Hindawi Publishing Corporation
Journal of Applied Mathematics
Volume 2013, Article ID 123738, 8 pages
http://dx.doi.org/10.1155/2013/123738

Research Article

Advanced Harmony Search with Ant Colony Optimization for Solving the Traveling Salesman Problem

Ho-Yoeng Yun,[1] Suk-Jae Jeong,[2] and Kyung-Sup Kim[1]

[1] *Department of Industrial Information Engineering, Yonsei University, Seoul 120-749, Republic of Korea*
[2] *Department of Business School, Kwangwoon University, Seoul 139-701, Republic of Korea*

Correspondence should be addressed to Kyung-Sup Kim; kyungkim@yonsei.ac.kr

Received 27 June 2013; Revised 23 September 2013; Accepted 30 September 2013

Academic Editor: Chung-Li Tseng

We propose a novel heuristic algorithm based on the methods of advanced Harmony Search and Ant Colony Optimization (AHS-ACO) to effectively solve the Traveling Salesman Problem (TSP). The TSP, in general, is well known as an NP-complete problem, whose computational complexity increases exponentially by increasing the number of cities. In our algorithm, Ant Colony Optimization (ACO) is used to search the local optimum in the solution space, followed by the use of the Harmony Search to escape the local optimum determined by the ACO and to move towards a global optimum. Experiments were performed to validate the efficiency of our algorithm through a comparison with other algorithms and the optimum solutions presented in the TSPLIB. The results indicate that our algorithm is capable of generating the optimum solution for most instances in the TSPLIB; moreover, our algorithm found better solutions in two cases (kroB100 and pr144) when compared with the optimum solution presented in the TSPLIB.

1. Introduction

The Traveling Salesman Problem (TSP) is a typical example of an NP-complete problem of computational complexity theory and can be understood as a "Maximum Benefit with Minimum Cost" that searches for the shortest closed tour that visits each city once and only once. As is well known, the TSP belongs to a family of NP-complete problems. Generally, when solving this type of problem with integer programming (IP), determining the optimum solution is impossible because the computational time to search the solution space increases exponentially with increasing problem sizes. As a result, the general approach involves determining a near-optimal solution within a reasonable time by applying metaheuristics. In the last decade, TSP has been well studied by many metaheuristic approaches, such as Genetic Algorithm (GA), Simulated Annealing (SA), Tabu Search (TS), Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO), Harmony Search (HS), Cuckoo Search (CS), and Firefly Algorithm (FA). Among these approaches, the general procedures of GA, SA, and TS have already been introduced in many articles [1–3]. ACO is a metaheuristic approach that is inspired by the behavior of ants searching for their food source [4–7]. PSO is originally attributed to Kennedy and Eberhart [8] and was first intended for simulating social behavior as a stylized representation of the movement of organisms in a bird flock or fish school. HS, proposed by Geem et al. [9, 10], is a metaheuristic that was inspired by the improvisation process of musicians. CS is modeled after the obligate brood parasitism of some Cuckoo species by laying their eggs in the nests of other host birds (of other species) [11]. FA is a metaheuristic algorithm that mimics the flashing behavior of fireflies [12]. Freisleben and Merz [13] realized GA by using a new mutation operator of Lin-Kernighan-Opt for finding the high-quality solution in a reasonable amount of time of the asymmetric TSP. Wang and Tian [14] introduced the improved simulated annealing (ISA), which is the integration of the basic simulated annealing (BSA) with the four vertices and three lines inequality to search the optimal Hamiltonian circuit (OHC) or near-OHC. Fiechter [15] proposed the TS for obtaining near-optimal solution of large-size TSPs. The remarkable idea of his research is that while TS seeks a high global quality of the solution, the local search inside TS performed several independent searches

without much loss of quality. Stüzle and Hoos [16] proposed max-min ant system and demonstrated that their proposed algorithm can be significantly improved in performance over the general ant system in most cases discussed of the TSP examples. Wang et al. [17] designed an advanced PSO with the concept of a swap operator and a swap sequence that exhibited good results in a small-size TSP having 14 nodes. Geem et al. [9] applied HS for solving the 20-cities TSP. They combined two operators (neighboring city-going and city-inverting operators) inside the HS to arrive at the global optimum quickly. One of two operators was able to find the closest city that will be visited next after the current city. The other is used to produce a new path by exchanging the sequence of the two nodes selected randomly in one feasible path. Ouyang et al. [18] presented the advanced CS with the "search-new-nest" and "study" operator, derived from idea of "inver-over" operator for solving the spherical TSP. Their experiments demonstrated that CS provided better solutions over GA in HA30 from TSPLIB. Kumbharana and Pandey [19] implemented FA and demonstrated that it provides better solutions than ACO, GA, and SA in most cases of the TSP examples. Many articles mentioned previously are examples of the application of only single metaheuristics or a metaheuristic with a local search.

