Hindawi Publishing Corporation
Abstract and Applied Analysis
Volume 2014, Article ID 614569, 10 pages
http://dx.doi.org/10.1155/2014/614569

Research Article

An Improved Differential Evolution Algorithm for Maritime Collision Avoidance Route Planning

Yu-xin Zhao,[1] Wang Li,[1] Shaojun Feng,[2] Washington Y. Ochieng,[2] and Wolfgang Schuster[2]

[1] *College of Automation, Harbin Engineering University, Harbin 150001, China*
[2] *Center for Transport Studies, Imperial College London, London SW7 2AZ, UK*

Correspondence should be addressed to Yu-xin Zhao; zhaoyuxin@hrbeu.edu.cn

Received 6 August 2014; Revised 8 October 2014; Accepted 8 October 2014; Published 12 November 2014

Academic Editor: Shen Yin

High accuracy navigation and surveillance systems are pivotal to ensure efficient ship route planning and marine safety. Based on existing ship navigation and maritime collision prevention rules, an improved approach for collision avoidance route planning using a differential evolution algorithm was developed. Simulation results show that the algorithm is capable of significantly enhancing the optimized route over current methods. It has the potential to be used as a tool to generate optimal vessel routing in the presence of conflicts.

1. Introduction

Collision avoidance technologies are increasingly becoming relevant in the field of marine transport to ensure safety [1]. These technologies are underpinned by three key components: navigation and surveillance technologies and optimization logic. Technologies such as differential global positioning systems (DGPS), automatic radar plotting aid (ARPA), automatic identification system (AIS), and electronic chart display and information system (ECDIS) have significantly improved positioning and surveillance performance. Furthermore, optimization and decision-making theories have rapidly developed in recent years [2, 3], enabling the computation of better vessel trajectories in multivessel environments. These include artificial intelligence algorithms such as genetic algorithms (GA) (see [4, 5]), ant colony algorithms (ACA) (see [6, 7]), fuzzy decision methods (see [8–10]), and various other approaches [11–15]. Smierzchalski proposed an evolutionary algorithm to model a ship's trajectory in collision situations (see [16, 17]). However, the low computational efficiency puts into question the practical application of such approach. Inspired by the maze route algorithm [18], a practical model was proposed by Chang et al. [19] to transform the decision-making problem in collision avoidance into a route distance optimization problem. Szlapczynski [20] improved this model by accounting for ship maneuvering characteristics, for the COLREGS (The International Regulations for Preventing Collisions at Sea) and by introducing other operational constraints (e.g., speed reduction capability). In [21–23] a concept of a set of safe (though usually not optimal) trajectories is proposed, based on collision avoidance path-searches for scenarios involving multiple ships and stationary constraints. These soft computing methods can be used not only to find an optimal route from port to port, but also to avoid collision [24, 25].

However, the quality of the solution and the convergence time of current optimization methods are typically not sufficient for practical applications. Moreover, significant other challenges remain, including accounting for regulations of collision avoidance, as well as navigation and maneuverability factors. Additionally, the computational efficiency and the ability to adapt to the dynamic target saturation state in restricted waters need further enhancement. In order to address these issues, this paper introduces a modified differential evolution algorithm.

This paper addresses some of the above limitations by developing a modified neighborhood differential evolution (MNDE) algorithm, with global and local neighborhoods. Differential evolution (DE) algorithms were first introduced by Storn and Price [26], resulting in significant variants

with improved performance [27]. These algorithms were designed to address multiobjective, constrained, large-scale, optimization problems. Analyses show that their performance in terms of accuracy, robustness, and convergence time is significantly better than GA, ACA, and particle swarm optimization (PSO) algorithms [28, 29].

Section 2 provides a brief outline of the basic DE family of algorithms. Section 3 introduces the proposed variant of the differential evolution with global and local neighborhoods (DEGL) algorithm, and Section 4 characterizes the performance of the developed algorithm. The algorithm is applied in Section 5 to ship collision avoidance route planning and the results are analyzed and discussed.

2. State-of-the-Art Differential Evolution Algorithms

DE is an evolutionary algorithm based on modelling swarm behavior, which uses optimization techniques based on cooperation and competition between individuals within populations. DE is based on a real-code greedy genetic algorithm. Like other evolutionary algorithms, DE is based on the following: from a randomly generated initial population and according to certain rules of operation (such as mutation, crossover, and selection), the best individuals are retained following an iterative process which accounts for each individual's fitness and eliminates the inferior individuals, thus guiding the search process in the search of the optimal solution.

Mathematically, DE is a simple parallel direct search method that attempts to find the global optimal solution in a D-dimensional real parameter space $\mathfrak{R}^D$. Here we denote by f the decision rule defined in the fitness function and by Ω a nonempty finite set in the search space, resulting in the following optimization problem:

$$f\left(\overrightarrow{X^*}\right) < f\left(\vec{X}\right), \quad \forall \vec{X} \in \Omega, \quad f : \Omega \subseteq \mathfrak{R}^D \longrightarrow \mathfrak{R}. \quad (1)$$

The goal of DE is to find a parameter vector $\overrightarrow{X^*}$, such that the fitness function (or objective function) is minimized. Assuming that each element of the population in DE is denoted as $P = 0, 1, \ldots, P_{\max}$, the ith vector of the population at the current generation is represented as

$$\overrightarrow{X_{i,P}} = \left[x_{1,i,P}, x_{2,i,P}, x_{3,i,P}, \ldots, x_{D,i,P}\right]. \quad i = 0, 1, \ldots, N-1, \quad (2)$$

where N is the total population, which is invariant during the minimization process. The mutation is carried out by the donor vector based on the difference between two randomly chosen solution vectors; in this sense, its mutation is like an exploration move in pattern search. The crossover operation of DE can be performed in either binomial or exponential manner. DE is based on a global population search strategy, using a simple differential mutation operation and one-on-one competitive strategies for survival. Moreover, DE has specific memory capabilities enabling dynamic tracking search to adjust the search strategy without requiring the detailed knowledge of the characteristics of the problem. DE shows strong global convergence ability and robustness.

