Harmony-Scatter Search to Solve Travelling Salesman Problem

Ahmed T. Sadiq Al-Obaidi*
Department of Computer Science, University of Technology, Baghdad, Iraq.

Abstract
This paper presents a hybrid metaheuristic algorithm which is Harmony-Scatter Search (HSS). The HSS provides Scatter Search (SS) with random exploration for search space of problem and more of diversity and intensification for promising solutions. The SS and HSS have been tested on Traveling Salesman Problem. A computational experiment with benchmark instances is reported. The results demonstrate that the HSS algorithm produce better performance than original Scatter Search algorithm. The HSS in the value of average fitness is 27.6% comparing with original SS. In other hand the elapsed time of HSS is larger than the original SS by small value. The developed algorithm has been compared with other algorithms for the same problem, and the result was competitive with some algorithm and insufficient with another.
Keywords: Metaheuristic; Scatter Search; Harmony Search; Combinatorial Problems; Traveling Salesman Problem

البحث الايقاعي المنتشر لحل مشكلة البائع المتجول

أحمد طارق صادق*
قسم علوم الحاسوب، الجامعة التكنولوجية، بغداد، العراق.

الخلاصة:
يقدم هذا البحث خوارزمية مهجنة تنقيبية (وصفية) هي خوارزمية البحث الايقاعي المنتشر. توفر هذه الخوارزمية للبحث المنتشر استكشاف عشوائي لمجال بحث المشكلة ومزيداً من التنوع والتكثيف لايجاد مختلف الحلول. تم اختبار الخوارزمية المقترحة لحل مشكلة البائع المتجول. أظهرت النتائج ان خوارزمية البحث الايقاعي المنتشر أعطت نتائج أفضل من الخوارزمية الاصلية للبحث المنتشر وزادت نسبة دالة الكفاءة بنسبة 27.6% عن الخوارزمية الاصلية. ومن جانب آخر فأن وقت التنفيذ للخوارزمية المقترحة كان أكبر بقليل من الخوارزمية الاصلية. وقد تم مقارنة الخوارزمية المقترحة مع خوارزوميات آخرى لنفس المشكلة المعنية وكانت النتيجة بأن خوارزمية البحث الايقاعي المنتشر أفضل من بعض الخوارزميات وعدم أفضليتها على البعض الاخر.

1. INTRODUCTION

There are several heuristic and metaheuristic algorithms have been used to solve a wide range of NP-hard problems. A large number of real-life optimization problems in science, engineering, economics, and business are complex and difficult to solve. They can't be solved in an exact manner within a reasonable amount of time [1]. Real-life optimization problems have two main characteristics, which make them difficult:

*Email: Drahmaed_tark@yahoo.com

they are usually large, and they are not pure, i.e.; they involve a heterogeneous set of side constraints [2]. Metaheuristic techniques are the basic alternative solution for this class of problems. Recently, many researchers have focused their attention on a metaheuristics. A metaheuristic is a set of algorithmic concepts that can be used to define heuristic methods applicable to a wide set of different problems. The use of metaheuristics has significantly increased the ability of finding solutions practically relevant combinatorial optimization problems in a reasonable time [3]. Prominent examples of metaheuristics are Evolutionary Algorithms, Simulated Annealing, Tabu Search, Scatter Search, Variable Neighborhood Search, Memetic Algorithms, Ant Colony Optimization, Cuckoo Search, Harmony Search and others, which successfully solved problems include scheduling, timetabling, network design, transportation and distribution problems, vehicle routing, the traveling salesman problem and others [4].

Harmony Search was inspired by the improvisation of Jazz musicians. Specifically, the process by which the musicians (who may have never played together before) rapidly refine their individual improvisation through variation resulting in an aesthetic harmony. Each musician corresponds to an attribute in a candidate solution from a problem domain, and each instrument’s pitch and range corresponds to the bounds and constraints on the decision variable. The harmony between the musicians is taken as a complete candidate solution at a given time, and the audiences aesthetic appreciation of the harmony represent the problem specific cost function. The musicians seek harmony over time through small variations and improvisations, which results in an improvement against the cost function. The information processing objective of the technique is to use good candidate solutions already discovered to influence the creation of new candidate solutions toward locating the problems optima. This is achieved by stochastically creating candidate solutions in a step-wise manner, where each component is either drawn randomly from a memory of high-quality solutions, adjusted from the memory of high-quality solutions, or assigned randomly within the bounds of the problem. The memory of candidate solutions is initially random, and a greedy acceptance criteria is used to admit new candidate solutions only if they have an improved objective value, replacing an existing member [5].

