The Importance of Diversity in the Application of Evolutionary Algorithms to the Sudoku Problem

Carlos Segura*, S. Ivvan Valdez Peña*, Salvador Botello Rionda*, Arturo Hernández Aguirre*

*Centro de Investigación en Matemáticas, A.C. (CIMAT), Área de Computación,
Jalisco S/N, Col. Valenciana, Guanajuato, 36023, México
Email: carlos.segura@cimat.mx, ivvan@cimat.mx, botello@cimat.mx, artha@cimat.mx

***Abstract*—The past few years have seen several variants of Evolutionary Algorithms (EAs) applied to solving Sudoku puzzles. Given that EAs with simple components do not work properly, considerable efforts have gone into designing ad-hoc evolutionary operators that profit from a knowledge of the problem. In this paper, we show that one of the main reasons for the improper behavior of EAs when dealing with difficult Sudoku puzzles is the appearance of premature convergence. Memetic algorithms with readily available genetic operators can be used to solve the hardest known Sudoku puzzles when general and well-known methods for avoiding premature convergence are incorporated. Among the approaches tested, a recently proposed method that is based on adopting multi-objective concepts for the solution of single-objective problems has shown remarkable performance. To our knowledge, the methods presented in this paper are the first EAs capable of solving the three Sudoku puzzles that are regarded as the most difficult ones known to date.**

I. Introduction

Evolutionary Algorithms (EAs) are one of the most popular approaches for several kinds of optimization, such as continuous, multimodal and combinatorial optimization [1]. In spite of their success, adapting EAs to new problems is not an easy task, as this usually involves many difficult design decisions [2]. Particularly, since the inception of EAs, premature convergence has been recognized as one of their recurrent drawbacks [3], so when applying EAs to new problems, special attention must be paid to this matter. Premature convergence arises when all of the population members are located in a small part of the search space — different from the optimal region — and the components selected do not allow escaping from this region. Since this drawback is well-known, several approaches have been proposed to address this difficulty. However, the approaches devised influence different components of the EAs and in some cases their design is problem-dependent, so adapting them to new problems is not trivial. In any case, there are several general methods that can be easily applied regardless of the problem at hand, so these kinds of schemes should be one of the first alternatives to try when facing such difficulties.

The Sudoku puzzle is a very popular number-based logic puzzle consisting of an $N^2 \times N^2$ grid that is divided into N^2 blocks of size $N \times N$. Initially, numbers are given in some of the cells and the objective is to fill the remaining cells with digits from 1 to N^2 in a way that, each row, column and block must contain only one instance of each digit. The most popular variant of Sudoku involves a 9×9 grid. This is probably because since this problem is NP-Complete [4], larger grids might be too complicated to be fun. Given the properties of Sudoku puzzles, in recent years they have been widely used to test the abilities and performance of different proposals. For instance, most of the popular metaheuristics have been adapted to tackle this problem [5], [6]. In the particular case of EAs, it is known that its basic variants can solve simple Sudoku puzzles [7]. However, they encounter significant difficulties when tackling more difficult instances, even if more complex components are included [8], [9], [10]. In order to alleviate these drawbacks, many hybrid variants and ad-hoc genetic operators have been proposed [11]. While these advances represent important achievements, these lines of research could be indicating that incorporating problem-specific knowledge is mandatory for designing successful EAs for Sudoku. However, the methods presented in this paper show that this is not the case.

In this paper, an alternative way of handling these difficulties is proposed by incorporating into a simple memetic algorithm some of the advances that have been devised to deal with premature convergence. Specifically, several mature methods that were designed to control the diversity of the population [12] are successfully incorporated into a simple memetic algorithm that uses some of the first genetic operators ever designed. In addition, a more recent method that was proposed in [13], [14] was also used and shows remarkable performance when compared to more mature methods. The novelty of this last method is that 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. Thus, the contributions of this paper are two-fold. First, we show that by controlling the diversity of the population, successful Sudoku evolutionary solvers that do not incorporate problem-specific knowledge in the genetic operators can be designed. Second, the generality and robustness of a recently proposed replacement strategy that explicitly controls the diversity of the population is shown.

The rest of the paper is organized as follows. A discussion of the relevant background, including general methods for avoiding premature convergence and evolutionary solvers for

978-1-5090-0623-6/16/$31.00

Sudoku, is given in Section II. In Section III our recently proposed method for controlling diversity is introduced and the way it is adapted to the Sudoku puzzle is described. A novel memetic algorithm that includes the different methods for avoiding premature convergence is outlined in Section IV. Then, our experimental validation is presented in Section V. Finally, conclusions and some lines of future work are given in Section VI.

II. Related Work

A. Diversity Preservation in Evolutionary Algorithms

Premature convergence is a well-known drawback of EAs, so a large number of techniques to deal with this problem have been devised [12], [3]. In this section, we review some of the most popular general techniques, which will be used to show that using problem-specific knowledge is not required to successfully solve difficult Sudoku puzzles.

