Improved Scatter Search Using Cuckoo Search

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

***Abstract*— The Scatter Search (SS) is a deterministic strategy that has been applied successfully to some combinatorial and continuous optimization problems. Cuckoo Search (CS) is heuristic search algorithm which is inspired by the reproduction strategy of cuckoos. This paper presents enhanced scatter search algorithm using CS algorithm. The improvement provides Scatter Search with random exploration for search space of problem and more of diversity and intensification for promising solutions. The original and improved Scatter Search has been tested on Traveling Salesman Problem. A computational experiment with benchmark instances is reported. The results demonstrate that the improved Scatter Search algorithms produce better performance than original Scatter Search algorithm. The improvement in the value of average fitness is 23.2% comparing with original SS. 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-component; Metaheuristic; Scatter Search; Cuckoo Search; Combinatorial Problems; Traveling Salesman Problem

I. 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: 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, and others. Which successfully solved problems include scheduling, timetabling, network design, transportation and distribution problems, vehicle routing, the traveling salesman problem and others [4].

II. BACKGROUND

There is several literature surveys applied to improve or hybridization of Scatter Search algorithm. Ali M. *et al* [5] 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* [6] 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* [7] presented the SSKm algorithm proposed methodology for global optimization of computationally expensive problems. Saber *et al* [8] 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* [9] 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* [10] 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* [11] 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 [12] 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 then is efficient in solving the problem in comparison with other algorithms which is descent the search. Jue Wang *et al* [13] 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 CS which is one of the several swarm intelligence methods that was proposed to solve Combinatorial Optimization problems.

The contribution is that the improved Scatter Search with CS reaching to the nearest optimal solutions than original Scatter Search.

The Scatter Search algorithm is proven successful in travelling salesman problem [14]. 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].

The rest of the paper is organized as follows. Scatter Search Technique is described in Section 3. Section 5 presents brief description for CS Algorithm. The first enhanced SS is proposed in Section 6. Section 7 includes the second enhanced SS. In section 8. The experimental results are presented. Finally, some concluding remarks are presented in Section 9.

III. 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, Fig. 1 illustrates the basic SS algorithm [1, 15].

The design of a SS algorithm is generally based on the following five steps [15, 16]:

- *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 [15]:

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
End while
Return the best of solutions

Fig. 1. Basic Scatter Search Algorithm

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.

IV. CUCKOO SEARCH ALGORITHM

CS is a heuristic search algorithm which has been proposed recently by Yang and Deb [17]. The algorithm is inspired by the reproduction strategy of cuckoos. At the most basic level, cuckoos lay their eggs in the nests of other host birds, which may be of different species. The host bird may discover that the eggs are not its own and either destroy the egg or abandon the nest all together. This has resulted in the evolution of cuckoo eggs which mimic the eggs of local host birds. To apply this as an optimization tool, Yang and Deb used three ideal rules [17, 18]:

1) Each cuckoo lays one egg, which represents a set of solution co-ordinates, at a time and dumps it in a random nest;

2) A fraction of the nests containing the best eggs, or solutions, will carry over to the next generation;

3) The number of nests is fixed and there is a probability that a host can discover an alien egg. If this happens, the host can either discard the egg or the nest and this result in building a new nest in a new location. Based on these three rules, the basic steps of the Cuckoo Search (CS) can be summarized as the pseudo code shown as in Fig. 2.

Cuckoo Search via Levy Flight Algorithm
Input: Population of the problem;
Output: The best of solutions;
Objective function f(x), $x = (x_1, x_2, ...x_d)^T$
Generate initial population of n host nests x_i $(i = 1, 2, ..., n)$
While (t <Max Generation) or (stop criterion)
Get a cuckoo randomly by Levy flight
Evaluate its quality/fitness F_i
Choose a nest among n(say,j)randomly
If ($F_i > F_j$) replace j by the new solution;
A fraction(pa) of worse nests are abandoned and new ones are built;
Keep the best solutions (or nests with quality solutions);
Rank the solutions and find the current best;
Pass the current best solutions to the next generation;
End While

