Wuhan University
Journal of Natural Sciences

2024, Vol.29 No.1, 059-066

Article ID 1007-1202(2024)01-0059-08 DOI https://doi.org/10.1051/wujns/2024291059

Cite this article: WU Zhongjian, LI Junyan. Variational Data Assimilation Method Using Parallel Dual Populations Particle Swarm Optimization Algorithm[J]. *Wuhan Univ J of Nat Sci*, 2024, 29(1): 59-66.

Variational Data Assimilation Method Using Parallel Dual Populations Particle Swarm Optimization Algorithm

WU Zhongjian[1], LI Junyan[2†]

1. Detroit Green Institute of Technology, Hubei University of Technology, Wuhan 430068, Hubei, China;
2. School of Information Management, Central China Normal University, Wuhan 430079, Hubei, China

Abstract: In recent years, numerical weather forecasting has been increasingly emphasized. Variational data assimilation furnishes precise initial values for numerical forecasting models, constituting an inherently nonlinear optimization challenge. The enormity of the dataset under consideration gives rise to substantial computational burdens, complex modeling, and high hardware requirements. This paper employs the Dual-Population Particle Swarm Optimization (DPSO) algorithm in variational data assimilation to enhance assimilation accuracy. By harnessing parallel computing principles, the paper introduces the Parallel Dual-Population Particle Swarm Optimization (PDPSO) Algorithm to reduce the algorithm processing time. Simulations were carried out using partial differential equations, and comparisons in terms of time and accuracy were made against DPSO, the Dynamic Weight Particle Swarm Algorithm (PSOCIWAC), and the Time-Varying Double Compression Factor Particle Swarm Algorithm (PSOTVCF). Experimental results indicate that the proposed PDPSO outperforms PSOCIWAC and PSOTVCF in convergence accuracy and is comparable to DPSO. Regarding processing time, PDPSO is 40% faster than PSOCIWAC and PSOTVCF and 70% faster than DPSO.

Key words: parallel algorithm; variational data assimilation; dual-population particle swarm optimization algorithm; diffusion mechanism

CLC number: TP301.6

0 Introduction

Advances in numerical weather prediction represent a quiet revolution because they have resulted from a steady accumulation of scientific knowledge and technological advances over many years. Initial conditions play a pivotal role in numerical weather prediction. In variational data assimilation methods, by incorporating the influence of observational data from various time points and constrained by the forecast model, variational data assimilation can furnish improved initial values for numerical models, consequently enhancing the predictive

Received date: 2023-05-18

Foundation item: Supported by Hubei Provincial Department of Education Teaching Research Project (2016294, 2017320), Hubei Provincial Humanities and Social Science Research Project (17D033), College Students Innovation and Entrepreneurship Training Program (National) (20191050013), Hubei Province Natural Science Foundation General Project (2021CFB584), and 2023 College Student Innovation and Entrepreneurship Training Program Project (202310500047, 202310500049)

Biography: WU Zhongjian, male, Undergraduate, research direction: artificial intelligence. E-mail: 2111611303@hbut.edu.cn

† Corresponding author. E-mail: jasmineli@mails.ccnu.edu.cn

accuracy of the model [1-4]. This delineates a nonlinear optimization problem. Due to the substantial computational demand, model intricacy, and the influence of assimilation outcomes based on initial condition choice, stringent criteria exist for algorithmic convergence and accuracy [5].

Researchers have done extensive research on this subject. Tian *et al*[6] introduced the NLS-4DVar method, a unique ensemble-variational data assimilation approach leveraging "big data", showing significant performance enhancement over the traditional NLS-4DVar method without adding computational burden.

Tamang *et al*[7] presented a new variational data assimilation (VDA) approach for the formal treatment of bias in model outputs and observations. This methodology leverages the Wasserstein metric, derived from the principles of optimal mass transport, to penalize the separation between the probability histograms characterizing the analytical state and a preceding reference dataset. This reference dataset is expected to possess more significant uncertainty but exhibits lesser bias than the model and the observations.

