A Standardized Bare Bones Particle Swarm Optimization Algorithm for Traveling Salesman Problem

Jia Guo and Yuji Sato

***Abstract*—The traveling salesman problem (TSP) plays an important role in theoretical computer science. It can be used to solve different route planning problems in the real world and has been proved to be NP-hard. Plenty of researchers tried to solve the TSP by the population-based algorithms. However, as a real-world problem, the TSP has two major differences with the benchmark functions. One is the visiting order of the TSP is a combination of integers and the other one is each city in the problem has to be exactly visited once. To cross the two problems, the standardized bare bones particle swarm optimization (SBBPSO) algorithm is proposed in this work. The Gaussian distribution is used to select the positions of particles in the next generation. A standardized converter is used to ensure the dimensions of each particle are integers and each city has been visited exactly once. To test the performance of the SBBPSO, several famous instances are used in the experiments. Also, the standard bare bones particle swarm optimizations algorithm is used as the control group. The experimental results confirm that the SBBPSO is able to solve the TSP.**

***Index Terms*—Bare bones, particle swarm optimization, standardized, traveling salesman problem.**

I. Introduction

The particle swarm optimization (PSO) algorithm is first proposed by Kennedy and Eberhart in 1995 [1]. Particles are designed to have memories to record the best position they have ever been. Also, the swarm can record the best position from all particles. The next position of a particle is calculated from the current position and the velocity. The velocity is affected by the personal best position and the global best position. The bare bones particle swarm optimization (BBPSO) algorithm [2] is a simple version of the PSO algorithm. The next position of each particle is selected from the Gaussian distribution. With the cancel of the velocity item, the algorithm becomes parameter free. No human iterative is needed during the iteration. The BBPSO based algorithms are used in data clustering [3], feature selection [4], image classification [5] and so on.

The traveling salesman problem (TSP) is a classic route planning problem. In the TSP, a salesman needs to visit several cities. He or she has to set up from one city, traverses every city only once, then return to the starting point. Our mission is finding the shortest route in all traveling plans. The description of the TSP is simple and it is able to be solved by the enumeration when the number of the city is small. However, researchers noticed that the total routes increase crazily when the number of the city goes up. It can be calculated that the total number of routes is $n!$, where n is the number of cities. It is obviously unable to list all the routes then find the best one. Researchers begin to try different methods on the TSP. However, to the best of the writer's knowledge, most of the researches try to solve the TSP by ant clone optimization (ACO) [6], [7], bee colony algorithms [8] and the genetic algorithm (GA) [9]. These methods need a lot of preliminary work and parameter debugging, which makes them difficult to apply to different problems.

To solve this problem, a standardized bare bones particle swarm optimization algorithm is proposed in this rest of this paper. Section II gives a brief review of the history of the TSP and the BBPSO based methods. Section III gives an introduction of the proposed methods. Section IV introduces the experimental methods and results. Section V gives the conclusion of this paper. Also, since all test functions used in this paper are minimal problems, a better position means a position has a smaller function value.

Manuscript received August 1, 2019; revised February 12, 2020. This work is supported by JSPS KAKENHI Grant Numbers JP19K12162.

Jia Guo is with the Graduate School of Computer and Information Sciences, Hosei University, Tokyo, Japan (e-mail: guojia314@gmail.com).

Yuji Sato is with the Faculty of Computer and Information Sciences, Hosei University, Tokyo, Japan (e-mail: yuji@hosei.ac.jp).

II. Relate Works

A. The History of the Traveling Salesman Problem

The TSP has been studied over years. Researchers tried to solve it by different methods. For instances, an ant colony algorithm based method which is called pattern reduction enhanced ant colony optimization (PREACO) is used to solve the TSP in [10]. The PREACO is motivated by the observation that many of the computations of ACO on its convergence process are essentially redundant and thus can be eliminated to reduce its computation time. Also, Arshad proposed a genetic algorithm for the TSP in [11]. A two-phase hybrid approach for the TSP is proposed in this research. The first phase of the proposed method is based on a sequence based genetic algorithm (SBGA) with an embedded local search scheme. Within the SBGA, a memory is introduced to store good sequences (sub-tours) extracted from previous good solutions and the stored sequences are used to guide the generation of offspring via local search during the evolution of the population. Additionally, some techniques are also applied to adapt the key parameters based on whether the best individual of the population improves or not and maintain the diversity. After SBGA finishes, the hybrid approach enters the second phase, where the inver over (IO) operator, which is a state-of-the-art algorithm for

the TSP, is used to further improve the solution quality of the population.

