Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 20 (2013) 58 – 64

Procedia Computer Science

Complex Adaptive Systems, Publication 3
Cihan H. Dagli, Editor in Chief
Conference Organized by Missouri University of Science and Technology
2013- Baltimore, MD

# EA-EMA Optimization Applied to Killer Sudoku Puzzles

David Haynes[a]*, Steven Corns[b]

[a]*Aclara, 945 Hornet Dr., Hazelwood MO 63042, USA*
[b]*MS&T, 205 Engineering Mgt, Rolla, MO 65409, USA*

**Abstract**

This paper studies techniques to reduce the search space when an optimizer seeks an optimal value. The paper promotes a new Evolutionary Algorithm (EA) mutation technique called the "Exponential Moving Average" algorithm (EMA). The paper compares its performance to two other similar Computational Intelligence (CI) algorithms to solve a multi-dimensional problem which has a large search space. Testing of the various algorithms is performed against the same Killer Sudoku puzzle and the results compared. The EMA-based solver outperforms an ordinary Evolutionary Algorithm based solver and a "Mean-Variance Optimization" (MVO) solver.


Selection and peer-review under responsibility of Missouri University of Science and Technology
*Keywords*: Computational Intelligence; Evolutionary Computation; Games

## 1. Introduction

Classic Sudoku is a game played in which numbers are arranged in a 9x9 matrix. An example Sudoku puzzle is shown in Table 1(a). Some number of cells are blanked out and not revealed to the game player. The player must guess the missing numbers in the matrix in such a way that:

Rule 1: Every row must contain the numbers 1 through 9.
Rule 2: Every column must contain the numbers 1 through 9
Rule 3: The numbers 1 through 9 must be contained in each nonet.

There is also a fourth rule which perhaps often goes unspoken:

Rule 4: Every cell that contains an initial value which is given as part of the solution may not be altered.

Killer Sudoku adds another rule to the classic Sudoku game:

Rule 5: The cell(s) within a cage must sum to the given value. Table 1(b) contains example coloured cages. The required sum is shown in small print based on data from Table 1(a).

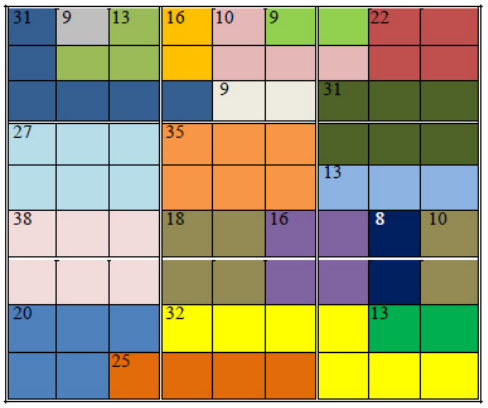
Killer Sudoku can be easier to solve for a given number of cell clues provided. But by the same token, difficult puzzles are easily created by supplying no (or sparse) cell clues.


Selection and peer-review under responsibility of Missouri University of Science and Technology
doi:10.1016/j.procs.2013.09.239

Table 1 – (a) Solved Sudoku puzzle, (b) Example Killer Sudoku Cages

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 5 | 9 | 4 | 7 | 1 | 6 | 3 | 8 | 2 |
| 6 | 8 | 1 | 9 | 2 | 3 | 4 | 7 | 5 |
| 7 | 2 | 3 | 8 | 5 | 4 | 6 | 9 | 1 |
| 1 | 3 | 8 | 5 | 4 | 9 | 2 | 6 | 7 |
| 4 | 5 | 6 | 2 | 7 | 8 | 9 | 1 | 3 |
| 2 | 7 | 9 | 6 | 3 | 1 | 8 | 5 | 4 |
| 9 | 4 | 7 | 1 | 8 | 2 | 5 | 3 | 6 |
| 8 | 1 | 2 | 3 | 6 | 5 | 7 | 4 | 9 |
| 3 | 6 | 5 | 4 | 9 | 7 | 1 | 2 | 8 |

(a)




(b)

## 2. Problem Description

A search algorithm which draws symbols from a set of 9 possibilities and places them into a blank 9x9 matrix creates approximately a $9^{9^9} \approx 10^{77}$. This is the upper bound for a puzzle, but nevertheless typical of a completely blank Killer Sudoku puzzle. If cell clues are provided, and they are used to reduce the potential candidates it can reduce the search space to a much smaller number. It's been calculated that when the rules are enforced there are $10^{21}$ possible legitimate (classic 9x9) Sudoku grids. [1] Sudoku is considered an NP-Complete problem. [2]

Solvers examining a large search space can fail to converge. The EMA algorithm leverages failed solutions to identify combinations that don't work and guide the mutation process toward solutions that do work.