Most of the initial approaches focused on devising new parent selection schemes where the selection pressure could be controlled [1]. However, some studies found that parent selectors were not able to maintain proper diversity *per se* [15] even if relatively large populations were used. Thus, these schemes have to be combined with other strategies to successfully preserve diversity. The population model has also been studied with the aim of improving diversity preservation in EAs. EAs with structured populations — instead of panmictic schemes — have gained popularity in recent years [16]. In these schemes, some recombination restrictions are imposed by taking into account the positions of the individuals in the population. Schemes based on mating restrictions [9] are similar to those described above in the sense that some interactions between individuals are avoided. However, in these cases, this is not done in a static way that depends on the positions of the individuals, and additional features of the individuals are usually taken into account to impose the restrictions.

Another alternative resides in adapting the variation stage. For instance, in many cases, several genetic operators are considered simultaneously [17]. However, this kind of control usually depends on the problem and on the features of the genetic operators. Finally, restarting schemes are also quite popular. In these schemes, instead of avoiding fast convergence, this event is allowed, and then all or part of the population is restarted [18]. A particular case of this type of scheme is the saw-tooth genetic algorithm (saw-tooth GA) [19]. This scheme uses a variable population size and periodic partial reinitialization of the population in the shape of a saw-tooth function, which is configured with the period (P) and the amplitude (D).

Some diversity-preservation techniques that rely on modifying the replacement phase have also been devised. The basic principle of these schemes is that by diversifying the survivors, more exploration can be induced in successive generations [14]. For instance, in *crowding* the basic principle is to force new individuals entering the population to replace similar individuals. Some of the most popular crowding methods are *Mahfoud's deterministic crowding* [20] (DETCR), and the *Restricted Tournament Selection* [21] (RTS). RTS is not a parameter-less approach. Once a new individual (C) is created, CF individuals from the current population are randomly selected. Then, C and its most similar individual — from those in the selected set — compete for a place in the population using a traditional binary tournament.

Several other replacement strategies that promote diversity have been proposed. One of the most popular is probably the *clearing* strategy (CLR) [22]. Clearing can be regarded as an extension of fitness sharing. However, while in fitness sharing the fitness of each individual is normalized depending on the number of individuals in its region — defined via the parameter σ —, in the clearing procedure the resources of a niche are attributed to the best W elements in each niche. Moreover, the winners of each niche are automatically preserved by copying them to the next population. Another popular strategy involves *aging* methods. In these schemes, the probability of selecting an individual depends not only on its objective function but also on its age, i.e. the number of generations elapsed since its creation. This principle can be applied both to the parent selection and the replacement phase.

In some methods, maximizing diversity is considered as an objective. For instance, in COMB [23] the individuals are sorted by the original cost and by their contribution to diversity. Then, the rankings of the individuals are combined to generate the fitness value using two parameters (N_{Close} and N_{Elite}). In each step of the replacement phase, the individual with the lowest fitness is erased and the ranks are recalculated. Another alternative is the *contribution-diversity replace-worst* method devised in [24] (CD/RW). In the CD/RW method, a new individual enters the population by replacing another one that is worse both in quality and diversity contribution. If such an individual is not found, a *replace-worst* strategy (RW) is applied, i.e., the worst individual in the population is erased if it is worse than the newly generated individual.

Finally, another quite popular alternative is to explicitly consider diversity as an objective and apply ideas that arise in the multi-objective optimization field [25]. These kinds of schemes are usually referred to as diversity-based multi-objective EAs (MOEAs) [26], [27]. An extension of these methods, which is applied in this paper, is detailed in Section III.

B. Evolutionary Algorithms for Solving Sudoku Puzzles

Since the popularization of Sudoku, several different stochastic optimization algorithms have been applied to solving Sudoku puzzles [5], [6]. Of all the different metaheuristics, the application of EAs is probably the most popular. Mantere has been one of the most active researchers in this field. One of his first contributions is related with the encoding of solutions [7]. In order to solve a Sudoku puzzle, four different kinds of constraints (the given values cannot be altered and the constraints associated with the rows, columns and blocks) must be satisfied. Mantere showed that, with a

proper representation, two of these types of restrictions can be fulfilled in every individual generated. For instance, in [7], a representation where the only constraints that must be taken into account are the ones associated with rows and columns is devised. Basically, the chromosome consists of a list of sub-chromosomes, where each of them is a permutation of the values that are absent in a block. Note that instead of dividing per block, the division can also be made by rows or columns [28]. Also note that the generation of solutions that take into account three of the constraints simultaneously is NP-Complete [8], so 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 [8].