Fabelet *et al*[8] developed an end-to-end learning approach using automatic differentiation embedded in a deep learning framework. The key novelty of the proposed physics-informed approach is to allow the joint training of the representation of the dynamical process of interest and the solver of the data assimilation problem, which may perform this joint training using supervised and unsupervised strategies. Numerical experiments on the Lorenz-63 and Lorenz-96 systems demonstrated a conventional gradient-based minimization of the variational cost, encompassing both reconstruction performance and optimization intricacy.

In Ref. [9], the primary field update method in the optimization process allowed the nonlinearity of the observation operator and numerical weather prediction model to be incorporated into the solution of the optimization problem in the incremental four-dimensional variational (4D-Var). The outer/inner models used in the incremental 4D-Var method are based on Unified Concept for Atmoshpere (ASUCA), with suitable configurations for each resolution and applied linearization. Observation operators are implemented for various observations, with unified interfaces encapsulating external simulators. Variational quality control and variational bias correction are also introduced for advanced observation handling within the variational system. Parallelization is introduced to enhance computational efficiency, including adjoint calculations.

To address the observational data from remote sensing instruments, Dennis *et al*[10] scrutinized the code that operationalizes the widely used Spline Analysis at Mesoscale Utilizing Radar and Aircraft Instrumentation (SAMURAI) technique for estimating atmospheric conditions based on a designated collection of observations. They deployed several strategies to substantially enhance the code's efficiency, encompassing adapting it for operation on typical high-performance computing (HPC) clusters, evaluating and refining its single-node performance, introducing a more efficient nonlinear optimization approach, and facilitating Graphics Processing Unit (GPU) utilization through Open Accelerators (OpenACC).

To enhance the accuracy and convergence efficiency of variational data assimilation, classic swarm intelligence algorithms such as the Particle Swarm Optimization (PSO)[11] and Genetic Algorithm (GA) have been incorporated into data assimilation practices. However, the precision and speed achieved in specific assimilation processes remain suboptimal. In the research on intelligent optimization algorithms within data assimilation, this aspect remains a focal point.

Liu *et al*[12] introduced a "prematurity" judgment mechanism, applying chaotic perturbations to the updated positions of particles to avoid falling into local optima. However, the algorithm's speed still requires improvement.

Wang *et al*[13] proposed a weight-optimizing particle swarm algorithm, which dynamically adapts to update the inertia weight, enhancing the algorithm's precision without improving its computational time. Addressing the issue of the low accuracy of PSO in specific assimilation processes and its poor robustness under noise interference, Liu *et al*[14] applied the time-varying dual compression factor PSO algorithm to variational data assimilation with discontinuous "switching" processes, establishing a novel assimilation model. Li *et al*[15] designed a parallel molecular motion theory particle swarm optimization algorithm (PMPSO). The fundamental idea is to divide the particle swarm into N subsets (with N not exceeding the number of CPU cores). Each subset undergoes simultaneous particle iteration to enhance the algorithm's processing speed. After each iteration, the elite particle data from each subset is transferred to the common section before the next iteration. In this manner, the

algorithm enhances processing speed without compromising evolutionary accuracy.

Inspired by the above, this article addresses the issue of low accuracy observed in PSO during specific assimilation processes. To enhance the assimilation precision, we apply a Dual Population Particle Swarm Optimization algorithm based on diffusion mechanisms (DPSO) in variational data assimilation. Additionally, considering the extensive computational demand, prolonged processing time, and high equipment requirements encountered while handling massive datasets, we have improved the DPSO by employing the principles of parallel computing. Consequently, we have designed a Parallel Diffusion Dual Population Algorithm (PDPSO) aimed at reducing the algorithm's processing time. This approach not only enhances the processing speed but also retains evolutionary accuracy. The PDPSO was applied to the data assimilation process. It was compared in terms of time and accuracy with the Diffusion Dual-Population Particle Swarm Optimization (DPSO), the Time-Varying Dual Compression Factor Particle Swarm Algorithm (PSOTVCF)[16], and the Dynamic Weight Particle Swarm Algorithm[17] (PSOCIWAC). Experimental results indicate that PDPSO is notably superior in convergence precision and time compared with the PSOCIWAC and the PSOTVCF. Moreover, while retaining accuracy, its processing time significantly outperforms the DPSO.

