International Journal of Swarm Intelligence Research
Volume 14 • Issue 1

Dynamic Robust Particle Swarm Optimization Algorithm Based on Hybrid Strategy

Jian Zeng, Guilin Power Supply Bureau of Guangxi Power Grid Company, China*
Xiaoyong Yu, Electric Power Science Research Institute of Guangxi Power Grid Company, China
Guoyan Yang, Guilin Power Supply Bureau of Guangxi Power Grid Company, China
Haitao Gui, Guilin Power Supply Bureau of Guangxi Power Grid Company, China

ABSTRACT

Robust optimization over time can effectively solve the problem of frequent solution switching in dynamic environments. In order to improve the search performance of dynamic robust optimization algorithm, a dynamic robust particle swarm optimization algorithm based on hybrid strategy (HS-DRPSO) is proposed in this paper. Based on the particle swarm optimization, the HS-DRPSO combines differential evolution algorithm and brainstorms an optimization algorithm to improve the search ability. Moreover, a dynamic selection strategy is employed to realize the selection of different search methods in the proposed algorithm. Compared with the other two dynamic robust optimization algorithms on five dynamic standard test functions, the results show that the overall performance of the proposed algorithm is better than other comparison algorithms.

KEYWORDS

Brain Storm Optimization, Differential Evolution Algorithm, Dynamic Robust Optimization, Particle Swarm Optimization

INTRODUCTION

Optimization problems are subject to change in response to the dynamics and uncertainty of the environment, leading to what are known as dynamic optimization problems (DOPs) (Jin & Branke, 2005). Most of the current research in this area has focused on tracking moving optimization (TMO) (Parrott & Li, 2006; Chen et al., 2023; Falahiazar et al., 2022), which involves an algorithm that seeks to identify a new optimal solution after each environmental modification. Despite its effectiveness in addressing dynamic optimization problems, this approach may present some limitations in practical applications. Firstly, it may face challenges in quickly identifying the optimal solution in each dynamic environment within a limited timeframe. Secondly, even if it manages to identify the optimal solution in the new environment, it will require a considerable amount of computational resources.

DOI: 10.4018/IJSIR.325006 *Corresponding Author

Based on the aforementioned considerations, Yu et al. (2010) introduced the concept of robust optimization over time (ROOT) with the primary aim of discovering a set of robust solutions that can adapt to multiple dynamic environments both in the present and future. Following this, Jin et al. (2012) proposed a framework for tackling dynamic robust problems, which involves an optimizer, a database containing historical information, an approximator, and a predictor. Along with robustness, the ROOT approach also takes into account switching costs, which was considered in literature (Huang et al., 2017) that proposed dynamic robust optimization algorithm (robust optimization over time considering switching cost, ROOT/SC).

However, the ROOT/SC algorithm has two limitations: 1) the search dimensions cannot be expanded sufficiently, leading to considerable practical restrictions; 2) the feasible solutions from non-dominated solution sets cannot be sought using the algorithm. To address these problems, Huang et al. (2020) proposed a more efficient dynamic robust multi-objective algorithm named ROOT/SCII (improved ROOT/SC), which incorporates minimizing switching costs as an additional objective by weighing the robustness of the high-dimensional decision space and switching costs. Yazdani et al. (2019) applied multiple swarm methods to the ROOT problem, using multi-swarm PSO to identify and track optimal values while collecting information about peaks in the decision space over time, which was used to select the next robust solution. Moreover, most dynamic robust algorithms employ prediction models to solve ROOT problems; however, the accuracy of such models in practical applications is dependent on the availability of data. In addition, for dynamic problems with high-dimensional search spaces and high change frequencies, a large amount of data is often required to obtain accurate predictions. Consequently, Yazdani et al. (2017) proposed a new ROOT framework that eliminates the original predictor in ROOT (Jin et al., 2012) and substitutes the prediction of future fitness values with the prediction of future behavior of peaks, using the behavioral information of peaks to predict robust feasible solutions that satisfy the future dynamic environment when the resulting feasible solution does not satisfy the dynamic environment.

It has been demonstrated that the effectiveness of search engines is crucial to addressing dynamic robust problems. To further enhance the ability to solve such problems, this article proposes a dynamic robust optimization of particle swarm optimization algorithm based on a hybrid strategy (HS-DRPSO). In the HS-DRPSO algorithm, the two variation strategies of the differential evolution algorithm, i.e., "DE/rand/1" and "DE/best/1," are first combined with the particle swarm algorithm in each search period using a weight dynamic adjustment strategy. The population is then clustered, and the variation strategy of the brainstorming algorithm is used to select the central variation of the clusters to generate new individuals in the population and improve population diversity. By comparing the results with two other dynamic robust optimization algorithms across five dynamic standard test functions, it is demonstrated that the proposed algorithm's overall performance is superior.

RELATED WORK

Dynamic Optimization

A dynamic optimization problem refers to a sequence of optimization problems where the objective function varies with time or environment (Jin et al., 2005). Its mathematical representation is expressed as follows (Cruz et al., 2011):

$$\mathrm{F}(x,t) = \max\ \mathrm{f}(x,t)\ \ s.t.\ x \in X(t) \subseteq S \tag{1}$$

where f represents the objective function, a function of time t and decision variables x; S denotes the search space and $\boldsymbol{X}$(t) represents the set of decision variables at time t.

