8

Under-Updated Particle Swarm Optimization for Small Feature Selection Subsets from Large-Scale Datasets

Victor Trevino and Emmanuel Martinez
Tecnológico de Monterrey, Campus Monterrey, Cátedra de Bioinformática
México

1. Introduction

Feature selection is the process of choosing a subset of features from the original larger set that are related to a certain outcome, such as disease type, dose, income, and time to event. The use of feature selection procedures is almost compulsory and complex in biology and medicine because the generation of massive datasets is nowadays common for many state-of-the-art technologies such as transcriptomics, proteomics, metabolomics, and genomics where a single, conventional, and relatively cheap experiment may yield the measurement of several thousands of features per sample (Hieter and Boguski, 1997; Sauer et al., 2005). In such cases, feature selection is used to reduce complexity and large computational costs, as well as to improve pattern recognition accuracy, data interpretability and hypothesis generation (Shen et al., 2008; Vapnik, 1998; Guyon et al., 2002).

Filter, embed, and wrapper methods have been proposed for feature selection (Neumann et al., 2005; Guyon and Elisseeff, 2003; Saeys et al., 2007). Filter methods, also known as univariate methods, apply a rule to every feature ignoring any other feature. The filter consists of a classical statistical test such a t-test, a ranking procedure such a signal-to-noise ratio, or an empirical permutation test as the evaluation procedure (Saeys et al., 2007; Golub et al., 1999). The search engine is composed of a model selection procedure such as forward and backward elimination strategies or a procedure that choose top features by a threshold. Embed methods couple the evaluation rule to the search, for example, by using the loadings vector associated to a component in Principal Component Analysis (Carzaniga et al., 2007), or by using the weight vector from Support Vector Machines (Guyon and Elisseeff, 2003; Guyon et al., 2002). Wrapper methods utilize independently a search engine and an evaluation procedure to choose good feature combinations. The search engine is guided depending on the evaluation of feature combinations. Because of the independency in the implementation of the search engine and the evaluation procedures, wrapper methods are highly attractive as a general tool for many types of datasets.

Although in large datasets the number of features related to an outcome could be high, there are some reasons why one would like to design a predictive model containing a small feature subset. The models that contain many variables tend to over-fit the data (Occam's razor) and to be more complex, difficult to interpret, expensive, and hard to implement in

practice. For example, in a clinical test, assessing a large number of indicators is complex, tedious, and more expensive than testing only a handful of indicators. We have previously shown that models containing around 10-20 features can be found in functional genomics data (Trevino and Falciani, 2006; Martinez et al., 2010).

Particle Swarm Optimization (PSO) is a recent and very successful wrap-based search engine (Eberhart et al., 2001). However, standard implementations for feature selection fail or their performance is severely affected in selecting small feature subsets from large datasets (see section 3.1 for some grounds). For these reasons, some authors have adapted the standard binary PSO (sbPSO) presented by (Eberhart et al., 2001) implementing ad-hoc algorithms to overcome the dataset size problem (Wang et al., 2007; Chuang et al., 2008; Xu et al., 2007; Takahashi Monteiro and Yukio, 2007; Bello et al., 2007; Alba et al., 2007). Nevertheless, these implementations are mainly focused on maximizing the evaluation accuracy and little or no consideration is done to minimize the number of features selected (for an exception see (Alba et al., 2007)).

In this paper, we propose a wrapper feature selection method based on PSO named Under-Updated Particle Swarm Optimization (uuPSO) designed to efficiently select small feature subsets from large-scale functional genomics data. For illustrative purposes, we used a 10-fold-cross-validation error estimation coupled within a nearest centroid classifier as the evaluation procedure in five large datasets. We focused on selecting small feature subsets at maximum accuracy from functional genomics data that hopefully will help in designing cheap and easy to implement clinical assays. We show that our algorithm is capable of selecting the most relevant genes. We also demonstrate that uuPSO is able to find alternative models that could not be found with univariate or ranking methods such as SVM-RFE.

The remainder of the paper is organized as follows. Section 2 describes PSO methodology and the standard feature selection algorithm. Section 3 describes the rationale of the implemented procedures in uuPSO to improve the selection of small feature subsets. Section 3 also includes the datasets used and the experiments performed to show their effects. Section 4 describes and interprets the results of each experiment. Section 5 provides a general discussion and practical advices for the selection of parameter values. Finally, section 6 includes a summary and concluding remarks.

2. Background

2.1 Particle Swarm Optimization

Recalling, PSO is an optimization technique inspired in social behavior observed in nature (Eberhart et al., 2001). PSO algorithm starts by creating a swarm of particles placed randomly in the solution landscape. The swarm explores the solution landscape guided by particle leaders. Particle leaders are those particles better placed in the solution landscape. This is achieved using a goodness function whose input is the particle position. The leadership can be global attracting all particles, or local pulling only neighboring particles. If the new goodness function evaluation results in higher values than those of the leaders, leadership is changed. Thus, particles will update their position depending on their own experience following global and local bests but attempting being better on their way. The process of updating all particle positions and velocities is called *epoch* in the PSO paradigm. The position of a particle is updated as

$$x_{id}(t) = x_{id}(t - 1) + v_{id}(t) \quad (1)$$

where i is the i-particle of the swarm, d is the space dimension, t is the epoch, v is the velocity, and x_{id} is the position of the i-particle in the dimension d. All $x_{id}(0)$ are set randomly. The velocity of a particle i for dimension d is defined as

$$v_{id}(t) = w * v_{id}(t - 1) + r_1 * c_1 * (p_{ld} - x_{id}(t - 1)) + r_2 * c_2 * (p_{gd} - x_{id}(t - 1)) \quad (2)$$

where p_{gd} is the position of the best particle in the dimension d, p_{ld} is the position of the local best particle in the dimension d, w is used as an inertial value, r_1 and r_2 are positive random numbers, and c_1 and c_2 are local and global weights respectively (commonly summing 4). The local best is commonly designated as the best of the c neighbors in the swarm array. When a particle i has changed its position x_i, the evaluation function will determine whether it is the new global or local best depending on the goodness compared to the swarm best goodness or the goodness of the c neighbor particles respectively. Note that for the global best particle only the inertial term will be actually used because p_{ld} and p_{gd} will be both i cancelling both terms. To avoid this, an elitism scheme is usually used, which consists on ignoring the global best particle from the update procedure.

2.2 Binary PSO for feature selection

The common implementation for feature selection using PSO, known as standard binary PSO (sbPSO), simply sets the maximum dimension as the total number of features, then employs the value of v_{id} to decide whether the feature d for particle i is selected. This is achieved by the use of a sigmoid function s (Eberhart et al., 2001):

$$\text{if } r_{id} < s(v_{id}(t)) \text{ then } x_{id}(t) = 1; \text{ else } x_{id}(t) = 0$$

$$s(v_{id}) = 1/(1 + exp(-v_{id})) \quad (3)$$

where r_{id} is a random number drawn from a uniform distribution between 0 and 1. In this way, the goodness function would use only the selected features for the evaluation (those whose $x_{id}(t) = 1$). In practice, v_{id} is limited between the range $[-v_{max}; v_{max}]$ where v_{max} is usually 4. This establishes practical limits for the sigmoid function and helps to change direction faster. Also, the inertial w is gradually changed during the course of the procedure, e.g. from 0.9 in early epochs to 0.4 in late epochs.

2.3 Others implementations for feature selection using PSO

Some authors have proposed modifications to sbPSO algorithm in order to improve performance in terms of search and evaluation capabilities (Chuang et al., 2008; Xu et al., 2007; Wang et al., 2007; Takahashi Monteiro and Yukio, 2007; Bello et al., 2007). However, these implementations are mainly focused in maximizing the evaluation accuracy. Little or no consideration is done to minimize the number of features selected (Alba et al., 2007). Therefore, in most of these methods a large number of features were selected (Chuang et al., 2008; Xu et al., 2007). Others implementations are more difficult to review in the context of small feature subset selection because datasets used contained only around a hundred features (Takahashi Monteiro and Yukio, 2007; Wang et al., 2007). One of the adaptations

that attempted to control the number of selected features was introduced by (Xu et al., 2007) defining a parameter *f* to bias the random decision for activating features as follows:

$$\text{if } r_{id} + f < s(v_{id}(t)) \text{ then } x_{id}(t) = 1; \text{ else } x_{id}(t) = 0$$

In this way, high values of f would tend to turn off most of the variables. The experiments from (Xu et al., 2007) using $f = 0.45$ resulted in models containing from 60 to 120 features which still seems too high for our purposes. However, this adaptation provides the concept of activating a smaller number of features by changing the selection decision.

A great effort to select small feature subsets was introduced by (Wang et al., 2007) in which only a subset of x_{id} was updated. They used a unique velocity per particle v_i which roughly means how many x_{id} bits should be changed. If v_i is less than or equal to x_g (the number of different variables of x_{id} from the best global particle), v_i bits are randomly changed different from those of x_g. If v_i is greater than x_g, x_g is copied to x_i (the complete particle) and v_i - x_g bits are also swapped to explore neighboring space. If v_i is close to zero this would mean that the particle is approaching to a stable and optimal solution whereas high values of v_i would indicate poor fitting and large navigation distances. Although the authors tested datasets containing less than a hundred features, this work is a milestone because it introduces the concept of updating only a subset of dimensions rather than all dimensions. Other adaptation of PSO replaces the concept of PSO velocity by a complex crossover operator resembling genetic algorithms rather than to PSO (Alba et al., 2007).

3. Methods

In this section, we describe the adaptations to sbPSO in order to select small feature subsets from massive datasets such as gene expression data. The parameters introduced in these adaptations are intended to show the effects of our adaptations rather than to burden the exquisite simplicity of the PSO algorithm. Finally, the datasets used for the experiments are mentioned and the values of parameters employed are listed.

3.1 Initialization

sbPSO has been successful in a variety of feature selection problems (Eberhart et al., 2001). However, the performance in accuracy, feature selection, and computation time of sbPSO is inadequate when data consists of thousands of variables, which is common for functional genomics data. In the sbPSO, the random initialization of v_{id} and x_{id} would select 50% of the features just by chance. This issue affects drastically the performance and goodness in some aspects. First, the performance of multivariate methods (classification or regression) that would be inevitably part of the goodness function depends on the number of features and matrix operations such as inversions. This would consume unnecessary CPU time. Second, the error estimation consists of the evaluation of these multivariate methods several times, such as the cross-validation methods worsening the CPU time consumption. Third, given that particles velocities depend on the global and local best, which will contain a large number of features selected, several epochs would be needed to decrease the number of selected features from thousands to a handful by the standard PSO selection. Fourth, the overall goodness could be decreased by the overload of selected features that might not be related to the predictor. This could confuse multivariate methods generating random effects.

For these reasons, our first proposal is to control the number of active features in the initialization procedure. To achieve this, we used a predefined constant b, which specifies the number of random features that will be initialized to 1 ($x_{ib} = 1$). This procedure is independent of the initialization of v_{id}, which is random in all dimensions as usual.

3.2 Updates

If initialization as stated in previous section is used alone, it would not have long-term effects because v_{id} is initialized as random, so updates in further epochs will reset initial x_{id} values. Thus, we included an adaptation to limit the total number of v_{id} updates, which is controlled by a user-specified parameter u. In order to choose which features will be updated, we thought that particles should maintain the essence of the PSO algorithm, that particles should follow the global and local best expecting to be better on their way. So, to manage this, a candidate list of updates is formed for each particle i. This list is populated from the union of the active features (x_{id}) for particle i, the active features from global best g, and those from local best l. In addition, the candidate list also considers e random updates, which is presented in the next paragraph as "innovation". Finally, only u random features are updated from the candidate list generated. In this way, particles can even fly in direction of the global best and local best, although slower than in sbPSO, still leaving some room for innovation to fulfill the expectation of being even better. In social terms, as in the original PSO paradigm, this could be interpreted as if a particle cannot make an effort to imitate all the features of the best particle (global and local) at the same time, sometimes it will imitate certain features and in other occasions it will imitate another. This constrained imitation seems, by common sense, a reasonable social behavior.