1 Governing Equation

Without loss of generality, this paper adopts the partial differential equation used in Ref. [17] that evolves only for a time as the control equation in data assimilation:

$$\begin{cases} \dfrac{\partial q}{\partial t}+a\dfrac{\partial q}{\partial l}=F(t)-gH(q-q_c), & 0\leqslant l\leqslant L, 0\leqslant t\leqslant T \\ q(t,l)\big|_{t=0}=q_0(l), & 0\leqslant l\leqslant L \\ \dfrac{\partial q(t,l)}{\partial l}\bigg|_{t=0}=0, & 0\leqslant t\leqslant T \end{cases} \tag{1}$$

where $q(t,l)>0$ denotes the specific humidity; q_c is the saturation specific humidity, called the threshold; l_i represents the horizontal direction x, y or vertical direction z; t is the time variable; $F(t)$ is the source term for other physical processes and g is the source term for parametric processes; $a=a(t,l)$ indicates the l-direction velocity, which is a given continuous function with the first-order continuous partial derivative; $q_0(l)$ is the initial specific humidity, which satisfies continuously differentiable on the interval $[0,L]$, while satisfying $\mathrm{d}q_0/\mathrm{d}l<0$; $H(q-q_c)$ is a unit step function:

$$H(q-q_c)=\begin{cases} 1, & q>q_c \\ 0, & q\leqslant q_c \end{cases} \tag{2}$$

an "on-off" switch during parameterization. The numerical pattern corresponding to (1) is:

$$\begin{cases} q_0^i=q_0(l_i), & i=0,1,\cdots,M \\ q_k^0=q_{k-1}^0[F(t_{k-1})-gH(q_{k-1}^0-q_c)]\Delta t, & 1\leqslant k\leqslant N \\ q_k^i=q_{k-1}^i-\dfrac{\Delta t}{\Delta l}a(t_{k-1},l_i)(q_{k-1}^i-q_{k-1}^{i-1}) \\ \quad +[F(t_{k-1})-gH(q_{k-1}^0-q_c)]\Delta t, & 1\leqslant k\leqslant N, 1\leqslant i\leqslant M \end{cases} \tag{3}$$

where Δl represents the space step size and i is the space grid point; Δt denotes the time step size, and $t_k=k\Delta t$, in which k is the space layer. $N=T/\Delta t$ is the total time layer in the integration process; $M+1=(T/\Delta t)+1$ is the total number of spatial discrete points.

2 PSO Algorithm Based on Diffusion Mechanism

2.1 Particle Swarm Algorithm

Let the search space be characterized by a dimensionality of D, where a particle population of N entities is established. The position and the velocity of the i-th particle in the population can be expressed as $X=(x_{i1}, x_{i2}, \cdots, x_{id})$, and $V=(v_{i1}, v_{i2}, \cdots, v_{id})$, respectively. The current optimal position searched by particle i is denoted as $p=(p_{i1}, p_{i2}, \cdots, p_{id})$, and the optimal position searched in the whole population is recorded as $P=(p_{g1}, p_{g2}, \cdots, p_{gd})$. The positions and velocities of the particles are updated iteratively according to (4):

$$v_i^d=v_i^d+A_1Y_1(p_i^d-x_i^d)+A_2Y_2(p_g^d-x_i^d) \tag{4}$$

$$x_i^d=x_i^d+v_i^d \tag{5}$$

where $i=1, 2, \cdots, N$ is the particle number; $d=1, 2, \cdots, D$ is the dimension of the particle; A_1, A_2 are the learning factors; Y_1, Y_2 are random numbers in $[0, 1]$[11].

Drawing inspiration from the diffusion phenomenon, Xu[18] proposed an innovative methodology that employs particle communication within a dual-population framework to replicate diffusion mechanics. This endeavor culminated in the development of the DPSO algorithm. The DPSO algorithm incorporates key concepts, including population temperature, particle dif-

fusion energy, and particle diffusion probability. Utilizing these fundamental ideas as a base, the basic principles of the DPSO algorithm, in conjunction with the detailed step-by-step algorithmic procedure, are comprehensively expounded upon.