Dynamic Robust Optimization Performance Evaluation Index

The crux of dynamic robust optimization is to discover adaptable and resilient solutions that can perform well across multiple dynamic environments. To effectively appraise the performance of robust solutions, this paper proposes using two evaluation metrics, namely survival time and average fitness value, as introduced in the literature (Jin et al., 2005). The survival time is defined as follows:

$$f_s\left(x,t,\eta\right)=\max\left\{0\cup\left\{f_h\left(x\right)\geq\eta,\forall i,t\leq h\leq t+l\right\}\right\} \tag{2}$$

where x denotes the feasible solution; f_h(x) denotes the fitness value corresponding to the h-th moment; η denotes the set threshold; l denotes the number of times that the fitness value lasts no less than the threshold since the t moment; and fs denotes the maximum number of time steps that the individual x at the t-th moment can satisfy the threshold in the future.

The expression for the average fitness values is given as follows:

$$f_a\left(x,t,T\right)=\frac{1}{T}\sum\nolimits_{h=t}^{t+T-1}f_h\left(x\right) \tag{3}$$

where T represents the time window value, and f_a represents the average value of the individual x within the time window T.

In order to assess the efficiency of the algorithm presented in this paper, we utilize the performance evaluation metrics for dynamic robust optimization algorithms found in the literature (Fu et al., 2013). These metrics are expressed in equation (4):

$$Z_I=\frac{1}{P}\sum\nolimits_{j=1}^{P}f_j \tag{4}$$

where f_j represents the robustness of the solution obtained by the algorithm in the j-th environment and P represents the total number of environment types.

In addition, to provide a more comprehensive evaluation of the robustness of the proposed algorithm, this paper employs an evaluation index based on the fitness function value proposed in the literature (Yang et al., 2020), as shown in the following equation:

$$G\left(x\right)=g_{robustness}\left(x\right)+g_{funvalue}\left(x\right) \tag{5}$$

where $g_{robustness}(x)$ represents the normalized robust function value of the resulting solution. If the survival time obtained in equation (2) or the average fitness obtained in equation (3) is greater than 0, $g_{robustness}(x) = 1$; otherwise, it equals 0. $g_{funvalue}(x)$ is the result of L_2 parametric normalization of the optimal function value of the resulting solution.

Basic Particle Swarm Optimization Algorithm

Particle Swarm Optimization (PSO) is a global search algorithm based on population, which involves velocity and position updates using the following formulas (Yang et al., 2020):

$$v_{id}\left(e+1\right)=\omega v_{id}\left(e\right)+c_1r_1\left(pbest_{id}\left(e\right)-x_{id}\left(e\right)\right)+c_2r_2\left(gbest_d\left(e\right)-x_{id}\left(e\right)\right) \tag{6}$$

$$x_{id}\left(e+1\right)=x_{id}\left(e\right)+v_{id}\left(e+1\right) \tag{7}$$

where ω is the inertia weight, e represents the current iteration number, $\boldsymbol{v}_{id}(e)$ denotes the speed of particle i in the d-th dimensional component at the e-th iteration, $x_{id}(e)$ denotes the position of particle i in the d-th dimensional component at the e-th iteration, c_1 and c_2 are learning factors, r_1 and r_2 are random numbers in the (0, 1) interval, $pbest_{id}$ the the component of historical optimal position of particle i in the d-th dimension, and $gbest_d$ denotes the component of position of the global extremum *gbest*. in the d-th dimensional.

Brainstorming Variation Strategy

Brainstorm Optimization (BSO) is a novel population-based optimization algorithm introduced by Shi in 2011 (Shi, 2011). The key concept of BSO is to mimic the human brainstorming process by treating each member of the population as a potential solution to the problem at hand. In each iteration, a k-means clustering algorithm is applied to cluster all individuals into several groups. Then, one of these groups is chosen at random according to probability, and the center of the chosen group is updated by adding perturbations based on the following equation:

$$\begin{aligned} y_{id} &= x_{id}+\xi\left(e\right)*N\left(\mu,\sigma^2\right) \\ \xi\left(e\right) &= \log sig\left(\frac{0.5\times E_{\max}-e}{k}\right)*rand() \end{aligned} \tag{8}$$

where x_{id} represents the positional component of the i-th individual to be mutated in d dimensions, y_{id} represents the positional component of the i-th individual to be generated in d dimensions, $N(\mu,\sigma^2)$ represents a Gaussian random number with μ as the mean and σ as the variance, $\xi\left(e\right)$ represents the coefficient of variation at the e-th iteration, E_{max} represents the maximum number of iterations, k is a factor that regulates the slope of the sig function and controls the convergence rate of the algorithm, and *rand*() represents a random number between 0 and 1.

Upon selecting one individual from each of the two classes, the algorithm uses a probability value to determine which individuals to fuse together in order to create the individual to be mutated. The formula for fusing the two selected individuals is as follows:

$$y_{newd}=rand()*x_{i_1d}+\left(1-rand()\right)x_{i_2d} \tag{9}$$

where x_{i_1d} and x_{i_2d} are the individuals selected from the two classes, and y_{newd} are the offspring individuals resulting from the fusion of the two individuals.

