Noname manuscript No.
(will be inserted by the editor)

Feature Selection for High Dimensional Classification using A Competitive Swarm Optimizer

Shenkai Gu · Ran Cheng · Yaochu Jin

Received: date / Accepted: date

Abstract When solving many machine learning problems such as classification, there exists a large number of input features. However, not all features are relevant for solving the problem, and sometimes, including irrelevant features may deteriorate the learning performance. Therefore, it is essential to select the most relevant features, which is known as feature selection. Many feature selection algorithms have been developed, including evolutionary algorithms or particle swarm optimization (PSO) algorithms, to find a subset of the most important features for accomplishing a particular machine learning task. However, the traditional PSO does not perform well for large scale optimization problems, which degrades the effectiveness of PSO for feature selection when the number of features dramatically increases. In this paper, we propose to use a very recent PSO variant, known as competitive swarm optimizer (CSO) that was dedicated to large-scale optimization, for solving high-dimensional feature selection problems. In addition, the CSO, which was originally developed for continuous optimization, is adapted to performing feature selection that can be considered as a combinatorial optimization problem. An archive technique is also introduced to reduce computational cost. Experiments on six benchmark datasets demonstrate that compared to the canonical PSO based and a state-of-the-art PSO variant for feature selection, the proposed CSO-based

Shenkai Gu
Department of Computer Science, University of Surrey, Guildford, Surrey, GU2 7XH, United Kingdom

Ran Cheng
Department of Computer Science, University of Surrey, Guildford, Surrey, GU2 7XH, United Kingdom

Yaochu Jin
1 Department of Computer Science, University of Surrey, Guildford, Surrey, GU2 7XH, United Kingdom
2 School of Management Science and Engineering, Dalian University of Technology, Dalian, China; 116023
E-mail: yaochu.jin@surrey.ac.uk

feature selection algorithm not only selects a much smaller number of features, but result in better classification performance as well.

Keywords Feature selection · High dimensionality · Large-scale optimization · Classification · Competitive swarm optimization

1 Introduction

In machine learning and data mining, the target concepts of a dataset is usually described by a group of features. To build reliable models for solving problems such as classification, it is expected that the features contain as much useful information as possible, and the number of features can be as small as possible. However, since there is often little priori knowledge on the dataset, it is difficult to distinguish which features are relevant and which are not. As a consequence, there are usually a large number of features to be taken into consideration, including many irrelevant and redundant features. Unfortunately, irrelevant and redundant features will not only reduce the training efficiency, but also negatively influence the performance of machine learning thus trained with them, which is mainly caused by the *curse of dimensionality* [16].

To eliminate the negative impact of the irrelevant and redundant features, various feature selection methods have been proposed [6]. The main target of feature selection is to choose relevant features from a large feature set [24, 17]. From the optimization point of view, feature selection is a difficult combinatorial optimization problem [16]. Firstly, since the size of the feature subset is not known *a priori*, the dimensionality of the decision space is non-reducible. Secondly, since the features may have complementary or contradictory interactions with each other, the decision space is non-separable. Thus, given an m-dimensional feature set, the number of all possible feature subsets is as large as 2^m, which makes it very unlikely (if not impossible) to solve it with traditional exhaustive search approaches.

Due to the inefficiency of traditional search approaches in solving complex combinatorial optimization problems, various metaheuristics have been proposed, such as genetic algorithms (GAs) [19], differential evolution (DE) [31], and particle swarm optimization (PSO) [4], among many others [5]. Among various metaheuristics, PSO is well known for its algorithmic simplicity and computational efficiency. Recently, some researchers have proposed to apply PSO to feature selection [36, 13]. However, canonical PSO has many limitations for feature selection [41, 42]. Firstly, PSO was originally proposed for continuous optimization problems, while feature selection is a combinatorial optimization problem. Secondly, although PSO shows promising performance on low-dimensional problems, it suffers from the *curse of big dimensionality* [43]. Very recently, Cheng and Jin have proposed a PSO variant, known as competitive swarm optimizer (CSO), for large-scale optimization [10]. In CSO, both the global best position and the personal best position are removed. Instead of learning from the global and personal best positions, the particles in

CSO learn from randomly selected competitors. CSO shows promising performance on a variety of continuous test problems of a dimension up to 5000, in comparison with some state-of-the-art algorithms for large-scale optimization.

In this work, we propose to adopt CSO for high-dimensional feature selection. To this end, CSO is modified to be suited for combinatorial optimization. The modified CSO variant is then embedded in a wrapper feature selection approach. The remainder of this paper is organized as follows. Section 2 briefly describes related work, including the canonical PSO algorithm and its relevant variants. Section 3 details the proposed method. We present the experimental results and discussions in Section 4. Finally, Section 5 concludes this paper.

2 Background

2.1 The Canonical PSO Algorithm

The canonical PSO algorithm was developed by Kennedy and Eberhart in 1995 to solve optimization problems by emulating social swarm behaviors of animals like bird flocking [21].

In PSO, each particle maintains a position and a velocity in an n-dimensional search space, representing a candidate solution and direction to a potentially better solution. To search for the position of the global optimum, each particle is iteratively updated as follows:

$$v_i^{t+1} = \omega v_i^t + \phi_1 R_1^t(\hat{g}^t - x_i^t) + \phi_2 R_2^t(\hat{x}_i^t - x_i^t) \tag{1}$$

$$x_i^{t+1} = x_i^t + v_i^{t+1} \tag{2}$$

where t denotes the generation number, x_i^t and v_i^t denote the position and velocity of the i-th particle in the t-th generation, respectively, ω is termed inertia weight [33], ϕ_1 and ϕ_2 are acceleration coefficients, R_1^t and R_2^t are two randomly generated vectors within the range $[0,1]^n$, $\hat{g}^t$ is the best solution found by all particles so far, also known as the global best position, and $\hat{x}_i^t$ is the best solution found by i-th particle so far, also known as the personal best position.

Although PSO has witnessed a great success over the past two decades, its performance is still limited when the optimization problem has a high-dimensional and complex search space [25,11]. In order to enhance the performance of PSO, a number of PSO variants have been proposed, including the parameter adaptation based variants [44,20], the new topological structure based variants [26,9], and the hybridization based variants [7,37], to name a few.

