Hybrid Gravitational Search Algorithm with Random-key Encoding Scheme Combined with Simulated Annealing

Huiqin Chen†, Sheng Li and Zheng Tang††,

Graduate School of Innovative Life Science, University of Toyama, Toyama-shi, Japan

Summary
This paper is devoted to the presentation of a novel hybrid method by combining gravitational search algorithm (GSA) with simulated annealing (SA) method. In GSA, the representation of the problem on hand is based on the random-key encoding scheme. While GSA is employed as a global search algorithm, a multi-type local improvement scheme is incorporated into it, performing as a local search operator. Furthermore, SA is utilized to manipulate the iteration progress algorithmically. The resultant proposed hybrid random-key gravitational search algorithm (Hr-GSA) is tested on the famous traveling salesman problem. The experimental results show that Hr-GSA is more robust and efficient than other seven traditional population based algorithms, such as genetic algorithm, particle swarm optimization, artificial immune system, and so on.

Key words:
Gravitational search, simulated annealing, local improvement, population-based algorithm, hybridization.

1. Introduction

Population-based optimization algorithms are the techniques which are in the set of the nature based optimization algorithms. The creatures and natural systems which are working and developing in nature are one of the interesting and valuable sources of inspiration for designing and inventing new systems and algorithms in different fields of science and technology. Evolutionary Computation [1], Neural Networks [2], Time Adaptive Self-Organizing Maps [3], Ant Systems [4], Particle Swarm Optimization [5], Simulated Annealing [6], Bee Colony Optimization [7] and DNA Computing [8] are among the problem solving techniques inspired from observing nature. These algorithms have been used to solve different optimization problems, and have received promising results. Moreover, some algorithms give a better solution for some particular problems than others. Nevertheless, there is no specific algorithm to achieve the best solution for all optimization problems. As a result, searching for new heuristic optimization algorithms is still a challenging problem [9].

Among these heuristic optimization problems, Gravitational search algorithm (GSA) [10] is a global search algorithm appropriate for problems with huge search spaces. It is a novel population-based optimization approach based on the law of gravity. In GSA, the individuals are a collection of masses which interact with each other based on the Newtonian gravity and the laws of motion. A population of candidate solutions is modeled as a swarm of objects. At each iteration time, the objects update their position (and solution) by moving stochastically towards regions previously visited by the other objects. The object with heavier mass has a larger effective attraction radius and hence a greater intensity of attraction. By lapse of time, the objects tend to move towards the heaviest object. The simplicity, robustness, and adaptability of GSA, enable it to have applications in a wide-range of function optimization problems; and it has been shown that the global search ability of GSA is superior to that of other famous algorithms (such as the particle swarm optimization (PSO) [5]) in most cases.

The original developed GSA is designed to search solution in a continuous space. For the purpose of applying it to the traveling salesman problem (TSP), which actually is within a discrete state space, a random-key encoding technique is used. Furthermore, because GSA's local search ability is weaker than global searching ability, in order to get better solution, some local search schemes should be integrated with the GSA. In this paper, we embedded a multi-type local improvement scheme based on the simulated annealing technique into GSA. The resultant algorithm (Hr-GSA) enhances the object's searching ability and is suitable to solve the TSP. The experimental results show that the proposed algorithm Hr-GSA with multi-type local improvement scheme outperforms the original GSA and is more efficient than those of existing meta-heuristics methods such as neural networks, particle swarm optimization, genetic algorithm, and artificial immune system for TSP, respectively.

The remainder of this paper is organized as follows: a brief introduction of the GSA is given in the following section. Section 3 gives a general description of the traveling salesman problems involving its mathematics representation. Section 4 provides the details of the proposed Hr-GSA and apply it on the TSP. Section 5 discusses the experimental results. Finally, some remarks

Manuscript received May 5, 2011
Manuscript revised May 25, 2011

and conclusions are summarized.

2. Gravitational search algorithm

Gravitational search algorithm (GSA) is a recently proposed method used on optimization problem [10, 11]. It has been compared with some well-known heuristic optimization methods exiting, and the obtained results showed the high performance of the method. The GSA is constructed on the law of Newtonian Gravity: "Every particle in the universe attracts every other particle with a force that is directly proportional to the product of their masses and inversely proportional to the square of the distance between them". In the algorithm, all the individuals can be viewed as objects with masses. The objects attract each other by the gravity force, and the force makes all of them move towards the ones with heavier masses. The objects transform information by the gravitational force, and the objects with heavier masses become heavier. Until now, GSA has received more and more attentions and has been applied to multiple-objective optimization problem [12], data clustering [13], static var compensator allocation [14], parameters identification of hydraulic governing system [15], and so on.