DESCRIPTION OF DYNAMIC ROBUST OPTIMIZATION ALGORITHM BASED ON HYBRID STRATEGY OF PARTICLE SWARM ALGORITHM

Hybrid Differential Variation Strategy

The fundamental particle swarm optimization algorithm updates the velocity and position of each particle based on the global optimum and historical optimum, which increases convergence speed but may cause the particles to become stuck in local optima during the search process. To address this issue and balance the algorithm's global search ability and local exploitation ability within the

solution space, we incorporate a weighting factor and combine two differential evolutionary variation strategies, DE/rand/1 and DE/best/1, following Zhang et al. (2017). The former strategy uses random individuals for strong global breadth search ability, whereas the latter is guided by the population's historical optimum for stronger local exploration ability. The two strategies are merged using weighting factors as shown in equation (10):

$$\begin{aligned} x_l^{e+1} &= \beta(x_{l_1}^e + K*(x_{l_2}^e - x_{l_3}^e)) + (1-\beta)*(x_{gbest}^e + K*(x_{l_4}^e - x_{l_5}^e)) \\ K &= K_{\max} - \frac{(K_{\max} - K_{\min})}{e / E_{\max}} \\ \beta &= \exp(-\sqrt{e-1}) \end{aligned} \tag{10}$$

where x_l^{e+1} denotes the mutated new individuals corresponding to the *i*-th individual of population x^e at the *e*-th iteration; *K* is the mutation control parameter, where K_{max} is set to 0.7 and K_{min} is set to 0.5; l_1, l_2, l_3, l_4, l_5 are mutually different three random numbers taken from the range of [1, 2, ..., *N*], and none of them is equal to *i*; β is a monotonically decreasing function with the number of iterations. The first period emphasizes the DE/rand/1 variation strategy for global search, while the later period emphasizes the DE/best/1 strategy for local exploitation, thus enhancing the population's search performance.

Particle Swarm Optimization Algorithm Based on Brainstorming Algorithm

While the PSO algorithm exhibits a strong global search capability, it is prone to the issues of prematurely converging to local optima and losing diversity in the later stages of evolution. To address this problem, we suppose to augment the PSO algorithm with a brainstorming variation strategy that applies BSO variation to individuals within the population. Moreover, the step size of the variation operator is inversely proportional to the number of population iterations, thereby further enhancing the efficiency of the algorithm update process. The specific steps of the proposed brainstorming variation-based particle swarm algorithm (Algorithm 1) are outlined below:

Algorithm 1: Particle Swarm Optimization Algorithm Based on Brainstorming Algorithm

```
Inputs: velocity-position; probability parameter P1;
Output: new population;
1  Input the velocity position information of the particles in the
   population and cluster the population into s classes using the
   k-means algorithm;
2  Rank the individuals in each class using equation (5) and
   select the best value as the center of the class;
3  Randomly select the center of a class and determine with random
   probability whether it is replaced by a new variant of
   individuals generated;
4  If new individuals need to be generated, the new mutant
   individuals are generated as follows: A random number P0
   between [0,1] is generated and compared with the probability
   parameter P1 set in advance in the following manner:
5  If P0< P1
6  a random number Pa between [0,1] is generated and compared with
   the probability parameter P2 set in advance:
7  If Pa< P2
```

Two individuals are randomly selected and fused according to equation (8) to produce new individuals;
else
Randomly select the non-class-centered individual as the new individual;
end
else
A random number P_b between [0,1] is generated, and compared with the probability parameter P_2 set in advance:
If $P_b < P_2$
Two class-centered individuals are randomly selected and fused to produce new individuals according to equation (9);
else
Two non-class-centered individuals are randomly selected and fused to produce new individuals according to equation (9);
end
end
Compare the centers of the selected classes with the new mutant individuals and retain the better ones;
Update the population and perform Step 1 again until the termination condition is satisfied.
In this paper, the individual selection strategy described in equation (5) is employed to dynamically determine the new gbest. Additionally, the parameters c1 and c2 in the particle swarm velocity update equation, as described in equation (6), are adjusted by utilizing techniques from the literature (Xiao, 2017) in the following manner:

$$c_1 = 2.05 + \cos\left(\frac{\pi * e}{E_{\max}}\right) \tag{11}$$

$$c_2 = 2.05 + \cos\left(\frac{\pi * \left(E_{\max}\right)}{E_{\max}}\right) \tag{12}$$

HS-DRPSO Algorithm Implementation Steps

Algorithm 2 presents the pseudo-code for the proposed dynamic robust particle swarm optimization algorithm based on the hybrid strategy in this paper.