The performance of DE depends on the correct balance between global exploration and local development capacity, with significant reliance on the control parameters, including population size, scaling factor, and crossover rate. In recent years, significant variants of DE, such as the DE family of Storn and Price [26, 27] which consists of 10 classical mutation strategies, were developed. However, improving the mutation strategy cannot entirely solve DE weaknesses such as premature convergence or likelihood of local convergence. For example, *DE/target-to-best/1* is one of the most effective mutation strategies proposed by Storn and Price:

$$\overrightarrow{K_{i,P}} = \overrightarrow{X_{i,P}} + S \cdot \left(\overrightarrow{X_{\text{best},P}} - \overrightarrow{X_{i,P}}\right) + S \cdot \left(\overrightarrow{X_{r_1^i,P}} - \overrightarrow{X_{r_2^i,P}}\right), \quad (3)$$

where $\overrightarrow{X_{\text{best},P}}$ is the best individual vector with the best fitness (i.e., lowest objective function value for a given minimization problem) in the population at generation P. The scaling factor S is a positive control parameter for scaling the difference vectors. In the mutation strategy, the difference between any two of these three random selected vectors is scaled by the factor S to generate the corresponding donor vector. It is constant during the global optimum searching process. For each iteration, all of the vectors are close to the optimal location of the fitness surface, resulting in the DE quickly converging to the optimal solution. However, the "development" ability may result in the optimal solution in some local search space, causing the DE to lose its global search capability after a few iterations. Furthermore, the DE algorithm adopts the "greedy" selection rule (choosing always the better vector between the target vector and the trial vector), with a fixed scaling factor S (typically selected in the range $[0.4, 1]$). Therefore, if the vector $\overrightarrow{X_{r_1^i,P}} - \overrightarrow{X_{r_2^i,P}}$ for generating the disturbance is small (when the individual vectors are close to each other, usually the population converges to a small range), the individual vectors would not explore a better search space, and it is difficult to avoid becoming trapped in a suboptimal solution [30]. In order to address this issue, inspired by PSO, Das et al. [31] proposed a novel DE algorithm, DEGL (differential evolution using a neighborhood-based mutation operator with global and local neighborhoods).

In Das et al.'s work, a strategy similar to (3) is adopted to design two types of neighborhood-based models for local mutation and global mutation separately, using the globally optimal vector $\overrightarrow{X_{\text{best},P}}$ at the current generation P to mutate a given population member. The neighborhood concept comes from the PSO algorithm. Assuming a given differential evolution population $\overrightarrow{X_P} = \{\overrightarrow{X_{1,P}}, \ldots, \overrightarrow{X_{i,P}}, \ldots, \overrightarrow{X_{N,P}}\}$ with $\overrightarrow{X_{i,P}} = [x_{1,i,P}, x_{2,i,P}, \ldots, x_{D,i,P}]$, a neighborhood with radius k (k is a nonzero integer within the interval $[0, (N-1)/2]$, where $2k + 1 \leq N$) is defined for each vector $\overrightarrow{X_{i,G}}$. This results in a neighborhood population $\overrightarrow{X_K} = \{\overrightarrow{X_{i-k,P}}, \ldots, \overrightarrow{X_{i,P}} \ldots, \overrightarrow{X_{i+k,P}}\}$. Here, the ring topology structure is adopted, so that the two contiguous neighborhoods of $\overrightarrow{X_{1,G}}$ are $\overrightarrow{X_{N,P}}$ and $\overrightarrow{X_{2,P}}$. Based

on the above assumptions, Das et al. give the local and global neighborhood-based mutation models as follows.

The local mutation model is described by

$$\overrightarrow{L_{i,P}} = \overrightarrow{X_{i,P}} + \alpha \cdot \left(\overrightarrow{X_{n\_\text{best}_i,P}} - \overrightarrow{X_{i,P}}\right) + \beta \cdot \left(\overrightarrow{X_{p,P}} - \overrightarrow{X_{q,P}}\right), \tag{4}$$

where $\overrightarrow{X_{n\_\text{best}_i,P}}$ represents the optimal vector in the neighborhood population of $\overrightarrow{X_{i,P}}$ and $p, q \in [i-k, i+k]$ $(p \neq q \neq i)$.

Similarly, the global mutation model is given by

$$\overrightarrow{G_{i,P}} = \overrightarrow{X_{i,P}} + \alpha \cdot \left(\overrightarrow{X_{g\_\text{best},P}} - \overrightarrow{X_{i,P}}\right) + \beta \cdot \left(\overrightarrow{X_{r_1,P}} - \overrightarrow{X_{r_2,P}}\right). \tag{5}$$

The final donor vector of DEGL is a combination of the local and global donor vectors using a scalar weight $\lambda \in (0, 1)$:

$$\overrightarrow{V_{i,P}} = \lambda \cdot \overrightarrow{G_{i,P}} + (1-\lambda) \cdot \overrightarrow{L_{i,P}}. \tag{6}$$

When $\lambda = 1$ and $\alpha = \beta = S$, (5) will be transformed into the mutation strategy *DE/target-to-best/1* (see (3)). DEGL is effectively a generalized model providing a better balance between local and global strategies.