Each agent in GSA has four specifications: position, inertial mass, active gravitational mass, and passive gravitational mass. The position of agent corresponds to a solution of the optimization problem at hand. Moving the position of agent can result in an improvement of the solution's quality. From a view of optimization context, GSA approach can be regarded as a population-based algorithm that performs a parallel search on the space of solutions. Several solutions of a given problem constitute a population (the swarm). Each solution is seen as an object. All objects attract each other by a gravity force, and this force causes a movement of all objects globally towards the objects with heavier masses. The heavy masses correspond to good solutions of the problem. These objects search the problem's solution space by balancing the intensification and the diversification efforts. By lapse of time, the objects will be attracted by the heaviest object, which presents an optimum solution in the search space. The process iterates until a stopping condition is fulfilled. Different from other population-based algorithms, especially the famous PSO algorithm, several characteristics of GSA can be remarked: (1) GSA is a memory-less algorithm, indicating that only a smaller memory capability of hardware is required during implementation. (2) The movement direction of an agent is calculated based on the overall force obtained from its surrounding agents. (3) In GSA, the force is proportional to the fitness value but reversely proportional to the distance between solutions, in such a way heavy masses have large effective attraction radius and great intensities of attraction, thus inferring that the agents always tend to move towards the best agent.

The GSA algorithm can be described as follows:
First assuming there are *N* objects and each of them has *m* dimensions, we define the *i*-th object by:

$$X_i = (x_i^1, \ldots, x_i^d, \ldots, x_i^m) \qquad (1)$$

According Newton gravitation theory, the force acting on the ***i***-th mass from the ***j***-th mass is defined as:

$$F_{ij}^d(t) = G(t)\frac{M_i(t) \times M_j(t)}{\left\| X_i(t), X_j(t) \right\|_2}(x_i^d(t) - x_j^d(t)) \qquad (2)$$

where M_i and M_j are masses of agents, ***G(t)*** is the gravitational constant at time ***t***. M_i is calculated through comparison of fitness:

$$M_i = \frac{fit_i(t) - worst(t)}{best(t) - worst(t)} \qquad (3)$$

Here, ***best(t)*** and ***worst(t)*** are the best and worst fitness of all agents, respectively. For the ***i***-th agent, the randomly weighted sum of the forces exerted from other agents:

$$F_i^d(t) = \sum_{j \neq i} rand_j F_{ij}^d(t) \qquad (4)$$

Based on the law of motion, the acceleration of the ***i***-th agent is calculated by:

$$a_i^d(t) = \frac{F_i^d(t)}{M_i(t)} \qquad (5)$$

Then, the searching strategy on this concept can be described by following equations.

$$V_i^d(t+1) = rand_i \times v_i^d(t) + a_i^d(t) \qquad (6)$$

$$x_i^d(t+1) = x_i^d(t) + v_i^d(t+1) \qquad (7)$$

where x_i^d represents the position of the ***i***-th agent in ***d***-th dimension, v_i^d is the velocity, a_i^d is the acceleration.
It is worth pointing out that the gravitational constant ***G(t)*** is important in determining the performance of GSA, it is defined as a function of time ***t***:

$$G(t) = G_0 \cdot \exp(-\beta \cdot \frac{t}{\max_t}) \qquad (8)$$

where G_0 is the initial value, β is a constant, *t* is the current iteration number, and $\max_t$ is the maximum number of iterations.

3. Traveling Salesman Problem

The Travelling Salesman Problem (TSP) is a representative of a large class of problems known as combinatorial optimization problems. In the ordinary form of the TSP, a map of cities is given to the salesman and he has to visit all the cities only once to complete a tour such that the length of the tour is the shortest among all possible tours for this map. The data consist of weights assigned to the edges of a finite complete graph, and the objective is to find a Hamiltonian cycle, a cycle passing through all the vertices, of the graph while having the minimum total weight. In the TSP context, Hamiltonian cycles are commonly called tours.

In general, the TSP includes two different kinds, the symmetric TSP and the asymmetric TSP. In the symmetric form known as STSP there is only one way between two adjacent cities, i.e., the distance between cities *A* and *B* is equal to the distance between cities *B* and *A*. But in the ATSP (asymmetric TSP) there is not such symmetry and it is possible to have two different costs or distances between two cities. Hence, the number of tours in the ATSP and STSP on *n* vertices (cities) is *(n-1)!* and *(n-1)!/2*, respectively. Please note that the graphs which represent these TSPs are complete graphs. In this paper we mostly consider the STSP. It is known that the TSP is an NP-hard problem [16] and is often used for testing the optimization algorithms. Finding Hamiltonian cycles or traveling salesman tours is possible using a simple dynamic program using time and space $O(2^n n^{O(1)})$, that finds Hamiltonian paths with specified endpoints for each induced subgraph of the input graph [17]. The TSP has many applications in different engineering and optimization problems. The TSP is a useful problem in routing problems e.g. in a transportation system.

There are different approaches for solving the TSP. Solving the TSP was an interesting problem during recent decades. Almost every new approach for solving engineering and optimization problems has been tested on the TSP as a general test bench. First steps in solving the TSP were classical methods. These methods consist of heuristic and exact methods. Heuristic methods like cutting planes and branch and bound [18], can only optimally solve small problems whereas the heuristic methods, such as 2-opt [19], 3-opt, Markov chain [20], simulated annealing [6] and tabu search [21] are good for large problems. Besides, some algorithms based on greedy principles such as nearest neighborhood, and spanning tree can be introduced as efficient solving methods. Nevertheless, classical methods for solving the TSP usually result in exponential computational complexities. Hence, new methods are required to overcome this shortcoming, population based optimization algorithms just meet this need.