Li [12] described the colored traveling salesman problem (CTSP). The multiple traveling salesman problem (MTSP) is an important combinatorial optimization problem. Meng [13] presented a variable neighborhood search for the CTSP. A colored traveling salesman problem (CTSP) is a generalization of the well-known multiple traveling salesman problem. A variable neighborhood search (VNS) approach was proposed in this work. Extensive simulation was conducted and the results shown that the proposed VNS is able to solve CTSP.

The particle swarm optimization is combined with the k-means algorithm to solve the TSP in [14]. The K-means algorithm is employed to find the city clustering and then solve a sequence of sub-city in a given order by the PSO. The performance of the proposed method is tested against a number of instances from the TSPLIB. Results demonstrate the effectiveness of the proposed method. Moreover, the novel method gives better results in the standard TSP problem than the exist algorithms.

B. The Bare Bones Particle Swarm Based Methods

With the canceling of the velocity term, the BBPSO becomes much easier to apply to applications. The next position of each particle is selected by the Gaussian distribution. On the foundation of the BBPSO, Guo [15] proposed a pair-wise bare bones particle swarm optimization algorithm (PBBPSO). Particles are designed to be members in a society. The big society is consist by sever small units. Two particles are designed in one unit. The two particles will evolve with different rules. This method delays the diversity losing of the particle swarm. Besides the Gaussian distribution, the Cauchy jumps is combined with the BBPSO in [16].

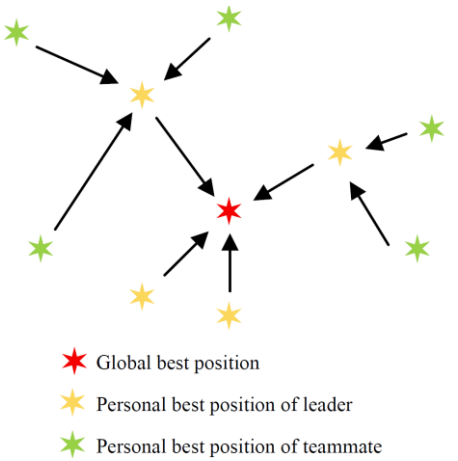


Fig. 1. The possible situation of DNS.

In 2017, Guo [17] proposed a bare bones particle swarm optimization algorithm with dynamic local search (DLS-BBPSO). The swarm in the DLS-BBPSO will be spilled to several local groups. Each local group has one leader and several teammates. It is possible that no teammate exists in a local group. It is also possible that all particles are in one local group. The algorithm is called dynamic is because the number of local groups and the number of particles in each group are not certain. Also, no parameter is needed during the dynamic splitting process. One of the possible local structures of DLS-BBPSO is shown in Fig. 1.

In the other direction, a dynamic update rule is formulated for the BBPSO [18]. In 2019, Guo [19] combined a fission and fusion strategy with the bare bones particle swarm optimization (FHBBPSO). In the FHBBPSO, local groups get better positions by competing with each other. The fission strategy aims at splitting the search space. Particles are assigned to different local groups to sample the corresponding regions. The fusion strategy aims at narrowing the search space. Marginal groups will be gradually merged by the central groups until there is only one group left. The two strategies work together for the theoretical optimal value. Moreover, the FHBBPSO is able to apply to different problems without parameter adjustment because no parameter is used during the iteration process. To verify the optimization ability of the FHBBPSO, the algorithm runs over the CEC2014 benchmark functions. The test functions are composed of four parts: the unimodal functions, the simple multimodal functions, the hybrid functions, and the composition functions. The experimental results confirmed the optimization ability of FHBBPSO.

On the other hand, a Scale Matrix Adaptation is used to improve the search ability of the BBPSO [20]. Guo [21] also implemented a dynamic allocation strategy to the BBPSO (DABBPSO). In the DABBPSO, particles with different personal best values are considered to have different capacities. To be specific, the particles which have a higher position in the personal best value ranking will be good at finding more accurately global best position while the rest are good at escaping from the current local optimum. The top particles are assigned to the main group while others are allotted into the ancillary group. The two groups working in different ways and their cooperation helps the algorithm to find the global best position. The proposed method increases the search ability by appointing suitable works to suitable particles.