3.3 Innovation

In sbPSO, innovation is the combined result of updating all dimensions and the random decision of activating a feature. However, considering only the first two adaptations described so far, we would update only u dimensions chosen from the particle itself, the local best and global best. Consequently, the universe of updatable x_{id} would be limited to the initial random activation. If some features were not initially activated within the swarm population, those features would never be explored. For instance, for a swarm population of 50 and an initial activation of 5 features, there will be at most 250 updatable features for the entire swarm. This number seems scarce compared to the functional genomics datasets we are focusing on, which may contain thousands of features. In addition, global best would be stuck since it has no way to update other features. Hence, the swarm could converge quickly to a poor local optimum. For these reasons, we introduced a mechanism to control how many random features will be included in the candidate list of updates per particle, managed by a parameter e. In the PSO social terms, this would mean that imitation of the best particles is combined with a sense of uniqueness, hopefully surpassing those best particles. To show the effects of innovation, we ran some experiments varying e.

3.4 Number of selected features

The adaptations shown above do not ensure that the number of selected features will be small. Although the constrained number of updates would limit over-activation, activation

might change freely during the course of the epochs. Therefore, we introduced another adaptation to keep the number of selected features within a range. We set n as the minimum number of active features and m as the maximum. After updating, if the number of active features a is larger than m, then a-m randomly chosen features are turned off. On the contrary, if a is less than n, n-a randomly chosen features are turned on, which would provide other mechanism for innovation.

3.5 Under-updated Particle Swarm Optimization algorithm

Considering the adaptations described in previous paragraphs, the uuPSO pseudo-code algorithm to maximize the Goodness function is shown in Algorithm 1. The algorithm can be easily thought as a generalization of sbPSO. If the total number of features is k, setting $b = k / 2$, $u = k$, $n = 1$, and $m = k$ should behave as sbPSO.

```
Initialize v_id randomly;
/ / Adaptation 3.1
q = draw b features;
for (j in q) { x_ij(0) = 1; }
epoch = 0;
while (epoch < limit or Goodness(p_gd) > goal) {
        epoch = epoch + 1;
        for (i=1 to number of particles) {
                if (Goodness(x_i) > Goodness(p_g)) then p_g = x_i;
                if p_g != x_i OR NOT Elitism then p_l = x_i;
                for (j in the indexes of neighbors) {
                        if (Goodness(x_j) > Goodness(p_l)) then p_l = x_j;
                }
/ / Adaptation 3.2
                candidates = union(active(x_i), active(p_l), active(p_g));
                innovation = draw e features not present in candidates;
                candidates = union(candidates, innovation);
                toupdate  = draw u  features from candidates;
                for (d in toupdate) {
                        vi_d (t) = w * v_id (t - 1) + r_1 * c_1 * ( p_ld - x_id (t - 1)) + r_2 * c_2 * ( p_gd - x_id (t - 1))
                        if v_id(t) > v_max then v_id(t) = v_max;
                        if v_id(t) < -v_max then v_id(t) = -v_max;
                        if r_id < s(v_id(t)) then x_id(t) = 1 else x_id(t) = 0
                }
/ / Adaptation 3.3
                while (active(x_i) < n) {
                        q  = draw one feature not active(x_i);
                        x_iq(t) = 1; }
                while (active(x_i) > m) {
                        q  = draw one feature not active(x_i);
                        x_iq(t) = 0; }
        }
}
```

Algorithm 1. Under-Updated Particle Swarm Optimization Algorithm.

3.6 Datasets

In this paper we focused on the selection of small subsets of features from large-scale datasets. In this context, we mainly used two datasets generated using microarray data from breast cancer and leukemia that had been previously reported and studied in a similar fashion (Trevino and Falciani, 2006). For comparison with other PSO algorithms, we also used other three datasets in order to show overall effects and the generality of our adaptations (see Table 1).

Dataset	*Molecular Data*	*Features*	*Classes*	*Samples per class*
Breast Cancer	mRNA Rosetta	2,920	2	44 No metastases, 34 metastases
Leukemia Yeoh	mRNA Affymetrix	2,435	5	27 E2A-PBX, 79 TEL, 64 Hyp+50, 20 MLL, 43 T
Leukemia Golub	mRNA Affymetrix	7,219	2	47 AML, 25 ALL
Colon Cancer	mRNA Affymetrix	2,000	2	40 Tumor, 22 Normal
Ovarian Cancer	Proteomics	15,154	2	162 Tumor, 91Normal

Table 1. Datasets considered for this study.

The breast cancer (BC) dataset was originally developed and published by van't Veer et al. (2002) obtained from the gene expression taken from 44 patients with no metastases developed within the first five years, and 34 patients positive for metastases within the first five years. Data were normalized as described in the original publication. Genes with a p-value larger than 0.001 were filtered out (confidence level that a gene's mean ratio is significantly different from 1). Finally, the data used consisted of a matrix of 2,920 features (genes) * 78 samples and a binary vector indicating the state of metastases. It is worth mentioning that some of the best multivariate-selected classifiers published for this dataset lie at around 80% accuracy whereas the univariate-selected classifiers lie at around 65% ((Trevino and Falciani, 2006), supplementary material). Thus, this dataset is considered a difficult dataset.

The leukemia-golub dataset (Golub et al., 1999) contains 72 bone marrow samples that correspond to two types of leukemia: 47 Acute Myeloid Leukemia (AML) and 25 Acute Lymphoblastic Leukemia (ALL). It was obtained from an Affymetrix high-density oligonucleotide microarray that includes 7129 genes.

The colon cancer dataset was obtained from the expression levels of 2000 genes using Affymetrix oligonucleotide microarrays (Alon et al., 1999). The genes correspond to 40 tumor and 22 normal colon tissue samples. Data were quantile-normalized before processing (Bolstad et al., 2003).

The leukemia-yeoh (LY) dataset was developed by (Yeoh et al., 2002) using Affymetrix microarrays, and describes the gene expression profile of 327 acute lymphoblastic leukemia

patients representing 7 different disease sub-classes. In this paper, we have selected the five largest classes: E2A-PBX, Hyp+50, MLL, T, and TEL. These sub-classes include respectively 27, 64, 20, 43, and 79 samples. Data have been filtered to eliminate the most invariant genes. The standard deviation and difference between maximum and minimum expression values were calculated for each gene. The genes were ranked by these values and selected if they were within the top 15%. Finally, the dataset used consisted of 2,435 features (genes) * 233 samples and a string label indicating the sample sub-class. Although this dataset compromise 5 classes, it is considered an easy dataset because there are several features correlated to classes (Yeoh et al., 2002) and both univariate and multivariate searches have found models around 98% accuracy ((Trevino and Falciani, 2006), supplementary material).

The ovarian cancer dataset consists of proteomic spectral data from serum (Petricoin et al., 2002).This dataset comprises on 15,154 mass/charge (M/Z) identities obtained from 91 normal individuals and 162 ovarian cancers. Data were quantile normalized before processing. A summary of the datasets used is listed by cancer type or author in Table 1.

3.7 Experiments

As explained in previous paragraphs, we tested the algorithm varying the controlling parameters. For each combination of parameters, we ran the uuPSO algorithm 100 times and the goodness and active features of the global best was monitored during 1000 epochs. No other termination criteria were active. The median of the 100 runs for each epoch is reported. If not stated, the values of the parameters used were b = 10, n = 10, m = 10, u = 10, e = 10, *elitism* = true. The standard PSO parameters used were v_{max} = 4, w = 0.9-0.0005*epoch (corresponding to an initial w = 0.9 and final w = 0.4), and $c_1 = c_2 = 2$. The swarm size was 20. The goodness function was 1-*cve*. *cve* is a 10-fold cross-validation error estimation procedure. Error in each fold was estimated by the percentage of miss-classified test samples using a nearest centroid method in Euclidean space. A centroid is defined as the mean of a given set of samples of the same class. Thus, after estimating all centroids for corresponding training classes in each gene, the nearest centroid for a predicting sample is the centroid whose Euclidean distance is minimal, as follows:

$$class = \min_k(\sqrt{\sum(x_i - c_{ki})^2}) \tag{4}$$

where x_i is the value of the i gene and centroid c_{ki} is the mean for gene i in class k. We have used, mainly, a nearest centroid classifier for its simplicity and high speed. However, we also made comparisons with more powerful classifiers like SVM.

All runs were performed in a Dell PowerEdge SC1435 with two dual AMD opteron processors and 8Gb of memory based on CentOS Linux Distribution (http://www.centos.org/). No more than one run per core processor was performed at the same time. Scripts were written in Java. We used official Sun Java(TM) SE Runtime Environment (build 1.6.0-10-b33) and the official virtual machine Java HotSpot (TM) 64-Bit Server VM (build 11.0-b15, mixed mode) for all runs.

For the support vector machines-recursive feature elimination, we used the R implementation in the package e1071 (Chang and Lin, 2001). For the support vector machines-forward selection, we added genes one by one following the order given by the p-

value from an f-test. This forward-selection strategy is similar to that in PAM (Tibshirani et al., 2002) and the Prophet tool in GEPAS (Montaner et al., 2006).

Dataset	*All*	*PSO*	*Best Model*		*Accuracy*		*Features*		*Time*
	Acc	*Algorithm*	*Size*	*Acc*	*Mean*	*SD*	*Mean*	*SD*	*(hr)*
Breast-Cancer	0.667	uuPSO	13	0.923	**0.842**	0.026	**13**	2	**0.74**
		sbPSO	1470	0.756	0.686	0.009	1463	26.1	113.67
		XuPSO	7	0.872	0.827	0.016	‡20	17.6	5.14
		Wang	1386	0.782	0.758	0.015	1447	25.9	89.11
Leukemia-Yeoh	0.983	uuPSO	14	1	0.981	0.016	**14**	1.3	**0.61**
		sbPSO	1143	1	**1**	0	1210	23.8	*515.88
		XuPSO	35	1	0.999	0.002	170	77.3	5.8
		Wang	1154	1	0.997	0.002	1210	25.3	*403.76
Colon	0.871	uuPSO	12	0.984	**0.949**	0.013	**12**	2.2	**1.03**
		sbPSO	957	0.887	0.884	0.007	996	10.3	61.55
		XuPSO	5	0.952	0.937	0.006	65	33.2	3.74
		Wang	946	0.887	0.879	0.008	1000	22.4	56.07
Ovarian	0.846	uuPSO	10	1	**0.999**	0.003	**13**	2.2	**6.34**
		sbPSO	7591	0.877	0.874	0.001	7594	71.2	*2293.44
		XuPSO	6	1	0.997	0.005	40	33.2	42.17
		Wang	7424	0.881	0.88	0.002	7540	72.7	*1424.68
Leukemia-Golub	0.806	uuPSO	14	0.944	**0.879**	0.023	**14**	1.6	**1.54**
		sbPSO	3515	0.875	0.859	0.007	3550	47.9	*407.72
		XuPSO	100	0.875	0.844	0.012	184	146.9	12.43
		Wang	3529	0.903	‡0.874	0.015	3555	44.1	195.05

Table 2. Comparison of PSO algorithms: the accuracy (Acc) and number of feature (size) of the best model found among 100 runs is shown in best-model columns. accuracy is the number of correctly predicted samples divided by the total. The mean and standard deviation (SD) of the accuracy and number of features were estimated from the resulted models (100 in most runs). Compared to uuPSO, all model accuracies and number of features were significant using a Wilcoxon rank sum test at $p < 0.0005$ except those marked with ‡ where $p > 0.05$. time was estimated for the 100 runs. A star (*) marks times estimated from partial results where at least 20 runs have finished. This estimated time should not be largely affected since run-time standard deviation is low. For example, for the ovarian dataset the standard deviation per run was 0.138 and 0.696 hours for sbPSO and Wang PSO respectively. The accuracy using all features is shown in column "All Acc" for comparison. Xu PSO was ran at $f = 0.1, 0.3, 0.5, 0.7$, and 0.9. The best result is shown. In four datasets $f = 0.5$ was the best in terms of accuracy and number of features except for the leukemia Golub dataset where $f = 0.7$ was better. Best results are shown in bold.

