A Novel Hybrid Flower Pollination Algorithm with Chaotic Harmony Search for Solving Sudoku Puzzles

Osama Abdel-Raouf*, Mohamed Abdel-Baset and Ibrahim El-Henawy

1Department of Operations Research, faculty of Computers and Information, Menoufia University, Menoufia, Shebin-el-Kome, Egypt.
Postal code: 32511.

2Department of Operations Research, faculty of Computers and Informatics, Zagazig University, El-Zera Square, Zagazig, Sharqiyyah, Egypt.
Postal code: 44519.

3Department of Computer Science, faculty of Computers and Informatics, Zagazig University, El-Zera Square, Zagazig, Sharqiyyah, Egypt.
Postal code: 44519.

Abstract— Flower Pollination algorithm (FP) is a new nature-inspired algorithm, based on the characteristics of flowering plants. In this paper, a new hybrid optimization method called improved Flower Pollination Algorithm with Chaotic Harmony Search (FPCHS) is proposed. The method combines the standard Flower Pollination algorithm (FP) with the chaotic Harmony Search (HS) algorithm to improve the searching accuracy. The FPCHS algorithm is used to solve Sudoku puzzles. Numerical results show that the FPCHS is accurate and efficient in comparison with standard Harmony Search, (HS) algorithm.

Keywords— Flower pollination algorithm, Harmony search, meta-heuristics, optimization, Sudoku, chaos, evolutionary algorithms.

I. INTRODUCTION

Sudoku was invented by American Howard Garns in 1979 and published by Dell Magazines as "Numbers in place" [1]. In 1984, Maki Kaji of Japan published it in the magazine of his puzzle company Nikoli. He gave the game its modern name of Sudoku, which means "Single Numbers." The puzzle became popular in Japan and was discovered there by New Zealander Wayne Gould, who then wrote a computer program that would generate Sudokus. He was able to get some puzzles printed in the London newspaper The Times beginning in 2004. Soon after, Sudoku-fever swept England. The puzzle finally became popular in the U.S. in 2005. It has become a regular feature in many newspapers and magazines and is enjoyed by people all over the globe.

The standard version of Sudoku consists of a 9 × 9 grid and 3 × 3 blocks for all the 81 cells. Each puzzle starts with some cells that already have numbers as shown in figure 3 (the numbers in white cells are originally given).

The goal of the puzzle is to find numbers for the remaining cells with three rules:

i. Each horizontal row should contain the numbers 1 - 9, without repeating any.
ii. Each vertical column should contain the numbers 1 - 9, without repeating any.
iii. Each 3 × 3 block should contain the numbers 1 - 9, without repeating any.

This problem can be mathematically represented as an optimization problem as follows [2]:

\[
\min Z = \sum_{i=1}^{9} \sum_{j=1}^{9} x_{ij} - 45 \left( \sum_{j=1}^{9} x_{ij} - 45 \right) + \sum_{j=1}^{9} \sum_{i=1}^{9} x_{ij} - 45 \left( \sum_{i=1}^{9} x_{ij} - 45 \right) + \sum_{j=1}^{9} \sum_{i=1}^{9} x_{ij} - 45
\]

(1)

where \( x_{ij} \) is a cell at row \( i \) and column \( j \), which has an integer value from 1 to 9; and \( B_k \) set of coordinates for block \( k \). The first term in Equation 1 represents the penalty function for each horizontal row; the second term for each vertical column; and the third term for each block [2]. It should be noted that, although the sum of each row, each column, or each block equals 45, it does not guarantee that the numbers 1 through 9 are used exactly once. However, any violation of the uniqueness affects other row, column, or block which contains the wrong value jointly [2].

Harmony Search (HS) is an evolutionary algorithm, which mimics musicians’ behaviors, such as random play, memory-based play, and pitch-adjusted play when they perform. HS has proved to be a powerful tool for solving several optimization problems [22-33]. It does not need any mathematical calculations to obtain the optimal solutions. In recent years, HS was been applied to many optimization problems, demonstrating its efficiency...
The Flower Pollination Algorithm (FPA) was founded by Yang in the year 2012. Inspired by the flow pollination process of flowering plants, the following rules were proposed:

**Rule 1:** Biotic and cross-pollination can be considered as a process of global pollination, and pollen-carrying pollinators move in a way that obeys Le'vy flights.