The followings give a mathematical description for TSP. Let $K_n = (V_n, E_n)$ be the complete undirected graph with $n = |V_n|$ nodes and $m = |E_n| = \binom{n}{2}$ edges. An edge *e* with endpoints *i* and *j* is also denoted by *ij*, or by *(i,j)*. We denote by $\Re^{E_n}$ the space of real vectors whose components are indexed by the elements of E_n. The component of any vector $z \in \Re^{E_n}$ indexed by the edge $e = ij$ is denoted by z_e, z_{ij} or *z(i,j)*. Given an objective function $c \in \Re^{E_n}$, that associated a length c_e with every edge *e* of K_n, the symmetric traveling salesman problem consists of finding a Hamiltonian cycle (a cycle visiting every node exactly once) such that its *c*-length (the sum of the lengths of its edges) is as small (large) as possible. Without loss of generality, we only consider the minimization version of the problem. From now on we use the abbreviation TSP only for the symmetric traveling salesman problem, i.e. STSP. Of special interest are the Euclidean instances of the traveling salesman problem. In these instances the nodes defining the problem correspond to points in the 2-dimensional plane and the distance between two nodes is the Euclidean distance between their corresponding points. More generally, instances that satisfy the triangle inequality, i.e., $c_{ij} + c_{jk} \geq c_{ik}$ for all three distinct *i* ,*j* and *k*, are of particular interest. In this paper, when applying GSA on TSP, we only consider Euclidean instances.

4. Hr-GSA for TSP

Before actually introducing the hybrid random-key encoding gravitational search algorithm (Hr-GSA), some issues are in applying GSA to solve TSP. The original GSA design is developed to solve continuous function. However, TSP is a combinatorial problem, the solution space is discrete.

(1) The first issue is to find a suitable representation which the particles of GSA can simulate an operation permutation schedule of TSP. In this paper, based on an operation permutation, the continuous GSA combined with a random-key (RK) encoding scheme is used to solve the first issue. The detailed description will be discussed in Section 4.1.
(2) The second issue is how to enhance GSA's local search ability by applying GSA to solve the combinatorial problems. No matter applying continuous GSA [10] or binary GSA [11] or discrete GSA to the combinatorial problem, embedding the local search ability in GSA algorithm is an effective way to get a better solution. A multi-type local improvement scheme based on simulated annealing

algorithm (SA) is applied to enhance the local search ability of GSA. The detailed description is stated in Section 4.3. The complete algorithm named Hr-GSA is shown in Section 4.5, which consists of random-key (RK) encoding scheme, multi-type local improvement scheme based on SA and gravitational search algorithm.

4.1 Representation of a position for TSP

The search space is created in a $1 \times N$ dimensions space for N cities TSP problem. The position of a mass in GSA is represented by the $1 \times N$ vector *V* (where *N* denotes the number of cities), such that $V_i = k$ if city *k* is in position *i* in the tour. Then, a solution $V = (v_1, \ldots, v_k, \ldots, v_N)$ represents that the first city to be visited is the value of v_1 and the *k*-th city to be visited is the value of v_k. The last city to be visited before going back to the city v_1 is the city v_N.

4.2 Random-key encoding scheme

The random-key encoding scheme can be used to transform a position in a continuous space to a discrete space. A vector in the random-key space consists of real numbers. According to random-key scheme, one mass represented by real numbers can simulate an operation permutation that consists of discrete numbers.

In the random key method, we assign each position of a mass a random number drawn uniformly from *[0, 1)*. To decode the position from the random-key space into the real solution space, we visit the nodes in ascending order of their value of each dimension. An example can be illustrated as shown in Fig. 1.

Random key:	0.42	0.06	0.38	0.48	0.81
Decodes as:	3	1	2	4	5

(Ties are broken arbitrarily.)

Fig. 1: An example of the random-key encoding scheme.

Nodes that should be early in the tour tend to evolve the value of dimension closer to 0 and those that should come later tend to evolve genes closer to 1. By doing so, the position of an object in GSA, $X_i = (x_i^1, \ldots, x_i^d, \ldots, x_i^m)$ can be transformed to a valid solution in TSP search space, i.e. $V = (v_1, \ldots, v_k, \ldots, v_N)$.

4.3 Multi-type local improvement scheme

Since the GSA is a global search algorithm, its local search ability is weak. For the purpose of enhancing the exploitation capability of GSA and get a better solution, in this paper, we developed a new multi-type local improvement scheme for TSP. The multi-type local improvement scheme is composed of swapping operation, insertion operation, inversion operation and long-distance movement operation which can be used to search an individual's neighborhood to get a better solution.

The following illustrates the details of the multi-type local improvement scheme. Swapping operation scheme is to swap two weighting numbers that indirectly represent two operations in the *p*-th and *q*-th dimension ($p \neq q$) of an individual in the random-key virtual space. Insertion operation is to remove the one in the *p*-th dimension and insert it into the *q*-th dimension ($p \neq q$) of an individual. In general, it is enough to get a better solution for most problems by using these two types of enhancement scheme. By the experimental experience, it needs a scheme to jump away from the local optimal for some hard problems which have higher dimensions. So, we incorporated another two types of enhancement scheme to the proposed algorithm. The inversion operation scheme is to pick two dimensions *p* and *q* ($p \neq q$) first and invert the weighting numbers between them. The last enhancement scheme is the long distance movement operation. At the beginning, pick two dimensions *p* and *q* ($p \neq q$) of an individual, remove all weighting numbers between them and insert these removed weighting numbers to the place where it begins at the *r*-th dimension.

