Hybrid Coevolutionary Particle Swarm Optimization for Linear Variational Inequality Problems

Liangdong Qu*
College of Mathematics and Computer Science, Guangxi University for Nationalities, Nanning 530006, China
Email: quliangdong@163.com

Dengxu He
College of Mathematics and Computer Science, Guangxi University for Nationalities, Nanning 530006, China
Email: dengxuhe@126.com

Yong Huang
College of Mathematics and Computer Science, Guangxi University for Nationalities, Nanning 530006, China
Email: Yonghuang@163.com

***Abstract*—Solving linear variational inequality by traditional numerical iterative algorithm not only can not satisfy parallel, but also its precision has much relationship with initial values. In this paper, a novel hybrid coevolutionary particle swarm optimization is used to solve linear variational inequality, which sufficiently exerts the advantage of particle swarm optimization such as group search, global convergence and it satisfies the question of parallel solving linear variational inequality in engineering. It also overcomes the influence of initial values. Several numerical simulation results show that the coevolutionary algorithm offers an effective way to solve linear variational inequality, high convergence rate, high accuracy and robustness.**

***Index Terms*—linear variational inequality; extragradient method; particle swarm optimization; parallel; global convergence; coevolution**

I. Introduction

Linear variational inequality problem (LVIP) is a fundamental field. It plays an important role in differential equation, mechanics, cybernetics, game theory economy and finance, transport and many other fields. Linear variational inequality problem can be described as follows:

Suppose that S is a nonempty closed convex subset of R^n, find a vector $x^* \in S$ such that the inequality

$$\langle Mx^* + q, x - x^* \rangle \geq 0, \tag{1}$$

i.e.

$$(x - x^*)^T (Mx^* + q) \geq 0, \tag{2}$$

for all $x \in S$.

Where $M = (m_{ij})_{n \times n}$ is an $n \times n$ real matrix, $q = (q_1, q_2, \cdots, q_n)^T \in R^n$, $\langle \cdot , \cdot \rangle$ denotes the inner product in R^n.

So far, this problem has been studies extensively and many methods developed for solving the problem, such as projection methods, extragradient method, and Newton methods, etc [1-3]. Some method [4] may be applied to the inequality (1) by reformulating the problem as mixed complementarity problem through its KKT system. But those approaches increase the dimension of the problem by introducing Lagrangian multipliers. Although there are plenty of methods proposed for solving the inequality (1), most of them rely on iterative methods. It usually claims parallel solving linear variational inequality problem in engineering and technology, but the iterative algorithms that based on traditional digital computer are hard to satisfy the demands of parallel solving. Furthermore, the convergence of iterative methods has much relationship with initial values.

To overcome these drawbacks, it has important actual meaningful to parallel solving the inequality (1) by studying and using particle swarm optimization.

Particle swarm optimization (PSO) [5] is a population-based, self-adaptive search optimization method motivated by the observation of simplified animal social behaviors such as fish schooling, bird flocking, etc. It is becoming very popular due to its simplicity of implementation and ability to quickly converge to a reasonably good solution [6, 7].

This paper presents a solution to solve linear variational inequality by hybrid coevolutionary particle swarm optimization. The algorithm can actualize parallel search in the feasible region and converge to the global optimal solution, which efficiently overcome the defects that the traditional methods can not parallel solve linear variational inequality in engineering technique and its precision has much relationship with initial values.

* Corresponding author: Liangdong Qu.

The rest of this paper is organized as follows: in Section 2, we give the brief theories of linear variational inequality. In Section 3, we propose extragradient method for linear variational inequality. Section 4, we introduce particle swarm optimization. In Section 5, we propose a hybrid coevolutionary particle swarm optimization for linear variational inequality. In Section 6, we report numerical results of the methods for some test problems. Finally, Section 7 gives conclusions.

II. The Theories of Linear Variational Inequality

In this paper, we select the 2-norm as the vector norm. For explicitness, we define the subset:

$$S=\{x\in R^n \mid x=(x_1,x_2,\cdots,x_n)^T, x_i\in[a_i,b_i], i=1,2,\cdots,n\},$$

where $a_i \le b_i$.

$\forall p=(p_1,p_2,\cdots,p_n)^T\in R^n$, we define the function:

$$F_S(p)=(F_{S1}(p_1),F_{S2}(p_2),\cdots,F_{Sn}(p_n))^T,$$

where

$$F_{Si}(p_i)=\begin{cases} a_i & (p_i<a_i),\\ p_i & (a_i\le p_i\le b_i),\\ b_i & (p_i>b_i),\end{cases} \quad (3)$$

Obviously, $F_{Si}(p_i)$ is a monotone continuous function whose global Lipschitz constant is 1.

