2016 International Symposium on Nonlinear Theory and Its Applications,
NOLTA2016, Yugawara, Japan, November 27th-30th, 2016

Piecewise-linear particle swarm optimizer networks

Tomoyuki Sasaki†‡, Hidehiro Nakano†, Arata Miyauchi† and Akira Taguchi†

†Department of Computer Science, Tokyo City University, 1-28-1, Tamazutsumi, Setagaya-ku, Tokyo, 158-8557, Japan
‡Research Fellow of Japan Society for the Promotion of Science Email: sasaki@ic.cs.tcu.ac.jp

Abstract—Piecewise-linear particle swarm optimizer (PPSO) is one of the deterministic metaheuristics algorithms. In order to solve a large scale optimization problems, a large number of PPSO swarms are required. However, a parallel computing method for PPSO swarms has not been studied. In this paper, we propose a model of networking PPSO (PPSON) for parallel computing. The effectiveness of PPSON is confirmed by numerical simulations.

1. Introduction

Particle swarm optimizer (PSO) [1] which has been developed by Kennedy and Eberhart in 1995 is one of the population-based stochastic algorithms. PSO mimics social behaviors of creatures such as birds flocking and fish schooling. These creatures are represented by particles as solution candidates, which search a multi dimensional search space and find feasible solutions. PSO has following advantages of: (1) relatively few control parameters; (2) that it is easy to implement PSO to applications; and (3) quick convergence characteristics. Therefore, PSO is applied to various applications.

In order to find good feasible solutions for large scale optimization problems, it is required that many particles search a search space. However, calculation costs are increased in proportion to the number of particles. The calculation costs of particles can be distributed by using multiple PSO circuits in parallel, which calculate the behavior of each particle in the original PSO [2]; however, the PSO circuit has random number generators and multiple floating point multipliers, because the original PSO particle has the stochastic factors. As such, the circuit amount of the original PSO becomes large, and it is hard to implement large number of the circuits on hardware. Therefore, decreasing the circuit amount is required.

In our previous study, piecewise-linear particle swarm optimizer (PPSO) [3] which is one of the deterministic PSOs has been proposed. PPSO particle has two dynamics, which are the convergence and the divergence modes, and searches a search space by switching both dynamics. The solving performances of PPSO are substantially same as those of the original PSO. Furthermore, since PPSO does not have stochastic factors, it can be realized that the size of a PPSO circuit is smaller than that of the PSO circuit [2]. In order to solve large scale optimization problems, a large number of PPSO circuits is required. However, a parallel computing method of PPSO circuits has not been studied.

PSO Network (PSON) [4], which is one of the parallel computing methods for PSO, has been proposed. In PSON, a population is divided into multiple sub-PSOs, and each sub-PSO is connected to neighbor sub-PSOs via network structure. Each sub-PSO searches a search space independently, and communicates own best solution to the neighbor sub-PSOs. When each sub-PSO is assigned by single processor, evaluation costs of the population can be distributed. Furthermore, PSON has better solving performances than the original PSO.

In this paper, we propose a model of networking PPSO (PPSON) for parallel computing. In PPSON, the concept of PSON is applied to PPSO; a population of PPSO is divided into multiple sub-PPSOs, and each sub-PPSO searches a search space independently. Each sub-PPSO is connected to neighbor sub-PPSOs which are determined via network structure, and communicates own best information to the neighbor sub-PPSOs. The effectiveness of PPSON is investigated by numerical experiments compared with PSON and PPSO.

2. Piecewise-linear Particle Swarm Optimizer (PPSO) [3]

In this section, the basic idea of a piecewise-linear particle swarm optimizer (PPSO) is explained. The ith particle has velocity vector $\mathbf{v}_i = (v_{i1}, v_{i2}, \ldots, v_{iD})$, position vector $\mathbf{x}_i = (x_{i1}, x_{i2}, \ldots, x_{iD})$, and $\mathbf{pb}_i = (pb_{i1}, pb_{i2}, \ldots, pb_{iD})$ which is the personal best solution, and shares $\mathbf{gb} = (gb_1, gb_2, \ldots, gb_D)$ which is the global best solution in a swarm. D denotes the number of design variables. Furthermore, PPSO particle has convergence and divergence modes.