3. An Improved DE Algorithm Based on DEGL

A key limitation of the DEGL model developed by Das et al. is that the global optimum searching capability and the convergence speed are very sensitive to the choice of these scaling factors. The choice of scale factor is thus crucial in optimizing the performance of DEGL. If the scale factors are small, the disturbance added to the population is negligible, resulting in the algorithm premature convergence. Consequently, it is difficult for the algorithm to converge to the minimum. If on the other hand the vectors are getting close to the location of the global optimal, a smaller scaling factor would help to reduce the search time, which contributes significantly to accelerating the convergence speed. On the other hand, a large scaling factor could be conducive to enhance the diversity of individuals in the process of mutation; as a consequence, the premature convergence is likely to be avoided, but at the expense of the convergence rate.

In order to address the limitation of using fixed scaling factors and thereby improve the performance of DEGL, this paper develops a new variant of DE, referred to as the MNDE (modified neighborhood-based mutation operator DE). Two key novelties are introduced: firstly, we adopt variable scaling jitter factors instead of the fixed scaling factor to enhance the convergence performance; secondly, the global optimal individual vector is used to replace the first item in the global mutation model, significantly accelerating the convergence speed.

MNDE combines the local mutation model and the new global mutation model and redefines the global mutation model as

$$\overrightarrow{G_{i,p}} = \overrightarrow{X_{g\_\text{best},p}} + \alpha_g \cdot \left(\overrightarrow{X_{g\_\text{best},p}} - \overrightarrow{X_{i,p}}\right) + \beta_g \cdot \left(\overrightarrow{X_{r_1,p}} - \overrightarrow{X_{r_2,p}}\right). \tag{7}$$

Here, α_g and β_g are the variable scaling jitter factors based on the fixed scaling factor S, which is used to scale the difference vectors to generate the corresponding donor vector in the mutation strategy. Replacing the fixed scaling factor with variable scaling jitter factors can effectively maintain the diversity of the population in the global neighborhood model; moreover, the convergence speed is significantly improved. More specifically, when the variable scaling factor is relatively large, the individuals tend to sample diverse zones of the search space during the global optimum search, which prevents the population from getting trapped in local minima. Meanwhile, smaller scaling factors in the second step of the global optimum search help adjust the movements of the trial solutions of the first step and allow to better explore the interior space in which the suspected global optimum lies, hence allowing the population to converge to the global optimum solution.

The scaling jitter factors randomly jitter in a small neighborhood of the fixed scale factor S. In other words, a small disturbance is introduced to modify the scale factor to increase the population diversity and thus avoid the potentially premature convergence observed with state-of-the-art DEGL. The jitter factors are computed as

$$\alpha_g = \beta_g = \gamma * \text{rand}(N, D) + S. \tag{8}$$

α_g and β_g are given by the sum of the scale factor S and the $N * D$ dimensional random matrix $\gamma * \text{rand}(N, D)$, where N and D are the dimensions of the individual vectors and D generally corresponds to the dimension of the optimization problem. We introduce control parameter γ to adjust the jitter range of the scaling factor in order to maintain both exploitation and exploration performance. As mentioned previously, if the scale factor is too small, the difference vector is very small (i.e., the individual vectors are very close to each other, typical during population convergence to a small domain), and the individual vectors would not explore a better search space, making it difficult to avoid becoming trapped in a suboptimal solution. Under these circumstances we choose a larger γ to yield a bigger scaling factor to preserve the diversity of the vectors belonging to the same neighborhood. On the other hand, if the scaling factor is too large, the number of iterations needed to search for the global optimum is large, delaying solution convergence. We therefore reduce the control parameter γ to generate a suitable scaling factor so as to improve convergence performance.

The second novelty is that, unlike in the DEGL algorithm, we adopt the global optimal individual vector at the current generation $\overrightarrow{X_{g\_\text{best},P}}$ as the first item in the global mutation model. The global mutation model is used to guide the

Table 1: Benchmark functions and parameter setup.

Function	Expression	Search range	Optimum solution x^*	Optimum value $f(x^*)$
Sphere	$f_1(x) = \sum_{i=1}^{D} x_i^2$	$[-5.12, 5.12]^D$	$[0, 0 \cdots 0]$	0
Axis parallel hyperellipsoid	$f_2(x) = \sum_{i=1}^{D} i x_i^2$	$[-5.12, 5.12]^D$	$[0, 0 \cdots 0]$	0
Sum of different powers	$f_3(x) = \sum_{i=1}^{D} \lvert x_i \rvert^{(i+1)}$	$[-1, 1]^D$	$[0, 0 \cdots 0]$	0
Rosenbrock	$f_4(x) = \sum_{i=1}^{D-1} \left[100\left(x_i^2 - x_{i+1}\right)^2 + \left(x_i - 1\right)^2\right]$	$[-2, 2]^2$	$[1, 1]$	0
Schwefel's problem	$f_5(x) = \sum_{i=1}^{D} \lvert x_i \rvert + \prod_{i=1}^{D} \lvert x_i \rvert$	$[-10, 10]^D$	$[0, 0 \cdots 0]$	0
Step function	$f_6(x) = \sum_{i=1}^{D} \left(\lfloor x_i + 0.5 \rfloor\right)^2$	$[-100, 100]^D$	$[0.5, 0.5 \cdots 0.5]$	0
De Jong's function 4 (no noise)	$f_7(x) = \sum_{i=1}^{D} i x_i^4$	$[-1.28, 1.28]^D$	$[0, 0 \cdots 0]$	0
Rastrigin	$f_8(x) = \sum_{i=1}^{D} \left[x_i^2 - 10\cos(2\pi x_i) + 10\right]$	$[-6, 6]^D$	$[0, 0 \cdots 0]$	0

mutation strategy to mutate a population member, which improves the convergence ability especially in the global multimodal optimization. Choosing the best vector in the whole population ensures that the new trial vector with minor fitness value may be generated in a small search space near the global optimum solution, thereby enhancing converging ability. Furthermore, if the vectors are getting close to the location of the global optimal solution, the convergence rate is accelerated by choosing the global optimum individual vector instead of the current target individual vector.