However, most of the above researches aim at benchmark functions. There are sever inevitable differences between benchmark functions and the real world problems. One of them is that most problems in the real world need to implement binary choice. For example in the TSP, the traveling plan is an order of visiting the city. Each city has to appear exactly once. This feature makes it difficult to use the BBPSO to the TSP. Hence, to solve these problems, a standardized bare bones particle swarm optimization algorithm will be presented in the next section.

III. The Standardized Bare Bones Particle Swarm Optimization Algorithm for the Traveling Salesman Problem

A. The Dynamic Neighbor Selection

The performance of the original BBPSO is limited by its iterative pattern. Every particle is searching around the

global best particle makes the swarm losing diversity very fast. This feature makes the BBPSO performances very weak when facing the multimodal problems. To cross this shortage, the dynamic neighbor selection (DNS) strategy is proposed. At the beginning of the DNS, we will select a particle *k* and a random neighbor of it. Then we will compare the fitness value of the *k* and its neighbor. Since we are talking about the shortest problems in this paper, we will define that a particle with a smaller fitness value is a better particle. If the particle is better than its neighbor, it will ignore this neighbor and moving to the global best particle. In this situation, the next position of this particle is selected by the equation (1):

$$\begin{aligned} \alpha &= \frac{(pbest(k)+gbest)}{2} \\ \beta &= |pbest(k)-gbest| \\ x(k)_{new} &= N(\alpha,\beta) \end{aligned} \tag{1}$$

where the $pbest=(pbest(1), pbest(2),..., pbest(n))$ is a matrix used for recording the best position each particle has ever reached; the $X=(x(1),x(2),...,x(n))$ is a matrix used to record the next position of particles; the $x(k)_{new}$ is the next position of the particle k ; the $gbest$ is the best position that all particles has ever reached; the $N(\alpha,\beta)$ is a Gaussian distribution with a mean value α and standard deviation β.

On the other side, if the particle k is worse than the selected neighbor, it will search around the neighbor. The next position of the particle k is selected by the equation (2):

$$\begin{aligned} \gamma &= \frac{(pbest(k)+pbest(neighbor))}{2} \\ \delta &= |pbest(k)-pbest(neighbor)| \\ x(k)_{new} &= N(\gamma,\delta) \end{aligned} \tag{2}$$

where the $pbest=(pbest(1), pbest(2),..., pbest(n))$ is a matrix used for recording the best position each particle has ever reached; the $X=(x(1), x(2),..., x(n))$ is a matrix used to record the next position of particles; the $x(k)_{new}$ is the next position of the particle k ; the $N(\gamma,\delta)$ is a Gaussian distribution with a mean value γ and standard deviation δ . After that, the next particle will find a random neighbor and select a new position using the same logic. The DNS ends when every particle finds a new position. Obviously, there is no grantee that the new position of a particle is better than the old one. If the new position is worse than the current personal best position, the personal best position will still be used in the next generation.

In the DNS method, we will not expect every particle can find a better position in every iteration. On the other hand, sometimes we are more willing to see some particles stay far from the central area. Compare with the BBPSO, particles in the SBBPSO are given an additional chance. This strategy keeps the diversity of the swarm. Since the selection of the neighbor is random, it is possible that one particle is very from the selected neighbor. This article will implement a global search in the next generation. Conversely, if the particle and the neighbor stay close, the particle will implement a local search. The schematic diagram of global and local search are shown in Fig. 2. The Fig. 2 consists of three parts. In the left part, the selected particle moves to its neighbor while the global best particle is in the middle of them. This search pattern is called the global search because the moment is region-crossed. In the middle of the figure, the global best particle has a long distance from the region that the selected particle and its neighbor belong to. The search in this pattern will implement near the selected particle. Hence we named its local search. At the right part of the figure, both the selected particle and its neighbor are moving to the global best particle. This pattern is the same as the BBPSO.

