International Journal of Control and Automation
Vol. 10, No. 2 (2017), pp.187-200
http//dx.doi.org/10.14257/ijca.2017.10.2.16

An Improved Particle Swarm Optimization Algorithm for Traveling Salesman Problems

Xuesong Yan[1], Qinghua Wu[2,3*], Yuanyuan Fan[1], Qingzhong Liang[1] and Chao Liu[1]

[1] School of Computer Science, China University of Geosciences, Wuhan, P. R China
[2] Hubei Provincial Key Laboratory of Intelligent Robot, Wuhan Institute of Technology, Wuhan, P. R China
[3] School of Computer Science and Engineering, Wuhan Institute of Technology, Wuhan, P. R China
** Corresponding author E-mail: wuqinghua@sina.com*

Abstract

Particle Swarm Optimization algorithm (PSO) is a meta-heuristic algorithm. It makes few or no assumptions about the problem being optimized, and can search a very large space of candidate solutions. However, it does not guarantee to find an optimal solution. In this paper with the guidance of the analysis of the advantages and disadvantages of the standard PSO, we propose a novel Particle Swarm Optimization algorithm, which introduces an extra mechanism for sharing information and a competition strategy. The proposed algorithm keeps not only the fast convergence speed characteristic of PSO, but effectively improves the capability of global searching as well. Our experimental results show it performs much better than the standard PSO on benchmark functions, especially for difficult functions. We also apply it to solve the traveling salesman problems (TSP). It significantly improves the success rate of finding the optimal solutions.

***Keywords**: particle swarm optimization, genetic algorithm, Travelling salesman problem, global optimal*

1. Introduction

The travelling salesman problem (TSP) [1] is one of the most widely studied NP-hard combinatorial optimization problems. Its statement is deceptively simple, and yet it remains one of the most challenging problems in Operational Research. The simple description of TSP is: Give a shortest path that covers all cities along. Let $G=(V;E)$ be a graph where V is a set of vertices and E is a set of edges. Let $C=(c_{ij})$ be a distance (or cost) matrix associated with E. The TSP requires determination of a minimum distance circuit (Hamiltonian circuit or cycle) passing through each vertex once and only once. The matrix C is said to satisfy the triangle inequality, if and only if $c_{ij}+c_{jk}\geq c_{ik}$ for all $i,j,k\in V$.

Due to its simple description and wide applications in real practices, such as path problems, routing problems and distribution problems, it has attracted researchers of various domains to work out its better solutions. Those traditional algorithms, such as the Cupidity Algorithm and the Dynamic programming Algorithm, are all facing the same obstacle, which is when the problem scale N reaches a certain degree, the so-called "Combination Explosion" will occur. For example, if N = 50, then it will take 5×10^{48} years under a super mainframe executing 100 million instructions per second to reach its approximate best solution.

A lot of algorithms have been proposed to solve TSP [2, 3, 4, 5, 6, 7]. Some of them (based on dynamic programming or branch and bound methods) provide the global optimum solutions. Other algorithms are heuristic, which are much faster, but they do not guarantee the optimal solutions. There are well known algorithms based on 2-opt or 3-opt change operators, Lin-Kerninghan algorithm (variable change) as well algorithms based on greedy principles (nearest neighbor, spanning tree, *etc*). The TSP was also approached by various modern heuristic methods, like simulated annealing, evolutionary algorithms and tabu search, even neural networks.

The Particle Swarm Optimization (PSO) algorithm was an intelligent technology first presented in 1995 by Eberhart and Kennedy. It was developed under the inspiration of behavior laws of bird flocks, fish schools and human communities [8]. If we compare PSO with Genetic Algorithms (GAs), we can find that they are all maneuvered on the basis of population operated. However, PSO doesn't rely on genetic operators, *i.e.*, selection, crossover and mutation operators, which manipulate individual chromosomes. Instead, PSO optimizes the population through information exchange among individuals. It achieves its optimum solutions by starting from a group of random solutions and then searching the optimum ones repeatedly. Once PSO was published, it attracted widespread attentions among scholars in the optimization fields. Shortly afterwards it had become a studying focus for several years. A number of scientific achievements had emerged into these fields [9-14].

PSO was proved to be a high efficient optimization algorithm by numerous research and experiments. PSO is a meta-heuristic algorithm. It makes few or no assumptions about the problem being optimized, and can search a very large space of candidate solutions. However, meta-heuristics algorithms, like PSO, do not guarantee to find an optimal solution. More specifically, PSO does not use the gradient of the problem being optimized. That is, PSO does not require that the optimization problem be differentiable, which is required by classic optimization methods, such as gradient descent and quasi-Newton methods. PSO can therefore be used on optimization problems that are partially irregular, noisy, change over time, *etc*. However, the standard PSO is easily trapped into a local optimum. Thus, it is needed to improve PSO to avoid this shortcoming. This paper proposes an improved PSO algorithm. Its implementation is as simple as the standard PSO. However, our experimental results show that proposed algorithm is more efficient in searching for global optima.

2. Related Work

In recent years, since the TSP is a good ground for testing optimization techniques, many researchers in various fields such as artificial intelligence, biology, mathematics, physics, and operations research devote themselves to trying to find the efficient methods for solving the TSP, such as genetic algorithms (GAs) [15], ant colony optimization (ACO) [16], simulated annealing (SA) [17], neural networks (NN) [18], particle swarm optimization (PSO) [19], evolutionary algorithms (EA) [20], memetic computing [21], *etc*. Besides, there are many practical applications of the TSP in the real world [22, 23], such as data association, vehicle routing (with the additional constraints of vehicle’s route, such as capacity’s vehicles), data transmission in computer networks, job scheduling, DNA sequencing, drilling of printed circuits boards, clustering of data arrays, image processing and pattern recognition, analysis of the structure of crystals, transportation and logistics.

