A Novel Velocity Reinforced Mechanism on Improving Particle Swarm Optimization for Ill-conditioned Problems

Fengyang Sun#, Chunxiuzi Liu#, Linping Wu, Lin Wang*, Shuangrong Liu and Bo Yang
Shandong Provincial Key Laboratory of Network Based Intelligent Computing
University of Jinan
Jinan 250022, China
* Corresponding Author: wangplanet@gmail.com

***Abstract*—Particle swarm optimization (PSO) in recent years has been widely applied to solve various real world problems. However, for ill conditioned problems with largely different sensitivity to the objective function, classical PSO cannot search for optimal solution efficiently due to the best position-guided strategy that wastes lots of source searching undesirable areas. Therefore, this paper proposes a novel velocity reinforced mechanism (VR) for solving ill-conditional problems. Two implementations of the mechanism, velocity reinforced particle swarm optimization and velocity reinforced search, are introduced in this paper. VR updates its velocity by learning and correcting best velocity directly, instead of using classical best position-guided updating rules. In this way, it increases the possibility that finds better directions for ill-conditional problems. Experiments indicate that the novel approaches improve the final results and efficiency.**

***Keywords*—Particle Swarm Optimization; Velocity Reinforced; Best Velocity; Ill Condition; Local Search**

I. INTRODUCTION

The ill conditional property of an optimization problem, generally concerns that the sensitivity of function value changing with respect to different variables or search directions is distinct [1]. Formally, the degree of ill conditional property is specially defined as the condition number of convex quadratic function's Hessian matrix. Given $f(x) = \frac{1}{2}x^T Hx$, the condition number of H is equal to the ratio between the maximum and minimum eigenvalue of H, where Hessian matrix H is symmetric positive definite [1], [2]. Therefore, an ill conditional problem means a convex function with extremely high condition number on Hessian matrix. In addition, we generally call any function where small solution displacement instills large function value changing as ill conditional function. For instance, given function $f = 0.001x_1 + 10000x_2$, the same displacement of 1 can trigger function value change of 0.001 working on x_1 dimension, but f changing of 10000 on x_2 dimension. The sensitivity distinction can be 10^7 times! The sensitivity distinction forces the optimizer to carefully control the step sizes over different dimensions or search directions. The ill conditional problems are a class of common problems in scientific research and technological development, and the large sensitivity distinction causes the optimization difficulty [3]–[5].

In recent years, particle swarm optimization (PSO) [6] has been used as a powerful practical optimization tool for real world problems due to concision and convenience. Inspired by foraging of birds, each particle in classical PSO updates its own search direction by learning the global historical best position and its own historical best position and iteratively searching global optima of the problems. Currently Because of the easily understandable mechanism and relatively simple implementation, quantities of improvements of PSO have emerged and widely applied in various fields [7]–[10].

Some studies indicate that PSO in classical form, however, cannot search for best solution on ill conditional problems efficiently [3]. On the one hand, when the search agent is around the best region, the intrinsic conditional property that the sensitivity of function values on different dimensions or directions varies a lot makes it naturally difficult to find the right search direction efficiently. On the other hand, the classical strategies of PSO [10] (and most of the improved strategies, to the best of our knowledge) adopt best position-guided velocity updating rules. The identical distribution of random variables results in approximately linear updating rules, in other word, linear invariance [3]. The difference between linear invariance and rotation invariance is that, the former prefers to search in a long narrow valley and seems like a line search. Especially in high dimensional space, the linear invariance could result in that the algorithm converges in a subspace formed by part of dimensions while other dimensions remaining unchanged in the process [3]. The degeneration on search diversity induce relatively low possibility finding promising search directions on such kind of ill-conditioned problems, wasting lots of source repeatedly searching undesirable areas. Therefore, the population of classical PSO usually stagnates in the area far from the global optimum, leading to premature convergence or slow convergence.

The question that arises here is whether we can increase diversity on search directions to break the limitation of linear updating and improve the performance of classical PSO on ill-conditioned problems?

Both authors contribute equally to this article.

978-1-7281-6929-3/20/$31.00 ©2020 IEEE

To address this issue, we propose a novel velocity reinforced mechanism (VR), which updates its velocity by learning best velocities directly instead of using the best position-guided linear updating rules and increases the possibility that finds better directions when solving ill-conditioned problems. The mechanism is integrated into PSO, namely velocity reinforced particle swarm optimization (VRPSO) and the mechanism can be further designed as a new independent algorithm, namely velocity reinforced search (VRS). Inspired by reinforcement learning, the agent searches multiple potential good directions simultaneously. The direction with high fitness value or largest improvement in fitness value is implicitly rewarded by positive signal, which means in the next iteration this direction (and its surroundings) has higher probability being exploited. Correspondingly, the agent will not move if it cannot find better direction. In this way, when solving ill-conditioned problems, each direction is consistently tuned to potential global best direction and individual best direction, approaching actual best direction and global best position fast. In addition, the novel local optimizers combining global strategy can also improve the performance of PSO on multimodal problems in the future.

The remainder of this paper is organized as follows. Section II provides a review of the particle swarm optimization and its improvements. Section III describes the proposed VR. Section IV discusses and analyzes the experimental results. Section V concludes the study.

