JOURNAL OF INFORMATION SCIENCE AND ENGINEERING 21, 809-818 (2005)

Short Paper

A Parallel Particle Swarm Optimization Algorithm with Communication Strategies

JUI-FANG CHANG[1], SHU-CHUAN CHU[2], JOHN F. RODDICK[3] AND JENG-SHYANG PAN[4]
[1]*Department of International Trade*
[4]*Department of Electronic Engineering*
National Kaohsiung University of Applied Sciences
Kaohsiung, 807 Taiwan
[2]*Department of Information Management*
Cheng Shiu University
Kaohsiung, 833 Taiwan
[3]*School of Informatics and Engineering*
Flinders University of South Australia
Adelaide 5001, South Australia

Particle swarm optimization (*PSO*) is an alternative population-based evolutionary computation technique. It has been shown to be capable of optimizing hard mathematical problems in continuous or binary space. We present here a parallel version of the particle swarm optimization (*PPSO*) algorithm together with three communication strategies which can be used according to the independence of the data. The first strategy is designed for solution parameters that are independent or are only loosely correlated, such as the Rosenbrock and Rastrigrin functions. The second communication strategy can be applied to parameters that are more strongly correlated such as the Griewank function. In cases where the properties of the parameters are unknown, a third hybrid communication strategy can be used. Experimental results demonstrate the usefulness of the proposed *PPSO* algorithm.

Keywords: particle swarm optimization (PSO), parallel particle swarm optimization (PPSO), communication strategies, Rosenbrock and Rastrigrin functions, Griewank function

1. INTRODUCTION

The particle swarm optimization (*PSO*) algorithm is based on the evolutionary computation technique [1-3]. *PSO* optimizes an objective function by conducting population-based search. The population consists of potential solutions, called particles, similar to birds in a flock. The particles are randomly initialized and then freely fly across the multi-dimensional search space. While flying, every particle updates its

Received August 26, 2003; revised August 9, 2004; accepted September 23, 2004.
Communicated by Chin-Teng Lin.

velocity and position based on its own best experience and that of the entire population. The updating policy will cause the particle swarm to move toward a region with a higher object value. Eventually, all the particles will gather around the point with the highest object value. *PSO* attempts to simulate social behavior, which differs from the natural selection schemes of genetic algorithms.

PSO processes the search scheme using populations of particles, which corresponds to the use of individuals in genetic algorithms. Each particle is equivalent to a candidate solution of a problem. The particle moves according to an adjusted velocity, which is based on that particle's experience and the experience of its companions. For the D-dimensional function $f(\cdot)$, the ith particle for the tth iteration can be represented as

$$X_i^t = (x_i^t(1), x_i^t(2), ..., x_i^t(D)). \tag{1}$$

Assume that the best previous position of the ith particle at the tth iteration is represented as

$$P_i^t = (p_i^t(1), p_i^t(2), ..., p_i^t(D)); \tag{2}$$

then,

$$f(P_i^t) \le f(P_i^{t-1}) \le ... \le f(P_i^1). \tag{3}$$

The velocity of the ith particle at the tth iteration, V_i^t, can be represented as

$$V_i^t = (v_i^t(1), v_i^t(2), ..., v_i^t(D)). \tag{4}$$

The *best* position amongst all the particles, G^t, from the first iteration to the tth iteration, where *best* is defined by some function of the swarm, is

$$G^t = (g^t(1), g^t(2), ..., g^t(D)). \tag{5}$$

The original particle swarm optimization algorithm can be expressed as follows:

$$V_i^{t+1} = V_i^t + C_1 \cdot r_1 \cdot (P_i^t - X_i^t) + C_2 \cdot r_2 \cdot (G^t - X_i^t), \tag{6}$$

$$X_i^{t+1} = X_i^t + V_i^{t+1},\ i = 0, 1, ..., N-1, \tag{7}$$

where N is the particle size, $-V_{\max} \le V_i^{t+1} \le V_{\max}$ ($V_{\max}$ is the maximum velocity), and r_1 and r_2 are random variables such that $0 \le r_1, r_2 \le 1$.