The progress of multi-type local improvement scheme is to select an operation scheme from multi-type local improvement scheme, to operate on an object (mass), and to compare the fitness obtained before the selected scheme and that obtained after the selected scheme. If the latter is better than the former, update the real vector of the individual by the selected operation scheme. If not, the new real vector can be accepted and updated according to a threshold that is generated by the simulated annealing algorithm (SA). If a random probability is less than a threshold, the new real vector can be accepted and updated; otherwise, drop the new real vector and keep the previous real vector as a next position to carry out the local search operation. After finishing one scheme, continue to select another scheme to operate on the individual until it meets the stop criterion.

Figs. 2-5 give an example of the usage of the multi-type local improvement scheme in order to explain this scheme more clearly. In Fig. 2, the individual (0.71, 0.56, 0.08, 0.13, 0.92, 0.45) is obtained by swapping two items located at the second dimension and the fourth dimension from the individual (0.71, 0.13, 0.08, 0.56, 0.92, 0.45). According to the RK encoding scheme, the individual can be transformed into a permutation solution in TSP. If the fitness of the latter solution is better than that of the former solution, the position of the mass is updated from (5, 2, 1, 4, 6, 3) to (5, 4, 1, 2, 6, 3).

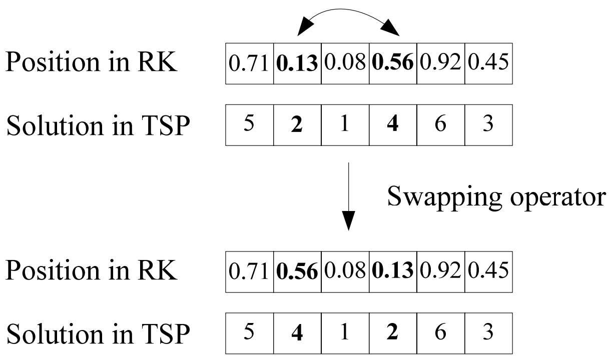

Fig. 2: Swapping operator scheme, p=2, q=4.

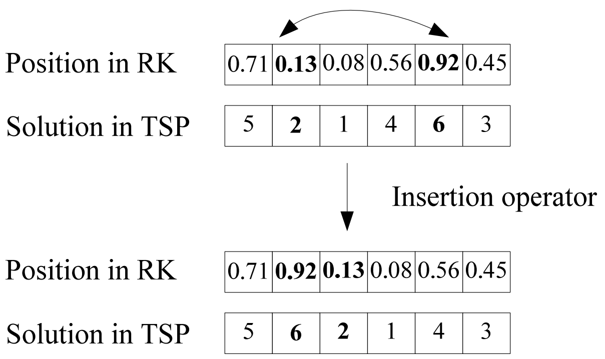

Fig. 3: Insertion operator scheme, p=2, q=5.

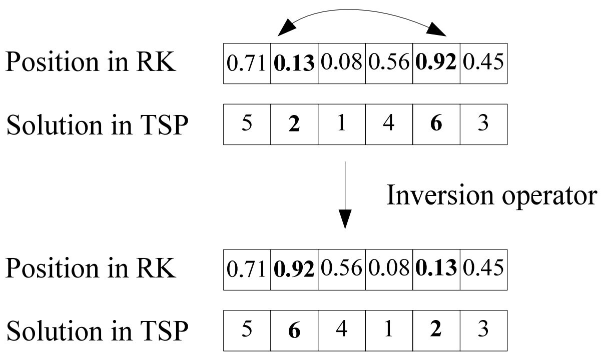

Fig.4: Inversion operator scheme, p=2, q=5.

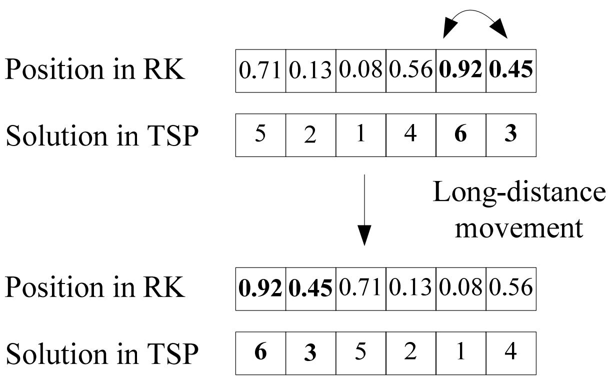

Fig. 5: Long-distance movement operator scheme, p=5, q=6, r=1.

Fig. 2 illustrates the insertion operator scheme by inserting the fifth dimension into the second dimension, that is, the individual (0.71, 0.13, 0.08, 0.56, 0.92, 0.45) will be varied to (0.71, 0.92, 0.13, 0.08, 0.56, 0.45). As a result, the fitness comparison is implemented, determining whether the update process carries out. Similar descriptions can be made on Figs. 3-5. In general, the four operators involving the swapping operator, insertion operator, inversion operator, and long-distance movement operator work together to act as the local improvement scheme.

Algorithm 1 depicts a partial algorithm concerning four type operations. $Prob_s$ means the probability of executing the swapping scheme; $Prob_i$ means the probability of executing the insertion scheme; $Prob_{inv}$ means the probability of executing the inversion scheme; $Prob_{long}$ means the probability of executing the long distance movement scheme, respectively.