II. Related Works

PSO [10], [11] is a global optimization method based on the foraging of birds. In this method, each particle updates its own search direction by learning the global historical best position and its own historical best position and iteratively searching global optima of the problems. The velocity updating rules and sampling rules are shown in Eqs. 1 and 2.

$$\mathbf{v}_i^{t+1} = \omega \mathbf{v}_i^t + \varphi_1 \mathbf{u}_1 \circ (\mathbf{x}_{i\text{pbest}}^t - \mathbf{x}_i^t) + \varphi_2 \mathbf{u}_2 \circ (\mathbf{x}_{\text{gbest}}^t - \mathbf{x}_i^t), \quad (1)$$

$$\mathbf{x}_i^{t+1} = \mathbf{x}_i^t + \mathbf{v}_i^{t+1}, \quad (2)$$

where t is the iteration number. $\mathbf{x}_{\text{gbest}}^t$ is the historical global best position and $\mathbf{x}_{i\text{pbest}}^t$ is historical individual best position of the ith particle. $\mathbf{v}_i^t$ and $\mathbf{v}_i^{t+1}$ are the current velocity and updated velocity vectors of the ith particle. $\mathbf{x}_i^t$ and $\mathbf{x}_i^{t+1}$ are the current position and updated sampling position vectors of the ith particle. ω denotes the inertia weight,which gives better control of exploration and exploitation. φ_1 and φ_2 are acceleration constants. $\mathbf{u}_1, \mathbf{u}_2 \sim \mathscr{U}(0,1)$ are D-dimensional independent uniform random vectors and D is the dimension of optimization proble m.

These PSO improvement technologies discussed in this paper simply change the structure, which is not helpful to solve the ill posed problems [12], so we have the methods and work proposed.

Numerous studies focus on improving classical PSO in terms of velocity updating rules and population topology. Ratnaweera et al. [13] proposed a novel strategy of PSO, which dynamically changes acceleration coefficients during the search. Zhan et al. [14] designed a self-adaptive PSO algorithm that all the control factors can be automatically tuned to improve search efficiency. Kennedy and Mendes [15] designed different population topology to improve performance of PSO. Later Mendes et al. [16] further proposed the fully informed particle swarm FIPS to comprehensively collect population knowledge. Lin et al. [17] proposed a dynamic tournament topology strategy for PSO. Liang and Suganthan [18] proposed a novel dynamic mutil-population structure to utilize more information facilitating optimization. El-Abd and Kamel [19] integrated population-based incremental learning into PSO and improve the efficiency by shrinking the interval bounds. Other researchers attempted to combine different optimization mechanisms to PSO. Higashi and Iba [20] adopted Gaussian mutation to improve the diversity of classical PSO. Other improved versions of PSO are for some specific problems or features. Samma et al. [21] also proposed a reinforcement learning-based PSO but concerns more about when and how to conduct local search, which is totally different from our method. Bonyadi and Michalewicz [22] proposed a new general form of velocity updating rule for PSO that contains a user-definable function to enhance rotational invariance property. In recent years, Jadoun et al. [23] proposed a correction algorithm which can transform infeasible particles into feasible ones according to a new truncated sinusoidal function. Gupta et al. [24]controlled the operators of the PSO dynamically by modifying the cognitive and social behaviors of the swarm. Jadoun et al. [25] suggested three acceleration coefficients to better exploration with less computational.

A. Motivation

Despite the great success in the development of PSOs, all of these variants update velocity through the best position. In these algorithms, the search directions are learned from a random combination of *pbest* and *gbest* positions, which indirectly results in a case that the combined "position" can get stuck somewhere undesirable, which is a common scenario in ill condition problems that own narrow optimal valleys [26].

In this paper, the velocity reforced mechanism is proposed to improve the performance of the PSO on ill-conditional problems by learning best velocities directly. Inspired by reinforcement learning, the agent searches multiple potential good directions simultaneously and updates each possible direction by global best direction and individual best direction. In this way, VR enables consistently tuning the search directions and approaching global optimum on ill-conditional problems.

III. Velocity Reinforced Mechanism

In this section, we describe the mechanism and innovation of the velocity reinforced particle swarm optimization and compare PSO with VRPSO. Inspired by VRPSO, we further implement a velocity reinforced search method.

A. Comparison of PSO and VR

Classical PSO is guided by best position of population. *pbest* is the best position found by the particle itself and *gbest*