A modified version of the particle swarm optimizer [4] and an adaption using the inertia weight*, W, a parameter for controlling the dynamics of flying of the modified particle swarm [5], have also been presented. The latter version of the modified particle swarm optimizer can be expressed as

*Strictly speaking, the term should be simply *inertia* but we use the term as per [5].

$$V_i^{t+1} = W^t \cdot V_i^t + C_1 \cdot r_1 \cdot (P_i^t - X_i^t) + C_2 \cdot r_2 \cdot (G^t - X_i^t), \tag{8}$$

$$X_i^{t+1} = X_i^t + V_i^{t+1},\ i = 0, 1, ..., N-1, \tag{9}$$

where W^t is the inertia weight at the tth iteration, and C_1 and C_2 are factors used to control the related weighting of corresponding terms. The weighting factors, C_1 and C_2, achieve a compromise between exploration and exploitation. In this paper, the concept of parallel processing is applied to particle swarm optimization, and *Parallel Particle Swarm Optimization* (*PPSO*) is presented based on a different solution space.

2. PARALLEL PARTICLE SWARM OPTIMIZATION

Parallel processing aims to produce the same results achievable using multiple processors with the goal of reducing the run time. In this study, the spirit of the data parallelism method was utilized to create a *parallel particle swarm optimization* (*PPSO*) algorithm. The purpose of applying parallel processing to particle swarm optimization goes further than merely being a hardware accelerator. Rather, a distributed formulation is developed which gives better solutions with reduced overall computation.

It is difficult to find an algorithm which is efficient and effective for all types of problems. Our research has indicated that the performance of *PPSO* can be highly dependent on the level of correlation between parameters and the nature of the communication strategy. The mathematical form of the parallel particle swarm optimization algorithm can be expressed as follows:

$$V_{i,j}^{t+1} = W^t \cdot V_{i,j}^t + C_1 \cdot r_1 \cdot (P_{i,j}^t - X_{i,j}^t) + C_2 \cdot r_2 \cdot (G_j^t - X_{i,j}^t), \tag{10}$$

$$X_{i,j}^{t+1} = X_{i,j}^t + V_{i,j}^{t+1}, \tag{11}$$

$$f(G^t) \le f(G_j^t), \tag{12}$$

where $i = 0, 1, 2, \ldots, N_j - 1, j = 0, 1, 2, \ldots, S-1, S\ (= 2^m)$ is the number of groups (and m is a positive integer), N_j is the particle size for the jth group, $X_{i,j}^t$ is the ith particle position in the jth group at the tth iteration, $V_{i,j}^t$ is the velocity of the ith particle in the jth group at the tth iteration, G_j^t is the best position among all the particles of the jth group from the first iteration to the tth iteration, and G^t is the best position among all the particles in all the groups from the first iteration to the tth iteration.

Three communication strategies have been developed for *PPSO*. The first strategy, shown in Fig. 1, is based on the observation that if parameters are independent or are only loosely correlated, then the better particles are likely to obtain good results quite quickly. Thus, multiple copies of the best particles for all groups G^t are mutated, and these mutated particles migrate and replace the poorer particles in the other groups every R_1 iterations.

However, if the parameters of a solution are loosely correlated, the better particles in each group will tend to not obtain optimum results particularly quickly. In this case, a second communication strategy may be applied as depicted in Fig. 2. This strategy is

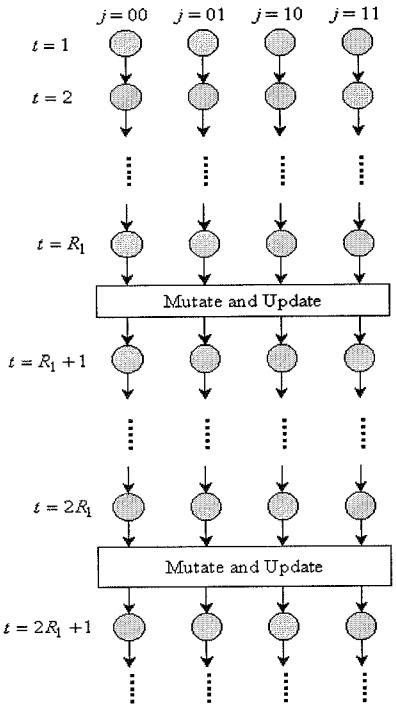

Fig. 1. Communication strategy 1 for loosely correlated parameters.

