Modified Harmony Search Algorithm for Solving the Four-Color Mapping Problem

Bnar Faisal A. Daham
Software Engineering Dept.
College of Engineering, University of Salahaddin, Erbil - Iraq

Mohammed N. Mohammed
Surveying Engineering Dept.

Kanar Sh. Mohammed
Software Engineering Dept.

ABSTRACT

The Four-Color Mapping Problem has been solved using different optimization algorithms. Harmony Search (HS) is one of those algorithms, which is based on the imitation of the behavior of musicians when composing their music. The HS algorithm can be summarized in three steps... initialization, improvisation, and selection.

We introduced in this paper an approach to enhance the performance of HS algorithm, in solving the Four-Color Mapping Problem. A modification has been applied to the initialization section of the HS algorithm, which affects the improvisation process, resulting in a boost in the performance of the improvisation process, and consequently, reducing the time and number of cycles taken to solve the Four-Color Mapping Problem compared to the HS algorithm. In this paper, tests have been carried out on maps with different numbers of regions, using both HS and Modified Harmony Search (MHS) algorithms. The obtained results of the MHS algorithm are better than those of the original HS one.

Keywords
Optimization, Four-Color Mapping Problem, Harmony Search Algorithm.

1. INTRODUCTION

Four Color Mapping is one of the combinatorial optimization problems that was first conjectured in 1852 by Francis Guthrie [1]. The 4-Color Mapping can be said that the regions of any simple planar map can be colored with only four colors, in such a way that any two adjacent regions have different colors. Two regions are said to be adjacent if and only if a common boundary is shared, not just a common point. Figure 1 illustrates the adjacency of regions in the given sample. Regions A and B have a common boundary, same goes with regions C and D. But regions A and D only share a common point, so do regions B and C. Therefore, by definition, regions A and B are adjacent, as well as regions C and D, but not regions A and D, and regions B and C.

Figure 2 illustrates 29 regions have been colored with four colors and no adjacent regions have the same color.

As the HS algorithm randomly generates harmonies or candidate solutions, a modification is applied to it in this paper by letting it generate the harmonies, then randomly assigning and fixing - but separately from HS improvisation process - the four different colors to the first four regions. Thus saving time and cycles specifically in solving the Four-Color Mapping Problem.

In the rest of the paper; HS algorithm is overviewed in Section 2. Section 3 describes MHS algorithm, and solving the Four-Color Mapping Problem using MHS algorithm is described in section 4. In section 5, experimental results are presented. Finally, some concluding remarks are presented in Section 6.

2. HARMONY SEARCH ALGORITHM

Computer scientists have found a significant relationship between music and the process of looking for an optimal solution. This interesting connection led to the creation of the HS algorithm. It is a new kind of meta-heuristic algorithm mimicking a musicians’ approach to finding harmony while playing music [7]. When musicians try to create music, they may use one or a combination of the three possible methods for musical improvisation which are as follows: (1) playing the original piece, (2) playing in a way similar to the original piece, and (3) creating a piece through random notes [6][8].

Zong Woo Geem et al. [7][9][10], noticed the similarity of this behavior in achieving the optimal solution to a problem. In 2001, he proposed three methods corresponding to the three techniques namely the use of (1) random selection, (2) memory consideration, (3) and pitch adjustment. These became the elements of the newly developed meta-heuristic optimization algorithm called the HS algorithm [6][7]. Figure 3 illustrates the basic Harmony Search Algorithm [6][7][8].

```
Begin
Define Objective function f(x), x=(x1,x2, ...,xd)T
Define harmony memory accepting rate (r acceptance)
Define pitch adjusting rate (r pitch), pitch limits and bandwidth
Generate Harmony Memory with random harmonies
While ( t < max number of iterations (cycles))
    While ( t < max number of variables)
        If (rand < r acceptance), choose a value of var i
        If (rand < r pitch), adjust the value
    End if
    Else choose a random value
End if
End while
Accept a new harmonics (solutions) if better
End while
Choose the best solution
End
```

Figure 3: Basic Harmony Search Algorithm
Mathematically, the design of a basic HS algorithm is generally based on the following steps [10][11][12]:

Step 1. Initialize the problem and algorithm parameters.
Step 2. Initialize the Harmony Memory (HM).
Step 3. Improvise a New Harmony memory.
Step 4. Update the Harmony memory.
Step 5. Check the stopping criterion.

The capability of Harmony Search Algorithm in solving problems has been proven effective in various studies. In this study, the researchers aim to determine the feasibility of the said algorithm and its modified in solving Four-Color Mapping problem [6].

3. MODIFIED HARMONY SEARCH (MHS) ALGORITHM

In this paper an approach is introduced to enhance the performance of HS algorithm specifically for solving the Four-Color Mapping Problem. A modification has been applied to the initialization section of the HS algorithm. That is by excluding the first four elements of each vector (candidate solution) from any computations of improvisation process of HS algorithm.