Theorem 1 [8, 9]: *x^* is a solution of the linear variational inequality (1) if and only if*

$$x^*=F_S(x^*-Mx^*-q). \quad (4)$$

Theorem 2 [10]: *In S, the fixed point equation:*

$$x=F_S(x-Mx-q) \quad (5)$$

must has a solution.

III. Extragradient Method for Linear Variational Inequality

Extragradient method often been used to solve linear variational inequality.

Suppose that $\max\{|m_{ij}| \mid i=1,2,\cdots,n; j=1,2,\cdots,n\}$ is denoted by k. Korpelevich [11] introduced the following so-called extragrient method:

$$\begin{cases} x^{(0)}=x\in R^n,\\ y^{(t)}=F_S(x^{(t)}-\lambda(Mx^{(t)}+q)),\\ x^{(t+1)}=F_S(x^{(t)}-\lambda(My^{(t)}+q)) \quad t=0,1,\cdots,\end{cases} \quad (6)$$

where $\lambda\in(0,1/k)$. $x^{(t)}$ and $y^{(t)}$ refer to some vectors and t is iteration index.

He showed that the sequences $\{x^{(t)}\}$ and $\{y^{(t)}\}$ generated by (6) converge to the same point.

The advantage of the method is simple, but its disadvantage is its precision has much relationship with initial values.

IV. Particle Swarm Optimization

A. Particle Swarm Optimization and Its Improvement

Particle Swarm Optimization (PSO) is a population based algorithm that exploits a set of potential solutions to the optimization problem. It provides a method for finding the global optimum of a function of several real variables. As in the case of genetic algorithms it starts with a population of potential solutions. The population is called a *swarm* and the members are called *particles*. The particles belong to a group which may be the population as a whole or may be a subpopulation. The particles change (evolve, learn) over time in response to their own experience and the experience of the other particles in their group. As this interpretation suggests, the principles underlying the algorithm can be applied not only to the solution of optimization problems, but also to the representation of social behavior, including economic behavior.

A swarm consists of Np particles moving around in an n-dimensional search space. The *i*th particle at the *t*th iteration has a position $x_i^{(t)}=(x_{i1},x_{i2},\cdots,x_{in})^T$, a velocity $v_i^{(t)}=(v_{i1},v_{i2},\cdots,v_{in})^T$, the best solution achieved so far by itself (pbest) $pb_i=(pb_{i1},pb_{i2},\cdots,pb_{in})^T$. The best solution achieved so far by the whole swarm (gbest) is represented by $pg=(pg_1,pg_2,\cdots,pg_n)^T$. The position of the *i*th particle at the next iteration will be calculated according to the following equations:

$$v_i^{(t+1)}=w\cdot v_i^{(t)}+c_1\cdot r_1\cdot(pb_i-x_i^{(t)})+c_2\cdot r_2\cdot(pg-x_i^{(t)}), \quad (7)$$

$$x_i^{(t+1)}=x_i^{(t)}+v_i^{(t+1)}, \quad (8)$$

where c_1 and c_2 are two positive constants, called cognitive learning rate and social learning rate, respectively; r_1 and r_2 are two separately generated uniformly distributed random numbers in the range [0,1]; w is inertia weight factor. PSO has received much attention. Many researchers have devoted to improving its performance in various ways and developed many interesting variations. Most variations can be roughly grouped into the following categories.

(a) The improvement depends on incorporating the new coefficient into the velocity and position equations of the PSO algorithm or rationally selecting the values of the coefficients. Angeline [12] pointed that the original version of PSO had poor local search ability. In order to overcome this disadvantage, Shi and Eberhart [13] proposed linearly decreasing weight particle swarm optimization (LDWPSO) where a linearly decreasing inertia factor was introduced into the velocity update equation of the original PSO. Because the inertia factor effectively balances the global and local search abilities of the swarm, performance of PSO is significantly improved. Clerc [14] presented constriction PSO where a constriction factor was introduced into PSO to control the magnitude of velocities. It was mathematically proved that the resulting algorithm could guarantee the convergence even without clamping the velocity. However, if the strategy of clamping the velocity was combined into the algorithm, the performance could be improved further. The constriction PSO has faster convergence than LDWPSO, but it is prone to be trapped in the local optima when multi-modal functions are presented.