This paper presents a new improvement of scatter search using harmony search algorithm. Section 2 includes the related works of improved scatter search methods. Section 3presents a background of scatter search algorithm. The basic concept of harmony search introduced algorithm in section 4. Section 5 includes the proposed Harmony-Scatter Search Algorithm. Section 6 presents the experimental results of the proposed algorithm compared with some related scatter algorithms types to solve the Travelling Salesman Problem. The final section 7 presents the conclusions of this paper.

2. RELATED WORKS

There is several literature surveys applied to improve or hybridization of Scatter Search (SS) algorithm. Ahmed T. [6] proposed a new improvement of scatter search algorithm using nature inspired swarm intelligence algorithm which is Cuckoo search. The new hybrid enhanced scatter search uses cuckoo search via Levy flight, the improved SS enhances the value of average fitness in 23.2% comparing with original SS. Ali M. *et al* [7] presented improved SS using Bees Algorithm. The improvement provides SS with random exploration for search space of problem and more of intensification for promising solutions. The experimental results prove that the improved SS algorithm is better than original SS algorithm in reaching to nearest optimal solutions. Juan José *et al* [8] presented development for multiple object visual trackers based on the Scatter Search Particle Filter (SSPF) algorithm. It has been effectively applied to real-time hands and face tracking. Jose A. *et al* [9] presented the SSKm algorithm proposed methodology for global optimization of computationally expensive problems. Saber *et al* [10] presented hybrid genetic Scatter Search algorithm that replaced two steps in Scatter Search (combination and improvement) with two steps in genetic (crossover and mutation). This algorithm leads to increase the efficiency and exploration of the solution process. T. Sari *et al* [11] evaluate Scatter Search and genetic algorithm. Resource constrained project scheduling problem which is an NP-hard problem is solved with two algorithms. They conclude that genetic algorithm outperformed Scatter Search. Tao Zhang *et al* [12] presented development of new Scatter Search approach for the stochastic travel- time vehicle routing problem with simultaneous pick-ups and deliveries by incorporating a new chance-constrained programming method. A generic genetic algorithm approach is also developed and used as a reference for performance comparison. The evaluation shows the performance characteristics and computational results of the SS solutions are superior to the GA solutions. Oscar Ibáñez *et al* [13] parented a new skull-face overlay method based on the Scatter Search algorithm. This approach achieves faster and more robust solutions. The performance compared to the current best performing approach in the field of automatic skull-face overlay. The presented approach has shown an accurate and robust performance when solving the latter six face-skull overlay problem instances. Ying Xu and Rong Qu [14] presented a hybrid Scatter Search meta-heuristic to solve delay-constrained multicast routing problems, this approach intensify the search using Tabu and variable neighborhood search (VNS) then is efficient in solving the problem in comparison with other algorithms which is descent the search. Jue Wang *et al* [15] proposed novel approach to feature selection based on rough set using Scatter Search to improve cash flow and credit collections. The conditional entropy is regarded as the heuristic to search the optimal solutions. The experimental result has a superior performance in saving the computational costs and improving classification accuracy compared with the base classification methods.

Regarding the previous works discussed above, this paper presents new improvement to the Scatter Search algorithm using Harmony Search which is one of the several physical inspired methods that was proposed to solve Combinatorial Optimization problems. The contribution is that the improved Scatter Search with Harmony Search reaching to the nearest optimal solutions than original Scatter Search.

The Scatter Search algorithm is proven successful in travelling salesman problem [16]. The Traveling Salesman Problem (TSP) is a classical NP-hard combinatorial problem. Let given a graph G = (N, E), where N = {1, ..., n} is the set of nodes and E = {1, ..., m} is the set of edges of G, which represent the costs. The c_{ij}, associated with each edge linking vertices, i and j. The problem consists in finding the minimal total length Hamiltonian cycle of G. The length is calculated by the summation of the costs of the edges in a cycle. If for all pairs of nodes {i,j}, the cost's c_{ij} and c_{ji} are equal, then the problem is said to be symmetric, otherwise it is said to be asymmetric. It represents an important test ground for many evolution algorithms [1].

3. Scatter Search Technique

Scatter Search (SS) algorithm is one of the population-based Metaheuristics. It works on a population of solutions, which are stored as a set of solutions called the Reference Set. The solutions to this set are combined in order to obtain new ones, trying to generate each time better solutions. According to quality and diversity criteria, Figure 1 illustrates the basic SS algorithm [1, 17].

The design of a SS algorithm is generally based on the following five steps [17, 18]:

- *A Diversification Generation Method* to generate a population (Pop) of diverse trial solutions within the search space.
- *An Improvement Method* to transform a trial solution into one or more enhanced trial solutions.
- *A Reference Set Update Method* to build and maintain a Reference Set. The objective is to ensure diversity while keeping high-quality solutions. For instance, one can select $RefSet_1$ solutions with the best objective function and then adding $RefSet_2$ solutions with the optimal diversity solutions ($RefSet = RefSet_1 + RefSet_2$).
- *A Subset Generation Method* to operate on the reference set, to produce several subsets of its solutions as a basis for creating combined solutions.
- *A Solution Combination Method* to transform a given subset of solutions produced by the Subset Generation Method into one or more combined solution vectors.