Algorithm 2: HS-DRPSO

Inputs: population size np; maximum number of time windows T_{max}; maximum number of iterations E_{max};
Output: Robust solution sequence
Random initialization of populations pop
for t = 1: T_{max} **do**
for e =1: E_{max} **do**

```
for i = 1:np do
Update the individuals xi according to equation (6) and update the population pop;
Differential variation was performed according to equation (10) to obtain the population Apm;
Comparing individuals in pop and Apm according to the individual evaluation strategy (equation (5)), retaining the better individuals and updating them in pop;
end for
Using the k-means algorithm to cluster populations pop into s classes;
Use Algorithm 1 to brainstorm variants and update the population pop;
According to equation (5), G(x) is obtained by combining the evaluation of individuals in pop, and G(x) is arranged in reverse order;
Probability of selection of individual x:
Random integer generation ς ∈ [1, np], and random number r ∈ (0,1)
If p_ς > r then
g_best = g_best−ζ
else
Return to Step 13
end if
end for
end for
Output Robust Solution Sequence
```

ALGORITHM SIMULATION EXPERIMENTS

To demonstrate the efficacy of the proposed algorithm, it was empirically evaluated against the currently available dynamic robust correlation algorithm on five test functions, namely 1) ROOT, a time-domain robust optimization algorithm proposed by (Fu et al., 2015), and 2) DRPSO-DE, a hybrid particle swarm-based dynamic robust optimization algorithm proposed by (Yang et al., 2020).

Test Functions

To validate the efficacy of the proposed algorithm, we conducted tests using the modify Moving Peaks Benchmark (mMPB) function and four test functions (T_1f_1, T_2f_1, T_3f_1, and T_4f_1) generated by the CEC2009 dynamic rotating peaks standard generator. The mMPB test function is derived from the Moving peak Problems (MPB), where the height, width, and position of each wave peak change with the environment. The mMPB test function is expressed using equation (13):

$$
\begin{aligned}
&F\left(x,t\right)=\max_{m=1,2,\ldots,M}\left\{H_t^m-W_t^m*\parallel x-C_t^m\parallel_2\right\}\\
&H_{t+1}^m=H_t^m+height\_severity^m\cdot N\left(0,1\right)\\
&W_{t+1}^m=W_t^m+width\_severity^m\cdot N\left(0,1\right)\\
&C_{t+1}^m=C_t^m+v_{t+1}^m\\
&v_{t+1}^m=\frac{s\cdot\left(\left(1-\lambda\right)\cdot r+\lambda\cdot v_t^m\right)}{\left\|\left(1-\lambda\right)\cdot r+\lambda\cdot v_t^m\right\|}
\end{aligned}
\tag{13}
$$

where M represents the total number of peaks, H_t^m, W_t^m, C_t^m respectively represent the height, width, and peak center point location of the m-th peak at time *t*, v_t^m represents the movement of the *m*-th peak position at time *t*, *N*(0,1) represents a Gaussian distribution of random numbers with a mean value of 0 and variance of 1, and the max function is used to determine the highest peak value of the M peaks as the function value.

The expressions of the test functions for T_1f_1, T_2f_1, T_3f_1, and T_4f_1 are in the same form as shown in equation (14):

$$F\left(x,t\right)=\max_{m=1,2,\ldots,M}\left(\frac{H_t^m}{\left(1+W_t^m\right)*\sqrt{\sum_{d-1}^{n}\frac{\left(x_d-C_{td}^m\right)^2}{D}}}\right) \tag{14}$$

where x_d represents the component of the particle in the *d*-th dimension, while C_{td}^m denotes the component in the *d*-th dimension at the location of the *m*-th peak at the current time. The functions T_1f_1, T_2f_1, T_3f_1, and T_4f_1 are transformed in a manner consistent with the method described in (Yang et al., 2020).

Experimental Parameter Setting

Test Function Parameter Setting

To facilitate comparison, the test function parameters for all algorithms were set to the same values as the standard algorithm parameters. The specifics of the parameter settings for the modified moving peak test function and the dynamic rotation peak test function can be found in Table 1.

Algorithm Parameter Setting

Both the proposed algorithm and the comparison algorithm were run independently for 30 iterations. The time windows T were varied between 2, 4, and 6, and the thresholds η_1 for the mMPB function were set to 40, 45, and 50, while the thresholds η_2 for the dynamic rotation peak function were set to

Table 1. Test function parameter setting

Parameters	mMPB	Dynamic Rotation Peak
Change frequency G	2500	3000
Number of peaks M	5	10
Dimension of decision variables *D*	2	10
Initial height	50	50
Initial width	6	5
Width range	[1,12]	[1,10]
Height range	[30,70]	[10,100]
Height_severity	[1,10]	5
Width_severity	[1,10]	0.5
Search Scope	[0,50]	[-5,5]

10, 15, and 20. The remaining algorithm parameters were set according to the following specifications in Table 2.

Analysis of Experimental Results

To evaluate the performance of the algorithms presented in this paper, they were tested and compared on the mMPB function and the dynamic rotation peak function, with the results displayed in Tables 3 and 4. Wilcoxon rank sum statistics were used at a significance level of 5% to assess the statistical significance of the obtained results. The mean of the 30 outcomes was selected for comparing the algorithm's performance, while the variance was employed for comparing the algorithm's stability, with the variance value given in parentheses. The best results from the experiments are highlighted in bold in the table. The symbols "-," "+" and "≈" indicate whether the HS-DRPSO algorithm proposed in this paper is "inferior," "superior" and "equivalent" to other proposed algorithms. The final statistical score determines the performance of the proposed algorithm, and the more "+" symbols there are, the greater the algorithm's significant performance advantage.