(b) A key feature of PSO algorithms is social sharing information among the neighborhood. Therefore, various information sharing mechanisms were proposed to improve the performance. Kennedy [15] investigated the impacts of various neighborhood topologies and pointed out that the von Neumann topology results in superior performance. Suganthan [16] introduced a variable neighborhood operator. During the initial stages of the optimization, the neighborhood will be an individual particle itself. As the number of generations increases, the neighborhood will be gradually extended to include all particles. Mohais [17] proposed dynamically adjusted neighborhood where randomly generated, directed structures were used as the topology of the initial population and then edges of the structures were randomly migrated from one source node to another during the course of run. Liang et al. [18] presented a new learning strategy to make particles have different learning exemplars for different dimensions. Van den Bergh and Engelbrecht [19] proposed to split the solution vector into several sub-vectors which were then allocated their own swarms. Peram et al. [20] proposed to utilize the additional information of the nearby higher fitness particle that was selected according to fitness-distance-ratio (FDR) that denoted the ratio of fitness improvement over the respective distance. Baskar and Suganthan [21] proposed a novel concurrent PSO algorithm where modified PSO and FDR-PSO algorithms were simulated concurrently with frequent message passing between them. Janson [22] used dynamic hierarchy to define the neighborhood structure. He introduced an additional component, passive congregation, into the velocity update equation.

(c) The operators of other evolutionary algorithms were combined with PSO. Angeline [23] used selection operator to improve the performance of the PSO algorithm. Lovbjerg et al. [24] combined the PSO algorithm with the idea of breeding and subpopulations. Poli et al. extended PSO via genetic programming [25]. Zhang and Xie [26] introduced differential evolution operator into PSO. Lovbjerg et al [24] combined PSO, genetic algorithms and hill climbers. Various mutation operators were also incorporated into PSO [27].

(d) Some mechanisms were designed to increase the diversity in order to prevent premature convergence to local minimum. Silva et al. [28] presented a predator−prey model to maintain population diversity. Zhang et al. [29] proposed to re-initialize the velocities of all particles at a predefined extinction interval, which simulated natural process of mass extinction in the fossil record. Krink [30] proposed several collision strategies to avoid crowding of the swarm. Lovbjerg [31] used self-organized criticality (SOC) to add diversity. Riget et al. [32] introduced attractive and repulsive PSO where the two phases, attraction and repulsion, alternated during the search according to a diversity measure. Although variations of PSO have applied different strategies and parameters, all of them follow the same principle of swarm intelligence. Therefore, all variations appear similar features of social behavior. Within a swarm, individuals are relatively simple, but their collective behavior becomes quite complex. A group of particles in a swarm move around in the defined search space to find the optimum. Each particle relies on direct and indirect interaction and cooperation with other particles to determine the next search direction and step-size, so the swarm will move around and gradually converge toward the candidates of global optima or local optima. Thus the center of the swarm is probably near to the optimum. While this position changes during the search process, it can supply very useful information for capturing the optimum.

Empirical results showed the linearly decreased setting of inertia weight can give a better performance, such as linearly decreases from 1.4 to 0, and 0.9 to 0.4 through the search process [33]. In addition, the velocities of the particles are confined within $[-v_{\max}, v_{\max}]$. If an element of velocities exceeds the threshold $-v_{\max}$ or $v_{\max}$, it is set equal to the corresponding threshold.

In this paper, to improve validity of PSO, we add extragradient method into LDWPSO. They frequently exchange the best particle of the swarm during the search and then guides the whole search process.

B. Comparison to Genetic Algorithms

The PSO shares many similarities with evolutionary computation techniques such as Genetic Algorithms. Both techniques begin with a group of a randomly generated population; utilize a fitness value to evaluate the population and search for the optimum in a partially stochastic manner by updating generations. There are however important differences. Although PSO algorithm retains the conceptual simplicity of the GAs, its' evolutionary process does not create new population members from parent ones; all swarm individuals survive. The original PSO does not employ genetic operators such as crossover and mutation and particles only evolve through their social behavior. Particles' velocities are adjusted, while evolutionary individuals' positions are acted upon. In GAs chromosomes share information with each other, thus the whole population moves like one group towards an optimal area whereas in PSO the information exchange information is a one-way process since only the (local) global best provides information to members of the (sub) swarm. PSO in comparison with GAs, has less complicated operations and it is much easier to implement and apply to design problems with continuous parameters. Moreover in [34] it is shown that binary PSO outperforms GAs in terms of convergence, results and scalability of the problems at hand. In the same study it is suggested that a hybrid model combining characteristics of GA and PSO is preferable. In [35] PSO is shown to exhibit faster convergence compared to GAs.

V. Hybrid Coevolutionary Particle Swarm Optimization for Linear Variational Inequality

A. The Principle of Solving Linear Variational Inequality by Hybrid Coevolutionary Particle Swarm Optimization (HCPSO)

Many methods for solving linear variational inequality adopt the idea of reformulating the problem as a system of nonlinear equations or an optimization problem with zero global minimum value [36]. To reformulate the LVIP as an optimization problem, we usually use so-called merit functions. Any global minimum of a merit function, say θ, with zero function value is a solution of LVIP (1) and vice versa. So the LVIP (1) is equivalent to the following global optimization problem with zero minimum value:

$$\min \ \theta(x) \tag{9}$$

i.e.

$$\min \ \| x - F_S(x - Mx - q) \|_2 \tag{10}$$

s.t. $x \in S$.

