrades a lot when considering an encoding different to BP. For instance, MULTI_DYN with RP attains a success probability equal to 58.95%, which is inferior to the success probabilities attained by RTS and COMB when taking into account the BP encoding. Finally, the last row shows, for each proposal, the amount of instances where at least one encoding was similar in performance to the scheme with largest success probability. The obtained values confirm the superiority of MULTI_DYN. In fact, MULTI_DYN with a suitable encoding was never statistically exceeded by any other variant.

Finally, it is also remarkable that the most suitable encoding is similar in all the best-performing schemes. For instance, in ES4, MULTI_DYN, RTS and COMB attain high-quality results with the BP and RP encoding, whereas the CP encoding degrades the performance in all of them. Similar circumstances arises in most of the instances, meaning that there is not a large dependency between the way of managing diversity and the most adequate encoding.

TABLE III
SUCCESS RATES ATTAINED WITH DIFFERENT WAYS OF MANAGING DIVERSITY

Inst.	MULTI_DYN			RTS			COMB			SawTooth-GA			RW			GEN_ELIT		
Inst.	BP	RP	CP	BP	RP	CP	BP	RP	CP	BP	RP	CP	BP	RP	CP	BP	RP	CP
ES1	**95**	27	**98**	87	30	76	91	28	84	84	14	74	10	0	3	4	1	0
ES2	**100**	54	**100**	68	32	68	78	33	77	25	15	43	3	0	2	1	2	2
ES3	**98**	93	31	76	59	11	94	66	19	2	16	0	5	2	0	6	5	0
ES4	**100**	**100**	23	89	97	12	**99**	**100**	18	1	76	0	0	5	1	0	4	0
ES5	**100**	**100**	95	**99**	**100**	91	**100**	**100**	86	**100**	**100**	**100**	70	48	34	63	42	22
ES6	**100**	61	35	**100**	47	25	**100**	44	31	84	7	6	6	0	0	18	0	0
ES7	**99**	97	90	97	49	83	94	64	77	**100**	47	75	10	1	7	18	2	8
ES8	**100**	84	**100**	**100**	68	**99**	**100**	75	**99**	**100**	97	86	89	4	25	80	11	15
ES9	59	**97**	51	67	78	53	52	77	37	13	4	54	1	3	1	4	1	4
ES10	**100**	45	74	96	39	67	**98**	36	66	44	10	26	4	0	1	3	1	1
TS1	**98**	57	90	92	31	57	85	27	48	71	28	13	0	0	0	8	0	2
TS2	**52**	3	33	39	6	17	37	4	9	15	0	6	0	1	1	6	0	3
TS3	**100**	90	81	95	39	62	98	43	53	92	10	33	2	0	0	12	0	3
TS4	**92**	33	52	80	26	59	63	19	29	**93**	10	48	5	0	0	24	1	2
TS5	**83**	9	50	25	3	10	32	5	20	3	0	2	0	0	0	0	0	0
TS6	**45**	**54**	17	**44**	26	18	41	23	7	31	0	12	0	0	1	6	0	0
TS7	**27**	15	**21**	**18**	7	**18**	**18**	6	7	4	1	1	0	0	0	0	0	0
TS8	**76**	47	**70**	61	18	56	55	14	37	49	7	27	1	0	1	11	1	1
TS9	**100**	83	64	90	45	29	88	47	28	**100**	30	26	3	0	1	37	3	3
TS10	**93**	30	**89**	78	37	61	72	33	52	54	7	53	5	1	1	11	1	2
Mean	85.85	58.95	63.2	75.05	41.85	48.6	74.75	42.2	44.2	53.25	23.95	34.25	10.7	3.25	3.95	15.6	3.75	3.4
Wins	20			5			6			5			0			0		

V. CONCLUSIONS AND FUTURE WORK

Many studies have shown that 9x9 Sudoku puzzles are solvable with several variants of MAs. Particularly, variants that incorporate mechanisms to delay the convergence of the population have been the most promising ones. However, since the Sudoku is an NP-Complete problem, increasing the dimensionality of the puzzles provokes important drawbacks in the methods applied for lower dimensionalities. This paper focuses on analyzing the importance of the individual encoding in MAs when applied to 16x16 Sudoku puzzles. Since the best results in lower dimensions have been obtained with proposals that incorporate specific mechanisms to control the loss of diversity, our studies focus on those kinds of proposals, and the relations between diversity mechanisms and encoding are also studied. Specifically, the MULTI_DYN, RTS, COMB and Saw-Tooth-GA, which are methods that can readily solve 9x9 Sudoku puzzles are taken into account. Some methods that do not rely in a special control of diversity are also included for comparison purposes. These schemes are studied with three different individual encodings. They are based on representing permutations with the missing values in blocks, rows, and columns, respectively.