2.2 The CSO Algorithm

Despite many PSO variants have been proposed, not much work has been done on developing PSO for large-scale optimization. The performance improvement

of most existing PSO variants is at the cost of higher computational complexity and more complex algorithmic implementation. In addition, existing PSO variants attempt to modify the global or personal best positions, resulting in only limited performance improvement for large-scale optimization.

The recently proposed CSO [10] has shown to be efficient for large-scale optimization. In CSO, the particles learn from randomly selected competitors, instead of from the global or the personal best position. In each iteration, the swarm is randomly divided into two groups and pairwise competitions are carried out between the particles from each group. After each competition, the winner particle is directly passed to the next iteration, while the loser particle will update its position and velocity by learning from the winner particle:

$$v_l^{t+1} = R_1^t v_l^t + R_2^t(x_w^t - x_l^t) + \phi R_3^t(\bar{x}^t - x_l^t) \tag{3}$$

$$x_l^{t+1} = x_l^t + v_l^{t+1} \tag{4}$$

where t is the iteration counter, R_1^t, R_2^t, R_3^t are three randomly generated vectors within $[0,1]^n$, x_w^t and x_l^t denote the winner particle and the loser particle, respectively, $\bar{x}^t$ denotes the mean position of current swarm in iteration t, and ϕ controls the influence of $\bar{x}^t$. The detailed procedure of CSO is summarized in Algorithm 1.

Algorithm 1 The CSO algorithm

$t \leftarrow 0$
for all particle $p_i^t = \langle x_i^t, v_i^t \rangle$ in swarm P^t **do**
    initialize position x_i^t and velocity v_i^t
end for
while termination criteria is not met **do**
    for all particle p_i^t **do**
        calculate fitness $f(x_i^t)$
    end for
    $P^{t+1} \leftarrow \varnothing$
    while $P^t \neq \varnothing$ **do**
        randomly chose two different particles p_{r1}^t and p_{r2}^t from P^t
        if $f(x_{r1}^t)$ is better than $f(x_{r2}^t)$ **then**
            $p_w^t \leftarrow p_{r1}^t$, $p_l^t \leftarrow p_{r2}^t$
        else
            $p_w^t \leftarrow p_{r2}^t$, $p_l^t \leftarrow p_{r1}^t$
        end if
        $v_l^{t+1} = R_1^t v_l^t + R_2^t(x_w^t - x_l^t) + \phi R_3^t(\bar{x}^t - x_l^t)$
        $x_l^{t+1} = x_l^t + v_l^{t+1}$
        $P^{t+1} \leftarrow P^{t+1} \cup \{p_w^t, p_l^{t+1}\}$
        $P^t \leftarrow P^t \setminus \{p_{r1}^t, p_{r2}^t\}$
    end while
    $t \leftarrow t + 1$
end while

2.3 Metaheuristics for Feature Selection

Traditional feature selection approaches can be roughly categorized into two classes, filter approaches and wrapper approaches. Filter approaches are independent of any specific learning algorithms [22,2], while wrapper approaches involve learning algorithms as part of the evaluation procedure [39,23]. Some representative wrapping methods include Sequential Floating Selection (SFS) [39], Sequential Forward Floating Selection (SFFS) [32] and sparse logistic regression based methods [27,29,14,34].

Using metaheuristics for feature selection has been popular recently. For example, Zhu *et al.* have proposed to use a GA combined with local search in a hybrid wrapper and filter feature selection algorithm [45]. Neshatian and Zhang designed a genetic programming (GP) based multi-objective algorithm for filter feature selection in [30]. Chen *et al.* have applied ant colony optimization (ACO) together with rough set theory for feature selection [8]. In particular, PSO, as a popular metaheuristics, has also been widely adopted for feature selection [40]. Chuang *et al.* have developed an improved binary PSO algorithm for feature selection using gene expression data [12]. Wang *et al.* have suggested a filter feature selection approach based on rough set and PSO [38]. Li and Chen have reported a wrapper feature selection algorithm based on PSO and a linear discrimination analysis (LDA) algorithm, known as the PSOLDA [28]. Xue *et al.* have presented a multi-objective PSO algorithm for feature selection in [41]. In spite of the various metaheuristics applied to feature selection, however, little study has been dedicated to selecting a subset from a large-scale feature except some recent works [15,35].

3 Proposed Method

In our proposed method, feature selection can be formulated as the following minimization problem:

$$\begin{aligned} &\min_{x} \ f(x) \\ &\text{s.t.} \ \ x \in \mathcal{X} \end{aligned} \tag{5}$$

where $\mathcal{X} \in \mathbb{R}^N$ denotes the feasible solution set. To represent the selected feature sets, x is encoded by a number of N binary bits, where N is the total number of features in the original feature set. For each bit in x, '1' and '0' denotes that the corresponding feature is or is not selected, respectively. In this way, feature selection becomes a combinatorial optimization problem where the objective is to find the best feature subset x^* to minimize the error rate of the classification models thus trained with the selected features, which is represented by the fitness function $f(x)$.

To solve high-dimensional feature selection problem presented in (5), CSO algorithm is employed in this work. However, in feature selection, the search landscape is discrete, while the original CSO algorithm has been proposed for continuous optimization. In order to address this issue, we use a threshold

parameter λ to determine whether a feature is selected or not, as shown in Algorithm 2.

Algorithm 2 Converting continuous values to discrete (binary) values for feature selection

1: $\mathcal{S} \leftarrow \varnothing$, $\mathcal{S}$ is the selected feature subset;
2: **for all** $d_i \in x, i = 1, \ldots, N$ **do**, x is the particle, d_i is the i-th dimension of x;
3: **if** $d_i > \lambda$ **then**
4: $\mathcal{S} \leftarrow \mathcal{S} \cup \{i\}$;
5: **end if**
6: **end for**

It is worth noting that since training a classifier model with large amount of data is usually highly time-consuming, fitness evaluations in a high-dimensional feature selection is computationally very expensive. As we found empirically that many particles may have the similar positions, it is possible to avoid some computationally expensive fitness evaluations that will result in the same selected feature subset. For this purpose, an archive is designed to record the historical fitness values of all previous selected feature subset such that we can check if a certain feature selection result has already been evaluated before performing real fitness evaluation for it.