As shown in Table 3, it is evident that HS-DRPSO outperforms Fu and DRPSO-DE for most thresholds of the mMPB function. With regards to survival time, Wilcoxon statistics indicate that HS-DRPSO significantly outperforms the comparison algorithm at thresholds η_1 of 40 and 45. While at a threshold η_1 of 50, HS-DRPSO achieves improved but statistically insignificant results, similar to the comparison algorithm. As for the average fitness value, when the time window T is set as 2 and 6, the results obtained by HS-DRPSO are improved but not significantly different from the comparison algorithm. When the time window T is set as 4, the performance of HS-DRPSO is slightly worse than DRPSO-DE, but the difference is not statistically significant. Therefore, the experimental results demonstrate that HS-DRPSO is capable of achieving a more robust solution for the mMPB function.

From Table 4, it is evident that HS-DRPSO surpasses the comparison algorithm in terms of the performance of the dynamic rotation peak test function. According to Wilcoxon statistics, only in the

Table 2. Algorithm parameter setting

Parameter	Value	Parameter	Value
Population size *np*	50	Probability parameter P_1	0.8
Inertia weight ω	0.729844	Probability parameter P_2	0.2
Max environment number *L*	200	*Sig* lope factor *k*	20
Evaluate the number of time Windows T_{max}	150	Mutation control parameter K_{max}	0.7
Clustering number *s*	4	Mutation control parameter K_{min}	0.5

Table 3. Comparison of algorithm results on mMPB test function

Test Questions	Threshold	Fu	DRPSO-DE	HS-DRPSO
mMPB	η_1=40	2.73E+00(7.38E-02)+	3.15E+00(5.06E-02)+	**3.33E+00**(5.01E-02)
	η_1=45	2.30E+00(5.27E-02)+	2.40E+00(4.62E-02)+	**2.49E+00**(4.29E-02)
	η_1=50	1.72E+00(3.76E-02)≈	1.71E+00(2.78E-02)≈	**1.74E+00**(2.97E-02)
	T=2	5.25E+01(2.52E-01)+	5.41E+01(1.33E+00)≈	**5.46E+01**(1.47E-01)
	T =4	4.20E+01(4.39E-01)+	**4.61E+01**(2.45E-01)-	4.52E+01(2.57E-01)
	T =6	3.29E+01(6.99E-01)+	3.83E+01(2.82E-01)≈	**3.86E+01**(2.79E-01)
Wilcoxon	+/-/≈	5/0/1	2/1/3	

Table 4. Comparison of algorithm results on dynamic rotating peak test functions

Test Questions	Threshold	Fu	DRPSO-DE	HS-DRPSO
T_1f_1	η_2=10	9.20E-02(3.62E-02)+	2.67E-01(4.89E-02)+	**1.41E+00**(4.57E-02)
	η_2=15	4.67E-03(3.50E-03)+	2.04E-01(3.05E-02)+	**1.13E+00**(3.10E-01)
	η_2=20	1.78E-03(2.94E-03)+	1.67E-02(1.11E-02)+	**1.03E+00**(4.30E-02)
	T=2	7.33E+00(6.99E-02)+	9.93E+00(1.27E-01)+	**1.72E+01**(1.84E+00)
	T=4	7.14E+00(7.44E-02)+	8.70E+00(8.85E-02)+	**1.10E+01**(1.21E-01)
	T=6	7.04E+00(6.76E-02)+	8.27E+00(7.41E-02)+	**9.11E+00**(5.34E-02)
T_2f_1	η_2=10	3.26E+00(2.01E-01)+	3.92E+00(3.72E-01)+	**4.28E+00**(3.83E-01)
	η_2=15	6.35E-01(6.69E-02)+	5.46E-01(8.19E-02)+	**1.71E+00**(3.50E-01)
	η_2=20	2.29E-01(3.29E-02)+	1.34E-01(2.14E-02)+	**5.83E-01**(4.09E-01)
	T=2	1.26E+01(7.49E-02)+	1.27E+01(9.88E-02)+	**2.78E+01**(4.61E+00)
	T=4	1.26E+01(8.45E-02)+	1.27E+01(6.86E-02)+	**1.88E+01**(1.84E+00)
	T=6	1.26E+01(8.53E-02)+	1.28E+01(8.80E-02)+	**1.60E+01**(8.03E-02)
T_3f_1	η_2=10	5.48E+00(5.93E-01)+	5.90E+00(8.75E-01)+	**9.52E+00**(1.21E+00)
	η_2=15	3.41E-01(7.58E-02)+	4.98E-01(1.45E-01)+	**6.60E-01**(8.45E-02)
	η_2=20	6.36E-02(1.95E-02)+	1.06E-01(2.20E-02)+	**1.71E+00**(7.63E-02)
	T=2	**1.12E+01**(7.03E-02)-	1.07E+01(7.04E-02)≈	1.08E+01(4.43E-02)
	T=4	1.06E+01(7.67E-02)+	1.08E+01(7.67E-02)+	**1.43E+01**(7.87E-02)
	T=6	1.07E+01(5.45E-02)+	1.09E+01(8.52E-02)+	**1.44E+01**(1.14E-01)
T_4f_1	η_2=10	1.10E+00(3.01E-02)+	1.06E+00(2.27E-02)+	**1.12E+00**(2.33E-02)
	η_2=15	9.94E-01(7.45E-03)+	4.89E-01(3.91E-02)+	**1.00E+00**(6.89E-03)
	η_2=20	9.26E-01(1.72E-02)+	6.58E-02(1.75E-02)+	**9.68E-01**(9.97E-03)
	T=2	1.73E+01(3.75E-01)+	1.10E+01(1.09E-01)+	**1.94E+01**(2.50E-01)
	T=4	8.67E+00(9.34E-02)+	8.91E+00(7.67E-02)+	**1.08E+01**(1.17E-01)
	T=6	8.03E+00(6.14E-02)+	8.25E+00(6.27E-02)+	**9.50E+00**(1.17E-01)
Wilcoxon	+/-/≈	23/1/0	23/0/1	