After generating the new solutions which are generated from Solution Combination Method, these solutions will be improved by Improvement Method, and this solution will become a member of the reference set if one of the following rules is satisfied [17]:

1) The new solution has a better objective function value than the solution with the worst objective value in $RefSet_1$.
2) The new solution has a better diversity value than the solution with the worst diversity value in RefSet2.

The search is continued while RefSet is changed. If no change in RefSet, the algorithm will check if the number of iteration (*itr*) reach the max iteration (*MaxItr*) that detected by the user, then the algorithm will display the good solution(s) reached, else, the new population will be generated, and RefSet1 will be added to the start of this population.

Scatter Search Algorithm
Input: Population of the problem.
Output: The best of solutions
Initialize the population Pop using a Diversification Generation Method.
Apply the Improvement Method to the population.
Reference Set Update Method (Good solutions for $RefSet_1$ and Diversity solutions for $RefSet_2$).
While (*itr* < *MaxItr*) do
    While (Reference set is changed) do
        Subset Generation Method
            While (subset-counter < > 0) do
                Solution Combination Method.
                Improvement Method.
                Reference Set Update Method;
            End while
    End while

Figure 1- Basic Scatter Search Algorithm

4. HARMONY SEARCH ALGORITHM

Harmony search algorithm is aspect of relationship between music and optimization. In the Harmony Search Algorithm, the two interesting fields of studies, music and computer science are combined. The behavior of musicians when they are creating their music has a resemblance in the optimization process: Each musical instrument represents a decision variable; a musical note corresponds to the value of each variable; and the harmony corresponds to a solution [19, 20].

Jazz musicians, when they are composing their music, either play notes randomly, play notes based on experiences, or adjusting the pitch in order to find a fantastic harmony. In order to find an optimal solution, variables in the harmony search algorithm are assigned with values that are either random or are taken from previously-memorized good values [20]. To better understand the harmony search model it is important to study first the inspiration that led to the creation of the said algorithm. It is believed that when musicians create their music, they use three techniques to achieve harmony. These are (1) playing using randomly selected notes, (2) playing music from their experiences, and (3) adjusting the tone to better harmonize the music [19].

Geem et al [19, 20], noticed the similarity of this behavior in achieving the optimal solution to a problem. Thus in 2001, they 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 Harmony Search Algorithm [21, 22, 23].

Just like when musicians play a random pitch within the instrument range, in random selection, random values are picked from the range of possible values of a certain variable. Also, similar to a musician that plays any preferred pitch from his previous composition or memory,in memory consideration, values are chosen from the vectors stored in harmony memory [21, 22].

Once a pitch is obtained from memory, a musician can further adjust the pitch to the neighboring pitches to obtain a better harmony. In pitch adjustment, the value is adjusted with a certain probability. This value may or may not move to neighboring values with a definite probability [21, 22], figure (2) shows the harmony search algorithm code sequence [21].

The overall process of the harmony search algorithm can be illustrated in Figure 3, where it can be generalized into three main steps [21]:

1. **Initialization:** it is where parameters are defined and the harmony memory is being filled with random harmonies or candidate solutions.
2. **Improvisation:** in this process, a new solution is created using the three methods of the harmony search algorithm. This step is repeated until a termination condition is met.
3. **Selection:** after improvisation, the best harmony is selected in the harmony memory to represent the solution to the problem.

In order for the Harmony Search Algorithm to start, certain factors must be considered first. Some parameters must be defined first before the optimization process begins.

1. Number of decision variables: each harmony is composed of several decision variables.
2. Number of cycles of iteration: one of the termination conditions of the optimization process.
3. Harmony Memory Size: refers to the number of solutions that will be stored in the harmony memory.
4. Harmony Memory Consideration Rate (raccept): 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.

5. Pitch Adjustment Rate (rpa): the probability that the decision variable picked from the harmony memory be altered by some certain amount.

The capability of Harmony Search Algorithm in solving problems has been proven effective in various studies such as the Traveling Salesman Problem, Sudoku Puzzle and the Four Color Map Problem [24, 25, 26].