The updating rules of the jth component of the ith particle in PPSO are described by

$$q_{ij} = (1-r)pb_{ij} + rgb_j \quad (1)$$

$$y_{ij} = x_{ij} - q_{ij} \quad (2)$$

$$\begin{bmatrix} v_{ij}^{new} \\ y_{ij}^{new} \end{bmatrix} = \delta_{ij} \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} v_{ij}^{old} \\ y_{ij}^{old} \end{bmatrix} \quad (3)$$

where r denotes a constant parameter. y_{ij} denotes a relative position from the equilibrium point q_{ij} to the ith particle's position x_{ij}. δ_{ij} denotes a damping factor, and θ denotes a

rotation angle. Each particle has two search modes, convergence mode and divergence mode, by switching the damping factor δ_{ij}. The switching rule from the convergence mode to the divergence mode is given by the following.

$$\text{if } v_{ij}^{old} \cdot v_{ij}^{new} \leq 0 \text{ and } |y_{ij}^{new}| < Th_{c,ij} \tag{4}$$

$$\begin{cases} \delta_{ij} = \delta_d \\ v_{ij}^{new} = 0 \\ Th_{c,ij}^{new} = \alpha |y_{ij}^{new}| \end{cases} \tag{5}$$

where $\delta_d > 1$ denotes a damping factor in the divergence mode. Th_c denotes a switching threshold in the convergence mode, which is updated at the end of the convergence mode. α denotes a scaling parameter of Th_c.

On the other hand, the switching rule from the divergence mode to the convergence mode is given by the following.

$$\text{if } v_{ij}^{old} \cdot v_{ij}^{new} \leq 0 \text{ and } |y_{ij}^{new}| > Th_{d,ij} \tag{6}$$

$$\begin{cases} \delta_{ij} = \delta_c \\ v_{ij}^{new} = 0 \\ Th_{d,ij}^{new} = Th_{d,ij} + \beta(Th_{c,ij} - Th_{d,ij}) \end{cases} \tag{7}$$

where $0 < \delta_c < 1$ denotes a damping factor in the convergence mode. Th_d denotes a switching threshold in the divergence mode, which is updated at the end of the divergence mode. β denotes a scaling parameter of Th_d.

The procedures of PPSO algorithm for minimizing objective function $f(\mathbf{x})$ are explained bellow.

Step1: Initialization

Set the maximum iterations t_{max} and the total number of particles N. For all i, $\mathbf{x}_i$ and $\mathbf{v}_i$ are initialized at random. Let $t = 0$.

Step2: Update the best solutions

For all i, evaluate the fitness value of the ith particle and update $\mathbf{pb}_i^t$ by the following equation.

$$\mathbf{pb}_i^t = \begin{cases} \mathbf{x}_i^t & \text{, if } f(\mathbf{x}_i^t) < f(\mathbf{pb}_i^t) \text{ or } t = 0 \\ \mathbf{pb}_i^{t-1} & \text{, otherwise} \end{cases} \tag{8}$$

Then, update $\mathbf{gb}^t$ by the following equations.

$$k = \arg\min_i f(\mathbf{pb}_i^t) \tag{9}$$

$$\mathbf{gb}^t = \mathbf{pb}_k^t \tag{10}$$

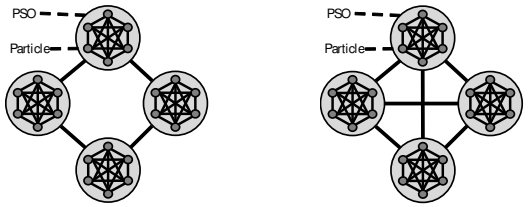
Step3: Update velocity and position vectors

For all i, the velocity and position vectors of the ith particle are updated by Eqs. (3) ~ (7).

Step4: Judgment of termination

$t = t + 1$. Then, if $t \neq t_{max}$, go to Step 2.

The idea of PPSO is very simple and it is easy to implement PPSO on digital or analog circuit. As PPSO is implemented on an analog circuit, it can be realized by nonlinear resistor, capacitor, inductor and voltage sources.