T_3f_1 function, when the time window T is 2, the maximum fitness value obtained by the algorithm is slightly worse than Fu, but the difference is not significant and is similar to the results of DRPSO-DE. However, for the other three test functions, HS-DRPSO shows a significant advantage in both maximizing the survival time and the average fitness value. The key factor behind this advantage is the variational strategy of HS-DRPSO, which enhances the search performance of the algorithm and makes it more adaptable to the intricate dynamic rotation peak testing environment.

To visually represent the dynamic performance of the proposed HS-DRPSO algorithm, line graphs are plotted to show the robust solutions found by the algorithm on 150 moments, as depicted in Figures 1 to 5. Upon observation of Figures 2 to 5, it becomes evident that the algorithm outperforms the comparison algorithms significantly, especially on T_1f_1, T_2f_1, T_3f_1 *and* T_4f_1. Only on T_3f_1, when the time window T is 2, the algorithm's performance is marginally worse than that of Fu. Furthermore, as the survival time threshold η_1 and η_2 of each test function, and the time window threshold T increases, the robustness performance of the solution decreases, owing to the higher quality requirements of the robust solution.

As shown in Tables 4, one can see that the HS-DRPSO has better performance at the most thresholds of the mMPB function and is significantly better than the compared algorithms of the

Figure 1. Performance of robust solution under mMPB function

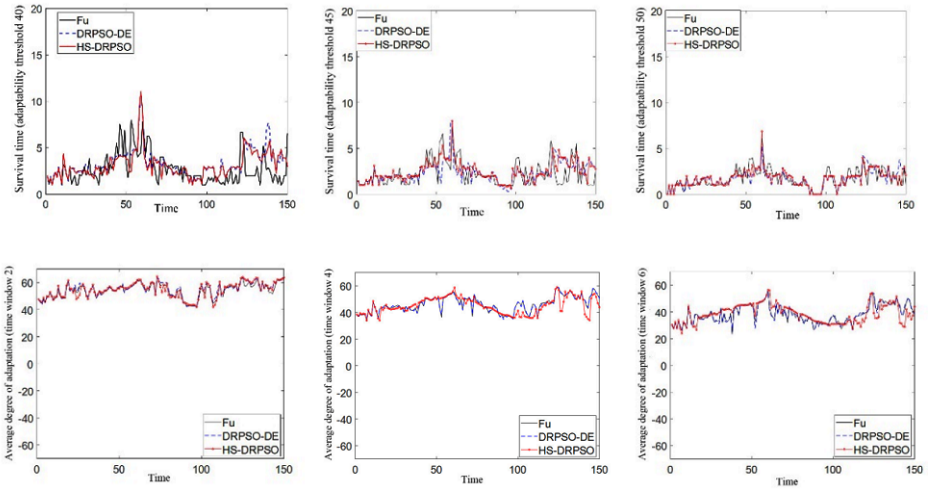
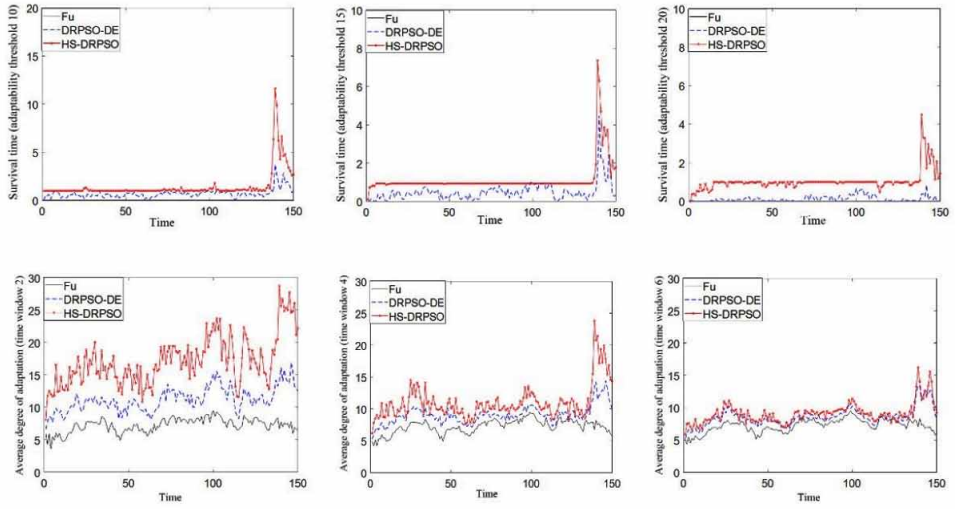


Figure 2. Performance of robust solution under T_1f_1 function



survival time. In addition, for the dynamic rotation peak test function, the HS-DRPSO has obvious advantages in both the survival time and the average fitness value. Moreover, Figures 1-5 further show the advantages of the proposed HS-DRPSO. Thus, the overall performance of HS-DRPSO is substantially better than the algorithms proposed in the literature (Fu et al., 2015) and literature (Yang et al., 2020), providing evidence that HS-DRPSO is an effective algorithm for solving dynamic robust problems.