Where $\|\cdot\|_2$ represents the 2-norm. Therefore, we can define the fitness function:

$$fitness = \| x - F_S(x - Mx - q) \|_2 . \tag{11}$$

Obviously, the solution x^* to the linear variational inequality is the best solution to the function optimization. Hence our task is to

minimize $\| x - F_S(x - Mx - q) \|_2$

$s.t.\ \ x \in S$.

Extragradient method are very simple and efficient, but its precision has much relationship with initial values; for PSO, it has group search, global convergence, parallel and robustness, but its efficiency is low in the latter half evolution process.

We use PSO and extragradient method not only group search, global convergence, parallel and robustness but also to improve the validity of PSO algorithm. The evolutionary algorithm is a population-based algorithm. It makes use of a fitness function to evaluate the goodness of a solution, thereby keeping the population set consisting of promising solutions. If, during the search, the population set gets stuck around a point which is not a global minimum, extragradient method direct the population set to another possible region which may contain a global solution.

We combine the two algorithms so that they complement each other, to design a hybrid coevolutionary algorithm to solve linear variational inequality.

B. The Process of Solving Linear Variational Inequality by Hybrid Coevolutionary Particle Swarm Optimization (HCPSO)

We give a new concept: memory extragrient method as follows:

Definition 1 In extragrient method, the vector x iterate l times, $l-1$ times don't output, only after l th time, its result output. It is called memory extragrient method.

Memory extragrient method as follows:

```
Function memory_extragrient( $l$ , $x$ )
    While (t<l) do
        $x^{(0)} = x;$
        $y^{(t)} = F_S(x^{(t)} - \lambda(Mx^{(t)} + q));$
        $x^{(t+1)} = F_S(x^{(t)} - \lambda(My^{(t)} + q));$
    End While
    Output the result $x^{(t+1)}$ .
```

Where l is a positive integer, is called Memory length.

Then, the coevolutionary algorithm of PSO and extragrient method for LVIP flow chart is the following Fig.1.

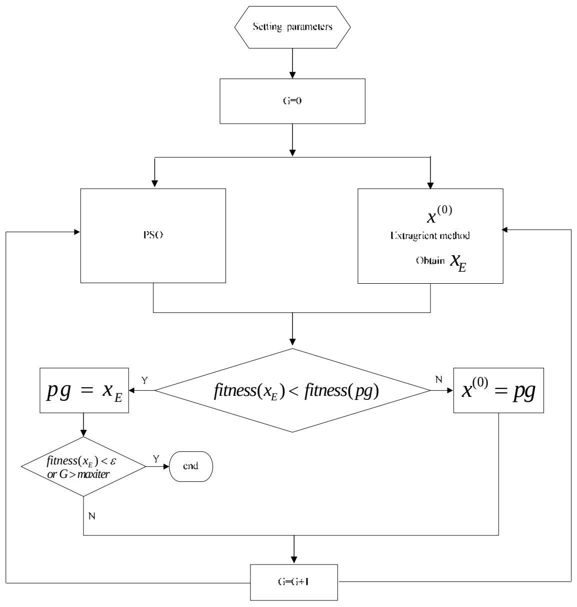

Figure 1. The flow chart of the hybrid coevolution

The steps of solving linear variational inequality by hybrid coevolutionary particle swarm optimization are illustrated as follows:

Step1. Setting evolution parameters;

Step2. Initializing the position vector and velocity vector of each component of each particle, determining the personal best location and the best location among the entire swarm; and initializing $x^{(0)}$

Step3. Calculate the weight value in current iteration according to (12):

$$w = w_{\max} - iter \cdot \frac{w_{\max} - w_{\min}}{maxiter} , \tag{12}$$

where $w_{\max}$ and $w_{\min}$ are the maximum value and minimum value for weights, respectively, $iter$ is the current iteration, and $maxiter$ is the maximum iteration;

Step4. Calculate the fitness value of all particles according to (11), and derive the best particle of current iteration, and the best position that every particle has reached so far.

Step5. If $fitness(x_E) < fitness(pg)$, then $pg = x_E$, else $x^{(0)} = pg$.

Step6. Update velocity and position of every particle according to (7) and (8).

Step7. If the stop criteria are satisfied, output the obtained best solution and the algorithm is ended; otherwise, go to Step 3.

This hybrid algorithm has the following characteristics:

(a) Structure supplementary, enhances parallelism.