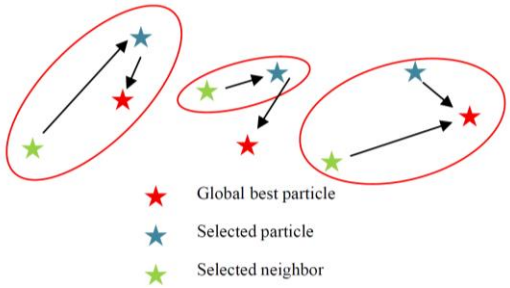


Fig. 2. The possible situation of DNS.

B. The Standardized Method

On the mathematic side, the TSP can be described by the equations (3):

$$\begin{aligned} City &= (c(1),c(2),...,c(NC)) \\ Dis &= \begin{pmatrix} d(1,1) & d(1,2) & ... & d(1,n) \\ d(2,1) & d(2,2) & ... & d(2,n) \\ ... & ... & ... & ... \\ d(n,1) & d(n,2) & ... & d(n,n) \end{pmatrix} \\ X &= (x_1,x_2,...,x_n) \\ Total &= \sum_{i=1}^{n-1} (d(x_i, x_{i+1})) + d(x_n, x_1) \end{aligned} \tag{3}$$

where the *City* is a matrix to record all cities, the n is the number of cities, the *D* is a matrix to record the distances between all cities, the X is a matrix used to record the visiting order of the cities, the *Total* is the total distance after a round trip. The aim of the TSP is find a suitable X which can minimize the *Total* .

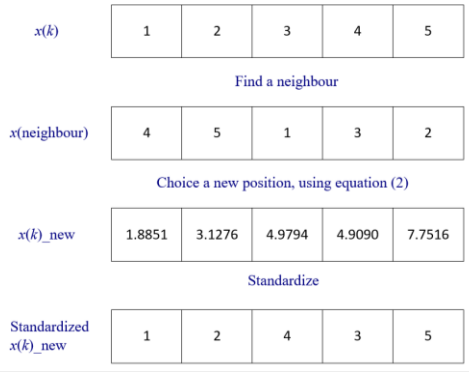


Fig. 3. An example of the standardized method.

Algorithm 1 SBBPSO

Require: Max iteration time, T
Require: The number of cities, NC
Require: Fitness function, F
Require: Particle swarm, $S = (s(1), s(2), ..., s(N))$
Require: The number of particles in the particle swarm, N
Require: Personal best position,
$Pbest = (pbest(1), pbest(2), ..., pbest(n))$
Require: Global best position, $Gbest$

```
t = 0
while t < T do
  i = 1
  while i in range [1, NC] do
    initialize a random integer j in range [1, NC]
    if pbest(i) <= pbest(j) then
      Update s(i) with Equation (1)
    else
      Update s(i) with Equation (2)
    end if
    i = i + 1
  end while
  Update S using the Equation (4)
  Update Pbest
  Update Gbest
  t=t+1
end while
```

Fig. 4. The pseudo-code of the SBBPSO.

It can be seen that the TSP has two major differences from the normal benchmark functions. First is the solution X is a combination of integers. After all, we are unable to visit a half city. Second is every solution starts and ends in the same city, other cities have to appear exactly once. These features make the normal methods are unable to use on the TSP directly. To ensure the position of every particle is legal, a standardized method is proposed in this part. The standardized method can be described by the equation (4):

$$X = (x_1, x_2, ..., x_n)$$
$$new\ x_i = Rank(x_i)\ i \in [1, n] \tag{4}$$

where $X = (x(1), x(2), ..., x(n))$ is a matrix used to record the next position of particles and the $Rank(x)$ is a rank function. The $Rank(x)$ will rank each dimension of X, then replace the original value. The process of the standardized method is shown in Fig. 3.

C. The Process of the SBBPSO

Before iteration, a set of combination of integers from 1 to *n* is used to initialize the particles. Then the SBBPSO will calculate the first personal best of each particle, the first global best of the swarm. The pseudo-code of the SBBPSO is shown in Fig. 4.

IV. Experiments

A. Experimental Methods and Results

To verify the performance of the SBBPSO, four famous instances are used in experiments. Instance1 and Instance2 are selected form the National TSP Problem, Instance3 and Instance4 are selected from the TSBLIB. In the control group, the BBPSO is implemented under the same conditions. The standard BBPSO is combined with the DNS method in experiments. All of the experiments are implemented on a computer with an Intel Core i7-6700 CPU and a 16G RAM. The operating system is Windows 10 and all codes are written in the Matlab R2016. The experimental results are shown in Table I. The *Best* and *Worst* in the table means the best and the worst result from the 30 independent runs, the *Mean* means the average result from the 30 runs. The number of particles is 100, the max iteration time is 5000. The empirical error (EE) is defined as |*result-solution*| where the *result* is the current output of an algorithm, and the *solution* is the theoretical optimal solution of the test instance.

