# Postprint of Multi-objective Particle Swarm Feature Selection Algorithm Based on Differential Evolution

**Authors:** Li Min, Zhang Guohao, Chen Ziliang, Guo Zhiyong, Hu Xiaomin

**Date:** 2018-11-29T00:00:00+00:00

## Abstract

Feature selection techniques play a significant role in big data analytics, image processing, bioinformatics, and other domains. In practical applications, reducing classification error rate and decreasing the number of extracted features to facilitate subsequent data utilization are often two conflicting objectives. The crowding, mutation, and dominance particle swarm optimization for feature selection (CMDPSOFS) algorithm is a bi-objective optimization algorithm for feature selection applications that aims to minimize both the number of features and the classification error rate. It employs three different mutation mechanisms to maintain population diversity and balance global and local search capabilities; however, the uniform mutation significantly increases the algorithm' s randomness, generates numerous solutions with poor fitness values, and reduces the convergence speed. The improved CMDPSOFS-II algorithm introduces the mutation operator and selection operation from the differential evolution algorithm into the CMDPSOFS algorithm. Experimental results demonstrate that the CMDPSOFS-II algorithm achieves superior results in feature selection compared to the original method and better balances global and local search capabilities.

## Full Text

## Preamble

**Vol. 37 No. 1**
**Application Research of Computers**
**ChinaXiv Cooperative Journal**

**Multi-Objective Particle Swarm Optimization Algorithm Using Differential Evolution for Feature Selection**

Li Mina,b, Zhang Guohaoa, Chen Ziliangb, Guo Zhiyongb, Hu Xiaominb†

aSchool of Information Engineering, bSchool of Computers
Guangdong University of Technology, Guangzhou 510006, China

**Abstract:** Feature selection technology plays a crucial role in big data analysis, image processing, bioinformatics, and other fields. In practical applications, reducing classification error rate and minimizing the number of extracted features for subsequent data utilization often represent two conflicting objectives. The multi-objective particle swarm optimization algorithm based on crowding, mutation, and dominance for feature selection (CMDPSOFS) is a bi-objective optimization algorithm designed for feature selection applications that aims to minimize both the number of features and the classification error rate. CMDPSOFS employs three different mutation mechanisms to maintain population diversity and balance global and local search capabilities. However, the uniform mutation significantly increases algorithmic randomness, generating numerous solutions with poor fitness values and reducing convergence speed. This paper proposes an improved CMDPSOFS-II algorithm that introduces the mutation operator and selection operation from differential evolution into CMDPSOFS. Experimental results demonstrate that CMDPSOFS-II achieves superior results in feature selection compared to the original method, providing better balance between global and local search capabilities.

**Keywords:** feature selection; particle swarm optimization; mutation; differential evolution

## 0 Introduction

Classification, as a crucial step in machine learning and data mining, assigns each instance to different categories based on features in the dataset [1]. For classification, irrelevant and redundant features are clearly meaningless and may even degrade classification performance due to the expanded search space. Feature selection addresses this by extracting representative features from the data, which is a common approach to reduce classifier training time and improve classification capability [2].

Feature selection technology holds significant importance in big data analysis [3], image processing [4], and bioinformatics [5]. As the number of features in a dataset increases, the search space grows exponentially, making exhaustive search impossible in most cases. To address this challenge, numerous search methods have been applied to feature selection. For instance, Wang et al. [6] proposed a novel bacterial algorithm for feature selection, Belciug et al. [7] introduced a regression-based feature selection algorithm, Duan et al. [8] developed a multi-label classification feature selection algorithm based on neighborhood rough sets, and Xie et al. [9] presented a feature selection algorithm combining DFS and SVM.

Particle swarm optimization (PSO), as a robust and highly applicable swarm intelligence method, has also been applied to feature selection. Naeini et al. [4] utilized PSO for feature selection in high spatial resolution satellite image classification, achieving higher recognition rates with selected feature combinations. Zhang et al. [10] proposed a swarm intelligence-based algorithm for feature selection in acoustic defect detection. In practical applications, beyond the single objective of reducing classification error rate, it is often necessary to simultaneously minimize the number of extracted features to reduce the total cost of obtaining required feature values. These requirements frequently conflict with the objective of reducing classification error rate.