Algorithm 1: the operation of multi-type local improvement scheme.

```
**Input**: p, the individual to be enhanced
**Output**: p', one individual after executing multi-type Local improvement
1: q= rand()
2: **If** ($0<q<Prob_s$) then execute swapping scheme for individual p
3: **Else if** ($Prob_s<q<Prob_s+Prob_i$) then execute inserting scheme for individual p
4: **Else if** ($Prob_s+Prob_i<q<Prob_s+Prob_i+Prob_{inv}$) then execute inversion scheme for individual p
5: Finally, (q will match with $Prob_{long}$)
       **Else** execute long distance movement scheme for individual p
6: **End if**
```

For instance, suppose that $Prob_s=0.4$, $Prob_i=0.4$, $Prob_{inv}=0.1$ and $Prob_{long}=0.1$, if rand()=0.33, the individual *p* will be enhanced by swapping scheme; if rand()=0.66, the individual *p* will be enhanced by inserting scheme; if rand()=0.88, the individual *p* will be enhanced by inversion scheme; if rand()=0.98, the individual *p* will be enhanced by long distance movement scheme.

4.4 Simulated annealing algorithm

When the number of feasible pedigrees is too large for exhaustive enumeration, an alternative is the use of simulated annealing, an optimization tool which has proven effective in a large variety of combinatorial optimization problems [6].

Suppose we wish to maximize a function f on a state space *X*. We construct a Monte Carlo Markov chain (MCMC) on *X* as follows. For each $x \in X$ there is a neighborhood of states, assumed to be a constant size *N*. A transition from state x_i to x_{i+1} is defined by first selecting at random a proposal state x_{i+1}' from the neighborhood of x_i. Then x_{i+1}' is accepted as the subsequent state with probability

$$\alpha_c(x'_{i+1}, x_i) = \begin{cases} 1 & if \quad f(x'_{i+1}) \ge f(x_i) \\ \exp\left(\frac{f(x'_{i+1}) - f(x_i)}{c}\right) & if \quad f(x'_{i+1}) < f(x_i) \end{cases} \qquad (9)$$

where the constant *c* is referred to as the temperature and, in a simulated annealing algorithm, is allowed to decrease to zero. If the proposal state is accepted we set $x_{i+1}= x_{i+1}'$; otherwise $x_{i+1}= x_i$. This acceptance rule is known as the Metropolis criterion.

Algorithm 2: Multi-type local improvement scheme manipulated by simulated annealing algorithm.

```
Input: P, the individual (object) to be enhanced; a starting temperature T;
a final temperature T_f ; a cooling rate w
Output: an enhanced individual
1: Fitness(p)= Fitness of a solution position represented by p
2: While (T > T_f ) do
3:   Randomly select an operation from the multi-type local improvement
scheme by Algorithm 1,
and generate a new individual p' by the selected operation.
4:   Fitness(p')=Fitness of a solution position represented by p'
5:   Delta=Fitness(p') – Fitness(p)
6:   If (Delta > 0) then //p' is worse than p
       // randomly generate a probability rand() to accept the worse p' with
a probability
7:   If (R=rand() < min {1,exp^{-Delta/T}} then
8:     p= p' //update P to be a enhanced p'
9:     Fitness(p)=Fitness(p')
10:  End if
11: Else
12:  p= p' // to accept a better p'
13:  Fitness(p)=Fitness(p')
14:  T=w*T
15: End if
16: End while
```

Algorithm 3: The implementation progress of Hr-GSA.

```
Input: Prob_total, the probability to execute the multi-type local
improvement scheme; set user-defined parameters
Output: Global best solution
1: Initialize the position and velocity for all objects of a population
2: While the stop condition is not met do
3:   For each object do
4:     If (S=Rand()<Prob_total ) then
5:       Execute the multi-type local improvement scheme shown in
Algorithm 2 for the masses
6:     End if
7:     Compute the Forces and acceleration according to Eqs. 2-5
8:   End for
9:   Update the global best of the population
10:  Update the Positions and velocities of masses according to Eq. 6 and
Eq. 7
11: Compute the gravitational constant based on Eq. 8
13: End for
14: End while
```

By SA algorithm, we can decide whether to accept an individual (object) that is enhanced by Algorithm 1 but its fitness is not better than the individual not being enhanced by Algorithm 1 or not. For an enhanced individual that did not make improvement for fitness, if one random probability is smaller than $\min\left\{1, \exp^{\frac{f(x'_{i+1})-f(x_i)}{c}}\right\}$, the individual's position can be accepted as a new position of the individual; otherwise, we drop the position and keep the previous position for the individual. A complete multi-type local improvement scheme based on simulated annealing algorithm (SA) is shown in Algorithm 2.