1. Lookup the *archive* $\mathcal{H}$ if the current selected feature has been evaluated;
2. If it has been evaluated, extract the fitness value and assign it to the particle;
3. If it has not been evaluated, build the classifier model with selected features, and test its error rate. Assign the fitness, i.e., the average error rate, to the particle and add the selection and fitness into $\mathcal{H}$.

Our empirical tests show that this simple strategy has significantly reduced the time consumption in the search, especially when the swarm is converging and many particles have a similar position.

The proposed CSO based feature selection method together with the above two strategies is summarized in Algorithm 3, where $\mathcal{H}$ denotes the archive that records the fitness values of all the particles in history.

4 Experimental Studies

4.1 Experimental Settings

To assess the performance of proposed algorithm, we conducted a set of experiments on several datasets from UCI Machine Learning Repository [3]. The properties of datasets are listed in Table 1.

For each dataset, we use 70% samples in the dataset as training data, and the rest for testing. The selection of training and test sets is randomized, while the original ratio of class distribution is preserved in both sets.

Algorithm 3 The proposed algorithm

$t \leftarrow 0, \mathcal{H} \leftarrow \varnothing$
for all particle $p_i^t = \langle x_i^t, v_i^t \rangle$ in swarm P^t **do**
    initialize position x_i^t and velocity v_i^t
end for
while termination criteria not met **do**
    for all particle p_i^t **do**
        $\mathcal{S}_i^t \leftarrow \varnothing$
        for all $d_a^{i,t} \in x_i^t, a = 1 \dots N$ **do**
            if $d_a^{i,t} > \lambda$ **then**
                $\mathcal{S}_i^t \leftarrow \mathcal{S}_i^t \cup a$
            end if
        end for
        if $\langle \mathcal{S}_i^t, * \rangle \in \mathcal{H}$ **then**
            extract $f(\mathcal{S}_i^t)$ from $\mathcal{H}$
        else
            calculate fitness $f(\mathcal{S}_i^t)$ by n-fold cross validation
            $\mathcal{H} \leftarrow \mathcal{H} \cup \{\langle \mathcal{S}_i^t, f(\mathcal{S}_i^t) \rangle\}$
        end if
    end for
    $P^{t+1} \leftarrow \varnothing$
    while $P^t \neq \varnothing$ **do**
        randomly chose two different particles p_{r1}^t and p_{r2}^t from P^t
        if $f(x_{r1}^t)$ is better than $f(x_{r2}^t)$ **then**
            $p_w^t \leftarrow p_{r1}^t$, $p_l^t \leftarrow p_{r2}^t$
        else
            $p_w^t \leftarrow p_{r2}^t$, $p_l^t \leftarrow p_{r1}^t$
        end if
        $v_l^{t+1} = R_1^t v_l^t + R_2^t (x_w^t - x_l^t) + \phi R_3^t (\bar{x}^t - x_l^t)$
        $x_l^{t+1} = x_l^t + v_l^{t+1}$
        $P^{t+1} \leftarrow P^{t+1} \cup \{p_w^t, p_l^{t+1}\}$
        $P^t \leftarrow P^t \setminus \{p_{r1}^t, p_{r2}^t\}$
    end while
    $t \leftarrow t + 1$
end while

Table 1: Dataset characteristics

Dataset	Features	Size	Class
movement	90	360	15
musk	167	6598	2
arrhythmia	279	452	16
madelon	500	2600	2
isolet5	617	1559	26
InterAd	1588	3279	2

The algorithms are implemented on Java SE 8 (revision 1.8.0_45), using Weka [18] data mining library version 3.7.12 for the base classification algorithm.

In order to test the effectiveness of the feature selection, we chose to use only simple classification models, i.e., the k-nearest-neighbour (kNN) classifier [1] with $k = 5$. In order to reduce the risk of overfitting, we use the average

error rates of n-fold cross-validation (with $n = 10$) on training data as the fitness function. PSO, four variants of PSO proposed in Xue's paper for feature selection [41] and the principal component analysis (PCA) are compared with the proposed algorithm, along with classification with all features.

To make comparisons between CSO and PSO-based feature selection, we accordingly modify the PSO algorithm to be suited for feature selection. The modification consists of the conversion from continuous values into discrete values for feature selection and the archive strategy. The swarm size is set to 100 for both algorithms, and the maximal number of generations is set as 200. Note that, as CSO only updates half of the population in each iteration, it only generates a half of new solutions that the PSO-based algorithms generate.

The PSO variants in Xue's paper are denoted as Xue1-kNN, Xue2-kNN, Xue3-kNN and Xue4-kNN, respectively. The major difference between Xue's algorithms is the number of features selected in the initial population, while Xue1-kNN uses the traditional initialization strategy in which about half of the features are selected in each individuals, Xue2-kNN uses small initialization strategy where only about 10% features are selected in each individuals, Xue3-kNN uses large initialization strategy where more than half (we used about 2/3 in the experiment) of the features are selected in each individuals, and Xue4-kNN uses a combined initialization where a major (we used 2/3 in the experiment) of the individuals are initialized with the mall initialization strategy (about 10% features in the experiment), while the rest are initialized with the large initialization strategy (about 2/3 features in the experiment). Another main difference between Xue's algorithms and traditional PSO-based algorithms is that in Xue's algorithm, the threshold parameter λ is set to 0.6 while the traditional algorithm uses 0.5 as the threshold parameter.

Other parameters of the training algorithms are: w in the PSO-kNN is set to 0.7298, and both $c1$ and $c2$ are 1.49618; ϕ in the CSO-kNN is set to 0.1. The particles in all algorithms are randomly initialized between $[0, 1]$ and the threshold parameter $\lambda = 0.5$ is applied on CSO-kNN and PSO-kNN while $\lambda = 0.6$ is applied on Xue's algorithms. The variance covered in PCA-based feature selection (PCA-kNN) is 0.95. To obtain statistical results, each algorithm is run for 30 times independently.