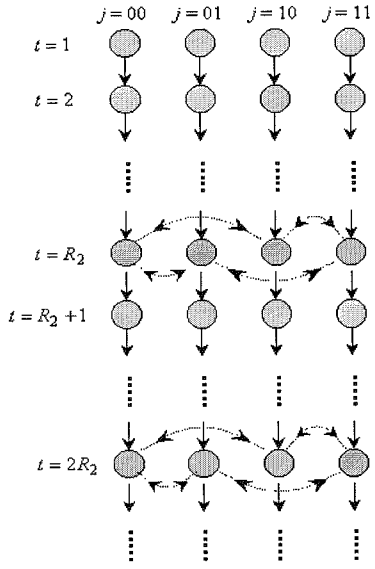

Fig. 2. Communication strategy 2 for strongly correlated parameters.

based on self-adjustment in each group. The best particle in each group G_j^t is migrated to its neighboring groups to replace some of the more poorly performing particles every R_2 iterations. Since we have defined the number of clusters S as a power of two, *neighbors* are defined as those clusters whose the binary representations j differ by one bit.

When the correlation property of the solution space is known, the first and second communication strategies work well. However, they may perform poorly if they are applied in the wrong situation. As a result, in the cases where the correlation property is unknown, a hybrid communication strategy (i.e., strategy 3) can be applied. This hybrid strategy separates the groups into two equal sized subgroups with the first subgroup applying the first strategy every R_1 iterations and all the groups applying the second strategy every R_2 iterations as depicted in Fig. 3.

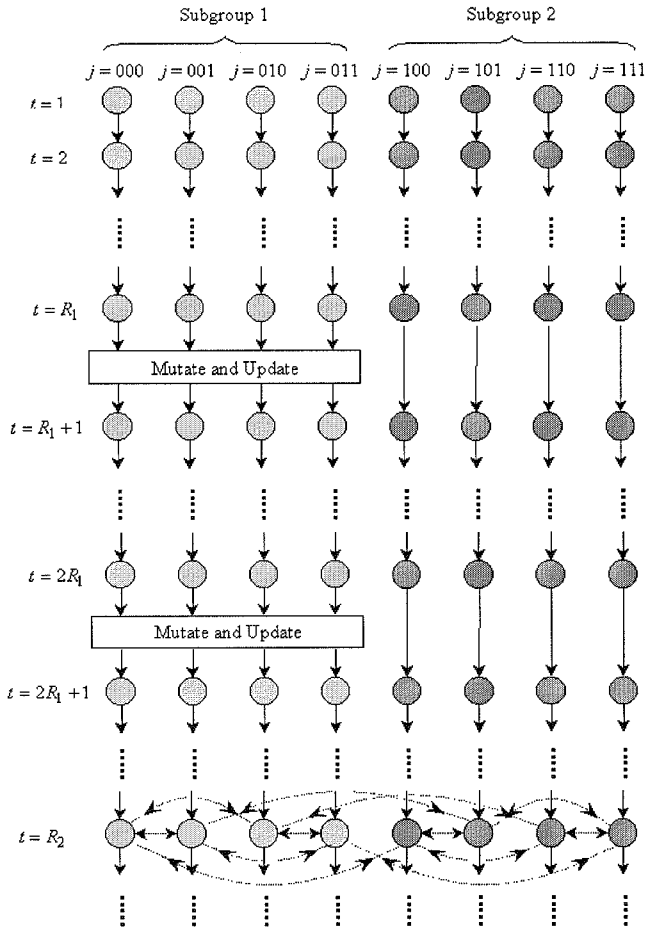

Fig. 3. General communication strategy 3 for parameters with unknown correlation.

The complete parallel particle swarm optimization (*PPSO*) algorithm with its three communication strategies is as follows:

1. **Initialization:** Generate N_j particles $X_{i,j}^t$ for the jth group, $i = 0, 1, \ldots, N_j - 1, j = 0, 1, \ldots, S - 1$, where S is the number of groups, N_j is the particle size for the jth group and t is the iteration number. Set $t = 1$.
2. **Evaluation:** Evaluate the value of $f(X_{i,j}^t)$ for every particle in each group.
3. **Update:** Update the velocity and particle positions using Eqs. (10), (11) and (12).
4. **Communication:** Three possible communication strategies are as follows:

   Strategy 1: Migrate the best particle among all the particles G^t to each group, mutate G^t to replace the poorer particles in each group and update G_j^t with G^t for each group every R_1 iterations.