4.5 The Hr-GSA algorithm

Based on the above schemes, in this paper, we integrate the random-key encoding scheme, multi-type local improvement scheme into gravitational search algorithm, named it as Hr-GSA. In order to test the effectiveness of Hr-GSA, the famous traveling salesman problems are utilized as benchmark instances. In Hr-GSA, a position of a mass in GSA is represented by a real vector as shown in Fig. 1. Every mass moves its position in the random-key virtual space by Eq. 6 and Eq.7, and the objective function of one position corresponding to the solution space of TSP can be evaluated by the transformation from random-key space to a solution space of TSP. For increasing the local search capability of GSA, multi-type local improvement scheme is used as an effective way to search the local neighborhood of one position in the solution space of TSP. The random-key encoding scheme provides a search space for the continuous gravitational search algorithm (GSA) and an easy way to encode the representation of GSA. According to the random-key encoding scheme, we enhance the position by multi-type local improvement scheme that is corresponding to make a local search for the mass. One mass is selected with a probability $Prob_{\text{total}}$ as an individual to be enhanced in the multi-type local improvement algorithm. After multi-type local improvement algorithm, the selected particles can be in a better position than the previous one. Then, each mass of the population moves to a new position according to Eq. 6 and Eq. 7. The process of multi-type local improvement scheme and GSA is executed until it gets the optimal solution or the maximum iteration number is reached. The total implementation progress of Hr-GSA can be referred to as in Algorithm 3.

5. Simulation Results

In order to verify the performance of the proposed Hr-GSA, we use five benchmark instances taken from the TSPLIB [22] to be the test suit. Seven typical bio-inspired population-based algorithms are used to make a comparison involving the genetic algorithm (GA), ant colony optimization (ACO), particle swarm optimization (PSO), artificial immune system (AIS), intelligent water drops algorithm (IWDA), bee colony optimization algorithm (BCO) and finally electromagnetism-like mechanism (EM).

First, we summarize the descriptions of the above seven algorithms one by one except the proposed Hr-GSA. GA was introduced by Holland in the 1970s [23]. These algorithms are adaptive search techniques based on the mechanisms of natural selection and the survival of the fittest concept of biological evolution. By simulating biological evolution, GAs can solve searching problem domains effectively and easily apply to many of the current engineering problems. GAs have been widely used in many applications of TSP and its extensions throughout the literature.

Ant Colony Optimization (ACO), first proposed by M. Dorigo et al. [24][25], is a population-based, general-purpose heuristic approach to combinational optimization problems. The earliest ACO algorithm, Ant System (AS), was applied to the TSP (mainly because the TSP is “a shortest path problem to which the ant colony metaphor is easily adapted and that it is a didactic problem”. After that, most improved ACO algorithms also used the TSP as a test problem and the result is promising.

The particle swarm optimization (PSO) was originally presented by Kennedy and Eberhart in 1995 [5]. It is an algorithm based stochastic optimization technique which inspired by social behavior among individuals. In the PSO system, individuals (we call them particles) move around a multidimensional search space. Each particle represents a potential solution of the problem, and can remember the best position (so1ution) it has reached. All the particles can share their information about the search space, so there is a global best solution.

Like the natural immune systems the AIS is a set of techniques, which try to algorithmically mimic natural immune systems’ behavior [26]. The immune system is susceptible to all of the invaders, also the outer influences, like vaccines which are artificial ways of raising individual's immunity. The first work in investigating potential application of the immune system in solving numerical optimization problems was the study by Bersini and Varela [27]. After that, many studies have been performed that focus on the AIS, also its application on TSP.

Based on the observation on the behavior of water drops, an artificial water drop algorithm (IWDA) [28] possesses some of the remarkable properties of the natural water drop. This Intelligent Water Drop, IWD for short, has two important properties: one is the amount of the soil it carries now, soil and the other is the velocity that it is moving now, velocity, flows in its environment. This environment depends on the problem at hand. In an environment, there are usually lots of paths from a given source to a desired destination, which the position of the destination may be known or unknown. If we know the position of the destination, the goal is to find the best (often the shortest) path from the source to the destination. In some cases, in which the destination is unknown, the goal is to find the optimum destination in terms of cost or any suitable measure for the problem.

The bee colony optimization algorithm (BCO) [29] according to nature is as follows. At first, each bee belonging to a colony looks for the feed individually. When a bee finds the feed, it informs other bees by dancing. Other bees collect and carry the feed to the hive. After relinquishing the feed to the hive, the bee can take three different actions. With a certain probability that is dependent on the obtained feed quality, its distance from the hive and the number of the bees which are now engaged with this feed resource, a bee selects one of the stated actions and follows its work in a similar repetitive form.

The Electromagnetism-like mechanism (EM) is a heuristic that was introduced by Birbil and Fang [30]. The method utilizes an attraction-repulsion mechanism to move the sample points towards the optimality. In other words, EM simulates the attraction-repulsion mechanism of electromagnetism theory which is based on Coulomb’s law. The main concentration of the first introduction of this heuristic was on the problems with bounded variables. One of the most attractive approaches for solving TSP using EM is cited in [31].

Second, the performances of the selected eight algorithms were put together to make a comparison. Table 1 summarized the simulation results on the five TSP benchmark instances taken from TSPLIB. All these instances belong to the Euclidean distance type. In Table 1, Opt. denotes the known-so-far optimal solution quality, and the other results recorded were the ratio of the solutions found by each algorithm to the optimal solution over 30 runs. In TSP which is a minimum optimization problem, it was clear that the smaller the values, the better quality of the solution. Furthermore, in the seventh column of Table 1, the recorded values represented the average quality of each algorithm over the five instances. Besides, Fig. 6 depicted the comparison illustration of the simulation results in Table 1.