4. Results

4.1 Comparison with other PSO methods

We first compared our algorithm with the sbPSO. For the uuPSO runs, we used an overall controlling proxy parameter, size =10, then b = size, u = size, e = 1, n = 0.5*size, and m = 1.5*size. The results summarized in Table 2 are encouraging. Our algorithm increased remarkably the classification accuracy in all datasets, decreased the selected features from thousands to a handful, and decreased the computation time from days to about an hour. Best model accuracy improved from around 7.3% in Leukemia-Golub to 18.1% in BC. Moreover, remarkable differences are observed in model size and computation time where our algorithm gets better results by around two orders of magnitude. Overall, these results clearly show that our algorithm is able to obtain better predictive models from functional genomics data with thousands of features in a fraction of time.

			uuPSO			***Tabu Search IBPSO***		
Dataset		#	*Genes Selected*			*Genes Selected*		
Type	**Classes**	**Genes**	**Fitness**	**#**	**%**	**Fitness**	**#**	**%**
9_Tumors	11	5726	**100.00**	149	2.6	81.63	2941	51
11_Tumors	26	12533	**100.00**	535	4.27	97.35	3206	26
14_Tumors	9	15009	**100.00**	634	4.22	74.76	8539	57
Brain_Tumor1	5	5920	**100.00**	14	0.24	95.89	2913	49
Brain_Tumor2	4	10367	**100.00**	161	1.55	92.65	5086	49
Leukemia1	3	5327	95.83	5	0.09	**100.00**	2577	48
Leukemia2	3	11225	94.44	5	0.04	**100.00**	5609	50
Lung_Cancer	5	12600	**100.00**	14	0.11	99.52	6958	55
SRBCT	4	2308	95.18	5	0.22	**100.00**	1084	47
Prostate_Cancer	2	10509	93.14	4	0.04	**95.45**	5320	51
DLBCL	2	5469	94.81	3	0.05	**100.00**	2671	49
Average			**97.58**		**1.22**	**94.30**		**17**

Table 3. Comparison between uuPSO and tabu search ibPSO: the fitness and number of feature (#) of the best model found among 100 runs is for the uuPSO algorithm while the results from tabu search ibPSO were extracted from (chuang et al., 2009). (%) shows the percentage of genes selected from the total (# genes).

The Xu PSO implementation (Xu et al., 2007) also tries to decrease the number of selected features by using a different strategy. There, the number of selected features is affected by biasing the binary activation decision using a parameter *f* (see section 2.3). For comparison, we ran the Xu algorithm (XA) for several values of *f*. We also compared the Wang algorithm (WA) that introduced the concept of updating a subset of features by an unusual implementation of a unique particle velocity whereas uuPSO use a candidate list of features. For the uuPSO runs, we used size = 5 or 10, then b = size, u = size, e = 1, n = 0.5*size, and m = 1.5*size. Results are shown in Table 2. This table shows that our implementation performed better than WA and XA in accuracy, number of selected features, and computation time. The WA algorithm is not able to decrease the number of active features presumably due to the initialization procedure. In XA, the maximum accuracy found at f = 0.5 may compete, but goodness is not sustained probably due to abrupt changes in the number of selected features. Contrary, uuPSO goodness is stable in terms of number of features along the run (lower SD). In addition, uuPSO is far quicker. Also, in order to find competitive models, we would have to perform several XA runs changing the *f* parameter.

We previously published (Martinez et al., 2010) that we obtained models with better classifications and fewer selected features than IBPSO (Chuang et al., 2008) which is another implementation of PSO. In these works, the algorithms were tested using 11 different datasets with distinct characteristics. The same authors of IBPSO presented an optimized algorithm based on tabu search and support vector machines (Chuang et al., 2009), tested with the same datasets and obtained better results than their previous version. In Table 3, we show the results running the same models found in (Martinez et al., 2010) with support vector machines and one-versus-all mode. In average (and in 6 of 11 datasets), we got higher classification accuracies than the Tabu-Search IBPSO algorithm, reinforcing the idea that is possible to choose models with few variables without sacrificing prediction power.

In summary, our algorithm is superior to four different PSO implementations in accuracy, number of features and time, supporting our objective of obtaining models with fewer genes.

4.2 Biological interpretation of features in models

In order to summarize the 100 models generated from the BC dataset, we selected the top 10 most frequent genes as a representative model (see Figure 1). The gene MMP14, a metalloprotease, has been recently related in breast cancer progression in the transition from preinvasive to invasive growth (Ma et al., 2009).This agrees with the expression of MMP14 in the BC dataset where higher values are observed in metastatic tumors. SLC27A2, a cholesterol homeostasis mediator, has been implicated in pancreatic neoplasm (Hansel et al., 2004). FMO1 is a monooxygenase involved in the NADPH-dependent oxidative metabolism of many drugs such as tamoxifen (Katchamart and Williams, 2001), which is used as breast cancer treatment in postmenopausal treatments. The expression levels of LAMB3 have been found to be increased in malignant tumors and correlated with the depth of invasion (Kita et al., 2009). However, in the BC dataset, LAMB3 seems to be less expressed in metastatic tumors. ORM1 is increased in the plasma of cancer patients (Budai et al., 2009). This concurs in the BC dataset where the expression of ORM1 appears to be increased in metastatic tumors. These results suggest that our algorithm is detecting genes related to BC physiology.

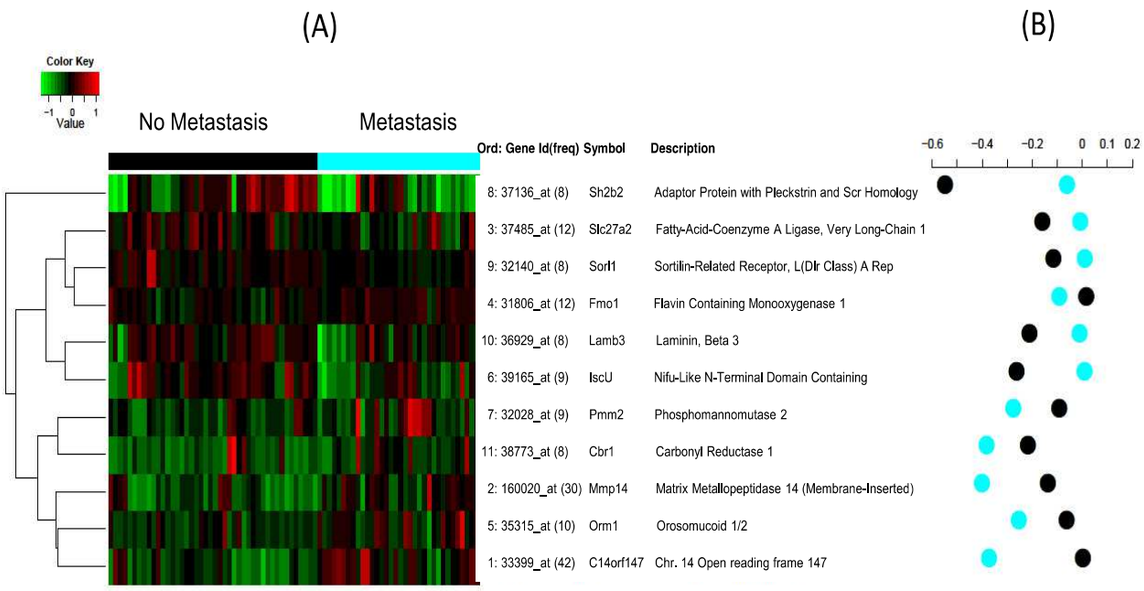


Fig. 1. Representative genes selected by our algorithm from 100 runs for the BC dataset. (A) A heatmap representation of the most frequent genes. Darker (red) or lighter (green) colors represent higher or lower expression respectively. Ord represents the order of genes given by frequency in 100 models (shown in parenthesis). (B) Class centroid for each gene. Negative values represent under-expression relative to a common reference.

4.3 Analysis of the adaptations of the *uuPSO* algorithm

Previous paragraphs have shown that our algorithm is capable of generating better and more compact models in a fraction of the time used by other methods. The next few paragraphs provides further details of the uuPSO algorithm to show the main effects of proposed adaptations. For illustrative purposes, we used only Breast Cancer and Leukemia-Yeoh datasets.

4.3.1 Overall size tests

The inspiration for our proposal is driven by the idea of decreasing the number of active features. Therefore, we compared the overall behavior attempting to control this number from 5 to 400. For this, we set size={5,10,20,100,400} as a proxy and the parameters of our adaptations were kept proportional to *size*: b=*size*, u=*size*, e=0.2**size*, n=*size*-(*size*/2), m=*size*+(*size*/2). Results are shown in Figure 2. In the BC dataset (Figure 2 A-B), overall accuracy seems more similar in late epochs than in early epochs. During the first 250 epochs, the best run achieved 84% of average accuracy using *size*=25 while the worst reached 79% using *size*=5. However, at the end of the run (epoch=1000), the former remained in 84% while the last increased to 83%. That is, it was more difficult to find 25 features than only 5. The average number of active features for the best particle slightly increased during the first 50 epochs then decreased slowly. Nevertheless, this number was very close to the initial value of size and was not limited by the limits imposed by n and m. For the LY dataset, overall accuracy was very similar (Figure 2 C-D). The number of active features also tended

to increase from the initial value of *size*. The smallest average accuracy was 95% for *size*=5 but tending for model sizes around 8, which was limited by the *m* parameter used. This suggests that the decrease in accuracy is due to the low number of features used. This is sensible because the number of classes for LY dataset is 5, so, the use of 8 features is a compromise between high accuracy at low number of features. Overall, these results show that our implementation is able to control the number of active features and support our proposal that a high number of features for accurate classification of functional genomics data is not necessary.