The extragrient method's structure is simple, is the serial structure; PSO is the parallel structure. After mix, those enhance the entire algorithm parallelism.

(b) Behavior supplementary, raises the convergence speed.

The extragrient method iterates quickly, but it hasn't the population information, easy to fall into the local minimum; The PSO iteration does not be quick, but has certain overall situation ability. After the mix, may enhance PSO the convergence rate.

(c) Weakens the stringency of parameters.

The two algorithms have some parameters, the latter experiments show that the mixed algorithm is dependent on the parameters is not strong.

VI. Simulation Experiments

Several experiments have been conducted to evaluate the performance of the proposed CHPSO. In the experiments, the number of particles of the swarm Np was set at 30 and the maximum number of iterations $maxiter$ was set at 50. Other parameters were set at $w_{\max} = 1.4$, $w_{\min} = 0$, $c_1 = 2$, $c_2 = 2$, $v_{\max} = 2$, $l = 10$ and $\lambda = 0.5$.

The experimental conditions are: Intel(R) Core (TM) 2 Duo 2.20GHz CPU, 1G Memory and Windows XP operation system. The programs are realized in Matlab7.

To decrease the impact of randomicity, 20 independent runs were performed for each test case. The best optimal solution, the west optimal solution, the average solution and between them to the exact solution were recorded, meanwhile the average error curve of each iteration in evolution of 20 independent runs for each test case were given.

Example 1: Let $q = (1, -5)^T$,

$$M = \begin{pmatrix} 1+\sqrt{2}/2 & \sqrt{2}/3 \\ -\sqrt{2}/3 & 1+\sqrt{2}/3 \end{pmatrix}$$

and $S = [1,2] \times [3,4]$. We want to find a vector $x^* \in S$ such that (1) for all $y \in S$. Its exact solution is (1, 3.718491035931836)^T. For extragrient method, its initial value $x^{(0)} = (2,3)^T$, $\lambda = 0.5$. For LDWPSO and HCPSO, 20 independent runs were performed, respectively. Table1 and Fig.2 show the results of 20 runs.

TABLE I.
EXPERIMENTAL RESULTS FOR EXAMPLE 1

Algorithm	Optimal solution x		Error $\| x - x^* \|_2$
HCPSO	west	(1.000000000000000, 3.718491035931838)^T	2.220446049250313e-015
	best	(1.000000000000000, 3.718491035931836)^T	4.440892098500626e-016
	average	(1.000000000000000, 3.718491035931835)^T	7.105427357601002e-016
LDWPSO	west	(1.000000000000000, 3.718490685079734)^T	3.508521020378908e-007
	best	(1.000000000000000, 3.718491033847485)^T	2.084351358178083e-009
	average	(1.000000000000000, 3.718491005321227)^T	4.595095624004841e-008
Extragrient method	---	(1.000000000000000, 3.718484456990989)^T	6.578940846768688e-006

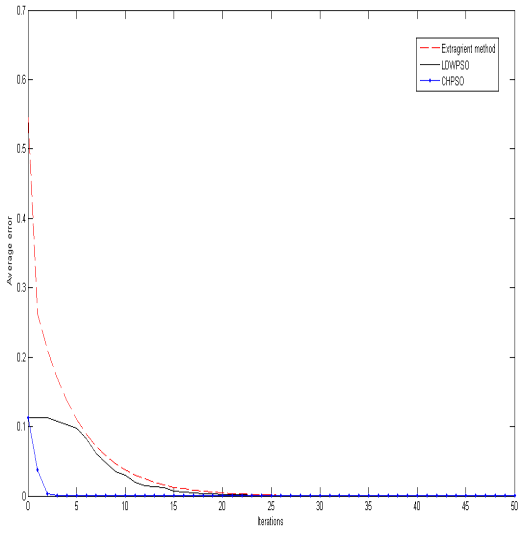

Figure 2. Average error curve for Exmple 1

Example 2: Let $q = (-1, 1, -0.5)^T$,

$$M = \begin{pmatrix} 0.1 & 0.1 & -0.5 \\ 0.1 & 0.1 & 0.5 \\ 0.5 & -0.5 & 1 \end{pmatrix} \text{ and}$$

$S = [-5,5] \times [-3,1] \times [-4,3]$.

We want to find a vector $x^* \in S$ such that (1) for all $y \in S$. Its exact solution is (2.5, -2.5, -2)^T. For extragrient method, its initial value $x^{(0)} = (0, -1, 2)^T$. For LDWPSO and HCPSO, 20 independent runs were performed, respectively. Table2 and Fig.3 show the results of 20 runs.

