Annals of Mathematics and Artificial Intelligence (2022) 90:893–913
https://doi.org/10.1007/s10472-021-09752-4

# Chunking and cooperation in particle swarm optimization for feature selection

**Malek Sarhani**[1] · **Stefan Voß**[1]

Accepted: 2 May 2021 / Published online: 19 July 2021


**Abstract**
Bio-inspired optimization aims at adapting observed natural behavioral patterns and social phenomena towards efficiently solving complex optimization problems, and is nowadays gaining much attention. However, researchers recently highlighted an inconsistency between the need in the field and the actual trend. Indeed, while nowadays it is important to design innovative contributions, an actual trend in bio-inspired optimization is to re-iterate the existing knowledge in a different form. The aim of this paper is to fill this gap. More precisely, we start first by highlighting new examples for this problem by considering and describing the concepts of chunking and cooperative learning. Second, by considering particle swarm optimization (PSO), we present a novel bridge between these two notions adapted to the problem of feature selection. In the experiments, we investigate the practical importance of our approach while exploring both its strength and limitations. The results indicate that the approach is mainly suitable for large datasets, and that further research is needed to improve the computational efficiency of the approach and to ensure the independence of the sub-problems defined using chunking.

**Keywords** Particle swarm optimization · Cooperative learning · Chunking · Feature selection · Bio-inspired optimization

## 1 Introduction

Bio-inspired computation has emerged as one of the most studied branches of artificial intelligence during the last decades. Most frequently the aim is to propose simple algorithmic approaches suitable for complex real-world problems. That is, the aim of such approaches is to mimic principles of real life and turn them into simple algorithmic solutions that can be applied to address complex real-world problems [1].

✉ Malek Sarhani
malek.sarhani@uni-hamburg.de

Stefan Voß
stefan.voss@uni-hamburg.de

[1] Institute of Information Systems, University of Hamburg, Hamburg, Germany

In particular, over the last few decades, optimization has been strongly affected by this concept. The aim of the proposed approaches, which fall under the umbrella of bio-inspired (or nature-inspired) optimization, is to mimic the behavior and internal functioning of physical, biological and social systems in order to design enhanced optimization algorithms with potential value in real-world applications [2]. In the literature, the aforementioned approaches are generally referred to as a special kind of metaheuristics, most often population-based [3]. Such methods are in many cases based either on evolutionary computation or swarm intelligence [4]. Nowadays, such ideas have shown promise in different optimization problems, in which the computational complexity of exact algorithms grows exponentially with the problem dimension [5], and the performance of classical approximate algorithms (e.g. greedy methods) deteriorates significantly with the size of the search space [6]. In fact, in the age of big data, the need to solve such problems with a large number of decision variables is becoming increasingly crucial.

Hence, this growing attention has resulted in a considerable rise in the number of publications. However, this rapid increase in these has led to various methods which are similar or even identical, although presented in another form and under a different name. Readers are referred to [7] for more details and clarifications on these issues that challenge research in the field of metaheuristics, and to [8–10] for examples of such approaches. Moreover, as highlighted in the next section, another related problem is that some papers extend or apply some approaches without cross-referencing them. Therefore, instead of re-iterating the existing knowledge in a different form, a line of research that could make a real contribution is to properly and newly combine ideas already approved and to propose a new design suited to a specific optimization problem. In this paper, we first strengthen awareness of the problem of re-iterating the knowledge by taking two bio-inspired concepts that could be used to solve complex optimization problems, which are chunking and cooperative learning. The former, which is inspired by human behavior, consists in grouping basic information units allowing to build a higher level solution. Regarding the latter, it can be noted here that swarm intelligence approaches are by principle based on cooperation, and that most bio-inspired algorithms belong to the category of swarm intelligence [2], even if some of them were criticized for instance in [7]. In fact, while competition among individuals usually improves their performance, much greater improvement can be achieved through cooperation, as it is nowadays quasi-admitted in both real and artificial life [11].

More precisely, our contribution in this paper is first to give some examples, with regard to two concepts, not provided in the preceding papers (e.g. [8]). Second, we propose a new design combining the two approaches, which adopt them in a complementary way. Chunking aims to decompose a high-dimensional problem into independent sub-problems that could be optimized in a better way than the initial problem, while cooperative learning aims to share the solution's sub-components (sub-problems) or to concatenate them in the most efficient way. In particular, in this paper we focus on particle swarm optimization (PSO) as a typical and one of the best known bio-inspired optimization approaches. In fact, PSO is a swarm intelligence optimization algorithm inspired by the intelligent collective behavior of animals (e.g. birds and fish), that have been designed for continuous optimization problems [12]. Moreover, it has also been applied and adapted to combinatorial optimization problems [13].

In this work, after reviewing the state of the art of the two concepts, we aim to project them into the case of PSO and to propose a new design that matches the problem of feature selection (FS). FS is a combinatorial optimization problem arisen in the field of machine learning (ML). The aim of FS is to improve the generalization ability of learning algorithms

by removing redundant, irrelevant and noisy features. FS is a combinatorial optimization problem with $2^n$ possible combinations, where $n$ is the number of original features. Therefore, the search space grows exponentially with the number of features, and thus FS is still a problematic issue, especially on high-dimensional data due to its huge search space.

There are three main approaches for FS [14]. Filters evaluate feature subsets based on some predefined metrics, most often using information theory (e.g. entropy, mutual information (MI)). An example of a classical filter approach is correlation-based feature selection [15]. More information on filter approaches could be found for instance in [16]. Wrapper and embedded approaches both utilize a ML algorithm to score subsets of features according to their predictive capability. The main difference is that, in embedded approaches, the selection is done in the training process while in wrappers the ML classifier is used as a black box. Therefore, optimization algorithms such as PSO have been more frequently adopted in wrappers than in embedded approaches [17]. Regarding the comparison of wrappers and filters, in general, the former can promise better results than the latter. But, it is more computationally demanding [14]. Thus, in this paper, we aim to take advantage of both approaches as follows: the problem chunking is based on a filter approach and the optimization of each sub-problem (chunk) is optimized with a wrapper approach.[1]

The rest of the paper is organized as follows: In the next section, we present a literature review while highlighting the problem of knowledge re-iteration. In section 3, we expose our design of combining chunking and cooperative learning for FS. Section 4 shows the experiments. Finally, a conclusion is presented.

## 2 Literature review

In this review, we focus on three issues. First, we present the current work on the use of chunking in optimization, while exposing the problem of re-iterating knowledge in a different form. Next, while highlighting the same problem, we describe the well-known cooperative learning approaches that have been adopted in PSO in particular and swarm intelligence in general. Finally, we project these ideas onto the FS problem.

### 2.1 Chunking

The first bio-inspired concept considered in this paper is chunking. Its purpose is to limit the cost of computation, and to divide the problem into sub-problems easier to be solved than the initial problem. It was first introduced in psychology [19] and then extended into ML [20, 21]. To the best of our knowledge, its first introduction in optimization was in [22]. There were few attempts to apply this work (e.g. [23] and [24]). Other works interested on using this idea can be found within evolutionary computation. For instance, [25] integrated this idea into a variable-length genetic algorithm (GA) (though ignoring the earlier work of [23]). The basic idea of this GA variant is to use an increasing length of chromosomes in order to solve progressively the optimization problem through solving some smaller sub-problems. This idea was also incorporated into PSO [26].

The concept of chunking was also used for other problems: [27] adopted it for the problem of optimal control, and [28] for redundancy elimination of network packets.

[1] An earlier version of this paper was presented at LION 2020 [18].

We see that, although the idea of dividing the problem into sub-problems is widely embraced in optimization, few of the works directly adopted the concept and the term of chunking in optimization. Moreover, there is a lack on the connection and cross-referencing of some of them. That is, works interested in evolutionary optimization (e.g. [25]) often do not cite previous works on local search (e.g. [23]). However, we can see that other related concepts could be based on a similar idea. An example of such a concept that was introduced in PSO is clustering because it has the same objective of grouping the solution attributes. One of the best-known articles in this respect is [29]; this paper investigates a clustering PSO approach for dynamic optimization problems. Their approach consists also of generating multi-swarms using a single-linkage hierarchical clustering. Here, we adopt the term chunking as being associated and more suited to learning and bio-inspired behaviors.