2.2 Diffusion Theory

2.2.1 Diffusion energy Q

The energy that an object has due to its mechanical motion is called kinetic energy[18]. In analogy to kinetic energy, the energy consumed by a particle to overcome the work of gravitational potential energy to displace it from its initial position to any other position is called diffusion energy. It may be assumed that the mass of all particles is one single unit, and the magnitude of the diffusion energy of each particle can be defined as the sum of the squares of each dimensional component of the particle velocity vector and then squared. The energy associated with mechanical motion in an object is termed kinetic energy. Similar to kinetic energy, particles expend energy to overcome the gravitational potential energy and shift from their original positions. This process is known as diffusion energy. The mass of all particles is assumed to be uniform. For each particle, the magnitude of diffusion energy can be calculated by summing the squares of its velocity vector's dimensional components and then squaring the result.

$$Q_i = \frac{1}{2}\sum_{j=1}^{\mathrm{Dim}}\left(v_i^d\right)^2 \tag{6}$$

v_i^d is the j-th dimension component of the particle velocity vector, i is the subscript of the particle in the population, and Dim is the dimension of the particle in the search space.

2.2.2 Temperature T

Temperature is a scalar physical quantity utilized to quantify the heat content within an object[18]. It signifies the extent of molecular thermal motion's activity from a microscopic vantage point. Molecular motion theory postulates that the surface temperature of an object serves as an additional reflection of the mean kinetic energy of molecules it houses. When molecular motion is sluggish, resulting in diminished molecular kinetic energy, the object's temperature is correspondingly lower. Conversely, heightened molecular movement translates to increased molecular kinetic energy and, consequently, a higher object temperature. Temperature serves as a macroscopic manifestation of the thermal motion exhibited by the molecules comprising the object. Therefore, we regulate the temperature of the population as follows:

$$T = \sum_{i=1}^{M} Q_i / M \tag{7}$$

where M is the population size.

2.2.3 Diffusion probability P

The diffusion probability[18] of a particle is defined as follows:

$$P_i = 1 - \mathrm{e}^{-Q_i/T} \tag{8}$$

where T represents the temperature of the particle population, Q_i is the diffusion energy of particle i, and R=1 denotes the gas constant. The temperature of the current particle population and the diffusion energy of the particles determine the probability value of random diffusion of each particle. If the temperature of the population is greater than the diffusion energy of the particles, the particles will have a smaller diffusion possibility; otherwise, the particles will have a larger diffusion possibility.

2.3 Diffusion-Based PSO Algorithm

The DPSO algorithm employs dual populations, designated as populations A and B, respectively. The operations executed on both populations are identical. During each iteration of the algorithm, the diffusion energy for every particle is computed based on the velocity vector of each particle within population A(B). Subsequently, the temperature of population A(B) for the current iteration is determined, relying on the diffusion energy of all particles. Furthermore, the diffusion probability value of each particle is calculated using formula (6), generating a random number adhering to a uniform distribution. If this random number is less than the particle's diffusion probability value, the particle is placed into the diffusion pool of population A(B). Subsequently, two particles are randomly selected from the diffusion pool to generate a difference vector, thereby perturbing the global extremum within population A(B). A replacement occurs if the perturbed vector outperforms the global extremum within the other population B(A). In scenarios where the count of particles in diffusion pool A is two or more, two particles (m and n) are randomly selected as diffusion agents. These particles generate a difference vector, introducing a random perturbation to the global extremum, yielding a provisional vector. If this provisional vector is proved to superior to the global extremum of population B, a replacement is executed; otherwise, it remains unaltered. Parallelly, in cases where the particle count within diffusion pool B is two or more,

two particles (a and b) are randomly selected as diffusion agents. These particles generate a difference vector, similarly introducing a random perturbation to the global extremum, resulting in a provisional vector. If this provisional vector outperforms population A's global extremum, a replacement occurs; otherwise, it remains unchanged. Through these sequential steps, the mechanism facilitates the exchange of information and diffusion between the dual populations.

