International Journal of Scientific Research in Information Systems and Engineering (IJSRISE)
Volume 1, Issue 2, December-2015. ISSN 2380-8128

# A Performance Study for Harmony Search Algorithm

Esranur Galip, Computer Engineering Department, Gediz University,
Gediz University Seyrek Campus, Menemen, Izmir, TURKEY
E-mail: esranurgalip@gmail.com
Feyza Galip, Computer Engineering Department, Gediz University,
Gediz University Seyrek Campus, Menemen, Izmir, TURKEY
E-mail: feyzagalip@gmail.com

**Abstract** - Mathematical techniques such as Linear Programming, Non-Linear Programming, Dynamic Programming are being used for simple and ideal models to find the global optimum solution. But these techniques are not sufficient methods for real world problems. This problem causes to be developed of heuristic optimization techniques. One of these techniques is Harmony Search (HS) algorithm, developed to find better solution with decreased number of iterations against the existing optimization techniques. The algorithm has parameters such as Harmony Memory Considering Rate (HMCR), Pitch Adjustment Rate (PAR), whose values have to be predetermined before conducting the experiments. Values of these parameters play a critical role by generating solutions. To reveal this assumption, Bridge and Torch (BT) problem is solved with HS algorithm. BT is a combinatoric problem and it's optimal solution is known for several parameters. Adapted HS algorithm is run for several n, PAR and HMCR values. Experimental results show that if the best values for parameters aren't selected, the global solution may not be reached in acceptable time.

***Keywords*** *– Bridge and Torch Problem, Harmony Search Algorithm, Heuristic Algorithms, Optimization problems.*

## 1. INTRODUCTION

Mathematical techniques are not sufficient methods for finding solution of the real world problems. This problem is caused by the development of heuristic optimization techniques. One of these techniques is Harmony Search (HS) algorithm, developed to find better solution with decreased number of iterations against the existing optimization techniques. It is inspired from composing music with harmony of several notes. Defined number of (HMS) harmonies (solution) are stored into Harmony Memory (HM). While generating a new solution, Harmony Memory Considering Rate (HMCR) is the value that shows the possibility of using a part of a random solution in HM. Pitch Adjustment Rate (PAR) is used to prevent to be catched local optimum. So, these parameters play a critical role in generating solutions while searching the global solution(s) and they have to be predetermined before algorithm runs [1] [2] [3]. Unforeseen bad value may cause catastrophic results such as lack of solution or extending time of finding solution. To point out the importance, HS algorithm is adapted to solve Bridge and Torch problem (BT) which is a combinatoric problem and it's optimal solution is known for several parameters. Adapted HS algorithm is run for several *n*, PAR and HMCR values and experimental results are compared. It can be said that HS algorithm's parameters should be selected carefully by taking properties of the problem to be solved into account. Because unsuitable values caused it to reach the global solution in a very long time. For different *n* value, it causes not to find global optimum solution.

This paper is organized as follows. In section 2, BT problem and HS algorithm are discussed in detail. Adapted harmony search algorithm, it's solution structure are given in Section 3. Experimental results and conclusion are covered in Section 4 and Section 5, respectively.

## 2. BACKGROUND

### 2.1. Bridge and Torch Problem

The problem is about accrossing a bridge of *n* people at night. There is only one torch to be used and it has a capacity. Hereby, it cannot host number of people more than it's capacity. The aim of the problem is to find the minimum transfer time of all people under given constraints such as capacity. So, the solution should organize people transfer by taking cross time of people into account. If we define

Corresponding Author
Esranur Galip, Computer Engineering Department, Gediz University, Gediz University Seyrek Campus, Menemen, Izmir, TURKEY
E-mail: esranurgalip@gmail.com

transfer time of $person_i$ as $t(i)$, transfer time of a group of people is equal to the slowest person's transfer time ($\min(t(i))$, $0<i<n+1$) [4]. In Table 1. Example for Bridge and Torch Problem [5]1, there is an example, where $n$=4 and capacity of the torch is 2. As seen in the table, in the first step, two fastest people (A, B) in the group cross the bridge. In the second step, the fastest person (A) returns to the starting side. Then two slowest people (C,D) in the starting side go to the ending side in the third step. The fastest member (B) in the ending side returns to the starting side in the forth step. Until no person is left in the starting time, the process is repeated from the first step.