After generating the harmonies, the first four elements in each of them will be assigned randomly four different values. The assignment should involve the available different values to achieve and guarantee the condition of Four-Color Mapping Problem. Also the assigned values will be fixed per a solution because they are not included in the improvisation process (Random Selection, Memory Consideration, and Pitch Adjustment). In other words, the improvisation process will treat \((n-4)\) elements, where \(n\) is the number of the regions in the processed map. As a result, the path to solve the Four-Color Mapping Problem will be shortened compared to the original HS algorithm through reducing the time and number of cycles taken to reach the optimal solution. Figure 4 and 5 illustrates the MHS algorithm and the design of general steps of the MHS algorithm respectively.

---

**Figure 4: Modified Harmony Search algorithm**

**Figure 5: Steps of Modified Harmony Search algorithm**
4. SOLVING THE FOUR-COLOR MAPPING PROBLEM USING MHS ALGORITHM

In this section, we will discuss the approach that the researchers took in achieving a four-colored map solution. Figure 6 illustrates the general flow of the proposed approach. Each candidate solution is generated as a vector whose variables correspond to the regions of the map. Vectors are arrays of integer variables that serve as the solution to the problem. Regions to be colored take values 1, 2, 3, or 4 representing the four colors where each number stands for a color in the solution such as 1=red, 2=green, 3=blue and 4=yellow. The size of the solution is equivalent to the number of regions in the map. Figure 7 shows the representation of the solution. The general formula of each candidate solution is as follows:

\[ R = <R_1, R_2, R_3 ..., R_n> \]

where \( R \) represents the regions having values 1, 2, 3, or 4 and \( n \) is the number of regions in the map. The optimization process starts by generating random solutions in the harmony memory. Equation (1) illustrates the characteristics of each candidate solution in the memory. After initializing the harmony memory, it is followed by the improvisation process. Each possible solution is evaluated for its respective objective value. Then the optimization process ends if the process generates an optimal solution or when the maximum improvisation is reached. This is where a new harmony or a possible solution is generated through random selection, harmony memory consideration or pitch adjustment.

- Harmony Memory Size: Refers to the number of solutions that will be stored in the harmony memory.
- Harmony Memory Consideration Rate: The rate at which the value of the decision variables from the harmony memory is picked as elements of the new harmony that will be created.
- Pitch Adjustment Rate: The probability that the decision variable picked from the harmony memory will be altered by some certain amount.

Table 1 shows the standard parameter values used for the experiments.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>No. of Variables</th>
<th>Cycles</th>
<th>HMC</th>
<th>HMC Rate</th>
<th>PA Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Values</td>
<td>No. of Regions in the map</td>
<td>100000</td>
<td>0</td>
<td>0.9</td>
<td>0.4</td>
</tr>
</tbody>
</table>

2. Initialize MHS

In this step we generate the initial MHS normally then randomly assign the four different colors to the first four regions as many as HMS as shown in Figure 8.

**Begin:**

**Input:**

- Integer: HMS % Harmony Memory Size
- Integer: varIndex% Index of array
- Integer: NCHV% Pattern
- Integer: NVAR% Number of Variable

**Process:**

For \( i=1 \) to HMS do

For \( j=1 \) to NVAR do

\[ MHM[i][j]=\text{random(low[j],high[j])} \]

\[ NCHV[j]=MHM[i][j] \]

End for

//Assign randomly the four different colors to the first four regions for each pattern in the MHS. Then://

\[ NCHV[1]=1 \]

\[ MHM[i][1]=1 \]

\[ NCHV[2]=2 \]

\[ MHM[i][2]=2 \]

\[ NCHV[3]=3 \]

\[ MHM[i][3]=3 \]

\[ NCHV[4]=4 \]

\[ MHM[i][4]=4 \]

\[ curFit = \text{Fitness} (NCHV) \]

\[ MHM[i][NVAR]=\text{curFit} \]

End for

**Output:** MHM[i][NVAR]

**End**
from the available four options. To be noticed, in our case, random selection, memory consideration, and pitch adjustment will start from region five and upwards.

4. Memory Consideration

Values or colors in the harmony memory have a chance to be selected when performing memory consideration. So memory consideration for region five will be the available colors for region five in the modified harmony memory.