Swarm intelligence is an important research topic based on the collective behavior of decentralized and self-organized systems in computational intelligence. It consists of a population which simulates the animals’ behavior in the real world. Now there are many swarm intelligence optimization algorithms, such as genetic algorithms, particle swarm optimization, ant colony optimization, bee colony algorithm, differential evolution, fishwarm algorithm, *etc*. Due to the simple concept, easy implementation and quick convergence, PSO has gained much attention and been successfully applied in a variety of fields mainly for optimization problems.

So far, as for the constrained optimization problems, relatively less work based on PSO can be found than those based on other EAs. Parsopoulos and Vrahatis [24] proposed a non-stationary multi-stage assignment penalty function method to transform the constrained problem to the unconstrained problem. Simulation results showed that PSO outperformed other EAs, but the design of the multi-stage assignment penalty function is too complex. In the work of Hu and Eberhart [25], the initial swarm contains only feasible solutions and a strategy to preserve feasibility is employed. Motivated by multi-objective optimization techniques, Ray and Liew [26] proposed a swarm algorithm with a multilevel information sharing strategy to deal with constraints. In their work, a better performer list (BPL) is generated by a multilevel Pareto ranking scheme treating every constraint as an objective, while the particle which is not in the BPL gradually congregates round its closest neighbor in the BPL.

In these swarm intelligence algorithms, the most popular and widely used algorithms for solving the TSP are GAs, ACO and PSO. Clerc [27] develops several algorithm variants with those operations and methods and applies them to the asymmetric TSP instance br17.atsp. In his algorithm the positions are defined as TSP tours represented in vectors of permutations of the |N| vertices of the graph correspondent to the considered instance. This approach was applied to tackle the real problem of finding out the best path for drilling operations [29].

Particle swarm optimization is presented to solve traveling salesman problem [28], where the authors have proposed the concept of swap operator and swap sequence, and redefined some operators on the basis of them. This paper has designed a special PSO, but the special PSO does not improve the updating formula, and the experiment results are worse than ours.

Hendtlass [30] proposes the inclusion of a memory for the particles in order to improve diversity. The memory of each particle is a list of solutions (target points) that can be used as an alternative for the current local optimal point. There is a probability of choosing one of the points of the particle’s memory instead of the current best point of the whole swarm Pgdb. The size of the memory list and the probability are new parameters added to the standard PSO algorithm. The algorithm is applied to the benchmark TSP instance burma14. The results obtained with algorithmic versions with several parameter settings are compared with the results of an Ant Colony Optimization algorithm.

Pang *et al.* [31] extends the work of Wang *et al.* [28]. Their algorithm alternates among the continuous and the discrete (permutation) space. In order to avoid premature convergence, Pang *et al.* [31] use a chaotic operator. This operator changes randomly the position and velocity in the continuous space, multiplying these vectors by a random number. Four versions of their algorithm are applied to four benchmark instances with 14 to 51 cities: burma14, eil51, eil76 and berlin52. The algorithm variations comprise the presence or not of chaotic variables and the two local search procedures. In the set of instances tested, the results showed that the version that includes chaotic variables and the 2-opt local search presented the best results. Pang *et al.* [32] present a fuzzy-based PSO algorithm for the TSP. They apply their algorithm to instances burma14 and berlin52. No average results or comparisons with other algorithms are reported.

A hybrid approach that joins PSO, Genetic Algorithms and Fast Local Search is presented by Machado & Lopes [33] for the TSP. The positions of the particles represent TSP tours as permutations of |N| cities. The value assigned to each particle (fitness) is the rate between a constant Dmin and the cost of the tour represented in the particle’s position. The hybrid PSO is applied to the following symmetric TSP benchmark instances: pr76, rat195, pr299, pr439, d657, pr1002, d1291, rl1304, d2103.

Goldbarg *et al.* [34] present a PSO algorithm for the TSP where the idea of distinct velocity operators is introduced. The velocity operators are defined according to the possible movements a particle is allowed to do. This algorithmic proposal obtained very promising results. It was applied to 35 benchmark TSP instances with 51 to 7397 cities. The results were comparable to the results of state-of-the-art algorithms for the TSP.

Yuan *et al.* [35] propose new concepts for " chaos variables" and memory for particles. The memory of each particle is a |N|-dimensional vector of chaos variables. The chaos variables are numbers in the interval (0, 1) and are generated with a method proposed by the authors. They apply their algorithm to four benchmark instances with 14 to 51 cities: burma14, oliver30, att48, eil51. The results obtained for instances oliver30 and att48 are compared with the results obtained by algorithms based on: Simulated Annealing, Genetic Algorithm and Ant Colony Systems. Their algorithm outperforms the others regarding quality of solution of these two instances.

Fang *et al.* [36] present a PSO algorithm for the TSP where an annealing scheme is used to accept the movement of a particle. They apply their algorithm to instances oliver30 and att48. The results are compared with the results of algorithms based on: Simulated Annealing, Genetic Algorithms and Ant Colony. In the two instances tested, their algorithm presents the best average results.