In addition, other optimization approaches are also inspired by similar ideas. For example, this concept is incorporated into the newly proposed metaheuristic fixed set search [30]. Also, the referenced paper highlighted a number of other methods, known as matheuristics [31], which use mathematical programming to decompose the problem into independent sub-problems, e.g. POPMUSIC [32]. There are also other methods which have leveraged the divide-and-conquer paradigm [33].

We can conclude in this part that the concept of chunking has been introduced in different forms and that there is a lack of comprehensive literature on this. In other words, we can notice that there is a number of papers which re-introduced the concept of chunking under a different name while few attempts have been made to improve the idea initially proposed. Therefore, it is necessary to unify these different works to take advantage of different ideas. In this paper, we aim to use this idea as a basis for including cooperative learning in PSO.

### 2.2 Cooperative learning

In this part, it should first be mentioned that the fundamental principle of PSO is based on learning through cooperation and sharing of information in such a manner that each particle participates in the evolution of the population and in the improvement of the solution. However, in the canonical version, all the particles follow the best at each iteration, then the algorithm could be easily trapped in a local optimum. Therefore, several learning approaches, which have a more decentralized sharing of information, have been proposed such as comprehensive learning [34]. We refer to [35] for other examples of learning approaches that were adopted in PSO. However, as further illustrated in [36], in these learning approaches, as for the canonical PSO, it is possible that some vector components have moved closer to the optimal solution, while others have moved away from it. So, as long as the effect of the enhancement outweighs the impact of the weakening components, these PSO variants would consider the new vector as an overall enhancement, although some vector components may have shifted further from the solution. Therefore, another form of learning was introduced in [36] under the name of cooperative PSO.

We can notice that a main difference between most of the PSO variants (including the canonical) highlighted in the past paragraph is that in the former each particle forms a complete solution while in the latter, a swarm aims to optimize a part of the solution that could be complemented by the other swarms (we note here that a swarm could represent a single variable as shown later). However, in the literature, the term cooperative PSO has been adopted for both cases. Regarding the first, [37] introduced the concept of multi-swarm optimization, which incorporates interaction mechanisms between the different swarms through a "master–slave" model. The same design has also been included in the cooperative multi-swarm PSO proposed in [38]. In addition, [39] defined a cooperative PSO, with a different

design: two surrogate assisted PSO variants (classical and social learning) cooperate to find the final (or a best) solution. The term surrogate refers to a function that approximates an objective function for time-consuming reasons, and in the paper, two PSO variants are sharing solutions, in which classical PSO [12] assisted by a fitness estimation function is used for exploration and a surrogate-assisted social-learning PSO [40] is used for exploitation. With respect to the second, some cooperative PSO extensions have been proposed, mainly to divide large-scale problems (e.g. [41]). In this paper, we are interested in the second type which has shown promise and is associated with chunking. Below, we reveal the origin of this concept which has been used in swarm intelligence in general and adapted to combinatorial optimization.

The basic concept of using cooperative learning within swarm intelligence, as designed in [36], is to build a family of swarms, each trying to optimize a component of the solution, and cooperating to find the final solution. It was initially designed for ant colony systems [42]. In this approach, the ants cooperate using an indirect form of communication mediated by pheromones which they deposit on the edges of the traveling salesman problem (TSP) graph while constructing the solutions. This work was extended in [43]; the authors of this paper tried to use a more diverse and dynamic form of agent communication through adaptive memory programming (AMP) processes, and applied the method to the same problem (TSP). The idea was adapted to PSO in [36]; their approach consists in using several swarms to optimize the various components of the vector solution cooperatively.

We can see from the literature that most of the adopted approaches are tailored to continuous optimization. In contrast, less attention has been paid to understanding how learning and cooperative approaches can be adapted to combinatorial optimization. An example of such approaches is given in [44]. In this paper, a discrete adaptation of the cooperative PSO was used to solve the problem of field programmable gate array placement. Also, other swarm intelligence approaches were adopted for combinatorial optimization. An example of this is the Multi-leader Migrating Birds Optimisation [45] (even if the term cooperation was not used directly). In fact, the approach was successfully applied for a combinatorial optimization problem using a cooperative approach [46]. Multi-swarm optimization [47] is another metaheuristic which is based on the idea of introducing cooperative learning into PSO. We can then observe that cooperative learning is another concept that has been adopted and reiterated in different forms. Moreover, the same term has been adopted in PSO to reflect two different designs, as illustrated above.

In this paper, we have chosen FS as a combinatorial optimization problem to be examined. In the next section, we shortly describe the problem of FS, and highlight some related papers which are interested in FS. It should be noted that in our previous paper [18], we have defined the design of combining the two approaches. In this paper, as an extension of the above, we present a more complete version with a more extensive literature review and experiment. Therefore, [18] is not included in the review below.

### 2.3 Feature selection

Nowadays, FS has become an essential technique in data pre-processing especially on high-dimensional data in different ML applications [48]. Typical feature subset selection methods are sequential forward selection and sequential backward selection. However, such greedy approaches are prone to be stuck in local optima, especially now in the big data era. A global search technique is then needed to explore huge search spaces more efficiently, and therefore bio-inspired optimization has gained much attention in solving this problem. In particular, concerning the use of PSO for FS, it has been applied for FS in different forms;

the most used form is the canonical binary PSO proposed by [13] as presented in the following section. This approach has been applied, for instance, in [49, 50].

With regard to the concepts outlined above, to the best of our knowledge, the unique paper which used such kind of cooperation in swarm intelligence for FS is [51]. Their cooperative design adopted the first ant colony system to detect optimal cardinality features and the second to select the most relevant based on this information. However, the two problems are dependent, and cannot be optimized separately.

When dealing with the problem of FS, we can notice that most of the papers that are interested in the concept of chunking define it under the term of clustering. For instance, in [52], the features are divided into clusters by using a graph-theoretic clustering method. Their idea is to form a set of clusters of independent features. Another approach is presented in [53] in which features are divided into several clusters using a community detection algorithm after presenting them in a form of a graph. Also, other related concepts could be applied to FS. [26] proposed a variable-length PSO method for FS, where particles in a swarm can have different lengths which can also be modified during the evolutionary process. Another approach is proposed in [54] for FS with a high-dimensional dataset in which the hamming distance is introduced as a proximity measure to update the velocity of particles.

## 3 The proposed approach

In this section, we start by showing how PSO could be applied to the problem of FS, and then expressing the fitness function we have adopted in this paper. After that, we expose how we have incorporated both the concepts of chunking and cooperative learning to extend it.

### 3.1 Particle swarm optimization for feature selection

PSO was initially designed for continuous optimization problems, but was extended in several binary and discrete forms to deal with combinatorial optimization problems such as FS. In this paper, we adopt the traditional and most known discrete and binary version which was proposed in [13]. In this binary PSO form, the update of the population of particles is done according to 1, 2 and 3, concerning the inertia weight, and it is updated as in (4) (more details can be found in [13]).

$$v_i^j(t+1) = w v_i^j(t) + c_1 r_1 (p_i^j(t) - x_i^j(t)) + c_2 r_2 . (p_g^j(t) - x_i^j(t)) \tag{1}$$

where $v_i^j(t)$ and $x_i^j(t)$ correspond to the $j^{th}$ dimension of velocity and position vectors of the particle $i$. The position $p_i^j(t)$ represents the $j^{th}$ dimension of the best previous position of particle $i$, while $p_g^j(t)$ represents the $j^{th}$ dimension of the best position among all particles in the population. The variables $r_1$ and $r_2$ are two independently uniformly distributed random variables, whereas the parameters $c_1$ and $c_2$ are the acceleration factors and $w$ is the inertia weight.

$$v_i^j(t+1) = sig(v_i^j(t+1)) = \frac{1}{1 + e^{-v_i^j(t+1)}} \tag{2}$$

$$x_i^j(t+1) = \begin{cases} 1 & \text{if } \text{rand}_i \leq sig(v_i^j(t+1)) \\ 0 & else \end{cases} \tag{3}$$

where $sig$ is the sigmoid function used to transform continuous values into binary ones, and $rand_i$ is a uniform random variable in the interval [0,1].

$$w = w_{max} - \frac{t}{t_{max}}(w_{max} - w_{min}) \tag{4}$$

$t$ and $t_{max}$ are, respectively, the current iteration and the maximum number of iterations.