**Table 1.** Example for Bridge and Torch Problem [5]

| Person | *t(i)* (min.) |
|---|---|
| A | 1 |
| B | 2 |
| C | 5 |
| D | 8 |

| Elapsed Time | Starting Side | Action | Ending Side |
|---|---|---|---|
| 0 min. | A B C D | *No Action* | ∅ |
| 2 min. | C D | A and B cross the bridge | A B |
| 3 min. | A C D | A returns | B |
| 11 min. | A | C and D cross the bridge | B C D |
| 13 min. | A B | B returns | C D |
| 15 min. | ∅ | A and B cross the bridge | A B C D |

*2.2. Harmony Search Algorithm*

In optimization problems, mathematical techniques such as Linear Programming, Non-Linear Programming, Dynamic Programming are being used for simple and ideal models to find the global optimum solution. But these techniques are not sufficient methods for the real world problems. Hereby, heuristic optimization techniques have been developed to find solution close to the global optimum in acceptable time and usage of the memory. Harmony Search (HS) algorithm is a meta-heuristic technique developed by Geem in 2001 [2], to obtain better solution with decreased number of iterations against the existed optimization techniques. In HS, harmony memory (HM) is used to hold the generated solutions. Each row of the HM (see Figure 1. Appearance of Harmony Memory [1].1) denotes a solution, each column shows a value of a solution's variable and also each solution has a function value which is tried to be minimized or maximized according to the problem. The algorithm has Harmony Memory Size (HMS), Harmony Memory Considering Rate (HMCR), Pitch Adjustment Rate (PAR) parameters whose values have to be predetermined. HMS shows the number of the solutions should be held in HM. HMCR is the possibility value of using a value of a solution in HM by generating a new solution. PAR is used to prevent the algorithm from being catched to the local optimum. It is the possibility value of changing the solution part with a neighbour value or adding random value.

$$HM = \begin{bmatrix} x_1^1 & x_2^1 & \dots & \dots & x_{n-1}^1 & x_n^1 \\ x_1^2 & x_2^2 & \dots & \dots & x_{n-1}^2 & x_n^2 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ x_1^{HMS-1} & x_2^{HMS-1} & \dots & \dots & x_{n-1}^{HMS-1} & x_n^{HMS-1} \\ x_1^{HMS} & x_2^{HMS} & \dots & \dots & x_{n-1}^{HMS} & x_n^{HMS} \end{bmatrix} \begin{matrix} \rightarrow f(x^1) \\ \rightarrow f(x^2) \\ \rightarrow \dots \\ \rightarrow f(x^{HMS-1}) \\ \rightarrow f(x^{HMS}) \end{matrix}$$

**Figure 1.** *Appearance of Harmony Memory* [1].

The first step of the algorithm is initilization of HM. It includes HMS number random generated solutions. In the second step, a new harmony is generated. For each value of the variable, a random number between 0 and 1 is generated. If the generated number is equal to or less than HMCR, random selected value of corresponding variable of solution in HM is assigned to new solution's variable value. Otherwise a value between possible values is randomly selected. After finding value of the variable, another random number between 0 and 1 is generated. If the generated number is equal to or less than PAR, according to the problem, the value is changed with one neighbour value . In the third step, function value of newly generated solution is compared with the lowest function value of the solution in HM. If function value of newly generated solution is better than the solution with lowest function value in the HM, the solution is changed with newly generated solution. The fourth step is a decision step. If the algorithm reaches the global optimum or the stopping condition is assured, the algorithm is terminated and the solution with the best function value is the solution of the problem. Otherwise the algorithm continues with the second step [1] [2] [6].