It can be seen that the SBBPSO gives better results in three of the four test instances. In *Instances1*, the mean EE of SBBPSO is 19.23\% smaller than the EE of BBPSO. In *Instances2*, the mean EE of SBBPSO is 6.07\% smaller than the EE of BBPSO. In *Instances3*, the mean EE of SBBPSO is 12.99\% smaller than the EE of BBPSO. In *Instances4*, the mean EE of SBBPSO is 7.33\% larger than the EE of BBPSO.

B. Discussion

According to the experimental results, the SBBPSO performances better than the BBPSO. It can be attributed to the dynamic neighbor searching method. In the BBPSO, the next position of every particle is connected with the global best particle. The overusing of the information makes the swarm losing diversity very fats. Also, the swarm is easy to be trapped in a local minimum when every particle evolving in the same direction. Obviously, the DNS method made up for this defect. Some particles will do the local search and some will do the global search at the same time. The global search gives the swarm more chances to escape from a local minimum. The local search enhances a more precise search in a local area.

TABLE I: Experimental Results

Test Instances	The number of cities	BBPSO/EE			SBBPSO/EE		
		Best	Worst	Mean	Best	Worst	Mean
Instance1 = wi29	29	5082	16798	1.04E+04	3546	19207	8.40E+03
Instance2 = dj38	38	3236	6884	4944	2571	6864	4644
Instance3 = pr124	124	188295	291868	2.31E+05	162220	262222	2.01E+05
Instance4 = rat195	195	6362	10502	8.18E+03	6880	10553	8.78E+03

V. Conclusions

A standardized bare bones particle swarm optimization (SBBPSO) algorithm, which aims at solving the traveling salesman problem (TSP), is proposed in this paper. A coordinate transformation (CT) method is used to convert the coordinate of cities to weights. Then, a Gaussian distribution is used to select the next generation of the routing plan. After that the CT method will convert the weights to coordinates for counting the fitness value. Also, different from the

BBPSO, a new particle evolution model is implemented in the SBBPSO. Each particle will search a random neighbor and make a comparison with it. If the neighbor wins the particle will search around the neighbor, otherwise, the particle will search around the global best particle. To verify the search ability of the SBBPSO, four famous instances are used in the experiments. Finally, the experimental results and the statistical analysis confirmed the ability of the SBBPSO on the TSP.

Although SBBPSO shows good performance on the tested problems, we will not stop here. We will improve the accuracy of the algorithm in future works. Also, we are planning to use the SBBPSO on more complete problems like colored TSP, vehicle routing problem and dynamic planning for traffic lights.

Conflict of Interest

The authors declare no conflict of interest.

Author Contributions

Yuji Sato directed this research. Jia Guo designed the experiments, analyzed the experimental data, and wrote the paper.

References

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

[2] J. Kennedy, "Bare bones particle swarms," in *Proc. Swarm Intelligence Symposium*, 2003, pp. 80–87.

[3] B. Jiang and N. Wang, "Cooperative bare-bone particle swarm optimization for data clustering," *Soft Computing*, vol. 18, no. 6, pp. 1079–1091, 2014.

[4] C. Li, H. Hu, H. Gao, and B. Wang, "Adaptive bare bones particle swarm optimization for feature selection," in *Proc. the 28th Chinese Control and Decision Conference*, 2016, pp. 1594–1599.

[5] M. Omran and S. Al-Sharhan, "Barebones particle swarm methods for unsupervised image classification," in *Proc. 2007 IEEE Congress on Evolutionary Computation*, Sep. 2007, pp. 3247–3252.

[6] A. Uchida, Y. Ito, and K. Nakano, "An efficient GPU implementation of ant colony optimization for the traveling salesman problem," in *Proc. 2012 Third International Conference on Networking and Computing*, 2012, pp. 94–102.

[7] M. Dorigo and L. 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, Apr. 1997.