PSO could be integrated with a wrapper approach as in Algorithm 1 [55]. As can be seen in Algorithm 1, the proposed approach consists in evaluating a fitness function at each iteration $fit(t)$. In the next part, we reveal it.

### 3.2 Fitness function

In this part, we first emphasize that the FS optimization problem has been formulated in different ways. The reason is that the optimization algorithm, in many cases, should not only maximize the prediction accuracy (or equivalently minimize the prediction error), but also minimize the number of features [50]. Indeed, in addition to the computational advantage of minimizing the number of features, it enables to improve the generalization capacity of the ML algorithm (i.e. the performance on the test set). In fact, as we have shown in our previous work [18], when relying only on the prediction accuracy to choose the best optimization algorithm, its performance differs with respect to training and test sets. That is, an optimization approach could give better results in the training set and worse in the test set. Such a situation is not desired, as the model should perform well also on unseen datasets. Therefore, some fitness functions have been proposed to balance prediction accuracy and generalization capacity.

The trade-off between maximizing the prediction accuracy and minimizing the number of features can be done in different manners. In this paper, as in [56], we adopt an aggregate fitness function as described in (5):

$$F = \alpha F_1 + (1 - \alpha) F_2 \tag{5}$$

$F_1$ is the error rate. $F_2$ is the percentage of the selected features as described in (6):

$$F_2 = \frac{p}{n} \tag{6}$$

where $p$ is the number of selected features and $n$ is the complete number of features.

We note that we have adapted the fitness function proposed in [56], as to define the problem as a minimization function. Nevertheless, the same spirit is adopted to aggregate the two objectives.

The value of $\alpha$ is set to 0.8 because the error rate has to be considered more than the number of features. Regarding this issue, Such a value has been recommended in the literature (e.g. [57] which proposed to adjust the value of $\alpha$ between 0.7 and 0.9).

On the other hand, other fitness functions have been proposed in [57, 58] with the same purpose of balancing between the two objectives using similar or related aggregation functions.

### 3.3 Feature selection by combining chunking and cooperative learning

In this part, our objective is to bridge the gap between the different concepts highlighted in the previous section. That is, we propose a design adapted to the FS problem which adopts cooperative learning in PSO using chunking/clustering.

In the conventional PSO presented in the previous section, each individual in the swarm represents a complete solution vector. But, in our proposed approach, instead of having one swarm (of $n$ particles) trying to find the optimal $d$-dimensional vector, the vector is split into its clusters of features that we can consider independent of the others. In other words, a solution vector of the selected features is a combination of the different solutions provided by each swarm according to the same principle of the cooperative PSO [36]. However, a challenging issue is to assign each particle to its appropriate swarm in order to obtain swarms that optimize independent sub-problems. In this paper, we associate with each swarm a cluster of features that their F-correlation with the remaining features could be neglected, and could then be considered independent of the others. For this, we adopt the idea defined in [52] which we summarize in the next paragraph.

In [52], after computing the F-correlation for each pair of features, the feature set is considered as a vertex set (graph), where each feature is considered as a vertex and the weight of the edge between vertices $f_i$ and $f_j$ is their F-correlation, the concept of a minimum spanning tree (MST) was adopted for grouping features, as it does not suppose that the points are grouped around the centers. More concretely, after having connected all the vertices so that the sum of the weights of the edges is minimal, the clusters are formed by removing the edges whose weights are smaller than the T-Relevance of both features with the class. We must note that we are based here on the consideration of [52] that the highly correlated features are assembled in a cluster and that is why we defined our assumption of independence of different clusters.

We note that our approach is a hybridization of filter and wrapper approaches, in which a correlation-based filter approach [15] is used for clustering and a PSO-based wrapper approach is used to select the features of each cluster. Indeed, as noted above, each of these FS approaches has its advantages (wrappers are generally more accurate and filters are the fastest), and our motivation of defining our approach is to take advantage of both.

In Algorithm 2, we define the different steps of the proposed approach. Algorithm 2 returns the subset of features found by the PSO search and the associated fitness function. After the initialization of the particle swarms, which is done as in the conventional PSO [13], it starts with the clustering of the features using F-correlation (as described above) followed by the main PSO wrapper loop. In other words, PSO is executed $d$ times, where $d$ is the number of chunks determined via the chunking/clustering process. The next step in the algorithm is to concatenate the different sub-solutions found by each PSO, and then to evaluate the fitness function and choose the corresponding subset.

As we can see above, the approach is simple and consists mainly of combining the approaches explored before. In fact, unless it is necessary, it is generally recommended to define the approaches in the simplest form. Our proposed approach is based on two concepts chunking/clustering and cooperative learning, and our contribution is to newly combine them to better address the problem of FS, and to fairly evaluate our approach in order to see to which extent we can move forward in this direction.

**Algorithm 1:** FS using PSO.

**Data**: Dataset $(X, Y) = (x_1, y_1), ..., (x_m, y_m)$ , $c_1, c_2$ (PSO parameters), $n$ population size
**Initialization:** Initialize the vector positions $(p_1, ..., p_n$ where $p_i = p_i^1, ..., p_i^d \ \forall i = 1, ..., n)$ and velocities $(v_1, ..., v_n$ where $v_i = v_i^1, ..., v_i^d \ \forall i = 1, ..., n)$ of particles, the fitness $fit$ , $pbest$ and $gbest$ ;
Initialize the scalars inertia weight ( $w$ ) and $l$ (index of the selected feature $f_l$);
Set $t$ value to 1 ;
**while** *number of iterations* $t \leq t_{max}$ **do**
  **for** $i$ = *1 to* $n$ **do**
    **for** $j$ = *1 to* $d$ **do**
      Compute the velocity $v_i^j(t)$ according to the Eq. (2) ;
      Compute the position $p_i^j(t)$ according to the Eq. (3) ;
    **end**
    Evaluate the fitness $fit_i(t)$ of the particle i ;
    **if** $(fit_i(t) \leq fit(pbest))$ **then**
      $(pbest \leftarrow p_i(t))$ & $(fit(pbest) \leftarrow fit_i(t))$
    **end**
  **end**
  Update the inertia weight $w$ according to the Eq. (4) ;
  $t \longrightarrow t + 1$
**end**
Return $gbest$ ;
**for** $k$ = *1 to* $d$ **do**
  **if** $(gbest^k = 1)$ **then**
    $(f_l \leftarrow k)$ ;
    $l \leftarrow l + 1$
  **end**
**end**
**for** $i$ = *1 to* $m$ **do**
  $x_i' = (x_i^{f_1}, ..., x_i^{f_{d^*}})$;
**end**
**Result**: A data subset of $(x_1', y_1), ..., (x_m', y_m)$ of d* features $(f_1, ..., f^{d^*})$ (where $x_i' = x_i^{f_1}, ..., x_i^{f_{d^*}}$)

**Algorithm 2:** Cooperative PSO with chunking for FS.

**Data**: Same of Algorithm 1
**Initialization:** Positions and velocities of particles and their parameters;
Set $k$ value to 1 ;
Generate $d$ cluster of features by determining a minimum spanning tree [52] ;
**for** $k$ = *1 to* $d$ **do**
  Run Algorithm 1 ;
**end**
Concatenate the selected features to obtain the final solution ;
Evaluate the fitness function ;
**Result**: The chosen subset based on the best solution in the dataset

## 4 Experiments

In this section, we first describe the parameter setting of the experiments, then we compare and discuss the results.

### 4.1 Experimental setup

In our experiments, the concept of support vector machines (SVM) [59] is chosen as it has shown good performance in ML classification tasks. In other words, SVM is adopted in this paper in order to measure the predictive power of the selected features. To implement PSO for FS, we have adopted the "PySwarms" research toolkit [60] and the Scikit-learn package. Concerning PSO parameters, we consider the acceleration coefficients $c_1 = 0.5$ and $c_2 = 0.5$. We have chosen these values, which are the default parameters provided in "PySwarms", as they enable a balance between exploration and exploitation; more details on the impact of these parameters can be found in [35]. The number of iterations is set to 100. We choose this value (not a higher value) as the proposed method must find a good solution within a reasonable time to challenge some typical FS methods.