Recently, however, to complement the weakness of single metaheuristics, a few research studies involving the hybridization of two or more heuristics have been introduced. Pang et al. proposed the combinations of PSO with Fuzzy theory for solving the TSP. In their study, Fuzzy matrices were used to represent the position and velocity of the particles in the PSO, and the symbols and operators in the original PSO formulas were redefined for transformation into the form of the Fuzzy matrices [21]. Thamilselvan and Balasubramanie [22] presented a genetic Tabu Search algorithm, a combined heuristics with the dynamic switching of the GA and the TS. The experimental results indicated that the combination has better solution over the respective individual use of the GA and the TS. Yan et al. [23] introduced a mixed heuristic algorithm to solve the TSP. In their algorithm, SA and ACO were mixed to obtain improved performance. By comparison with the TSPLIB, they determined that the mixed form is much better than (a) the original ACO and the max-min ant system in the convergence rate and (b) the SA in the probability of converging to optimal solution. Chen and Chien [24] presented the parallelized genetic ACO for solving the TSP. They demonstrated improved solutions in three cases of the TSPLIB over Chu et al. [25] with original ACO. Chen and Chien [26] proposed the combination of four metaheuristics (GA, SA, ACO, and PSO) for obtaining a better solution in the TSP. Their experiments tested the combination of four metaheuristics by using 25 datasets of the TSPLIB and demonstrated that it provided better solutions through a comparison with four articles previously published. According to a review of many articles that focused on the combinations of two or more heuristics published since 2006, the combination of HS and other heuristics for solving TSP has been little studied. Therefore, in this paper, we propose the hybridized HS and ACO to solve the TSP. In Section 4, our algorithm will be introduced in detail. The rest of this paper is organized as follows; in Section 2, we introduce the simple overview of both ACO and HS. In Section 3, we describe the advanced HS for solving the TSP, and we explain the overall procedures of the algorithm proposed in Section 4. In Section 5, experiments performed with 20 data sets of TSPLIB are described, and the results of our algorithm and others are compared in the cases of 11 instances involved in TSPLIB. Finally, the conclusion is provided in Section 6.

2. Overview of Ant Colony Optimization and Harmony Search

2.1. Ant Colony Optimization. Ant Colony Optimization (ACO), originally proposed by Dorigo, [4] is a stochastic-based metaheuristic technique that uses artificial ants to find solutions to combinatorial optimization problems. The concept of ACO is to find shorter paths from their nests to food sources. Ants deposit a chemical substance called a pheromone to enable communication among other ants. As an ant travels, it deposits a constant amount of pheromone that the other ants can follow. Each ant moves in a somewhat random fashion, but when an ant encounters a pheromone trail, it must decide whether to follow it. If the ant follows the trail, the ants own pheromone reinforces the existing trail, and the increase in pheromone increases the probability that the next ant will select the path. Therefore, the more ants that travel on a path, the more attractive the path becomes for the subsequent ants. In addition, an ant using a shorter route to a food source will return to the nest sooner. Over time, as more ants are able to complete the shorter route, pheromone accumulates more rapidly on shorter paths and longer paths are less reinforced. The evaporation of pheromone also makes less desirable routes more difficult to detect and further decreases their use. However, the continued random selection of paths by individual ants helps the colony discover alternate routes and ensures successful navigation around obstacles that interrupt a route. ACO, thus, is an algorithm that reflects the stochastic travels of ants by the probability, the evaporation, and the update of pheromone over time. ACO is composed of the state transition rule, the local updating rule, and the global updating rule. Based on the state transition rule as expressed in (1), ants move between nodes. Consider

$$s = \begin{cases} \arg\max\limits_{u \in J_k(r)} [\tau(r,u)]^{\alpha} [\eta(r,u)]^{\beta}, & \text{if } q \le q_0, \\ S, & \text{otherwise,} \end{cases} \tag{1}$$

$\tau(r,u)$ in state transition rule, s is the reciprocal of distance between nodes r and u. $J_k(r)$ means the set of nodes to which ant k in node r can visit in the next time. α, β are parameters that determine the relative importance of pheromone and distance of nodes, respectively.