CONCLUSION

To enhance the search performance of dynamic robust optimization algorithms, we present the HS-DRPSO algorithm that utilizes a hybrid approach, incorporating differential evolution and

Figure 3. Performance of robust solution under T_2f_1 function

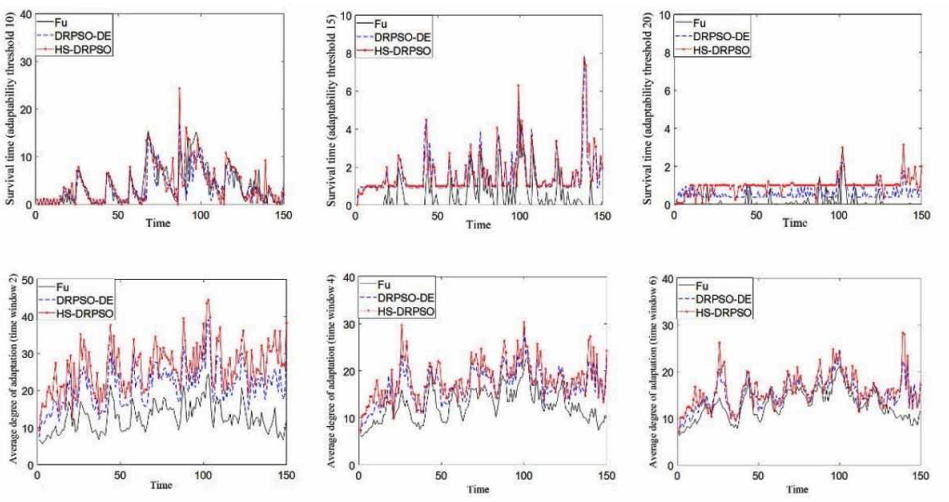
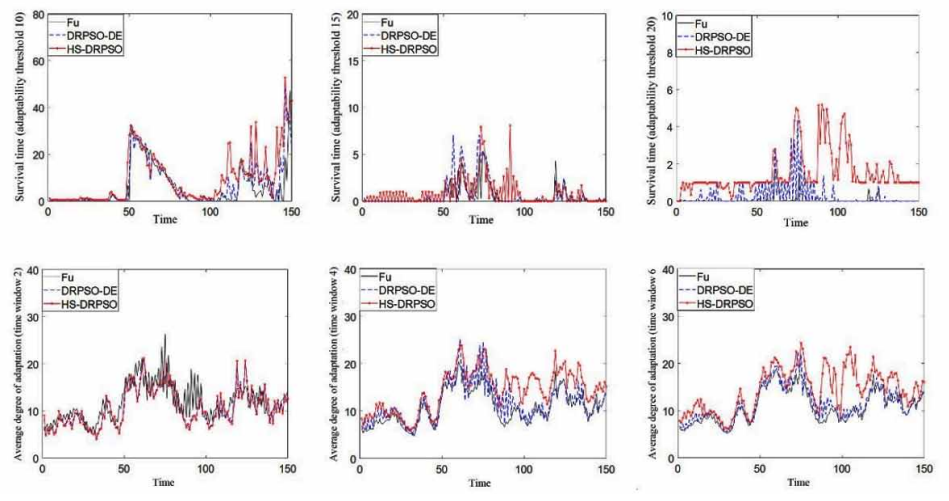


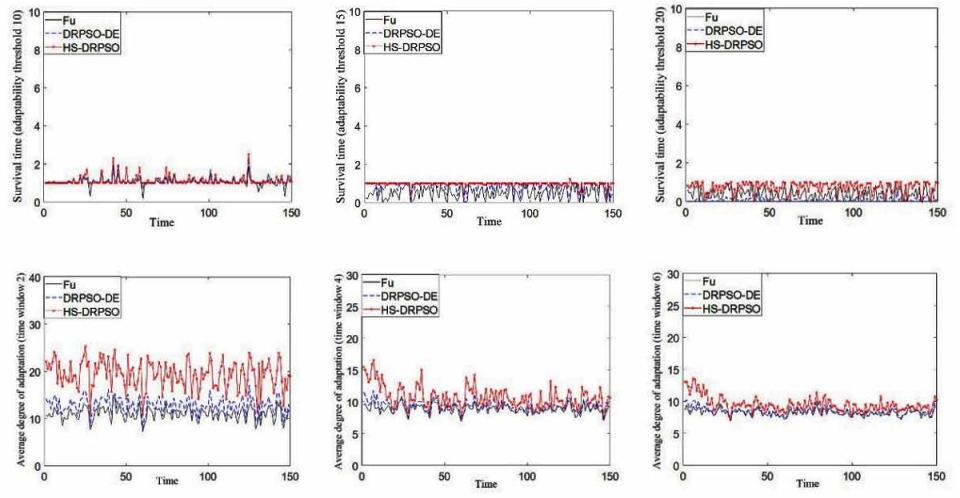
Figure 4. Performance of robust solutions under T_3f_1 function