3 Optimization of the Variational Data Assimilation Model Using the DPSO Algorithm

3.1 Calculation of Fitness Function

In PSO, individuals are evaluated according to their fitness. Individuals with higher fitness are closer to the optimal solution of the objective function, and individuals with lower fitness are farther away from the optimal solution of the objective function. That is, the state with high fitness should correspond to the more optimal state of the objective function. Since variational assimilation is a minimization problem, we apply the above molecular-kinetic-theory-based particle swarm optimization algorithm to the variational assimilation problem to find the minimum value of the variational assimilation cost function, that is, the inverse relationship between the objective function value and the fitness. The variational assimilation cost function is defined as follows:

$$J(q_0)=\frac{1}{2}\int_0^T\int_0^L (q-q^{\text{obs}})^2\,\mathrm{d}l\mathrm{d}t \tag{9}$$

where q_0 belongs to the solution space:

$$S_0=\{q_0(l)|q_0(l)\in C^L_{[0,L]}, q_0(l)<0, 0<l<L; q_0(0)=0\}$$

which satisfies the physical constraints and compatibility conditions; q is the solution of q_0 by substituting into mode (1), and the discrete cost function of the corresponding formula (9) is:

$$J(q_0)=\frac{1}{2}\sum_{k=0}^{N-1}\sum_{i=0}^{M-1}[q_k^i-(q^{\text{obs}})_k^i]^2\,\Delta l\Delta t \tag{10}$$

where q_i^k is the numerical solution of mode (3), and $(q^{\text{obs}})_i^k$ is the observation at time level $t_k=k\Delta t$ and spatial grid point $l_i=i\Delta l$.

The fitness function[18] is defined as:

$$f(q_0)=\frac{1}{1+J_i(q_0)} \tag{11}$$

where

$$J_1(q_0)=\frac{1}{2}\sum_{k=0}^{N}\sum_{i=0}^{M}[q_k^i-(q^{\text{obs}})_k^i]^2\,\Delta l\Delta t \tag{12}$$

3.2 Basic Process

The calculation process of applying the dual-population particle swarm optimization algorithm based on parallel diffusion mechanism to variational data assimilation is as follows.

a) Initialize the particles and parameters in populations A and B, including velocity, acceleration, and position, and set the maximum number of iterations to 400.

b) Evaluate the data assimilation cost function of each particle in populations A and B.

c) Update the global optimal values of particles in populations A and B and the historical optimal values of particles in populations A and B, denoted by P_g^A, P_g^BP, p_i^{kA} and p_i^{kB}, respectively.

d) Calculate the diffusion energy of all particles in populations A and B according to formula (6), where v_i^d is the particle velocity, and M is the total number of particles.

e) Calculate the temperature of populations A and B according to formula (7).

f) Calculate the diffusion probability of all particles in populations A and B according to formula (8).

g) Determine whether or not each particle in populations A and B is put into the diffusion pool.

h) Enable the communication and transmission of information between different populations through these steps by exchanging difference vectors for populations according to the rules of the DPSO algorithm.

i) Output the best of the global extremum sums of populations A and B.

j) Adjust the speed and position of particles in populations A and B according to formulas (4) and (5). Y_1 and Y_2 are two random numbers from 0 to 1, and the number of iterations in the group is increased by one.

k) If the number of iterations does not reach 400, go to step b); otherwise, the DPSO algorithm ends.

4 PDPSO Algorithm for Optimizing Variational Data Assimilation

Although the dual-population algorithm based on the diffusion mechanism in the previous section improves the accuracy of variational data assimilation, it does not improve the assimilation time or the speed of

the algorithm, due to the large amount of data to be processed in variational data assimilation. To address this problem, DPSO is improved, and a parallel diffusion double population algorithm (PDPSO) is designed. The basic idea of the algorithm is to divide the particle swarm into n subsets (n is not greater than the number of CPU cores), with each subset given to a thread control while, at the same time, performing particle iteration operations in order to increase the processing speed of the algorithm[8]; the data of the head elite particle in each subset is passed to the public part after each iteration, and then the next iteration is performed, to allow information exchange between each subset. This approach adds diversity because parallel computing in the form of asynchronous communication effectively avoids accuracy degradation while enabling information exchange; in addition, since the essence of parallel algorithms is to maximize the utilization of hardware, the algorithm results can still maintain good accuracy.