In the initial proposals the genetic operators applied were quite straightforward [7], [29]. For instance, in [7], the mutation is based on performing swaps in the sub-chromosomes while a one-point crossover that interchanges complete sub-chromosomes is applied. While these simple operators yielded good results with simple Sudoku puzzles, harder versions could not be solved properly. Much of the research in this field has focused on the design and application of more complex crossover and mutation operators. First, some additional general operators have been tested. For instance, in [8], [28] several permutation-based crossovers that combine information on sub-chromosomes were analyzed, while variants of two-point crossover and uniform crossover that interchange complete sub-chromosomes have been also quite popular [18], [9]. While these operators improved on the results obtained with the initial operators, they were not able to provide proper solutions for several of the hardest Sudoku puzzles. Due to these weaknesses of general operators, genetic operators that take into account specific knowledge of the problem were developed. For instance, in the mutation operator devised in [30], some ineffective swaps are avoided. This principle was further extended in the operators devised in [10]. A different alternative is the mutation devised in [31], where the behavior of the mutation depends on a score that is established for each sub-chromosome. In the case of crossover, one of the most effective was proposed in [32]. The idea behind this crossover is to assign a score to each row and column, and take this information into account with the aim of preserving the building blocks of the individuals involved.

Another research line involves hybridizing EAs with other population-based metaheuristics. For instance, in [33] some principles of particle swarm optimization are incorporated into an EA, while in [11] — which is one of the best known stochastic optimizers for Sudoku known to date —, EAs and Ant Colony Optimization are applied together. In addition, it is very typical to incorporate local searches in the EAs. In most cases, simple definitions of neighborhoods are used. For instance, in most typical cases each neighbor is generated by simply swapping two positions of a sub-chromosome [34], [32]. Incorporating local searches greatly reduces the amount of time required to converge, so in this paper we incorporate a simple stochastic hill climbing method.

Another important issue for the performance of EAs is the definition of the fitness function. Some of the first proposals considered fitness functions that were quite complex. For instance, in [7] 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 [30]. 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 kinds of penalties have been confirmed in several variants of EAs [36], [31].

Finally, it is important to remark that none of the previous papers have taken into account different traditional schemes for managing premature convergence; however, by analyzing some of the reported schemes, it is quite clear that there are important issues related to premature convergence. First, schemes that do not include any mechanisms to alleviate premature convergence yield quite disappointing results [5], [37]. Second, some of the most effective solvers do in fact include mechanisms to avoid premature convergence. For instance, in [11], 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. Additionally, the advantages of imposing mating restrictions, which seek to decelerate the loss of diversity, are shown in [9]. More evidence of this is provided in [38], where the *parallel independent runs model* [16] 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.

III. Diversity-based Replacement Strategy

Recently, a new replacement strategy (MULTI_DYN) was proposed that explicitly takes diversity into account [13], [14]. This replacement strategy was incorporated into our proposals to solve Sudoku puzzles. Its method of operation and how it was adapted are described in this section.

Our proposal is an extension of the replacement strategy initially described in [39]. In that scheme, the replacement strategy simultaneously considers the diversity introduced by an individual and its objective function. The operation works as follows. In the first step of the replacement strategy, the best individual — in terms of the objective function — is selected to survive. Then, to select each additional survivor, the non-dominated front that takes into account the original objective function and the diversity contribution is calculated. The diversity contribution of each individual is measured as the distance to the closest individual (DCN) already selected to survive. Then, a survivor is selected at random from the non-dominated front. The main advantage of the extension applied in this paper [13], [14] is that the balance between

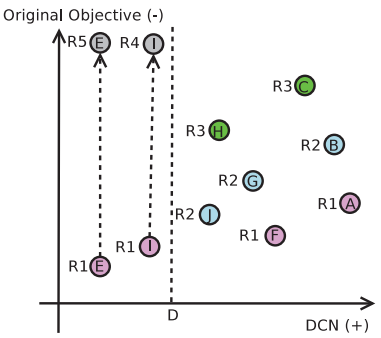

Fig. 1. Effect of the penalty approach with distance D

exploration and exploitation is automatically adjusted based on the given stopping criterion. Thus, the stopping criterion, as well as the elapsed time, are used as inputs to the replacement strategy. Thus, for shorter stopping criteria the method induces a faster reduction in diversity than for longer stopping criteria. Specifically, in order to implement such an adaptation in the balance between exploration and exploitation, we have defined a dynamic penalty approach that punishes those individuals that contribute too little to diversity.

One of the basic principles of MULTI_DYN is that individuals that contribute too little to diversity should not be accepted regardless of their original objective value. In our approach, individuals that contribute too little are penalized by setting their original objective value to a very low quality value. In this paper, the objective is set to the maximum integer value that can be represented with 32 bits. This operation is illustrated in Fig. 1. In this figure, the value D represents the minimum DCN required to avoid being penalized. As the figure shows, any individual whose DCN value is lower than this threshold value is penalized. As a result, the non-domination rank (shown at the left of each individual) of the penalized individuals might be increased. Obviously, with this approach, the penalized individuals will not belong to the non-dominated front, unless every pending individual has been penalized.