brainstorming variation strategies. This algorithm dynamically combines the two variation strategies of the differential evolution algorithm with the particle swarm algorithm to improve global search ability in the early stages and local search ability in the later stages, while the variation strategy of the brainstorming algorithm is employed to mutate individuals in the population during iteration to further improve the algorithm's diversity. Experimental comparisons with two other dynamic robust solution algorithms on five dynamic standard test functions demonstrate the effectiveness of the proposed algorithm, which outperforms the comparison algorithms overall.

Since the accuracy of model prediction has a significant impact on the performance of dynamic robust optimization algorithms, future work can focus on improving the prediction

Figure 5. Performance of robust solutions under T_4f_1 function



capabilities by using more advanced machine learning methods and on applying the proposed method to real-word applications.

ACKNOWLEDGMENT

This research was funded by Science and Technology Project of Guangxi Power Grid Corporation, grant number GXKJXM20190606.

COMPETING INTERESTS

The authors of this publication declare there are no competing interests.

REFERENCES

Chen, M., Guo, Y., Jin, Y., Yang, S., Gong, D., & Yu, Z. (2022). An environment-driven hybrid evolutionary algorithm for dynamic multi-objective optimization problems. *Complex & Intelligent Systems*, *9*(1), 659–675. doi:10.1007/s40747-022-00824-4

Cruz, C., Gonzalez, J. R., & Pelta, D. A. (2011). Optimization in dynamic environments: A survey on problems, methods and measures. *Soft Computing*, *15*(7), 1427–1448. doi:10.1007/s00500-010-0681-0

Falahiazar, A., Sharifi, A., & Seydi, V. (2022). An efficient spread-based evolutionary algorithm for solving dynamic multi-objective optimization problems. *Journal of Combinatorial Optimization*, *44*(1), 794–849. doi:10.1007/s10878-022-00860-3

Fu, H., Sendhoff, B., Tang, K., & Yao, X. (2015). Finding robust solutions to dynamic optimization problems. *Proceedings of the Applications of Evolutionary Computation*, *24*(4), 616–625. doi:10.1109/TEVC.2014.2377125

Huang, Y., Ding, Y., Hao, K., & Jin, Y. (2017). A multi-objective approach to robust optimization over time considering switching cost. *Information Sciences, 394*(1), 183-197. Doi..2017.02.029org/10.1016/j.ins

Huang, Y., Jin, Y., & Hao, K. (2020) Decision-making and multi-objectivization for cost sensitive robust optimization over time. *Knowledge-Based Systems, 199*(1), 1-22. doi..105857org/10.1016/j.knosys.2020

Jin, Y., & Branke, H. (2005). Evolutionary optimization in uncertain environments - A survey. *IEEE Transactions on Evolutionary Computation*, *9*(3), 303–317. doi:10.1109/TEVC.2005.846356

Jin, Y., Tang, K., Yu, X., Sendhoff, B., & Yao, X. (2012). A framework for finding robust optimal solutions over time. *Memetic Computing*, *5*(1), 3–18. doi:10.1007/s12293-012-0090-2

Parrott, D., & Li, X. D. (2006). Locating and tracking multiple dynamic optima by a particle swarm model using speciation. *IEEE Transactions on Evolutionary Computation*, *10*(4), 440–458. doi:10.1109/TEVC.2005.859468

Shi, Y. (2011). Brain storm optimization algorithm. *Proceedings of the International Conference on Swarm Intelligence (ICSI),* 303-309.

Xiao, Y. (2017). *Application of hybrid particle swarm optimization in array antenna synthesis*. Hangzhou Dianzi University.

Yang, X., Liao, Q., & Fan, Q. (2020). Dynamic robust optimization based on hybrid particle swarm optimization. *Zidonghua Yu Yibiao*, *41*, 30–35.

Yazdani, D., Nguyen, T. T., & Branke, J. (2017). A new multi-swarm particle swarm optimization for robust optimization over time. *Applications of Evolutionary Computation*, 99-109. doi:10.1007 /978-3-319-55792-2_7

Yazdani, D., Nguyen, T. T., & Branke, J. (2019). Robust optimization over time by learning problem space characteristics. *IEEE Transactions on Evolutionary Computation*, *23*(1), 143–155. doi:10.1109/TEVC.2018.2843566

Yu, X., Jin, Y., & Tang, K. (2010). Robust optimization over time - A new perspective on dynamic optimization problems. *Proceedings of the 2010 IEEE Congress on Evolutionary Computation (CEC),* 1-6. doi:10.1109/CEC.2010.5586024

Zhang, J., Song, L., & Zhang, Y. (2017). Weighted variation strategy dynamic differential evolution algorithm. *Computer Engineering and Applications*, *53*, 156–162.