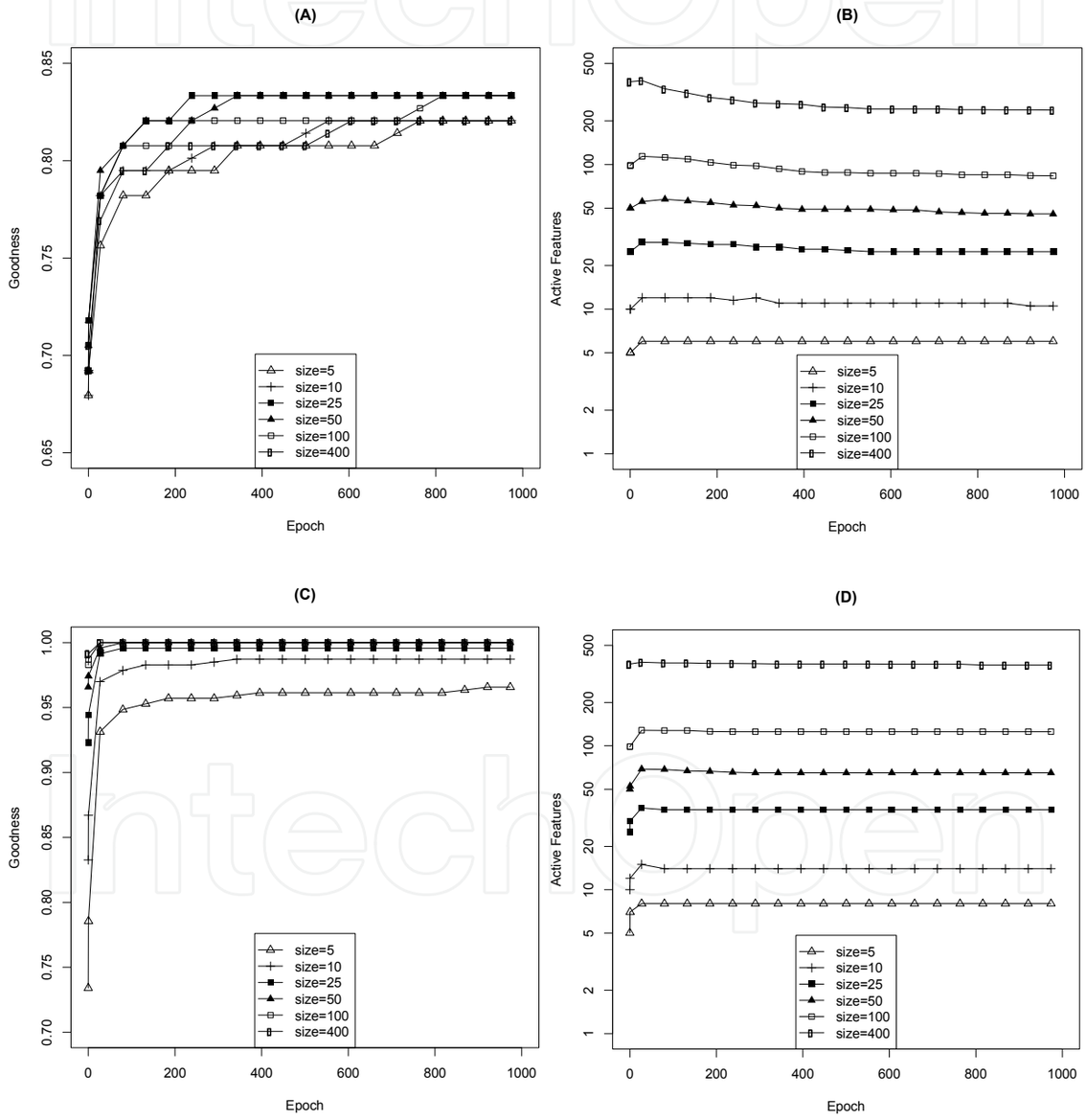


Fig. 2. Size tests for the BC (A-B) and LY (C-D) datasets. (A-C) Accuracy evolution. (B-D) Dynamics of the number of active features. Every point represents the median of the best particle in 100 runs.

4.3.2 Initialization

The runs presented above, which changed all parameters coordinately, show the overall behavior for different settings. However, the effects of each adaptation are not well appreciated. Therefore, we tested each adaptation separately. To challenge the initialization adaptation, we varied b from 5 to 400 setting b={5,10,25,50,100,400} while other parameters were kept fixed (u=10, e=1, n=2, m=600). Results are shown in Figure 3. For the BC dataset (Figure 3 A-B), accuracy was clearly smaller for b=400 within the first 50 epochs. On the other hand, for the LY dataset the accuracy was higher for larger values of b (Figure 3 C-D). For both BC and LY datasets, the final number of active features decreased when b decreased. Such decrease would be beneficial for our purposes since potential biomarker

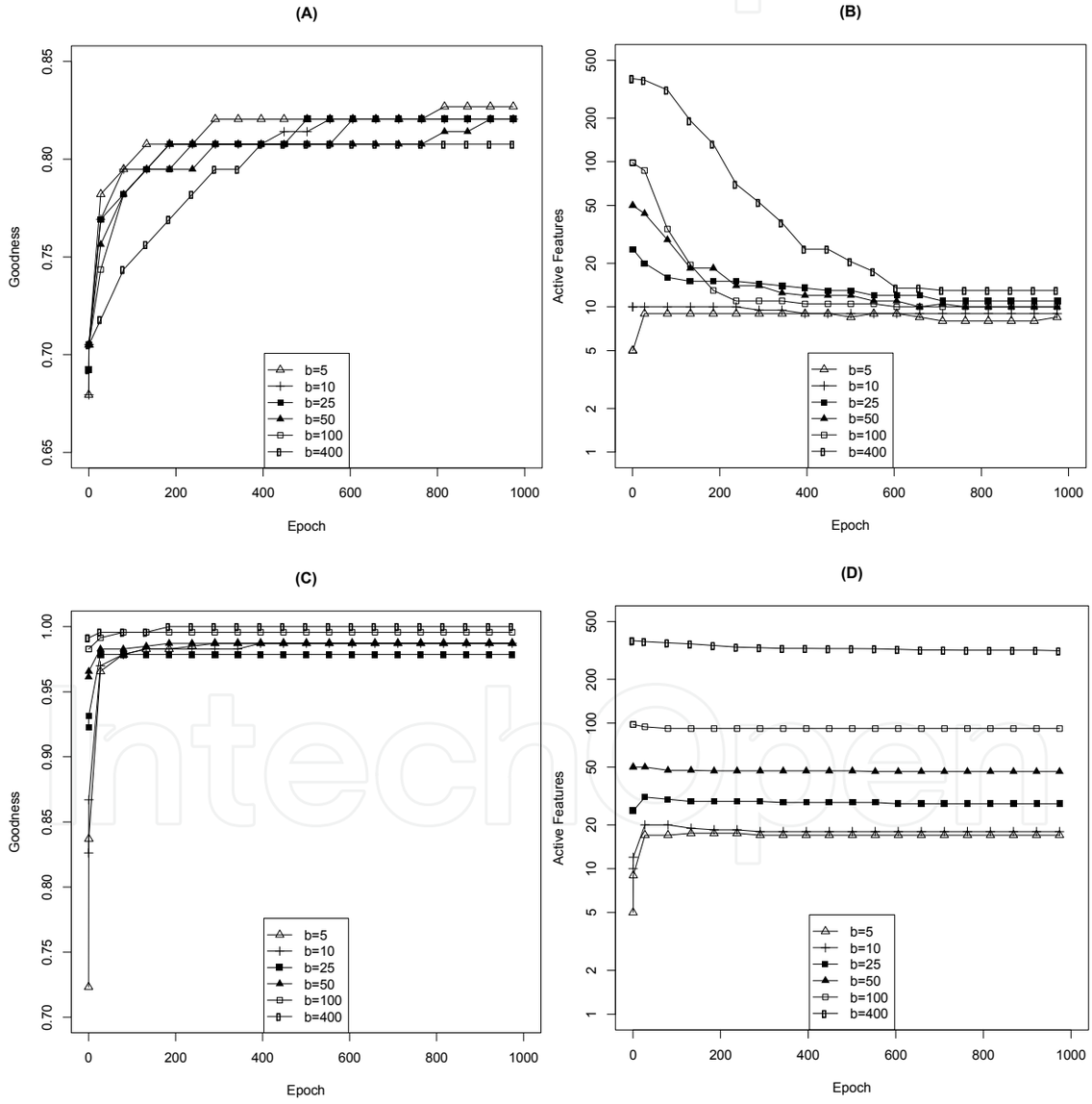


Fig. 3. Initialization tests for the BC (A-B) and LY (C-D) datasets. (A-C) Accuracy evolution. (B-D) Dynamics of the number of active features. Points as in Figure 2.

models would become more compact. Nevertheless, when initialization was smaller than update (b =5) for the BC dataset, the number of active features increases. This increase was also observed in LY for b <= 20. For the BC dataset a trend for around 10 active features is clearly observed even for b =400. In this case, it took 600 epochs to decrease the number of active features from 400 to less than 20 and to reach the same level of accuracy than in the other runs. The decrease in the number of features and the poor accuracy for runs with high b suggest that the number of active features is excessive. This may happen when the classifier is overloaded with noisy features. On the other hand, for the LY dataset, the number of active features does not decrease systematically such as that in the BC dataset. Presumably due to high accuracies in LY that are already close to 1 (the maximum possible value); thus no major evolutive pressure is present. Overall, runs tend to adjust the number of active features depending on the starting point and the accuracy pressure.

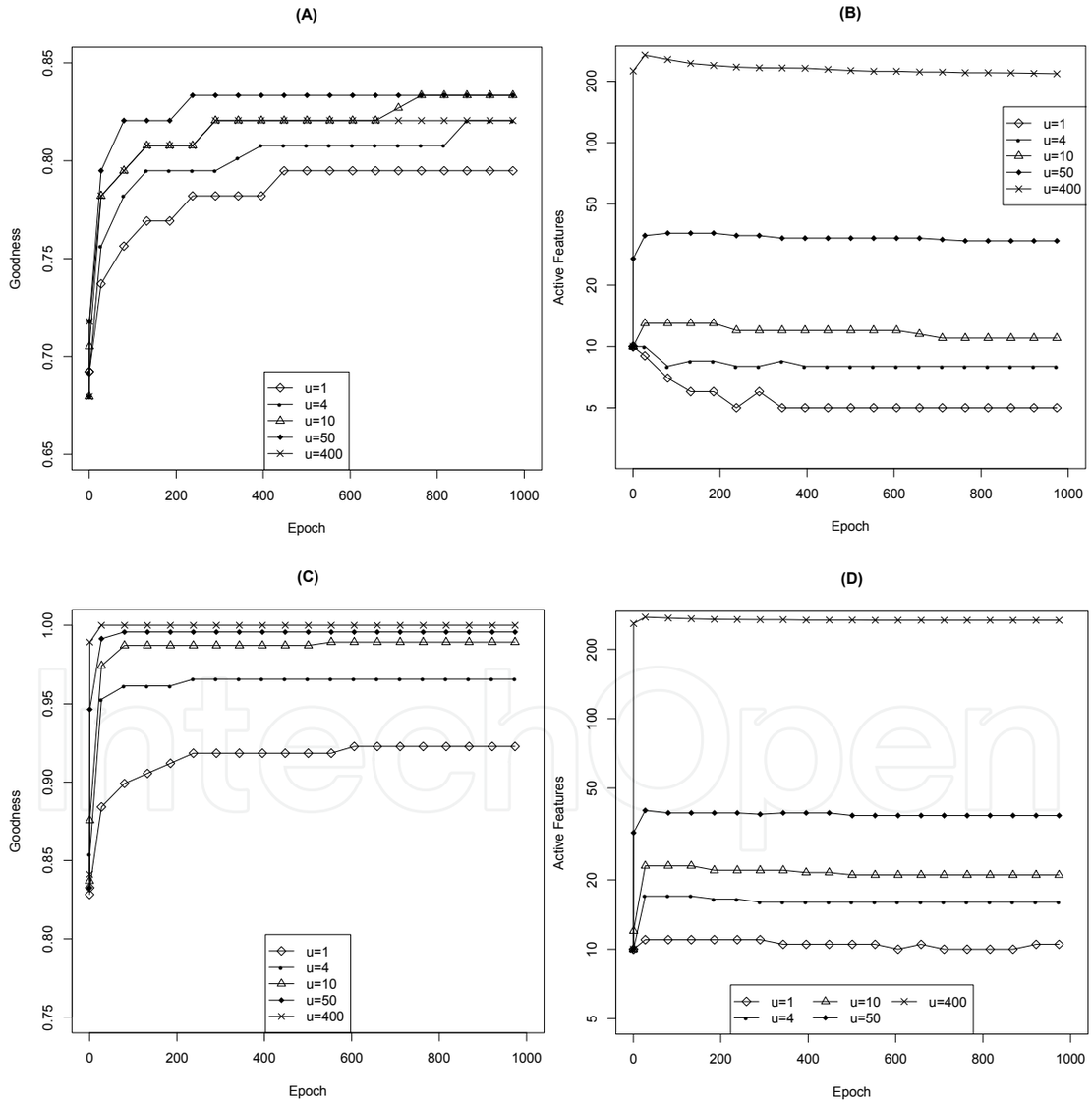


Fig. 4. Update tests for the BC (A-B) and LY (C-D) datasets. (A-C) Accuracy evolution. (B-D) Dynamics of the number of active features. Points as in Figure 2.