Whenever ants visit their nodes through the state transition rule, pheromone is updated by the local updating rule. It can be expressed by

$$\tau(r,s) \longleftarrow (1-\rho)\,\tau(r,s) + \rho \Delta\tau(r,s). \tag{2}$$

The pheromone evaporation coefficient ρ is a decimal number in range of 0 to 1. $\Delta\tau(r,s) = \tau_0 = (n \times L_{nn})^{-1}$ is

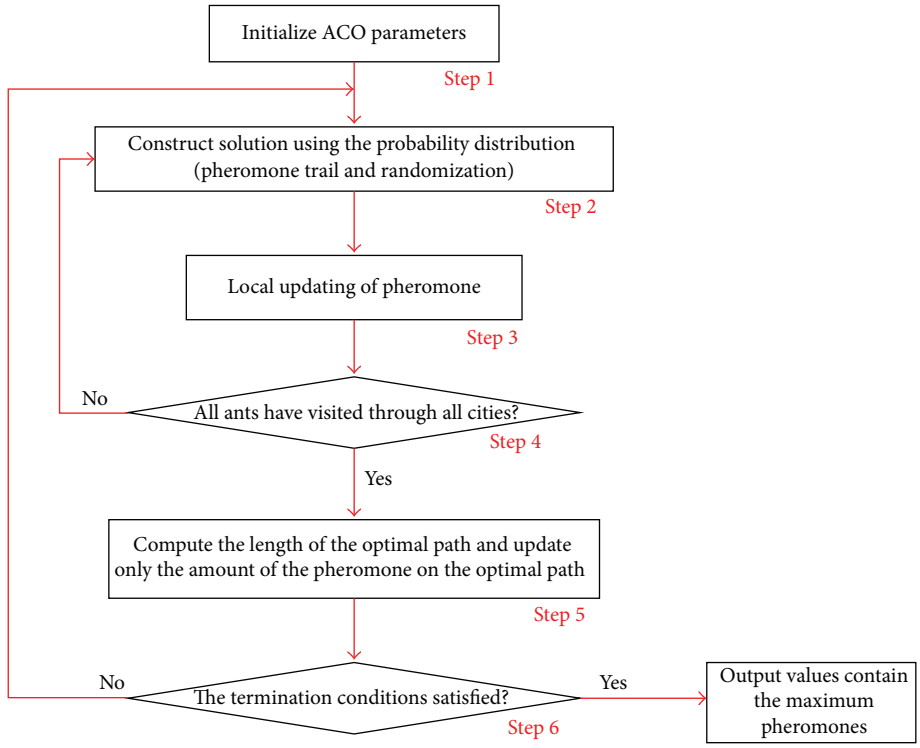

Figure 1: Flow chart of the Ant Colony Optimization.

the amount of initial pheromone. Here, n means the number of cities and L_{nn} is the cost produced by the nearest neighbor heuristic. After all ants have visited through all cities, global updating rule is performed with

$$\tau(r,s) \longleftarrow (1-\rho)\,\tau(r,s) + \rho\Delta\tau(r,s),$$
$$\Delta\tau(r,s) = \begin{cases} L_{gb}^{-1}, & \text{if } (r,s) \in \text{global best tour}, \\ 0, & \text{otherwise}, \end{cases} \tag{3}$$

where ρ is constant and L_{gb} is the global best tour. The general structure of ACO algorithms can be described as follows, and Figure 1 shows the flow chart of the ACO algorithm.

Step 1. Initialize the pheromone table and the ACO parameters.

Step 2. Randomly allocate ants to every node. Every ant must move to next city, depending on the probability distribution.

Step 3. The local pheromone update is performed.

Step 4. If all ants have not visited through all cities, go to Step 2.

Step 5. Compute the optimal path and global update of pheromone.

Step 6. If stopping criteria are not satisfied, go to Step 2.

2.2. Harmony Search. Harmony Search (HS) is a metaheuristic algorithm that mimics the improvisation process of music players and has been very successful in wide variety of optimization problems [9, 10]. In the HS algorithm, the fantastic harmony, the aesthetic standard, pitches of instruments, and each practice in performance process of HS indicate the global optimum, the objective function, the value of variables, and each iteration in optimization process, respectively. HS is composed of optimization operators, such as the harmony memory (HM), the harmony memory size (HMS), the harmony memory considering rate (HMCR), and the pitch adjusting rate (PAR).