Harmony Search Algorithm
Input: Population of the problem.
Output: The best of solutions
Begin
Define objective function *f(x), x = (x1, x2… xd);*
Define harmony memory accepting rate *(raccept);*
Define pitch adjusting rate *(rpa)* and other parameters;
Generate Harmony Memory with random harmonies;
While *(t < max number of cycles)*
    While *(i < number of variables)*
        If *(rand<raccept)* choose a value for var i
            If *(rand<rpa)* adjust the value
        Else
            choose a random value;
        End If
    End While
    Accept the new memory if better;
End While
Pick the best solution;
End

Figure 2- Basic Harmony Search Algorithm

5. THE PROPOSED HARMONY-SCATTER SEARCH

The improvement to SS algorithm was accomplished by using nature inspired physical algorithm, which is Harmony Search Algorithm. Harmony search algorithm has proven its ability in solving some combinatorial problems and finding the nearest global optimum solution in reasonable time and good performance. Because the SS algorithm is composed of several steps, there will be several places to improve the SS algorithm. However, by the applied experiments, Subset Generation Method, Improvement Method and Reference Set Update Method are the most effective steps in improving the SS algorithm.

The adjustment of pitch to the neighboring pitches to obtain a better harmony is the power of Harmony search algorithm. The proposed improvement uses the Harmony search algorithm as an improvement method in the Scatter search algorithm. The Improvement Method in SS algorithm is applied on all populations rather than to each new solution produced from Combination Method, so this will take a large amount of time, this will affect the SS algorithm as one of the metaheuristic algorithms that the main goal of it in solving the problems is to find the optimal solution in reasonable time.

However, when trying to improve the SS algorithm in Reference Set Update Method in SS algorithm, the results were good and in reasonable time. The steps of Harmony search algorithm will take its solutions from steps in SS, which is Reference Set Update Method and explore more of solutions and retrieve the best solutions reached to complete SS steps. See Figure 3, which is show the improved SS algorithm using Harmony search.

In Reference Set Update Method, $RefSet_1$ of b_1 of the best solutions and $RefSet_2$ of b_2 of diversity of solutions will be chosen. $RefSet_1$ will enter to the new steps that added from Harmony search to Scatter search. The new steps provide a more diversity to the $RefSet_1$ which is benefit from the neighborhood search in the cuckoo search steps. Also the updated RefSet will contain more enhanced solutions than the old because the substitution operator forms the Harmony solutions.

```
Harmony Scatter Search Algorithm
Input: Population of the problem.
Output: The best of solutions
Initialize the population using Diversification
  Generation Method;
Apply the Harmony search algorithm as an
  Improvement Method to the population;
Reference Set Update Method (Good
  solutions for RefSet1 and Diversity solutions
  for RefSet2);
    While (itr < MaxItr) do
        While (Reference set is changed) do
          Subset Generation Method;
          While (subset-counter < > 0) do
              Solution Combination Method;
              Apply the Harmony search
              algorithm as an Improvement
              Method to the population;
              Reference Set Update Method;
          End while
    End while
    End while
Return the best solutions;
```

Figure 3- Proposed Harmony Scatter Search (HSS) Algorithm

6. EXPERIMENTAL RESULTS

TSP is one of the main combinatorial problems that used as test ground for most search techniques. This paper applies original SS and HSS algorithms to symmetric TSP as a tool to measure the performance of the proposed HSS.

SS and its improvement algorithms were implemented in Microsoft Visual C# 2005 Express Edition and run on a computer whose processor is Intel Core2 Duo T657064 2.0 GHz, with 2 GB main memory, 200 GB hard disk. The algorithms were applied to symmetric instances of the benchmark TSPLIB [27] with sizes ranging over from 26 to 1379.

The stop criteria are chosen as follows:

1. If no change in Reference Set.
2. To reach a maximum number of iterations = 35.

The following parameters are chosen:

- Initial population *P* =100,
- The size of | $RefSet_1$| =b_1=10, the size of |$RefSet_2$| =b_2=10 and the size of reference set |RefSet| = |$RefSet_1$|+ |$RefSet_2$|=20.
- Pitch Adjustment Rate = 0.45.
- Harmony Memory Accepting Rate = 0.85.

A first experiment compared SS with HSS. Twenty five independent runs of each algorithm were performed. The results are shown in Table 1.

TABLE 1-COMPARISON OF SS AND PROPOSED HSS FOR AVERAGE OPTIMALITY

Instances	*Averages Of SS*	*Average of Proposed HSS*
Fri26	1600	**1201**
Dantzig42	1990	**1523**
Att48	100995	**83009**
Eil51	1133	**890**
Eil101	2616	**2011**
KroA100	127667	**108953**
KroB100	124799	**103121**
KroC100	126565	**104784**
KroD100	123197	**100097**
KroE100	129005	**102312**
KroB200	269085	**231436**
Lin105	91707	**70081**
Lin318	513090	**389012**
Pr76	432145	**236790**
Pr124	537678	**321762**
Pr299	646297	**490234**
Pr439	1692199	**1207120**
Pr1002	6050966	**5189099**
Nrw1379	1344099	**989012**
Berlin52	20811	**11007**
Bier127	520107	**340190**
A280	29046	**13479**

To see clearly the difference between SS and HSS see Figure 4.