Multi-objective optimization techniques offer an effective means to solve multiple conflicting objectives. Zhang et al. [11] proposed a multi-objective PSO algorithm for cost-based feature selection, optimizing both classification capability and feature cost. Xue et al. [12] formulated minimizing the number of extracted features and classification error rate as two objectives for feature selection, proposing two multi-objective PSO algorithms: non-dominated sorting PSO for feature selection (NSPSOFS) and crowding, mutation, and dominance PSO for feature selection (CMDPSOFS). The key distinction between these algorithms lies in CMDPSOFS maintaining inheritance of particle local optima during iterations, aligning more closely with PSO principles rather than using sorting that makes new particles nearly unrelated to previous generations. Tests also demonstrated that CMDPSOFS generally obtains better solutions than NSPSOFS. However, the uniform mutation in CMDPSOFS significantly increases algorithmic randomness, generating many solutions with poor fitness values and reducing convergence speed.

This paper proposes an improved CMDPSOFS-II algorithm that integrates the mutation and selection operations from differential evolution [13] into CMDPSOFS, replacing uniform mutation for generating new particles and incorporating differential evolution' s selection operation. By introducing differential evolution's mutation and selection mechanisms, the algorithm can adaptively select mutation step sizes based on vector differences during iterations, maintaining population diversity while improving mutation efficiency. Through comparative testing on a series of feature selection problems, experimental results show that the improved CMDPSOFS-II algorithm achieves superior results compared to the original method, providing better balance between global and local search capabilities.

---

# 1 Background

## 1.1 Particle Swarm Optimization Algorithm

The particle swarm optimization algorithm was first proposed by Kennedy et al. [14]. The algorithm initializes a swarm of random particles (random solutions), with each particle possessing its own velocity and position. The entire

particle swarm then undergoes iterative updates, where each particle updates itself based on its personal best solution (pbest) and the global best solution (gbest). Let $v_{i,j}$ and $x_{i,j}$ denote the velocity and position of particle $i$ in dimension $j$, respectively. The velocity and position update formulas are given by:

$$v_{i,j} = \omega v_{i,j} + c_1 r_1 (p_{i,j} - x_{i,j}) + c_2 r_2 (g_{i,j} - x_{i,j}) \tag{1}$$

$$x_{i,j} = x_{i,j} + v_{i,j} \tag{2}$$

where $\omega$ is the inertia weight controlling the influence of previous evolution results on the current evolution; $c_1$ and $c_2$ are acceleration coefficients measuring the guidance degree of the particle' s historical optimum on its evolution; $r_1$ and $r_2$ are random numbers in $[0, 1]$; $p_{i,j}$ and $g_{i,j}$ represent the personal best solution (pbest) and global best solution (gbest) of particle $i$ in dimension $j$, respectively. The velocity $v_{i,j}$ is limited by a preset maximum velocity $v_{\max}$, with $v_{i,j} \in [-v_{\max}, v_{\max}]$. The algorithm terminates when preset results are achieved or the maximum iteration count is reached.

### 1.2 Multi-Objective Optimization

When selecting an optimal solution requires balancing two or more objectives that have contradictory relationships, the problem is termed a multi-objective optimization problem. Multi-objective optimization problems include maximization and minimization problems. A multi-objective minimization problem can be expressed as:

$$\min F(x) = (f_1(x), f_2(x), \dots, f_k(x))$$

where $x$ is the decision vector, $f_i(x)$ is the objective function regarding $x$, and $k$ is the number of objectives to be optimized.