(*a*) $DBG = 2$ (*b*) $DBG = 3$

Figure 1: Examples of PSON

3. Networks of PSOs and PPSOs

3.1. PSO Networks (PSON) [4]

PSON is one of the sub-swarm PSO methods. In PSON, a population is divided into multiple sub-PSOs. Each sub-PSO is connected to neighbor sub-PSOs which are determined by network structure. The gth sub-PSO searches a search space independently, and communicates the own local best solution $\mathbf{lb}_g = (lb_{g,1}, lb_{g,2}, \ldots, lb_{g,D})$ among neighbor sub-PSOs. The gth sub-PSO has the group local best solution $\mathbf{gl}_g = (gl_{g,1}, gl_{g,2}, \ldots, gl_{g,D})$ among the neighbor sub-PSOs. Particles update velocity and position by referring to **pb**, **lb** and **gl**. Updating rules of the jth component of the ith particle's information in the gth sub-PSO are described by the following equations.

$$v_{g,ij}^{t+1} = wv_{g,ij}^t + c_1 r_1 (pb_{g,ij}^t - x_{g,ij}^t) + c_2 r_2 (lb_{g,j}^t - x_{g,ij}^t) \tag{11}$$

$$x_{g,ij}^{t+1} = x_{g,ij}^t + v_{g,ij}^{t+1} + c_3 r_3 (gl_{g,j}^t - x_{g,ij}^t) \tag{12}$$

The network topology of PSON is characterized by DBG, which denotes the degree of connection between sub-PSOs. Figure 1 shows examples of PSON. The number of sub-PSOs is 4 and the number of particles of each sub-PSO is 6 in these examples. In (a), each sub-PSO is connected to two neighbor sub-PSOs (Ring topology), while in (b), to all other sub-PSOs (Fully connected topology). In PSON, even if one sub-PSO converges to a local optimum solution, the sub-PSO can escape from the local optimum solution by referring to **gl**. Furthermore, parallel computing can be realized by assigning single processor to each sub-PSO.

3.2. PPSO Network (PPSON)

In this section, we propose PPSO Network (PPSON). The networking method of PSON is applied to PPSO. A population of PPSO is divided into multiple sub-PPSOs which have the own best solution **lb** and group local best solution **gl**, and each sub-PPSO is connected to neighbor sub-PPSOs. Each sub-PPSO searches a search space independently, and communicates own **lb** to the neighbor sub-PPSOs. In PPSON, there are cases where sub-PPSOs do not converge to **gl** if an equilibrium point **q** is set by the gravity of **pb**, **lb** and **gl**. Because the information of **gl** involves the information of **lb**, an equilibrium point of each sub-PPSO is defined as follows.

$$q_{ij} = (1 - r) pb_{ij} + r gl_j \tag{13}$$

Table 1: Simulation settings

	PPSON	PSON	PPSO
No. of Groups (G)	10		1
No. of Particles (N)	5		50
DBG	2,9		–
Dimension (D)	50		
Iterations (t_{max})	20000		
Trials	100		

where r denotes a constant parameter.

The procedures of PPSON algorithm for minimizing objective function $f(\mathbf{x})$ are explained bellow.

Step 1: Initialization

Set the maximum iterations t_{max}, the total number of sub-PPSOs G, and the total number of particles N for each sub-PPSO. Let $t = 0$. For all i, $\mathbf{x}_i^0$ and $\mathbf{v}_i^0$ are initialized at random, and $\mathbf{pb}_i^0$ is initialized by $\mathbf{x}_i^0$. For all g, $\mathbf{lb}_g^0$ are initialized by the best $\mathbf{pb}^0$ in PPSO_g, and $\mathbf{gl}_g^0$ is initialized by $\mathbf{lb}_g^0$.

Step 2: Update velocity and position vectors

For all i, the velocity and position vectors of the ith particle are updated by Eqs. (3) ~ (7).

Step 3: Update the best solutions

For all g, PPSO_g is evaluated. For all i, the ith particle in PPSO_g is evaluated, and $\mathbf{pb}_{g,i}^t$ is updated. Then, $\mathbf{lb}_g^t$ is updated by the best $\mathbf{pb}_g^t$.

Step 4: Communication for each sub-PPSOs

Perform synchronization of all sub-PPSOs at regular iterations which are decided by a constant parameter *Period*, and each sub-PPSO sends the own **lb** to neighbor sub-PPSOs. The updating rules of the gth sub-PPSO for $\mathbf{gl}_g$ are described by the following equations.

$$k = \arg\min_{j \in n(g)} f(\mathbf{lb}_j^t) \quad (14)$$

$$\mathbf{gl}_g^t = \mathbf{lb}_k^t \quad (15)$$

where $n(g)$ denotes the gth sub-PPSO's neighbor sub-PPSOs.