While the above approach is quite logical, one of the key decisions involves how to set the value of D. In our scheme, an initial D_I value must be set. Then, a linear reduction of D is done. The reduction is calculated in such a way that by the end of the execution, the resulting value is 0. This means that, initially, a large degree of exploration is induced. In this paper, the stopping criterion is set by time. Thus, if T_{End} is the number of seconds allocated to the run and $T_{Elapsed}$ is the elapsed time, D is calculated as $D = D_I - D_I * \frac{T_{Elapsed}}{T_{End}}$.

Algorithm 1 shows the pseudocode of MULTI_DYN. Note that the only components that depend on the problem at hand are the calculation of the distance between individuals and the objective function. In the case of the Sudoku puzzle, the Hamming distance is used, i.e., the distance between two individuals is the number of cells where different numbers have been assigned. The fitness function is detailed in the next section.

Algorithm 1 MULTI_DYN survivor selection scheme

```
CurrentMembers = Population ∪ Offspring
Best = Individual with best f(x) in CurrentMembers
NewPop = { Best }
CurrentMembers = CurrentMembers - { Best }
while (|NewPop| < N) do
    Calculate DCN of CurrentMembers, using NewPop as a reference
    D = D_I − D_I * T_Elapsed / T_End
    Penalize(CurrentMembers, D)
    ND = Non-dominated individuals of CurrentMembers (without repetitions)
    Selected = Randomly select an individual from ND
    NewPop = NewPop ∪ Selected
    CurrentMembers = CurrentMembers - {Selected}
end while
Population = NewPop
```

Algorithm 2 Diversity-based Lamarckian Memetic Algorithm

```
Initialization: Generate an initial population P_0 with N individuals. Assign t = 0.
Local Search: Perform a local search for every individual in the population.
while (not stopping criterion) do
    Evaluation: Evaluate all individuals in the population.
    Mating selection: Perform binary tournament selection on P_t in order to fill the mating pool.
    Variation: Apply genetic operators to the mating pool to create a child population CP.
    Local Search: Perform a local search for every individual in the offspring.
    Survivor selection: Combine P_t and CP, and apply the MULTI_DYN replacement scheme to create P_{t+1}.
    t = t + 1
end while
```

IV. A Memetic Algorithm for Sudoku Puzzles

Our base memetic algorithm is a quite simple EA that has been integrated with a simple stochastic hill-climbing local search. Instead of defining new genetic operators and neighborhoods, we have used some of the simplest ones already defined in the literature. Note that the operators selected have only been successful with easy Sudoku puzzles, so they are suitable choices to show that by properly managing diversity, general and simple operators can be successfully applied. Algorithm 2 shows the variant of the memetic algorithm that applies the MULTI_DYN survivor selection strategy. Other algorithms were defined by using alternative ways of controlling premature convergence. These schemes, as well as their parameterizations, are listed in Section V.

In order to properly instantiate our approach, the following components were selected. In the case of the encoding, we selected the most popular representation [7]. Specifically, an individual consists of a set of sub-chromosomes, each of which is a permutation of the numbers absent in a block. Thus, the only constraints that are considered by the fitness function are those involving rows and columns. The fitness function is calculated as the sum of two terms: the first is the number of repetitions that appear in the rows and columns; and the

second takes into account the numbers given initially in the board. Specifically, the number of conflicts with these numbers is counted and the value is multiplied by 100. In this way, conflicts with the given values are quickly avoided, resulting in faster convergence.

In the case of the variation scheme, very straightforward operators were applied. First, a crossover operator is applied with probability p_c. The operator exchanges complete blocks between solutions uniformly, i.e. we use the traditional uniform crossover but instead of working at the gene level, it operates at the sub-chromosome level. The mutation operator iterates over each gene and with probability p_m performs a swap with another element selected at random from its corresponding sub-chromosome.

Finally, the local search is also quite simple. A neighbor of a candidate solution is generated by swapping two elements of a block. Then, using this definition of neighborhood, a stochastic hill-climbing is applied, i.e., the neighbors are considered in random order and only movements that result in an improvement are accepted. This process is repeated until a local optimum is reached.

V. Experimental Evaluation

In this section we describe the experiments conducted with our memetic algorithm and we discuss the results obtained using a large number of different ways to control its convergence. The optimization schemes were implemented using METCO (*Metaheuristic-based Extensible Tool for Cooperative Optimization*) [40]. Tests have been run on bi-processor machines with 32Gb RAM. Each processor is an Intel(R) Xeon(TM) CPU E5-2620 at 2.10GHz. The analyses were performed with 26 different Sudoku puzzles[1]. Twenty Sudoku puzzles were generated with two of the most popular web pages dedicated to Sudoku [41], [42]. In the case of the Sudoku puzzles generated in [41] (tagged as SW), the evil level was selected, whereas for the ones generated in [42] (tagged as SS) the hard level was used. Note that in both cases, the level selected is the most difficult one. Three additional Sudoku puzzles (tagged as SD), categorized as super-difficult, were taken from [32]. Note that one of them is the popular AI-Escargot puzzle, which was considered to be the most difficult Sudoku puzzle when it was created. Finally, the last three are now regarded as the most difficult, based on the level assigned by Sudoku Explainer and by other metrics [43]. One of them was designed by Arto Inkala, whereas the other two were proposed by David Filmer.