Decision vector $u$ is said to dominate $v$ when the following two conditions are satisfied:

$$\forall i \in \{1, 2, 3, \dots, k\} : f_i(u) \leq f_i(v) \quad \text{and} \quad \exists j \in \{1, 2, 3, \dots, k\} : f_j(u) < f_j(v)$$

As shown in the bi-objective minimization problem in [Figure 1: see original paper], $x_1$ dominates both $x_2$ and $x_3$, while $x_2$ and $x_3$ do not dominate each other. When a solution is not dominated by any other solution, it is called a Pareto-optimal solution. The surface formed by all Pareto-optimal solutions in the search space is called the Pareto front. The solid points in Figure 1 represent solutions on the Pareto front, which are mutually non-dominated and not dominated by any other solutions.

## 2 Problem Definition and Existing Algorithms

### 2.1 Problem Definition

Feature selection is the process of selecting an optimal set of features from a dataset with a certain number of features. However, due to complex relationships among features, the enormous number of possible feature combinations makes it impossible to evaluate every combination. Therefore, feasible algorithms are required for optimization. The difficulty of feature selection lies in feature combinations: a relevant feature may become redundant when combined with others, while an irrelevant or redundant feature may become relevant when combined with different features. An ideal feature subset should be a complementary set that correctly classifies different instances when features are combined.

Currently, although many feature selection algorithms have been proposed, most focus solely on reducing classification error rate. While increasing feature count can reduce classification error, excessive features make classification difficult and hinder subsequent data analysis. Simultaneously improving acceptable classification error rates while reducing feature count represents two conflicting objectives. Multi-objective optimization techniques can optimize both objectives simultaneously, yielding a set of multi-objective optimal solutions—the Pareto front. Through further analysis of these solutions, we can trade off between feature count and classification error rate to select feature combinations with relatively fewer features but lower classification error for practical classification applications.

This paper studies a multi-objective feature selection problem using classification error rate $f_1$ and feature count $f_2$ as two optimization objectives, defined as follows:

$$f_1 = \frac{FP + FN}{TP + TN + FP + FN}$$

$$f_2 = |V|$$

In equation (10), $P$ or $N$ represents whether an observed sample (actual sample) belongs to a certain class, while $T$ or $F$ represents the prediction result. If the sample' s observation matches the prediction result, it is $T$; otherwise, it is $F$. This equation calculates the classification error rate. Equation (11) corresponds to the number of elements in the selected feature set $V$.

For this bi-objective feature selection optimization problem, Xue et al. [12] proposed the non-dominated sorting PSO for feature selection (NSPSOFS) algorithm and the crowding, mutation, and dominance PSO for feature selection (CMDPSOFS) algorithm. The following sections describe these algorithms' implementation, while Section 3 details the proposed improved algorithm.

### 2.2 NSPSOFS Algorithm

NSPSOFS applies a non-dominated sorting-based multi-objective PSO to feature selection. The two most important steps are selecting the global best solution (gbest) and updating the swarm during iterations. In each iteration, NSPSOFS first evaluates each particle' s fitness values (feature count and classification error rate), then identifies the non-dominated particle set in the swarm. It calculates the crowding distance for each non-dominated particle and sorts the non-dominated particles in descending order of crowding distance. When updating swarm particles, gbest is randomly selected from non-dominated particles with the smallest crowding distance, while the personal best solution (pbest) is a solution that the particle has not been dominated by in each iteration. When an updated solution dominates a particle' s pbest, the pbest is replaced.

After determining pbest and gbest, particles update their positions and velocities according to equations (1) and (2). Both updated and previous particles are added to a union set, which is then partitioned into subsets $F = (F_1, F_2, F_3, \dots, F_k)$ based on different non-domination levels, where $k$ represents the maximum number of non-dominated subsets. The swarm is cleared, and particles are added from subset $F_1$ onward. If the swarm requires more particles than the current non-dominated subset contains, the entire current non-dominated subset is added; otherwise, particles from the current non-dominated subset are added in descending order of crowding distance until the preset swarm size $P$ is reached.

### 2.3 CMDPSOFS Algorithm

Due to NSPSOFS' s deficiency in reducing particle diversity and its sorting process in each iteration causing pbest records not to be generated from the particle' s own solutions, Xue et al. [12] further investigated CMDPSOFS, which adds crowding, mutation, and dominance methods to multi-objective PSO. This algorithm better aligns with PSO principles and maintains inheritance of particle local optima during iterations.