Computational results with 20 instances—that were created by using two different generators—have been used to validate the proposals. Results show that both the individual encoding and the way of controlling the diversity are important to attain high-quality results. The incorporation of a diversity control mechanisms is somewhat more important that the encoding. In fact, methods that do not incorporate any mechanism to control diversity, obtain a poor performance with any of the tested encodings. However, when even simple ways of controlling diversity are incorporated, the selection of the encoding becomes a crucial step. In fact, our experimental validations show that simple strategies to control the diversity might obtain a better performance than more complex strategies with a not properly selected individual encoding. The block-based encoding demonstrates a remarkable behavior when taking the mean performance into account. However, it is not the winning strategy in every case and the statistical tests show its inferiority in one case, meaning that selecting always the same encoding is not the best way to proceed. It could be verified that for the best-performing schemes, the relative performance between the different encodings does not vary, i.e. the selection of the most appropriate encoding does not depend on the way of delaying the convergence. An analysis of the search space sizes associated to the different encodings reveals that selecting the one with the lowest size is not appropriate in every case. Thus, it is an open topic to develop a mechanism to automatically select the most adequate encoding. Finally, our experimental validations shows that, while in low dimensionalities all the schemes perform similarly except in some of the most complex puzzles, in higher dimensionalities, there are significant differences among the schemes. In fact, MULTI_DYN significantly exceeded any other techniques in more than half of the instances.

Several lines of future work might be explored. First, given the differences in performance attained with the various encodings tested in this paper, it seems promising to develop methods that use the different encodings simultaneously. In other problems, heterogeneous island-based models have been successfully applied to integrate different encodings and/or operators, so this line of research seems to be very promising. Additionally, developing a method to properly select the encoding by inspecting the features of the instances before the run starts, seems encouraging. Finally, we would like to increase even further the dimensionalities of the puzzles, with the aim of better inspecting the limitations of the diversity-based MAs.

ACKNOWLEDGMENTS

This work has been supported by the Spanish Ministry of Economy, Industry and Competitiveness inside the program "I+D+i Orientada a los Retos de la Sociedad" with contract number TIN2016-78410-R.

REFERENCES

[1] T. Yato and T. Seta, "Complexity and Completeness of Finding Another Solution and Its Application to Puzzles," *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.*, vol. E86-A, no. 5, pp. 1052–1060, 2003.

[2] T. Cazenave, "A search based sudoku solver."

[3] B. Felgenhauer and F. Jarvis, "Mathematics of Sudoku I," *Mathematical Spectrum*, vol. 39, p. 15–22, 2006.

[4] E. Russell and F. Jarvis, "Mathematics of Sudoku II," *Mathematical Spectrum*, vol. 39, p. 54–58, 2006.

[5] M. Perez and T. Marwala, "Stochastic optimization approaches for solving sudoku," *CoRR*, vol. abs/0805.0697, 2008. [Online]. Available: http://arxiv.org/abs/0805.0697

[6] J. Jilg and J. Carter, "Sudoku evolution," in *2009 IEEE International Games Innovations Conference*, Aug 2009, pp. 173–185.

[7] L. Clementis, "Advantage of parallel simulated annealing optimization by solving sudoku puzzle," in *Emergent Trends in Robotics and Intelligent Systems*, ser. Advances in Intelligent Systems and Computing, P. Sinčák, P. Hartono, M. Virčíková, J. Vaščák, and R. Jakša, Eds. Springer International Publishing, 2015, vol. 316, pp. 207–213.