To evaluate our approach, we use twelve widely adopted datasets for comparing FS approaches. These datasets represent different domains (e.g. health sciences, pattern recognition) and differ widely in the number of features, the number of samples and the number of classes (three of them are multi-classification problems). We note that in this experiment, facing the FS problem, we are more interested in the difference with respect to the number of features. Most of these datasets are available in the UCI ML repository and some of them are used in NIPS 2003 FS challenge. Details are provided in Table 1. [2]

We compare our approach with the PSO-based wrapper approach (with the same design adopted in [55]) and with two classical filter FS approaches. The considered fitness function is depicted in (5). Also, we compare the corresponding prediction accuracy, number of selected parameters and the CPU time (in seconds). The algorithms are executed on a computer equipped with an Intel i7-9750H and 16GB of RAM.

We have limited our comparison to the basic PSO-based approach, not including other metaheuristics or bio-inspired approaches, as our aim is to investigate and study the impact of chunking and cooperative learning on it, and if they improve its performance. Also, we compare with a $\chi^2$-based FS approach [61] and a MI-based approach [62], which are typical methods for performing FS with the Scikit-learn package. The comparison with these approaches is included in order to see to which extent our method (and bio-inspired optimization in general) could be adopted as an alternative to classical FS methods. We note here that the first approach (PSO) is a wrapper approach while the latter ($\chi^2$ and MI) are filter ones. We should also note that the $\chi^2$ and MI approaches, as implemented in Scikit-learn, require manual selection of the number of features to maximize their performance, and we choose moderate values for both. Regarding the approach defined in [52], it is not easy to compare it as it was implemented in the "Weka" software and most of the adopted datasets require a pre-processing step (e.g. transformation of categorical features and classes in numerical values), which is not developed in "Weka" but in Python.

We note at the end of this section that we have considered in this experiment that the number of chunks is 2. In fact, our aim is to see the impact of problem chunking, which

[2] For some of the datasets, we have adopted a subset of the data used in the challenge with the values shown in Table 1.

**Table 1** Benchmark dataset

| | Data name | Features | Samples | Target |
|---|---|---|---|---|
| (a) | Zoo | 16 | 101 | 7 classes |
| (b) | Breast cancer | 30 | 569 | 2 classes |
| (c) | Ionosphere | 34 | 351 | 2 classes |
| (d) | Sensorless Drive Diagnosis | 49 | 5851 | 11 classes |
| (e) | Lung Cancer | 56 | 32 | 2 classes |
| (f) | Sonar | 60 | 208 | 2 classes |
| (g) | Digits | 60 | 1797 | 10 classes |
| (h) | Madelon | 100 | 4400 | 2 classes |
| (i) | Arcene | 200 | 500 | 2 classes |
| (j) | Dorothea | 204 | 500 | 2 classes |
| (k) | Semeion | 256 | 1593 | 2 classes |
| (l) | Voice Rehabilitation (LSVT) | 309 | 126 | 2 classes |
| (m) | Human Activity Recognition (HAR) | 561 | 1020 | 2 classes |
| (n) | Speech | 756 | 754 | 2 classes |
| (o) | Gisette | 3783 | 1000 | 6 classes |
| (p) | Leukemia | 7128 | 72 | 2 classes |

could be done with this before further examining on a higher number. In fact, as the chunking process could be computationally demanding, it is preferable to examine it first in the simplest form before exploring a higher chunking dimension, to better analyze and manage the CPU time of the approach.

### 4.2 Comparison of the results

In this part, for each of the four approaches (namely our approach, PSO, $\chi^2$ and MI), we display in Table 2 the results obtained for the fitness function presented in Section 3.2.

Also, in Table 3, we show the prediction accuracy along with the number of selected features (in parenthesis) for each one of the four FS approaches.

We can see from Tables 2 and 3 that the approaches give different results, which vary depending on each dataset. Next, we provide a graphical comparison of both approaches to enable a better analysis of the results.

In fact, to further compare the performance of our approach with the conventional PSO, and to analyze the convergence of the two approaches, we depict in Fig. 1 the evolution of these results during the optimization process. Here we note that plotting the solution progress at each iteration for the native PSO is straightforward, and is implemented in the PySwarms package; To implement it in our approach, we extract the corresponding position of the global best at each iteration (referred in (1)) for the two chunks, and then concatenate them to determine the corresponding solution. The purpose of this unprecedented analysis in [52] and other papers is to track the contribution of chunking at each iteration and its impact on the convergence of the algorithm. Hence, the evolution of the results for both algorithms during the optimization process is displayed in Fig. 1 for the 16 datasets.

**Table 2** Comparison of the obtained values for the fitness function for each algorithm

| Data | Our approach | PSO approach | $\chi^2$ approach | MI approach |
|---|---|---|---|---|
| (a) | 0.210 | **0.074** | 0.130 | 0.114 |
| (b) | 0.077 | **0.028** | 0.053 | 0.049 |
| (c) | **0.078** | 0.083 | 0.140 | 0.135 |
| (d) | 0.601 | 0.689 | **0.515** | 0.616 |
| (e) | 0.21 | **0.096** | 0.186 | 0.179 |
| (f) | 0.237 | 0.221 | **0.203** | 0.206 |
| (g) | 0.047 | 0.053 | 0.036 | **0.035** |
| (h) | **0.432** | 0.456 | 0.446 | 0.450 |
| (i) | **0.335** | 0.388 | 0.416 | 0.416 |
| (j) | **0.092** | 0.111 | **0.092** | **0.092** |
| (k) | 0.350 | 0.324 | **0.298** | 0.346 |
| (l) | **0.072** | 0.076 | 0.118 | 0.118 |
| (m) | 0.087 | 0.096 | 0.078 | **0.063** |
| (n) | **0.076** | 0.083 | 0.161 | 0.174 |
| (o) | 0.083 | 0.088 | **0.081** | **0.081** |
| (p) | **0.106** | **0.106** | 0.178 | 0.267 |

In this table, for each dataset, the best result obtained is highlighted in bold

We can see from Fig. 1 that our extension is useful mainly in large datasets, whereas it could not improve the results on the first examples. More concretely, for the first datasets, mainly "Breast Cancer" and "Lung Cancer" which are similar medical datasets, it is clear that our approach is not useful. In fact, the proposed approach could not find good initial solutions and the results obtained are then inferior to those of the PSO-based approach. This

**Table 3** Comparison of the prediction accuracy and the number of selected features

| Data | Our approach | PSO approach | $\chi^2$ approach | MI approach |
|---|---|---|---|---|
| (a) | 0.800 (4) | 0.970 (4) | 0.900 (4) | 0.920 (4) |
| (b) | 0.912 (5) | 0.998 (4) | 0.998 (8) | 1.000 (8) |
| (c) | 0.912 (5) | 0.957 (8) | 0.906 (8) | 0.892 (8) |
| (d) | 0.317 (13) | 0.304 (13) | 0.443 (13) | 0.242 (12) |
| (e) | 0.750 (23) | 0.969 (20) | 0.812(10) | 0.812 (8) |
| (f) | 0.750 (11) | 0.779 (13) | 0.788 (10) | 0.784 (10) |
| (g) | 0.999 (15) | 1.000 (17) | 0.997 (10) | 0.998 (10) |
| (h) | 0.500 (16) | 0.500 (33) | 0.500 (24) | 0.500 (30) |
| (i) | 0.694 (42) | 0.612 (36) | 0.704 (98) | 0.704 (98) |
| (j) | 0.950 (53) | 0.946 (70) | 0.923 (31) | 0.923 (31) |
| (k) | 0.665 (109) | 0.699 (111) | 0.714 (89) | 0.654 (89) |
| (l) | 1.000 (111) | 1.000 (117) | 0.935 (103) | 0.935 (103) |
| (m) | 0.980 (199) | 0.980 (225) | 0.960 (128) | 0.960 (87) |
| (n) | 1.000 (287) | 1.000 (311) | 0.861 (251) | 0.866 (252) |
| (o) | 1.000 (1580) | 1.000 (1667) | 1.000 (1524) | 1.000 (1524) |
| (p) | 0.986 (3367) | 0.986 (3367) | 0.861 (2376) | 0.750 (2386) |

is also the case in the "Zoo" dataset which has the smallest number of features, although it is a multi-class dataset. In contrast, we can see that the proposed approach is useful for large and high dimensional datasets. That is, for the "Madeleon", "Digits" and "Gisette" datasets, which have a high number of samples, our approach is able to find better initial solutions and obtain better fitness values. Regarding the "LSVT", "HAR", "Speech", "Gisette" and "Leukemia" datasets, which have the five highest numbers of features among the studied examples, we can see that our approach has a better convergence ability and that chunking has been useful in helping PSO to find better solutions, and move more efficiently in the search space.