**Rule 2:** For local pollination, a biotic and self-pollination are used.

**Rule 3:** Pollinators such as insects can develop flower constancy, which is equivalent to a reproduction probability that is proportional to the similarity of two flowers involved.

**Rule 4:** The interaction or switching of local pollination and global pollination can be controlled by a switch probability $p \in [0, 1]$, with a slight bias toward local pollination.

Sudoku puzzles in this research were formulated as an optimization problem with number-uniqueness penalties. In order to compare the performance of the proposed algorithm and standard HS, the proposed algorithm was applied to solving the Sudoku puzzles taken from Figure 1 (easy level) and Figure 3 (hard level) of Geem [2] and it was found that the proposed algorithm in example 1 takes time less than standard HS, it takes 3 second after 76 function evaluations, but the standard HS takes 285 function evaluations, taking 9 seconds. In example 2 the proposed algorithm takes 6 second after 237 function evaluations with the penalty of 0 but standard harmony search entrapped in one of local optima with the penalty of 14 after 1,064 function evaluations.

This paper is organized as follows: after introduction, literature review of Sudoku is shortly displayed. In section 3, the Flower Pollination Algorithms briefly introduced. In section 4, the harmony search is briefly introduced. In section 5, the proposed algorithm is described, while the results are discussed in section 6. Finally, conclusions and future work are presented in section 7.
In order to formulate updating formulas, we have to convert the aforementioned rules into updating equations. For example, in the global pollination step, flower pollen gametes are carried by pollinators such as insects, and pollen can travel over a long distance because insects can often fly and move in a much longer range [7]. Therefore, Rule 1 and flower constancy can be represented mathematically as:

\[ x_{t+1}^i = x_i^t + \gamma L(\lambda)(x_j^t - B) \]  (2)

Where \( x_i^t \) is the pollen \( i \) or solution vector \( x_i \) at iteration \( t \), and \( B \) is the current best solution found among all solutions at the current generation/iteration. Here \( \gamma \) is a scaling factor to control the step size. In addition, \( L(\lambda) \) is the parameter that corresponds to the strength of the pollination, which essentially is also the step size. Since insects may move over a long distance with various distance steps, we can use a Lévy flight to imitate this characteristic efficiently. That is, we draw \( L \sim \) from a Lévy distribution:

\[ L \sim \frac{\lambda \Gamma(\lambda) \sin(\pi \lambda / 2)}{\pi} \frac{1}{S^{\lambda+1}}, (S \gg S_0 > 0) \]  (3)

Here, \( \Gamma(\lambda) \) is the standard gamma function, and this distribution is valid for large steps \( s \gg 0 \). Then, to model the local pollination, both Rule 2 and Rule 3 can be represented as:

\[ x_{t+1}^i = x_i^t + U(x_j^t - x_k^t) \]  (4)

Where \( x_j^t \) and \( x_k^t \) are pollen from different flowers of the same plant species. This essentially imitates the flower constancy in a limited neighborhood. Mathematically, if \( x_j^t \) and \( x_k^t \) comes from the same species or selected from the same population, this equivalently becomes a local random walk if we draw \( U \) from a uniform distribution in \([0, 1]\). Though Flower pollination activities can occur at all scales, both local and global, adjacent flower patches or flowers in the not-so-far-away neighborhood are more likely to be pollinated by local flower pollen than those faraway. In order to imitate this, we can effectively use the switch probability like in Rule 4 or the proximity probability \( p \) to switch between common global pollination to intensive local pollination. To begin with, we can use a naive value of \( p = 0.5 \) as an initially value. A preliminary parametric showed that \( p = 0.8 \) might work better for most applications [7].

The basic steps of FP can be summarized as the pseudo-code shown in Figure 1.