[8] L.-P. Wong, M. Yoke, H. Low, and C. S. Chong, "Bee colony optimization with local search for traveling salesman problem," in *Proc. 6th IEEE International Conference on Industrial Informatics*, 2008, pp. 1019–1025.

[9] H. D. Nguyen, I. Yoshihara, K. Yamamori, and M. Yasunaga, "Implementation of an effective hybrid GA for large-scale traveling salesman problems," *IEEE Transactions on Systems, Man and Cybernetics, Part B (Cybernetics)*, vol. 37, no. 1, pp. 92–99, Feb. 2007.

[10] S.-P. Tseng, C.-W. Tsai, M.-C. Chiang, and C.-S. Yang, "A fast ant colony optimization for traveling salesman problem," in *Proc. IEEE Congress on Evolutionary Computation*, 2010, pp. 1–6.

[11] S. Arshad and S. Yang, "A hybrid genetic algorithm and inver over approach for the travelling salesman problem," in *Proc. IEEE Congress on Evolutionary Computation*, 2010, pp. 1–8.

[12] J. Li, S. Member, M. Zhou, Q. Sun, X. Dai, and X. Yu, "Colored traveling salesman problem," *IEEE Transactions on Cybernetics*, vol. 45, no. 11, pp. 2390–2401, 2015.

[13] X. Meng, J. Li, S. Member, X. Dai, and J. Dou, "Variable neighborhood search for a colored traveling salesman problem," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 4, pp. 1018–1026, 2018.

[14] M.-A. Munlin, "Hybrid K-means and particle swarm optimization for symmetric traveling salesman problem," in *Proc. 2015 IEEE 10th Conference on Industrial Electronics and Applications (ICIEA)*, 2015, pp. 671–676.

[15] J. Guo and Y. Sato, "A pair-wise bare bones particle swarm optimization algorithm for nonlinear functions," *International Journal of Networked and Distributed Computing*, vol. 5, no. 3, p. 143, May 2017.

[16] R. A. Krohling and E. Mendel, "Bare bones particle swarm optimization with Gaussian or Cauchy jumps," in *Proc. 2009 IEEE Congress on Evolutionary Computation*, 2009, pp. 3285–3291.

[17] J. Guo and Y. Sato, "A bare bones particle swarm optimization algorithm with dynamic local search," in *Proc. International Conference on Swarm Intelligence*, 2017, pp. 158–165.

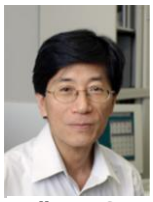
[18] T. Blackwell, "A study of collapse in bare bones particle swarm optimization," *IEEE Transactions on Evolutionary Computation*, vol. 16, no. 3, pp. 354–372, 2012.

[19] J. Guo and Y. Sato, "A fission-fusion hybrid bare bones particle swarm optimization algorithm for single-objective optimization problems," *Applied Intelligence*, vol. 49, no.10, pp. 3641–3651, 2019.

[20] M. Campos, R. A. Krohling, and I. Enriquez, "Bare bones particle swarm optimization with scale matrix adaptation," *IEEE Transactions on Cybernetics*, vol. 44, no. 9, pp. 1567–1578, 2014.

[21] J. Guo and Y. Sato, "A dynamic allocation bare bones particle swarm optimization algorithm and its application," *Artificial Life and Robotics*, vol. 23, no. 3, pp. 353–358, 2018.

Jia Guo received the M.E. degree from the Huazhong University of Science and Technology, China, in 2016. He is currently pursuing a Ph.D. degree with Hosei University, Japan. His research interests lie in the evolutionary computation and its applications, including swarm optimization algorithms. He received the Young Author Award from the International Symposium on Artificial Life and Robotics in 2017.



Yuji Sato received the BE and PhD degrees in engineering from the University of Tokyo, Japan in 1981 and 1997, respectively. From 1981 to 2000, he was with Hitachi Ltd, Tokyo, Japan. In April 2000, he joined the Faculty of Computer and Information Sciences at the Hosei University, Japan, as an associate professor, and became a professor in April 2001. He received the 2015 Highly Commended Paper Award of International Journal of Intelligent Computing and Cybernetics. His current research areas include parallel and distributed evolutionary multi-objective optimization on many-core architecture and evolution of machine learning techniques in design.