4.3.3 Updates

In the initialization test there was a tendency for sizes around 10 for the BC dataset, which is similar to the number of updates in those runs. This may suggest that number of active features is somehow related to updates. For this reason and to analyze the overall effect of the number of updates, we next tested the number of updates from 1 to 10 (u=1,4,10,50,400) holding all other parameters fixed (b=10, e=1, n=5, m=500). Results shown in Figure 4 propose that in general, the overall goodness increased when increasing the number of updates for low values of u in the BC and LY datasets. However for u =400, a decrement in fitness was observed in the BC dataset. This is consistent with our idea that updating a large number of dimensions is not beneficial. For the number of active features, a similar but undesirable behavior was observed in early epochs, increasing the number of updates was accompanied by an increase in the number of active features. Nevertheless, in the BC dataset, the number of active features decreased for u =1,4. Under these configurations, every particle must select 1 or 4 features respectively from at least 11 to at most 31 features composed by extreme scenarios from the set of global best, local best, its own, and random features. So, in u =1,4, deactivation was more productive than activation in the update process. This may be related to the fact that the BC is a two-classes dataset whereas LY is a five-classes one where more genes would be beneficial. Therefore, the observed results suggest that the number of updates is indirectly related to the number of active features through accuracy pressure. However, a specific number of updates may increase or decrease the number of initially active features depending on the value of the parameter and particularities of the dataset such as the number of classes.

4.3.4 Innovation

Differences between datasets were also observed when testing the number of innovations from 1 to 10 (e={1,2,4,6,8,10}; b=10, u=10, n=5, m=15). Results shown in Figure 5 for the BC and LY datasets respectively indicate that accuracies did not change drastically. However, the number of active features was sensitive to the number of innovation. For the BC dataset, the number of active features increased along with the innovation parameter. When e =10, every particle has to choose 10 updates from around 20 to 40 possible features in which 10 are new random features. Consequently, the probability of choosing new features increases, which explains the increase in active features in early epochs when the swarm is heterogeneous. For the LY dataset, the number of active features was marginally sensitive to innovation and opposite to the BC dataset. This can be explained because the LY dataset contains several features related to classes (represented by high accuracies in runs at all sizes within the first 50 epochs). In this case, new random features are not essential for the swarm because prediction is highly accurate using those already contained within the swarm. Indeed, results suggest that a large number of new random features are not beneficial and that new features are not a major driving force in the LY dataset.

In summary, there is tendency to increase the number of active features when the swarm is heterogeneous which may or may not be advantageous depending on the dataset. Innovation is a mechanism to increase heterogeneity.

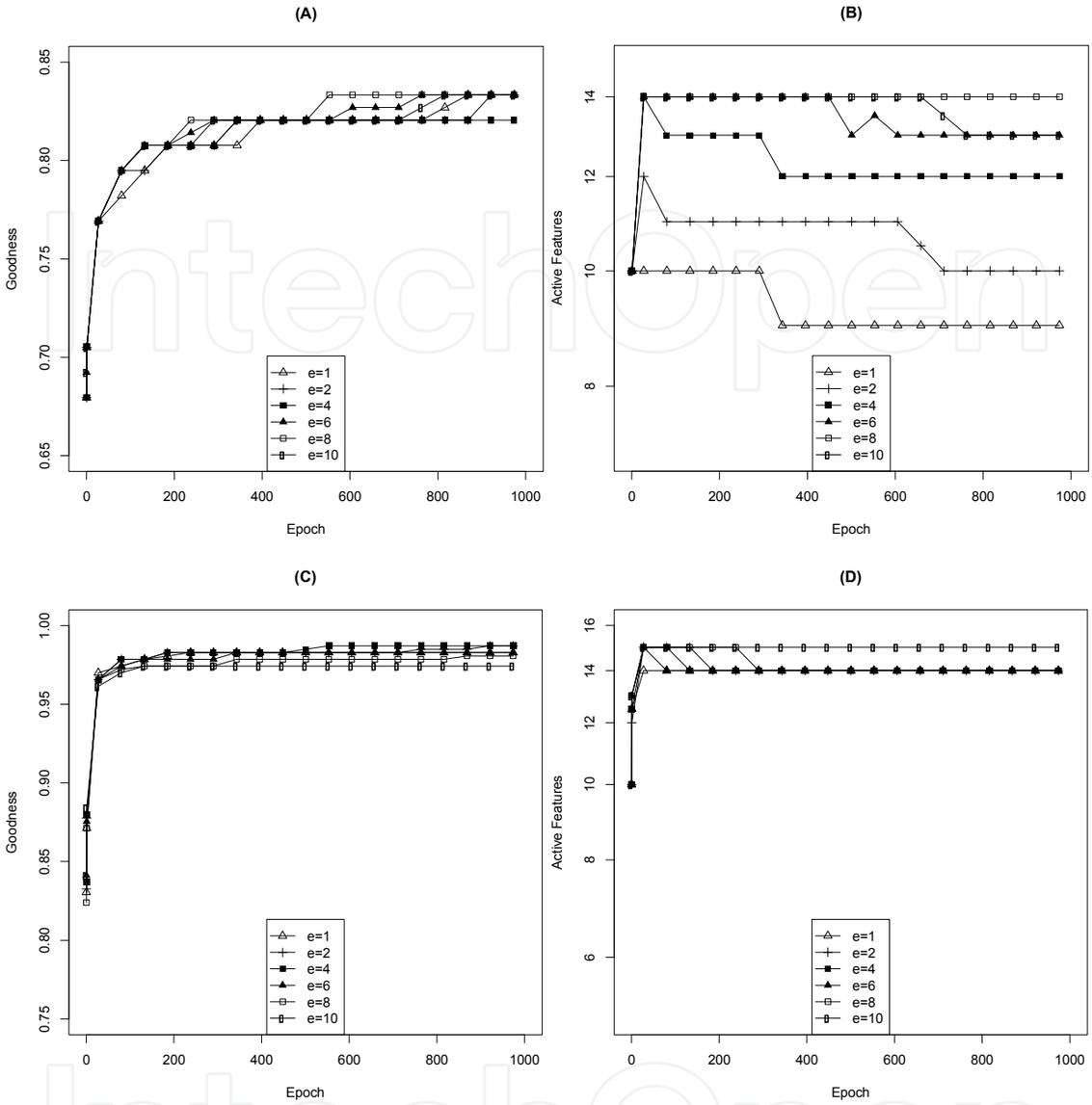


Fig. 5. Innovation tests for the BC (A-B) and LY (C-D) datasets. (A-C) Accuracy evolution. (B-D) Dynamics of the number of active features. Points as in Figure 2.

4.3.5 Constrained number of features

The last adaption was designed in order to force the number of active features to lie within a range. In this context, we would not expect major accuracy differences in changing the values of n and m. To confirm this, we conducted runs varying n =5,6,7,8,9 coordinately with m =15,14,13,12,11 respectively (using e=1, b=10, and u=10) (only ran for the BC dataset for illustrative purposes). No differences were apparent in accuracy (data not shown), but small changes were observed in the number of active features that corresponded to the limits imposed. Interestingly, a peak in the distribution of the number of active features of the final best particle was observed at the value of n in all runs as shown in Table 4. For example, for n=9 and m=11, around 60%, 30% and 10% of the best particles were generated with 9, 10,

and 11 active features respectively. This tendency toward lower values indicates that the optimal number of active features needed could be lower than 9. Considering these observations, there is a tendency for the number of active features closer to 6. A consequence of this result is that the shape of this distribution may suggest an estimation of the optimal value of the number of active features.

n	m	5	6	7	8	9	10	11	12	13	14	15
10	10						**100**					
9	11					**62**	28	10				
8	12				**41**	13	18	20	8			
7	13			**24**	13	18	15	13	15	2		
6	14		15	7	16	**18**	8	17	13	5	1	
5	15	7	8	9	18	16	**20**	8	5	4	4	1

Table 4. Distribution of the number of active features, from 5 to 15, in runs at different values of *n* and *m* for the BC dataset. The highest frequency is shown in bold.

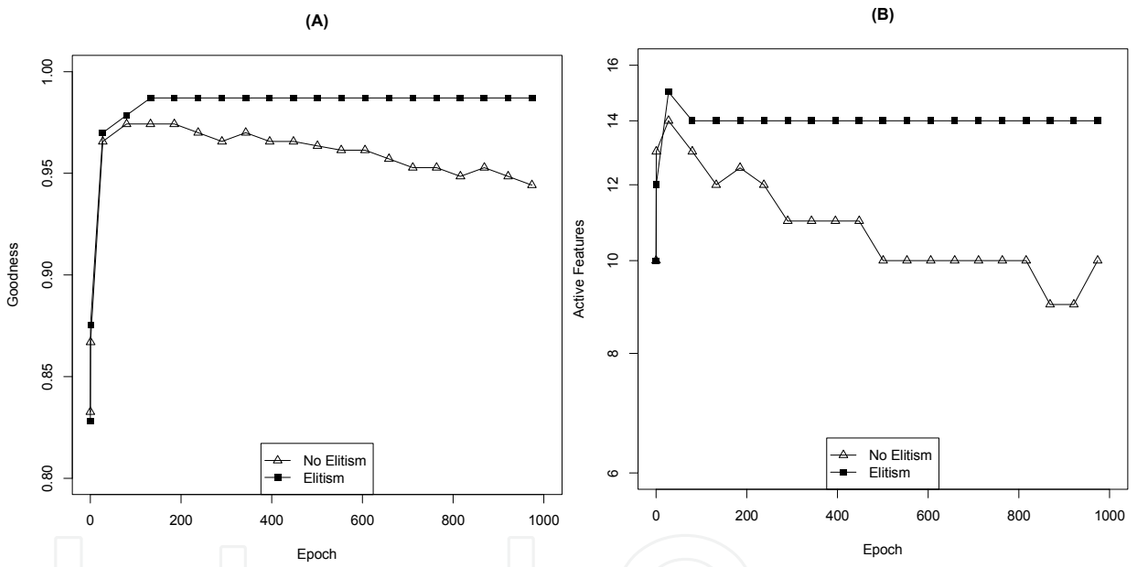


Fig. 6. Elitism tests for the LY dataset. (A) Accuracy evolution. (B) Dynamics of the number of active features. Points as in Figure 2.