---

**Flower pollination algorithm**

**Define Objective function** \( f(x) \), \( x = (x_1, x_2, ..., x_d) \)

**Initialize a population of \( n \) flowers/pollen gametes with random solutions**

**Find the best solution \( B \) in the initial population**

---

**Define a switch probability \( p \in [0, 1] \)**

**Define a stopping criterion (either a fixed number of generations/iterations or accuracy)**

**while** \( t < \text{MaxGeneration} \)

**for** \( i = 1 : n \) (all \( n \) flowers in the population)

**if** \( \text{rand} < p \)

**Draw a (d-dimensional) step vector \( L \) which obeys a Lévy distribution**

**Global pollination via** \( x_{t+1}^i = x_i^t + L(B - x_i^t) \)

**else**

**Draw \( U \) from a uniform distribution in \([0, 1]\)**

**Do local pollination via** \( x_{t+1}^i = x_i^t + U(x_j^t - x_k^t) \)

**end if**

**Evaluate new solutions**

**If new solutions are better, update them in the population**

**end for**

**Find the current best solution \( B \)**

**end while**

**Output the best solution found**

---

**IV. HARMONY SEARCH ALGORITHM**

Harmony search (HS) algorithm is a metaheuristic algorithm for solving optimization problems [14]. It has been applied to solve diverse types of problems in the past years and the results were very effective compared with other meta-heuristic algorithms and traditional techniques that computationally expensive.

In HS, the Harmony Memory (HM) stores feasible vectors, which are all in the feasible space. When a musician improvises one pitch, usually one of three rules is used:

i. Generating any one pitch from his/her memory, i.e. choosing any value from harmony memory, defined as memory consideration;

ii. Generating a nearby pitch of one pitch in his/her memory, i.e. choosing an adjacent value of one value from harmony memory, defined as pitch adjustments;

iii. Generating totally a random pitch from possible sound ranges, i.e. choosing totally random value from the possible value range, defined as randomization.

Similarly, when each decision variable chooses one value in HS, it can apply one of the above mentioned rules in the entire algorithm. If a New Harmony vector is better than the worst harmony vector in HM, then the New Harmony vector replaces it. This procedure is repeated until a stopping criterion is satisfied [14]. The procedure of HS is composed of five steps:

---
Step 1: Parameters Initialization

The optimization problem is specified as follows:

Minimize \( f(x) \)

Subject to

\[ x_i \in X_i = 1.2, ..., N \]

Where \( f(x) \) is an objective function; \( x \) the set of each decision variable \( x_i \); \( N \) the number of decision variables, \( X_i \) a set of the possible range of values for each decision variable, that is, \( x_i^{\text{lower}} \leq x_i \leq x_i^{\text{upper}} \) and \( x_i^{\text{lower}} \) and \( x_i^{\text{upper}} \) are the lower and upper boundaries for each decision variable, respectively.

The algorithm requires several parameters: Harmony Memory Size (HMS), Maximum number of Improvisations (MaxImp), Harmony Memory Considering Rate (HMCR), Pitch Adjusting Rate (PAR), Bandwidth vector (Bw).

Step 2: Harmony Memory Initialization and Evaluation

The harmony memory (HM) is a memory location where all the solution vectors and corresponding objective function values are stored. In this step, the HM matrix is filled with solution vectors by taking the possible ranges into account and fitness function values are calculated for each solution vector.

Step 3: Improvise a new harmony from the HM

A new harmony vector \( \mathbf{X} = (x_1, x_2, x_3, ..., x_N) \) is generated based on three mechanisms, namely, random selection, memory consideration, and pitch adjustment. The value of a design variable can be selected from the values stored in HM with a probability HMCR. It can be further adjusted by moving it to a neighbor value of a selected value from the HM with a probability of pitch adjusting rate (PAR), or, it can be selected randomly from the set of all candidate values without considering the stored values in HM, with the probability of \((1 - \text{HMCR})\).

Step 4: Update the HM

If the new generated harmony vector is better than the current worst vector, based on the objective value and/or constraint violation, the new vector will replace the worst one.

Step 5: Stopping criterion check

If the stopping criterion (or maximum number of improvisations) is satisfied, the computation is terminated. Otherwise, Steps 3 and 4 are repeated.