[8] T. Mantere and J. Koljonen, "Solving and Rating Sudoku Puzzles with Genetic Algorithms," in *Proceedings of the 12th Finnish Artificial Intelligence Conference (STeP 2006)*. Espoo, Finland: Finnish Artificial Intelligence Society, 2006, pp. 86–92.

[9] A. Moraglio, J. Togelius, and S. Lucas, "Product geometric crossover for the sudoku puzzle," in *IEEE Congress on Evolutionary Computation, 2006. CEC 2006*, 2006, pp. 470–476.

[10] T. Y. Lim, M. Al-Betar, and A. Khader, "Monogamous pair bonding in genetic algorithm," in *2015 IEEE Congress on Evolutionary Computation (CEC)*, May 2015, pp. 15–22.

[11] Z. Wang, T. Yasuda, and K. Ohkura, "An evolutionary approach to sudoku puzzles with filtered mutations," in *2015 IEEE Congress on Evolutionary Computation (CEC)*, May 2015, pp. 1732–1737.

[12] T. Mantere, "Improved ant colony genetic algorithm hybrid for sudoku solving," in *2013 Third World Congress on Information and Communication Technologies (WICT)*, Dec 2013, pp. 274–279.

[13] Y. Borenstein and A. Moraglio, *Theory and Principled Methods for the Design of Metaheuristics*. Springer Publishing Company, 2014.

[14] M. Črepinšek, S.-H. Liu, and M. Mernik, "Exploration and exploitation in evolutionary algorithms: A survey," *ACM Computing Surveys*, vol. 45, no. 3, pp. 35:1–35:33, Jul. 2013.

[15] C. Segura, S. I. V. Peña, S. B. Rionda, and A. H. Aguirre, "The importance of diversity in the application of evolutionary algorithms to the sudoku problem," in *2016 IEEE Congress on Evolutionary Computation (CEC)*, July 2016, pp. 919–926.

[16] H. M. Pandey, A. Chaudhary, and D. Mehrotra, "A comparative review of approaches to prevent premature convergence in GA," *Applied Soft Computing*, vol. 24, pp. 1047 – 1077, 2014.

[17] C. Segura, S. Botello Rionda, A. Hernández Aguirre, and S. I. Valdez Peña, "A novel diversity-based evolutionary algorithm for the traveling salesman problem," in *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*, ser. GECCO'15. New York, NY, USA: ACM, 2015, pp. 489–496.

[18] C. Segura, C. Coello Coello, E. Segredo, and A. Aguirre, "A novel diversity-based replacement strategy for evolutionary algorithms," *IEEE Trans. Cybern.*, vol. 46, no. 12, pp. 3233–3246, December 2016.

[19] J. Almog, "Evolutionary computing methodologies for constrained parameter, combinatorial optimization: Solving the sudoku puzzle," in *AFRICON, 2009. AFRICON '09.*, Sept 2009, pp. 1–6.

[20] E. Galvan-Lopez and M. O'Neill, "On the effects of locality in a permutation problem: The sudoku puzzle," in *IEEE Symposium on Computational Intelligence and Games, 2009. CIG 2009*, Sept 2009, pp. 80–87.

[21] K. N. Das, S. Bhatia, S. Puri, and K. Deep, "A Retrievable GA for Solving Sudoku Puzzles A Retrievable GA for Solving Sudoku Puzzles," Department of Electrical Engeneering, Indian Institute of Technology Roorkee, Tech. Rep., 2007.

[22] T. Mantere and J. Koljonen, "Solving, rating and generating sudoku puzzles with GA," in *IEEE Congress on Evolutionary Computation, 2007. CEC 2007*, Sept 2007, pp. 1382–1389.

[23] S. Felman, "Solving Sudoku Using Genetic Operations and Sub-Blocks With Optimized Mutation Rates," St. Petersburg College, Tech. Rep., 2007.

[24] Y. Sato and H. Inoue, "Solving sudoku with genetic operations that preserve building blocks," in *2010 IEEE Symposium on Computational Intelligence and Games (CIG)*, Aug 2010, pp. 23–29.

[25] X. Q. Deng and Y. D. Li, "A novel hybrid genetic algorithm for solving sudoku puzzles," *Optimization Letters*, vol. 7, no. 2, pp. 241–257, 2013.