[Figure 2: see original paper] illustrates the CMDPSOFS algorithm flow, where $N$ represents the swarm size. The algorithm uses a LeaderSet to store non-dominated solutions, with each particle' s gbest selected from particles with smallest crowding distance in LeaderSet using binary tournament selection. Non-dominated particles in each generation update LeaderSet and are added to the Archive set. The algorithm iterates until stopping conditions are met.

CMDPSOFS employs mutation operations to increase population diversity. In the mutation step, swarm particles are randomly divided into three equal groups: - **Group a**: No mutation is performed - **Group b**: Uses uniform mutation with mutation probability equal to the reciprocal of total feature count, obtaining mutation vectors through random values within the vector' s domain - **Group c**: Uses non-uniform mutation with the same probability as Group b. As iteration count increases, the mutation value range shrinks, making this mutation

approach highly localized in later stages

---

## 3 CMDPSOFS-II Algorithm Based on Differential Evolution Improvement

### 3.1 Improvements to CMDPSOFS

The proposed CMDPSOFS-II algorithm combines CMDPSO with differential evolution' s mutation and selection mechanisms for feature selection. Building upon CMDPSOFS' s mutation step, the uniform mutation operator is replaced with differential evolution' s mutation operator, and differential evolution' s selection step is added. The specific process is shown in [Figure 3: see original paper].

When CMDPSOFS-II uses the differential mutation operator for particle mutation, if a mutated value exceeds the variable' s preset range $[l_k, u_k]$, there is a 50% probability of taking the middle value of the preset range, i.e., $(l_k + u_k)/2$, or a 50% probability of taking the pre-mutation value. Since parent individuals are randomly selected for differential recombination, the algorithm' s mutated particles only use difference vectors from two random particles in the parent population for correction, maintaining population diversity and improving mutation operation efficiency.

The differential mutation process in CMDPSOFS-II also incorporates a selection operation: when the mutated particle' s fitness value is better than the parent' s fitness value, the mutated particle is selected; otherwise, the parent particle is retained. This operation accelerates algorithm convergence speed.

### 3.2 CMDPSOFS-II Algorithm Flow

**Step 1**: Initialize the swarm, LeaderSet, and Archive. Calculate crowding distances for particles in LeaderSet, and randomly divide swarm particles into three equal groups.

**Step 2**: Iterate for the preset number of iterations: - **2.1**) For each particle in the swarm: - Select the particle's gbest from LeaderSet using binary tournament based on crowding distance - Update particle velocity and position - Execute mutation operation: 1. Particles in Group 1 undergo no operation 2. Particles in Group 2 undergo the proposed differential mutation. Evaluate the mutated particle's fitness; if it dominates the current particle, replace the current particle with the mutated one 3. Particles in Group 3 undergo non-uniform mutation - Calculate each particle' s fitness value and update its pbest - **2.2**) Identify non-dominated particles in the swarm and use them to update LeaderSet - **2.3**) Save particles from LeaderSet to Archive - **2.4**) Calculate crowding distance for each particle in LeaderSet

**Step 3**: Calculate fitness values (feature count and classification error rate) for each particle in Archive and return the results.

The algorithm' s inertia weight $\omega$ and acceleration coefficients $c_1$ and $c_2$ are set the same as in CMDPSOFS, with mutation rate at $1/n$ where $n$ is the problem dimension (i.e., feature count). The mutation factor $K$ in differential mutation is set to 0.5.

---

## 4 Experimental Results

### 4.1 Datasets and Parameter Settings

This study selected datasets from the UCI Machine Learning Repository [18] as shown in . Each dataset was divided into training and test sets, comprising 70% and 30% of total instances, respectively. The experiments compared the proposed CMDPSOFS-II with NSPSOFS [12] and CMDPSOFS [12]. All algorithms used support vector machines for classification training and testing, employing Chih-Chung' s LIBSVM [19] software on a Java platform.