Since stochastic algorithms were considered in this study, each execution was repeated 30 times. In this kind of problem, the only successful outcome is to solve the problem. Thus, when the puzzle is not solved, there is no point in comparing the resulting fitness values. Consequently, the analyses consider the success rate in achieving a solution. Additionally, the time required to solve a Sudoku is also important. In order to compare the times among those schemes that attained a 100% success rate, we conducted a series of statistical tests that relied on a guideline similar to that applied in [14]. This guideline takes into account the ANOVA, *Shapiro-Wilk*, *Levene*, *Welch* and *Kruskal-Wallis* tests. They were applied assuming a significance level of 5%.

[1]The Sudoku puzzles are available at http://www.cimat.mx/~carlos.segura/Sudoku/SudokuPuzzles.tar.gz

A. Comparison of Premature Convergence Control Methods

Given that the main objective of this paper is to show the advantages of introducing methods to delay convergence when solving Sudoku puzzles, some of the best-known methods were incorporated in our simple memetic algorithm. In every case, we used typical values for the population size (N), mutation probability (p_m) and crossover probability (p_c). Specifically, they were set to 100, 0.01 and 0.8, respectively. Other parameterizations were also tested but for close parameterizations, the results were quite similar, meaning that the algorithm is not too sensitive to the parameterization. Every variant of our memetic algorithms considered the genetic operator and local search variants explained earlier. Additionally, in every case the stopping criterion was set to 5 minutes.

The following variants were tested. First, two schemes that did not include any mechanism to delay convergence were included. The first one was a generational scheme with elitism (GEN_ELIT). In this case, the best member of the population is automatically copied to the next generation while the rest of the population is created with the variation scheme. A less explorative version, the replace-worst strategy (RW), was also tested. In RW the offspring and the previous population are joined in a set of $2 \times N$ individuals in which the best N individuals survive. Additionally, seven schemes that incorporate mechanisms to delay the convergence were tested. In those cases where additional parameters must be set, we carried out some initial experiments with the last six instances, selecting the parameters that yielded the highest success rate. The first one incorporates the MULTI_DYN replacement. In this case, the parameter D_I was set to 20. In our initial experiments we also tested the values 0, 10, 30 and 40. The CLR strategy was also tested. In our initial experiments σ was set to 5, 10, 15, 20 and 25, whereas k was set to 1, 3 and 5. The best combination was obtained by setting σ to 10 and k to 1. Two variants of crowding were also considered: the RTS and DETCR. The RTS scheme with W set to 2, 5, 10, 25, 50, 75 and 100 was analyzed. The best reported solutions were obtained when W was set to 75. In the case of DETCR no additional parameters are required. As an example of a method with reinitialization, we included the SAW-TOOTH GA. In our initial experiments we set D to 25, 50, 75 and 99 and T to 25, 50, 100 and 500. The best combination resulted from setting D to 99 and T to 50. The COMB strategy also takes into account two parameters. In both parameters the values 1, 3, 8, 15 and 25 were tested. In the best configuration N_{Close} was set to 3, whereas N_{Elit} was set to 8. Finally, the CD/RW was also included. In this case, no additional parameterization is required.

Table I shows the results obtained with each scheme for the 26 Sudoku puzzles. For each method and puzzle, the table shows the success rate (SR) and the time in seconds

TABLE I
SUMMARY OF THE RESULTS OBTAINED BY THE DIFFERENT TESTED SCHEMES (5 MINUTES)