   Strategy 2: Migrate the best particle position G_j^t of the jth group to its neighboring groups to replace some poorer particles every R_2 iterations.

   Strategy 3: Separate the groups into two subgroups. Apply communication strategy 1 to subgroup 1 every R_1 iterations and communication strategy 2 to both subgroup 1 and subgroup 2 every R_2 iterations.

5. **Termination:** Repeat step 2 to step 5 until the predefined value of the function is achieved or the maximum number of iterations has been reached. Record the best value of the function $f(G^t)$ and the best particle position among all the particles G^t.

3. EXPERIMENTAL RESULTS

Let $X = \{x_1, x_2, \ldots, x_n\}$ be an n-dimensional real-value vector. The Rosenbrock function is as follows:

$$f_1(X) = \sum_{i=1}^{n} (100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2). \tag{13}$$

The second function used in the experiments was the generalized Rastrigrin function, which can be expressed as

$$f_2(X) = \sum_{i=1}^{n} (x_i^2 - 10\cos(2\pi x_i) + 10). \tag{14}$$

The third function used was the generalized Griewank function:

$$f_3(X) = \frac{1}{400}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1. \tag{15}$$

Experiments were carried out to test the performance of the *PPSO* communication strategies. The results confirmed that the first strategy works best when the parameters of

the solution are loosely correlated, as in the case of the Rastrigrin and Rosenbrock functions, while the second strategy works best when the parameters of the solution are more strongly correlated, as is the case of the Griewank function. A final experiment tested the performance of the third strategy for all three functions. The results of all three experiments were compared with the linearly decreasing inertia weight *PSO* [5] for 50 runs.

The parameters of the functions for *PSO* and *PPSO* were set as shown in Table 1. We did not limit the value of X, the values of C_1 and C_2 were set to be 2, the maximum number of iterations was 2000, $W_t^0 = 0.9$, $W_t^{2000} = 0.4$, and the number of dimensions was set to be 30.

Table 1. Asymmetric initialization ranges and V_{max} values.

Function	Asymmetric Initialization Range	V_{max}
f_1	$15 \le x_i \le 30$	100
f_2	$2.56 \le x_i \le 5.12$	10
f_3	$300 \le x_i \le 600$	600

To ensure a fair comparison between *PSO* and *PPSO*, the product of the number of groups and the number of particles per group was kept constant – the particle size for the *PSO* was set to be 160. For *PPSO*, the particle size was also set to be 160; they were divided into 8 groups with 20 particles in each group (i.e., 8×20), into 4 groups with 40 particles in each group (i.e., 4×40), and into 2 groups with 80 particles in each group (i.e., 2×80), respectively. For the first experiment, the number of iterations for communication was set to be 20, and the best particle position was migrated and mutated so as to replace 25%, 50%, 75%, and 100% of the poorer particles in the receiving group. As shown in Tables 2 and 3, the first communication strategy was effective for solution parameters that were independent or loosely correlated.

For the second experiment, the number of iterations for communication was set to be 100, and the number of poorer particles replaced in each receiving group was set to be 1 or 2. The experimental results are shown in Table 4. They show that the second communication strategy for 8 groups could improve the performance by up to 66%.

Table 2. Performance comparison of *PSO* and *PPSO* based on the first communication strategy and the Rosenbrock function.

Percentage of Migration	Cost of $f_1(X)$			
	PSO	*PPSO* (2, 80)	*PPSO* (4, 40)	*PPSO* (8, 20)
None	108.739			
25%		65.384	98.558	75.948
50%		75.994	61.095	67.178
75%		61.190	64.507	59.955
100%		68.444	60.203	50.883

Table 3. Performance comparison of *PSO* and *PPSO* based on the first communication strategy and the Rastrigin function.

Percentage of Migration	Cost of $f_2(X)$			
	PSO	*PPSO* (2, 80)	*PPSO* (4, 40)	*PPSO* (8, 20)
None	24.544			
25%		16.875	17.909	16.058
50%		15.123	12.875	12.835
75%		12.877	11.183	11.024
100%		11.243	10.507	10.030

Table 4. Performance comparison of *PSO* and *PPSO* based on the second communication strategy and the Griewank function.