Following literature [12], the test parameters were set as: $v_{\max} = 0.6$, population size $N = 30$, maximum iterations $T = 500$, and $\theta = 0.6$ for determining feature selection. In NSPSOFS, inertia weight $\omega = 0.7298$ and acceleration coefficients $c_1 = c_2 = 1.49618$. In CMDPSOFS and CMDPSOFS-II, inertia weight $\omega$ is a random number in $[0.1, 0.5]$, acceleration coefficients $c_1$ and $c_2$ are random numbers in $[1.5, 2.0]$, and mutation rate is set to $1/n$. All algorithms were independently run 30 times in tests.

shows the dataset characteristics, including total features and classification error rates when using all features.

### 4.2 Experimental Results

Comparing CMDPSOFS-II, NSPSOFS, and CMDPSOFS, [Figure 4: see original paper] presents the Pareto fronts corresponding to the optimal non-dominated solution sets of the three algorithms. The curves represent the best non-dominated results obtained from 30 independent runs of CMDPSOFS, NSPSOFS, and CMDPSOFS-II on each dataset. Subtitles indicate dataset names with data in parentheses showing total feature count and classification error rate when using all features. Using all features may degrade training accuracy due to irrelevant features, so algorithm-optimized solutions may achieve lower classification error rates than using all features.

**Analysis of NSPSOFS results**: Except for Ionosphere, NSPSOFS achieved classification error rates better than using all features across all other datasets, with feature counts smaller than total features. Notably, on the WBCD dataset, only 10% of features (3 out of 30) were needed to achieve lower classification error than using all features. On Vehicle, only 11% of features (2 out of 18)

were selected with error rates below the overall classification error. Across tested datasets, NSPSOFS reduced feature count to an average of 20% of total features.

**Analysis of CMDPSOFS results**: CMDPSOFS obtained at least one feature subset with classification error rate lower than using all features in every dataset test. CMDPSOFS reduced the average feature count to 22% of total features. On datasets WBCD and Australian, CMDPSOFS selected only 7% of total features (2 out of 30 on WBCD, 1 out of 14 on Australian) while achieving classification error rates lower than using all features. In most solution sets, CMDPSOFS obtained more and more uniformly distributed non-dominated solutions compared to NSPSOFS, and achieved lower classification error rates for the same feature count, indicating superior feature quality.

**Analysis of CMDPSOFS-II results**: CMDPSOFS-II also obtained at least one feature subset with classification error rate lower than using all features in every dataset test. CMDPSOFS-II reduced average feature count to 14% of total features. On dataset Vehicle, it selected only 6% of total features (1 out of 18), and on WBCD, 7% (2 out of 30), while maintaining classification error rates below those achieved using all features.

lists the minimum feature numbers with error rates lower than using all features for the three algorithms, with classification error rates in parentheses. CMDPSOFS-II achieves lower feature counts in most cases. When feature counts are equal, CMDPSOFS-II selects feature combinations with lower error rates, particularly on Hill-Valley where both feature count and error rate are lower than NSPSOFS and CMDPSOFS.

Comparing CMDPSOFS-II with NSPSOFS and CMDPSOFS, CMDPSOFS demonstrates clear advantages over NSPSOFS in solution diversity, feature count, and classification error rate optimization. In all cases, CMDPSOFS-II outperforms NSPSOFS in optimizing both feature count and classification performance. For example, on the German dataset with equal feature counts, CMDPSOFS-II achieves lower classification error rates. Except for Ionosphere and Australian where CMDPSOFS slightly outperforms CMDPSOFS-II on a few solutions, CMDPSOFS-II' s non-dominated solution sets are superior to CMDPSOFS in all other tests. CMDPSOFS-II not only inherits CMDPSOFS' s population diversity but also enhances optimization capability for feature count and classification performance.

---

## 5 Conclusion

CMDPSOFS-II is an optimization algorithm based on differential evolution operations, proposed for feature selection applications of CMDPSOFS. By introducing differential evolution' s mutation and selection mechanisms into a portion of swarm particles, CMDPSOFS-II inherits CMDPSOFS' s particle diversity while suppressing algorithmic randomness to some extent, improving mutation

effectiveness and accelerating CMDPSOFS' s convergence speed.