	MULTI_DYN		RTS		COMB		SAWTOOTH		CLR		DETCR		CD/RW		RW		GEN_ELIT	
Inst.	SR	T	SR	T	SR	T	SR	T	SR	T	SR	T	SR	T	SR	T	SR	T
SW1	100	3.6	100	2.4	100	3.7	100	2.1	100	0.6	100	1	73.3	0.8	63.3	0.5	53.3	3.2
SW2	100	2.4	100	2	100	1.9	100	0.6	100	0.3	100	0.5	96.7	0.6	100	0.3	100	0.5
SW3	100	5.4	100	3	100	2.6	100	0.8	100	0.5	100	0.9	83.3	0.9	63.3	3.1	73.3	1.4
SW4	100	1.2	100	1.2	100	1.2	100	0.6	100	0.3	100	0.4	96.7	0.5	80	0.5	76.7	0.7
SW5	100	4.8	100	3.6	100	2.9	100	1	100	0.8	96.7	1	70	2.9	53.3	6.9	66.7	8
SW6	100	3.7	100	5.1	100	5.2	100	8.8	100	2.1	96.7	2.1	33.3	-	13.3	-	20	-
SW7	100	2.3	100	1.8	100	1.3	100	0.5	100	0.3	100	0.4	90	0.5	80	0.5	83.3	0.3
SW8	100	2.1	100	1.9	100	1.8	100	0.8	100	0.4	100	0.7	86.7	0.8	86.7	0.6	80	0.6
SW9	100	5.4	100	2.4	100	3.6	100	1	100	0.7	100	1	70	2.1	43.3	-	50	1.6
SW10	100	4.2	100	3	100	2.3	100	0.7	100	0.5	100	0.8	96.7	0.7	90	0.8	90	0.9
SS1	100	1.8	100	1	100	1.1	100	0.4	100	0.2	100	0.4	96.7	0.7	93.3	0.5	93.3	0.7
SS2	100	12.5	100	4.8	100	8.6	100	11.3	100	1.9	96.7	1.5	16.7	-	36.7	-	10	-
SS3	100	4.8	100	4.1	100	2.2	100	2.2	100	0.9	93.3	1.1	73.3	5.2	36.7	-	46.7	-
SS4	100	63.4	96.7	37.7	90	87.4	66.7	201.5	16.7	-	13.3	-	10	-	6.7	-	0	-
SS5	100	5.9	100	3.2	100	4.1	100	1.2	100	0.7	100	1.8	60	1.4	43.3	-	56.7	13.6
SS6	100	7.1	100	3.8	100	3.2	100	1.1	100	0.6	100	1.3	83.3	1.1	70	6	80	1.2
SS7	100	1.4	100	1.6	100	1.2	100	0.5	100	0.4	100	0.5	90	0.9	73.3	0.6	70	0.7
SS8	100	2.2	100	1.6	100	2.2	100	0.6	100	0.5	100	0.6	93.3	0.8	66.7	0.9	70	0.9
SS9	100	1.5	100	2.3	100	2	100	0.6	100	0.4	100	0.6	86.7	1.1	73.3	0.9	66.7	12
SS10	100	5.4	100	4.4	100	6.5	100	2.3	100	0.6	96.7	1.2	86.7	4.2	40	-	26.7	-
SD1	100	14.5	100	12.5	100	23	100	10.2	100	6.1	90	5	23.3	-	13.3	-	10	-
SD2	100	7.6	100	22.6	100	10.2	100	6.9	100	4.4	96.7	2.9	30	-	33.3	-	16.7	-
SD3	100	10.8	100	8.6	100	6.6	100	5.3	100	3	100	3.9	60	55.2	30	-	23.3	-
Inkala	100	67.3	86.7	71.6	63.3	199.2	76.7	147.1	80	154.6	30	-	3.3	-	0	-	0	-
Film1	83.3	155	33.3	-	16.7	-	20	-	36.7	-	0	-	3.3	-	0	-	3.3	-
Film2	63.3	208	20	-	23.3	-	16.7	-	0	-	3.3	-	6.7	-	0	-	0	-

TABLE II
STATISTICAL COMPARISON OF TIMES FOR THE 22 EASIEST PUZZLES

	MULTI_DYN		
	↑	↓	↔
CLR	0	20	2
SAW-TOOTH GA	0	15	7
RTS	1	8	13
COMB	0	5	17

(T) required for obtaining, at least, a 50% success rate. A dash indicates those cases where the success rate was lower than 50%. The methods are sorted by the number of successful runs. Note that the worst schemes were the two schemes that did not incorporate any mechanisms to delay convergence, i.e., the GEN_ELIT and the RW methods. The remaining methods yielded important benefits, meaning that all of the schemes tested that were devised to control premature convergence are useful for improving the behavior of EAs for Sudoku. However, there are important differences between their behaviors.

The scheme that provided the highest success rate for all the Sudoku puzzles as a whole was the MULTI_DYN method. In fact, it obtained a 100% success rate in every Sudoku puzzle, except for those devised by David Filmer, for which the success rate was higher than 50%. The RTS, COMB, SAW-TOOTH GA and CLR also provided quite important benefits. In fact, they only encountered difficulties with SS4 and with the puzzles generated by Arto Inkala and David Filmer. Furthermore, based on the times required to attain a 50% success rate, they converged faster than MULTI_DYN in most cases. In order to confirm this behavior, we performed a statistical comparison between the times required by MULTI_DYN and those required by RTS, COMB, SAW-TOOTH GA and CLR in the 22 Sudoku puzzles where every scheme obtained a 100% success rate. Table II shows the number of puzzles where MULTI_DYN was quicker (↑) and slower (↓) than the corresponding approaches listed in the rows. The number of cases where the differences were not statistically significant are shown in the column with the symbol ↔. In light of these results and those in Table I, it is clear that in the less challenging instances, the CLR and SAW-TOOTH GA strategies are preferred, whereas in the most difficult puzzles the MULTI_DYN method offers significant advantages. Note that this behavior by MULTI_DYN was also evident in the Traveling Salesman Problem [13], where the advantages were clearer when dealing with large-scale and difficult instances.