HS algorithm is commonly used in optimization problems of several research areas. One example for these research areas is computer science. Bouzidi et al.[1] study adapts HS algorithm to solve Travelling Salesman Problem (TSP). They run the algorithm several times with different PAR values to find suitable one and find *0.45* as PAR value which minimizes the average execution time of the

algorithm while searching the solution. The other study is about optimization problem in transportation engineering [6]. In this study, they construct the problem and the function which should be minimized. No study is done for finding any suitable HMCR, PAR duple. If they had done this study, maybe they would have found the solution in less execution time. In [3] study, they solve 0-1 knapsak problem using HS algortihm. They know the importance of HMCR value in searching global optimum solution. So, they developed simple mechanishm which generates HMCR value according to declared equation and developed algorithm updates dynamically HMCR value with the geneterated value.

## 3. METHODOLOGY

In this study, one of the heuristic search algorithms, Harmony search algorithm, is adapted to bridge and torch problem. It aims to find the global optimum solution by minimizing function value and observe time of reaching the solution for different values of HMCR, PAR and *n*. In Figure 2. Adapted Harmony Search Algorithm 2, the adapted harmony search algorithm is given. Firstly, the algorithm starts to run with initiliazing HM with random solutions. In Figure 3. Meaning of the Harmony solution with an example.3, solution example is given. The algorithm continues running until reaching global optimum solution. In step of generating a solution, if the solution part is about finding person who should cross the bridge and the possibility of HMCR is satisfied, solution part is selected from a solution in HM which is randomly selected. The person is selected from starting side randomly. If solution part is selected from existing solutions and if possiblity of PAR is satisfied, one person is selected from starting side and solution part is changed with new person. After completing the solution, it's function value is evaluated. If it is better than the worst function value in the HM, corresponding solution is omitted and new solution is located to HM.

```
Algorithm: Adapted Harmony Search Algorithm
1.  Initiliaze HM with random solutions
2.  while(F(the best solution in HM) not equal to global optimum)
3.      for(i = 1, i <= solution.length, i + +)
4.          if(i mod C > 0) then
5.              if (P_HMCR < HMCR) then
6.                  solution[i] ← HM[random index][i]
7.                  if(P_PAR < PAR) then
8.                      solution[i] ←select one person from the starting side
9.                  end if
10.             else solution[i] ←select one person from the starting side
11.             end if
12.         else solution[i] ← select one person from the ending side
13.         end if
14.     end for
15.     if(F(solution) better than F(worst solution in HM)) then
16.         remove worst solution from HM
17.         add solution to HM
18.     end if
19. end while
20. return best solution in HM
```

**Figure 2.** *Adapted Harmony Search Algorithm*

Each row of HM denotes generated solutions. Each column value of the solution shows person identity (id such as A, B, etc.). In Figure 3. Meaning of the Harmony solution with an example.3, an adapted harmony solution example is given. As seen in the figure, as many columns as the number of the capacity denotes ids of people who will cross the bridge. The next column shows the id of the person who will return back to the starting side. Function value is equal to elapsed time of all people transfer.

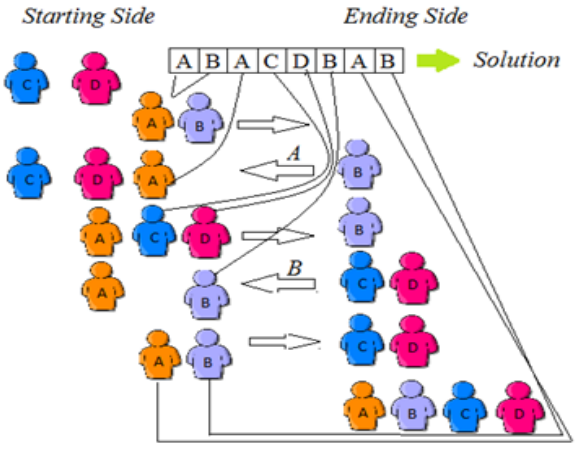


**Figure 3.** *Meaning of the Harmony solution with an example.*

## 4. EXPERIMENTAL RESULTS

Adapted harmony search algorithm is run by setting several values to corresponding parameters. But in advance, parameters' values have to be determined. For instance, one of them is HM. We set it to it's typical value 30. The other parameters, HMCR and PAR, generally vary from 0.7 to 0.99 and 0.1 to 0.5 respectively [1][7]. Their typical values