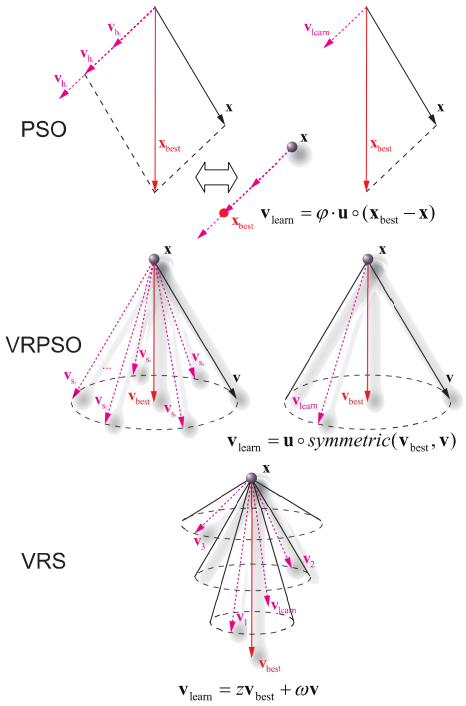

Fig. 1. A sketch plot of the velocity reinforced mechanism. In PSO, the set of $\mathbf{v}_{\text{h}}$ is on the learning direction from the current position $\mathbf{x}$ to the historical best position $\mathbf{x}_{\text{best}}$ and the learning term $\mathbf{v}_{\text{learn}}$ is relative to direction vector, where φ is a acceleration constant and $\mathbf{u}$ is a vector of uniform random numbers drawn from $[0, 1]$. In contrast, the VRPSO follows the angular learning mechanism instead of the parallelogram law or linear calculation. $\mathbf{v}_{\text{learn}}$ randomly selects one of the symmetric vectors of the current velocity $\mathbf{v}$ w.r.t historical best velocity $\mathbf{v}_{\text{best}}$ as its direction vector. The VRS further utilizes the comprehensive information of the vectors surrounded by the $\mathbf{v}_{\text{best}}$, where the set of ω control the weights of all the possible velocities and z is a acceleration factor.

is the historical global best position of the swarm. In other words, the learning term is linearly dependent with all the possible selected vectors if the random components are taken as scalars, which means the potential rich information is not utilized in a reasonable way (Fig. 1). However, the search behavior is different if the learning objectives are defined as best velocities.

By contrast, the search agent of VR corrects its velocities by learning the symmetric vector of the current vector with respect to the best velocity. The "symmetry" in this paper is defined as that the angle between the learning term and the current velocity is equal to the angle between the best velocity and the current velocity. In addition, the angle between the learning term and the current velocity can be uniformly randomly chosen. The angle between the new velocity and best velocity can also be chosen in a Gaussian random way. In this way, the agent refers to considerable information to aid the algorithm with a more valuable move by which the agent can find the optimum with a higher possibility.

B. Algorithm of VRPSO

In the VRPSO, the agent is the particle with best fitness historically. The agent sniffs multiple directions at the same

Algorithm 1: Algorithm of VRPSO

Input: The number of directions m, the step size control factor σ and Gaussian perturbation threshold P.
Output: The best solution.

```
Initialize v, x, x^t_best, v_best, best fitness f_best;
t = 0;
while termination condition has not met do
    Evaluate fitness values f for the sample points x on all
      the directions;
    Update the best position x^t_best and all the best velocities
      v_best according to the fitness values and the
      improvement in the fitness values;
    for i=1 to m do
        Calculate the symmetric vector v_s according to Eq.
          3;
        Calculate the geometric mean v_l according to Eq. 7;
        Update the velocity v^t_i according to Eq. 8;
        if the agent got stuck then
            Perturb the velocity;
        else
            Scale up the velocity;
        end
        Restrict the velocity inside the search boundary;
        Update the position x^t_i according to Eq. 9;
        Restrict the position inside the search boundary;
    end
    t++;
end
Return the best position x_best.
```

time and remembers historical best of each direction $\mathbf{v}_{\text{pbest}}$ and historical global best of all directions $\mathbf{v}_{\text{gbest}}$. At the next iteration, the agent will update each direction according to $\mathbf{v}_{\text{pbest}}$ and $\mathbf{v}_{\text{gbest}}$. $\mathbf{v}_{\text{s}}$ is defined as the symmetric vector of current velocity $\mathbf{v}$ with respect to $\mathbf{v}_{\text{gbest}}$ or $\mathbf{v}_{\text{pbest}}$ (all denoted as $\mathbf{v}_{\text{best}}$ in Fig. 1). We assume that the potential best direction is either between $\mathbf{v}_{\text{pbest}}$ and $\mathbf{v}_{\text{gbest}}$, or between $\mathbf{v}_{\text{pbest}}$ and $\mathbf{v}_{\text{s}}$. Then the velocity at the next iteration will be between the current velocity and the potential best direction. This strategy only considers the individual with best fitness, whereas the individual with the largest improvement in fitness value is also worthy being learned. To avoid unrestrained increase of velocity, we adopt Gaussian distribution and the velocity at the next iteration will also be possible between negative current direction and the potential best direction. After calculation of each next velocity, the agent will sniff along all the directions. To increase diversity of possible directions and avoid premature convergence, we use a linear Gaussian sampling strategy in each specific direction.

Instead of classical PSO updating particle velocity by best position, VRPSO learns search direction from best directions. Let $\mathbf{v}_{\text{best}}$ denotes one of the best velocities, which include global best velocity with best fitness, personal best velocity with best fitness, global best velocity with largest improvement, and personal best velocity with largest improvement. The calculation of the symmetric vector $\mathbf{v}_{\text{s}}$ requires a little trick. $\mathbf{v}_{\text{s}}$ can be defined as

$$\mathbf{v}_{\text{s}} = \mathbf{v} + \mathbf{v}^{*}, \tag{3}$$

where $\mathbf{v}$ denotes a current velocity and $\mathbf{v}^*$ is in $\mathbf{v}_{\text{best}}^{\perp}$, the orthogonal complement of $\mathbf{v}_{\text{best}}$. Then let $\mathbf{v}^* = \lambda \mathbf{v}_{\text{q}}$. $\mathbf{v}_{\text{q}}$ can be solved by matrix decomposition (in this paper we use quadrature rectangle decomposition). Given an appropriate λ, we can easily obtain the $\mathbf{v}_{\text{s}}$. To solve λ, we can use the equation derived from the definition of symmetry and the cosine law. We have