PDPSO uses the dual-population particle swarm algorithm based on the diffusion mechanism to search for feasible solutions. The detailed implementations of the PDPSO algorithm are as follows:

Initialize various algorithm parameters, such as the number of groups, the maximum number of iterations 400, etc.

a) Initialize the read-write synchronization lock and create the number of threads according to the number of groups.

b) Divide the threads to randomly initialize the particles in the group and divide the population in each thread into A and B randomly and equally.

c) Calculate the data assimilation cost function of each particle. If the individual optimal solution of the particle is better than the global optimal solution in the current group, replace the global optimal solution in the group with the individual optimal solution and turn to e); otherwise, turn to g).

d) The grouping thread acquires the read-write synchronization lock.

e) If the current global optimal solution is inferior to the optimal solution within the group, replace the global optimal solution within the group with the individual optimal solution.

f) The grouping thread releases the read-write synchronization lock.

g) The grouping thread calculates the relevant parameters according to formulae (8), (9), (10) and determines whether each exchanged difference vector can replace the optimal particle of another population.

h) The grouping thread updates the particle speed according to formula (4) and the particle position according to formula (5) and adds one to the number of iterations in the group.

i) The grouping thread judges whether the thread has reached the maximum iteration number 400 set by the thread. If it reaches 400 generations, end the thread; otherwise, go to d).

j) If all grouping threads are finished, output the final result.

5 Analysis of Numerical Experiment

5.1 Simulation Environment

By using the experimental data and experimental analysis methods in Ref. [17], the time-varying dual compression factor particle swarm optimization algorithm (PSOTVCF) , the dynamic weight particle swarm optimization algorithm (PSOCIWAC) and the dual-population parallel particle swarm optimization algorithm (PDPSO) based on the diffusion mechanism are compared in terms of convergence accuracy and time, and a comparison with the algorithm before parallelization is also made with respect to the processing time. Among them, the inertia weight in the latter two particle swarm optimization algorithms decreases from 0.7 to 0.1 with a constant derivative; the acceleration factors of the time-varying dual compression factor particle swarm optimization algorithm are as follows: the first compression factor is constant, A_1=2.6, A_2=1.2; the second compression factor is in a time-varying state, A_{1N}=2.88, A_{1M}=2.68, A_{2N}=2.45, A_{2M}=1.25; the scaling factor is set to 0.5. The initialization of 200 particles was carried out for 1 000 iterations, and the assimilation was recorded every 125 generations; in this paper, a random initialization combined with empirical knowledge was used. In 200 assimilation trials, the initial guess value of the adjoint method is taken as:

$$q_{01}(l)=[0.28-0.15\sin(\frac{\pi l}{2})]\left\{1+\cos[(\frac{j}{200})\pi]\right\},\ 1\leqslant j\leqslant 200$$

where j is not only a parameter, but also represents the sequence number of an instance, and the initial particle is generated by random disturbance on each component of q_{01} (l). The specific generation method is expressed as randomly generating three random numbers r_1, r_2 and r_3

in [0, 1], $d_0=r_1-r_2$, and the initial guess value is $q_{02}(l)=d_0 r_3 q_{01}(l)$. Since the focus is on the effectiveness of different optimization algorithms in data assimilation involving switching processes, perfect observations are generated from initial observations:

$$q_0^{\text{obs}}(l)=0.28-0.26\sin(\frac{\pi l}{2})$$

In the numerical experiment, the relevant parameters in the control equation are: $a=(1+t)(1-l)$, $F(t)=A-Bt$, A=8, B=11, q_c=0.58, g=7.0, M=20, N=100, Δt=0.01, Δl=0.05. All the experiments were implemented in a MATLAB R2017a programming environment on a PC with an Intel Core i5 CPU.

5.2 Convergence Accuracy

Figure 1 shows the comparison results after 1 000 assimilation trials when the number of iterations is 400. The *x*-axis represents the number of algorithm iterations, and the *y*-axis represents the logarithm of the convergence accuracy. The smaller the value, the closer the assimilated initial value is to the observed value. It can be seen from the figure that the accuracy of the PDPSO algorithm is significantly higher than that of either the PSOCIWAC algorithm or the PSOTVCF algorithm.