4.2 Results

4.2.1 Error rate

The ultimate target of classification is to improve generalization ability, which means a lower error rate on unseen data. Therefore we firstly examine the average error rate of all compared algorithms. The results are summarized in Table 2. We also adopted the Wilcoxon rank sum test to compare the results obtained by the CSO-kNN algorithm and other compared algorithms at a significance level of 0.05. The result is also listed in Table 2, where symbol '+' denotes the particular algorithm is significantly outperformed by the CSO-

kNN algorithm according to the Wilcoxon rank sum test, while '-' denotes the particular algorithm is significantly better than the CSO-kNN algorithm, and '=' denotes that there is no statistically significant difference between the results obtained by the CSO-kNN algorithm and the particular algorithm.

Table 2: Average error rate

Dataset	CSO-kNN	PSO-kNN	Xue1-kNN	Xue2-kNN	Xue3-kNN	Xue4-kNN	PCA-kNN	SC-kNN
movement	**0.2394**	0.2850^{+}	0.2850^{+}	0.2896^{+}	0.2832^{+}	0.2869^{+}	0.2630^{+}	0.2801
	(± 0.0326)	(± 0.0392)	(± 0.0399)	(± 0.0415)	(± 0.0409)	(± 0.0327)	(± 0.0369)	(± 0.0351)
musk	**0.0011**	0.0034^{+}	0.0033^{+}	$0.0015^{=}$	0.0039^{+}	0.0016^{+}	0.0033^{+}	0.0134
	(± 0.0009)	(± 0.0016)	(± 0.0017)	(± 0.0009)	(± 0.0019)	(± 0.0009)	(± 0.0020)	(± 0.0028)
arrhythmia	**0.3240**	0.4059^{+}	0.4076^{+}	0.3593^{+}	0.4098^{+}	0.3564^{+}	0.4588^{+}	0.4270
	(± 0.0225)	(± 0.0221)	(± 0.0203)	(± 0.0335)	(± 0.0243)	(± 0.0294)	(± 0.0079)	(± 0.0173)
madelon	**0.1572**	0.4111^{+}	0.4066^{+}	0.2704^{+}	0.4168^{+}	0.3736^{+}	0.4829^{+}	0.4371
	(± 0.0374)	(± 0.0189)	(± 0.0210)	(± 0.1006)	(± 0.0230)	(± 0.1027)	(± 0.0161)	(± 0.0124)
isolet5	**0.1491**	0.1875^{+}	0.1855^{+}	0.1917^{+}	0.1922^{+}	0.1954^{+}	0.4385^{+}	0.2140
	(± 0.0124)	(± 0.0118)	(± 0.0141)	(± 0.0157)	(± 0.0172)	(± 0.0149)	(± 0.0184)	(± 0.0134)
InterAd	**0.0289**	0.0404^{+}	0.0406^{+}	0.0403^{+}	0.0411^{+}	0.0401^{+}	0.0721^{+}	0.0452
	(± 0.0044)	(± 0.0071)	(± 0.0058)	(± 0.0055)	(± 0.0056)	(± 0.0050)	(± 0.0054)	(± 0.0057)
win/lose/tie	-	6/0/0	6/0/0	5/0/1	6/0/0	6/0/0	6/0/0	-

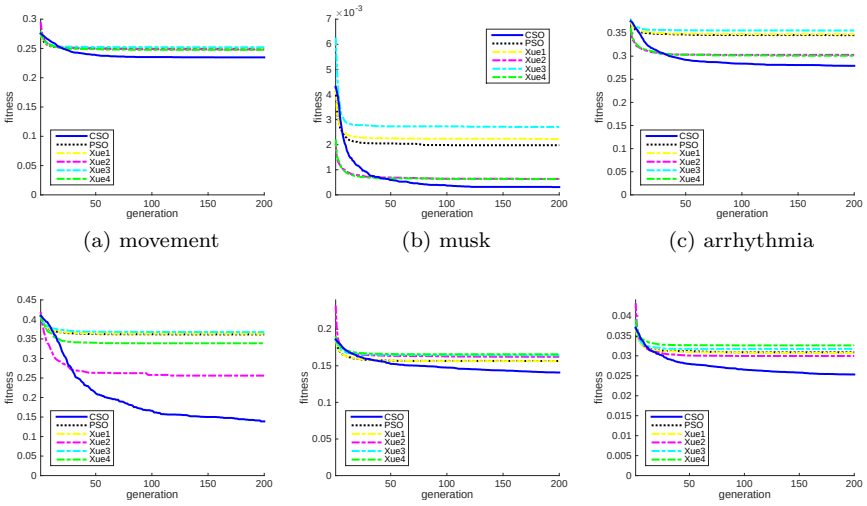
The experimental results show that the CSO-kNN has achieved a statistically lower error rate than all other compared algorithms on all the six datasets. On some datasets, e.g., the *musk* dataset and the *madelon* dataset, the error rate achieved by the CSO-kNN is even smaller than half of that achieved by most other algorithms. Compared with the kNN, the CSO-kNN is always better while all other algorithms have instances on which they performed worse than kNN. From these results, we can also see that PCA-kNN is the least effective algorithm as most results it has obtained are worse than those of kNN. This can be attributed to the fact that PCA is sensitive to noises and outliers, in other words, PCA is less efficient in reducing the accuracy degradation of class-irrelevant attributes.

4.2.2 Performance

In the proposed algorithm, the fitness function is to minimize the average error rate on the training data. The fitness convergence profiles of the algorithms are plotted in Fig. 1. In the figure, X-axes and Y-axes are the generations and the fitness values, respectively.

From Fig. 1, it can be seen that the CSO-kNN is able to always find better solutions, although it may be a bit slower at the early search stage. The PSO-kNN and Xue's algorithms tend to be trapped in a premature convergence around generation 20. The only exception is found on the first dataset, where the difference between the CSO-kNN and compared algorithms is negligible, which might be due to the fact that the dimension of this dataset is relatively low (only 90), on which the difference in search performance between PSO and CSO is minor.

As fitness selection is a combinatorial optimization problem, small changes in the particle positions may not result in a change in the selected features.