From Table 1 and Fig. 6, we can easily find that the proposed algorithm Hr-GSA performed better than other seven traditional population-based algorithms on all the tested benchmark instances, which also gave implications that Hr-GSA has potential applications on other combinatorial optimization problems. For a more clearly description, we also gave two final solutions of TSP found by Hr-GSA in Fig. 7, where the left side of the figure is for eil56, and the right one is for pr124, both of them showing well performances.

6. Conclusion

In this paper, we proposed a hybrid gravitational search algorithm (Hr-GSA) by incorporating the random-key encoding scheme, the multi-type local improvement scheme, and the simulated annealing algorithm. The random-key encoding scheme was utilized to transform the continuous search space in the original GSA into a discrete solution space, especially for discrete engineering problems. As the GSA is typically a population-based global search algorithm, the multi-type local improvement scheme was employed to enhance the exploitation capability for GSA. Furthermore, the simulated annealing algorithm was carried out to manipulate the search strategy of exploitation and exploration.

The resultant hybrid algorithm Hr-GSA was tested on several TSP benchmark instances and compared with other seven traditional population-based algorithms, such as the genetic algorithm, the particle swarm optimization. The simulation results indicated that the proposed Hr-GSA performed competitive solutions than the other algorithms, thus implying that it not only can be used to solve TSP, but also other discrete combinatorial optimization problem, for instance, the job shop scheduling problem.

Table 1: Simulation results of the eight algorithms on five TSP benchmark instances.

	eil51	st70	rd100	pr124	rat195	average
Opt.	426	675	7910	59030	2323	
GA	2.58%	2.35%	5.27%	2.74%	6.68%	3.92%
ACO	1.08%	1.98%	3.14%	1.23%	2.59%	2.00%
PSO	1.12%	2.32%	2.65%	1.98%	3.45%	2.30%
AIS	1.22%	1.79%	2.03%	1.45%	2.77%	1.85%
IWDA	4.08%	5.20%	4.97%	6.12%	5.34%	5.14%
BCO	2.19%	3.01%	2.44%	2.78%	3.43%	2.77%
EM	2.67%	3.05%	2.78%	3.45%	5.45%	3.48%
Hr-GSA	0.54%	0.34%	1.12%	1.05%	2.56%	**1.12%**

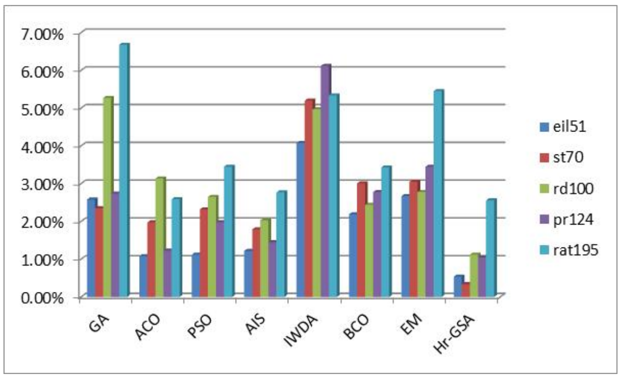

Fig. 6: The illustration of the simulation results.

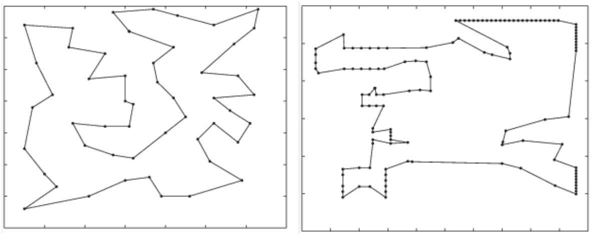
Fig. 7: Two typical optimal solutions found by Hr-GSA for TSP, the left side of the figure is for eil56, while the right one is for pr124.

References

[1] A. E. Eiben and J. E. Smith, Introduction to Evolutionary Computing. Springer-Verlag, 2003.

[2] S. Haykin, Neural Networks, Prentice-Hall, second edition, 1999.

[3] H. Shah-Hosseini, "The time adaptive self-organizing map is a neural network based on Artificial Immune System," In