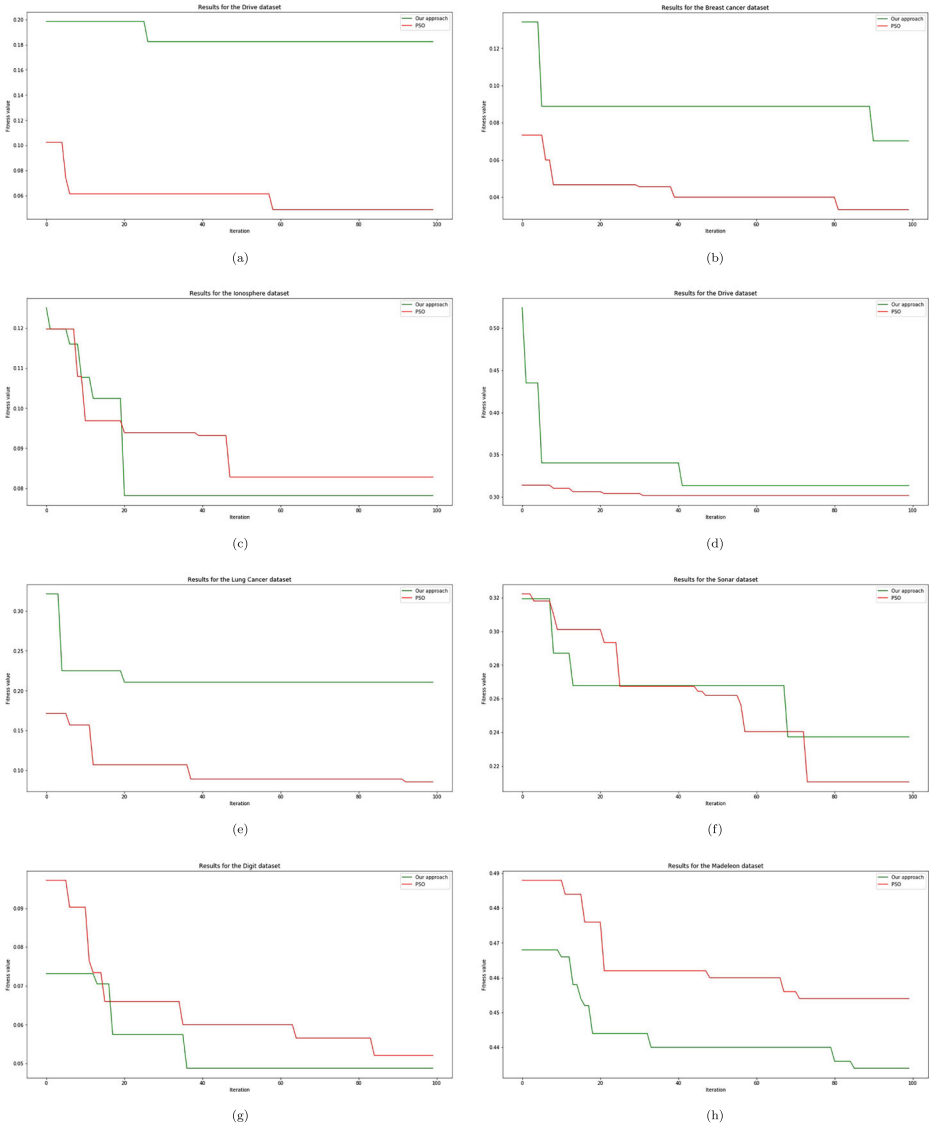



**Fig. 1** Comparison of our approach with the typical PSO-based approach

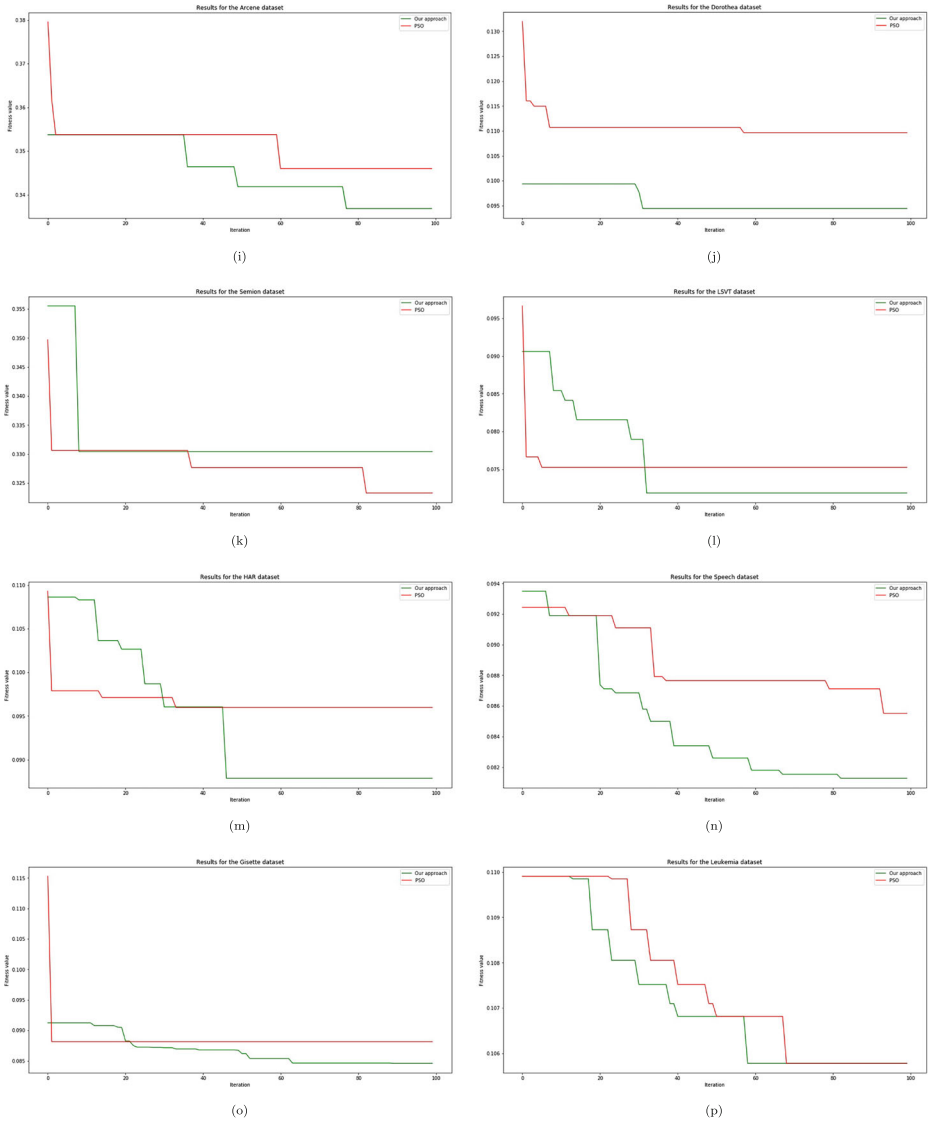
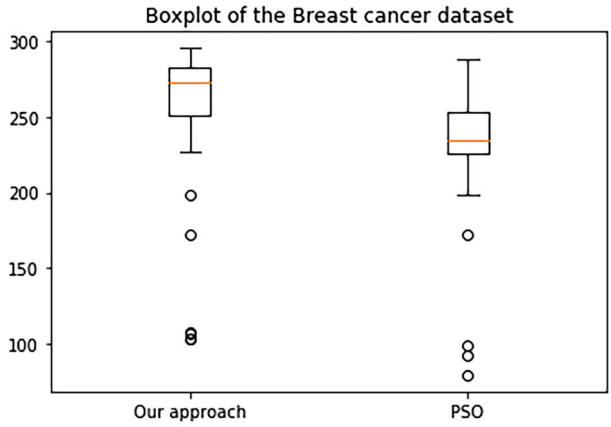