$$\frac{\mathbf{v}_{\text{s}} \cdot \mathbf{v}_{\text{best}}}{\|\mathbf{v}_{\text{s}}\| \|\mathbf{v}_{\text{best}}\|} = \frac{\mathbf{v} \cdot \mathbf{v}_{\text{best}}}{\|\mathbf{v}\| \|\mathbf{v}_{\text{best}}\|}, \tag{4}$$

or

$$\frac{(\mathbf{v} + \lambda \mathbf{v}_{\text{q}}) \cdot \mathbf{v}_{\text{best}}}{\|(\mathbf{v} + \lambda \mathbf{v}_{\text{q}})\| \|\mathbf{v}_{\text{best}}\|} = \frac{\mathbf{v} \cdot \mathbf{v}_{\text{best}}}{\|\mathbf{v}\| \|\mathbf{v}_{\text{best}}\|}, \tag{5}$$

where λ is the only unknown variable in the equation. Therefore, we can build a quadratic equation with one unknown. It has two solutions for λ, one of which is 0 (it means the symmetric vector $\mathbf{v}_{\text{s}}$ and the current velocity $\mathbf{v}$ are overlapped). The other solution for λ is

$$\lambda = \frac{-2(\mathbf{v}_{\text{q}} \cdot \mathbf{v}_{\text{best}} \|\mathbf{v}\|^2 - (\mathbf{v} \cdot \mathbf{v}_{\text{best}})^2 \mathbf{v}_{\text{q}} \cdot \mathbf{v})}{(\mathbf{v}_{\text{q}} \cdot \mathbf{v}_{\text{best}})^2 \|\mathbf{v}\|^2 - (\mathbf{v} \cdot \mathbf{v}_{\text{best}})^2 \|\mathbf{v}_{\text{q}}\|^2} \tag{6}$$

In this way, we can obtain the symmetric vector of each individual current velocity w.r.t the best velocity. Let m is the number of valuable directions (similar with population size in classical PSO). The detailed updating rules are shown in Eqs. 8 and 9.

$$\mathbf{v}_l = \frac{\mathbf{v} + \mathbf{v}_{\text{s}_l} + \mathbf{v}_{\text{best}_l}}{3}, \tag{7}$$

$$\mathbf{v}_i^{t+1} = \sigma z_1 \mathbf{v}_i^t + \sum_{l=1}^{4} \mathbf{u}_l \circ \mathbf{v}_l^t, \tag{8}$$

$$\mathbf{x}_i^{t+1} = \mathbf{x}_{\text{best}}^t + \mathbf{v}_i^{t+1}, \tag{9}$$

where $l = 1, 2, 3, 4$ indicates the aforementioned four types of best velocities. v_l is the geometric mean of the current velocity, the corresponding symmetric vector $\mathbf{v}_{\text{s}_l}$ and the corresponding best velocity $\mathbf{v}_{\text{best}_l}$. t is the iteration number. $\mathbf{u}_l$ is a D-dimensional uniform random vector drawn from $[0, 1]$. $\mathbf{v}_i^t$ is the current status of the ith direction. $\mathbf{x}_{\text{best}}^t$ is the historical best position, which is the current position of the search agent. $\mathbf{x}_i^{t+1}$ is a sample in the ith updated direction. z_1 represent random numbers drawn from a Gaussian distribution $\mathcal{N}(0,1)$. r_1 and r_2 are uniform random numbers drawn from $(0,1)$. $\sigma > 0$ is the step size control factor.

If the agent does not update its position for 2 iterations (it indicates the agent gets stuck somewhere), the velocity will further update in a Gaussian perturbation way according to the scale of the velocity. Otherwise, the velocity will scale up. The complete algorithm details of VRPSO is shown in Algorithm 1.

TABLE I
PARAMETER SETTINGS OF THE INVOLVED PSO ALGORITHMS.

Algorithms	Parameter settings
PSO [15]	Global ring, $\omega : 0.792$, $\varphi_1 = \varphi_2 = 2$, $V_{\max} = 0.2 \times Range$
PSO_Bounds [19]	Fully connected, $\omega : 0.9 - 0.1$, $\varphi_1 = \varphi_2 = 2$, $V_{\max} = Range$
DMSPSO [18]	Multi-swarm, $\omega : 0.9 - 0.2$, $\varphi_1 = \varphi_2 = 2$, $V_{\max} = 0.2 \times Range$
FIPS [16]	Local ring, $\chi = 0.7298$, $\sum \varphi = 4.1$
VRPSO	$\sigma = 0.3$
VRS	$\sigma = 1$

C. Improvement in VRS

Inspired by VRPSO, we further implement a velocity reinforced search method (Fig. 1). The agent updates the velocities by generating a group of vectors that are surrounded by the best velocity. The norm of the vector is scaled with the angle between the vector and the best velocity, in which the closer to the best velocity, the longer the vector is, and vice versa. The key formula of velocity updating rule is defined as follows.