Preserving diversity in particle swarm optimization is used to solve traveling salesman problem [37]. This paper showed that by adding a memory capacity to each particle in a PSO algorithm performance can be significantly improved to a competitive level to ACO only on the smaller TSP problems.

Particle swarm optimization-based algorithms are presented for TSP and generalized TSP [38], where an uncertain searching strategy and a crossover eliminated technique are used to accelerate the convergence speed. Compared with the existing algorithms for solving TSP using swarm intelligence, it has been shown that the size of the solved problems could be increased by using the proposed algorithm.

A hybrid multi-swarm particle swarm optimization algorithm is presented to solve the probabilistic traveling salesman problem [39] . The Probabilistic Traveling Salesman Problem (PTSP) is a variation of the classic Traveling Salesman Problem (TSP) and one of the most significant stochastic routing problems. In the PTSP, only a subset of potential customers needs to be visited on any given instance of the problem. The number of customers to be visited each time is a random variable. In the paper, a new hybrid algorithmic nature inspired approach based on Particle Swarm Optimization (PSO), Greedy Randomized Adaptive Search Procedure (GRASP) and Expanding Neighborhood Search (ENS) Strategy is proposed for the solution of the PTSP. The proposed algorithm is tested on numerous benchmark problems from TSPLIB with very satisfactory results. Comparisons with the classic GRASP algorithm, with the classic PSO, and with a Tabu Search algorithm are also presented. Also, a comparison is performed with the results of a number of implementations of the Ant Colony Optimization algorithm from the literature. Our proposed algorithm provides a new best solution.

An efficient method based on a hybrid genetic algorithm–particle swarm optimization (GA–PSO) is presented for various types of economic dispatch (ED) problem [40]. The arithmetic crossover operator is used as crossover operator in the genetic algorithm. It can be defined to produce a new child as linear combination of two chromosomes. As randomly generated a coefficient is used to produce new child. In this method, a new approach for obtaining a coefficient is proposed benefit from the similarity of parent chromosomes. The obtained results showed that the proposed method has been applied successfully in various ED problems.

A novel two-stage hybrid swarm intelligence optimization algorithm is presented to solve traveling salesman problem [41]. This paper presents a novel two-stage hybrid swarm intelligence optimization algorithm called GA–PSO–ACO algorithm that combines the evolution ideas of the genetic algorithms, particle swarm optimization and ant colony optimization based on the compensation for solving the traveling salesman problem.

3. Basic Particle Swarm Optimization Algorithm

The Particle Swarm Optimization (PSO) algorithm was proposed by Eberhart and Kennedy. PSO was presented under the inspiration of bird flock immigration during the course of finding food. It is widely used in solving the optimization problems.

The standard PSO algorithm works by having a population (called a swarm) of candidate solutions (called particles). These particles are moved around in the search space according to a few simple formulae. The movements of the particles are guided by their own best known position in the search space as well as the entire swarm's best known position. When improved positions are being discovered, they are used to guide the movements of the swarm. The process is repeated until that a satisfactory solution is discovered.

Formally, let $f: R^n \to R$ be the function which needs to be minimized. The function takes a candidate solution as argument in the form of a vector of real numbers, and then produces a real number as output, which indicates the objective function value of the given candidate solution. The gradient of f is not known. The goal is to find a solution A for which $f(A) \le f(B)$ for all B in the search space. That is, the solution A is the global minimum. Maximization can be performed by considering the function $h = -f$ instead.

In PSO, each solution is regarded as a bird in the search space and called “particle”. Each particle has a fitness value which is determined by a target function. The status of each particle includes its position and velocity. Its velocity determines its flying direction. All particles (the swarm) work together in searching for the optimal solution in the solution space. PSO achieves the searches via updating the status of each particle. At the beginning, it randomly initiates a group of particles (random solutions) with a specific position and velocity for each. It updates the status (its velocity and position, referring Formula 1 and 2) of each particle using the recorded best position experienced for this particle (called P_{idb} in Formula 1) and the best position of the whole swarm (called P_{gdb} in Formula 1) until now. This updating procedure continues until it reaches the maximum number of iterations, which is set up at the beginning. Through this iteratively updating, PSO finds (or approaches to) optimal solutions in the solution space.

Thus, the status of each particle is updated not only based on itself best position (P_{idb}) experienced, but also based on the best position of the whole swarm (its companions). That is, the particles inside the swarm share the information (P_{gdb}). Specifically, for a particle id, its velocity and its position is updated according to the formula as follows respectively:

$$V_{id}^{'} = \omega V_{id} + \eta_1 rand()(P_{idb} - X_{id}) + \eta_2 rand()(P_{gdb} - X_{id}) \qquad (1)$$

$$X_{id}^{'} = X_{id} + V_{id}^{'} \qquad (2)$$

where ω is called the inertia weight. It is a proportion factor concerned with former velocity ($0 < \omega < 1$). η_1 and η_2 are constant accelerating factors, normally $\eta_1 = \eta_2 = 2$. The random function $rand()$ is to generate random numbers. X_{id} represents the position of particle id. V_{id} represents the velocity of particle id. P_{idb} and P_{gdb} represent the best position of the particle id found and the best position of the whole swarm found respectively until this moment.

In the formula 1 above, the first part represents the impact of the former velocity of the particle. It enables the particle to possess expanding tendency in the searching space, and thus makes the algorithm be more capable in global searching. The second part is called a cognition part. It represents the process of absorbing individual experience knowledge of the particle. The third part is called a social part. It represents the process of learning from the experience of other particles. It also shows the information sharing and social cooperation among particles.