HS is conducted by the following steps, and the overall flow chart of the HS algorithm is shown in Figure 2.

Step 1. Initialize the HM and the algorithm parameters.

Step 2. Improvise a new harmony from the HM. A new harmony vector is generated from the HM, based on memory consideration, pitch adjustments, and randomization. $pHMCR$ and $pPAR$ were generated randomly between 0 and 1, respectively, and each operator is selected according to the following conditions.

(i) Condition 1: $pHMCR \le HMCR$ and $pPAR > PAR$; select the memory consideration.

(ii) Condition 2: $pHMCR \le HMCR$ and $pPAR \le PAR$; select the pitch adjustments.

(iii) Condition 3: $pHMCR > HMCR$; select the randomization.

Step 3. If a new harmony is better than the worst harmony in the HM, update the HM.

Step 4. If stopping criteria are not satisfied, go to Step 2.

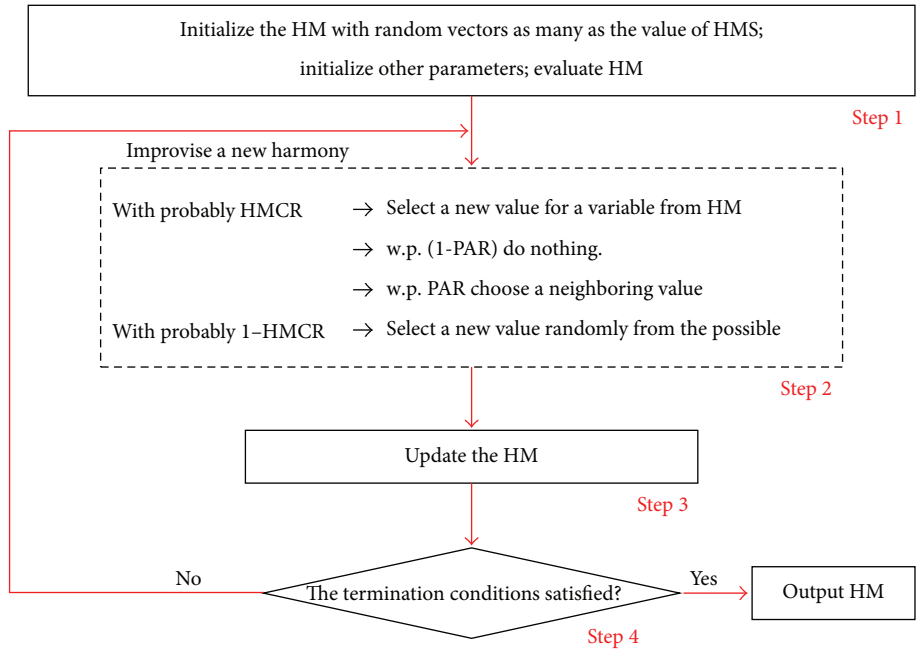

Figure 2: The flow chart for the Harmony Search [20].

3. Advanced Harmony Search for Traveling Salesman Problems

The HS algorithm exhibits good performance in solving a diverse set of problems; however, it has some drawbacks in terms of the sequential problems, such as the TSP and the vehicle routing problem. In case of sequential problems, a close positioning between the nodes implies a strong correlation. HS, however, uses a uniform probability regardless of the correlation between nodes when choosing the new value in a new harmony from the historic values stored in the same index of the existing HM. The memory consideration operator does not even function under the following case: when generating the value of a new harmony under that ith value of index is city 1, if all the values of $(i + 1)$th in the existing HM are city 1, if all the values of $(i+1)$th in the existing HM are city 1. To remedy these shortcomings, we propose the advanced HS (AHS), which includes the fitness, elite strategy, and mutation operators of the GA.

3.1. Revised Memory Consideration and Pitch Adjustments. The memory consideration operator of the original HS runs randomly from the historic values in the HM. In the advanced HS algorithm, however, the memory consideration operator is implemented by using a roulette wheel, so that the fittest index of HM has a greater chance of survival than the weaker ones. Fitness and distance are inversely related. Meanwhile, although the memory consideration operator runs a certain number of times, if the ith value and the candidate $(i + 1)$th value that are selected by the memory consideration operator are the same, the $(i + 1)$th value of a new harmony is determined by a randomization operator. In the case of satisfying condition 1 of Section 2.2, the pitch adjustment generates the $(i + 1)$th index value of a new harmony that is the closest value to the ith value from the possible range of values.