Step 5: Judgment of termination

$t = t + 1$. Then, if $t \neq t_{max}$, go to Step 2.

4. Numerical experiments

In order to confirm effectiveness of PPSON, PPSON is compared with PSON and PPSO. Table 1 and Table 2 show the simulation conditions and benchmark functions of which the optimal solution is 0, respectively. We adopt the combination of parameters which leads to the best result, shown in Table 3. In PPSON and PPSO, we set $\delta_c = 0.65$, $\delta_d = 1.75$, $\alpha = 1.0$, $\beta = 0.05$, and $\theta = 41°$. Initial Th_d is a maximum range of each benchmark function, because particles should search a search space globally in an initial search stage. These parameter conditions realize that each particle can converge to the equilibrium point.

Table 2: Benchmark functions

f	Landscape	Range
$Sphere$	Separable unimodal	[-5.12,5.12]
$Rosenbrock$	Non-separable unimodal	[-2.048,2.048]
$Ridge$	Non-separable unimodal	[-64,64]
$Rastrigin$	Separable multimodal	[-5.12,5.12]
$Schwefel$	Separable multimodal	[-512,512]
$Ackley$	Separable multimodal	[-32,32]
$Griewank$	Non-separable multimodal	[-600,600]
$Schaffer$	Non-separable multimodal	[-100,100]

Table 4 shows the simulation results. In Table 4, "Mean" denotes average fitness value, and "SD" denotes the standard deviation. As shown in Table 4, the solving performances of PPSON are better than those of the others for multimodal functions. While, in non-separable unimodal functions, PSON has better solving performances than PPSON, because PSON has better local search ability than PPSON. Since each sub-PPSO can search a better solution by communicating to neighbor sub-PPSOs, the solving performances of PPSON are improved more than those of PPSO. In addition, when each sub-PPSO is assigned by single processor, high parallelism can be realized. As such, the networking method for PPSO can realize that the solving performances are improved and the calculation costs are distributed. The solving performances of PPSON for fully connected topology ($DBG = 9$) are better than for ring type topology ($DBG = 2$). Therefore, each sub-PPSO should refer to the best information in all sub-PPSOs. In addition, r should be set suitable value for problems.

5. Conclusion

In this paper, we proposed the model of networking PPSO for parallel computing. PPSO was applied to the concept of PSON, and the solving performances of PPSON were compared with PSON and PPSO. The simulation results showed that PPSON has better solving performances than PSON and PPSO for multimodal problems. In addition, when each sub-PPSO is assigned by single processor, high parallelism can be realized. Therefore, it was clear that PPSON is effective for parallel computing.

In our future works, we would like to implement PPSON algorithm on digital or analog circuit to solve engineering optimization problems.