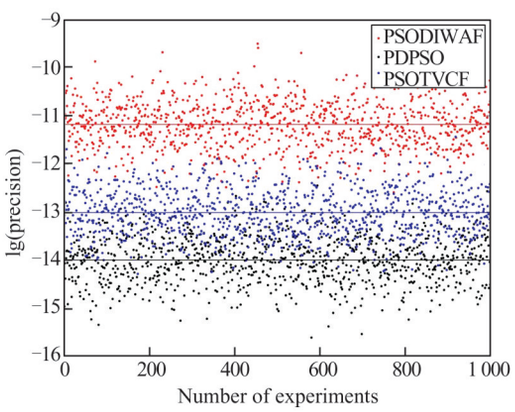

Fig. 1 Convergence accuracy comparison

Figure 2 shows the trend of convergence accuracy of the two algorithms when the number of stack iterations is 50, 100, 150, 200, 250, 300, 350, and 400, respectively. As can be seen from Fig. 2, in the early stage of assimilation (before 50 generations), the results of the three methods are approximately the same; in the middle stage of assimilation (after 100 generations), PSOTVCF takes the lead and the quality of assimilation is much higher than that of PSOCIWAC and PDPSO, and there are still particles that have not yet converged; in the late stage of assimilation (after 200 generations), both

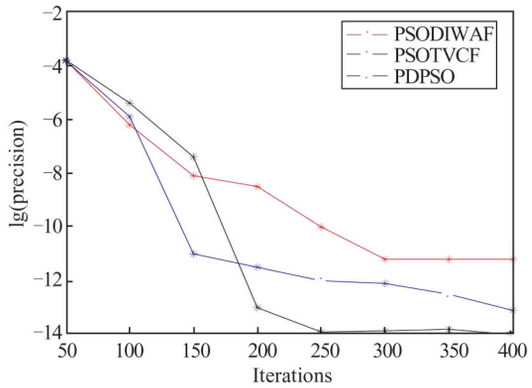

Fig. 2 Convergence accuracy change trend diagram

PSOTVCF and PSOCIWAC have basically converged, but PDPSO still presents a state of incomplete convergence. The accuracy of PDPSO has far exceeded that of the other two methods. When the number of iterations reaches 400, PSOCIWAC converges to −11.2 on average, PSOTVCF converges to −13 on average, and PDPSO converges to −14. This shows that the quality of PDPSO assimilation results is much higher than that of the dynamic weight particle swarm optimization and time-varying dual compression factor particle swarm optimization.

5.3 Assimilation Time

Table 1 shows the assimilation time spent when the number of iterations is 100, 200, 300, and 400 times, respectively. To avoid abnormal data, a total of 4 groups of assimilation tests were carried out for each assimilation window, and the values are shown in Table 1 in seconds.

From Table 1, we can see that PDPSO is always about 40% faster than PSOCIWAC and PSOTVCF, and 70% faster than DPSO during the whole iteration.

Table 1 Assimilation cost time

s

Generation	PSOCIWAC	PSOTVCF	PDPSO	DPSO
100	754.364 5	814.354 6	486.154 9	1 345.549
200	1 424.383 7	1 484.164 5	700.845 6	2 565.264
300	1 822.632 4	1 924.287 8	1 196.154 5	4 694.537
400	2 848.767 4	2 624.383 7	1 559.246 8	6 531.945

6 Conclusion

In this paper, a dual-population particle swarm optimization algorithm (PDPSO) based on the diffusion mechanism is applied to the variational data assimilation, which improves the assimilation accuracy; to ad-

dress the problem of slow speed in processing extensive data, the algorithm is improved by using the idea of parallel computing to maximize the computer hardware resources. In terms of the population update strategy, the data pertaining to the top-performing elite particles within each subset is shared with the collective after each iteration concludes. Subsequently, the ensuing iteration is executed, ensuring the integration of diversity. The proposed algorithm shows a considerable improvement in convergence accuracy and assimilation time compared with the dynamic weight PSO algorithm and the dual time-varying compression factor PSO algorithm. However, this paper solely applies the algorithm to variational data assimilation featuring a "switch" process. In the future, extending its application to include spatial evolution could amplify the complexity of the control equation.