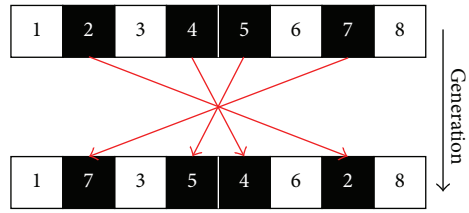

Figure 3: Operator of inversion mutation.

3.2. Elite Preserving Rule and Mutation. The HS algorithm updates in a manner that a new harmony is included in the HM and the existing worst one is excluded from the HM when a new harmony is better than the worst harmony in the existing HM. This mechanism forces to the convergence of all the elements in the HM to the same value that could be the local optimum, when it is repeated infinitely. To escape such case, we consider the inversion operator, one of all mutations of the GA. It is performed for HM that satisfies the following equations: (1 − Elite Rate) × HMS, where (1 − Elite Rate) means the rate of noting the performance of the elite strategy. Inversion mutation operator meanwhile selects a few nodes among all nodes randomly, and the nodes selected are rearranged in inverse order. As shown in example of Figure 3, the previous node (1, 2, 3, 4, 5, 6, 7, 8) is converted to new node (1, 7, 3, 5, 4, 6, 2, 8) through the inversion mutation.

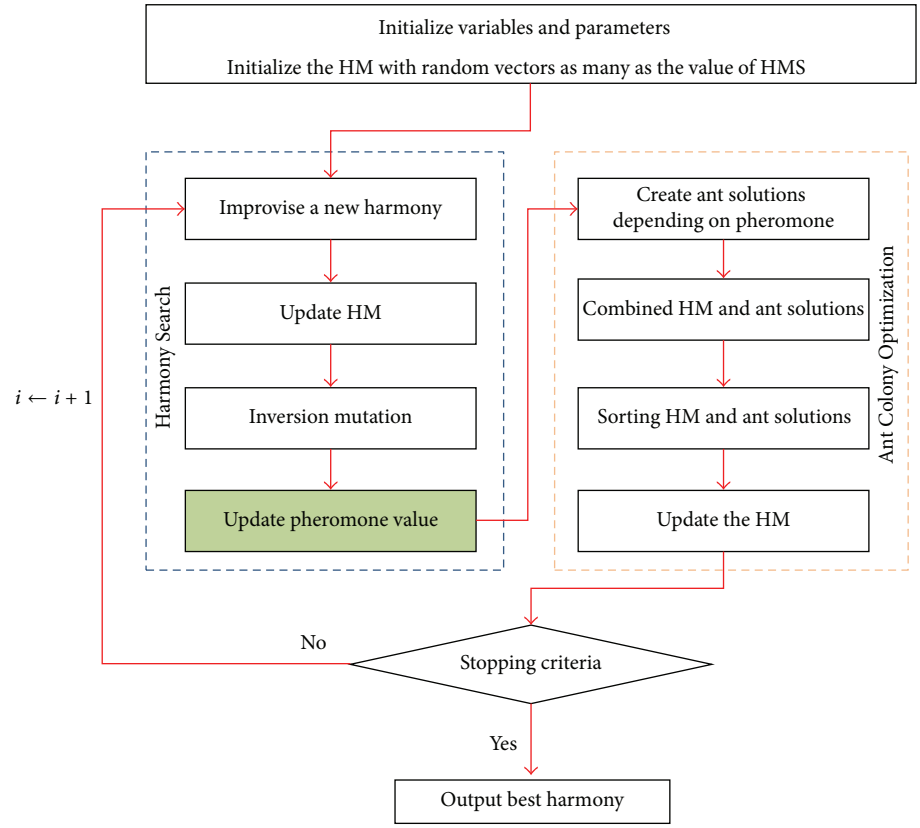

Figure 4: The flow chart for the proposed algorithm.

4. The Proposed Algorithm for the TSP

The overall procedures of our algorithm that combines the AHS and ACO algorithms are shown in Figure 4. First, we generate an initial solution randomly. A pheromone trail is updated. By using memory consideration, pitch adjustment, and randomization under each condition mentioned in Section 2.2, we create a new harmony and check whether an update occurs. When the mutation operator is implemented at a certain probability, the inversion mutation is performed to the rest, except the HM as regarded as Elite, and then the pheromone is updated. After that, ant solutions with the size of HMS are generated by using the ACO algorithm, based on the pheromone trail determined by the HS algorithm. The combined ant solution and HM are stored in a temporary memory that has twice the size of HMS, and they are sorted in ascending order by the total distance, defined as the objective function. The top 50% with higher value in the temporary memory are determined as the new HM. These procedures are repeated until the stopping criteria are satisfied. Pseudocode 1 describes the pseudocode of the proposed algorithm.