4.3.6 Elitism

In the experiments described in previous paragraphs we used an elitism scheme in which the best particle is not updated. In general, it is accepted that elitism is a good practice (Eberhart et al., 2001). However, the effects of elitism are not so obvious in our implementation. Therefore we tested also the consequences of using elitism versus not using such scheme. Other parameters were set to b=10, u=10, e=1, n=5, m=15. Representative results are shown in Figure 6. As expected, the use of elitism had benefits in accuracy. On the contrary, elitism has also adverse effects by increasing the number of active features.

4.4 Simulation studies

The experiments explained in detail in 2 datasets and overall in 5 to 11 diverse datasets, clearly show that the PSO adaptations proposed here are superior to other PSO algorithms for functional genomics datasets and improved efficacy on the search of small feature subsets from large datasets. Moreover, they also exhibit better control of the number of active features than other implementations. Although our results suggest that uuPSO may be used for any other dataset, how do we know that the algorithm is actually finding good features? In this context, the biological interpretation plays a major role. However, in order to test the general applicability of our proposed algorithm, we challenged the search capabilities of uuPSO testing whether few features can be found from a large dataset mainly filled of noisy features. In such tests, simulated datasets in which data is generated by a model then injecting features carrying desired properties are commonly the choice. In this context, we generated a simulated dataset containing 2426 features for 78 samples (34 metastases, 44 non-metastases). Each feature was generated by a normal distribution with $\mu = 0$ and $\sigma = 1$. This will be referred as a "noise filled" dataset. Then we added 5 features with $\mu = 2$ and $\sigma = 1$ for metastasis sample and 5 features with $\mu = -2$ and $\sigma = 1$ for the non-metastasis samples (features 2427 to 2436). We ran uuPSO at least 1,000 times before and after adding these 10 features. Results show that $uuPSO$ is able to find the 10 added features, which is represented by better accuracies, as shown in Table 5.

Simulated Dataset	*Accuracy Mean*	*SD*	*Top 10 Selected Features (frequency %)*
Noise Filled	0.80	0.02	1878(42), 2198(11), 408(9), 1280(7), 227(7), 243(7), 1799(7), 176(7), 1461(6), 2269(6)
Noise Filled + 10 predictive	0.99	0.02	2435(15), 2430(14), 2428(14), 2433(13), 2434(12), 2431(10), 2429(10), 2436(10), 2427(9), 2432(8)

Table 5. Comparison of the simulated dataset with and without 10 predictive features.

Although this result is encouraging, simulated datasets are bound to the model used, which may generate simpler datasets that could not capture the complexity of observed datasets. Therefore, to include the original functional genomics data complexity we used the BC dataset to generate a class-unrelated dataset. We generated a dataset as negative control by removing those features that are somehow related to samples class. For this, it is sensible to think that good candidates are precisely those selected using our feature selection procedure. Consequently we ran our algorithm 100,000 times ($b = 10$, $u = 10$, $e = 10$, $n = 5$, $m = 15$) resulting in the same number of feature subsets. Then, we counted the number of times each feature was selected within the 100,000 resulted models as shown in Figure 7A. The most frequent features were then removed to generate a negative dataset (see Figure 7B). To choose a cut-off value for feature removal, we estimated the expected number of times a feature would be selected by random chance in the total number of genes selected within the 100,000 runs. This was estimated by a binomial distribution (p=1/2920, tries=1,426,097 which is the total number of features in the 100,000 models generated). The expected frequency was 524 at p-value < 0.05.

We consider that this value is very conservative since we used raw p-values uncorrected for multiple tests. The number of features whose frequency surpasses this value was 494. Consequently the negative datasets carried 2,426 features. The number of 100,000 was chosen in order to be confident that the rank of top genes would not change among different runs. Indeed ranks between two runs of 10,000 and 90,000 have no major differences (see Figure 7A inset). If our algorithm cannot find related features in this negative dataset, a decrease in the final accuracy would be expected. As projected, the mean accuracy in 1000 runs for the negative dataset was 72% (labeled with BOTTOM in Figure 7B) while the same indicator in the original dataset was 82% (labeled with FULL in Figure 8B). Then, we inserted only the original top 10 features to the negative dataset (representing 0.4% of the 2,436 total features) and counted how many times the same top 10 features were found in the best models for 1000 runs. We also performed this procedure for top 20, 50, 100, 200, 300, 400 and 494 originally removed features. Results are shown in Figure 8A. We found that the high accuracy is easily restored after the first insertion (labeled TOP10 in Figure 8B). However, results show that the next 10 features were needed to fully restore the original observed accuracy distribution (labeled TOP20 in Figure 8B). We found all 10 and 20 top inserted features and more than 90% among the top 100 injected features. These results indicate that most frequent features (and presumably those more related to classes) are easily found even in highly noisy scenarios (99.6% of noisy features). Nevertheless, search efficiency was decreased to 60% when 494 top features were added. Within this, top 50 features were always found. Therefore, the search capability was uncertain only for those features ranked in the order of hundreds, presumably, because of random effects.

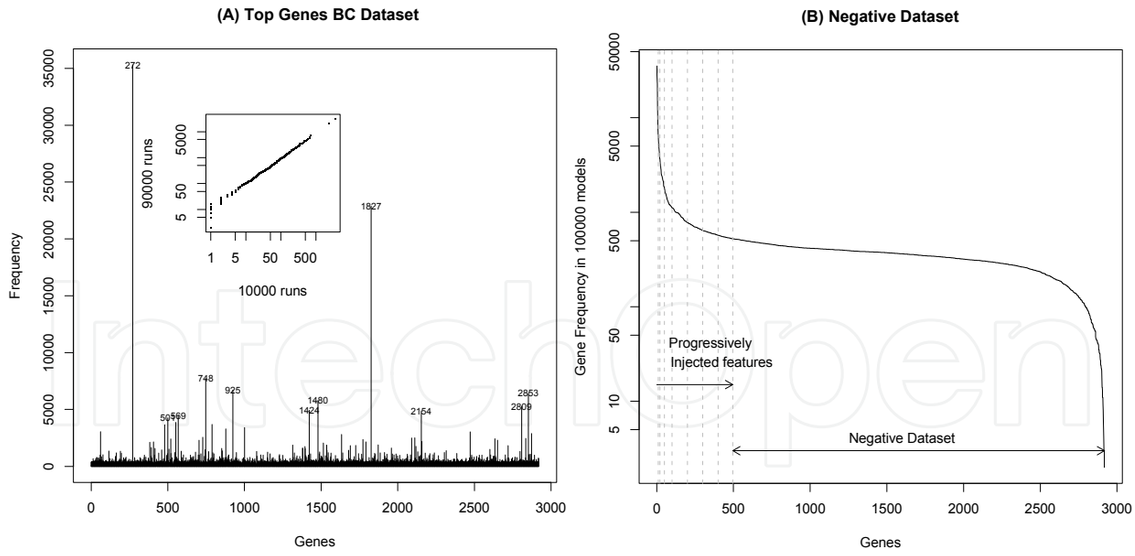


Fig. 7. (A) Top genes found in 100,000 runs for the BC dataset. Inset: Comparison of feature selection frequency for 10,000 and 90,000 runs. (B) Generation of the negative dataset and positive added features. Features were sorted by the number of times they were found in the best particle for 100,000 runs. Top 494 were removed to generate the negative dataset. Top features were then injected progressively to the negative dataset: first 10, then 20, 50, 100, 200, 300, 400 and 494 (from leftmost dashed line to rightmost dashed line).

We then asked about the amount of information that uuPSO is able to find in the original dataset. That is, the improvement attributed to the uuPSO. For this, we generated random models of equal sizes than those observed in the models generated for the negative dataset. These models would represent the amount of information (given by accuracy) found by random chance. Similarly, we generated series of 20 random models recording only the best in each series. These would represent the best models that drive the search at the beginning of the swarm optimization procedure. Results show that the baseline for the amount of information is about 54% given by random models in the negative dataset (labeled with Random in Figure 9B). The accuracy distribution from the best of the random series indicates that uuPSO starts with 66% accuracy (labeled with Best Random in Figure 8B), which is then improved to 72% during the uuPSO process (labeled with BOTTOM in Figure 8B). Overall, these results show that the average improvement by uuPSO in the BC dataset is 16% (from 66% to 82%) when good predictors are present.

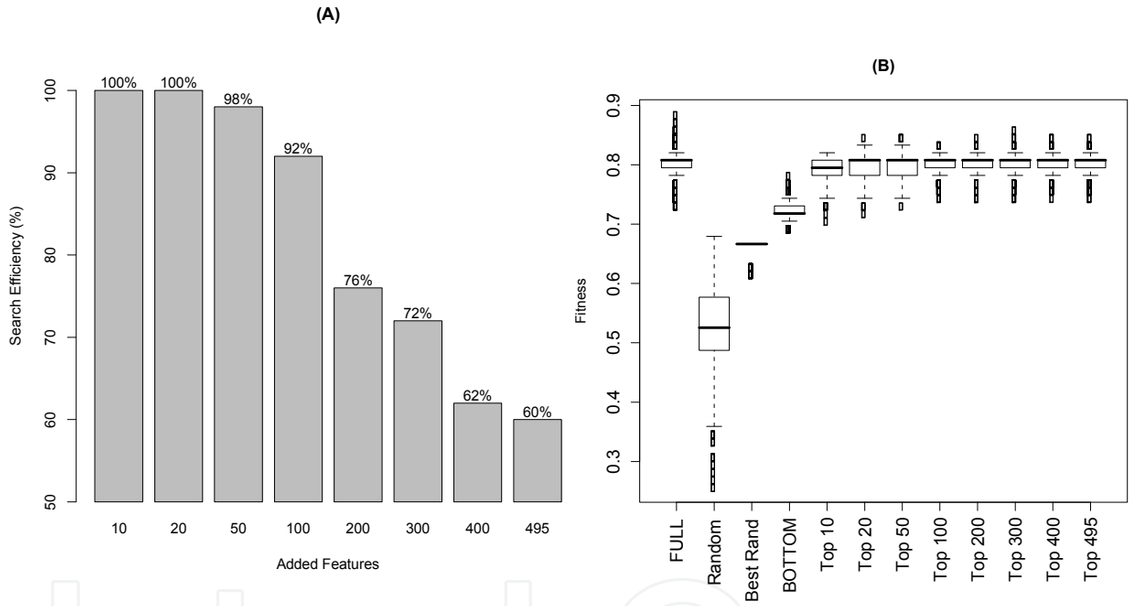


Fig. 8. (A) Search efficiency in the top added features. (B) Fitness distribution for uuPSO search capability tests.

4.5 Comparison with other feature selection methods and classifiers

In order to show that the uuPSO methodology is able to find alternative models and that is not dependent to the nearest centroid classifier (NC) used here, we also used a state-of-the-art support vector machine (SVM) classifier with a linear kernel ((Chang and Lin, 2001), java implementation). The results shown in Table 6 clearly demonstrate that uuPSO also finds good predictive models using a SVM classifier.

Although the differences are marginal relative to the NC classifier, the accuracy of SVM models was superior in all databases. Moreover, the models found using SVM were

generally smaller. These differences are consistent with the perception that SVM is a powerful classifier. However, we note a huge difference in processing time, which should correspond to the complex training procedure of SVM, to the java implementation used, or to our coupling implementation of this library. In the Ovarian dataset, uuPSO using SVM found models with 100% accuracy in early epochs explaining the small difference in processing time for this dataset.