$$\mathbf{v}_i^t = \frac{\|\mathbf{v}_{\text{best}}^t\|}{\|\mathbf{z}_i\|} \mathbf{z}_i, \tag{10}$$

$$\theta_i^t = \arccos\left(\frac{\mathbf{z}_i \cdot \mathbf{v}_{\text{best}}^t}{\|\mathbf{z}_i\| \|\mathbf{v}_{\text{best}}^t\|}\right), \tag{11}$$

$$\omega_i^t = \begin{cases} e^{-\frac{{\theta_i^t}^2}{2}}, 0 \le \theta_i^t \le \frac{\pi}{2} \\ e^{-\frac{(\pi - \theta_i^t)^2}{2}}, \frac{\pi}{2} < \theta_i^t \le \pi \end{cases}, \tag{12}$$

$$\mathbf{v}_i^{t+1} = z \mathbf{v}_{\text{best}}^t + \omega_i^t \mathbf{v}_i^t, \tag{13}$$

where $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ is a D-dimensional standard Gaussian random vector and $z \sim \mathcal{N}(0, 1)$ is a standard Gaussian random number. The other necessary components are the same with VRPSO.

IV. EXPERIMENTS

This study tries to test performance of VRPSO and VRS on ill-conditional benchmark functions. All of the 5 high conditioning functions and 1 moderate conditioning function are tested between 10, 20 and 40 dimensions. Table II shows the formulas of these functions.

A. Benchmark Functions and Compared Algorithms

Four PSO variants are compared with VRPSO and VRS. The parameter configurations for all involved algorithms are summarized in Table I. All of the PSO variants are tested with the same population size of 40 and maximum number of function evaluations $D \times 10^3$ for problems with different dimensions to ensure fairness. Each function randomly initializes 10 independent instances and each instance tests each algorithm for 10 independent repeated times. That is, each algorithm is tested 100 times independently for every function to decrease statistical errors.

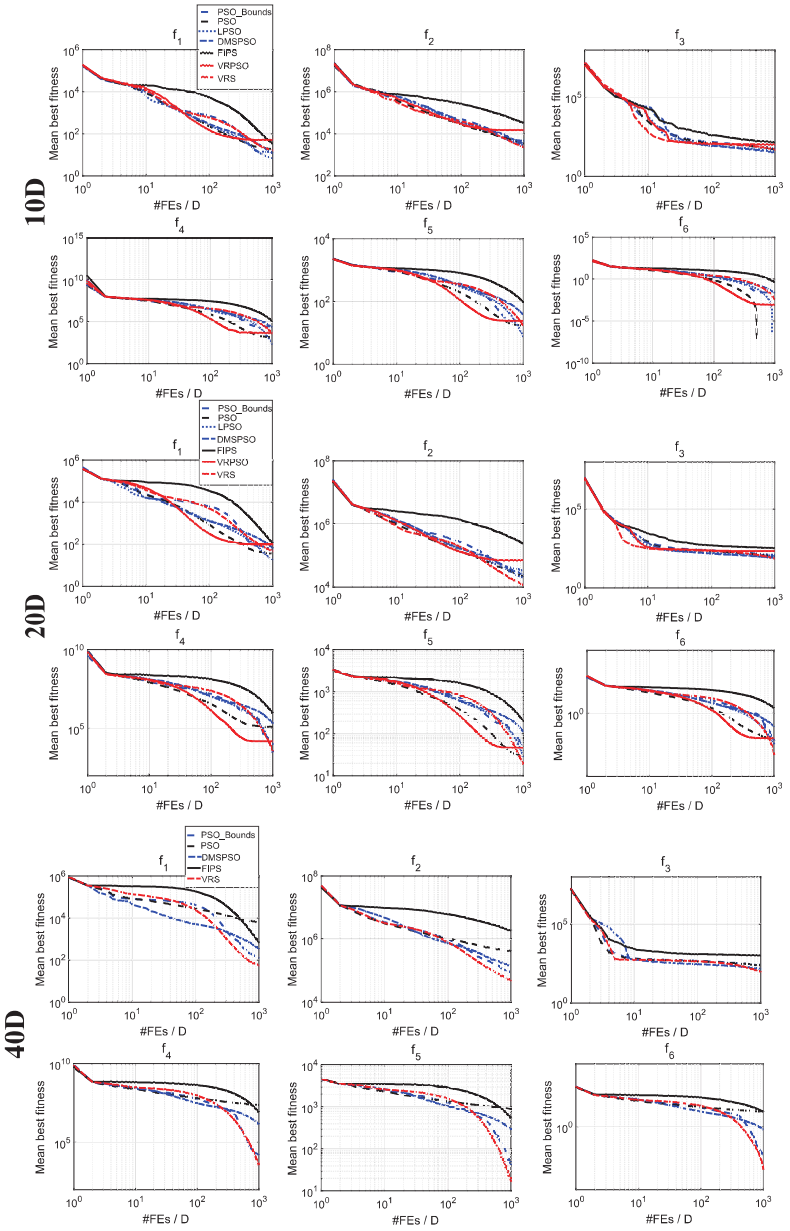

Fig. 2. The convergence plots of all the methods on 10,20 and 40 dimensional functions in logarithm scale. *#FEs* denotes the number of fitness evaluations and D is the dimension of function. Each curve is plotted with the median of best fitness values of the summarized 100 independent runs.