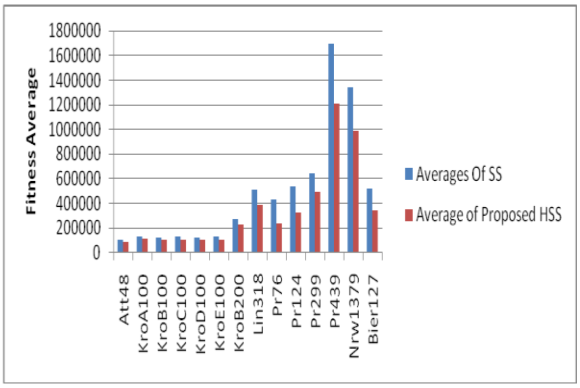

Figure 4- Difference between SS and proposed HSS for Some Instances

Computational experiments illustrate the differences between SS algorithm, and the HSS algorithm. The Nearest Optimal Solution (*NOPT*) for HSS has been indicated in Table 2 with bold font. The difference is increased whenever the size of instance is increased.

Averages of fitness *f(x)* required to reach the nearest optimal solutions that output from original SS, and its improvement have been computed. In all instances, the Improved SS obtained better results than original SS with little difference in time, averages of elapsed time and difference of the ratio between the averages of time required to reach optimal solution in HSS and SS is ≅0.45 second.

The ratio of difference was computed as follows (Averages of Elapsed Time (sec) for HSS – Averages of Elapsed Time (sec) for SS).

TABLE 2- COMPARISON OF SS AND PROPOSED HSS FOR ***NOPT***

Instances	***NOPT in SS***	***NOPT in Proposed HSS***
Fri26	1379	**1075**
Dantzig42	1810+	**1134**
Att48	86890	**80912**
Eil51	1003	**831**
Eil101	2423	**1092**
KroA100	113253	**100628**
KroB100	111239	**100004**
KroC100	113539	**101003**
KroD100	113245	**101081**
KroE100	120552	**95812**
KroB200	251029	**228725**
Lin105	82838	**71006**
Lin318	494126	**438715**
Pr76	401947	**319891**
Pr124	500592	**401170**
Pr299	618178	**500172**
Pr439	1611932	**1453981**
Pr1002	5889830	**5061903**
Nrw1379	1301255	**1129375**
Berlin52	17931	**13917**
Bier127	501161	**416625**
A280	27789	**20927**

Table 3 shows the averages of elapsed time for SS and HSS algorithms for the instances in Table- I.

TABLE 3- AVERAGE OF ELAPSED TIME FOR SS AND PROPOSED HSS

Instances	***Average of elapsed time for SS (Sec)***	***Average elapsed time for Proposed HSS (Sec)***
Fri26	0.48	0.58
Dantzig42	0.63	0.77
Att48	0.74	0.88
Eil51	0.61	0.75
Eil101	1.11	1.31
KroA100	1.07	1.32
KroB100	1.08	1.26
KroC100	1.07	1.33
KroD100	1.09	1.36
KroE100	1.08	1.45
KroB200	2.29	2.54
Lin105	1.19	1.49
Lin318	3.91	4.30
Pr76	0.88	1.78
Pr124	1.31	1.58
Pr299	3.64	4.33
Pr439	5.51	6.04
Pr1002	15.87	17.05
Nrw1379	23.56	25.07
Berlin52	0.64	0.82
Bier127	1.55	1.81
A280	3.38	4.54

The results of HSS will be the best because the added steps from Harmony Search algorithm in the improvement steps of SS provided a good diversity & intensification for the new and ratio of getting *NOPT* solutions will be increased. The ratio of getting *NOPT* solution will be increased respectively with increasing the size of $RefSet_1$.

In the second computational experiment we use the same parameters in first computational experiments except for the $|RefSet_1| = b_1 = 20$ where $|RefSet| = |RefSet_1| + |RefSet_2| = 30$.

By compute the averages of fitness and elapsed time with ten runs for the same instances in Table I. The results of the second experiments are illustrated in Table 4. When we increase the value of $RefSet_1$ to 20, we found the results for SS and HSS are better than the results in Table I.