[26] T. Lambert, E. Monfroy, and F. Saubion, "A generic framework for local search: Application to the sudoku problem," in *Computational Science – ICCS 2006*, ser. Lecture Notes in Computer Science, V. Alexandrov, G. van Albada, P. Sloot, and J. Dongarra, Eds. Springer Berlin Heidelberg, 2006, vol. 3991, pp. 641–648.

[27] J. M. Weiss, "Genetic Algorithms and Sudoku," in *Midwest Instruction and Computing Symposium (MICS 2009)*, 2009, pp. 1–9.

[28] Y. Sato, N. Hasegawa, and M. Sato, "Acceleration of genetic algorithms for sudoku solution on many-core processors," in *Massively Parallel Evolutionary Computation on GPGPUs*, ser. Natural Computing Series, S. Tsutsui and P. Collet, Eds. Springer Berlin Heidelberg, 2013, pp. 421–444.

[29] E. Alba, *Parallel Metaheuristics: A New Class of Algorithms*, ser. Wiley series on parallel and distributed computing. Wiley-Interscience, 2005.

[30] H. Chel, D. Mylavarapu, and D. Sharma, "A novel multistage genetic algorithm approach for solving sudoku puzzle," in *2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT)*, March 2016, pp. 808–813.

[31] G. Singh and K. Deep, "A new membrane algorithm using the rules of particle swarm optimization incorporated within the framework of cell-like p-systems to solve sudoku," *Applied Soft Computing*, vol. 45, pp. 27 – 39, 2016.

[32] A. M. Herzberg and M. R. Murty, "Sudoku squares and chromatic polynomials," *Notices of the American Mathematical Society*, vol. 54, no. 6, pp. 708–717, 2007.

[33] K. Oddson, "Math and sudoku: Exploring sudoku boards through graph theory, group theory, and combinatorics," in *Student Research Symposium, Paper 4*, May 2016.

[34] L. Taalman, E. Arnold, and S. Lucas, "Gröbner basis representations of sudoku," *College Mathematics Journal*, vol. 41, no. 2, pp. 101–112, march 2010.

[35] S. Jones, P. Roach, and S. Perkins, "Construction of heuristics for a search-based approach to solving sudoku," in *Research and Development in Intelligent Systems XXIV*, M. Bramer, F. Coenen, and M. Petridis, Eds. Springer London, 2008, pp. 37–49.

[36] T. Mantere and J. Koljonen, "Solving and analyzing sudokus with cultural algorithms," in *IEEE Congress on Evolutionary Computation, 2008. CEC 2008*, June 2008, pp. 4053–4060.

[37] C. Segura, A. Hernandez, F. Luna, and E. Alba, "Improving diversity in evolutionary algorithms: New best solutions for frequency assignment," *IEEE Trans. Evol. Comput.*, vol. In Press, no. 99, pp. 1–1, 2017.

[38] G. R. Harik, "Finding multimodal solutions using restricted tournament selection," in *Proceedings of the 6th International Conference on Genetic Algorithms*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1995, pp. 24–31.

[39] T. Vidal, T. G. Crainic, M. Gendreau, and C. Prins, "A hybrid genetic algorithm with adaptive diversity management for a large class of vehicle routing problems with time-windows," *Computers & Operations Research*, vol. 40, no. 1, pp. 475 – 489, 2013.

[40] V. Koumousis and C. Katsaras, "A saw-tooth genetic algorithm combining the effects of variable population size and reinitialization to enhance performance," *IEEE Trans. Evol. Comput.*, vol. 10, no. 1, pp. 19–28, Feb 2006.

[41] C. León, G. Miranda, and C. Segura, "METCO: A Parallel Plugin-Based Framework for Multi-Objective Optimization," *International Journal on Artificial Intelligence Tools*, vol. 18, no. 4, pp. 569–588, 2009.

[42] [Online]. Available: http://www.e-sudoku.fr/

[43] [Online]. Available: http://en.top-sudoku.com/

[44] É. D. Taillard, "A statistical test for comparing success rates," in *Proceedings of The Fifth Metaheuristic Interantional Conference (MIC'03)*, 2003.