5. Experimental Results

Table 1 lists the parameter setting of the proposed algorithm. To show the performance of the proposed algorithm, we performed experiments using a computer with an Intel Core-i5 processor and 2 GB RAM and used C# as the programming language to implement the algorithm. We tested the algorithm using 20 datasets from the TSPLIB (e.g., berlin52, st70, eil76, kroA100, kroB100, kroC100, kroD100, kroE100, eil101, lin105, ch130, pr144, ch150, pr152, d198, tsp225, pr226, pr264, a280, and pr299). For the exact comparison with other algorithms and known best solutions obtained from TSPLIB, the distance between any two cities is calculated as the Euclidian distance and rounded off to 1 decimal place. Each experiment was performed using 1000 iterations and 10 runs, and the best, worst, mean, and standard deviation were recorded for each run. As seen in Table 2, among the 20 datasets tested, we found the optimum solution in 19

Table 1: Parameters settings of the proposed algorithm.

Parameters	Values
HMS	100~200
HMCR	0.9
PAR	0.4
ρ	0.05
α	0.8
β	1.0
p_m	0.01
Iteration	1000

```
Procedure: The proposed algorithm for the TSP
Begin
    Objective function f(x), x = (x_1, x_2, ..., x_d)
    Generate initial harmonics (real number arrays)
    Define harmony memory considering rate (p_HMCR), pitch adjusting rate (p_pa), mutation rate (p_m)
    Initialize the pheromone tables
    Generate initial harmony randomly and apply pheromone update
    while (not_termination)
        for i = 1: number of nodes
            Generate random number variable (rand)
            if (rand < p_HMCR)
                Generate random number variable (rand)
                if (rand < p_pa), generate the nearest city to the previous harmonic
                else choose an existing harmonic the highest fitness probability
                end if
            else generate new harmonics via randomization
            end if
        end for
        Accept the new harmonics (solutions) if better
        Generate random number variable (rand)
        if (rand < p_m) operate inversion mutation end if
        Apply the pheromone update
        Create as many cities as the HMS based pheromone using Ant Colony Optimization
        Update harmony memory and apply pheromone update
    end while
    Find the current best solutions
End
```

Pseudocode 1: The pseudo-code for the proposed algorithm (AHS-ACO).

Table 2: Results of our algorithm (AHS-ACO) for 20 TSP instances from the TSPLIB.

TSPLIB	Known best solutions	Solution				Running Time	Relative error (%)
		Best	Worst	Mean	STDEV	Second (s)	Best
berlin52	7542	7542	7542	7542	0.000	15.12	0.000
st70	675	675	677	675.375	0.812	19.45	0.000
eil76	538	538	542	540.494	1.473	22.27	0.000
kroA100	21282	21282	21378	21307.554	34.983	40.97	0.000
kroB100*	22141	**22139***	22271	22193.114	53.678	42.87	**−0.009**
kroC100	20749	20749	20868	20770	34.937	39.62	0.000
kroD100	21294	21294	21467	21338.03	63.797	41.24	0.000
kroE100	22068	22068	22117	22093.099	14.819	42.38	0.000
eil101	629	629	643	634.355	4.479	49.47	0.000
lin105	14379	14379	14541	14434.947	59.097	56.27	0.000
ch130	6110	6110	6200	6173.038	24.544	69.24	0.000
pr144*	58537	**58534***	58902	58659.283	144.807	80.27	**−0.003**
ch150	6528	6528	6586	6554.589	17.303	97.34	0.000
pr152	73682	73682	74754	73846.437	325.185	111.59	0.000
d198	15780	15780	15963	15876.892	70.497	180.27	0.000
tsp225	3919	**3859**	4013.724	3977.047	26.516	208.48	0.000
pr226	80369	80369	80882	80558.519	174.915	213.78	0.000
pr264	49135	49135	49379	49205.87	75.675	279.69	0.000
a280	2579	2579	2726	2641.61	44.91	303.36	0.000
pr299	48191	48195	49989	49121.289	577.831	367.72	0.016

Table 3: Comparison result of our algorithm with the results of Randall and Montgomery (2003) [27].