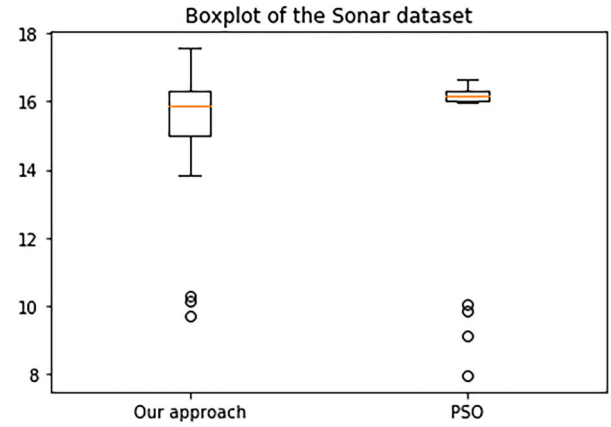

**Fig. 1** (continued)

Finally, we compare the approaches from a computational perspective. As the obtained CPU differs for each execution, we perform 30 runs to compare the CPU time of our approach with the PSO-based approach. Here, we show the results for three datasets, which are "Breast cancer", "Sonar" and "Semeion", and we start first by displaying the Boxplot for these datasets in Fig. 2.

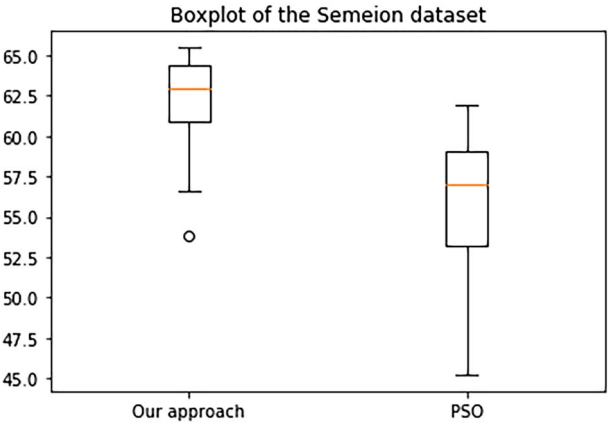
In addition, we have performed a parametric statistical test using a T-test for the three datasets and we found, based on the $p$-value, that the hypothesis of equal average is not rejected for the "Breast" and "Sonar" datasets, and is rejected for the "Semeion" dataset. In




(b)




(f)




(i)

**Fig. 2** Comparison of the CPU time (Boxplot)

other words, the CPU time of the PSO-based approach is significantly better only for the "Semeion" dataset, while there is no significant difference in the other datasets.

Regarding the comparison with respect to the CPU time obtained by using the $\chi^2$ and MI-based approaches, we can notice that the CPU time of both our approach and the PSO approach is significantly worse than their CPU time for most of the datasets.

### 4.3 Discussion of the results

The aim of this part is to summarize the results and to analyze the impact of our extension. First, based on the above results, we can conclude that our design of chunking and cooperative learning is mainly useful for large datasets. However, it should be noted that the approach implemented could not significantly improve the computation time. Instead, for examples like the "Semeion" dataset, the corresponding CPU time was worse than that of PSO. Also, for both, our approach and the PSO approach, the CPU time is significantly worse than the $\chi^2$ and MI approaches. Thereby, it is of utmost importance to take advantage of recent computational developments in order to improve the CPU time of the approach. Indeed, one of its advantages is that it is suitable for parallel computing (each sub-problem could be solved separately and in parallel) and, consequently, the CPU time could be considerably reduced by leveraging this feature.

By combining the results on accuracy and CPU time, we can conclude that both our approach and the PSO-based approach might be useful especially for large datasets. In fact, for small datasets, the $\chi^2$ and MI-based approaches could quickly give satisfactory results. The limitation of these filter approaches could appear in high-dimensional datasets and hence the examination of bio-inspired optimization will be justifiable mainly for large datasets. This can be noticed first by the fact that both approaches give better prediction accuracy, or the same for few examples, as shown in Table 3. Second, we can see from Table 2 that our approach is better suited than the native PSO for large-scale examined datasets. Both observations are valid for all datasets with at least 100 features, except the "Semeion" dataset. Therefore, we can see that the performance of bio-inspired algorithms differ on each dataset, and the findings of this paper suggest that advanced approaches such as the one proposed in this work will be of interest in large datasets, subject to providing the suitable implementation and computational tools. But, we note that other factors have also an impact on the results as are highlighted below.

First, we can notice that the initial solutions have an enormous effect on the performance of the algorithm. This finding is in line with previous FS research which strongly differentiates between forward and backward strategies [63]. Second, the results also depend on the nature of the dataset. Indeed, as we have seen, the behavior on cancer datasets is similar and our approach is not appropriate for both examples. In fact, to be able to completely understand the behavior of the approach, it is important to distinguish between the different types of datasets. For instance, for the "Leukemia" dataset, there are a number of specially tailored research studies that have attempted to deepen the understanding of it and its relevant features (e.g. [64]). Third, regarding the number of classes, it should be noted that our approach could not show to be effective in the first three multi-class datasets examined. But, the results seem to be due to the number of features and the data domain as stated above, and the number of classes does not appear to have a clear impact. In fact, the approach is better than the native PSO for the large multi-class dataset examined ("Gisette"). However, more results are needed to confirm this, and more datasets and a specific study are needed in this regard.

Another note is with respect to the independence of the different sub-problems. In this paper, our aimed contribution is to project the proposed ideas into the existing literature. However, except for [52], there is a lack of studies in FS in this regard. Also, the paper was not extended enough with this respect and, to the best of our knowledge, there is no way of independently dividing the problem of FS that was really validated. We think that this fact was an obstacle to obtaining more comprehensive results on the datasets examined. Moreover, the definition of the fitness function is also another important influencing factor that is not widely explored. Indeed, its definition also has an impact on the solution. In fact, while examining the approach on a related fitness function proposed in [57] on some of these datasets, we found quite different results. Therefore, this theme is also one of the open related issues.

In sum, our purpose in this paper is to provide an insight on the impact of the proposed extensions on PSO by providing a fair evaluation that shows both the strength and limitation of the approach, and to define proposals for improving the approach.

## 5 Conclusion

In this paper, we introduced a novel extension of PSO based on hybridization of two bio-inspired concepts which are chunking and cooperative learning. While introducing our method, we have explored how these approaches were introduced in different ways and re-iterated in a different form. That is, the concept of chunking was presented in different forms (e.g. clustering) and a lack of connection between different works was observed in this paper. Also, the concept of cooperative learning was adopted beyond swarm intelligence approaches (e.g. the POPMUSIC method). In this paper, we strengthened the awareness of the problem outlined in [7] by restricting our study to these concepts. Then, we combined and adapted the two concepts to the problem of feature selection. The experiment compared our method with a PSO-based wrapper approach and with two typical filter approaches. The results indicated that the approach is mainly suited for high-dimensional datasets and those with few classes, as shown in the different datasets examined. However, further study is needed to better explain these results.

On the one hand, future work should attempt to improve the computational efficiency of the approach using, for instance, parallel computing. On the other hand, the definition of the chunking approach is still an open issue. Instead of the minimum spanning tree-based method adopted in this paper, further research may try to incorporate domain knowledge into the chunking process. Indeed, as we have seen above, its impact could differ for each type of dataset, and the concept could be more intuitive for text classification problems (e.g. the "Digits" dataset), since the concept was originally proposed for language processing [19]. An example of such work could be found in [65]. Also, the referred paper proposed an in-depth mathematical analysis of the approach, which is also important for validating information theory (filter) approaches. Another option is to adopt the concept of variable-length PSO [26]. In fact, this idea, which was shortly described in Section 2.1, has shown promise for high-dimensional problems as the referenced paper asserted, and could then be a fruitful way to design an adequate generic approach for large datasets.