TABLE 4- COMPARISON OF SS AND PROPOSED HSS FOR AVERAGE OPTIMALITY WITH $REFSET_1$=20

Instances	***Averages Of fitness for SS***	***Average of fitness Proposed HSS***
Fri26	1461	**980**
Dantzig42	1751	**1016**
Att48	92156	**72976**
Eil51	1005	**891**
Eil101	2312	**1459**
KroA100	118654	**94091**
KroB100	115987	**96712**
KroC100	114982	**97012**
KroD100	111707	**93913**
KroE100	117233	**93701**
KroB200	251087	**110837**
Lin105	84590	**70379**
Lin318	475691	**338012**
Pr76	379328	**250019**
Pr124	500807	**390157**
Pr299	601011	**483910**
Pr439	1562181	**1109375**
Pr1002	5761184	**4291852**
Nrw1379	1104810	**880141**
Berlin52	17981	**10744**
Bier127	478521	**294041**
A280	27234	**15024**

To see clearly the difference between SS and HSS with $RefSet_1$=20 see Figure 5.

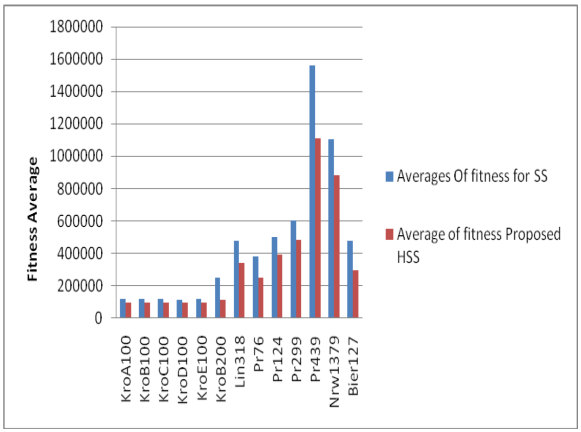

Figure 5- Difference between SS and proposed HSS for Some Instances with $RefSet_1$=20

In spite of the results are better with RefSet1=20, there is a still difference in time. This difference is caused by the new size of $RefSet_1$which increase the exploration and intensification for new solutions. Table 5 shows the NOPT results of SS, HSS with $RefSet_1$=20. Table VI shows the elapsed time for SS and HSS with $RefSet_1$=20. The increased time where $RefSet_1$=20 is ≅1.2 second for HSS.

TABLE 5- COMPARISON OF SS AND THE PROPOSED HSS FOR ***NOPT*** WITH $REFSET_1$=20

Instances	*NOPT in SS*	*NOPT in Proposed HSS*
Fri26	1364	**1016**
Dantzig42	1689	**1093**
Att48	84041	**80070**
Eil51	909	**797**
Eil101	2091	**1012**
KroA100	112772	**99825**
KroB100	114654	**98097**
KroC100	113124	**98023**
KroD100	110012	**98145**
KroE100	114789	**94971**
KroB200	231314	**227428**
Lin105	82139	**70714**
Lin318	467549	**429896**
Pr76	373254	**317812**
Pr124	498982	**400093**
Pr299	608723	**49791**
Pr439	1631578	**1449891**
Pr1002	5902741	**5047231**
Nrw1379	1298711	**1119986**
Berlin52	17172	**13172**
Bier127	480941	**414757**
A280	26576	**20435**

In the second experiments, for instances with large size such as Pr439, Pr1002 and Nrw1379 we noticed that the average of elapsed time with HSS is larger than original SS with approximately 3.4 second in average. This case can lead us to the fact that HSS with large instances can reach to the best *NOPT* solution with a very reasonable time than original SS.

In general, comparing the time with the *NOPT* solutions isn't important for those who are looking for *NOPT* solutions, and they aren't cared about the time.

In third experiment, the comparison of the *NOPTs* of HSS in Table 6 with results obtained by other algorithms. By compute the average deviation for the output solutions $SD = 100(NOPT - opt) / opt$, where *NOPT* is the Nearest Optimal Solution output from HSS and the *opt* is the optimal solution taken from TSPLIB [27].

TABLE 6- AVERAGE OF ELAPSED TIME FOR SS AND PROPOSED HSS WITH $REFSET_1$=20

Instances	*Average of elapsed time for SS (Sec)*	*Average elapsed time for Proposed HSS (Sec)*
Fri26	1.27	1.48
Dantzig42	1.67	1.99
Att48	1.96	2.28
Eil51	1.61	1.95
Eil101	2.93	3.39
KroA100	2.83	3.56
KroB100	2.86	3.38
KroC100	2.83	3.61
KroD100	2.89	3.69
KroE100	2.86	3.89
KroB200	6.06	6.87
Lin105	3.15	3.97
Lin318	10.36	11.26
Pr76	2.33	4.77
Pr124	3.47	4.21
Pr299	9.65	11.73
Pr439	14.60	16.07
Pr1002	42.05	45.02
Nrw1379	62.43	68.21
Berlin52	1.70	2.21
Bier127	4.11	4.91
A280	8.96	10.18