The flow of PSO can briefly be described as follows. First, PSO initiates a group of particles, and assigns each particle a random initial position X_{id} and a random initial velocity V_{id} . Then PSO calculates the fitness value f for each particle. Note that the best position P_{idb} of each particle is its current position X_{id} in the first updating. The best position of the swarm P_{gdb} is the best position which achieves the best fitness value of the swarm. With the two bests, PSO updates the velocity and position for each particle, which are calculated according to the formula 1 and 2 respectively. Then PSO reevaluates the fitness value for each particle using its current velocity and position. With the fitness value of each particle, PSO updates the best position P_{idb} for each particle and the best position P_{gdb} for the swarm. Then PSO uses the two bests to update the velocity and position for each particle again, according to the formula 1 and 2 respectively. This procedure repeats until using up the number of iterations set at the beginning.

Note that the best position P_{idb} used in the velocity updating formula (*i.e.*, Formula 1) enforces this particle to search for the optimal value around the position P_{idb}. This is represented as the cognition component in the formula 1. The best position of the swarm P_{gdb} is also used to adjust the velocity of the particle. This enforces the particle to approach the best position of the swarm P_{gdb} . This is the social component in the formula 1. These two adjustments enable particles to approach to two bests, and converge quickly.

The most advantage of PSO is its convergence speed. Clerc [13] has presented the proof on its convergence but the standard PSO algorithm can be easily trapped into local optima, especially for multimodal functions. we use three benchmark functions to verify it. The details of the test function see Table 1. In the Table 1, S behalf of the range of variables, $f_{\min}$ behalf of the minimization of the function. we run each algorithm 100 times for the ten functions described above. Our experimental results are shown in Table 2. Figure 1 to Figure 3 are the figures of the three functions.

Table 1. Test Function

Function	S	$f_{\min}$
$f_1(x) = \sum_{i=1}^{n} x_i^2$	(-100,100)	0
$f_2(x) = \frac{1}{4000}\sum_{i=1}^{n}(x_i - 100)^2 - \prod_{i=1}^{n}\cos(\frac{x_i - 100}{\sqrt{i}}) + 1$	(-300,300)	0
$f_3(x) = \sum_{i=1}^{n} -x_i \sin(\sqrt{\lvert x_i \rvert})$	(-500,500)	-12569.5

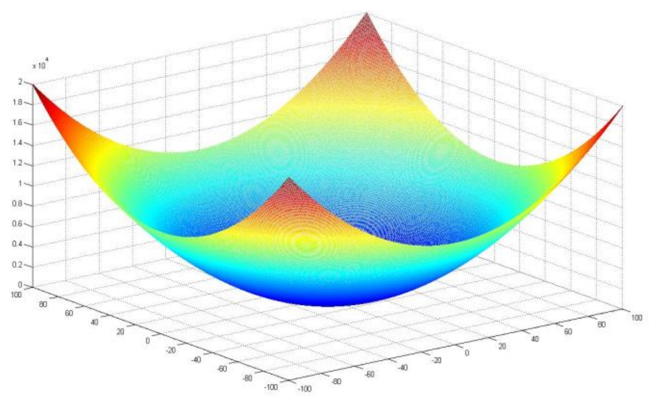


Figure 1. Test Function F1

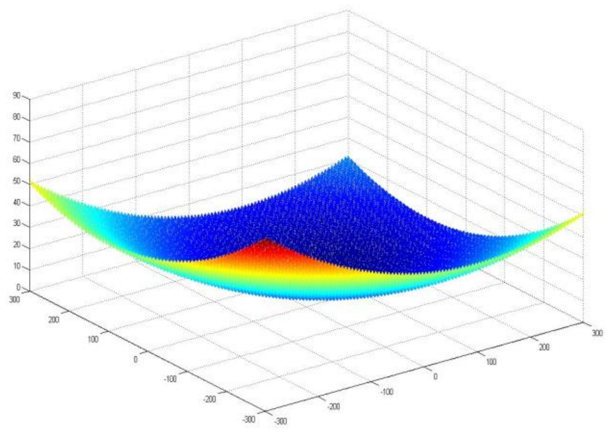


Figure 2. Test Function F2

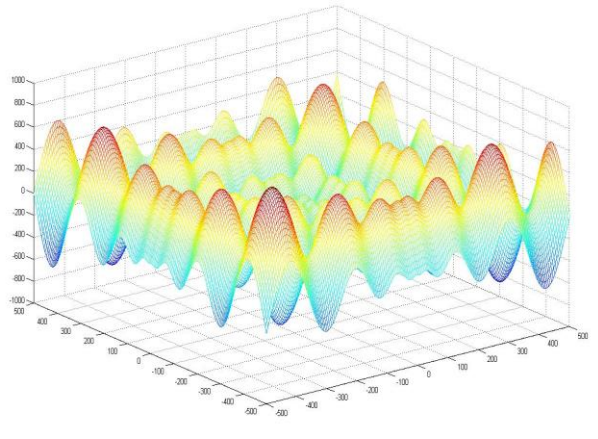


Figure 3. Test Function F3

Table 2. Experimental Results

Function	Best Value	Mean Value	Worst Value	$f_{\min}$
F1	1495.71	4224.77	7032.89	0
F2	72.51	101.41	123.95	0
F3	-5038.62	-4005.02	-3233.13	-12569.5

4. Improved PSO Algorithm (IPSO)

In the standard PSO algorithm above, the convergence speed of particles is fast, but the adjustments of cognition component and social component make particles search around the two best points: P_{gdb} and P_{idb}. According to the velocity and the position updating formula (Formula 1 and 2), once the best individual in the swarm is trapped into a local optimum, the information sharing mechanism in the standard PSO will attract other particles to approach this local optimum gradually. In the end, the whole swarm will be converged to the local optimum.