B. Results and Discussion

Table III and the Fig. 2 summarize the performances of all the involved PSO methods on 10,20 and 40 dimensional ill-conditioned functions. VRS does not work well on the 10D problems but ranks first on three out of six 20D functions. It is observed that on most of functions VRPSO spends less time finding the best regions compared to other methods, whereas the agent got stuck somewhere at the late phase of optimization and the best fitness value did not change due to the shrank scale and diversity of velocities. One of the reasons is the agent of VRPSO searches homogeneous directions on high dimensional problems since the velocity update is in reality a linear combination way, which results in that all velocities converge to the the best velocity, the geometric center of all the velocities, if the agent cannot find a better solution in a long time. It is worth noting that DMSPSO performs well on all the 10D functions and does not indicate premature convergence compared to the best algorithm (PSO) on several functions such as f_1 and f_4. DMSPSO adopts a dynamic multi-swarm topology to comprehensively utilize more information

TABLE II
ILL CONDITION BENCHMARK FUNCTIONS TESTED IN THIS PAPER.

Function name	Formula
Rotated Rosenbrock	$f_1 = \sum_{i=1}^{D-1} (100(z_i^2 - z_{i+1})^2 + (z_i - 1)^2) + f_{\text{opt}}$, $\mathbf{z} = \max(1, \frac{\sqrt{D}}{8})\mathbf{R}\mathbf{x} + 1/2$, $\mathbf{z}^{\text{opt}} = 1$
Ellipsoidal	$f_2 = \sum_{i=1}^{D} 10^{6\frac{i-1}{D-1}} z_i^2 + f_{\text{opt}}$, $\mathbf{z} = T_{\text{osz}}(\mathbf{R}(\mathbf{x} - \mathbf{x}^{\text{opt}}))$
Discus	$f_3 = 10^6 z_1^2 + \sum_{i=2}^{D} z_i^2 + f_{\text{opt}}$, $\mathbf{z} = T_{\text{osz}}(\mathbf{R}(\mathbf{x} - \mathbf{x}^{\text{opt}}))$
Bent Cigar	$f_4 = z_1^2 + 10^6 \sum_{i=2}^{D} z_i^2 + f_{\text{opt}}$, $\mathbf{z} = \mathbf{R}T_{\text{asy}}^{0.5}(\mathbf{R}(\mathbf{x} - \mathbf{x}^{\text{opt}}))$
Sharp Ridge	$f_5 = z_1^2 + 100\sqrt{\sum_{i=2}^{D} z_i^2} + f_{\text{opt}}$, $\mathbf{z} = \mathbf{Q}\Lambda^{10}\mathbf{R}(\mathbf{x} - \mathbf{x}^{\text{opt}})$
Different Powers	$f_6 = \sqrt{\sum_{i=1}^{D} \lvert z_i \rvert^{2+4\frac{i-1}{D-1}}} + f_{\text{opt}}$, $\mathbf{z} = \mathbf{R}(\mathbf{x} - \mathbf{x}^{\text{opt}})$

Note: All the candidate solutions $\mathbf{x}$ are transformed into $\mathbf{z}$ in both linear and nonlinear ways before input to the functions. Q and R are orthogonal rotation matrices. T_{asy}^{β} and T_{osz} are two nonlinear mappings. The optimal fitness value f_{opt} is randomly initialized for each function with all the dimensions. The search spaces for all the functions are $[-5,5]^D$. The accuracy levels for all the functions are set 10^{-6} in these experiments. The reader is referred to [2] for more implementation details.

and can continuously decrease the fitness. It implies that the hierarchical structure or multi-population topology may improve the diversity in search. Therefore, we may consider further increasing probability on diversified directions in terms of population structure for larger search space in future.

From dynamic two dimensional visualization of search process of VRPSO and VRS (not shown in this paper), we can observe that the agent searches optimum in three stages. In the first stage the agent finds the rough region of optimum in very short time. Then it gets stuck a little for correcting search directions (around $20 * D$ $\#FEs$ in Fig. 2). Finally it finishes tuning directions and fast approaches optimum. The three search stages conform the convergence process of VRPSO and VRS in Fig. 2. The agent of VRPSO and VRS sniffs multiple directions simultaneously and increases the probability to find better directions, which guide the agent to the optimum in a efficient way. We can also observe that there are rough curves that are falling and work well.

The performance results from Table III and Fig. 2 indicate that the VRPSO and VRS do not rank first on such on 10D and 20D ill-conditioned functions. Although VRPSO decreases the fitness value fast at the early stage of optimization, it gets stuck later. The VRS can continuously reduce the error yet still slower than other PSO methods finally because it spend too much time on exploring more valuable directions at first. However, VRS outperforms VRSPSO on 10D and 20D problems and VRS works better on 20D problems compared to 10D. It gives us a reasonable assumption that the VRS performs better on higher dimension problems due to its stronger exploration ability. An additional experiment on 40D functions is conducted to verify it.

The bottom part of Table III and Fig.2 summarize the performances of all the involved PSO methods on 40 dimensional ill-conditioned functions. VRS ranks first on most of the tested functions except f_4. In addition, there is a considerable difference between the fitness values of the best and the second best method. Although the convergence speed of VRS is slow at the beginning of optimization compared with other methods, it has ability to continuously search the optimum. Therefore, the diversity of search velocities is essential to the optimization process and it is worthy enhancing the exploration ability of the velocity reinforced mechanism to improvement its performance.