are HMCR=0.9 and PAR=0.3. Using typical values, first of all we run the algorithm 100 times for n=5,6,7,8, and 9, respectively. We recognize that if *n* increases, it takes a long time and get hard to reach the global optimum solution. Even, it may cause inaccessible global solution. The average number of iterations and average time results for several *n* values is given in Table 2. The results for several n values with HMCR=0.9 and PAR=0.32. According to the table, we can say that average time is extremely increasing for each increment of *n* value. We can reach the global solution for maximum n=9 with HMCR=0.9 and PAR=0.3.

**Table 2.** *The results for several n values with HMCR=0.9 and PAR=0.3*

| *n* | Average number of iteration | Average execution time(sc) |
|---|---|---|
| 5 | 273.53 | 0.04553 |
| 6 | 1916.73 | 0.31089 |
| 7 | 75091.92 | 14.24517 |
| 8 | 761678.65 | 144.80298 |
| 9 | 26860093.05 | 5736.50434 |

To find the best HMCR-PAR duple to reach global solution for n=9,10, etc., we observe average time of experiments' results for HMCR=0.9 (see Table 3. Average execution time of the algorithm with HMCR=0.9 for several n values3). Yellow highlighted values show the minimum average execution times for related *n* value and – means no solution in the table. The true, PAR=0.1 gives the best performance in HMCR=0.9, can be found if the table is examined. With the light of this outcome, we determined to detail the experiments with HMCR={0.9, 0.95, 0.99} and PAR={0.1, 0.15} to obtain the best average execution time for *n*>8.

**Table 3.** *Average execution time of the algorithm with HMCR=0.9 for several n values*

| *n* | PAR | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | *0.1* | *0.15* | *0.2* | *0.25* | *0.3* | *0.35* | *0.4* | *0.45* | *0.5* | Average Execution Time (sc) |
| 5 | 0.0372 | 0.0418 | 0.0440 | 0.0441 | 0.047 | 0.0371 | 0.0524 | 0.0537 | 0.0575 | |
| 6 | 0.2156 | 0.2418 | 0.2670 | 0.297 | 0.3109 | 0.377 | 0.4407 | 0.4463 | 0.4753 | |
| 7 | 5.5491 | 7.0881 | 8.1401 | 11.29 | 14.2452 | 16.9865 | 18.8944 | 24.9793 | 28.5362 | |
| 8 | 29.9037 | 43.7883 | 62.5623 | 102.4577 | 144.803 | 206.4953 | 226.7719 | 352.869 | - | |

Table 4. Average execution time of algorithms for several n values with different HMCR-PAR duples4 includes average times of experiments where {HMCR, PAR} duple takes {0.95,0.1}, {0.95,0.15}, {0.99,0.1} and {0.99,0.15} duples as parameter values. We can say that taken HMCR-PAR duples provide to reach the global solution for increased *n* values and the best duple is {0.99,0.1}.

**Table 4.** *Average execution time of algorithms for several n values with different HMCR-PAR duples*

| *n* | HMCR-PAR | | | | | | Average Execution Time (sc) |
|---|---|---|---|---|---|---|---|
| | *0.9-0.1* | *0.9-0.15* | *0.95-0.1* | *0.95-0.15* | *0.99-0.1* | *0.99-0.15* | |
| 5 | 0.03718 | 0.04175 | 0.04064 | 0.03933 | 0.03541 | 0.04709 | |
| 6 | 0.21561 | 0.24175 | 0.1881 | 0.21562 | 0.18743 | 0.20444 | |
| 7 | 5.54906 | 7.08809 | 4.62956 | 5.44005 | 5.54478 | 4.86112 | |
| 8 | 29.90372 | 43.78826 | 24.13146 | 34.1881 | 17.52175 | 27.11103 | |
| 9 | - | - | 2417.849 | 617.7988 | 1449.19 | 2707.961 | |
| 10 | - | - | 4298.257 | 4607.294 | 1712.876 | 4168.101 | |