Database	*Classifier*	*Best-model*		*Accuracy*		*Features*		*Time*
		Size	*Acc*	*Mean*	*SD*	*Mean*	*SD*	*(hr)*
Breast Cancer	NC	13	0.923	0.842	0.03	13	2	0.7
	SVM	12	0.974	**0.904**	0.03	13	1	*862.0
Leukemia-Yeoh	NC	14	1	0.981	0.02	14	1	0.6
	SVM	11	1	**0.996**	0.01	14	2	*954.3
Colon	NC	12	0.984	0.949	0.01	12	2	1
	SVM	8	1	**0.987**	0.02	13	2	*585.7
Ovarian	NC	10	1	0.999	0	13	2	6.3
	SVM	10	1	**1**	0	15	2	6.9
Leukemia-Golub	NC	14	0.944	0.879	0.02	14	2	1.5
	SVM	15	0.958	**0.927**	0.02	12	2	*5722.7

Table 6. Comparison of uuPSO algorithm using the nearest centroid (NC) and support vector machine (SVM) classifiers. The goodness (Accuracy) and number of features from the best model found among 100 runs is shown. Goodnesses and features mean and standard deviation (SD) were estimated along 1000 epochs and 100 runs. Time was estimated for 100 runs. A star (*) marks times estimated from partial results.

To show the utility of uuPSO as a competitive feature selection alternative, we compared uuPSO with a state-of-the-art backward elimination strategy such as SVM-Recursive Feature Elimination (Guyon et al., 2002) and a Forward-Selection strategy (Tibshirani et al., 2002; Montaner et al., 2006). The former (SVM-RFE) considers all features and remove the worst feature in each cycle whereas the last (SVM-FS) considers the best feature and includes the next ranked feature in each cycle (based in univariate ranking such as t-test or f-test). The results of the comparison summarized in Table 7 suggest that models generated by uuPSO are competitive. However, there are some differences in the comparison of SVM-RFE and SVM-FS with uuPSO. SVM-RFE and SVM-FS procedures "see" all data to create a rank of the importance of the features related to the problem irrespective of the 10-fold-CV used to estimate the error in the feature selection step. A consequence is that the estimated error is more optimistic than the estimation of others cross-validation strategies. In our experiments using uuPSO, the SVM classifier never is aware of all data. In each of the 10 cycles of the 10-fold-CV, the SVM classifier sees only the training set and makes prediction of the test set. To observe a possible effect of this issue, we made a simple experiment in SVM-RFE changing the estimation of the rank using all data by an average

rank generated by SVM-RFE from each of the 10 folds used in the cross-validation. The results, shown in Table 7 column SVM-RFE-CV, demonstrate that uuPSO is sometimes better generating smaller models.

Database	*uuPSO NC*		*uuPSO SVM*		*SVM RFE*		*SVM FS*		*SVM RFE-CV*	
	Acc	*Size*	*Acc*	*Size*	*Acc*	*Size*	*Acc*	*Size*	*Acc*	*Size*
Breast Cancer	0.92	13	0.97	12	1	13	0.83	9	1	50
Leukemia-Yeoh	1	14	1	11	1	8	1	51	1	11
Colon	0.98	12	1	8	1	9	0.89	4	1	27
Ovarian	1	10	1	10	1	3	1	34	1	21
Leukemia-Golub	0.94	14	0.96	15	1	12	0.86	9	1	13

Table 7. Comparison of uuPSO algorithm using the nearest centroid (NC) and support vector machine (SVM) classifiers versus support vector machine recursive feature elimination algorithm (SVM-RFE) and support vector machine forward selection algorithm (SVM-FS).

A more important difference of uuPSO compared to SVM-RFE and SVM-FS is the potential to generate several versions of good classifiers using a different set of features. In SVM-RFE and SVM-FS strategies, the pre-computed rank is fixed in the feature selection procedure. Therefore, alternative models are almost identical. Contrary, given the random nature of PSO, uuPSO is able to generate unrelated models with the same or similar predictive power (see Table 8 for the features selected in the top two models). For genomics data, this is useful in research and clinical scenarios to investigate the biological mechanism of the genes involved and to select models based on biological knowledge, already installed infrastructure, commercial availability, or patent protected issues. Indeed, as seen in Table 8,

Dataset	*Method*	*Accuracy*	*Selected Features*
Breast Cancer	uuPSO NC	0.923	78,116,329,672,745,760,835,925,2023,2064,2721,2847, 2871
		0.923	222,272,286,856,965,1204,1396,1480,1509,1610,2145, 2172,2422,2910
	uuPSO SVM	0.974	303,393,542,1286,1291,1370,2047,2172,2176,2721,2873, 2919
		0.974	18,78,832,1254,1515,1644,1965,2092,2185,2187,2367, 2462
	SVM RFE	1	2873,272,1002,706,268,2137,521,705,879,1695,1845, 1387,979
		1	2873,272,1002,706,268,2137,521,705,879,1695,1845, 1387,979,1793

Dataset	*Method*	*Accuracy*	*Selected Features*
Leukemia Yeoh	uuPSO NC	1	695,698,781,913,950,1069,1094,1275,1432,1609,1665, 1702,1913,2155
		0.996	496,754,928,942,1175,1188,1281,1287,1519,1589,2287
	uuPSO SVM	1	51,523,934,995,1029,1153,1370,1441,1590,1969,2333
		1	15,151,422,429,708,968,1193,1220,1313,1340,1699, 2007
	SVM RFE	1	708,2231,399,1139,1193,1588,781,1420
		1	708,2231,399,1139,1193,1588,781,1420,2118
Leukemia Golub	uuPSO NC	0.944	36,206,321,1830,2238,2685,3020,3239,3411,3621,4717,4 915,6451,6643
		0.917	259,514,518,1291,2125,2396,2403,2912,3367,3934,4735, 6352,6553
	uuPSO SVM	0.958	66,529,2610,2621,2644,3316,4377,4558,5147,5348,5751, 5897,6050,6541,6577
		0.944	20,811,1332,1882,1887,2293,2860,2890,4319,6024, 6761
	SVM RFE	1	997,1868,306,951,6876,6024,2520,6345,3559,5492,1650, 1805
		1	997,1868,306,951,6876,6024,2520,6345,3559,5492,1650, 1805,3036
Colon	uuPSO NC	0.984	182,234,549,566,613,738,788,880,1060,1210,1493, 1549
		0.968	137,508,739,924,1110,1597,1698
	uuPSO SVM	1	163,380,496,764,787,1065,1324,1796
		1	19,399,440,450,611,689,1146,1679,1843,1960
	SVM RFE	1	1924,1482,1649,1843,1668,788,1935,1597,1221,124, 1895,1976,1671
		1	1924,1482,1649,1843,1668,788,1935,1597,1221,124, 1895,1976,1671,1094,1325
Ovarian	uuPSO NC	1	2192,2236,3258,5036,7552,8062,13207,13612,13871, 15058
		1	747,2238,2633,3174,3224,3370,3561,5226,10968, 12915
	uuPSO SVM	1	2235,2493,4098,4642,5808,6494,7235,12560,14006, 14086
		1	1039,1682,2169,2235,2834,3703,8285,8556,11785, 14707,15053
	SVM RFE	1	2239,2240,1681
		1	2239,2240,1681,2527

Table 8. Feature content of the best two models selected by three methods.

models generated by uuPSO using SVM are not similar to those generated by SVM-RFE. This suggests that uuPSO is able to explore other possibilities that cannot be explored with SVM-RFE. Therefore, uuPSO can also be used in conjunction to other methods for feature selection. In addition, uuPSO may use any classifier that can generate competitive models with different gene content. All this indicates that uuPSO is a valuable tool for the computational biology community.

5. Discussion

We have proposed and studied some the uuPSO algorithm for feature selection that successfully select small feature subsets from large-scale datasets. The selection of few highly predictive features is important in diagnosis for biological and clinical fields where an experiment has to be done for each feature. Thus, a reduced number of features would be simpler, cheaper, and easier to understand and interpret. Other fields would also prefer small models than large ones to avoid overwhelming and perhaps overfitting. We have shown in several datasets containing thousands of features than our algorithm is superior in terms of accuracy, lower number of selected features, and processing time compared to others PSO algorithms. We have also shown that designed models are meaningful biologically. In addition, uuPSO can be seen as a generalization of PSO where sbPSO is a special case (for k = total number of features, $b = k / 2$; $u = k$; $e = 0$; $n = 1$; $m = k$).

Our novel combination of subset initialization, subset update, and number of features, namely under-updated PSO, was critical in the successful selection of small feature subsets. We found that the number of updates is very close to the total active number of features. Therefore, we believe that the update-all process is the main reason why sbPSO fails for large datasets. Presumably, because a high number of features introduce such high levels of noise that classification is confused resulting in poor prediction..

Our constrained updates adaption is somehow similar to that proposed by (Wang et al., 2007) in which the strategy was to update a subset of dimensions. However, in our approach, the choice of the specific subset to update is different and is based on dissimilar rationale. Wang used a unique velocity related to the number of bits allowed to change and does not consider the effect of the local best particle. In addition, Wang et al. initializes all features as in the usual PSO activating the half of the features by random chance. This may explain the poor performance of the Wang et al. procedure for datasets with thousands of features. Our vision to update a subset of particles is completely different. Here, contrary to Wang et al., we envisioned the number and the selection of updates as a constrained imitation in which not all features can be imitated at the same time. In this framework, our results indicate that the goodness increase when increasing the number of updates, until a certain limit. So, there are an optimal number of variables to update. In addition, the initialization plays an important role. Although uuPSO may recover from a high number of initial active features, a mild initialization may be more useful. We have shown that the proposed adaptations are successful and relevant in the context of small feature subset selection. Also our PSO implementation seems to be better than others.

One problem in our implementation is that it is necessary to start with a guess of the optimal number of features. However, which is the optimal number of features? So far we attempted to select small feature subsets of size *k*. Ideally, we would like that *k* lies between 5 and 15. Based in the results shown here, this goal is achievable. However this is far from being general and could correspond to dataset dependency. Thus, to analyze a dataset for the first time, we would suggest to use a conservative base number such as 5, 10 or 20 depending on the resulted goodness and the number of classes.

From the computational size, the superb performance in processing time may allow us to run systematic selections for several datasets and perhaps design a web service open to the scientific community. We will work in this direction shortly. Our algorithm should run faster and proportionally to the total number of features than the plain sbPSO. This is relevant since the future of microarray data analysis points out to the implementation of meta-analysis, where the number of samples are increased, thus more processing will be needed.

From the experiments performed with simulated datasets, our results clearly show that our algorithm is capable of finding the most differentially expressed genes. Table 5 supports our claim since the top 10 selected features after adding the predictive variables were precisely the variables with the best predictive power (features 2427 to 2436) and also with almost perfect classification accuracy. The search capability tests suggest that uuPSO is powerful to find important features even in the presence of 99.6% of noisy features. We showed that top 50 features are easily discovered among 1000 runs and that the search capability began to decrease for more than the top 100 genes. However, for our purpose of obtaining models composed from a few number of genes, we think that the search capability is good enough without losing information.

For illustrative purposes, we used 10-fold-cross-validation procedure considering all available data. However, it is recommended to use a blind subset to test models after the feature selection process. For the LY dataset, if the data is split in 66% for training and 33% for blind test, the accuracy of the representative model is 100% when evaluated in the blind set. This is equal to the accuracy obtained using all data. If the same procedure is performed in the BC dataset, the accuracy of the representative model is 66%. This discrepancy can be attributed to differences in the number of samples and the clinical definitions of classes. The BC dataset contains only 78 samples whereas the LY dataset contains 233. In addition, the classes in LY dataset were based on chromosomal rearrangements (Yeoh et al., 2002) whereas in the BC dataset were based on a progression end-point indicator which is biologically more complex. For example, the no metastasis class was determined by the fact that no metastases were present during the following 5 years. However, it is unknown whether some of these samples would develop metastases in the following years. Therefore, the particular molecular state of samples in the no metastasis class can be more heterogeneous than the classes in the LY dataset. Despite these concerns, the most frequent genes are conserved in both training schemes and the genes found shown in Figure 1 seem to be related to BC. For biomarker design purposes, we recommend to use large enough datasets that are representative of the biological phenomena and that can be used for proper train-test schemes.