4. Performance Testing and Analysis

To evaluate the performance of the MNDE algorithm, various simulations were carried out.

We selected 8 classical benchmark functions (shown in Table 1) and compared them using three DE algorithms including DEGL, *DE/best/1* (simplified as DE1), and *DE/target-to-best/1* (simplified as DE2). The hardware platform used was a ThinkPad laptop with configuration Core2 Duo CPU, 2.10 GHz, 4 GB Memory, and 64-bit Windows 7 OS; the software was MATLAB 2009a. For each simulation, all algorithms were run 40 times independently, and the optimal solution, worst solution, mean, standard deviation, and average processing time (s) for the given convergence threshold were computed.

In these simulations, we choose the population size of all DE variants $N = 200$, a scale factor $S = 0.85$, and a crossover rate $R = 0.9$. For DEGL, we set the fixed scale factors $\alpha = \beta = S$, and for MNDE, the jitter scale factors $\alpha_g = \beta_g = \gamma * \text{rand}(N, D) + S$, where $\gamma = 0.0001$, the radius of neighborhood $k = 3$, and a combination scalar weight λ in (8) is used to balance the global mutation strategy and local mutation strategy in the search of the optimal solution. In the preliminary stage of the optimization, a smaller λ could help to enhance the exploitation ability, while a larger λ improves the exploration ability in the later stage. In this paper, a linear increment schedule to change λ was adopted to achieve global optimization. Here, all four contestant algorithms and the same stopping criterion (i.e., the same maximum evolution of the number of iterations) were used. For each test function, the individual vector has to evolve within the limits of the search range, which is shown in Table 1. The dimension of each vector is $D = 30$, and up to 3000 iterations are carried out. When the vector evolves to the optimum solution x^*, the optimum value of the test function $f(x^*)$ is obtained.

The optimal solution, the worst solution, the mean, and the standard deviation of the best-of-run values of each of the four contestant algorithms are presented in Table 2. The average number of iterations required to achieve the same prescribed threshold fitness 10^{-10} is shown in Table 3, the symbol of "—" indicates that the algorithm failed to reach the prescribed threshold within 3000 iterations. Table 2 shows the improved performance of the MNDE algorithm over other algorithms. It can be clearly seen that the solution found by MNDE algorithm is the most precise among these comparative algorithms, and this is due to the use of the scaling jitter factor in MNDE algorithm. The scaling jitter factor can be adjusted itself according to the current solution during the mutation process; therefore the diversity of the population is enhanced, so the individuals tend to sample diverse zones of the search space during the global optimal searching process, which prevents the population from getting trapped in local minima. In most cases, MNDE algorithm also runs faster and converges more quickly, as shown in Table 3, and this is because of choosing the best vector in the global mutation operation strategy. Adopting the best vector in the whole population can make sure that the new trial vector with minor fitness value may be generated in a small search space near the global optimum solution; therefore the algorithm can avoid blind searching; thus the converging ability is greatly improved.

Table 2: The optimal solution, the worst solution, the mean, and the standard deviation tested on various benchmark functions.

Function	Algorithm	Optimal value/worst value	Mean/standard deviation
f_1	DE1	$8.26E-016/5.93E-014$	$4.75E-015/6.82E-030$
	DE2	$2.26E-018/7.75E-016$	$5.08E-018/3.88E-042$
	DEGL	$1.34E-050/5.04E-049$	$2.61E-049/3.49E-095$
	MNDE	**7.57E − 132/6.11E − 130**	**3.43E − 086/3.79E − 170**
f_2	DE1	$8.65E-014/4.25E-013$	$1.69E-014/2.28E-023$
	DE2	$8.28E-017/1.06E-016$	$9.41E-017/1.62E-034$
	DEGL	$3.08E-050/3.55E-049$	$1.93E-049/2.93E-086$
	MNDE	**3.19E − 130/5.89E − 129**	**3.11E − 129/3.94E − 189**
f_3	DE1	$6.01E-037/1.87E-036$	$1.24E-036/9.61E-060$
	DE2	$5.33E-042/2.64E-040$	$8.16E-042/1.63E-076$
	DEGL	$1.99E-068/1.50E-065$	$7.52E-066/1.06E-120$
	MNDE	**2.03E − 077/4.34E − 074**	**2.17E − 074/3.17E − 128**
f_4	DE1	$5.66E-002/2.36E-000$	$9.74E-001/6.39E-002$
	DE2	$3.90E-003/3.99E-001$	$1.99E-001/2.82E-003$
	DEGL	$6.40E-016/6.89E-014$	$3.48E-014/4.83E-028$
	MNDE	**9.49E − 029/2.54E − 028**	**1.74E − 028/2.54E − 051**
f_5	DE1	$4.14E-008/6.49E-007$	$5.19E-008/1.66E-015$
	DE2	$5.75E-009/1.52E-008$	$1.05E-008/6.62E-017$
	DEGL	$2.35E-025/2.01E-024$	$1.12E-024/1.26E-045$
	MNDE	**3.92E − 038/2.19E − 035**	**1.10E − 035/1.54E − 062**
f_6	DE1	$2.31E-015/8.61E-014$	$8.87E-14/2.66E-24$
	DE2	$1.65E-015/1.87E-014$	$1.76E-015/1.57E-029$
	DEGL	$0.00E-000/0.00E-000$	$0.00E-000/0.00E-000$
	MNDE	**0.00E − 000/0.00E − 000**	**0.00E − 000/0.00E − 000**
f_7	DE1	$1.43E-029/8.68E-027$	$5.05E-029/5.13E-059$
	DE2	$1.20E-033/2.11E-032$	$1.12E-032/1.42E-53$
	DEGL	$1.37E-085/5.97E-084$	$3.05E-084/4.13E-161$
	MNDE	**9.54E − 109/5.12E − 096**	**2.56E − 096/3.62E − 198**
f_8	DE1	$2.68E-002/3.58E-000$	$2.76E-001/6.61E-001$
	DE2	$5.38E-002/1.03E-001$	$7.70E-002/3.29E-002$
	DEGL	$1.79E-011/7.07E-010$	$3.53E-012/4.99E-022$
	MNDE	**6.19E − 030/1.28E − 028**	**1.24E − 028/6.02E − 056**

To further illustrate the performance of the MNDE, two benchmark functions were used to test and evaluate algorithm performance under challenging conditions.