TABLE 7- RESULTS OF IMPROVED SS ARE BETTER THAN SOME ALGORITHMS

Instances	**Optimal in TSPLIB in [27]**	***SD for NOPT* for Proposed HSS**	***SD for NOPT* forSS-CS in[6]**	***SD* for optimal solutions in[28]**	***SD* for optimal solutions in[29]**	***SD* for optimal solutions in[30]**	***SD* for optimal solutions in[31]**
Fri26	937	8.43	28.81	-	34.47	0	0
Dantzig42	699	56.37	70.95	-	119.45	0	0
Att48	10628	653.39	670.59	-	573.96	0	0
Eil51	426	87.09	101.40	-	125.35	0	0
Eil101	629	60.89	89.03	-	259.61	0.107	0
KroA100	21282	369.06	377.87	808.51	378.78	0	0
KroB100	22141	343.06	356.61	-	347.35	0.036	0
KroC100	20749	372.42	391.98	854.24	389.84	0	0
KroD100	21294	360.9	379.02	-	350.37	0.019	0
KroE100	22068	330.36	339.99	-	345.15	0.001	0
KroB200	29437	672.59	681.89	828.21	662.59	0.509	0
Lin105	14379	391.79	414.61	835.15	393.62	0	0
Lin318	41345	939.78	968.53	880.41	962.99	0.769	0.29
Pr76	108159	193.84	207.78	744.56	216.44	0	0
Pr124	59030	577.78	582.54	801.44	599.80	0	0
Pr299	48191	3.32	944.02	894.60	991.79	0.066	0.01
Pr439	107217	1252.3	1271.68	882.16	1209.28	0.572	0.18
Pr1002	259045	1848.4	1857.53	927.95	1910.50	-	-
Nrw1379	56638	1877.45	1928.84	891.17	2105.92	-	-
Berlin52	7542	74.65	96.38	-	127.45	0	0
Bier127	118282	250.65	253.31	724.70	259.06	0.064	0
A280	2579	692.36	717.72	872.48	900.34	0.305	0

Table 7 shows how the results of HSS are better than some results such as in [6], [28] and [29] in the most cases. Also the same table shows how the HSS results are far from other results of other algorithms such as [30] and [31].

7. CONCLUSIONS

Harmony-Scatter Search presented in this paper as a metaheuristic algorithm. The improvement provides SS with random exploration for search space of problem and more of diversity and intensification for promising solutions based on the Harmony search algorithm. From experimental results, the average of fitness value for HSS algorithm is better than original SS algorithm, the improvement in the value of average fitness is 27.6% comparing with original SS. From experimental results, the HSS algorithms are better than original SS algorithm in reaching to nearest optimal solutions. The elapsed time for the HSS is larger than the elapsed time for original SS in a reasonable value 1.2 in average. The difference in elapsed time to reach Nearest Optimal Solution isn't a problem for those whose look for optimal solutions, and they aren't cared about the time. In general, the ratio of difference isn't very large. Also, the optimal solution of the improved SS is better than some algorithms but is far away from some others.

For future work, the HSS algorithm for TSP give an enhanced results comparing with the original SS but not good results comparing with most dependent algorithms, so it is reasonable to improve the SS & other HSS with a mix techniques based on more than one improved steps to obtain the good results.

References

1. El-Ghazali, Talbi. **2009**. "*Metaheuristics from Design to Implementation*", John Wiley & Sons.
2. Glover, F. and Kochenberger, Gary A. **2003**. "*Handbook of Metaheuristics*," Kluwer Academic.
3. Marino, Dorigo and Thomas, Stützle. **2004**. "*Ant Colony Optimization*". MIT Press.
4. Eberhart, R., Y. Shi, and Kennedy, J. **2001**. "*Swarm Intelligence*" Morgan Kaufmann, San Francisco.
5. Brownlee, Jason. **2011**. "*Clever Algorithms Nature-Inspired Programming Recipes*", First Edition, LuLu Publishing.
6. Sadiq, Ahmed T. **2013**. "Improved Scatter Search Using Cuckoo Search", *IJARAI*, 2(2), pp:.
7. Sagheer, A. M.; S., Ahmed T. and Ibrahim, M. S. **2012**. "Improvement of Scatter Search Using Bees Algorithm", Proceedings of the 6th International Conference on Signal Processing and Communication Systems ICSPCS2012, Gold Coast, Australia, 12 - 14 December.

8. Pantrigo, Juan José.; Montemayor, Antonio S. and Cabido, Raúl. **2005**. " *Scatter Search Particle Filter for 2D Real-Time Hands and Face Tracking*", Springer.

9. Egea, Jose A.; Vazquez, Emmanuel.; Banga, Julio R. and Marti, Rafael. **2009**. " Improved scatter search for the global optimization of computationally expensive dynamic models", *Journal of Global Optimization*, 43, pp:175–190.