5. Pitch Adjustment

Selected value from the memory can be altered to obtain a variation through pitch adjustment. Figure 9 illustrates how pitch adjustment is done.

```
Begin:
Input:
Public double: BestHarmony[] = get;
Double: rand();
Integer: varIndex % Index of array
Integer: Temp % temporary variable
Integer: NCHV[varIndex] % Pattern
Integer: NVAR % Number of Variable
Process:
If rand < par, then
//pick one of the neighbors of the region in question at random and exchange colors between the region and the chosen neighbor, Specifically:
    BestHarmony = new double[NVAR+1]
    Random number generator
    Temp = NCHV[varIndex]
    NCHV[varIndex] = BestHarmony[varIndex]
End if
Output: NCHV
End
```

Figure 9: Pitch Adjustment process

6. Objective (Fitness) Function

The evaluation function will return a value that will represent the fitness of a harmony vector. The function aims to get a minimum result, wherein, the smaller the fitness value, the better the vector is. The value also represents the number of errors present in the solution. If the fitness value reaches zero, then that is the optimal solution and the optimization process will stop. Figure 10 illustrates how the objective function works.

```
Begin:
Input:
Integer: N % Number of regions
Integer: X[N] % Array of colored regions
Integer: Matrix[N][N] % Array of adjacency Matrix N by N depend on number of region
Process:
For i=1 to N do
    For j=1 to N do
        if i ≠ j then
            If Matrix[i][j]=1 then
                If X[i]=X[j] then
                    Fit += 1
                End If
            End If
        End for
    End for
Output: Fit
End
```

Figure 10: Objective function process

7. Illustrating the MHS Algorithm Process to Map Coloring

Using the map in Figure 7, Initialize random harmony vectors that will represent the solution for the colored map using equation 1. In this example, we set our harmony memory to two so that HMS=2, R1=[4,1,3,2,1,1,3] and R2=[1,3,2,4,3,4]. Then improve a new harmony vector through Random Selection, Memory Consideration, and Pitch Adjustment. After that we evaluate the harmony using the objective function process in Figure 10. Only half of the matrix is evaluated since the matrix is symmetric. The smaller resulting value is the better result. R1 is evaluated as shown bellow. Using the same function, the result of R2=1 also.

```
R1 = [0×0 + 1×0 + 0×0 + 0×0 + 0×0 + 0×0 ]
    [0×0 + 1×0 + 1×0 + 0×0 + 0×0 + 0×0 ]
    [0×0 + 1×0 + 1×0 + 1×0 + 0×0 + 0×0 ]
    [0×0 + 1×0 + 0×0 + 0×0 ]
    [0×0 + 1×0 + 0×0 + 0×0 ]
    [0×0 + 0×0 + 0×0 ]
```

Both improvisation of a new harmony vector and evaluation of a new harmony are repeated until a series of finite number of improvisations is achieved. The best harmony in the harmony memory is selected to become the solution. The map will be colored using the corresponding colors of the values in the solution.

5. TESTS AND RESULTS

HS and MHS algorithms were implemented in Microsoft Visual C# 2010 and run on a computer whose processor is Intel(R) Core™ 2.50 GHz, with 6 GB main memory. The algorithms were applied on 7 regions Australia map, 10 regions Austria map, 18 regions Iraq map, 29 regions Kurdistan-Iraq map, 32 regions China map, 48 regions USA map, and finally 50 regions Africa continent map.

The comparison between the original and modified HS algorithms will depend on the time and number of cycles required to find the optimal solutions for each vector of regions. Ten to fifteen runs for each instance have been achieved and the standard deviation of time and cycles required to reach the optimal solution have been reported in Table 2.

Table 2: Standard deviation of elapsed time and number of cycles for HS and MHS algorithms for solving Four-Color Mapping problem

<table>
<thead>
<tr>
<th>Regions</th>
<th>Time</th>
<th>Cycles</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>HS</td>
<td>MHS</td>
</tr>
<tr>
<td>7</td>
<td>0.007656975</td>
<td>0.005310974</td>
</tr>
<tr>
<td>10</td>
<td>0.006988317</td>
<td>0.006027705</td>
</tr>
<tr>
<td>18</td>
<td>0.173869271</td>
<td>0.086737074</td>
</tr>
<tr>
<td>29</td>
<td>0.889335656</td>
<td>0.316231162</td>
</tr>
<tr>
<td>32</td>
<td>1.07332541</td>
<td>0.616941191</td>
</tr>
<tr>
<td>48</td>
<td>6.400353566</td>
<td>0.622102219</td>
</tr>
<tr>
<td>50</td>
<td>7.961991411</td>
<td>7.513218207</td>
</tr>
</tbody>
</table>

Table 2 shows how the MHS algorithm has higher performance than the HS algorithm. Figures 11 and 12 are illustrating the difference between the performance of both MHS & HS algorithms in reaching the optimal solution in terms of time and number of cycles respectively.
6. CONCLUSIONS

The computational results illustrate that the MHS algorithm presents better performance than the original HS algorithm where the MHS reached the optimal solution faster than the original HS applying tests on maps with different number of regions. The results also indicate that the time and number of cycles increase as the size of the map increases. This is because it will be more difficult to find an optimal solution as the size of the map gets bigger.

For future work hybridization can be applied between HS and another optimization algorithm. One of the approaches is to use Ant or Bee colony algorithms in generating the harmonies (candidate solutions) instead of producing them randomly.

7. REFERENCE