**Acknowledgements** This work is supported by the Alexander von Humboldt Foundation.

**Funding** Open Access funding enabled and organized by Projekt DEAL.

## References

1. Kar, A.K.: Bio inspired computing – a review of algorithms and scope of applications. Expert Syst. Appl. **59**, 20–32 (2016). https://doi.org/10.1016/j.eswa.2016.04.018
2. Dokeroglu, T., Sevinc, E., Kucukyilmaz, T., Cosar, A.: A survey on new generation metaheuristic algorithms. Computers & Industrial Engineering **137**, 106040 (2019). https://doi.org/10.1016/j.cie.2019.106040
3. Glover, F., Sörensen, K.: Metaheuristics. Scholarpedia **10**(4), 6532 (2015). https://doi.org/10.4249/scholarpedia.6532
4. Yang, X.-S. (ed.): Recent advances in swarm intelligence and evolutionary computation. Springer International Publishing, Berlin (2015). https://doi.org/10.1007/978-3-319-13826-8
5. Caserta, M., Voß, S.: Metaheuristics: Intelligent problem solving. In: Matheuristics, pp. 1–38. Springer US (2009). https://doi.org/10.1007/978-1-4419-1306-7_1
6. Abdelhafez, A., Luque, G., Alba, E.: Parallel execution combinatorics with metaheuristics: Comparative study. Swarm and Evolutionary Computation **55**, 100692 (2020). https://doi.org/10.1016/j.swevo.2020.100692
7. Sörensen, K.: Metaheuristics-the metaphor exposed. Int. Trans. Oper. Res. **22**(1), 3–18 (2013). https://doi.org/10.1111/itor.12001
8. Del Ser, J., Osaba, E., Molina, D., Yang, X.-S., Salcedo-Sanz, S., Camacho, D., Das, S., Suganthan, P.N., Coello, C.A.C., Herrera, F.: Bio-inspired computation: Where we stand and what's next. Swarm and Evolutionary Computation **48**, 220–250 (2019). https://doi.org/10.1016/j.swevo.2019.04.008
9. Villalón, C.L.C., Stützle, T., Dorigo, M.: Grey wolf, firefly and bat algorithms: Three widespread algorithms that do not contain any novelty. In: Lecture Notes in Computer Science, 12421, pp. 121–133, Springer International Publishing (2020). https://doi.org/10.1007/978-3-030-60376-2_10

10. de Armas, J., Lalla-Ruiz, E., Tilahun, S.L., Voß, S.: Similarity in metaheuristics: a gentle step towards a comparison methodology. Nat. Comput., pp. 1–23, https://doi.org/10.1007/s11047-020-09837-9 (2021)
11. Reid, C.: Cooperation vs. competition in a technological environment. Ph.D. Thesis, University of Queensland Library. https://doi.org/10.14264/uql.2017.194 (2016)
12. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proceedings of ICNN'95 - International Conference on Neural Networks, IEEE (1995). https://doi.org/10.1109/icnn.1995.488968
13. Kennedy, J., Eberhart, R.C.: A discrete binary version of the particle swarm algorithm. In: IEEE international conference on systems, man, and cybernetics. computational cybernetics and simulation, IEEE (1997). https://doi.org/10.1109/icsmc.1997.637339
14. Guyon, I., Elisseeff, A.: An introduction to variable and feature selection. J. Mach. Learn. Res. **3**, 1157–1182 (2003). https://doi.org/10.1162/153244303322753616
15. Hall, M.A.: Correlation-based feature selection for machine learning. Ph.D. Thesis, University of Waikato (1999)
16. Bommert, A., Sun, X., Bischl, B., Rahnenführer, J., Lang, M.: Benchmark for filter methods for feature selection in high-dimensional classification data. Computational Statistics & Data Analysis **143**, 106839 (2020). https://doi.org/10.1016/j.csda.2019.106839
17. Raza, M.S., Qamar, U.: Introduction to feature selection. In: Understanding and Using Rough Set Based Feature Selection: Concepts, Techniques and Applications, pp. 1–25, Springer Singapore (2019). https://doi.org/10.1007/978-981-10-4965-1_1
18. Sarhani, M., Voß, S.: PSO-based cooperative learning using chunking. In: Lecture Notes in Computer Science, 12096, pp. 278–288, Springer International Publishing (2020). https://doi.org/10.1007/978-3-030-53552-0_26
19. Miller, G.A.: The magical number seven, plus or minus two: some limits on our capacity for processing information. Psychol. Rev. **63**(2), 81–97 (1956). https://doi.org/10.1037/h0043158
20. Nievergelt, J.: Information content of chess positions. ACM SIGART Bulletin, (62), pp. 13–15. https://doi.org/10.1145/1045398.1045400 (1977)
21. Laird, J.E., Rosenbloom, P.S., Newell, A.: Chunking in Soar: The anatomy of a general learning mechanism. Mach. Learn. **1**(1), 11–46 (1986). https://doi.org/10.1007/bf00116249
22. Woodruff, D.L.: Proposals for chunking and tabu search. Eur. J. Oper. Res. **106**(2-3), 585–598 (1998). https://doi.org/10.1016/s0377-2217(97)00293-2
23. Voß, S., Gutenschwager, K.: A chunking based genetic algorithm for the Steiner tree problem in graphs. In: Pardalos, P.M., Du, D.-Z. (eds.) Network Design: Connectivity and Facilities Location, 40, pp. 335–355, AMS, Princeton (1998). https://doi.org/10.1090/dimacs/040/20
24. Woodruff, D.L.: A chunking based selection strategy for integrating meta-heuristics with branch and bound. In: Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization, pp. 499–511, Springer US (1999). https://doi.org/10.1007/978-1-4615-5775-3_34
25. Stringer, H., Wu, A.S.: Winnowing wheat from chaff: The chunking GA. In: Genetic and Evolutionary Computation – GECCO 2004, pp. 198–209, Springer Berlin Heidelberg (2004). https://doi.org/10.1007/978-3-540-24855-2_18
26. Tran, B., Xue, B., Zhang, M.: Variable-length particle swarm optimization for feature selection on high-dimensional classification. IEEE Transactions on Evolutionary Computation **23**(3), 473–487 (2019). https://doi.org/10.1109/tevc.2018.2869405
27. Ramkumar, P., Acuna, D.E., Berniker, M., Grafton, S.T., Turner, R.S., Kording, K.P.: Chunking as the result of an efficiency computation trade-off. Nat. Commun., 7(12176). https://doi.org/10.1038/ncomms12176 (2016)
28. Yoon, M.: A constant-time chunking algorithm for packet-level deduplication. ICT Express **5**(2), 131–135 (2019). https://doi.org/10.1016/j.icte.2018.05.005
29. Yang, S., Li, C.: A clustering particle swarm optimizer for locating and tracking multiple optima in dynamic environments. IEEE Transactions on Evolutionary Computation **14**(6), 959–974 (2010). https://doi.org/10.1109/tevc.2010.2046667
30. Jovanovic, R., Tuba, M., Voß, S.: Fixed set search applied to the traveling salesman problem. In: International Workshop on Hybrid Metaheuristics, pp. 63–77, Springer (2019). https://doi.org/10.1007/978-3-030-05983-5_5
31. Maniezzo, V., Stüzle, T., Voß, S. (eds.): Matheuristics: Hybridizing metaheuristics and mathematical programming. Springer Verlag, Berlin (2010)
32. Taillard, E.D., Voß, S.: POPMUSIC — Partial Optimization Metaheuristic under Special Intensification Conditions. In: Essays and surveys in metaheuristics. Operations Research/Computer Science Interfaces Series, 15, pp. 613–629, Springer US (2002). https://doi.org/10.1007/978-1-4615-1507-4_27
33. Kronfeldner, M.: Divide and conquer. Oxford University Press, Oxford (2018). https://doi.org/10.1093/oso/9780198823650.003.0011