10. El-Sayed, S. M.; Abd EL-Wahed, Waiel F. and Ismail, Nabil A. **2008**. "A Hybrid Genetic Scatter Search Algorithm for Solving Optimization Problems", Faculty of Computers and Informatics, Operations Research Department, Egypt.

11. Sari, T.; Cakir, V.; Kilic, S. and Ece, E. **2011**. "Evaluation of Scatter Search and Genetic Algorithm at Resource Constrained Project Scheduling Problems", INES 2011 : 15th International Conference on Intelligent Engineering Systems, June 23–25, Poprad, Slovakia.

12. Zhang, Tao; Chaovalitwongse, **2012**. W. A. and Zhang, Yuejie. "Scatter search fort hestochastic travel-time vehicle routing problemwith simultaneous pick-ups and deliveries", Elsevier Ltd., Expert Systems with Applications 39, pp:6123–6128.

13. Ibáñez, Oscar.; Cordón, Oscar.; Damas, Sergio. and Santamaría, José. **2012**. "An advanced scatter search design for skull-face overlay in craniofacial superimposition", Elsevier Ltd., Expert Systems with Applications 39. pp:1459–1473.

14. Xu, Ying.; and Qu, Rong. **2012**. "A hybrid scatter search meta-heuristic for delay-constrained, multicast routing problems", Springer Science+Business Media, LLC 2010, Appl Intell 36, pp:229–241.

15. Wang, Jue.; Hedar, Abdel-Rahman.; Wang, Shouyang. and Ma, Jian. **2012**. "Rough set and scatter search metaheuristic based feature selection for credit scoring", Elsevier Ltd., Expert Systems with Applications 39. pp:6123–6128.

16. Devasena, M. S. Geetha. and Valarmathi, M. L. **2012**. "Meta Heuristic Search Technique for Dynamic Test Case Generation, *International Journal of Computer Applications* (0975 – 8887) 39(12), pp:.

17. Laguna, M. and R. Martí, **2003**. "*Scatter Search: Methodology and Implementations in C*" Kluwer Academic Press.

18. Glover, F. and Laguna, Manuel. **2000**. "Fundamentals of Scatter Search and Path Relinking," *Control and Cybernetics*, 29(3), pp:653-684.

19. Geem, Z. W.; Kim, J. H. and Loganathan, G. V. **2001**. "New Heuristic Optimization Algorithm: Harmony Search Simulation".

20. Geem, Z. W. **2009**. "Music-Inspired harmony Search Algorithm -Theory and Applications", *Studies in Computational Intelligence*, 191, pp:1-12

21. Romero, V. M.; Tomes, L. L. and Yusiong, J. P. T. **2011**. "Tetris Agent Optimization Using Harmony Search Algorithm", *International Journal of Computer Science* Issues, 8(1). pp:22 – 31.

22. Papadimitriou, C. H.; Dasgupta, S. and Vazirani, U. V. **2006**. "NP-complete Problems Algorithms", pp:257-258.

23. Holland, J. **1992**. "*Adaptation in Natural and Artificial Systems*", MIT Press, Cambridge, MA.

24. Appel, K. and Haken W. **1977**. "Every planar map is four colorable", Part I. *Discharging, Illinois Journal of Math*. 21. pp:429-490.

25. Appel, K.; Haken, W. and Koch, J. **1977**. "Every planar map is four colorable", Part II. *Reducibility, Illinois Journal of Math*. 21. pp:491-567

26. Horca, Romie B. and Yusiong, John Paul T. **2012**. "Using Harmony Search Algorithm to Solve the N-Region Four Color Map Problem", *Journal of Applied Computer Science & Mathematic*, 6(12).

27. Reinelt, G.: TSPLIB, **1995**. Available: http:// www.iwr.uni heidelberg.de / iwr / comopt/ software/TSPLIB95/.

28. D.T., Pham.; A., Ghanbarzadeh.; E., Koç.; S., Otri.; S., Rahim. and Zaidi, M. **2006**. "The Bees Algorithm - A Novel Tool for Complex Optimisation Problems," Proceedings of IPROMS Conference, pp:454-461.

29. Liu, S. B. Ng, K. M. and Ong, H. L. **2007**. "A New Heuristic Algorithm for the Classical Symmetric Traveling Salesman Problem", World Academy of Science, Engineering and Technology, pp:269-270.

30. Gottlieb, Jens. and Raidl, Günther R. **2006**. "Evolutionary Computation in Combinatorial Optimization," Lecture Notes in Computer Science 3906, Springer-Verlag, pp:44-46.

31. Gutin, G., Punnen, A. P. (Ed.), **2002**. "*Traveling Salesman Problem and Its Variations*," Kluwer Academic Publishers.