Let us look at the velocity and position updating formula (Formula 1 and 2) again. Once the whole swarm is trapped into a local optimum, its cognition component and social component become zero in the end. With the number of iterations, the velocity of particles will become zero in the end, because the value of the inertia weight is less than 1, *i.e.*, $0<\omega<1$.

When the velocities of particles gradually diminish and reach zero in the end, the whole swarm will be converged at one point in the solution space. At this time, if the current P_{gdb} value achieved by the particles is not the optimal solution, the whole swarm will be trapped into a local optimum. However, the capacity of the swarm jumping out of the local optimum is very weak. Especially, this occurs frequently for multimodal functions. To further verify this, we select three more multimodal functions (shown in Table 1). These experimental results confirm our conjuncture: the standard PSO algorithm is easily trapped into local optima. The main reason of the standard PSO algorithm can be easily trapped into local optima, especially for multimodal functions, is because it has no mechanisms to force the particles to jump out of the local optimum. This is the fatal weakness of PSO. This fatal weakness is the reason why PSO could not always achieve the optimal solution. With this guidance, we will develop a new solution in next subsection, which forces PSO to avoid trapping into local optima.

In order to overcome the shortcoming of the standard PSO algorithm, we propose a new PSO algorithm. The major goal of the new algorithm is to avoid being trapped into local optima. With the guidance of the major goal, our algorithm adopts a new information sharing mechanism. This information sharing mechanism makes particles move on the contrary direction of the worst individual positions and the worst whole swarm positions. Thus, it enlarges global searching space and reduces the possibility of particles to be trapped into a local optimum.

In the standard PSO algorithm, the flying direction of each particle is almost determined and optimistic. It always flies to the best particle/s of the whole swarm, and also hangs around its own best position experienced. From the analysis in the subsection above, we know that it is dangerous when the best particle of the whole swarm is trapped into a local optimum. In order to reduce the possibility of being trapped into the local optimum, the new algorithm creates two possible flying directions (an optimistic one and a pessimistic one) for each particle, instead of just one determined and optimistic direction. We use the pessimistic direction to save the particles when they trap into the local minimum. Since the pessimistic direction might not always be worse than the optimistic direction, the new algorithm will choose the best one from two possible flying directions to update each particle. This is a competition strategy introduced into the new algorithm. Both the worst information mechanism and the best information mechanism compete with each other. The competition not only further decreases the possibility of being trapped into a local optimum, but also makes the search converge to optimal solutions, although its complexity is only two times of the standard PSO.

How does the new algorithm create the two possible flying directions (an optimistic one and a pessimistic one) for each particle? We know that the standard PSO already creates one optimistic flying direction for each particle. This direction is still kept in the new algorithm as the optimistic flying direction. The new algorithm needs to create the pessimistic flying direction for each particle. The procedure of creating the pessimistic flying direction is the same as that of creating the optimistic flying direction. We know that when a particle is searching in the solution space, it doesn't know the exact position of the optimum solution. The standard PSO records the best position an individual particle has achieved, and the best ones that the whole swarm have experienced. Based on the best position of each particle and the best position of the whole swarm, the standard PSO creates the optimistic direction using Formula 1 and 2. To create the pessimistic direction, we can modify the process of creating the optimistic direction used in the standard PSO. That is, we can also record the worst position an individual particle has achieved, and the worst ones that the whole swarm have experienced. Based on the worst position of each particle and the worst position of the whole swarm, the new algorithm creates the pessimistic direction using Formula 3 and 4, which define the particle velocity and position updating as follows:

$$V_{id}^{'} = \omega V_{id} + \eta_1 rand()(X_{id} - P_{idw}) + \eta_2 rand()(X_{id} - P_{gdw}) \tag{3}$$

$$X_{id}^{'} = X_{id} + V_{id}^{'} \tag{4}$$

where P_{idw} represents the worst position that the particle id has found. P_{gdw} represents the worst positions that the whole swarm has found.

The new algorithm chooses the best one from the two potential flying directions (an optimistic one and a pessimistic one) to update each particle. This surely enlarges the global searching space of particles, and enables them to avoid being trapped into a local optimum too early and in the same time. Thus, the new PSO improves the possibility of finding the global optimum in the search space. Since the new algorithm has to calculate two potential velocities and positions for each particle in the swarm, its time complexity is twice of the standard PSO.

In order to investigate the improvement of our proposed algorithm, we compare it with the standard PSO on the three more multimodal functions (shown in Table 1), the experimental results (the best, mean, and worst values) are shown in Table 3. From Table 3, we can conclude that our new PSO also significantly improves the performance of PSO. For these three difficult functions, our new PSO can achieve very close to the optimal solutions shown in the last column (only a tiny difference). Its best, mean and worst values are also very close to each other.