TABLE II.
EXPERIMENTAL RESULTS FOR EXAMPLE 2

Algorithm	Optimal solution x		Error $\| x - x^* \|_2$
HCPSO	west	(2.499999999999998, -2.500000000000003, -2.000000000000000)^T	3.20237283398937 7e-015
	best	(2.499999999999998, -2.500000000000003, -2.000000000000000)^T	3.20237283398937 7e-015
	average	(2.499999999999998, -2.500000000000003, -2.000000000000000)^T	3.20237283398937 7e-015
LDWPSO	west	(2.193347775682247, -3.000000000000000, -2.064455381986260)^T	0.59007633654131 6
	best	(2.499550187663737, -2.499651619860490, -1.999819645112649)^T	5.96848175712219 3e-004
	average	(2.470433713720782, -2.549864350200948, -2.007069394971227)^T	0.06261088852596 1
Extragrient method	---	(2.495522823984001, -2.504477906996408, -1.999998078513037)^T	0.00633220023679 2

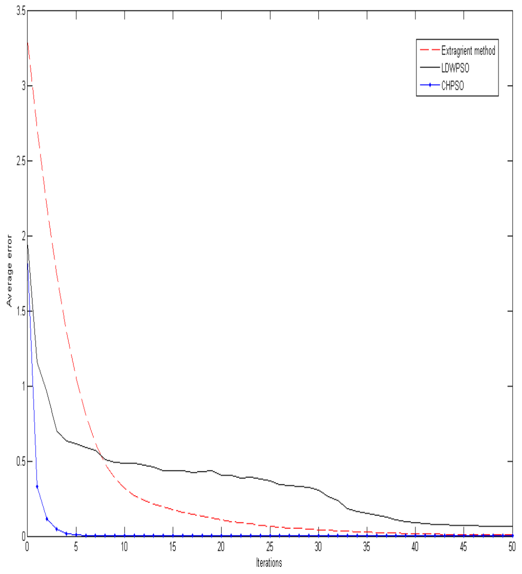

Figure3. Average error curve for Exmple 2

Example 3: Let $q = (-1, -1, -1, -1, -1, -1, -1, -1)^T$,

$$M = \begin{pmatrix} 1 & 2 & 2 & 2 & 2 & 2 & 2 & 2 \\ 0 & 1 & 2 & 2 & 2 & 2 & 2 & 2 \\ 0 & 0 & 1 & 2 & 2 & 2 & 2 & 2 \\ 0 & 0 & 0 & 1 & 2 & 2 & 2 & 2 \\ 0 & 0 & 0 & 0 & 1 & 2 & 2 & 2 \\ 0 & 0 & 0 & 0 & 0 & 1 & 2 & 2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \text{ and}$$

$S = [0,2] \times [0,2] \times [0,2] \times [0,2] \times [0,2] \times [0,2] \times [0,2] \times [0,2]$.

We want to find a vector $x^* \in S$ such that (1) for all $y \in S$. Its exact solution is (0, 0, 0, 0, 0, 0, 0, 1)^T. For extragrient method, its initial value $x^{(0)} = (1.5, 1.5, 1.5, 1.5, 1.5, 1.5, 1.5, 1.5)^T$. For LDWPSO and HCPSO, 20 independent runs were performed, respectively. Table3 and Fig.4 show the results of 20 runs.

TABLE III.
EXPERIMENTAL RESULTS FOR EXAMPLE 3

Algorithm	Optimal solution x		Error $\|\| x - x^* \|\|_2$
HCPSO	west	$(0, 0, 0, 0, 0, 0, 0, 1.000000000000000)^T$	4.440892098500626e-016
	best	$(0, 0, 0, 0, 0, 0, 0, 1.000000000000000)^T$	4.440892098500626e-016
	average	$(0, 0, 0, 0, 0, 0, 0, 1.000000000000000)^T$	4.440892098500626e-016
LDWPSO	west	$(0, 0, 0, 0, 0, 0, 0, 1.000000919252833)^T$	9.192528329649008e-007
	best	$(0, 0, 0, 0, 0, 0, 0, 0.999999973617297)^T$	2.638270291122069e-008
	average	$(0, 0, 0, 0, 0, 0, 0, 1.000000043209045)^T$	2.971536657903062e-007
Extragrient method	---	$(0, 0, 0, 0, 0, 0, 0, 1.000000283160828)^T$	2.831608281184117e-007