Fig. 2. Basic Cuckoo Search Algorithm

When generating new solution $x^{(t+1)}$ for, say cuckoo *i*, a

Levy flight is performed

$$x^{(t+1)}_i = x(t)_i + \alpha \oplus Levy(\beta) \quad \ldots\ldots. \quad (1)$$

where $\alpha > 0$ is the step size which should be related to the scales of the problem of interests. In most cases, we can use $\alpha = 1$. The product $\oplus$ means entry-wise walk while multiplications. Levy flights essentially provide a random walk while their random steps are drawn from a Levy Distribution for large steps

$$Levy \sim u = t^{-1-\beta} \qquad (0 < \beta \leq 2) \quad \ldots\ldots\ldots \quad (2)$$

this has an infinite variance with an infinite mean. Here the consecutive jumps/steps of a cuckoo essentially form a random walk process which obeys a power-law step-length distribution with a heavy tail. In addition, a fraction *pa* of the worst nests can be abandoned so that new nests can be built at new locations by random walks and mixing. The mixing of the eggs/solutions can be performed by random permutation according to the similarity/difference to the host eggs.

V. The Proposed Scatter Cuckoo Search

The improvement to SS algorithm was accomplished by using nature inspired swarm intelligent algorithm, which is Cuckoo Search. Cuckoo 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.

When we try to improve the SS algorithm, the time is the big problem that is found in the Improvement Method. Where the Improvement Method is applying 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 CS 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 Fig. 3, which is show the improved SS algorithm using CS.

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 CS to SS. 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 cuckoo solutions.

Improved Scatter Cuckoo Search Algorithm
Input: Population of the problem.
Output: The best of solutions
Initialize the population Pop using 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
Get a cuckoo randomly by Levy flight (from RefSet)
Evaluate its quality/fitness F_i
Choose a nest among n (say j) randomly
If ($F_i > F_j$) replace j by the new solution;
A fraction(pa) of worse nests are abandoned and new ones are built;
Keep the best solutions (or nests with quality solutions to substitute the RefSet);
Subset Generation Method
While (subset-counter < > 0) do
Solution Combination Method.
Improvement Method.
Reference Set Update Method;
End while
End while
End while
Output: The best of solutions

Fig. 3. Improved Scatter Search Algorithm Using Cuckoo Search

VI. Experimental Results

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

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 [20] 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 = 20.*

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.
- The fraction (pa) of CS is 0.25.

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

TABLE I. Comparison of SS and Proposed SS-CS for Average Optimality

Instances	*Averages Of SS*	*Average of Proposed SS-CS*
Fri26	1600	**1205**
Dantzig42	1990	**1597**
Att48	100995	**83544**
Eil51	1133	**907**
Eil101	2616	**2133**
KroA100	127667	**109542**
KroB100	124799	**104989**
KroC100	126565	**106912**
KroD100	123197	**101391**
KroE100	129005	**109802**
KroB200	269085	**240008**
Lin105	91707	**72879**
Lin318	513090	**401003**
Pr76	432145	**289019**
Pr124	537678	**359097**
Pr299	646297	**505992**
Pr439	1692199	**1041839**
Pr1002	6050966	**4700199**
Nrw1379	1344099	**1001899**
Berlin52	20811	**14768**
Bier127	520107	**371892**
A280	29046	**18901**

To see clearly the difference between SS and its improvement see Fig. 4.

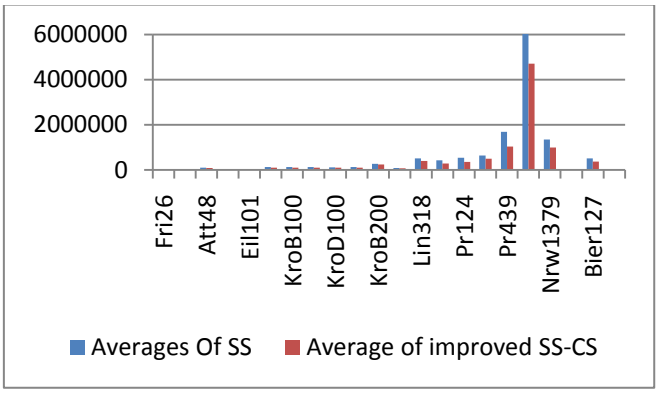

Fig. 4. Difference between SS and proposed SS-CS

Computational experiments illustrate the differences between SS algorithm, and the improved SS algorithm. The Nearest Optimal Solution (*NOPT*) for improved SS has been indicated in Table II 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 2 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 improved SS and SS is ≅0.33 second.