In the following section, a comparison between MNDE and other DE variants is carried out.

The Rosenbrock function is a classic two-dimensional test function with a single peak, which has a unique global minimum of 0 at (1, 1) as shown in Figure 1. The convergence characteristics of the four algorithms DE1, DE2, DEGL, and MNDE are shown in Figures 2 and 3. The MNDE strategy exhibits the shortest convergence time for a given threshold fitness and the fewest number of iterations to reach the global minimum value 0. As we use the global optimal individual vector $\overrightarrow{X_{g\_\text{best},P}}$ as the first item in the global mutation model in (7), the MNDE strategy has significantly better performance than DEGL in terms of convergence accuracy and the speed of evolution.

The Rastrigin function is a typical multipeak test function used to evaluate a global optimization problem:

$$DA + \sum_{i=1}^{D} \left(x_i^2 - A\cos\left(2\pi x_i\right)\right), \quad -W \le x_i \le W. \tag{9}$$

The Rastrigin function has a global optimum point at 0 and a large number of local minima, as shown in Figure 4. For example, when $A = 1$, $D = 6$, and $W = 5.12$, there are 1771561 local minima.

Figure 5 shows that the MNDE has significant advantages in the search of a global optimum solution over other algorithms. In the test results, the MNDE strategy converges to the optimum point after a relative small number of iterations (compared to the DE1 and DEGL strategies, an almost 50% reduction in the number of iterations). The DE2 strategy was unable to find any solution in the predefined number of iterations. It should be noted that, as we introduce the scaling jitter factors α_g and β_g, MNDE can effectively maintain the

Table 3: The average number of iterations required to reach 10^{-10}.

Function	Algorithm	Iterations
f_1	DE1	2076
	DE2	1925
	DEGL	496
	MNDE	**438**
f_2	DE1	2139
	DE2	1965
	DEGL	497
	MNDE	**488**
f_3	DE1	815
	DE2	719
	DEGL	266
	MNDE	**252**
f_4	DE1	—
	DE2	—
	DEGL	1525
	MNDE	**1049**
f_5	DE1	—
	DE2	—
	DEGL	812
	MNDE	**737**
f_6	DE1	2254
	DE2	2196
	DEGL	575
	MNDE	**522**
f_7	DE1	1163
	DE2	1096
	DEGL	304
	MNDE	**291**
f_8	DE1	—
	DE2	—
	DEGL	1605
	MNDE	**1061**

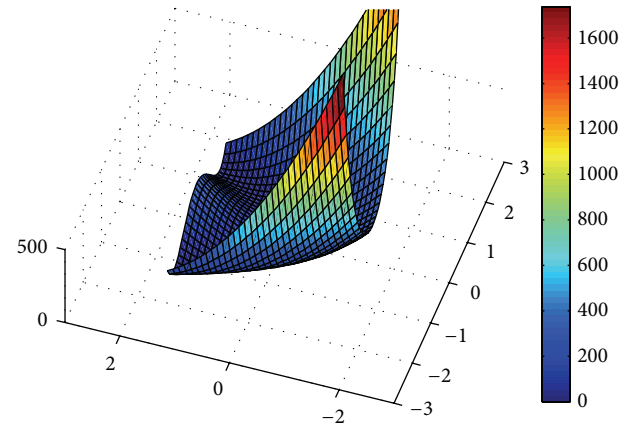

Figure 1: The solution space of the Rosenbrock function.

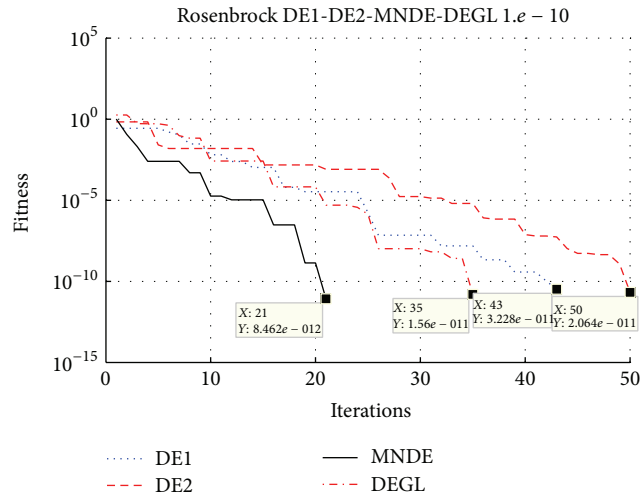

Figure 2: The evolution results for convergence to the prescribed threshold fitness (10^{-10}).

diversity of the population in the global neighborhood model and improve the global search over DEGL.

5. Application to Ship Collision Avoidance Route Planning

In recent studies, researchers have investigated ways of improving the collision avoidance route planning performance of evolutionary algorithms, for example, Lee [18] and Zaman et al. [10]. Here we test the dynamic ship collision avoidance route planning methodology of the improved differential evolution algorithm developed in this paper.