34. 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). https://doi.org/10.1109/tevc.2005.857610
35. Aoun, O., Sarhani, M., El Afia, A.: Particle swarm optimisation with population size and acceleration coefficients adaptation using hidden Markov model state classification. International Journal of Metaheuristics **7**(1), 1–29 (2018). https://doi.org/10.1504/ijmheur.2018.091867
36. van den Bergh, F., Engelbrecht, A.: A cooperative approach to particle swarm optimization. IEEE Trans. Evol. Comput. **8**(3), 225–239 (2004). https://doi.org/10.1109/TEVC.2004.826069
37. Niu, B., Zhu, Y., He, X., Wu, H.: MCPSO: A multi-swarm cooperative particle swarm optimizer. Appl. Math. Comput. **185**(2), 1050–1062 (2007). https://doi.org/10.1016/j.amc.2006.07.026
38. Aoun, O., El Afia, A., Talbi, E.-G.: A cooperative multi-swarm particle swarm optimizer based hidden Markov model. In: Heuristics for Optimization and Learning, Studies in Computational Intelligence, 906, pp. 315–334. Springer International Publishing (2020). https://doi.org/10.1007/978-3-030-58930-1_21
39. Sun, C., Jin, Y., Cheng, R., Ding, J., Zeng, J.: Surrogate-assisted cooperative swarm optimization of high-dimensional expensive problems. IEEE Transactions on Evolutionary Computation **21**(4), 644–660 (2017). https://doi.org/10.1109/tevc.2017.2675628
40. Cheng, R., Jin, Y.: A social learning particle swarm optimization algorithm for scalable optimization. Inform. Sci. **291**, 43–60 (2015). https://doi.org/10.1016/j.ins.2014.08.039
41. Li, X., Yao, X.: Cooperatively coevolving particle swarms for large scale optimization. IEEE Transactions on Evolutionary Computation **16**(2), 210–224 (2012). https://doi.org/10.1109/tevc.2011.2112662
42. Dorigo, M., Gambardella, L.M.: Ant colony system: a cooperative learning approach to the traveling salesman problem. IEEE Transactions on Evolutionary Computation **1**(1), 53–66 (1997). https://doi.org/10.1109/4235.585892
43. Sondergeld, L., Voß, S.: Cooperative intelligent search using adaptive memory techniques. In: Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization, pp. 297–312. Springer US (1999). https://doi.org/10.1007/978-1-4615-5775-3_21
44. El-Abd, M., Hassan, H., Anis, M., Kamel, M.S., Elmasry, M.: Discrete cooperative particle swarm optimization for FPGA placement. Appl. Soft Comput. **10**(1), 284–295 (2010). https://doi.org/10.1016/j.asoc.2009.07.011
45. Segredo, E., Lalla-Ruiz, E., Hart, E., Voß, S.: On the performance of the hybridisation between migrating birds optimisation variants and differential evolution for large scale continuous problems. Expert Syst. Appl. **102**, 126–142 (2018). https://doi.org/10.1016/j.eswa.2018.02.024
46. Lalla-Ruiz, E., Segredo, E., Voß, S.: A cooperative learning approach for the quadratic knapsack problem. In: Lecture Notes in Computer Science, 11353, pp. 31–35, Springer International Publishing (2018). https://doi.org/10.1007/978-3-030-05348-2_3
47. Blackwell, T., Branke, J.: Multi-swarm optimization in dynamic environments. In: Lecture Notes in Computer Science, 3005, pp. 489–500. Springer Berlin Heidelberg (2004). https://doi.org/10.1007/978-3-540-24653-4_50
48. Lessmann, S., Voß, S.: Feature selection in marketing applications. In: Advanced Data Mining and Applications, pp. 200–208. Springer Berlin Heidelberg (2009). https://doi.org/10.1007/978-3-642-03348-3_21
49. Unler, A., Murat, A.: A discrete particle swarm optimization method for feature selection in binary classification problems. Eur. J. Oper. Res. **206**(3), 528–539 (2010). https://doi.org/10.1016/j.ejor.2010.02.032
50. Xue, B., Zhang, M., Browne, W.N.: Particle swarm optimisation for feature selection in classification: Novel initialisation and updating mechanisms. Appl. Soft Comput. **18**, 261–276 (2014). https://doi.org/10.1016/j.asoc.2013.09.018
51. Vieira, S.M., Sousa, J.M.C., Runkler, T.A.: Two cooperative ant colonies for feature selection using fuzzy models. Expert Syst. Appl. **37**(4), 2714–2723 (2010). https://doi.org/10.1016/j.eswa.2009.08.026
52. Song, Q., Ni, J., Wang, G.: A fast clustering-based feature subset selection algorithm for high-dimensional data. IEEE Transactions on Knowledge and Data Engineering **25**(1), 1–14 (2013). https://doi.org/10.1109/tkde.2011.181
53. Moradi, P., Rostami, M.: Integration of graph clustering with ant colony optimization for feature selection. Knowl.-Based Syst. **84**, 144–161 (2015). https://doi.org/10.1016/j.knosys.2015.04.007
54. Banka, H., Dara, S.: A hamming distance based binary particle swarm optimization (HDBPSO) algorithm for high dimensional feature selection, classification and validation. Pattern Recogn. Lett. **52**, 94–100 (2015). https://doi.org/10.1016/j.patrec.2014.10.007
55. Sarhani, M., El Afia, A., Faizi, R.: Facing the feature selection problem with a binary PSO-GSA approach. In: Operations Research/Computer Science Interfaces Series, 62, pp. 447–462. Springer International Publishing (2017). https://doi.org/10.1007/978-3-319-58253-5_26

56. Vieira, S.M., Mendonça, L.F., Farinha, G.J., Sousa, J.M.C.: Modified binary PSO for feature selection using SVM applied to mortality prediction of septic patients. Appl. Soft Comput. **13**(8), 3494–3504 (2013). https://doi.org/10.1016/j.asoc.2013.03.021
57. Vignolo, L.D., Milone, D.H., Scharcanski, J.: Feature selection for face recognition based on multi-objective evolutionary wrappers. Expert Syst. Appl. **40**(13), 5077–5084 (2013). https://doi.org/10.1016/j.eswa.2013.03.032
58. Mafarja, M.M., Mirjalili, S.: Hybrid whale optimization algorithm with simulated annealing for feature selection. Neurocomputing **260**, 302–312 (2017). https://doi.org/10.1016/j.neucom.2017.04.053
59. Boser, B.E., Guyon, I.M., Vapnik, V.N.: A training algorithm for optimal margin classifiers. In: Proceedings of the fifth annual workshop on Computational learning theory - COLT'92. ACM Press (1992). https://doi.org/10.1145/130385.130401
60. Miranda, L.J.V.: PySwarms: a research toolkit for particle swarm optimization in Python. The Journal of Open Source Software **3**(21), 433 (2018). https://doi.org/10.21105/joss.00433
61. Liu, H., Setiono, R.: Chi2: feature selection and discretization of numeric attributes. In: Proceedings of 7th IEEE International Conference on Tools with Artificial Intelligence, pp. 388–391. IEEE (1995). https://doi.org/10.1109/tai.1995.479783
62. Ross, B.C.: Mutual information between discrete and continuous data sets. PLoS ONE **9**(2), e87357 (2014). https://doi.org/10.1371/journal.pone.0087357
63. Mao, K.Z.: Orthogonal forward selection and backward elimination algorithms for feature subset selection. IEEE Transactions on Systems, Man and Cybernetics, Part B (Cybernetics) **34**(1), 629–634 (2004). https://doi.org/10.1109/tsmcb.2002.804363
64. Srisukkham, W., Zhang, L., Neoh, S.C., Todryk, S., Lim, C.P.: Intelligent leukaemia diagnosis with bare-bones PSO based feature optimization. Appl. Soft Comput. **56**, 405–419 (2017). https://doi.org/10.1016/j.asoc.2017.03.024
65. Dhillon, I.S., Mallela, S., Kumar, R.: A divisive information-theoretic feature clustering algorithm for text classification. J. Mach. Learn. Res. **3**, 1265–1287 (2003). https://doi.org/10.1162/153244303322753661