V. THE PROPOSED ALGORITHM (FPCHS) FOR SOLVING SUDOKU

Newly chaos extends to various optimization areas, for example combinatorial optimization, because it can more easily escape from local minima in comparison with other stochastic optimization algorithms [15-20]. Using chaotic sequences in HS can be helpful to improve the global convergence, and to prevent getting stuck in local solutions than the classical HS algorithm which uses fixed values for HMCR, PAR and BW. In the proposed algorithm we used chaotic HS algorithm, where HMCR, PAR and BW values have been not fixed they have been modified by selecting chaotic maps. The steps of the FPCHS algorithm for solving Sudoku puzzles are shown in the figure 2.

The proposed algorithm

Begin

Define objective function \( f(x) \), \( x=(x_1,x_2,...,x_d)^T \)

Define the HS algorithm parameters.

Generate Harmony Memory by iterating the selected chaotic maps until reaching to the HMS.

while ( \( t \leq \text{max number of iterations} \) )

while ( \( i \leq \text{number of variables} \) )

if ( \( \text{rand} < \text{HMCR} \) ), Choose a value from HM for the variable \( i \)

if ( \( \text{rand} < \text{PAR} \)), Adjust the value by adding certain amount

end if

else

Choose a random value

end if

end while

Accept the new harmony (solution) if better

end while

Find the current best solution

end

The best solution found by HS is regarded as initial for FP algorithm B

Define a switch probability \( p \in [0, 1] \)

while ( \( t < \text{MaxGeneration} \) )

for \( i = 1 : n \) (all \( n \) flowers in the population)

if rand < p,

Draw a \((d\text{-dimensional})\) step vector \( \mathbf{L} \) which obeys a L’evy distribution

Global pollination via \( x_i^{t+1} = x_i^t + \mathbf{L}(B - x_i^t) \)

else

Draw \( U \) from a uniform distribution in \([0,1]\)

Do local pollination via \( x_i^{t+1} = x_i^t + U(x_j^t - x_k^t) \)

end if

Evaluate new solutions

If new solutions are better, update them in the population

end for

Find the current best solution B

end while

End

Fig. 2 Pseudo code of the FPCHS algorithm
VI. APPLICATIONS

Numerous examples have been used to test the performance of the proposed algorithm. To begin with, the initial parameters are set as follows: HMS=50 and MaxImp=10000. We performed 70 runs on all puzzles. The selected chaotic map for all examples is the logistic map, whose equation is shown below:

\[ Y_{n+1} = \mu Y_n (1 - Y_n) \]  

(6)

Clearly, \( Y_n \in [0,1] \) under the conditions that the initial \( Y_0 \in [0,1] \), where \( n \) is the iteration number and \( \mu = 4 \). All the experiments were performed on a Windows 7 Ultimate 64-bit operating system; processor Intel Core i5 760 running at 2.81 GHz; 4 GB of RAM and codes were written in Visual c#.

A. Example 1

This example has been introduced by Geem [2] and was successfully solved by the standard harmony search with 285 function evaluations, taking 9 seconds. In this paper the proposed algorithm could successfully solve it with a fewer number of function evaluation (only 76 function evaluations) and within three seconds.

\[
\begin{array}{cccc}
5 & 3 & 6 & 7 \\
9 & 8 & 4 & 2 \\
9 & 1 & 3 & 2 \\
5 & 7 & 2 & 6 \\
4 & 5 & 9 & 3 \\
1 & 5 & 7 & 2 \\
8 & 1 & 4 & 7 \\
\end{array}
\]

Fig. 3 Example 1 Sudoku puzzle

\[
\begin{array}{cccc}
2 & 5 & 4 & 3 \\
7 & 6 & 3 & 9 \\
1 & 9 & 8 & 4 \\
9 & 8 & 1 & 7 \\
6 & 3 & 2 & 8 \\
5 & 4 & 7 & 2 \\
4 & 7 & 5 & 6 \\
3 & 1 & 9 & 5 \\
8 & 2 & 6 & 1 \\
\end{array}
\]

Fig. 4 Final solution of example 1 Sudoku puzzle

B. Example 2

In this example, standard HS algorithm applied to this problem [20] by Geem [2] and failed to find the global optimum, as shown in figure 5[20]

\[
\begin{array}{cccc}
3 & 1 & 7 & 5 \\
5 & 6 & 8 & 4 \\
9 & 2 & 6 & 3 \\
1 & 8 & 2 & 1 \\
7 & 8 & 2 & 1 \\
5 & 7 & 4 & 6 \\
\end{array}
\]