In this paper, we test the performance using a method similar to that used in Lee [18] and Zaman et al. [10]. The new method in this paper compares GA, DE, and MNDE, using the same platform to evaluate the ship collision avoidance performance. All programs were run in the Visual studio 2008 and MATLAB R2009 mixed simulation environments.

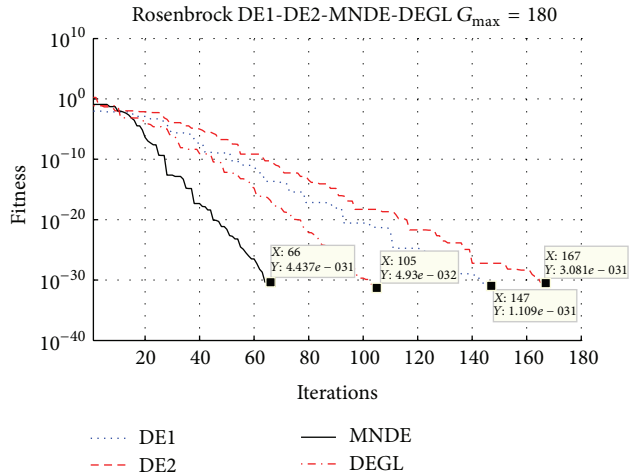

Figure 3: The evolution results for Rosenbrock function with the same predefined iterations.

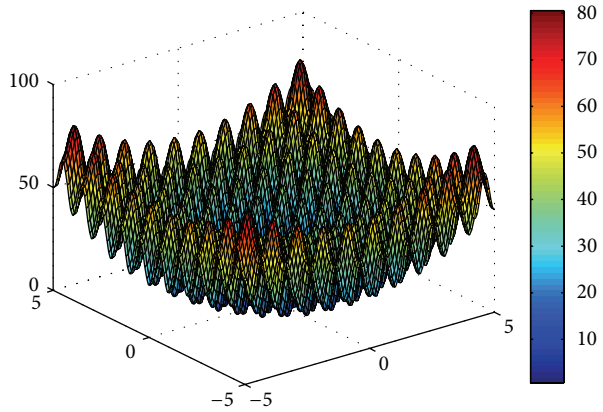

Figure 4: The solution space of the Rastrigin function.

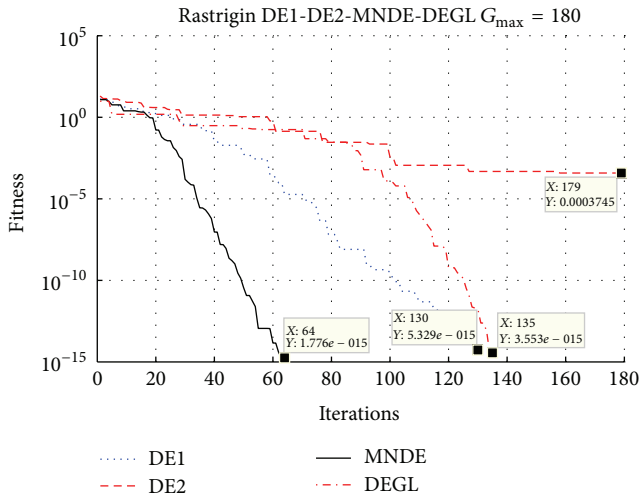

Figure 5: The evolution results of the Rastrigin function with a given predefined number of iterations.

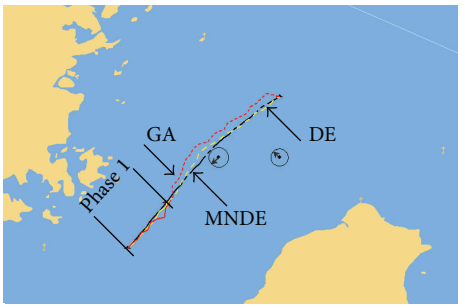

Figure 6: A noncollision avoidance scenario in open waters.

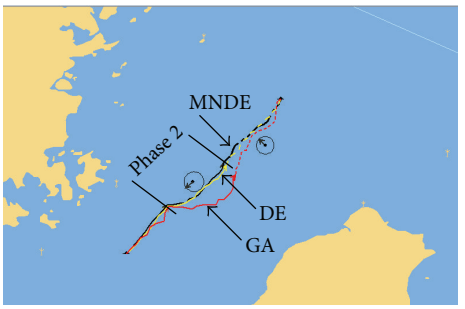

Figure 7: A head-on collision scenario in open waters.

Table 4 gives the detailed settings and operations for the different algorithms.

We simplify the fitness functions to reduce the execution time. Economy and safety are the most two important considerations for ship collision avoidance route planning. Here, we choose the shortest distance and minimum threat as the performance indicators, yielding the following least fuel consumption fitness function:

$$\min C_l = \int_0^L \omega_l \mathrm{d}l. \tag{10}$$

The minimum threat fitness function is given by

$$\min C_t = \int_0^L \omega_t \mathrm{d}l. \tag{11}$$

The total fitness function can be expressed as

$$\min C = kC_t + (1-k)\,C_l, \tag{12}$$

where L is the distance of the route, C_l is the fuel consumption model, a function of the route distance. Here we assume that ω_l equals 1, making the cost of the fuel linearly proportional to the route distance. Meanwhile, C_t indicates the threat associated with the route. Note that the weighing coefficient $k \in [0, 1]$ allows balancing cost efficiency and safety performance.

Using the settings and operation rules in Table 4, we assume that the course of the target ship is fixed, and the speed of the own and target ships is constant, as shown in Table 5. The experimental analysis focusses on two typical circumstances described in the COLREGS, namely, head-on and crossing scenarios. The avoidance operations for the following three shortest collision avoidance routes are obtained separately for the GA, DE, and MNDE algorithms.

Scenario 1. Noncollision avoidance scenario (Figure 6).