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

TABLE II. COMPARISON OF SS AND PROPOSED SS-CS FOR *NOPT*

Instances	*NOPT in SS*	*NOPT in Proposed SS-CS*
Fri26	1379	**1207**
Dantzig42	1810	**1195**
Att48	86890	**81899**
Eil51	1003	**858**
Eil101	2423	**1189**
KroA100	113253	**101702**
KroB100	111239	**101099**
KroC100	113539	**102081**
KroD100	113245	**102004**
KroE100	120552	**97099**
KroB200	251029	**230165**
Lin105	82838	**73997**
Lin318	494126	**441786**
Pr76	401947	**332900**
Pr124	500592	**402909**
Pr299	618178	**503128**
Pr439	1611932	**1470674**
Pr1002	5889830	**5070901**
Nrw1379	1301255	**1149099**
Berlin52	17931	**14811**
Bier127	501161	**417903**
A280	27789	**21089**

Table III shows the averages of elapsed time for SS and Improved SS algorithms for the instances in Table I.

TABLE III. AVERAGE OF ELAPSED TIME FOR SS AND PROPOSED SS-CS

Instances	*Average of elapsed time for SS (Sec)*	*Average elapsed time for Proposed SS-CS (Sec)*
Fri26	0.48	0.51
Dantzig42	0.63	0.70
Att48	0.74	0.80
Eil51	0.61	0.69
Eil101	1.11	1.20
KroA100	1.07	1.27
KroB100	1.08	1.21
KroC100	1.07	1.29
KroD100	1.09	1.31
KroE100	1.08	1.39
KroB200	2.29	2.47
Lin105	1.19	1.41
Lin318	3.91	4.21
Pr76	0.88	1.72
Pr124	1.31	1.51
Pr299	3.64	4.29
Pr439	5.51	5.91
Pr1002	15.87	16.91
Nrw1379	23.56	24.12
Berlin52	0.64	0.78
Bier127	1.55	1.75
A280	3.38	4.49

The results of improved SS will be the best because the added steps from CS in different 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 $|RefSet| = |RefSet_1| + |RefSet_2| = 30$.

We 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 IV. When we increase the value of $RefSet_1$ to 20, we found the results for SS and improved SS are better than the results in Table I.

TABLE IV. COMPARISON OF SS AND PROPOSED SS-CS FOR AVERAGE OPTIMALITY WITH $REFSET_1$=20

Instances	*Averages Of fitness for SS*	*Average of fitness Proposed SS-CS*
Fri26	1461	**1012**
Dantzig42	1751	**1129**
Att48	92156	**73987**
Eil51	1005	**973**
Eil101	2312	**1797**
KroA100	118654	**97341**
KroB100	115987	**99762**
KroC100	114982	**98967**
KroD100	111707	**97521**
KroE100	117233	**97939**
KroB200	251087	**113998**
Lin105	84590	**71017**
Lin318	475691	**347531**
Pr76	379328	**258023**
Pr124	500807	**401812**
Pr299	601011	**491763**
Pr439	1562181	**1170739**
Pr1002	5761184	**4348761**
Nrw1379	1104810	**914361**
Berlin52	17981	**12451**
Bier127	478521	**317659**
A280	27234	**19963**

To see clearly the difference between SS and its improvement with $RefSet_1$=20 see Fig. 5.

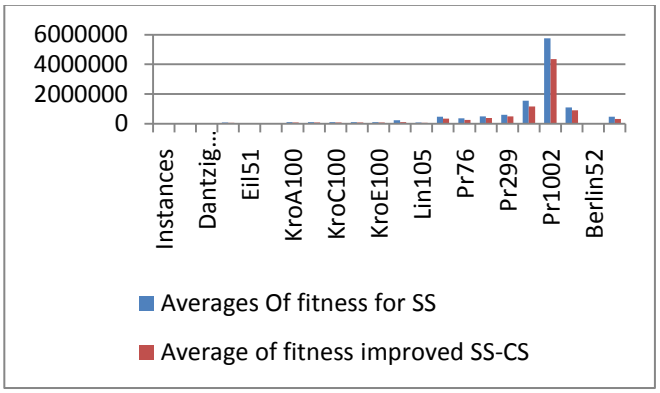

Fig. 4. Difference between SS and proposed SS-CS with RefSet1=20

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 V shows the NOPT results of SS, SS-CS with $RefSet_1$=20. Table VI shows the elapsed time for SS and improved SS with $RefSet_1$=20. The increased time where $RefSet_1$=20 is ≅1.2 second for SS-CS.