### *2.1. Methodology Description and Implementation*

All three solvers follow a similar representation. Each method uses the same fitness function, search space reduction technique, and crossover operator. Where they vary is in the mutation operation.

#### *2.1.1. Fitness Functions*

Fitness for all three solvers is calculated in the same manner, by counting the number of duplicated symbols in string "c". The fitness function for a row is found in (1), the fitness function for a column in (2), the fitness for a nonet in (3), cage fitness in (4), cell fitness in (5), and chromosome fitness in (6).

$$f_{rows}(c_{ij}) = \sum_{k=1}^{9} \begin{cases} 1, if\ c_{ijk} \in B \\ 0, otherwise \end{cases} \quad (1)$$

Where "$c_{ij}$" is the $j^{th}$ element (row, column, cell, etc.) to be examined belonging to chromosome "i". "B" is the set of elements in string $c_{ij}$ generated as the string is scanned according to index "k".

$$f_{cols}(c_{ij}) = \sum_{k=1}^{9} \begin{cases} 1, if\ c_{ijk} \in B \\ 0, otherwise \end{cases} \quad (2)$$

A nonet is a set of nine blocks in a 3x3 arrangement, as shown by the alternating colored regions of the Sudoku grid shown in figure 1(a).

$$f_{nonets}(c_{ij}) = \sum_{k=1}^{9} \begin{cases} 1, if\ c_{ijk} \in B \\ 0, otherwise \end{cases} \quad (3)$$

$$f_{cages}(c_{im}) = \sum_{m=1}^{numberOfCages} (abs(GivenCageTotal_m - ComputedCageTotal_m)) \quad (4)$$

$$f_{Cell}(c_{ijk}) = \left(f_{rows}(c_{ijk}) + f_{cols}(c_{ijk}) + f_{nonets}(c_{ijk}) + \frac{f_{cages}(c_{ijk})}{cageSize(c_{jk})}\right) \quad (5)$$

Where chromosome "i" contains row "j" and column "k".

The individual cell fitness is a function of the error of the row, column, nonet, and cage fitnesses. The cage computation is different than the other error functions, and in order to prevent it from overwhelming the other errors, its error is reduced in proportion to the cage size.

*2.1.1.1. Chromosome Encoding and Fitness*

The chromosome for the CI algorithms is encoded as a 9x9 matrix as described by **Error! Reference source not found.**(a). Each cell in the chromosome corresponds to a cell in the puzzle.

$$f_{chromosome}(c_i) = \sum_{j=1}^{9} \sum_{k=1}^{9} f_{cell}(c_{ijk}) \quad (6)$$

Chromosome "i" has a fitness function which is simply the sum of the constituent fitness functions of every gene in the chromosome. Each cell in the matrix is one gene in the chromosome. A solution which perfectly satisfies all of the rules will have a fitness function of zero. In addition to the entire chromosome having a fitness of zero; the fitness of every row, column, nonet, cage difference, and cell will also be zero. A fitness of zero has to be considered a stopping condition for the search.

*2.1.2. Restriction of Solution Search Space*

The space to be searched can be reduced early-on by identifying infeasible candidates. [3] The approach used here is defined in [4], where the set {1…9} is reduced to a smaller set, for each individual cell, based on the given conditions of neighbouring rows, columns, and other cells common to the nonet. No attempt was made to leverage clues given by the cage sum. Each chromosome in the population was evaluated according to all of the fitness formulae. The chromosome with the best fitness according to (5) was selected with an elitist policy. An offspring was then generated using mutation (asexual reproduction.) Once a chromosome has been selected as the parent, the offspring are generated by mutation.