Scenario 2. A head-on collision scenario (Figure 7), requiring the own ship to pass on the right side of the target ship.

Scenario 3. A crossing scenario (Figure 8), where the own ship is required to pass on the left side of the target ship and to avoid passing the front of the target ship.

In Figures 6, 7, and 8, the black, yellow, and red lines are the routes generated by the MNDE, the standard DE, and the GA algorithms, respectively. The solid lines display

Table 4: Settings and operating rules of the three optimizing algorithms.

Algorithm	Parameters	Settings	Operations	Settings
GA	Population size	90	Population initialization	Encode each individual with real coding; each real number string represents a route
	Individual dimension	30	Selection	Roulette wheel selection
	Crossover rate	0.4	Crossover	A single point crossover
	Mutation rate	0.2	Mutation	Select the jth gene of the ith individual for mutation
	Maximum number of iterations	100	Fitness value calculation	Minimum distance + minimum threat
DE	Population size	90	Population initialization	Generate initialisation vectors randomly
	Individual dimension	30	Selection	Greedy selection
	Crossover rate	0.85	Crossover	Binomial crossover
	Scalar weight	0.6	Mutation	Disturb current solution by using differential vectors
	Maximum number of iterations	100	Fitness value calculation	Minimum distance + minimum threat
MNDE	Population size	90	Population initialization	Generate initialization vectors randomly
	Individual dimension	30	Selection	Random selection for generating the neighborhood
	Crossover rate	0.85	Crossover	Binomial crossover
	Scalar weight	0.6	Mutation	Neighborhood-based mutations
	Jittering parameter	0.0001		
	Maximum number of iterations	100	Fitness value calculation	Minimum distance + minimum threat

Table 5: Courses and speed configurations of the own and target ships.

		Speed (knots)	Course (degrees)		
			GA	DE	MNDE
Scenario 1	Own ship	25	32.15°	32.15°	32.15°
	Target 1	12		230°	
	Target 2	8		320°	
Scenario 2	Own ship	25	30.930°	32.15°	32.15°
	Target 1	12		230°	
	Target 2	8		320°	
Scenario 3	Own ship	25	26.499°	32.15°	32.15°
	Target 1	12		230°	
	Target 2	8		320°	

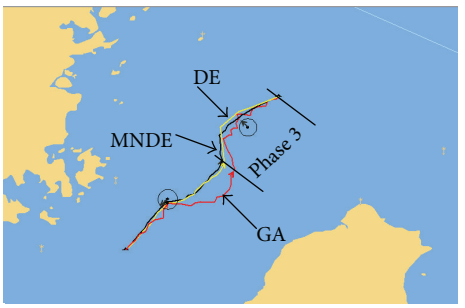

Figure 8: A crossing scenario in open waters.

the executed routes, while the dotted lines display the planned routes at the current time.

Figure 6 shows the real and planned routes in a noncollision scenario of three ships in open waters. The MNDE algorithm provides the shortest route for both the planned and executed routes.

Figure 7 shows a head-on collision scenario in open waters. In this scenario, the MNDE and standard DE algorithms are significantly better than the GA algorithm. For MNDE and standard DE, the own ship was able to resume its original route without any significant delay.

Figure 8 shows a crossing scenario. Here, we see again that the MNDE provides a better route than the standard DE or GA. The solutions provided by the MNDE and standard DE are also more reliable than that provided by the GA.

6. Conclusion

An improved differential evolution algorithm is developed in this paper. It introduces the concept of variable scaling jitter factors to maintain the diversity of the population and accelerate convergence speed. The variable scaling jitter factors enable a better trade-off between the exploitation and exploration abilities via the control parameter γ. By adopting the global optimal individual vector in the global mutation model, the convergence ability of the algorithm was significantly enhanced. Performance tests with various benchmark functions were carried out and application scenarios were evaluated to validate the algorithms' feasibility and efficiency in solving dynamic route planning problems. Three different algorithms (MNDE, DE, and GA) were run to search an ideal dynamic route in a GIS-based system. The results show

that the MNDE algorithm achieves a significantly better performance than the other two algorithms. Future work will investigate more complex scenarios, including larger number of vessels, varying ship types and speed settings, variable weather conditions and other operational restrictions, traffic conditions, and ship onboard equipment. This will lead to the development of a practical intelligent support system for vessel navigation.

Conflict of Interests

The authors declare no conflict of interests. They warrant that none of them have any financial or scientific conflict of interests with regard to the research described in this paper.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China under Grant no. 51379049 and the Fundamental Research Funds for the Central Universities of China (HEUCFX41302).

References

[1] J. M. Mou, C. V. D. Tak, and H. Ligteringen, "Study on collision avoidance in busy waterways by using AIS data," *Ocean Engineering*, vol. 37, no. 5-6, pp. 483–490, 2010.

[2] S. Yin, X. Li, H. Gao, and O. Kaynak, "Data-based techniques focused on modern industry: an overview," *IEEE Transactions on Industrial Electronics*, 2014.

[3] S. Yin, X. Ding, X. Xie, and H. Luo, "A review on basic data-driven approaches for industrial process monitoring," *IEEE Transactions on Industrial Electronics*, vol. 61, no. 11, pp. 6418–6428, 2014.

[4] F. Ahmed and K. Deb, "Multi-objective optimal path planning using elitist non-dominated sorting genetic algorithms," *Soft Computing*, vol. 17, no. 7, pp. 1283–1299, 2013.

[5] V. Roberge, M. Tarbouchi, and G. Labonte, "Comparison of parallel genetic algorithm and particle swarm optimization for real-time UAV path planning," *IEEE Transactions on Industrial Informatics*, vol. 9, no. 1, pp. 132–141, 2013.