Table 3. The Experimental Results of IPSO and PSO on Three Difficult Multimodal Functions

Function	Algorithm	Best Value	Mean Value	Worst Value	$f_{\min}$
F1	PSO	1495.71	4224.78	7032.89	0
	New PSO	4.14E-29	4.46E-26	1.09E-24	0
F2	PSO	72.51	101.41	123.95	0
	New PSO	2.19E-12	0.01	8.63E-25	0
F3	PSO	-5038.62	-4005.02	-3233.13	-12569.5
	New PSO	-9535.19	-8741.27	-8203.56	-12569.5

5. Improved PSO Algorithm for TSP

The Travelling Salesman Problem (TSP) is a classical combinatorial optimization problem, which is simple to state but very difficult to solve. The problem is to find the shortest possible tour through a set of N vertices so that each vertex is visited exactly once. This problem is known to be NP-complete, and cannot be solved exactly in polynomial time. The exact algorithms are typically derived from the integer linear programming (ILP) formulation of the TSP.

$$
\begin{aligned}
&Min\sum_i \sum_j d_{ij}x_{ij}\\
&s.t:\\
&\sum_j x_{ij} = 1, i = 1,2,3,...,N,\\
&\sum_i x_{ij} = 1, j = 1,2,3,...,N,\\
&(x_{ij}) \in X,\\
&x_{ij} = 0 \quad or \quad 1,\\
&\sum_{i,j\in S_V} x_{ij} < |S_V - 1|, (S_V \subset V, 2 \le |S_V| \le N-2)
\end{aligned}
\tag{5}
$$

where d_{ij} is the distance between vertices i and j and the $x_{ij's}$ are the decision variables: x_{ij} is set to 1 when arc (i,j) is included in the tour, and 0 otherwise. $x_{ij} \in X$ denotes the set of subtour-breaking constraints that restrict the feasible solutions to those consisting of a single tour. V is the set of all vertices, S_V is some subset of V and $|S_V|$ is the cardinality of S_V. These constraints prohibit subtours, that is, tours on subsets with less than N vertices.

5.1. Problem Code and Fitness Function

In this paper, we use the most direct way to denote TSP-paths. TSP-paths are considered as rings, or closed paths. The TSP-path is referred as a chromosome in GA, PSO and our IPSO. A fitness function is the only standard of judging whether an individual one is "good" or not. We take the reciprocal of the length of each path as the fitness function defined in Formula 6. The shorter the length, the higher the fitness value is.

$$f(S_i) = 1 \Big/ \sum_{i=1}^{N} d[C_{n(i)}, C_{n(i+1)\bmod N}] \tag{6}$$

5.2. Experimental Results

In order to verify that the proposed algorithm IPSO is useful for the TSP, we compare it with the standard PSO on ten TSP problems, which come from the TSPLIB(http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95). Their optimal solutions are given in the website. The number of cities in these ten TSP problems varies from 52 to 1432. We list them in Table 4 with our experimental results of our new PSO, the standard PSO, and the GA algorithm. The experimental results shown in Table 7 are best solutions obtained by the three algorithms, running 10 times. The parameters of our IPSO and the standard PSO are set as the same, *i.e.*, ω=0.6, η1=η2=2. All experiments are performed on an Intel CoreTM2 Duo CPU, 2.26GHz/4G RAM Laptop.

Table 4 shows that our IPSO achieves the optimal solutions for all ten TSP problems. Both the standard PSO and the GA algorithm only achieve the optimal solution for the first TSP problem. However, the standard PSO achieves better solutions for the rest nine

TSP problems than the GA algorithm. We can conclude that our new PSO is the best among the three algorithms, followed by PSO, followed by the GA algorithm.

Table 4. The Best Solutions Achieved By Our New PSO, the Standard PSO, and the GA Algorithm, Running 10 Times on the Ten TPS Problems

TSP Problems	Optimal in TSPLIB	Genetic Algorithm	PSO Algorithm	New PSO Algorithm
Berlin52	7542	7542	7542	7542
kroA100	21282	21315	21310	21282
kroA200	29368	30168	29968	29368
Pr299	48191	48568	48540	48191
Rd400	15281	15135	15135	15281
Ali535	202310	242310	231120	202310
D657	48912	50912	50612	48912
Rat783	8806	8965	8905	8806
U1060	224094	279094	269908	224094
U1432	152970	182780	177890	152970

6. Conclusion

This paper proposed a new PSO algorithm called IPSO. Comparing with the standard PSO algorithm, the proposed PSO algorithm has improvements: it introduces a new information sharing mechanism: the worse position of each particle and the worst position of the whole swarm. This information sharing mechanism makes particles move on the contrary direction of the worst individual positions and the worst whole swarm positions. Thus, it enlarges global searching space and reduces the possibility of particles to be trapped into a local optimum;

In summary, in this paper, we analyzed the shortcoming of the standard PSO. With the guidance of the analysis, we proposed a new PSO algorithm. Our experimental results showed that new PSO performs much better than the standard PSO on benchmark functions and TSP problems. Especially for difficult functions and TSP problems, our new PSO significantly improves the success rate of finding the optimal solutions.

Acknowledgements

This paper is supported by Natural Science Foundation of China. (No. 61402425, 61272470, 61305087, 61440060, 41404076), the Provincial Natural Science Foundation of Hubei (No. 2013CFB320, 2015CFA065).

References

[1] R. Durbin and D. Willshaw, "An Anlaogue Approach to the Traveling Salesman Problem Using an Elastic Net Approach", Nature, vol. 326, no. 6114, **(1987)**, pp. 689-691.

[2] T. Guo and Z. Michalewize, "Inver-Over operator for the TSP, In Parallel Problem Sloving from Nature (PPSN V)", Springer-Verlag press, **(1998)**, pp. 803-812.

[3] Z. Huang, X. Hu and S. Chen, "Dynamic Traveling Salesman Problem based on Evolutionary Computation", In Congress on Evolutionary Computation, **(2001)**, pp. 1283-1288

[4] A. Zhou, L. Kang and Z. Yan, "Solving Dynamic TSP with Evolutionary Approach in Real Time", In Congress on Evolutionary Computation, **(2003)**, pp. 951-957.

[5] H. Yang, L. Kang and Y. Chen, "A Gene-pool Based Genetic Algorithm for TSP", Wuhan University Journal of Natural Sciences, vol. 8, no. 1B, **(2003)**, pp. 217-223.

[6] X. Yan and L. Kang, "An Approach to Dynamic Traveling Salesman Problem", Proceedings of the Third International Conference on Machine Learning and Cybernetics, **(2004)**, pp. 2418-2420.

[7] X. Yan, A. Zhou and L. Kang, "TSP Problem Based on Dynamic Environment". Proceedings of the 5th World Congress on Intelligent Control and Automation, **(2004)**, pp. 2271-2274.

[8] J. Kennedy and R. C.Eberhart, "Particle Swarm Optimization", IEEE International Conference on Neural Networks, **(1995)**, pp. 1942-1948.

[9] M. Clare and J. Kennedy, "The Particle Swarm - Explosion, Stability, and Convergence in a Multidimensional Complex Space", IEEE Trans. on Evolutionary Computation, vol.6, no. 1, **(2002)**, pp. 58-73.

[10] C.A. Coello and M.S. Lechuga, " Mopso, A proposal for multiple objective particle swarm optimization", In IEEE Proceedings World Congress on Computational Intelligence, **(2002)**, pp. 1051-1056

[11] J. Kennedy, "The particle swarm: social adaptation of knowledge", In Proc. IEEE Conf. on evolutionary computation, **(1997)**, pp. 3003-3008.

[12] E. Oscan and C. K.Mohan, "Analysis of A Simple Particle Swarm Optimization System", Intelligence Engineering Systems Through Artificial Neural Networks, **(1998)**, pp. 253-258

[13] X. Yan, Q. Wu, C. Hu, H. Yao, Y. Fan, Q. Liang and C. Liu, "Robot Path Planning based on swarm Intelligence", International Journal of Control and Automation, vol. 7, no. 7, **(2014)**, pp. 15-32.

[14] Q. Wu, H. Liu and X. Yan, "An improved design optimisation algorithm based on swarm intelligence", International Journal of Computing Science and Mathematics, vol. 5, no. 1, **(2014)**, pp. 27-36.

[15] Z. Hua and F. Huang, "A variable-grouping based genetic algorithm for large-scale integer programming", Inf Sci., vol. 176, no. 19, **(2006)**, pp. 2869–2885.

[16] I. Ellabib, P. Calamai and O. Basir, "Exchange strategies for multiple ant colony system", Inf Sci, vol. 177, no. 5, **(2007)**, pp. 1248–1264.

[17] C.C. Lo and C.C. Hus, "Annealing framework with learning memory", IEEE Trans Syst Man Cybern Part A, vol. 28, no. 5, **(1998)**, pp. 1–13.

[18] T.A. Masutti and L.N. de Castro, "A self-organizing neural network using ideas from the immune system to solve the traveling salesman problem", Inf Sci, vol. 179, no. 10, **(2009)**, pp. 1454–1468.

[19] G.C. Onwubolu and M. Clerc, "Optimal path for automated drilling operations by a new heuristic approach using particle swarm optimization", Int J Prod Res, vol. 42, no. 3 9, **(2004)**, pp. 473–491

[20] G. Shen and Y.Q.Zhang, "A new evolutionary algorithm using shadow price guided operators", Appl Soft Comput, vol. 11, no. 2, **(2011)**, pp. 1983–1992.

[21] G. Acampora, M. Gaeta and V. Loia, "Combining multi agent paradigm and memetic computing for personalized and adaptive learning experiences", Comput Intell J, vol. 27, no.2, **(2011)**, pp. 141–165.

[22] Y. Marinakis, M. Marinaki and G. Dounias "A hybrid particle swarm optimization algorithm for the vehicle routing problem", Eng Appl Artif Intell, vol. 23, no. 4, **(2010)**, pp. 463–472

[23] D. Banaszak, G.A. Dale, A.N. Watkins and J.D. Jordan, "An optical technique for detecting fatigue cracks in aerospace structures", In: Proc 18th ICIASF, **(2009)**, pp. 1–7.

[24] K.E. Parsopoulos and M.N. Vrahatis, "Particle swarm optimization method for constrained optimization problems", In Proceedings of the second Euro-International Symposium on Computational Intelligence, **(2002)**, pp. 214–220

[25] X. Hu and R.C. Eberhart, "Solving constrained nonlinear optimization problems with particle swarm optimization", In Proceedings of the Sixth World Multi conference on Systemics, Cybernetics and Informatics, **(2002)**, pp. 203–206.

[26] T. Ray and K.M. Liew, "A swarm with an effective information sharing mechanism for unconstrained and constrained single objective optimisation problems", In Proceedings of the 2001 Congress on Evolutionary Computation, **(2001)**, pp. 75–80.

[27] M. Clerc, "Discrete particle swarm optimization, illustrated by the traveling salesman problem, In: Studies in Fuzziness and Soft Computing New optimization techniques in engineering", Babu, B.V. & Onwubolu, G.C. (Eds.), vol. 141, **(2004)**, pp. 219–239.

[28] K.P. Wang, L. Huang, C.G. Zhou and W. Pang, "Particle swarm optimization for traveling salesman problem", In International conference on machine learning and cybernetics, **(2003)**, pp.1583–1585.

[29] G.C. Onwubolu and M. Clerc, "Optimal path for automated drilling operations by a new heuristic approach using particle swarm optimization ", International Journal of Production Research, vol. 42, no. 3, **(2004)**, pp. 473–491.

[30] T. Hendtlass, "Preserving diversity in particle swarm optimization, Developments in Applied Artificial Intelligence", Proceedings of the 16th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems IEA/AIE 2003, vol. 2718, **(2003)**, pp. 4104-4108

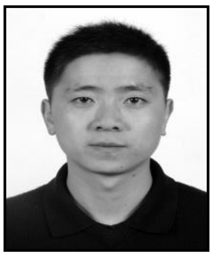
[31] W. Pang, K. Wang, C. Zhou, L. Dong, M. Liu, H. Zhang and J. Wang, "Modified particle swarm optimization based on space transformation for solving traveling salesman problem", In Proceedings of the Third International Conference on Machine Learning and Cybernetics, **(2004)**, pp. 2342-2346

[32] W. Pang, K. Wang, C. Zhou and L. Dong, "Fuzzy discrete particle swarm optimization for solving traveling salesman problem", In Proceedings of the Fourth International Conference on Computer and Information Technology, **(2004)**, pp. 796-800

[33] T.R. Machado and H.S. Lopes, "A hybrid particle swarm optimization model for the traveling salesman problem", In: Natural Computing Algorithms, Ribeiro, H.; Albrecht, R.F. & Dobnikar, A. (Eds.) , **(2005)**, pp. 255-258.

[34] E.F.G. Goldbarg, G.R. Souza and M.C. Goldbarg, "Particle swarm for the traveling salesman problem", Proceedings of the EvoCOP 2006, Gottlieb, J. & Raidl, G.R. (Ed.), Lecture Notes in Computer Science, vol. 3906, **(2006)**, pp. 99-110.
[35] Z. Yuan, L. Yang, Y. Wu, L. Liao and G. Li, "Chaotic particle swarm optimization algorithm for Traveling Salesman Problem", In Proceedings of the IEEE International Conference on Automation and Logistics, Jinan, China, **(2007)**, pp. 1121-1124.
[36] L. Fang, P. Chen and S. Liu, "Particle swarm optimization with simulated annealing for TSP", Proceedings of the 6th WSEAS International Conference on Artificial Intelligence, Knowledge Engineering and Data Bases, Long, C. A.; Mladenov, V. M. & Bojkovic Z. (Eds.), **(2007)**, pp. 206-210.
[37] T. Hendtlass, "Preserving diversity in particle swarm optimization", Lect Notes Comput Sci, vol. 2718, **(2003)**, pp. 4104–4108.
[38] X.H. Shi, Y.C. Liang, H.P. Lee, C. Lu and Q.X. Wang, "Particle swarm optimization-based algorithms for TSP and generalized TSP", Inf Process Lett, vol. 103, no. 5, **(2007)**, pp. 169–176.
[39] M. Yannis and M. Magdalene, "A hybrid multi-swarm particle swarm optimization algorithm for the probabilistic traveling salesman problem", Comput Oper Res, vol. 37, no.3, **(2010)**, pp. 432–442.
[40] U. Guvenc, S. Duman, B. Saracoglu and A. Ozturk, "A hybrid GA–PSO approach based on similarity for various types of economic dispatch problems", Electron Electr Eng Kaunas: Technologija, vol. 2, no. 108, **(2010)**, pp. 109–114.
[41] W. Deng, R. Chen, B. He, Y. Liu, L. Yin and J. Guo, "A novel two-stage hybrid swarm intelligence optimization algorithm and application", Soft Computing, vol. 16, **(2012)**, pp. 1707-1722.

Authors



Xuesong Yan, he received the MS degree in Computer Application Technology from China University of Geosciences in 2003, and the PhD degree in Computer Software and Theory from the School of Computer Science, Wuhan University in 2006. He is currently an associate professor in China University of Geosciences. His research interests are in the areas of evolutionary computation, evolvable hardware and machine learning.

Qinghua Wu, she received the MS degree in Computer Application Technology from China University of Geosciences in 2003, and the PhD degree in Earth Explore and Information Technology from China University of Geosciences, in 2011. She is currently a lecturer in Wuhan Institute of Technology. Her research interests are in the areas of evolutionary computation and image processing.

Yuanyuan Fan, she received the MS degree in Computer software and theory from Wuhan University in 2004, and the PhD degree in Computer software and theory from School of Computer Science, Wuhan University in 2011. Her is currently an lecture in China University of Geosciences. Her research interests are in the areas of evolutionary computation and evolvable hardware.

Qingzhong Lian, he received the MS degree in Computer application technolgoy from China University of Geosciences in 2004, and now the PhD Candidate in China University of Geosciences. He is currently an lecture in China University of Geosciences. His research interests are in the areas of network architecture, computer applications and information systems.

Chao Liu, he received the MS degree in Computer application technolgoy from China University of Geosciences in 2004, and now the PhD Candidate in Computer Architecture from School of Computer Science, Huazhong University of Science and Technology. He is currently an lecture in China University of Geosciences. His research interests are in the areas of network architecture, computer applications and information systems.