V. CONCLUSION

This study proposes a novel velocity reinforced mechanism (VR), which updates its velocity by learning best velocities directly and increases the possibility that finds better directions for solving ill-conditional problems. This scheme is implemented in two methods, namely velocity reinforced particle swarm optimization (VRPSO) and velocity reinforced search (VRS). The direction with high fitness value or largest improvement in fitness value and its surroundings have higher probability being exploited in the next iteration. In this way, the agent is consistently correcting its velocities and approaching actual best direction and global best position fast. Experiments have shown that the novel scheme performs promising results and is competitive to other PSO variants on ill-conditional problems.

In future, we will explore how to design step size control mechanism and further increase possibility finding better directions. In addition, we will improve the current method to search on high dimensional problems efficiently.

ACKNOWLEDGMENT

This work was supported by National Natural Science Foundation of China under Grant No. 61872419, No. 61573166, No. 61572230, No. 61873324, No. 81671785, No. 61672262, No. 61903156. Shandong Provincial Natural Science Foundation No. ZR2019MF040, No. ZR2018LF005. Shandong Provincial Key R&D Program under Grant No. 2019GGX101041, No. 2018CXGC0706, No. 2017CXZC1206. Taishan Scholars Program of Shandong Province, China, under Grant No. tsqn201812077.

REFERENCES

[1] N. Hansen, R. Ros, N. Mauny, M. Schoenauer, and A. Auger, "Impacts of invariance in search: When CMA-ES and PSO face ill-conditioned and non-separable problems," *Applied Soft Computing*, vol. 11, no. 8, pp. 5755 – 5769, 2011.

[2] N. Hansen, S. Finck, R. Ros, and A. Auger, "Real-parameter black-box optimization benchmarking 2009: Noiseless functions definitions," INRIA, Tech. Rep. RR-6829, 2009.

[3] N. Hansen, R. Ros, N. Mauny, M. Schoenauer, and A. Auger, "PSO facing non-separable and ill-conditioned problems," INRIA, Tech. Rep. RR-6447, 2008.

TABLE III
THE MEDIAN BEST FITNESS VALUES OF ALL THE METHODS ON ALL THE 10D,20D AND 40D FUNCTIONS AND THEIR RANKINGS.

	f_1	f_2	f_3	f_4	f_5	f_6	Average Rank
10D							
PSO_Bounds	8.26E+00	2.90E+03	5.48E+01	4.70E+00	9.35E+00	-5.78E-04	
rank	2	3	4	3	2	2	2.67
PSO	**7.16E+00**	2.68E+03	4.59E+01	**2.18E+00**	**8.51E+00**	**-1.00E-03**	
rank	1	2	2	1	1	1	1.33
DMSPSO	1.17E+01	3.03E+03	**3.09E+01**	2.28E+04	3.77E+01	2.35E-02	
rank	5	4	1	5	5	5	4.17
FIPS	2.92E+01	3.04E+04	1.25E+02	1.44E+05	7.86E+01	1.47E-01	
rank	6	6	6	6	6	6	6.00
VRPSO	9.31E+00	9.80E+03	9.80E+01	4.65E+00	1.24E+01	-1.09E-04	
rank	4	5	5	2	3	3	3.67
VRS	8.50E+00	**1.66E+03**	4.66E+01	2.40E+03	1.51E+01	-3.05E-05	
rank	3	1	3	4	4	4	3.17
20D							
PSO_Bounds	1.95E+01	2.30E+04	1.17E+02	1.41E+02	3.73E+01	1.27E-02	
rank	3	3	4	2	3	3	3.00
PSO	**1.78E+01**	1.87E+04	9.71E+01	**8.03E+00**	**9.69E+00**	-1.52E-04	
rank	1	2	3	1	1	2	1.67
DMSPSO	6.44E+01	2.41E+04	7.79E+01	1.92E+05	1.12E+02	1.45E-01	
rank	4	4	2	5	5	5	4.17
FIPS	1.08E+02	2.27E+05	3.11E+02	9.15E+05	1.94E+02	1.02E+00	
rank	6	6	6	6	6	6	6.00
VRPSO	7.17E+01	4.35E+04	2.09E+02	9.94E+02	4.16E+01	2.11E-02	
rank	5	5	5	3	4	4	4.33
VRS	1.95E+01	**9.44E+03**	**6.37E+01**	2.57E+03	1.61E+01	**-1.77E-04**	
rank	2	1	1	4	2	1	1.83
40D							
PSO_Bounds	7.16E+01	8.13E+04	2.54E+02	**1.50E+02**	3.83E+01	1.10E-02	
rank	2	2	4	1	2	2	2.17
PSO	6.12E+03	4.051E+05	2.35E+02	2.26E+07	9.16E+02	8.57E+00	
rank	5	4	3	5	5	5	4.50
DMSPSO	3.30E+02	1.35E+05	1.48E+02	1.27E+06	2.88E+02	6.15E-01	
rank	3	3	2	3	3	3	2.83
FIPS	6.87E+02	1.86E+06	8.97E+02	8.09E+06	5.30E+02	7.42E+00	
rank	4	5	5	4	4	4	4.33
VRS	**3.92E+01**	**4.03E+04**	**9.42E+01**	2.38E+03	**1.55E+01**	**7.00E-04**	
rank	1	1	1	2	1	1	1.17

[4] S. Saarinen, R. Bramley, and G. Cybenko, "Ill-conditioning in neural network training problems," *SIAM Journal on Scientific Computing*, vol. 14, no. 3, pp. 693–714, 1993.