To point out effect of parameters' values to the working of the algorithm, we can examine the execution time of the algorithm for different *n* values with HMCR-PAR duples. For instance, we get result for n=5 with both HMCR=0.9 and PAR=0.5 which is the worst execution time and HMCR=0.99 and PAR=0.1 which is the best execution time. Optimization percentage is evaluated using Eq (1). For n=5, we can reduce execution time 38.46% by changing HMCR and PAR value, The most attractive optimization percantage is taken in n=8. HMCR=0.9 and PAR=0.45 are the worst values for n=8 and the best duple is HMCR=0.99 and PAR=0.1. The percentage of the optimization is 95.03%. The other percentages are given in Table 5.

**Table 5.** *Optimization percetage in running time of algorithm*

$$\frac{100 \times |T_{worst} - T_{best}|}{T_{worst}} \quad (1)$$

| n | Optimization Percentage(%) |
|---|---|
| 5 | 38.46 |
| 6 | 60.57 |
| 7 | 83.78 |
| 8 | 95.03 |
| 9 | 77.19 |
| 10 | 62.82 |

## 5. CONCLUSION

Harmony Search (HS) algorithm is a meta-heuristic technique and aims to obtain better solution with decreased number of iterations. HS's parameters, HMS, HMCR, and PAR values have to be predetermined before the algorithm runs. The parameters' values play a critical role in way of searching global or near solution [1][2][3]. It may cause catastrophic results such as lack of solution or extending time of finding solution. So, to draw attention to this point, we adapt HS algorithm to known Bridge and Torch problem. The problem's global solutions are known for several *n* values. To observe execution time of the algorithm, it was run for several *n* values by changing HMCR and PAR values. Firstly, we focus on finding best pair of HMCR and PAR value. To reach global solution in the other *n* values, we detail HMCR-PAR duples more sensitively. We recognize that a HMCR-PAR duple gives the best performance for several *n* values. It means we can say that we find the best HMCR-PAR duple for Bridge and Torch problem. Also, we want to point out to running time of algorithms with different HMCR and PAR values. Experimental results show that if the best values for parameters are not set, there are two possibilities. One possibility is the algorithm cannot find global optimum or near solution in acceptable time. Second possibility is that it reaches global optimal solution but it a long time. As seen in Table 5. Optimization percetage in running time of algorithm5, the global solution can be reached with 95.03% decreased running time by setting suitable HMCR-PAR values.

## 6. REFERENCES

[1] Bouzidi, M., Riffi, M. E., "Adaptation of the Harmony Search Algorithm to Solve the Travelling Salesman Problem" Journal of Theoretical and Applied Information Tehnology, 2014, pp. 154-160.

[2] Geem, Z. W., Kim, J. H., Loganathan, G., "A New Heuristic Optimization Algorithm: Harmony Search" Simultion, 76(2), 2001, pp. 60-68.

[3] Kong, X., Gao, L., Ouyang, H., Li, S., "A simplified binary harmony search algorithm for large scale 0–1 knapsack problems" Expert Systems with Applications, vol. 42 , 2015, pp. 5337–5355.

[4] Backhouse, R., Truong, H., "The capacity-C torch problem" Science of Computer Programming, vol. 102, 2015, pp. 76–107.

[5] Crossing the Bridge at Night (2002) [Online]. Avaliable: https://page.mi.fu-berlin.de/rote/Papers/pdf/Crossing+the+bridge+at+night.pdf

[6] Temur, R., Tanrıverdi, S. C., "Ulaştırmada Talep Tahmin Modellerinde Harmoni Arama Yöntemi Uygulaması" in International Science and Technology Conference (ISTEC), Rome, Italy, 2013, pp. 365-372.

[7] Sachin A. Patil, D. A. Patel, "An Overview: Improved Harmony Search Algorithm and Its Applications in Mechanical Engineering" International Journal of Engineering Science and Innovative Technology (IJESIT), vol. 2, 2013, pp. 433-444.