*2.1.3. Cell Selection*

The CI algorithms used cell fitness to influence the probability of selecting the cell(s) to be mutated.

Copy most fit parent as new offspring "$c_i$".
UpperBound = The count of cells in puzzle which have candidateQty > 1.
m = a random number in the range [1, UpperBound ] to mutate.
FOR m cells
$mff(c_{ijk}) = f_{cell}(c_{ijk}) \times \check{x}_{jk} \times rand$
Select cell with highest mff for mutation
Mutate selected cell using EA, MVO, or EMA technique
ENDFOR
;Offspring completed.

Fig. 1 -- Psuedocode for cell selection

*2.1.4. Mutation*

Once a cell has been selected, its mutation is guided by the particular technique of the CI algorithm. Each algorithm utilizes a different technique. For the Evolutionary Algorithm, A candidate is selected at random (with uniform random distribution) from the candidate set determined for the cell in (7). The EA was also used to generate a baseline by drawing candidates at random from the domain set. The Mean Variance Optimization Algorithm uses a mutation operator described in [5]. The adjustments made to the algorithm to create the Sudoku solver were described in [4]. The Exponential Moving Average is different from the classic EA in that it attempts to guide the mutation process.

*EMA mutation -- theory of operation.*

Very complex puzzles can lead to inordinate search times and solutions that do not converge. The solver can spend considerable computational time finding inferior solutions. These computations need not be wasted. They can be used to identify choices which tend to result in poor performance. This can then be used to inform the mutation algorithm to avoid choices that tend to perform poorly. This is accomplished by maintaining statistics on each candidate's performance as used within each cell, then using the information to guide the solver to a superior solution.

The EMA can be viewed as a time-series geometric progression. The historical result is multiplied by a retention constant "r", while the most recent fitness function is multiplied by (1-r). (Where r < 1).

$$h_t = h_{t-1} r + f_t(1-r) \tag{7}$$

The Exponential Moving Average is known to be a very efficient means of forecasting an outcome. [3] It retains useful knowledge but ages it – giving precedence to recent findings.

*EMA applied to mutation*

```
WHILE (solution not obtained or maximum number of iterations not reached)
  Perform CI algorithm which uses an elitist strategy to retain the most fit chromosome.
    Compute cell fitness functions (1) through (6) including normalized cell fitness f_c and
    AverageNormalizedCellFitness f̄_c.
      IF (no improvement has occurred in the last iteration)
        THEN
          FOR each chromosome in population
            FOR each candidate used in each cell
              h_t=h_t-1*r+f_c*(1-r)    ;update historical average
            ENDFOR
          ENDFOR
       ELSE  ;reset historical average so all are the same
      h_t= f̄_c for current evaluation
    ENDIF
ENDWHILE
```

Fig. 2 -- Psuedocode for EMA historical average calculation

The cell's historical average fitness h(i) is normalized over the range (0,1). Mutation is performed by using the historical moving average and a random function to create a "modified fitness function" (mff). The mffs are then sorted to identify the candidate with the best fitness:

```
EMA_Contribution = rand
FOR every candidate of selected cell "i"
  mff(i) = h_t(i) + EMA_Contribution × rand
ENDFOR
Sort "mff(i)" to obtain most fit selection.
Use most fit candidate as mutation value.
```

Fig. 3 – Psuedocode for EMA mutation

## 3. Tests and Results

### *3.1. Testing the algorithms for efficacy*

The EA, MVO, and EMA solvers were tested for their ability to solve Killer Sudoku puzzles. In order to get comparable test results, all algorithms were set to operate in the same manner:

- The population of all CI algorithms was limited to '2'. (Classic EA ordinarily has a much larger population in order to facilitate crossover.)
- All algorithms performed crossover by relying on asexual reproduction from a single parent.
- A large number of evaluations were allowed during each trial (10,000.)
- Fifty (50) trials were performed for each test and the results averaged.