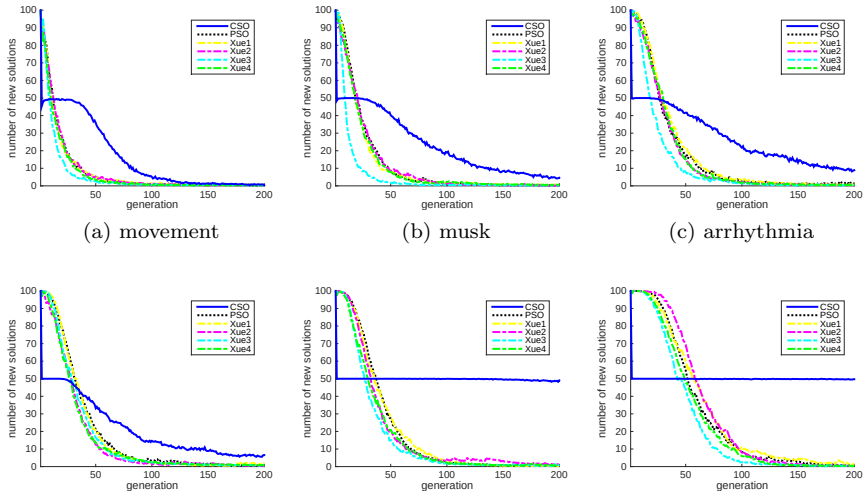
(a) movement (b) musk (c) arrhythmia

(d) madelon (e) isolet5 (f) InterAd

Fig. 1: Average fitness of the best particle respect to each generation.

Thus, it is easy to understand that many new solutions are found at the beginning of the evolutionary optimization. As the evolution proceeds, the number of new solutions that can be found may dramatically reduce. Thus, the number of new solutions found can be used as another indicator for convergence, which are presented in Fig. 2.

The plots in Fig. 2 visually show that the PSO-based algorithms seem to converge faster than the CSO-kNN, since the PSO-based algorithms have found only few solutions at the later stage of evolution. However, a slower convergence rate of CSO-kNN than the PSO variants is expected, which indicates that CSO-kNN has better exploration ability that helps maintain a better swarm diversity, which has also been confirmed by the empirical analyses presented in [10]. Comparing these plots with those in Fig. 1, we can see that the fitness of the PSO-based algorithms improves much more slowly than that of the CSO-kNN. The best fitnesses achieved by the PSO-based algorithms were always worse than those by the CSO-kNN, which strongly indicates that the PSO-based algorithms have got trapped in a local optimum. By contrast, the CSO-kNN can always find better solutions in term of the fitness value. The only exception was on the first dataset, i.e., the dataset *movement*. Both PSO-based algorithms and CSO-kNN have found a small number of new solutions probably because of the low dimensionality of this dataset. The fact that CSO-kNN continues to find new better feature subsets implies that feature selection is naturally a multi-modal optimization problem, i.e., many different combinations of feature subsets may have similar or the same generalization performance.



(a) movement (b) musk (c) arrhythmia

(d) madelon (e) isolet5 (f) InterAd

Fig. 2: Number of new solutions found in each generation.

It also worth mentioning that, apart from the CSO-kNN on the last two datasets, most other experiments have many identical solutions produced during search, meaning that many fitness evaluations may be redundant. As fitness evaluations are very time-consuming for high-dimensional feature selection, it would have taken significantly more time if we do not use the archive mechanism proposed in the proposed algorithm.

4.2.3 Selected features

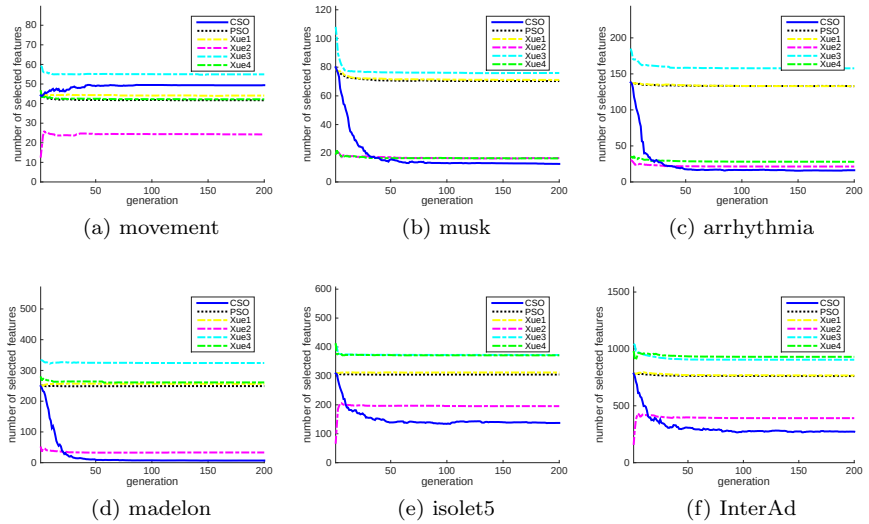
The second target of feature selection is to remove irrelevant features to enhance classification performance. Therefore, we have also taken a look at the average number of selected features obtained the PSO-based algorithms and the CSO-kNN. The results are presented in Table 3.

In this comparison, we can see that CSO-kNN selects statistically fewer features than most compared algorithms, except for Xue2-kNN, which initializes its population with only 10% of the total features. We found that the number of features selected by the PSO-based algorithms are proportional to the number of features initialized at the first generation. In other words, if particles are initialized with a large number of features, the number of features selected in the final population will be larger, and vice versa. Therefore, Xue2-kNN is more likely to outperform CSO-kNN in terms of the number of selected features. Furthermore, as Xue4-kNN has a mixed initialization, depending on the characteristics of the dataset, the best particle may be evolved from those particles initialized with a small or large number of features. As a result, the best solution in the final population selects either much more or