B. Analysis of Diversity

While all of the schemes applied try to delay convergence, each one works in drastically different ways. We can better understand the behavior of the different schemes by analyzing the trend in the diversity of the population. Several different metrics exist [3], with those based on calculating distances between individuals and the entropy probably being used the most. Since none of the schemes takes into account the entropy directly, we decided to base our analyses on the entropy.

Figure 2 shows the evolution of the mean entropy for the different schemes in the SW1 Sudoku puzzle. On the left are the methods that yielded a faster convergence. As expected, the two methods that do not incorporate any mechanisms to delay convergence are in this group. Also as expected, the four methods that attained the largest success rates are

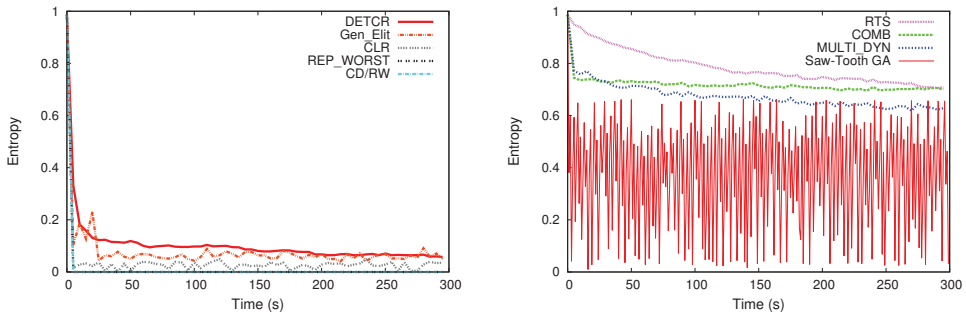

Fig. 2. Evolution of the mean entropy for all the schemes considered (SW1 instance)

in the group with slower convergence (right). In this group we included the Saw-tooth GA, which recovers diversity after each partial reinitialization. The MULTI_DYN and the COMB methods initially lose some diversity very quickly, and then they behave differently. COMB maintains the same degree of diversity throughout the execution, whereas MULTI_DYN gradually loses diversity, meaning that more intensification is induced over time. In the case of RTS, the convergence process is quite slow, meaning that too much time might be required for the most difficult puzzles. The behavior of CLR might also be somewhat surprising. In the initial phases it loses diversity very quickly, similarly to RW and CD/RW. However, while RW and CD/RW converge, in CLR the loss of diversity is stopped before complete convergence. This level of diversity is enough to solve some hard Sudoku puzzles, but not the most difficult ones. Note that the behavior of DETCR is somewhat similar. In the case of GEN_ELIT, while the evolution of diversity is not radically different, the results are much worse. This is because the survivor selection phase of DETCR and CLR take into account the quality of individuals when selecting them. Thus, they are able to maintain diverse, high-quality individuals. However, in the GEN_ELIT scheme, only one individual is selected with elitism, meaning that many of the individuals might not be fit. Thus, in GEN_ELIT such an increase in the diversity level is not helpful.

We would like to remark that similar analyses with other puzzles reported similar behavior. The most noticeable difference is that, at the end of the executions, MULTI_DYN reached lower diversity levels when solving easier puzzles. The reason are related to the findings of [28]. Particularly, in the difficult instances, there are several distant regions with proper fitness values, but only one of them is the region of the solution to the puzzle. In these cases, MULTI_DYN maintains several solutions in distant regions even at the end of the executions. This allows each region to be better explored, increasing the probabilities of solving the puzzle.

Finally, we would like to note that all the schemes were also executed by setting the stopping criterion to 30 minutes with the Filmer1 and Filmer2 puzzles. In this case, the only method that was able to obtain a 100% success rate was MULTI_DYN, whereas the second best method (RTS) obtained a success rate of 90% with Filmer1 and 70% with Filmer2.