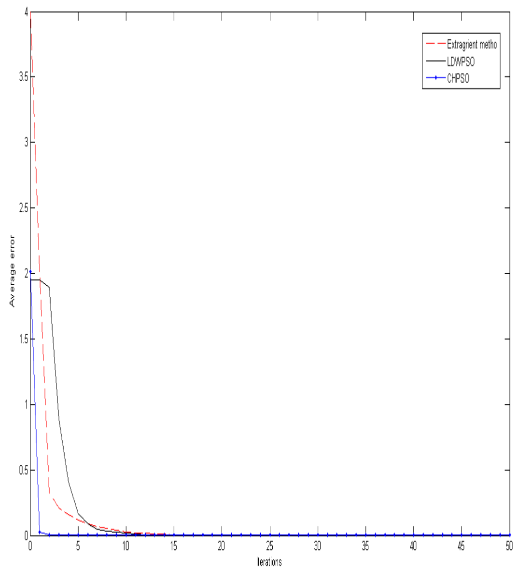

Figure4. Average error curve for Exmple 3

From Table1, Table 2, Table 3, Fig.2, Fig.3 and Fig.4, we find that the proposed algorithm enable global convergence to the best solution, has high accuracy, high convergence rate and parallelism. It is much better than LDWPSO and the extragrient method.

VII. CONCLUSIONS

In this paper, we present a novel hybrid coevolutionary algorithm for solving linear variational inequality by PSO such as group search, parallel and global convergence. This algorithm avoids the trivial deviate operation in solving linear variational inequality by traditional methods and overcome the defect that the convergence of traditional methods has much relationship with initial values, whose parallelism satisfies the question of parallel solving linear variational inequality in engineering technique. Experimental results show that the algorithm can converge to the best solution, and it has high accuracy, high convergence rate and parallelism make it easily use in engineering.

ACKNOWLEDGMENT

This work is supported by the Natural Science Foundation of Guangxi of China (No. 0728054 and 2010GXNSFB013052).

REFERENCES

[1] S. Dafermos, An iterative scheme for variational inequalities. Math. Programming, 1983, 6: 40-47.

[2] K. Taji, M. Fukushima and T. Ibaraki, A globally convergent Newton method for solving strongly monotone variational inequalities. Math. Programming, 1993, 58(3): 369-383.

[3] J.B. Jian and Y.L. Lai, Some Sorts of Methods for Solving Variational Inequalities. J. Applied Mathematics Chinese University, 1997, 14(2): 197-212. (in Chinese)

[4] Kanzow, C, Global optimization techniques for mixed complementarity problems, Journal of Global Optimization, 2000, 16: 1–21.

[5] J. Kennedy and R.C. Eberhart, Particle swarm optimization, Proceedings of IEEE International Conference on Neural Networks (1995) 1942 – 1948.

[6] H.Y. Shen, X.Q. Peng, J.N. Wang and Z.K. Hu, A mountain clustering based on improved PSO algorithm, Lecture Notes on Computer Science 3612, Changsha, China, 2005, pp. 477 – 481.

[7] R.C. Eberhart and Y. Shi, Extracting rules from fuzzy neural network by particle swarm optimization, in: Proceedings of IEEE International Conference on Evolutionary Computation, Anchorage, Alaska, USA, 1998.

[8] B.C. Eaves, On the Basic Theorem of Complementarily's. Math. Programming, 1971, 5: 68-75.

[9] U. Mosco, Introduction to Approximate Solutions for Variational Inequalities. Science and Technology of Shanghai Press, 1985.

[10] Z.G. Zeng and X.X. Liao, The solution to linear variational inequality by neural network. J. Huazhong Univ. of Sci. & Tech. (Nature Science Edition), 2002, 30(12):18-20.(in Chinese)

[11] G. M. Korpelevich, The extragradient method for finding saddle points and other problems, Matecon, 12 (1976), 747-756.