We tested two classifiers (nearest centroid and SVM) to guide the search of features with successful results. This suggest that the use of uuPSO can be extended to other applications such as regression, time to event (survival analysis), or other custom associations between features and outcomes by using a suitable goodness function. We used the classification accuracy to make selection pressure for smaller models. However, the goodness function can also be used to explicitly select fewer features (Alba et al., 2007). For example, using goodness = accuracy * w_1 + size * w_2. Setting w_2 to negative values would tend to select particles having smaller models. However, a problem arises when more accurate models are replaced by worse ones that are smaller. Nevertheless, incipient results suggest that improvements are only marginal and may depend on others factors such as number of epochs (data not shown).

Another advantage from our method is that it can provide models with distinct variables but with the same level of accuracy. This can be useful for biologists or medical doctors, because if they have more options of possible genes to study with techniques like PCR, they can select the genes according to other criteria like molecular function, pathways, etc.

6. Conclusions

We proposed an adapted sbPSO algorithm, named uuPSO, for feature selection that allow the selection of small feature subsets from large-scale datasets. All these adaptations have a significant impact in searching small and accurate multivariate feature subsets. Results showed that good subsets were composed of around 5-20 features. These subsets performed well in classification tasks previously shown to be difficult. The proposed algorithm was successful in selecting small feature subsets from five large functional genomics datasets. Biology and simulation results confirm that our algorithm is able to find features related to sample classes. We also showed that the algorithm is able to find important features even in the presence of 99.6% of noisy features. Comparisons with other methods show that uuPSO is able to find competitive models that could not be found with SVM-RFE or SVM-FS. Therefore, uuPSO can be used in addition to these or other feature selection methods. We also tested uuPSO coupled with two classifiers (SVM and nearest centroid) and show that both can find competitive models with different selected features but same accuracy. Consequently, uuPSO could be a framework and a valuable tool in computational biology for biomarker design.

7. Acknowledgements

We would like to thank Arturo Berrones for his helpful comments, advices and criticism, which had help to improve this work. We also appreciate the manuscript corrections kindly suggested by David Mathieson, Amalia Cuellar, Gabriela Ruiz, Jose Tamez, Lucio Florez, and Sergio Martinez. This research was supported by Catedras de Investigacion grant CAT172 from ITESM Campus Monterrey and by the CONACyT grant 083929.

8. References

Alba, E., García-Nieto, J., Jourdan, L., and Talbi, E. (2007). Gene selection in cancer classification using pso/svm and ga/svm hybrid algorithms. *In IEEE Congress on Evolutionary Computation*, 284-290.

Alon, U., Barkai, N., Notterman, D. A., Gish, K., Ybarra, S., Mack, D., and Levine, A. J. (1999). Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays. *Proceedings of the National Academy of Sciences of the United States of America* 96, 6745-6750.

Bello, R., Gomez, Y., Nowe, A., and Garcia, M. (2007). Two-step particle swarm optimization to solve the feature selection problem. In *ISDA '07: Proceedings of the Seventh International Conference on Intelligent Systems Design and Applications*, 691-696. IEEE Computer Society, Washington, DC, USA.

Bolstad, B. M., Irizarry, R. A., Astrand, M., and Speed, T. P. (2003). A comparison of normalization methods for high density oligonucleotide array data based on variance and bias. *Bioinformatics* 19, 185-193.

Budai, L., Ozohanics, O., Ludnyi, K., Drahos, L., Kremmer, T., Krenyacz, J., and Vkey, K. (2009). Investigation of genetic variants of ?-1 acid glycoprotein by ultra-performance liquid chromatography mass spectrometry. *Analytical and Bioanalytical Chemistry* 393, 991-998.

Carzaniga, T., Sarti, D., Trevino, V., Buckley, C., Salmon, M., Wild, D., Deh, G., and Falciani, F. (2007). Microarrays Technology through Applications. In: *The analysis of Cellular Transcriptional Response at the Genome Level: Two Case Studies with Relevance in Bacterial Pathogenesis*, 125-154. Taylor & Francis Group.

Chang, C. and Lin, C. (2001). LIBSVM: a library for support vector machines. Software available at http://www.csie.ntu.edu.tw/cjlin/libsvm.

Chuang, L., Chang, H., Tu, C., and Yang, C. (2008). Improved binary PSO for feature selection using gene expression data. *Computational Biology and Chemistry* 32, 29-38.

Chuang, L.-Y., Yang, Cheng-Huei, & Yang, Cheng-Hong. (2009). Tabu search and binary particle swarm optimization for feature selection using microarray data. *Journal of computational biology*, 16(12), 1689-1703.

Eberhart, R., Shi, Y., and Kennedy, J. (2001). *Swarm Intelligence*. The Morgan Kaufmann Series in Artificial Intelligence. Morgan Kaufmann, San Francisco, CA, USA.

Golub, T. R., Slonim, D. K., Tamayo, P., Huard, C., Gaasenbeek, M., Mesirov, J. P., Coller, H., Loh, M. L., Downing, J. R., Caligiuri, M. A., Bloomfield, C. D., and Lander, E. S. (1999). Molecular classification of cancer: Class discovery and class prediction by gene expression monitoring. *Science* 286, 531-537.

Guyon, I. and Elisseeff, A. (2003). An introduction to variable and feature selection. *Journal of Machine Learning Research* 3, 1157-1182.

Guyon, I., Weston, J., Barnhill, S., and Vapnik, V. (2002). Gene selection for cancer classification using support vector machines. *Machine Learning* 46, 389-422.

Hansel, D., Rahman, A., House, M., Ashfaq, R., Berg, K., Yeo, C., and Maitra, A. (2004). Met proto-oncogene and insulin-like growth factor binding protein 3 overexpression correlates with metastatic ability in well-differentiated pancreatic endocrine neoplasms. *Clinical Cancer Research* 10, 6152-6158.

Hieter, P. and Boguski, M. (1997). Functional genomics: It's all how you read it. *Science* 278, 601-602.

Katchamart, S. and Williams, D. E. (2001). Indole-3-carbinol modulation of hepatic monooxygenases cyp1a1, cyp1a2 and fmo1 in guinea pig, mouse and rabbit. *Comparative Biochemistry and Physiology Part C: Toxicology & Pharmacology* 129, 377-384.

Kita, Y., Mimori, K., Tanaka, F., Matsumoto, T., Haraguchi, N., Ishikawa, K., Matsuzaki, S., Fukuyoshi, Y., Inoue, H., Natsugoe, S., Aikou, T., and Mori, M. (2009). Clinical significance of lamb3 and col7a1 mRNA in esophageal squamous cell carcinoma. *European Journal of Surgical Oncology* (EJSO) 35, 52-58.

Ma, X., Dahiya, S., Richardson, E., Erlander, M., and Sgroi, D. (2009). Gene expression profiling of the tumor microenvironment during breast cancer progression. *Breast Cancer Research* 11, R7.

Martinez, E., Alvarez, M. M., and Trevino, V. (2010). Compact cancer biomarkers discovery using a swarm intelligence feature selection algorithm. *Computational Biology and Chemistry* 34, 244-250.

Montaner, D., Trraga, J., J.Huerta-Cepas, Burguet, J., Vaquerizas, J., Conde, L., Minguez, P., Vera, J., Mukherjee, S., Valls, J., Pujana, M., Alloza, E., Herrero, J., Al-Shahrour, F., and Dopazo, J. (2006). Next station in microarray data analysis: Gepas. *Nucleic Acids Research* 34, W486_W491. URL http://dx.doi.org/10.1093/nar/gkl197.

Neumann, J., Schnrr, C., and Steidl, G. (2005). Combined svm-based feature selection and classification. *Machine Learning* 61, 129-150.

Petricoin, E., Ardekani, A., Hitt, B., Levine, P., Fusaro, V., Steinberg, S., Mills, G., Simone, C., Fishman, D., Kohn, E., and Liotta, L. (2002). Use of proteomic patterns in serum to identify ovarian cancer. *Lancet* 359, 572-577. URL http://dx.doi.org/10.1016/S0140-6736(02)07746-2.

Saeys, Y., Inza, I., and Larrañaga, P. (2007). A review of feature selection techniques in bioinformatics. *Bioinformatics* 23, 2507-2517.

Sauer, S., Lange, B., Gobom, J., Nyarsik, L., Seitz, H., and Lehrach, H. (2005). Miniaturization in functional genomics and proteomics. *Nature Reviews Genetics* 6, 465-476.

Shen, K., Ong, C., Li, X., and Wilder-Smith, E. (2008). Feature selection via sensitivity analysis of svm probabilistic outputs. *Machine Learning* 70, 1-20.

Takahashi Monteiro, S. and Yukio, K. (2007). Applying particle swarm intelligence for feature selection of spectral imagery. In *ISDA '07: Proceedings of the Seventh International Conference on Intelligent Systems Design and Applications*, 933-938. IEEE Computer Society, Washington, DC, USA.

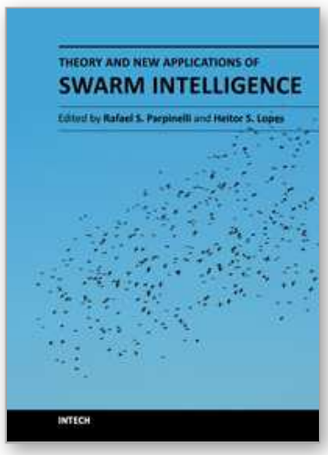
Tibshirani, R., Hastie, T., Narasimhan, B., and G.Chu (2002). Diagnosis of multiple cancer types by shrunken centroids of gene expression. *Proceedings of the National Academy of Science* USA 99, 6567-6572.

Trevino, V. and Falciani, F. (2006). Galgo: an r package for multivariate variable selection using genetic algorithms. *Bioinformatics* 22, 1154-1156.

van 't Veer, L., Dai, H., van de Vijver, M., He, Y., Hart, A., Mao, M., Peterse, H., van der Kooy, K., Marton, M., Witteveen, A., Schreiber, G., Kerkhoven, R., Roberts, C., Linsley, P., Bernards, R., and Friend, S. (2002). Gene expression profiling predicts clinical outcome of breast cancer. *Nature* 415, 530-536.

Vapnik, V., 1998. Statistical Learning Theory. Wiley-Interscience.

Wang, X., Yang, J., Teng, X., Xia, W., and Jensen, R. (2007). Feature selection based on rough sets and particle swarm optimization. *Pattern Recognition Letters* 28, 459-471.



Theory and New Applications of Swarm Intelligence

Edited by Dr. Rafael Parpinelli

ISBN 978-953-51-0364-6
Hard cover, 194 pages
Publisher InTech
Published online 16, March, 2012
Published in print edition March, 2012

The field of research that studies the emergent collective intelligence of self-organized and decentralized simple agents is referred to as Swarm Intelligence. It is based on social behavior that can be observed in nature, such as flocks of birds, fish schools and bee hives, where a number of individuals with limited capabilities are able to come to intelligent solutions for complex problems. The computer science community have already learned about the importance of emergent behaviors for complex problem solving. Hence, this book presents some recent advances on Swarm Intelligence, specially on new swarm-based optimization methods and hybrid algorithms for several applications. The content of this book allows the reader to know more both theoretical and technical aspects and applications of Swarm Intelligence.

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen