2014 International Symposium on Nonlinear Theory and its Applications
NOLTA2014, Luzern, Switzerland, September 14-18, 2014

Solving Shortest Path Problems by Adaptable Independent-minded Particle Swarm Optimization

Yoko Ishii, Haruna Matsushita and Yo Horikawa

Department of Electronics and Information Engineering, Kagawa University
2217–20 Hayashi-cho, Takamatsu, Kagawa 761–0396, Japan
Email: s14g452@stmail.eng.kagawa-u.ac.jp, haruna@eng.kagawa-u.ac.jp, horikawa@eng.kagawa-u.ac.jp

Abstract—This study proposes a method of solving the shortest path problem (SPP) using the adaptable independent-minded particle swarm optimization (AIPSO). The system of AIPSO is almost the same as PSO, however, a connection relationship between particles of AIPSO dynamically changes with each iteration. We apply the proposed method using AIPSO to solving two kinds of SPPs derived by the Small-world network and the Waxman network. We confirm that AIPSO can significantly improve the optimization performance from the basic PSO.

1. Introduction

The shortest path problem (SPP) is one of the most fundamental problems in the graph theory, and it is a familiar optimization problem for us. Examples of SPP include car navigation system, searching system of train connections and so on. Many researchers have investigated SPP and its solutions and SPP has gotten complex with the developments of communication tools, computer science and transport systems. These extended SPPs include multi-objective SPP, traveling salesman problems (TSPs) and the K-th shortest path problem, and most of the extended SPPs are NP-hard problems.

The most famous method of solving the simple SPP is Dijkstra's algorithm [1]. However, since the Dijkstra's algorithm cannot solve the extended SPPs being NP-hard problems, it is important to investigate various methods of solving SPPs.

The particle swarm optimization (PSO) [2] is a hot topic as an optimization method, and PSO is based on birds and fish flock behavior that search for foods. The most attractive feature of PSO is that it requires less calculating equations, and a SPP solution method using PSO has been proposed [3]. However, since PSO is easily trapped into local optima when it solves complex problems, many improved PSOs have been proposed. We therefore pay attention to the adaptable independent-minded particle swarm optimization (AIPSO) [4]. The system of AIPSO is almost the same as PSO, however, a connection relationship between particles of AIPSO dynamically changes with each iteration. AIPSO can significantly improve the optimization performance of the basic PSO although AIPSO does not need any additional parameter. This study modifies the conventional method using PSO and proposes a method of solving SPP using AIPSO. We investigate an effectiveness of the proposed method using AIPSO by applying it to solving two kinds of complex SPPs derived by the Small-world network and the Waxman network.

2. Adaptable independent-minded particle swarm optimization (AIPSO)

PSO is one of the optimization methods based on the swarm intelligence. Kennedy and Eberhart designed PSO inspired by the flock behavior of birds and fish in nature. PSO searches an optimum solution by using a group of solutions called "particle swarm". Each particle searches for the optimum solution with flying around a multi dimensional optimization space. PSO has been applied to various problems because it requires less computational and a few lines of codes, and various modified PSOs have been proposed.

AIPSO is one of the improved PSOs. A difference between the basic PSO and AIPSO is a connection relationship between the particles as shown in Fig. 1. AIPSO has almost the same behavior as the basic PSO, and the particles are affected only by connected particles. In the basic PSO, the connection relationship between the particles are not changed throughout a simulation, in other words, always full-connected, as shown in Fig. 1(a). However, the connection relationship in AIPSO dynamically changes with each iteration as shown in Fig. 1(b). Whether a particle i is affected by the swarm is stochastically-judged at every iteration and every dimension:

$$\begin{cases} i \in S_{Cd}, & \text{with probability } C_p(t) \\ i \in S_{Id}, & \text{with probability } 1 - C_p(t), \end{cases} \tag{1}$$

where t is the iteration, and S_{Cd} and S_{Id} are sets of connected and isolated particles in d_{th} dimension, respectively. The connection probability $C_p(t)$ is a time-varying linear function as described in

$$C_p(t) = (C_{p_T} - C_{p_0})\frac{t}{T} + C_{p_0}, \tag{2}$$

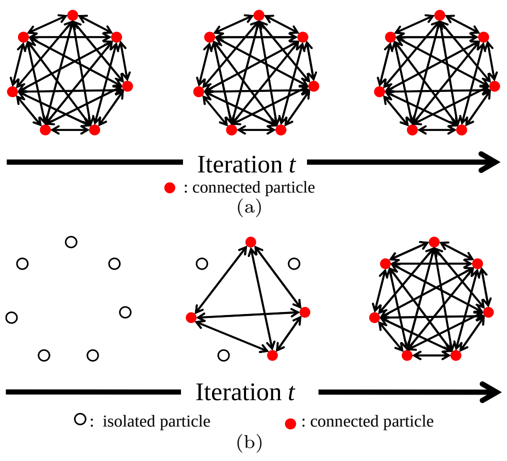


Figure 1: Connection relationship between particles depending on the iteration. (a) PSO. All the particles are always connected each other. (b) AIPSO. The connected particles are increased with time due to an increase in the connection probability.

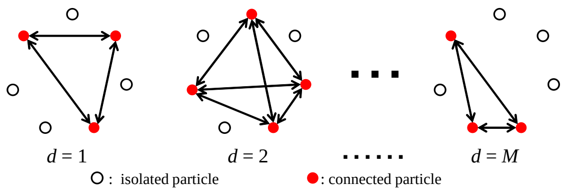


Figure 2: Example of a connection relationship in AIPSO at each dimension. AIPSO has different connection relationship at each dimension.

where T is the maximum number of the iteration, and C_{p_T} and C_{p_0} are the maximum and minimum value of $C_p(t)$, respectively. Figure 2 shows an example of a change in the connection relationship depending on the dimension. AIPSO has different connection relationship at each iteration and each dimension in contrast to the basic PSO being always fully-connected.

Each particle i $(i = 1, 2, \cdots, N)$ has two information; position vector $\boldsymbol{X}_i = (x_{i1}, x_{i2}, \cdots, x_{iM})$ and velocity vector $\boldsymbol{V}_i = (v_{i1}, v_{i2}, \cdots, v_{iM})$. AIPSO searches the solution by updating the particle information according to

$$v_{id}^{t+1} = \begin{cases} \omega v_{id}^t & + \varphi_1 r_1 (x_{p_{id}} - x_{id}^t) \\ & + \varphi_2 r_2 (x_{ld} - x_{id}^t), \quad i \in S_{Cd} \\ \omega v_{id}^t & + \varphi_1 r_1 (x_{p_{id}} - x_{id}^t), \quad i \in S_{Id}, \end{cases} \tag{3}$$

$$x_{id}^{t+1} = x_{id}^t + v_{id}^{t+1}, \tag{4}$$

where ω is the inertial weight, φ_1 and φ_2 are the acceleration coefficients, r_1 and r_2 are independent random numbers in the rage $[0,\ 1]$, $\boldsymbol{X_{pbest_i}} = (x_{p_{i1}}, x_{p_{i2}}, \cdots, x_{p_{iM}})$ is the past best position of the particle i, and $\boldsymbol{X_{lbest}} = (x_{l_1}, x_{l_2}, \cdots, x_{l_M})$ is the best position among all the connected particles.

3. Shortest path problem (SPP)

SPP is defined as follows. An undirected graph $G = (M, E)$ comprises a set of nodes M and a set of edges $E \in M \times M$ connecting nodes in M. Corresponding to each edge, there is a non negative number c_{yz} representing the cost of the edge from a node m_y to a node m_z. A path from a source node m_s to a destination node m_d is a sequence of nodes $\boldsymbol{PATH} = (m_s, m_j, \cdots, m_d)$. The path must not repeat the same node index in the sequence. For example, in Fig. 3, a path from the node 1 to the node 7 is represented as (1, 4, 2, 5, 7). A goal of SPP is to find a path between two nodes having the minimum total cost.

In this paper, SPP is solved by using PSO and AIPSO. In PSOs, the particle information is updated depending on the objective function. The objective function of the particle i is defined as

$$f_i = \sum_{j=1}^{N_i - 1} c_{yz}, \qquad y = \boldsymbol{PATH}^i(j),\ z = \boldsymbol{PATH}^i(j+1), \tag{5}$$

where $\boldsymbol{PATH}^i$ is the set of sequential node indexes for the particle i, $N_i = |\boldsymbol{PATH}^i|$ is the number of nodes that constitutes the path represented by the particle i, and c_{yz} is the cost of the link connecting the node y and the node z. Therefore, the objective function f_i takes the minimum when $\boldsymbol{PATH}^i$ is the shortest path. If the path made by a particle i is an unsuccessful path, its objective function value f_i is assigned a penalty value (large value) so that the particle i does not have an influence on other particles at next iteration.

4. Application of AIPSO to SPP

The main issue in applying AIPSO to SPP is how to combine the particle information with the network. We correlate the particle information of AIPSO with a node priority considering the connection relationship of the network.

⟨ STEP1 ⟩ Initialization

An iteration count t is initialized as $t = 0$, the particle position vectors $\boldsymbol{X}$ are initialized at random in range $[-1.0,\ 1.0]$ and the velocity vectors $\boldsymbol{V}$ are initialized $\boldsymbol{V} = 0$.

⟨ STEP2 ⟩ Update the connection relationship

The connection relationship between the particles is updated according to Eq. (1).

⟨ STEP3 ⟩ Determine the node priority

A node priority $\boldsymbol{P}_i = (p_{i1}, p_{i2}, \cdots, p_{iM})$ are an important variable to create the path from a source node

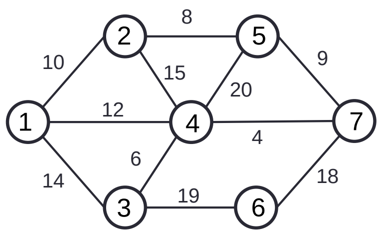


Figure 3: Network model ($M = 7$)

1	2	3	4	5	6	7
0.1	-0.4	0.7	0.6	0.3	0.5	0.2

$k = 0$, $\boldsymbol{PATH}^i = \{1\}$

⇓

1	2	3	4	5	6	7
∞	-0.4	0.7	0.6	0.3	0.5	0.2

($c_{12}p_{i2}$= -4, $c_{13}p_{i3}$= 9.8, $c_{14}p_{i4}$= 7.2)

$k = 1$, $\boldsymbol{PATH}^i = \{1, 2\}$

⇓

1	2	3	4	5	6	7
∞	∞	0.7	0.6	0.3	0.5	0.2

($c_{24}p_{i4}$= 9, $c_{25}p_{i5}$= 2.4)

$k = 2$, $\boldsymbol{PATH}^i = \{1, 2, 5\}$

⇓

1	2	3	4	5	6	7
∞	∞	0.7	0.6	∞	0.5	0.2

($c_{57}p_{i7}$= 1.8)

$k = 3$, $\boldsymbol{PATH}^i = \{1, 2, 5, 7\}$

Figure 4: How to determine the path from the source node 1 to the destination node 7. The numbers $1, 2, \cdots, 7$ denotes the node indexes, and the numbers under the node index are its node priority. Assuming $\boldsymbol{X}_i = \boldsymbol{P}_i = (0.1, -0.4, 0.7, 0.6, 0.3, 0.5, 0.2)$, we obtain a path $\boldsymbol{PATH}^i = \{1, 2, 5, 7\}$. The objective function f_i is 27.

to a destination node. The node priority of the particle i is correlated with the particle information by using a copy of $\boldsymbol{X}_i$, as $\boldsymbol{P}_i = \boldsymbol{X}_i$.

⟨ STEP4 ⟩ Making the PATH

We make the path from a source node to a destination node using the node priority $\boldsymbol{P}_i$. Figure 4 shows an example of creating a path corresponding to a network shown in Fig. 3. We substitute the source node for $\boldsymbol{PATH}^i(1)$. Next visit node $\boldsymbol{PATH}^i(k)$ is selected from the connecting nodes according to

$$\boldsymbol{PATH}^i(k) = \arg\min_d \{p_{id} \times c_{jd}\} \qquad (6)$$

where c_{jd} is the cost of edge from a node j to d. The value ∞ is assigned to the node priority corresponding to the next visit node to avoid selecting the node, which has been visited, again. We continue these processes until the destination node is visited.

⟨ STEP5 ⟩ Evaluation of the objective function

We evaluate the objective function f_i of each particle i by calculating the total path length according to Eq. (5). If the particle i fails to make the path, $f_i = \infty$. Here, if there is no visitable node before the particle i has not arrived yet at the destination node, it is defined as the particle i failed to make a path.

⟨ STEP6 ⟩ Update particle information

The velocity $\boldsymbol{V}_i$ and position $\boldsymbol{X}_i$ are updated according to Eq. (3) and Eq. (4), respectively.

⟨ STEP7 ⟩ Iteration

We repeat the steps from STEP2 to STEP6 until $t = T$ is satisfied.

5. Computer simulation

We apply the proposed SPP solving method using AIPSO to two kinds of complex networks derived by Small-world's [5] and Waxman's [6] method. The proposed solving method is compared with the conventional method using the basic PSO. For respective SPPs, the number of nodes M is 100, the number of edges is set as $edge = 300$ at the Small-world network and as $edge = 279$ at the Waxman network. The edge costs are determined at random. We used following parameters: $N = 30, C_{pT} = 1.0, C_{p0} = 5 \times 10^{-4}, \varphi_1 = \varphi_2 = 1.494$ and $\omega = 0.729$. We use different maximum iteration T depending on the network, and it is $T = 4000$ for the Small-world network and is $T = 3000$ for the Waxman network.

Table 1 shows results obtained by 100 simulations using different destination and source nodes selected at random. The error rate is determined by

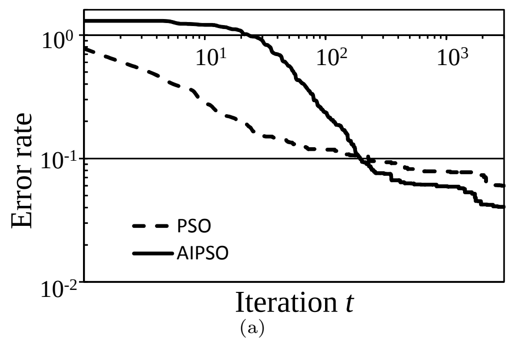
$$\text{Error rate} = \frac{f_g - f_{\text{opt}}}{f_{\text{opt}}} \ [\%] \qquad (7)$$

where f_g is the objective function value of the global best particle obtained by the simulation and f_{opt} is the optimal solution derived by using the Dijkstra method.

From this table, we can see that AIPSO obtains higher success rate and lower error rate than PSO, at both the networks. On the other hand, the improvement rate of AIPSO from PSO is different depending on the networks. This is because the total number of

Table 1: Simulation results

	Success rate [%]		Error rate [%]	
	PSO	AIPSO	PSO	AIPSO
Small-world network	66	**78**	6.02	**4.00**
Waxman network	67	**89**	5.88	**1.37**



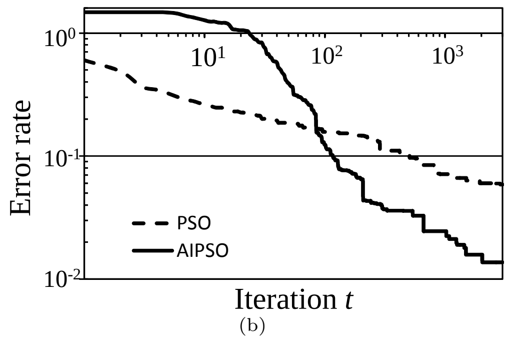


Figure 5: Convergence process. (a) Small-world network. (b) Waxman network.

edges in the network and the number of edges leaving from one node are different between two kinds of networks. Therefore, it is considered that SPP derived by the Small-world network has more local optima than the Waxman network.

In order to investigate a reason why AIPSO can obtain better results than PSO, we consider convergence processes of each method as shown in Fig. 5. In this figure, a horizontal axis shows the iteration t, and a vertical axis shows the error rate from the optimum solution f_{opt}. We can see that the convergence speed of AIPSO is slower than PSO. This is because AIPSO has dynamically-changed connection relationship between the particles. This effect increases the diversity of the particles, and because it avoids a premature convergence, AIPSO keeps on searching even in late stage of the simulation. This means that AIPSO has the ability of both the local search and global search and is hard to be trapped into the local optima.

From these results, we can conclude that the proposed method using AIPSO for solving SPP is more effective than the conventional method using PSO.

6. Conclusions

This study has proposed an application of AIPSO to the method for solving SPP. We have considered two kinds of SPPs derived by Small-world and Waxman method. The simulation results have shown that AIPSO improves the optimization performance from PSO because AIPSO is hard to be trapped into the local optima. These confirmed results mean that AIPSO is effective not only for the continuous optimization problems such as the benchmark functions, but also for discrete optimization problems.

Acknowledgment

This work was supported by KAKENHI 24700226 and "Research and development on constructing techniques of elastic optical networks", the Commissioned Research of National Institute of Information and Communications Technology (NICT).

References

[1] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik 1*, pp. 269–271, 1959.

[2] J. Kennedy and R. C. Eberhart, "Particle Swarm Optimization," *Proceedings of IEEE. Int. Conf. on Neural Networks*, pp. 1942–1948, 1995.

[3] A. W. Mohemmed and N. C. Sahoo, "Efficient Computation of Shortest Paths in Networks Using Particle Swarm Optimization and Noising Metaheuristics," *Discrete Dynamics in Nature and Society*, vol. 2007, pp. 1–25, 2007.

[4] H. Matsushita, T. Saito and Y. Nishio, "Behavior of Independent-Minded Particle Swarm Optimization," *Proceedings of NCSP'11*, pp. 103–106, 2011.

[5] D. J. Watts and S. H. Strogatz, "Collective dynamics of 'small-world' networks," *Nature*, vol. 393, pp. 440–442, 1998.

[6] B. M. Waxman, "Routing of multipoint connections," *IEEE Journal on Selected Areas in Communications*, vol. 6, pp. 1617–1622, 1988.