#### *3.1.1. EA performance comparison*

In order to generate a baseline for comparison, an Evolutionary Algorithm was used to search each blank cell over the entire domain space {1,2,3…9}. Each candidate was selected uniformly at random. The results are summarized in Table 2(a). Next, the EA solvers were allowed to draw from the restricted Candidate Set, with results shown in table 2(b). Results of EA solvers searching for "n" randomly imposed blanks in a Killer Sudoku puzzle and drawing from the restricted Candidate Set. For the final experiment, the EA solvers were replaced with MVO solvers.

Table 2 -- Results of EA solver searching for "n" randomly imposed blanks in a Killer Sudoku puzzle and drawing from (a) the unrestricted domain set {1..9}, (b) the restricted Candidate Set, (c) using the MVO solvers rather than EA and drawing from the Candidate Set, and (d) using EMA solvers rather than EA and drawing from the Candidate Set.

| # of blanks | Best fitness | | | | Trial Size | |
|---|---|---|---|---|---|---|
| | Min | Mean | σ | Max | Mean | σ |
| 10 | 0 | 0 | 0 | 0 | 648 | 707 |
| 20 | 0 | 1.23 | 4.25 | 24.26 | 2708 | 3419 |
| 30 | 0 | 11.4 | 10.9 | 34 | 9324 | 2133 |
| 40 | 14.9 | 35.8 | 7.6 | 5.65 | 10k | 0 |
| 50 | 27.2 | 45.6 | 8.1 | 63.9 | 10k | 0 |
| 60 | 33.7 | 52.1 | 8.2 | 68.8 | 10k | 0 |
| 80 | 43 | 56.1 | 7.5 | 74.6 | 10k | 0 |

(a)

| # of blanks | Best fitness | | | | Trial Size | |
|---|---|---|---|---|---|---|
| | Min | Mean | σ | Max | Mean | σ |
| 10 | 0 | 0 | 0 | 0 | 1 | 0 |
| 20 | 0 | 0 | 0 | 0 | 86.8 | 127 |
| 30 | 0 | 0 | 0 | 0 | 860 | 1011 |
| 40 | 0 | 0.583 | 1.03 | 4 | 4975 | 3529 |
| 50 | 20 | 41.9 | 8.42 | 57 | 9952 | 332 |
| 60 | 51 | 71.2 | 10.2 | 95 | 10k | 0 |
| 80 | 63 | 83.0 | 10.56 | 107 | 10k | 0 |

(b)

| # of blanks | Best fitness | | | | Trial Size | |
|---|---|---|---|---|---|---|
| | Min | Mean | σ | Max | Mean | σ |
| 10 | 0 | 0 | 0 | 0 | 1.5 | 1.0 |
| 20 | 0 | 0 | 0 | 0 | 11.46 | 8.06 |
| 30 | 0 | 0 | 0 | 0 | 745 | 1399 |
| 40 | 0 | 0 | 0 | 0 | 200 | 103 |
| 50 | 0 | 14.5 | 6.6 | 25 | 9852 | 1043 |
| 60 | 18 | 32 | 7.6 | 49 | 10k | 0 |
| 80 | 60 | 79.6 | 9.57 | 109 | 10k | 0 |

(c)

| # of blanks | Best fitness | | | | Trial Size | |
|---|---|---|---|---|---|---|
| | Min | Mean | σ | Max | Mean | σ |
| 10 | 0 | 0 | 0 | 0 | 1.5 | 1.0 |
| 20 | 0 | 0 | 0 | 0 | 11.46 | 8.06 |
| 30 | 0 | 0 | 0 | 0 | 745 | 1399 |
| 40 | 0 | 0 | 0 | 0 | 200 | 103 |
| 50 | 0 | 14.5 | 6.6 | 25 | 9852 | 1043 |
| 60 | 18 | 32 | 7.6 | 49 | 10k | 0 |
| 80 | 60 | 79.6 | 9.57 | 109 | 10k | 0 |

(d)

The results shown above in tabular form are depicted below in graphical form. Fig. 4 shows baseline performance and Fig. 5 with solvers drawing from a restricted candidate set. Efficiencies are depicted in Fig. 6.