Table 3: Average number of selected features

Dataset	CSO-kNN	PSO-kNN	Xue1-kNN	Xue2-kNN	Xue3-kNN	Xue4-kNN	PCA-kNN	#Features
movement	49.40 (± 6.56)	41.70$^-$ (± 5.15)	44.10$^-$ (± 5.77)	24.27$^-$ (± 8.01)	54.93$^+$ (± 9.01)	42.20$^=$ (± 18.52)	9$^-$	90
musk	12.57 (± 4.83)	70.33$^+$ (± 6.00)	71.13$^+$ (± 6.65)	16.63$^=$ (± 7.79)	75.93$^+$ (± 11.14)	16.43$^+$ (± 6.44)	122$^+$	167
arrhythmia	16.23 (± 6.67)	132.97$^+$ (± 8.84)	133.00$^+$ (± 11.21)	21.40$^=$ (± 12.40)	157.73$^+$ (± 15.68)	27.97$^+$ (± 31.24)	103$^+$	279
madelon	6.90 (± 1.81)	249.07$^+$ (± 12.58)	256.10$^+$ (± 15.34)	33.17$^=$ (± 50.73)	324.17$^+$ (± 30.97)	261.03$^+$ (± 147.09)	426$^+$	500
isolet5	137.40 (± 30.20)	304.73$^+$ (± 11.25)	311.87$^+$ (± 16.85)	195.47$^+$ (± 46.21)	372.37$^+$ (± 31.06)	371.10$^+$ (± 62.56)	182$^+$	617
InterAd	269.97 (± 94.97)	761.40$^+$ (± 21.72)	766.27$^+$ (± 37.90)	391.30$^+$ (± 127.37)	904.97$^+$ (± 88.46)	930.40$^+$ (± 170.62)	300$^=$	1558
win/lose/tie	-	5/1/0	5/1/0	2/1/3	6/0/0	5/0/1	4/1/1	-

less features than CSO-kNN does. By contrast, CSO-kNN is not sensitive to the initialization, which can always find the optimal feature subset regardless the number of features selected during the initialization.



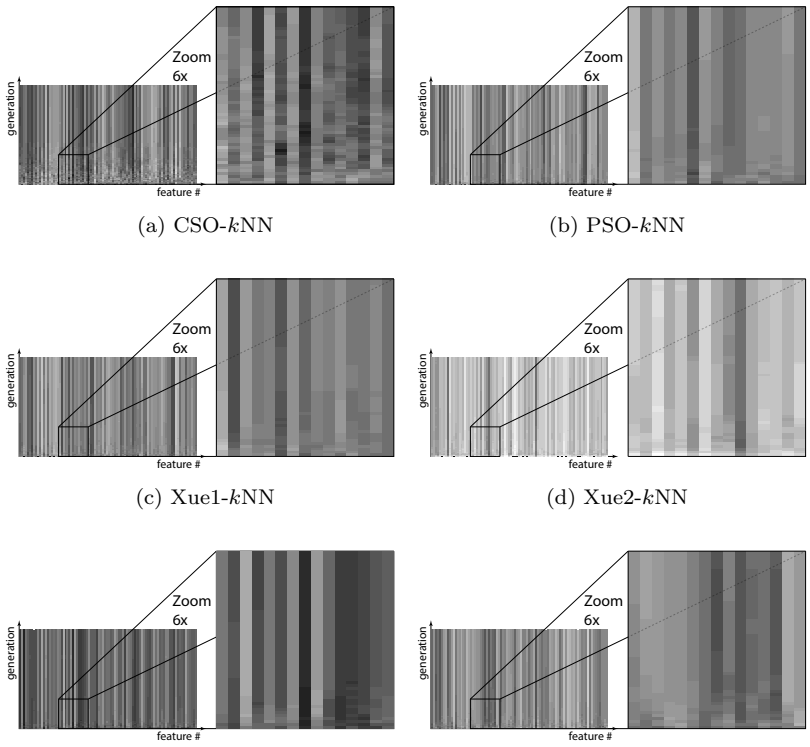
(a) movement (b) musk (c) arrhythmia

(d) madelon (e) isolet5 (f) InterAd

Fig. 3: Average number of selected features over generations.

Further observations regarding the number of features that have been selected during the search procedure by each algorithm can be found in Fig. 3. We can see that CSO-kNN not only finds smaller feature subsets than the PSO-based algorithms on large-scale problems, but the number of selected features also decreases much faster. The only exception is again the Xue2-kNN algorithm due to the same reason that we have discussed above.

From Table 2 and Fig. 3, we can conclude that CSO-kNN performs higher degree of exploration than the PSO-based algorithms, which enables it to explore the search space to find a solution that selects a smaller number of features and better performance.



(a) CSO-kNN (b) PSO-kNN

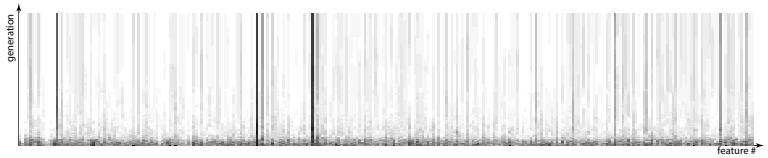
(c) Xue1-kNN (d) Xue2-kNN

(e) Xue3-kNN (f) Xue4-kNN

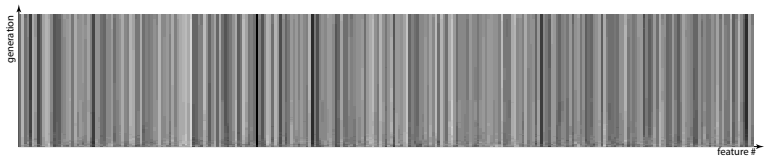
Fig. 4: Graphics showing that the selected features (x-axes) of the best individual over generations (y-axes) on dataset *movement*. The darkness of each block represents the frequency of the corresponding feature that has been selected in the respective generation across different experiments - a darker color represents a higher frequency, and vice versa.

In addition to the number of features that have been selected, it is also of interests to see what features have exactly been selected. To this end, we have also plotted the number of selected features over the generations during the evolution in Fig. 4 and 5.