Proc. IEEE World Congress on Computational Intelligence, Vancouver, Canada, July 2006, pp.1007-1014, 2006.
[4] M. Dorigo and T. Stutzle, Ant Colony Optimization, Prentice Hall, 2004.
[5] J. Kennedy and R. Eberhart, “Particle Swarm Optimization,” Proceedings of IEEE International Conference on Neural Networks, pp. 1942-1948, 1995.
[6] S. Kirkpatrick, C. D. Gelatt and M. P. Vechi, “Optimization by simulated annealing,” Science, vol. 220, no.4598, pp.671-680, 1983.
[7] D. Teodorovic and M. Dell’Orco, “Bee colony optimization: A cooperative learning approach to complex transportation problems,” Advanced OR and AI Methods in Transportation, pp. 51-60, 2005.
[8] L. M. Adleman, “Molecular computation of solutions to combinatorial problem,” Science, pp. 1021–1023, 1994.
[9] D. H. Wolpert and W. G. Macready, “No free lunch theorems for optimization,” IEEE Transactions on Evolutionary Computation, vol.1, pp.67–82, 1997.
[10] E. Rashedi, H. Nezamabadi-pour and S. Saryazd, “GSA: A Gravitational Search Algorithm,” Information Sciences, vol.179, no.13, pp. 2232-2248, 2009.
[11] E. Rashedi, H. Nezamabadi-pour and S. Saryazdi, “BGSA: binary gravitational search algorithm,” Natural Computing, vol.9, no.3, pp.727-745, 2010.
[12] H. R. Hassanzadeh and M. Rouhani, “A Multi-objective Gravitational Search Algorithm,” in Proc. 2010 Second International Conference on Computational Intelligence, Communication Systems and Networks, pp.7-12, 2010.
[13] M. Yin, Y. Hu, F. Yang, X. Li and W. Gu, "A novel hybrid K-harmonic means and gravitational search algorithm approach for clustering," Expert Systems with Applications, vol.38 pp.9319-9324, 2011.
[14] E. Rashedi, H. Nezamabadi-pour, S. Saryazdi and M. Farsangi, “Allocation of Static Var Compensator Using Gravitational Search Algorithm,” First Joint Congress on Fuzzy and Intelligent Systems Ferdowsi, University of Mashhad, Iran, pp.1-10, 2007.
[15] C. S. Li and J. Z. Zhou, "Parameters identification of hydraulic turbine governing system using improved gravitational search algorithm," Energy Conversion and Management, vol.52, no.1, pp.374-381, 2011.
[16] M. R. Garey and D. S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman, 1979.
[17] D. Eppstein, “TSP for Cubic Graphs,” Journal of Graph Algorithms and Applications (JGAA), vol.11, no.1, pp. 61–81, 2007.
[18] M. Padherg and R. Rinaldi, “Optimization of a 532-city symmetric travelling salesman problem by branch and cut,” Operations Research Letters, vol. 6, no.1, pp.1-7, 1987.
[19] S. Lin and B. Kernighan, “An effective heuristic algorithm for the traveling-salesman problem,” Operations Research, vol. 21, no. 2, pp. 498-516, 1973.
[20] O. Martin,S. Otto and E. Felten, “Large-step markov chains for the traveling salesman problem,” Complex Systems, vol. 5, no. 3, pp. 299-326, 1991.
[21] W. B. Carlton and J. W. Barnes, "Solving the traveling-salesman problem with time windows using tabu search," IIE transactions, vol.28, no.8, pp.617-629, 1996.
[22] G. Reinelt, “TSPLIB -a traveling salesman problem library,” ORSA Journal on Computing, vol.3, pp.376-384, 1991.
[23] S. Chatterjee, C. Carrera and L. A. Lynch, “Genetic algorithms and traveling salesman problems,” European Journal of Operational Research, vol.93, pp.490-510, 1996.
[24] M. Dorigo and L.M. Gambardella, “Ant colonies for the traveling salesman problem,” BioSystems, vol.43, pp. 73-81, 1997.
[25] 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.
[26] Dasgupta D. (Ed.), Artificial Immune Systems and Their Applications. Springer-Verlag. Berlin, 1999.
[27] H. Bersini and F. J. Varela, Workshop on Parallel Problem Solving from Nature. LNCS, Springer-Verlag 496 343, Proc. 1st, 1990.
[28] H. Shah-Hosseini, “Problem Solving by Intelligent Water Drops,” 2007 IEEE Congress on Evolutionary Computation (CEC 2007), pp. 3226-3231, 2007.
[29] D. Teodorovic and M. Dell Orco, “Bee colony optimization: A cooperative learning approach to complex transportation problems,” Advanced OR and AI Methods in Transportation, pp. 51-60, 2005.
[30] S. Birbil and Sh. Fang, “An Electromagnetism-like Mechanism for Global Optimization,” Journal of Global Optimization, vol. 25, pp.263-282, 2003.
[31] P. Wu, K. Yang and H. Fang, “A Revised EM-like Algorithm + K-OPT Method for Solving the Traveling Salesman Problem,” Proceedings of the First International Conference on Innovative Computing, Information and Control, 2006.

Huiqin Chen received the B.S. degree from HoHai University, Nanjing, China in 2006 and an M.S. degree from University of Toyama, Toyama, Japan in 2009. Now, she is working toward the D.E. degree at University of Toyama, Toyama, Japan. Her main research interests are intelligence computing, neural networks, swarm intelligent algorithms, and combinational optimization problems.

Sheng Li received the B.S. degree from Nanjing University of Science and Technology, Nanjing, China in 2006 and an M.S. degree from University of Toyama, Toyama, Japan in 2009. Now, he is working toward the D.E. degree at University of Toyama, Toyama, Japan. His main research interests are intelligence computing, neural networks, swarm intelligent algorithms, and combinational optimization problems.

Zheng Tang received the B.S. degree from Zhejiang University, Zhejiang, China in 1982 and an M.S. degree and a D.E. degree from Tshinghua University, Beijing, China in 1984 and 1988, respectively. From 1988 to 1989, he was an instructor in the institute of microelectronics at Tshinghua University. From 1990 to 1999, he was an associate professor in the department of electrical and electronic engineering, Miyazaki University, Miyazaki, Japan. In 2000, he joined Toyama University, Toyama, Japan, where he is currently a professor in the department of intellectual information systems.