Fig. 4 Example 2 of Sudoku puzzle

\[
\begin{array}{cccc}
3 & 8 & 4 & 2 \\
6 & 3 & 1 & 9 \\
4 & 9 & 7 & 5 \\
9 & 1 & 2 & 4 \\
5 & 2 & 9 & 6 \\
8 & 6 & 3 & 8 \\
1 & 7 & 6 & 1 \\
7 & 4 & 8 & 1 \\
2 & 5 & 6 & 1 \\
\end{array}
\]

Fig. 5 Standard HS solution for example 2

When applied the proposed algorithm for this example we found that the proposed algorithm could successfully find the global optimal without any row, column or block violation after 237 function evaluations taking 6 seconds. As shown in figure 6

\[
\begin{array}{cccc}
3 & 2 & 4 & 8 \\
5 & 6 & 1 & 2 \\
8 & 7 & 9 & 5 \\
9 & 3 & 2 & 1 \\
4 & 1 & 5 & 6 \\
6 & 8 & 7 & 4 \\
1 & 9 & 6 & 3 \\
7 & 4 & 8 & 9 \\
2 & 5 & 3 & 7 \\
\end{array}
\]

Fig. 6. Final solution of example 2 Sudoku puzzle with proposed algorithm
C. Example 3

Another Sudoku problem classified as “hard” as shown in Figure 7[21]. Also the proposed algorithm successfully found the global optimal without any row, column or block violation after 281 function evaluations taking 9 seconds. As shown in figure 8

<table>
<thead>
<tr>
<th>6</th>
<th>9</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>7</td>
<td>1</td>
<td>8</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>7</td>
<td>8</td>
<td>4</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>7</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>4</td>
<td>7</td>
</tr>
</tbody>
</table>

Fig.7. Example3 Sudoku puzzle

<table>
<thead>
<tr>
<th>4</th>
<th>8</th>
<th>6</th>
<th>9</th>
<th>1</th>
<th>5</th>
<th>7</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>9</td>
<td>1</td>
<td>7</td>
<td>4</td>
<td>3</td>
<td>6</td>
<td>5</td>
<td>8</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>7</td>
<td>6</td>
<td>2</td>
<td>8</td>
<td>4</td>
<td>9</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>4</td>
<td>8</td>
<td>2</td>
<td>5</td>
<td>1</td>
<td>3</td>
<td>6</td>
<td>9</td>
</tr>
<tr>
<td>6</td>
<td>5</td>
<td>2</td>
<td>3</td>
<td>7</td>
<td>9</td>
<td>1</td>
<td>8</td>
<td>4</td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td>3</td>
<td>8</td>
<td>6</td>
<td>4</td>
<td>2</td>
<td>7</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>7</td>
<td>4</td>
<td>5</td>
<td>9</td>
<td>6</td>
<td>8</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>8</td>
<td>6</td>
<td>5</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td>9</td>
<td>4</td>
<td>7</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>9</td>
<td>4</td>
<td>8</td>
<td>7</td>
<td>5</td>
<td>3</td>
<td>6</td>
</tr>
</tbody>
</table>

Fig.8. Final solution of example 3 Sudoku puzzle

VII. CONCLUSION AND FUTURE WORK

In this paper, the better results (i.e. solve a hard level of Sudoku and minimum numbers of iterations have been computed as a compare with paper [2] in same Sudoku examples). This paper presented an improved flower pollination algorithm by integrating it with chaotic harmony search algorithm for solving Sudoku puzzles. The HM members were fine-tuned by the chaos operator to improve their affinities. The proposed algorithm has been provided the better and clear way to find the solution of hard level of any given Sudoku.

The algorithm has been tested on a set of Sudoku problems. The results proved the superiority of the proposed methodology. The reason for getting better results in comparison with other algorithms considered is the search power of FP. In addition to that, using Chaos with harmony search helps the algorithm to escape from local solutions.

Future work includes different puzzles such as Futoshiki and KenKen can be solved using this algorithm.

Another direction can be the study of different mechanisms for adapting the HMCR, PAR, bw and P parameters. Also we can use this algorithm for solving most popular engineering optimization problems.

REFERENCES