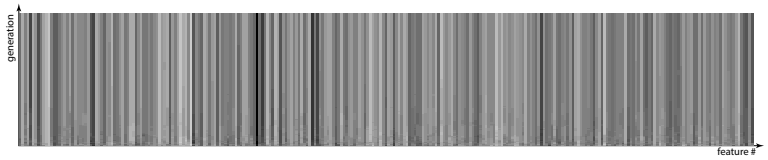
Fig. 4 shows the results on the *movement* dataset, which has the least number of features among the six tested datasets. On this dataset, all algorithms seem to have found a reasonable feature subset that gives the lowest testing



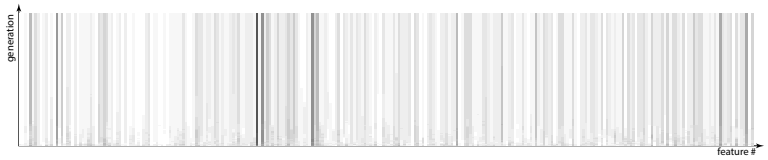
(a) CSO-kNN



(b) PSO-kNN

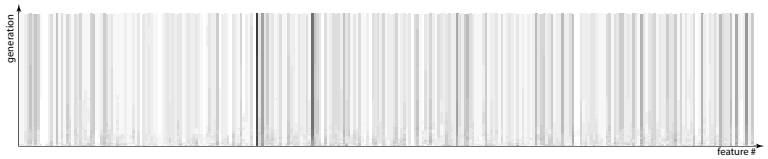


(c) Xue1-kNN



(d) Xue2-kNN

(e) Xue3-kNN



(f) Xue4-kNN

Fig. 5: Graphics showing that the selected features (x-axes) of the best individual over generations (y-axes) on dataset *arrhythmia*.

error possible (refer to Table 2 and Fig 3), and a similar number of features is selected, which is indicated by the similar overall darkness of in the plots. However, there are still visible differences between Xue2-kNN and Xue3-kNN, as the initial numbers of selected features are significantly different. Furthermore, these figures also show that CSO-kNN explores much more potential solutions in comparison to other compared algorithms.

Fig. 5 presents the results on the *arrhythmia* dataset, where the CSO-kNN has selected much fewer features and achieved a considerably lower error rate than the compared algorithms. In general, the darkness of the plot for CSO-kNN is much lighter than that of the PSO-kNN, Xue1-kNN and Xue3-kNN, which implies that the general number of features selected by CSO-kNN is much smaller than the these algorithms in all independent runs. It is also seen that the overall darkness among CSO-kNN, Xue2-kNN and Xue4-kNN are similar. However, by close inspection, we can find that the early populations of CSO-kNN are much darker than the other two algorithms, and during the population evolves, the overall darkness becomes lighter. By contrast, all PSO-based algorithms have a similar overall darkness across the generations. This difference indicates that the CSO-kNN is capable of escaping from local optima due to its better exploration ability than the PSO-based algorithms. Nevertheless, a few clearly discernible vertical dark lines can be seen in the plots, especially in the CSO-kNN, Xue2-kNN and Xue4-kNN. These lines represent the most commonly selected features over all independent runs, while most other lines remain light indicating the scarcely (or even never) selected features. The plots clearly show that the PSO-based algorithms tend to select many other features, which are mostly not helpful, if not harmful to improve the generalization ability.

5 Conclusion

This paper aims to propose an efficient feature selection algorithm to select a small feature subset from a large number of features while maintaining similar or even better classification performance than using all features. The objective has been successfully achieved by adapting the competitive swarm optimization algorithm to feature selection.

Our experimental results demonstrate that the proposed CSO-kNN outperforms the conventional PCA-based method, the PSO-kNN, and a few recently reported PSO variants on all large-scale datasets with a significant margin. Furthermore, we find that the number of new solutions found by CSO-kNN is considerably larger than that found by the PSO-based methods. Unlike the PSO-based methods whose final optimal results heavily rely on the initialization in term of the number of selected features, the purposed method performs consistently well and is less sensitive to initialization.

In the future, we will investigate further the selection mechanism of the metaheuristics, for example by introducing the multi-objective approach that

simultaneously maximizes the classification performance and minimizes the number of selected features.

Acknowledgements This work was supported in part by National Natural Science Foundation of China (No. 71533001), the Joint Research Fund for Overseas Chinese, Hong Kong and Macao Scholars of the National Natural Science Foundation of China (No. 61428302) and an EPSRC grant (No. EP/M017869/1).

References

1. Aha, D., Kibler, D., Albert, M.: Instance-based learning algorithms. Machine Learning **6**(1), 37–66 (1991)
2. Almuallim, H., Dietterich, T.G.: Learning boolean concepts in the presence of many irrelevant features. Artificial Intelligence **69**(1), 279–305 (1994)
3. Bache, K., Lichman, M.: UCI machine learning repository (2013). URL `http://archive.ics.uci.edu/ml`
4. Banks, A., Vincent, J., Anyakoha, C.: A review of particle swarm optimization. part ii: hybridisation, combinatorial, multicriteria and constrained optimization, and indicative applications. Natural Computing **7**(1), 109–124 (2008)
5. Bianchi, L., Dorigo, M., Gambardella, L.M., Gutjahr, W.J.: A survey on metaheuristics for stochastic combinatorial optimization. Natural Computing **8**(2), 239–287 (2009)
6. Chandrashekar, G., Sahin, F.: A survey on feature selection methods. Computers & Electrical Engineering **40**(1), 16–28 (2014)
7. Chen, W.N., Zhang, J., Lin, Y., Chen, N., Zhan, Z.H., Chung, H.S.H., Li, Y., Shi, Y.H.: Particle swarm optimization with an aging leader and challengers. IEEE Transactions on Evolutionary Computation **17**(2), 241–258 (2013)
8. Chen, Y., Miao, D., Wang, R.: A rough set approach to feature selection based on ant colony optimization. Pattern Recognition Letters **31**(3), 226–233 (2010)
9. Cheng, R., Jin, Y.: Demonstrator selection in a social learning particle swarm optimizer. In: 2014 IEEE Congress on Evolutionary Computation, pp. 3103–3110. IEEE (2014)
10. Cheng, R., Jin, Y.: A competitive swarm optimizer for large scale optimization. IEEE Transactions on Cybernetics **45**(2), 191–204 (2015)
11. Cheng, R., Jin, Y.: A social learning particle swarm optimization algorithm for scalable optimization. Information Sciences **291**, 43–60 (2015)
12. Chuang, L.Y., Chang, H.W., Tu, C.J., Yang, C.H.: Improved binary pso for feature selection using gene expression data. Computational Biology and Chemistry **32**(1), 29–38 (2008)
13. Chuang, L.Y., Tsai, S.W., Yang, C.H.: Improved binary particle swarm optimization using catfish effect for feature selection. Expert Systems with Applications **38**(10), 12,699–12,707 (2011)
14. Fei, H., Huan, J.: Boosting with structure information in the functional space: An application to graph classification. In: Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 643–652. ACM, New York, NY, USA (2010)
15. Fong, S., Wong, R., Vasilakos, A.V.: Accelerated PSO swarm search feature selection for data stream mining big data. IEEE Transactions on Services Computing **9**(1), 33–45 (2016)
16. Gheyas, I.A., Smith, L.S.: Feature subset selection in large dimensionality domains. Pattern recognition **43**(1), 5–13 (2010)
17. Guyon, I., Elisseeff, A.: An introduction to variable and feature selection. The Journal of Machine Learning Research **3**, 1157–1182 (2003)
18. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H.: The weka data mining software: an update. ACM SIGKDD explorations newsletter **11**(1), 10–18 (2009)