TABLE V. COMPARISON OF SS AND THE PROPOSED SS-CS FOR *NOPT* WITH $REFSET_1$=20

Instances	*NOPT in SS*	*NOPT in Proposed SS-CS*
Fri26	1364	**1108**
Dantzig42	1689	**1195**
Att48	84041	**79865**
Eil51	909	**801**
Eil101	2091	**1009**
KroA100	112772	**98124**
KroB100	114654	**98221**
KroC100	113124	**98722**
KroD100	110012	**96912**
KroE100	114789	**93582**
KroB200	231314	**211133**
Lin105	82139	**70997**
Lin318	467549	**400755**
Pr76	373254	**300991**
Pr124	498982	**400001**
Pr299	608723	**490074**
Pr439	1631578	**1410633**
Pr1002	5902741	**5000190**
Nrw1379	1298711	**1079812**
Berlin52	17172	**12018**
Bier127	480941	**400901**
A280	26576	**19754**

In the second experiments, for instances with large size such as Lin318, Pr299, Pr439, Pr1002 and A280 we noticed that the average of elapsed time with improved SS is larger than original SS with approximately 1 second only . This case can lead us to the fact that improved SS 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.

TABLE VI. AVERAGE OF ELAPSED TIME FOR SS AND PROPOSED SS-CS WITH $REFSET_1$=20

instances	*Average of elapsed time for SS (Sec)*	*Average elapsed time for Proposed SS-CS (Sec)*
Fri26	1.27	1.40
Dantzig42	1.67	1.92
Att48	1.96	2.20
Eil51	1.61	1.85
Eil101	2.93	3.32
KroA100	2.83	3.49
KroB100	2.86	3.31
KroC100	2.83	3.55
KroD100	2.89	3.60
KroE100	2.86	3.82
KroB200	6.06	6.79
Lin105	3.15	3.88
Lin318	10.36	11.15
Pr76	2.33	4.72
Pr124	3.47	4.15
Pr299	9.65	10.89
Pr439	14.60	15.85
Pr1002	42.05	43.34
Nrw1379	62.43	66.33
Berlin52	1.70	2.14
Bier127	4.11	4.82
A280	8.96	10.05

In third experiment we compare the *NOPTs* of improved SS in Table VII and VIII with results obtained by other algorithms. We compute the average deviation for the output solutions $SD = 100(NOPT - opt) / opt$, where *NOPT* is the Nearest Optimal Solution output from Improved SS and the *opt* is the optimal solution taken from TSPLIB [20].

TABLE VII. RESULTS OF IMPROVED SS ARE BETTER THAN SOME ALGORITHMS

Instances	Optimal in TSPLIB in [20]	*SD for NOPT* for Proposed SS-CS	*SD* for optimal solutions in[19]	*SD* for optimal solutions in[21]
Fri26	937	28.81	-	34.47
Dantzig42	699	70.95	-	119.45
Att48	10628	670.59	-	573.96
Eil51	426	101.40	-	125.35
Eil101	629	89.03	-	259.61
KroA100	21282	377.87	808.51	378.78
KroB100	22141	356.61	-	347.35
KroC100	20749	391.98	854.24	389.84
KroD100	21294	379.02	-	350.37
KroE100	22068	339.99	-	345.15
KroB200	29437	681.89	828.21	662.59
Lin105	14379	414.61	835.15	393.62
Lin318	41345	968.53	880.41	962.99
Pr76	108159	207.78	744.56	216.44
Pr124	59030	582.54	801.44	599.80
Pr299	48191	944.02	894.60	991.79
Pr439	107217	1271.68	882.16	1209.28
Pr1002	259045	1857.53	927.95	1910.50
Nrw1379	56638	1928.84	891.17	2105.92
Berlin52	7542	96.38	-	127.45
Bier127	118282	253.31	724.70	259.06
A280	2579	717.72	872.48	900.34

TABLE VIII. RESULTS OF IMPROVED SS ARE FAR FROM RESULTS OF SOME OTHER ALGORITHMS

Instances	*SD for NOPT* for Proposed SS-CS	*SD* for optimal solutions in[22]	*SD* for optimal solutions in[23]
Fri26	28.81	0	0
Dantzig42	70.95	0	0
Att48	670.59	0	0
Eil51	101.40	0	0
Eil101	89.03	0.107	0
KroA100	377.87	0	0
KroB100	356.61	0.036	0
KroC100	391.98	0	0
KroD100	379.02	0.019	0
KroE100	339.99	0.001	0
KroB200	681.89	0.509	0
Lin105	414.61	0	0
Lin318	968.53	0.769	0.29
Pr76	207.78	0	0
Pr124	582.54	0	0
Pr299	944.02	0.066	0.01
Pr439	1271.68	0.572	0.18
Pr1002	1857.53	-	-
Nrw1379	1928.84	-	-
Berlin52	96.38	0	0
Bier127	253.31	0.064	0
A280	717.72	0.305	0

Table VII shows how the results of improved SS-CS are better than some results such as in [19] and [21]. Also Table

VIII shows how the improved SS-CS results are far from other results of other algorithms such as [22] and [23].