Number of Migration	Cost of $f_3(X)$			
	PSO	*PPSO* (2, 80)	*PPSO* (4, 40)	*PPSO* (8, 20)
None	0.01191			
1		0.01137	0.00822	0.00404
2		0.01028	0.01004	0.00601

Table 5. Performance comparison of *PSO* and *PPSO* with the third communication strategy.

Function	Cost		
	PSO	*PPSO* (4, 40)	*PPSO* (8, 20)
Rosenbrock	108.74	76.59	82.62
Rastrigin	24.54	18.63	19.14
Griewank	0.01191	0.01053	0.00989

For the third experiment, the parameters were the same as those in the first and second experiments. 50% of the particles were replaced in the receiving group with the first communication strategy, and 2 particles were replaced in the receiving group with the second communication strategy. As shown in Table 5, the hybrid communication strategy was effective for all three functions.

4. CONCLUSIONS

In this paper, a parallelized version of the particle swarm optimization scheme is presented. Three communication strategies for PPSO are discussed, which can be used according to the strength of the correlation of parameters. A third strategy is proposed in cases in which the characteristics of the parameters are unknown. Our experimental results demonstrate the usefulness of the parallel particle swarm optimization algorithm with these three communication strategies.

REFERENCES

1. R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *Proceedings of 6th International Symposium on Micro Machine and Human Science*, 1995, pp. 39-43.
2. J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of IEEE International Conference on Neural Networks*, 1995, pp. 1942-1948.
3. P. Tarasewich and P. R. McMullen, "Swarm intelligence," *Communications of the ACM*, Vol. 45, 2002, pp. 63-67.
4. Y. Shi and R. Eberhart, "A modified particle swarm optimizer," in *Proceedings of IEEE World Congress on Computational Intelligence*, 1998, pp. 69-73.
5. Y. Shi and R. Eberhart, "Empirical study of particle swarm optimization," in *Proceedings of Congress on Evolutionary Computation*, 1999, pp. 1945-1950.

Jui-Fang Chang (張瑞芳) received the Ph.D. degree in Finance from United States International University, U.S.A. in 1997. Currently she holds chairwoman of International Business Department, National Kaohsiung University of Applied Sciences, Taiwan. She has a combined academic background and research experience in Finance, International Finance, International Marketing and International Business Management. Her current research concentrates on genetic algorithms, particle swarm optimization (*PSO*), fuzzy logic, and neural networks applied in portfolio and stock market.

Shu-Chuan Chu (朱淑娟) received the B.S. degree from National Taiwan University of Science and Technology, Taiwan in 1988. She got the Ph.D. from the School of Informatics and Engineering at the Flinders University of South Australia in 2004. Currently, she is the Assistant Professor in the Department of Information Management, Cheng Shiu University. Her current research interests include soft computing, pattern recognition, and data mining.

John F. Roddick received the B.S. (Eng) (Hons) degree from Imperial College, London, the M.S. degree from Deakin University, and the Ph.D. degree from La Trobe University. He currently holds the SACITT Chair of Information Technology in the School of Informatics and Engineering at the Flinders University of South Australia. He has also held positions at the Universities of South Australia and Tasmania and was a project leader and a consultant in the information technology industry. His technical interests include data mining and knowledge discovery, schema versioning, and enterprise systems. He is editor-in-chief of the Journal of Research and Practice in Information Technology, a fellow of the Australian Computer Society and the Institution of Engineers, Australia and a member of the IEEE Computer Society and the ACM.

Jeng-Shyang Pan (潘正祥) received the B.S. degree in Electronic Engineering from the National Taiwan University of Science and Technology in 1986, the M.S. degree in Communication Engineering from the Chiao Tung University, Taiwan in 1988, and the Ph.D. degree in Electrical Engineering from the University of Edinburgh, U.K. in 1996. Currently, he is a Professor in the Department of Electronic Engineering, National Kaohsiung University of Applied Sciences, Taiwan. Professor Pan has published 40 international journal papers and 100 conference papers. He is the co-editor-in-chief of International Journal of Innovative Computing, Information and Control. His current research interests include computational intelligence, information security, and signal processing.