## References

[1] Dash B M, Liu H. Feature selection for classification [J]. Intelligent Data Analysis, 1997, 1(3): 131-156.

[2] Unler A, Murat A. A discrete particle swarm optimization method for feature selection in binary classification problems [J]. European Physical Journal Applied Physics, 2009, 206(3): 528-539.

[3] Fong S, Wong R, Vasilakos A V. Accelerated PSO swarm search feature selection for data stream mining big data [J]. IEEE Trans on Services Computing, 2016, 9(1): 33-45.

[4] Naeini A A, Babadi M, Mirzadeh S M J, et al. Particle swarm optimization for object-based feature selection of VHSR satellite images [J]. IEEE Geoscience and Remote Sensing Letters, 2018, 15(3): 379-383.

[5] Han Fei, Yang Chun, Wu Yaqi, et al. A gene selection method for microarray data based on binary PSO encoding gene-to-class sensitivity information [J]. IEEE//ACM Trans on Computational Biology and Bioinformatics, 2017, 14(1): 85-96.

[6] Wang Hong, Niu Ben. A novel bacterial algorithm with randomness control for feature selection in classification [J]. Neurocomputing, 2017, 228: 192-201.

[7] Belciug S, Serbanescu M S. Regression-based approach for feature selection in classification issues. application to breast cancer detection and recurrence [J]. ACTA Universitatis Cibiniensis Technical Series, 2015, 67(1): 13-18.

[8] Duan Jie, Hu Qinghua, Zhang Lingjun, et al. Feature selection for multi-label classification based on neighborhood rough sets [J]. Journal of Computer Research and Development, 2015, 52(1): 56-65.

[9] Xie Juanying, Xie Weixin. Several feature selection algorithms based on the discernibility of a feature subset and support vector machines [J]. Chinese Journal of Computers, 2014, 37(8): 1704-1718.

[10] Zhang Tao, Ding Biyun, Zhao Xin, et al. A fast feature selection algorithm based on swarm intelligence in acoustic defect detection [J]. IEEE Access, 2018, 6: 28848-28858.

[11] Zhang Yong, Gong Dunwei, Cheng Jian. Multi-objective particle swarm optimization approach for cost-based feature selection in classification [J]. IEEE//ACM Trans on Computational Biology and Bioinformatics, 2017, 14(1): 64-75.

[12] Xue Bing, Zhang Mengjie, Browne W N. Particle swarm optimization for feature selection in classification: a multi-objective approach [J]. IEEE Trans on Cybernetics, 2013, 43(6): 1656-1671.

[13] Storn R, Price K. Differential evolution: a simple and efficient adaptive scheme for global optimization over continuous spaces [R]. Berkeley: University of California, 2006.

[14] Kennedy J, Eberhart R. Particle swarm optimization [C]// Proc of IEEE Neural Networks. 1995: 1942-1948.

[15] Storn R. On the usage of differential evolution for function optimization [C]// Proc of Fuzzy Information Processing Society. 1996: 519-523.

[16] Chen Ying, Lin Ying, Hu Xiaomin. Parallel differential evolution with multi-population and multi-strategy [J]. Journal of Frontiers of Computer Science & Technology, 2014, 8(12): 1502-1510.

[17] Chen Tao, Yong Longquan, Deng Fang' an, et al. Parameters selection of support vector machine based on differential evolution [J]. Computer Engineering and Applications, 2011, 47(5): 24-26.

[18] Lichman, M. UCI Machine Learning Repository [DB/OL]. (2007) [2013-04-04]. http://archive.ics.uci.edu/ml.

[19] Chang Chih Chung, Lin Chih Jen. LIBSVM: a library for support vector machines [EB/OL]. (2011) [2016-12-22]. http://www.csie.ntu.edu.tw/~cjlin/libsvm.

*Note: Figure translations are in progress. See original paper for figures.*