Table 3: Parameter settings for each method

f	PPSON($DBG = 2$)	PPSON($DBG = 9$)	PSON($DBG = 2$)	PSON($DBG = 9$)	PPSO
Sphere	$r = 0.95$	$r = 0.95$	$w = 0.7$, $c_1 = 1.5$	$w = 0.7$, $c_1 = 1.5$	$r = 0.95$
	$Period = 10$	$Period = 10$	$c_2 = 1.5$, $c_3 = 0.5$	$c_2 = 1.5$, $c_3 = 0.5$	–
Rosenbrock	$r = 0.85$	$r = 0.65$	$w = 0.7$, $c_1 = 1.0$	$w = 0.9$, $c_1 = 0.5$	$r = 0.50$
	$Period = 10$	$Period = 20$	$c_2 = 1.0$, $c_3 = 0.5$	$c_2 = 1.0$, $c_3 = 0.5$	–
Ridge	$r = 0.50$	$r = 0.95$	$w = 0.7$, $c_1 = 1.5$	$w = 0.7$, $c_1 = 0.5$	$r = 0.75$
	$Period = 10$	$Period = 10$	$c_2 = 0.5$, $c_3 = 1.5$	$c_2 = 1.5$, $c_3 = 1.5$	–
Rastrigin	$r = 0.95$	$r = 0.95$	$w = 0.7$, $c_1 = 1.0$	$w = 0.5$, $c_1 = 1.5$	$r = 0.95$
	$Period = 10$	$Period = 20$	$c_2 = 1.5$, $c_3 = 0.5$	$c_2 = 1.5$, $c_3 = 0.5$	–
Schwefel	$r = 0.95$	$r = 0.95$	$w = 0.9$, $c_1 = 0.5$	$w = 0.9$, $c_1 = 0.5$	$r = 0.95$
	$Period = 10$	$Period = 20$	$c_2 = 0.5$, $c_3 = 1.0$	$c_2 = 1.0$, $c_3 = 0.5$	–
Ackley	$r = 0.95$	$r = 0.95$	$w = 0.7$, $c_1 = 1.0$	$w = 0.5$, $c_1 = 1.5$	$r = 0.85$
	$Period = 10$	$Period = 20$	$c_2 = 1.5$, $c_3 = 1.5$	$c_2 = 1.5$, $c_3 = 1.5$	–
Griewank	$r = 0.45$	$r = 0.35$	$w = 0.7$, $c_1 = 1.5$	$w = 0.7$, $c_1 = 1.0$	$r = 0.85$
	$Period = 10$	$Period = 20$	$c_2 = 1.5$, $c_3 = 0.5$	$c_2 = 1.5$, $c_3 = 1.5$	–
Schaffer	$r = 0.95$	$r = 0.95$	$w = 0.7$, $c_1 = 1.0$	$w = 0.7$, $c_1 = 1.5$	$r = 0.95$
	$Period = 10$	$Period = 20$	$c_2 = 1.5$, $c_3 = 1.0$	$c_2 = 1.5$, $c_3 = 0.5$	–

Table 4: Simulation results

f		PPSON($DBG = 2$)	PPSON($DBG = 9$)	PSON($DBG = 2$)	PSON($DBG = 9$)	PPSO
Sphere	Mean	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	SD	0.00E+00	0.00E+00	**0.00E+00**	**0.00E+00**	0.00E+00
Rosenbrock	Mean	4.60E+01	4.59E+01	**3.67E-01**	5.18E-01	4.67E+01
	SD	6.28E-01	8.63E-01	1.14E+00	1.34E+00	9.87E-01
Ridge	Mean	1.54E+02	5.52E+01	**0.00E+00**	**0.00E+00**	6.88E+02
	SD	4.62E+01	3.32E+01	0.00E+00	0.00E+00	3.40E+02
Rastrigin	Mean	5.97E-02	**0.00E+00**	8.69E+01	7.89E+01	6.59E+00
	SD	2.36E-01	0.00E+00	1.92E+01	1.81E+01	4.20E+00
Schwefel	Mean	1.09E+03	**1.02E+03**	7.41E+03	6.99E+03	1.56E+03
	SD	4.16E+02	4.92E+02	6.66E+02	8.35E+02	4.82E+02
Ackley	Mean	**0.00E+00**	**0.00E+00**	3.21E-02	5.17E-02	7.00E-05
	SD	0.00E+00	0.00E+00	1.83E-01	2.54E-01	4.30E-04
Griewank	Mean	4.59E-03	4.29E-03	**2.54E-03**	8.73E-03	7.15E-03
	SD	7.22E-03	6.81E-03	6.55E-003	1.24E-02	1.01E-02
Schaffer	Mean	6.75E-01	**4.10E-01**	5.93E+01	7.64E+01	1.93E+01
	SD	1.16E+00	9.75E-01	4.10E+01	3.34E+01	1.38E+01

Acknowledgments

This work was supported by JSPS Grant-in-Aid for JSPS Fellows Grant Number JP16J11745.

References

[1] J. Kennedy and R. Eberhart, "Particle swarm optimization," *Proc. IEEE Int. Conf. Neural Networks*, pp. 1942-1948, 1995.

[2] H. Nakano and A. Miyauchi, "Design of a processor system for particle swarm optimizers," *Proc. NOLTA*, pp. 606-609, 2015.

[3] T. Sasaki, H. Nakano, A. Miyauchi and A. Taguchi, "Analysis for basic dynamics and performances of piecewise particle swarm optimizers," *Proc. IEEE SMC*, 2016.

[4] T. Sasaki, H. Nakano, A. Miyauchi and A. Taguchi, "Improvement of the solving performance by the networking of particle swarm optimization," *IEIEC Trans. Fundamentals*, vol. E98-A, no. 8, pp. 1777-1786, 2015.