References

[1] Lean P, Holm E V, Bonavita M, *et al*. Continuous data assimilation for global numerical weather prediction[J]. *Quarterly Journal of the Royal Meteorological Society*, 2021, **147** (734): 273-288.

[2] Ritchie H, Belair S, Bernier N B, *et al*. Recherche en prevision numerique contributions to numerical weather prediction[J]. *Atmosphere-Ocean*, 2022, **60**(1): 35-64.

[3] Schultz M G, Betancourt C, Gong B, *et al*. Can deep learning beat numerical weather prediction[J]. *Philosophical Transactions of the Royal Society A—Mathematical Physical and Engineering Sciences*, 2021, **379**(2146): 20200097-1 -20200097-22.

[4] Bauer P, Thorpe A, Brunet G. The quiet revolution of numerical weather prediction[J]. *Nature*, 2015, **525**(7567): 47-55.

[5] Dong Y Q, Zhou M R, Liu L Y, *et al*. Research on variational data assimilation based on improved parallel particle swarm optimization[J]. *Journal of Central China Normal University* (*Natural Science Edition*), 2021, **55**(1): 46-51 (Ch).

[6] Tian X J, Zhang H Q. A big data-driven nonlinear least squares four-dimensional variational data assimilation method: Theoretical formulation and conceptual evaluation [J]. *Earth and Space Science*, 2019, **6**(8): 1430-1439.

[7] Tamang S K, Ebtehaj A, Zou D M, *et al*. Regularized variational data assimilation for bias treatment using the Wasserstein metric[J]. *Quarterly Journal of the Royal Meteorological Society*, 2020, **146**(730): 2332-2346.

[8] Fablet R, Chapron B, Drumetz L, *et al*. Learning variational data assimilation models and solvers[J]. *Journal of Advances in Modeling Earth Systems*, 2021, **13**(10): e2021MS002572.

[9] Ikuta Y, Fujita T, Ota Y, *et al*. Variational data assimilation system for operational regional models at Japan meteorological agency[J]. *Journal of the Meteorological Society of Japan Ser II*, 2021, **99**(6): 1563-1592.

[10] Dennis J M, Baker A H, Dobbins B, *et al*. Enabling efficient execution of a variational data assimilation application[J]. *The International Journal of High Performance Computing Applications*, 2023, **37**(2): 101-114.

[11] Kennedy J F, Eberhart R C, Shi Y H. *Swarm Intelligence* [M]. San Francisco: Morgan Kaufmann Publishers, 2001.

[12] Liu X Y, Yao Y X, Sui Q R. Maximum power point tracking of PV based on particle swarm optimization algorithm[J]. *Laser Journal*, 2016, **37**(10): 129-132(Ch).

[13] Wang C S, Huo L M. Fault diagnosis of PV modules based on improved PSO and RBF optimization[J]. *Laser Journal*, 2019, **40**(12): 159-162(Ch).

[14] Liu Y D, Zhou M R, Xie J Y, *et al*. Research on solar radiation simulation and prediction data assimilation: Particle swarm optimization scheme[J]. *Acta Energiae Solaris Sinica*, 2021, **42**(4): 181-185(Ch).

[15] Li J Y, Tong Y L. Research on data assimilation in solar photovoltaics power generation based on improved PSO algorithm[J]. *Journal of Central China Normal University* (*Natural Sciences*), 2021, **55**(4): 567-572(Ch).

[16] Zhang C X. Particle swarm optimization based on time varying constrict factor[J]. *Computer Engineering and Applications*, 2015, **51**(23): 59-64(Ch).

[17] Zheng Q, Ye F H, Sha J X, *et al*. Effective application of particle-swarm optimization algorithm in variational data assimilation with discontinuous "on-off" switch[J]. *Meteorological Science and Technology*, 2013, **41**(2): 286-293(Ch).

[18] Xu X, Li Y X, Wu Y. Particle swarm optimization algorithm based on diffusion mechanism with dual populations [J]. *Computer Applications*, 2010, **27**(8): 2883-2885, 2898(Ch).

□