VI. Conclusions and Future Work

The Sudoku puzzle is an NP-Complete problem that has been used extensively to test the abilities of new optimization paradigms. In the case of EAs, several different variants have been devised. Unfortunately, until now the only successful approaches required hybridizations or the incorporation of knowledge of the problem in the design of the genetic operators. In this paper, we show that one of the main reasons for the improper behavior of simple EAs when solving Sudoku puzzles is the appearance of premature convergence. A simple memetic algorithm with readily available genetic operators is designed and several mechanisms to delay the convergence are compared. Computational results show that, even without the incorporation of problem-dependent knowledge, the most difficult Sudoku puzzles known to date can be solved. A recent survivor selection method (MULTI_DYN) that simultaneously takes into account the contribution to diversity and the objective function of an individual yielded a remarkable performance. In fact, it could solve the most difficult Sudoku puzzles known to date with a 100% success rate, which represents a remarkable achievement. Given that this method has been adopted successfully with other problems, these results are a testament to the generality and robustness of MULTI_DYN. However, since this method converges much slower than other schemes, such as *clearing* or SAW-TOOTH GA, it takes longer than said methods for easy instances.

Several lines of future work might be explored. First, in order to further analyze the generality of our scheme, we would like to apply it to different combinatorial and continuous problems. Second, our experiments have shown that due to the use of the multi-objective replacement approach, there is a time penalty involved in solving the easiest Sudoku puzzles. In order to alleviate this penalty, the use of some of the principles behind the design of SAW-TOOTH GA seems promising. Finally, we would like to develop some coarse-grained parallel schemes that include the principles of MULTI_DYN to explicitly control the diversity of the different sub-populations.

Acknowledgments

The authors acknowledge the financial support from CONCYTEG as part of the "Investigadores Jóvenes - DPP-2014" plan (project 14-IJ-DPP-Q182-11).

REFERENCES

[1] A. Eiben and J. Smith, *Introduction to Evolutionary Computing*, ser. Natural Computing Series. Springer, 2003.

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

[3] 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.

[4] 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.

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

[6] 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. Sink, P. Hartono, M. Virkov, J. Vak, and R. Jaka, Eds. Springer International Publishing, 2015, vol. 316, pp. 207–213.

[7] 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.

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

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

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

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

[12] 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.

[13] 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.

[14] C. Segura, C. Coello Coello, E. Segredo, and A. Aguirre, "A novel diversity-based replacement strategy for evolutionary algorithms," *IEEE Trans. Cybern.*, vol. In Press, no. 99, pp. 1–14, 2015.

[15] T. Blickle and L. Thiele, "A Comparison of Selection Schemes used in Evolutionary Algorithms," *Evolutionary Computation*, vol. 4, no. 4, pp. 361–394, 1996.

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

[17] A. K. Qin, V. L. Huang, and P. Suganthan, "Differential evolution algorithm with strategy adaptation for global numerical optimization," *IEEE Trans. Evol. Comput.*, vol. 13, no. 2, pp. 398–417, April 2009.

[18] 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.

[19] 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.

[20] S. W. Mahfoud, "Crowding and preselection revisited," in *Parallel problem solving from nature 2*, R. Männer and B. Manderick, Eds. Amsterdam: North-Holland, 1992, pp. 27–36.

[21] 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.

[22] A. Petrowski, "A clearing procedure as a niching method for genetic algorithms," in *Proceedings of IEEE International Conference on Evolutionary Computation 1996 (CEC'96)*, May 1996, pp. 798–803.

[23] 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.

[24] M. Lozano, F. Herrera, and J. R. Cano, "Replacement strategies to preserve useful diversity in steady-state genetic algorithms," *Information Sciences*, vol. 178, no. 23, pp. 4421 – 4433, 2008, including Special Section: Genetic and Evolutionary Computing.

[25] L. T. Bui, H. A. Abbass, and J. Branke, "Multiobjective optimization for dynamic environments," in *2005 IEEE Congress on Evolutionary Computation*, ser. CEC'05, vol. 3, 2005, pp. 2349 – 2356 Vol. 3.

[26] C. Segura, C. A. Coello Coello, G. Miranda, and C. León, "Using multi-objective evolutionary algorithms for single-objective optimization," *4OR*, vol. 11, no. 3, pp. 201–228, 2013.

[27] C. Segura, C. A. C. Coello, G. Miranda, and C. León, "Using multi-objective evolutionary algorithms for single-objective constrained and unconstrained optimization," *Annals of Operations Research*, p. In Press, 2015.

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

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

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

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

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

[33] 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.

[34] 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.

[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 *Evolutionary Computation, 2008. CEC 2008. (IEEE World Congress on Computational Intelligence). IEEE Congress on*, June 2008, pp. 4053–4060.

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

[38] 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.

[39] C. Segura, C. Coello, E. Segredo, G. Miranda, and C. Leon, "Improving the diversity preservation of multi-objective approaches used for single-objective optimization," in *Evolutionary Computation (CEC), 2013 IEEE Congress on*, June 2013, pp. 3198–3205.

[40] 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.

[41] [Online]. Available: http://www.websudoku.com/ Last visited on 21-01-2016

[42] [Online]. Available: http://www.sudoku-solutions.com/ Last visited on 21-01-2016

[43] [Online]. Available: http://www.sudokuwiki.org/Arto_Inkala_Sudoku Last visited on 21-01-2016