VII. CONCLUSIONS

This paper presented improved SS algorithms. The improvement provides SS with random exploration for search space of problem and more of diversity and intensification for promising solutions based on the Cuckoo search algorithm. From experimental results, the average of fitness value for improved SS algorithms are better than original SS algorithm, the improvement in the value of average fitness is 23.2% comparing with original SS. From experimental results, the 2improved SS algorithms are better than original SS algorithm in reaching to nearest optimal solutions.

The elapsed time for the improved SS is larger than the elapsed time for original SS in a reasonable value. 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 improved SS 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 improved SS with a mix techniques based on more than one improved steps to obtain the good results.

REFERENCES

[1] El-Ghazali Talbi, "Metaheuristics from Design to Implementation,"John Wiley & Sons, 2009.

[2] F. Glover, Gary A. Kochenberger, "Handbook of Metaheuristics," Kluwer Academic, 2003.

[3] Dorigo Mario, Stützle Thomas,"Ant Colony Optimization". MIT Press, 2004.

[4] Eberhart, R., Y. Shi, and J. Kennedy, "Swarm Intelligence," Morgan Kaufmann, San Francisco, 2001.

[5] Ali M. Sagheer, Ahmed T. Sadiq and Mohammed S. Ibrahim, "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, 2012.

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

[7] Jose A. Egea, Emmanuel Vazquez, Julio R. Banga, Rafael Martí," Improved scatter search for the global optimization of computationally expensive dynamic models", Journal of Global Optimization, vol. 43, pp. 175 – 190, 2009.

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

[9] T. Sari, V. Cakir, S. Kilic and E. Ece, 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, 2011, Poprad, Slovakia.

[10] Tao Zhang, W.A.Chaovalitwongse, YuejieZhang, Scatter search fort hestochastic travel-time vehicle routing problemwith simultaneous pick-ups and deliveries, Elsevier Ltd., Expert Systems with Applications 39 (2012) 6123–6128.

[11] Oscar Ibáñez, Oscar Cordón, Sergio Damas, José Santamaría, An advanced scatter search design for skull-face overlay in craniofacial superimposition, Elsevier Ltd., Expert Systems with Applications 39 (2012) 1459–1473.

[12] Ying Xu and Rong Qu ,A hybrid scatter search meta-heuristic for delay-constrained, multicast routing problems, Springer Science+Business Media, LLC 2010, Appl Intell (2012) 36:229–241.

[13] Jue Wang, Abdel-Rahman Hedar, Shouyang Wang, Jian Ma, Rough set andscatter search metaheuristic based feature selection for credit scoring, Elsevier Ltd., Expert Systems with Applications 39 (2012) 6123–6128.

[14] M. S. Geetha Devasena and M. L. Valarmathi, Meta Heuristic Search Technique for Dynamic Test Case Generation, International Journal of Computer Applications (0975 – 8887) Volume 39– No.12, February 2012.

[15] Laguna, M. and R. Martí, "Scatter Search: Methodology and Implementations in C," Kluwer Academic Press, 2003.

[16] Fred Glove, Manuel Laguna, "Fundamentals of Scatter Search and Path Relinking," Control and Cybernetics, Volume 29, Number 3, pp. 653-684, 2000.

[17] X. S. Yang and S. Deb, "Cuckoo search via Lévy flights". World Congress on Nature & Biologically Inspired Computing (NaBIC 2009). IEEE Publications. pp. 210–214, December, 2009.

[18] H. Zheng and Y. Zhou, "A Novel Cuckoo Search Optimization Algorithm Base on Gauss Distribution", Journal of Computational Information Systems 8: 10, 4193–4200, 2012.

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

[20] Reinelt, G.: TSPLIB, 1995. Available: http://www.iwr.uni heidelberg.de/iwr/comopt/ software/TSPLIB95/

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

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

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