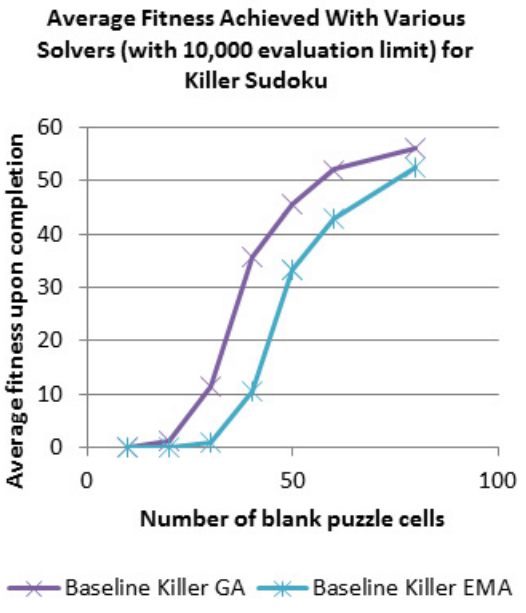



Fig. 4 -- End result comparison of Killer Sudoku solutions using unrestricted candidates

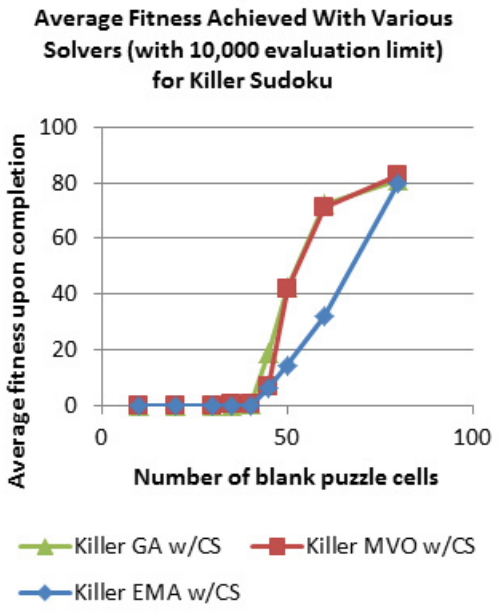



Fig. 5 -- End result comparison of Killer Sudoku solutions using a restricted candidate set

## 4. Discussion of Results

The MVO and EMA algorithms both significantly outperformed the classic EA algorithm when analysing classic Sudoku puzzles. [4] The EMA algorithm continued to perform well when applied to Killer Sudoku puzzles. This is true when the list of candidates from which each cell solution is formed is restricted at the outset or not.

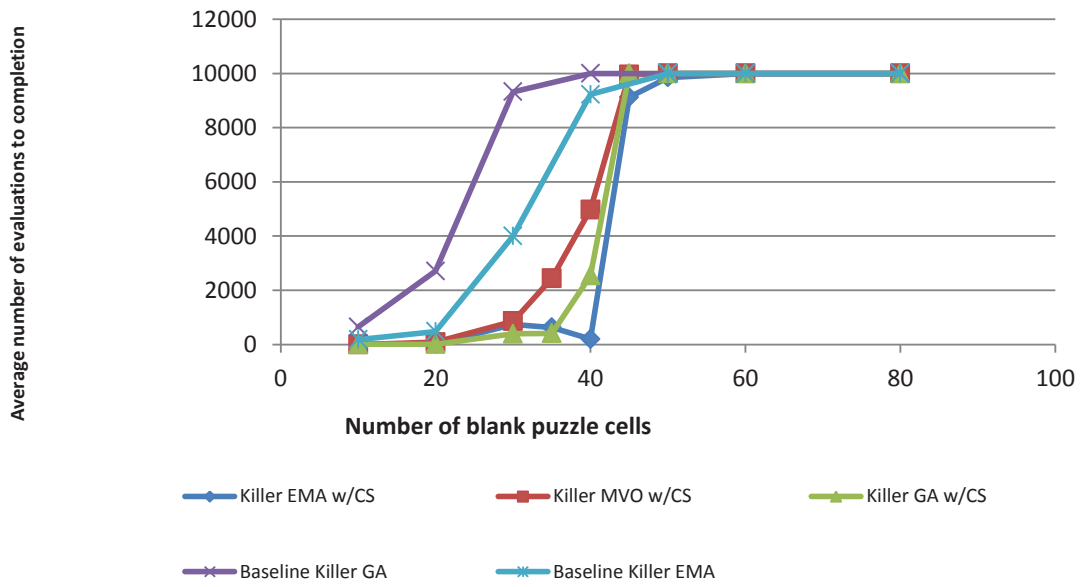



Fig. 6 -- Effort comparison in Killer Sudoku solution search (10,000 evaluation limit)