19. Han, K.H., Kim, J.H.: Quantum-inspired evolutionary algorithm for a class of combinatorial optimization. IEEE Transactions on Evolutionary Computation **6**(6), 580–593 (2002)
20. Hu, M., Wu, T.F., Weir, J.D.: An adaptive particle swarm optimization with multiple adaptive methods. IEEE Transactions on Evolutionary Computation **17**(5), 705–720 (2013)
21. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Neural Networks, 1995. Proceedings., IEEE International Conference on IS - SN -, pp. 1942–1948 vol.4 (1995)
22. Kira, K., Rendell, L.A.: A practical approach to feature selection. In: Proceedings of the International Workshop on Machine Learning, pp. 249–256 (1992)
23. Kohavi, R., John, G.H.: Wrappers for feature subset selection. Artificial Intelligence **97**(1), 273–324 (1997)
24. Kwak, N., Choi, C.H.: Input feature selection for classification problems. IEEE Transactions on Neural Networks **13**(1), 143–159 (2002)
25. Li, X., Yao, X.: Cooperatively coevolving particle swarms for large scale optimization. IEEE Transactions on Evolutionary Computation **16**(2), 210–224 (2012)
26. Liang, J.J., Qin, A.K., Suganthan, P.N., Baskar, S.: Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. IEEE Transactions on Evolutionary Computation **10**(3), 281–295 (2006)
27. Liao, J.G., Chin, K.V.: Logistic regression for disease classification using microarray data: model selection in a large p and small n case. Bioinformatics **23**(15), 1945–1951 (2007)
28. Lin, S.W., Chen, S.C.: PSOLDA: A particle swarm optimization approach for enhancing classification accuracy rate of linear discriminant analysis. Applied Soft Computing **9**(3), 1008–1015 (2009)
29. Liu, Z., Jiang, F., Tian, G., Wang, S., Sato, F., Meltzer, S.J., Tan, M.: Sparse logistic regression with Lp penalty for biomarker identification. Statistical Applications in Genetics and Molecular Biology **6**(1) (2007)
30. Neshatian, K., Zhang, M.: Pareto front feature selection: using genetic programming to explore feature space. In: Proceedings of the Annual Conference on Genetic and Evolutionary Computation, pp. 1027–1034. ACM (2009)
31. Price, K., Storn, R.M., Lampinen, J.A.: Differential evolution: a practical approach to global optimization. Springer Science & Business Media (2006)
32. Pudil, P., Novovičová, J., Kittler, J.: Floating search methods in feature selection. Pattern Recognition Letters **15**(11), 1119–1125 (1994)
33. Shi, Y., Eberhart, R.: A modified particle swarm optimizer. In: Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence., The 1998 IEEE International Conference on IS - SN - VO -, pp. 69–73 (1998)
34. Tan, M., Tsang, I.W., Wang, L.: Minimax sparse logistic regression for very high-dimensional feature selection. IEEE Transactions on Neural Networks and Learning Systems **24**(10), 1609–1622 (2013)
35. Tran, B., Xue, B., Zhang, M.: Bare-Bone Particle Swarm Optimisation for Simultaneously Discretising and Selecting Features for High-Dimensional Classification. In: G. Squillero, P. Burelli (eds.) Applications of Evolutionary Computation: 19th European Conference, EvoApplications 2016, Porto, Portugal, March 30 – April 1, 2016, Proceedings, Part I, pp. 701–718. Springer International Publishing (2016)
36. Unler, A., Murat, A.: A discrete particle swarm optimization method for feature selection in binary classification problems. European Journal of Operational Research **206**(3), 528–539 (2010)
37. Wang, H., Sun, H., Li, C., Rahnamayan, S., Pan, J.S.: Diversity enhanced particle swarm optimization with neighborhood search. Information Sciences **223**, 119–135 (2013)
38. Wang, X., Yang, J., Teng, X., Xia, W., Jensen, R.: Feature selection based on rough sets and particle swarm optimization. Pattern Recognition Letters **28**(4), 459–471 (2007)
39. Whitney, A.W.: A direct method of nonparametric measurement selection. IEEE Transactions on Computers **C-20**(9), 1100–1103 (1971)
40. Xue, B., Zhang, M., Browne, W., Yao, X.: A survey on evolutionary computation approaches to feature selection. IEEE Transactions on Evolutionary Computation **PP**(99), 1–1 (2016)

41. Xue, B., Zhang, M., Browne, W.N.: Particle swarm optimization for feature selection in classification: A multi-objective approach. IEEE Transactions on Cybernetics **43**(6), 1656–1671 (2013)
42. Xue, B., Zhang, M., Browne, W.N.: Particle swarm optimisation for feature selection in classification: Novel initialisation and updating mechanisms. Applied Soft Computing **18**, 261–276 (2014)
43. Zhai, Y., Ong, Y.S., Tsang, I.W.: The emerging "Big Dimensionality". Computational Intelligence Magazine, IEEE **9**(3), 14–26 (2014)
44. Zhan, Z.H., Zhang, J., Li, Y., Chung, H.S.H.: Adaptive particle swarm optimization. IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics **39**(6), 1362–1381 (2009)
45. Zhu, Z., Ong, Y.S., Dash, M.: Wrapper–filter feature selection algorithm using a memetic framework. IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics **37**(1), 70–76 (2007)