TSPLIB	Randall and Montgomery (2003) [27]			Proposed algorithm		
	Best	Mean	Worst	Best	Mean	Worst
berlin52	7547	7790	8148	**7542**	**7542**	**7542**
st70	678	687	712	**675**	**675.376**	**677**
eil76	546	555	559	**538**	**540.494**	**542**
kroA100	21373	21512	21915	**21282**	**21307.554**	**21378**
ch130	6180	6269	6407	**6110**	**6173.038**	**6200**
d198	16044	16209	16465	**15780**	**15876.892**	**15963**

Table 4: Comparison result of our algorithm with the results of Chen and Chien (2011) [26].

TSPLIB	Chen and Chien (2011) [26]			Proposed algorithm		
	Best	Mean	SD	Best	Mean	SD
berlin52	7542	7542	0.00	**7542**	**7542**	**0.00**
eil76	538	540.20	2.94	**538**	540.494	**1.473**
kroA100	21282	21370	123.36	**21282**	**21307.554**	**34.983**
kroB100	22141	22282.87	183.99	**22139**	**22193.114**	**53.678**
kroC100	20749	20878.97	158.64	**20749**	**20770**	**34.937**
kroD100	21309	21620.47	226.60	**21294**	**21338.03**	**63.797**
kroE100	22068	22183.47	103.32	**22068**	**22093.099**	**14.819**
eil101	630	635.23	3.59	**629**	**634.355**	4.479
lin105	14379	14406.37	37.28	**14379**	14434.907	59.097
ch130	6141	6205.63	43.70	**6110**	6173.038	**24.544**
ch150	6528	6563.70	22.45	**6528**	**6554.589**	**17.303**

datasets, except for pr299, indicated from the TSPLIB. In the case of kroB100 and pr144, in particular, our algorithm outperformed the known best solutions from the TSPLIB (see the asterisks of Table 2 for details).

To validate the superiority of our algorithm, we compared it with Randall and Montgomery [27] and Chen and Chien [24, 26]. Randall and Montgomery [27] proposed accumulated experience ant colony (AEAC) for using the previous experiences of the colony to guide in the choice of elements, and Chen an Chien [24, 26] solved TSP with combination of four metaheuristics having GA, SA, ACO, and PSO. Tables 3 and 4 show the comparative results with two previous researches, respectively.

6. Conclusion

In this paper, we proposed the AHS-ACO algorithm, which is a combination of the advanced Harmony Search and the Ant Colony Optimization algorithms, to solve the TSP. We modified the generic HS algorithm to produce a new HS algorithm that includes the fitness, elite strategy, and mutation operators in the GA, and we combined the ACO algorithm inside the HS algorithm to overcome the shortcomings of the HS algorithm for solving sequential problems. We performed experiments using the AHS-ACO algorithm on 20 datasets of the TSPLIB. As shown in the experimental results, we found the optimal solution obtained from the TSPLIB in almost all cases of the TSPLIB; moreover, our algorithm provided a better solution over the TSPLIB solution in the cases of kroB100 and pr144. The results of this paper indicate that the HS algorithm can be a good method, in combination with other heuristics, to solve sequential problems such as TSP, as well as many other problems.

Acknowledgments

This research was supported by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (2010-0023236).

References

[1] J. H. Holland, *Adaptation in Natural and Artificial Systems*, MIT Press, Cambridge, Mass, USA, 1975.

[2] S. Kirkpatrick, C. D. Gelatt, Jr., and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.

[3] F. Glover and M. Laguna, *Tabu Search*, Springer, 1997.

[4] M. Dorigo, *Learning and Nature Algorithm [Ph.D. thesis]*, Dipartimento di Electtonica, Politecnico di Milano, Milano, Italy, 1992.

[5] L. M. Gambardella and M. Dorigo, "Solving symmetric and asymmetric TSPs by ant colonies," in *Proceedings of the IEEE International Conference on Evolutionary Computation (ICEC '96)*, pp. 622–627, May 1996.

[6] M. Dorigo and L. M. Gambardella, "Ant colony system: a cooperative learning approach to the traveling salesman problem,"

IEEE Transactions on Evolutionary Computation, vol. 1, no. 1, pp. 53–66, 1997.

[7] M. Dorigo and L. M. Gambardella, "Ant colonies for the travelling salesman problem," *BioSystems*, vol. 43, no. 2, pp. 73–81, 1997.

[8] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of the IEEE International Conference on Neural Networks*, vol. 4, pp. 1942–1948, December 1995.