[12] P.J. Angeline, Evolutionary optimization versus particle swarm optimization: philosophy and performance differences, Lecture Notes in Computer Science, vol. 1447, Springer, Berlin, 1998, pp. 601–610.
[13] Y. Shi, R. Eberhart, A modified particle swarm optimizer, Proceedings of the IEEE Conference on Evolutionary Computation, 1998, pp. 69–73.
[14] M. Clerc, J. Kennedy, The particle swarm-explosion, stability, and convergence in a multidimensional complex space, IEEE Trans. Evol. Comput. 6 (2002) 58–73.
[15] J. Kennedy, R. Mendes, Population structure and particle swarm performance, Proceedings of the Congress on Evolutionary Computation, 2002, pp. 1671–1676.
[16] P.N. Suganthan, Particle swarm optimizer with neighbourhood operator, Proceedings of the Congress on Evolutionary Computation, 1999, pp. 1958–1962.
[17] A. Mohais, C. Ward, C. Posthoff, Randomized directed neighborhoods with edge migration in particle swarm optimization, Proceedings of the IEEE Congress on Evolutionary Computation, 2004, pp. 548–555.
[18] J.J. Liang, A.K. Qin, P.N. Suganthan, S. Baskar, Particle swarm optimization algorithms with novel learning strategies, Proceedings of IEEE Conference on Systems, Man and Cybernetics, 2004, pp. 3659–3664.
[19] F. van den Bergh, A.P. Engelbrecht, A cooperative approach to particle swarm optimization, IEEE Trans. Evol. Comput. 8 (3) (2004) 225–239.
[20] T. Peram, K. Veeramachaneni, C.K. Mohan, Fitness-distance-ratio based particle swarm optimization, Proceedings of the IEEE Swarm Intelligence Symposium, 2003, pp. 174–181.
[21] S. Baskar, P. Suganthan, A novel concurrent particle swarm optimization, Proceedings of the Congress on Evolutionary Computation, 2004, pp. 792–796.
[22] S. Janson, M. Middendorf, A hierarchical particle swarm optimizer and its adaptive variant, IEEE Trans. Syst. Man Cybern. Part B 35 (6) (2005): 1272–1282.
[23] P.J. Angeline, Using selection to improve particle swarm optimization, Proceedings of the IEEE Conference on Evolutionary Computation, 1998, pp. 84–89.
[24] M. Lovbjerg, T.K. Rasmussen, T. Krink, Hybrid particle swarm optimizer with breeding and subpopulations, Proceedings of the Third Genetic and Evolutionary Computation Conference, 2001, pp. 469–476.
[25] R. Poli, W.B. Langdon, O. Holland, Extending particle swarm optimization via genetic programming, Proceedings of the Eighth European Conference on Genetic Programming, 2005, pp. 291–300.
[26] W.J. Zhang, X.F. Xie, Depso: hybrid particle swarm with differential evolution operator, Proceedings of IEEE International Conference on Systems, Man and Cybernetics, 2003, pp. 3816–3821.
[27] N. Higashi, H. Iba, Particle swarm optimization with Gaussian mutation, Proceedings of the 2003 IEEE Swarm Intelligence Symposium, 2003, pp. 72–79.
[28] A. Silva, A. Neves, E. Costa, An empirical comparison of particle swarm and predator prey optimisation, Lecture Notes in Computer Science, vol. 2464, Springer, Berlin, 2002, pp. 103–110.
[29] W.J. Zhang, X.F. Xie, Z.L. Yang, Hybrid particle swarm optimizer with mass extinction, International Conference on Communication, Circuits and Systems, 2002, pp. 1170–1173.
[30] T. Krink, J.S. Vesterstrom, J. Riget, Particle swarm optimization with spatial particle extension, Proceedings of the Congress on Evolutionary Computation, 2002, pp. 1474–1479.
[31] M. Lovbjerg, T. Krink, Extending particle swarm optimisers with self-organized criticality, Proceedings of the Congress on Evolutionary Computation, 2002, pp. 1588–1593.
[32] J. Riget, J.S. Vesterstrom, A diversity-guided particle swarm optimizer—the ARPSO, Technical Report 2002-02, EVALife, Department of Computer Science, University of Aarhus, 2002.
[33] Y. Shi and R.C. Eberhart, A modified particle swarm optimizer, in: Proceedings of the IEEE International Conference on Evolutionary Computation, Anchorage, Alaska, USA, 1998, pp. 69–73.
[34] R. C. Eberhart, Y. Shi, "Comparison between Genetic Algorithms and Particle Swarm Optimization", in Evolutionary Programming VII: Proceedings of the Seventh Annual Conference on Evolutionary Programming, pp. 611–616, 1998.
[35] R. Hassan, B. K. Cohanim, O. D. Weck, G. A. Venter, "A Comparison of particle swarm optimization and the genetic algorithm", in Proceedings of the 1st AIAA Multidisciplinary Design Optimization Specialist Conference, April 2005.
[36] Kanzow, C. (2000), Global optimization techniques for mixed complementarity problems, Journal of Global Optimization, 16, 1–21.

Liangdong Qu He received the B.S. degree in mathematics from Henan Normal University, Xinxiang, China, in 2000. The M.S. degree in computer science from Guangxi University for Nationalities, Nanning, China, in 2009.

His current research interests focuses on computation intelligence and application, neural networks, algorithm design and analysis.

Dengxu He Professor. He received the B.S. degree in mathematics from Xihua Normal University, Nanchong, China, in 1987. The M.S. degree in computer science from Chongqing University, Chongqing, China, in 1990.

His current research interests focuses on computation intelligence and application, algorithm design and analysis.

Yong Huang He received the B.S. and M.S. degrees from Nanjing University of Science and Technology, Nanjing, China, in 2001 and 2004, respectively.The Ph.D. degree in computer science from Zhejiang University, Hangzhou, China, in 2009.

His research activity mainly focuses on information security, formal methods, computation intelligence algorithm and trusted software development.