[6] M.-C. Tsou and C.-K. Hsueh, "The study of ship collision avoidance route planning by ant colony algorithm," *Journal of Marine Science and Technology*, vol. 18, no. 5, pp. 746–756, 2010.

[7] A. Lazarowska, "Safe ship control method with the use of ant colony optimization," *Diffusion and Defect Data B: Solid State Phenomena*, vol. 210, pp. 234–244, 2014.

[8] L. P. Perera, J. P. Carvalho, and C. G. Soares, "Intelligent ocean navigation and fuzzy-Bayesian decision/action formulation," *IEEE Journal of Oceanic Engineering*, vol. 37, no. 2, pp. 204–219, 2012.

[9] C. M. Su, K. Y. Chang, and C. Y. Cheng, "Fuzzy decision on optimal collision avoidance measures for ships in vessel traffic service," *Journal of Marine Science and Technology*, vol. 20, no. 1, pp. 38–48, 2012.

[10] M. B. Zaman, E. Kobayashi, N. Wakabayashi, S. Khanfir, T. Pitana, and A. Maimun, "Fuzzy FMEA model for risk evaluation of ship collisions in the Malacca Strait: based on AIS data," *Journal of Simulation*, vol. 8, no. 1, pp. 91–104, 2014.

[11] M. L. Cummings, M. Buchin, G. Carrigan, and B. Donmez, "Supporting intelligent and trustworthy maritime path planning decisions," *International Journal of Human Computer Studies*, vol. 68, no. 10, pp. 616–626, 2010.

[12] H.-C. Kim, J.-S. Kim, Y.-K. Ji, and J.-H. Park, "Path planning of swarm mobile robots using firefly algorithm," *Journal of Institute of Control, Robotics and Systems*, vol. 19, no. 5, pp. 435–441, 2013.

[13] X. Zhong and X. Peng, "Velocity-Change-Space-based dynamic motion planning for mobile robots navigation," *Neurocomputing*, vol. 143, no. 2, pp. 153–163, 2014.

[14] S. Cao, Y. Qin, X. Jin, L. Zhao, and M. Shen, "Effect of driving experience on collision avoidance braking: an experimental investigation and computational modeling," *Behavior and Information Technology*, vol. 33, no. 9, pp. 929–940, 2014.

[15] Z. Li, J. Li, and C. He, "Artificial immune network-based anti-collision algorithm for dense RFID readers," *Expert Systems with Applications*, vol. 41, no. 10, pp. 4798–4810, 2014.

[16] R. Smierzchalski, "Evolutionary trajectory planning of ships in navigation traffic areas," *Journal of Marine Science and Technology*, vol. 4, no. 1, pp. 1–6, 1999.

[17] R. Śmierzchalski and Z. Michalewicz, "Modeling of ship trajectory in collision situations by an evolutionary algorithm," *IEEE Transactions on Evolutionary Computation*, vol. 4, no. 3, pp. 227–241, 2000.

[18] C. Y. Lee, "An algorithm for path connections and its applications," *IRE Transactions on Electronic Computers*, vol. EC-10, no. 3, pp. 346–365, 1961.

[19] K.-Y. Chang, G. E. Jan, and I. Parberry, "A method for searching optimal routes with collision avoidance on raster charts," *The Journal of Navigation*, vol. 56, no. 3, pp. 371–384, 2003.

[20] R. Szlapczynski, "A new method of ship routing on raster grids, with turn penalties and collision avoidance," *The Journal of Navigation*, vol. 59, no. 1, pp. 27–42, 2006.

[21] R. Szlapczynski, "A new method of planning collision avoidance manoeuvres for multi-target encounter situations," *The Journal of Navigation*, vol. 61, no. 2, pp. 307–321, 2008.

[22] R. Szlapczynski, "Evolutionary sets of safe ship trajectories: a new approach to collision avoidance," *The Journal of Navigation*, vol. 64, no. 1, pp. 169–181, 2011.

[23] R. Szlapczynski, "Evolutionary sets of safe ship trajectories within traffic separation schemes," *The Journal of Navigation*, vol. 66, no. 1, pp. 65–81, 2013.

[24] C. K. Tam, R. Bucknall, and A. Greig, "Review of collision avoidance and path planning methods for ships in close range encounters," *The Journal of Navigation*, vol. 62, no. 3, pp. 455–476, 2009.

[25] C. H. Shih, P. H. Huang, S. Yamamura, and C. Y. Chen, "Design optimal control of ship maneuver patterns for collision avoidance: a review," *Journal of Marine Science and Technology*, vol. 20, no. 2, pp. 111–121, 2012.

[26] R. Storn and K. D. Price, *Differential Evolution—A Simple and Efficient Adaptive Scheme for Global Optimization Over Continuous Spaces*, ICSI, Berkeley, Calif, USA, 1995.

[27] S. Das and P. N. Suganthan, "Differential evolution: a survey of the state-of-the-art," *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 1, pp. 4–31, 2011.

[28] M. A. Panduro, C. A. Brizuela, L. I. Balderas, and D. A. Acosta, "A comparison of genetic algorithms, particle swarm optimization and the differential evolution method for the design of scannable circular antenna arrays," *Progress in Electromagnetics Research B*, vol. 13, pp. 171–186, 2009.

[29] K. Harnrnouche, M. Diaf, and P. Siarry, "A comparative study of various meta-heuristic techniques applied to the multilevel thresholding problem," *Engineering Applications of Artificial Intelligence*, vol. 23, no. 5, pp. 676–688, 2010.

[30] R. Storn, K. Price, and J. Lampinen, *Differential Evolution—A Practical Approach to Global Optimization*, Springer, Berlin , Germany, 2005.

[31] S. Das, A. Abraham, U. K. Chakraborty, and A. Konar, "Differential evolution using a neighborhood-based mutation operator," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 3, pp. 526–553, 2009.