## 5. Conclusion and Future Work

It is interesting to note that while CI techniques are genetically inspired, the forecasting and selection technique presented here of an "exponential moving average" finds its roots in the world of finance. [7] The technique was found to be useful here in guiding the mutation process away from choices which have shown to be unproductive. There are many other forecasting techniques from the world of finance that could be applied to CI mutation and candidate selection. [8] [9] EMA-guided mutation can also find other applications beyond the example given here where text is analyzed to find an optimal solution within a large search-space.

The method presented here is one that can be generalized to many other applications using the psuedocode supplied above. A complex adaptive system could find the need to search a large space for an optimal solution. The constraints imposed by the problem can be formulated into chromosomes, their fitness evaluated, and the EMA evaluation history used to guide the mutation as described in the figures above.

One example of a complex adaptive system might be a communications network which self organizes. Transmitters and receivers could be uncoupled and not controlled by a centralized system. For example a new type of shortwave radio service could be created that transmits data between components. Transmitters and receivers could be constrained to operate within that spectrum, but within the allowed spectrum have a variety of frequency bands, digital transmission rates, and spreading-code encoding lengths. These could be adjusted in an effort to maximize bandwidth. (Slower transmissions and longer encodings tend to improve the success of a transmission but reduce the effective bandwidth.) These attributes could be varied to create a considerable space in which devices could operate, but at the same time create a considerable space which must be searched if a listener is to find a transmitter. A set of 40 bands, 10 rates, and code length of 20 bits would create a space of $40 \text{x} 10 \text{x} 2^{(20-2)} \approx 10^8$ places to search. To do so, each communication attribute could be encoded as a cell in chromosome. A Genetic Algorithm guided by EMA history could be used to favor settings that have been shown to work in the past, yet allow new configurations to be explored. Changes by transmitters could be loosely followed by receivers without centralized reconfiguration.

## References

[1] D. Eppstein, "Solving Single-digit Sudoku Subproblems," *arXiv:1202.5074*, Feb. 2012.
[2] L. Aaronson, "Sudoku Science," *IEEE Spectrum*, vol. 43, no. 2, pp. 16 – 17, Feb. 2006.
[3] C. C. Holt, "Forecasting seasonals and trends by exponentially weighted moving averages," *International Journal of Forecasting*, vol. 20, no. 1, pp. 5–10, Jan. 2004.
[4] D. Haynes, S. Corns, and G. Venayagamoorthy, "An Exponential Moving Average Algorithm: Evolutionary Algorithms Applied to Sudoku Puzzles," in *IEEE World Conference on Computational Intelligence*, Brisbane, Australia, 2012, vol. CFP12ICE-ART.
[5] I. Erlich, G. K. Venayagamoorthy, and N. Worawat, "A Mean-Variance Optimization algorithm," in *2010 IEEE Congress on Evolutionary Computation (CEC)*, 2010, pp. 1–6.
[6] killersudokuonline.com, "Killer Sudoku," *Killer Sudoku Online*, 29-Sep-2012. [Online]. Available: http://killersudokuonline.com/. [Accessed: 30-Sep-2012].
[7] F. R. Johnston and P. J. Harrison, "Discount Weighted Moving Averages," *The Journal of the Operational Research Society*, vol. 35, no. 7, pp. 629–635, Jul. 1984.
[8] J. W. Taylor, "Smooth transition exponential smoothing," *Journal of Forecasting*, vol. 23, no. 6, pp. 385–404, 2004.
[9] H. Markowitz, "Portfolio Selection," *The Journal of Finance*, vol. 7, no. 1, pp. 77–91, Mar. 1952.