[9] Z. W. Geem, J. H. Kim, and G. V. Loganathan, "A new heuristic optimization algorithm: Harmony search," *Simulation*, vol. 76, no. 2, pp. 60–68, 2001.

[10] K. S. Lee and Z. W. Geem, "A new meta-heuristic algorithm for continuous engineering optimization: harmony search theory and practice," *Computer Methods in Applied Mechanics and Engineering*, vol. 194, no. 36–38, pp. 3902–3933, 2005.

[11] X.-S. Yang and S. Deb, "Cuckoo search via Lévy flights," in *Proceedings of the World Congress on Nature and Biologically Inspired Computing (NABIC '09)*, pp. 210–214, IEEE Publications, December 2009.

[12] X. S. Yang, *Nature-Inspired Metaheuristic Algorithms*, Lunvie Press, Frome, UK, 2008.

[13] B. Freisleben and P. Merz, "Genetic local search algorithm for solving symmetric and asymmetric traveling salesman problems," in *Proceedings of the IEEE International Conference on Evolutionary Computation (ICEC '96)*, pp. 616–621, May 1996.

[14] Y. Wang and D. Tian, "An improve simulated annealing algorithm for traveling salesman problem," in *Proceedings of the International Conference on Information Technology and Software Engineering*, vol. 211 of *Lecture Notes In Electrical Engineering*, pp. 525–532, 2013.

[15] C.-N. Fiechter, "A parallel tabu search algorithm for large traveling salesman problems," *Discrete Applied Mathematics*, vol. 51, no. 3, pp. 243–267, 1994.

[16] T. Stüzle and H. Hoos, "MAX-MIN Ant system and local search for the traveling salesman problem," *Reference Future Generations Computer Systems*, vol. 16, no. 8, pp. 889–914, 1997.

[17] K.-P. Wang, L. Huang, C.-G. Zhou, and W. Pang, "Particle swarm optimization for traveling salesman problem," in *Proceedings of the International Conference on Machine Learning and Cybernetics*, pp. 1583–1585, Xi'an, China, November 2003.

[18] X. Ouyang, Y. Zhou, Q. Luo, and H. Chen, "A novel discrete cuckoo search algorithm for spherical traveling salesman problem," *Applied Mathematics & Information Sciences*, vol. 7, no. 2, pp. 777–784, 2013.

[19] S. N. Kumbharana and G. M. Pandey, "A comparative study of ACO, GA and SA for solving traveling salesman problem," *International Journal of Societal Applications of Computer Science*, vol. 2, no. 2, pp. 224–228, 2013.

[20] A. Kaveh and S. Talatahari, "Particle swarm optimizer, ant colony strategy and harmony search scheme hybridized for optimization of truss structures," *Computers and Structures*, vol. 87, no. 5-6, pp. 267–283, 2009.

[21] W. Pang, K.-P. Wang, C.-G. Zhou, and L.-J. Dong, "Fuzzy discrete particle swarm optimization for solving traveling salesman problem," in *Proceedings of the 4th International Conference on Computer and Information Technology (CIT '04)*, pp. 796–800, September 2004.

[22] R. Thamilselvan and P. Balasubramanie, "A genetic algorithm with a Tabu search(GTA) for traveling salesman problem," *International Journal of Recent Trends in Engineering*, vol. 1, no. 1, 2009.

[23] Y. Yan, X. Zhao, J. Xu, and Z. Xiao, "A mixed heuristic algorithm for traveling salesman problem," in *Proceedings of the 3rd International Conference on Multimedia Information Networking and Security (MINES '11)*, pp. 229–232, November 2011.

[24] S.-M. Chen and C.-Y. Chien, "Parallelized genetic ant colony systems for solving the traveling salesman problem," *Expert Systems with Applications*, vol. 38, no. 4, pp. 3873–3883, 2011.

[25] S.-C. Chu, J. F. Roddick, and J.-S. Pan, "Ant colony system with communication strategies," *Information Sciences*, vol. 167, no. 1–4, pp. 63–76, 2004.

[26] S.-M. Chen and C.-Y. Chien, "Solving the traveling salesman problem based on the genetic simulated annealing ant colony system with particle swarm optimization techniques," *Expert Systems with Applications*, vol. 38, no. 12, pp. 14439–14450, 2011.

[27] M. Randall and J. Montgomery, "The accumulated experience Ant colony for the traveling salesman problem," *International Journal of Computational Intelligence and Applications*, vol. 03, no. 2, pp. 189–198, 2003.