[5] E. Davoodi, M. T. Hagh, and S. G. Zadeh, "A hybrid improved quantum-behaved particle swarm optimization–simplex method (iqpsos) to solve power system load flow problems," *Applied Soft Computing*, vol. 21, pp. 171 – 179, 2014.

[6] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of ICNN'95 - International Conference on Neural Networks*, 1995.

[7] H. A. Nguyen, H. Guo, and K. Low, "Real-time estimation of sensor node's position using particle swarm optimization with log-barrier constraint," *IEEE Transactions on Instrumentation and Measurement*, vol. 60, no. 11, pp. 3619–3628, 2011.

[8] S. H. Zainud-Deen, W. M. Hassen, E. M. Ali, K. H. Awadalla, and H. A. Sharshar, "Breast cancer detection using a hybrid finite difference frequency domain and particle swarm optimization techniques," in *2008 National Radio Science Conference*, 2008, pp. 1–8.

[9] J.-C. Hung, "Adaptive fuzzy-garch model applied to forecasting the volatility of stock markets using particle swarm optimization," *Information Sciences*, vol. 181, no. 20, pp. 4673 – 4683, 2011, special Issue on Interpretable Fuzzy Systems.

[10] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *Proceedings of the Sixth International Symposium on Micro Machine and Human Science. MHS'95*, 1995, pp. 39–43.

[11] Y. Shi and R. Eberhart, "A modified particle swarm optimizer," in *Proceedings of the IEEE World Congress on Computation Intelligence*, 1998, pp. 69–73.

[12] M. R. Bonyadi and Z. Michalewicz, "Particle swarm optimization for single objective continuous space problems: A review," *Evolutionary Computation*, vol. 25, no. 1, pp. 1–54, 2017.

[13] A. Ratnaweera, S. Halgamuge, and H. Watson, "Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 3, pp. 240–255, 2004.

[14] Z.-H. Zhan, J. Zhang, Y. Li, and H. S.-H. Chung, "Adaptive particle swarm optimization," *IEEE Transactions Systems, Man, Cybernetics. Part B: Cybernetics*, vol. 39, no. 6, pp. 1362–1381, 2009.

[15] J. Kennedy and R. Mendes, "Population structure and particle swarm performance," in *Proceedings of the 2002 IEEE Congress on Evolutionary Computation. CEC'02*, vol. 2, 2002, pp. 1671–1676.

[16] R. Mendes, J. Kennedy, and J. Neves, "The fully informed particle swarm: simpler, maybe better," *IEEE transactions on evolutionary computation*, vol. 8, no. 3, pp. 204–210, 2004.

[17] L. Wang, B. Yang, and J. Orchard, "Particle swarm optimization using dynamic tournament topology," *Applied Soft Computing*, vol. 48, pp. 584–596.

[18] J.-J. Liang and P. N. Suganthan, "Dynamic multi-swarm particle swarm optimizer," in *Proceedings of the 2005 IEEE Swarm Intelligence Sym-*

posium. SIS '05., 2005, pp. 124–129.

[19] M. El-Abd and M. S. Kamel, "Black-box optimization benchmarking for noiseless function testbed using PSO_Bounds," in *Proceedings of the 11th Annual Conference Companion on Genetic and Evolutionary Computation Conference: Late Breaking Papers*, ser. GECCO '09. ACM, 2009, pp. 2275–2280.

[20] N. Higashi and H. Iba, "Particle swarm optimization with gaussian mutation," in *Proceedings of IEEE Swarm Intelligence Symposium*. IEEE, 2003, pp. 72–79.

[21] H. Samma, C. P. Lim, and J. M. Saleh, "A new reinforcement learning-based memetic particle swarm optimizer," *Applied Soft Computing*, vol. 43, pp. 276 – 297, 2016.

[22] M. R. Bonyadi and Z. Michalewicz, "A locally convergent rotationally invariant particle swarm optimization algorithm," *Springer Science+Business Media New York*, 2014.

[23] V. K. Jadoun, N. Gupta, A. Swarnkar, and K. R. Niazi, "Non-convex economic load dispatch using particle swarm optimization with elevated search and addressed operators," 2015.

[24] V. K. Jadoun, N. Gupta, K. R. Niazi, A. Swarnkar, and R. C. Bansal, "Short-term non-convex economic hydrothermal scheduling using dynamically controlled particle swarm optimization," in *Third Southern African Solar Energy Conference (SASEC2015)*, 2015.

[25] V. K. Jadoun, K. Niazi, A. Swarnkar, and N. Gupta, "Variable acceleration coefficient-based particle swarm optimization for non-convex economic load dispatch problem," in *2011 IEEE PES Innovative Smart Grid Technologies - India*. IEEE, 2011, pp. 126–130.

[26] P. T. Boggs, "An algorithm, based on singular perturbation theory, for ill-conditioned minimization problems," *Siam Journal on Numerical Analysis*, vol. 14, no. 5, pp. 830–843, 1977.