Objective Reduction Using a Feature Selection Technique

Antonio López Jaimes
CINVESTAV-IPN
Evolutionary Computation Group (EVOCINV)
Departamento de Computación
Av. IPN No. 2508
Col. San Pedro Zacatenco
México, D.F. 07360
tonio.jaimes@gmail.com

Carlos A. Coello Coello
CINVESTAV-IPN
Evolutionary Computation Group (EVOCINV)
Departamento de Computación
Av. IPN No. 2508
Col. San Pedro Zacatenco
México, D.F. 07360
ccoello@cs.cinvestav.mx

Debrup Chakraborty
CINVESTAV-IPN
Departamento de Computación
Av. IPN No. 2508
Col. San Pedro Zacatenco
México, D.F. 07360
debrup@cs.cinvestav.mx

ABSTRACT

This paper introduces two new algorithms to reduce the number of objectives in a multiobjective problem by identifying the most conflicting objectives. The proposed algorithms are based on a feature selection technique proposed by *Mitra et. al.* [11]. One algorithm is intended to determine the minimum subset of objectives that yields the minimum error possible, while the other finds a subset of objectives of a given size that yields the minimum error. To validate these algorithms we compare their results against those obtained by two similar algorithms recently proposed. The comparative study shows that our algorithms are very competitive with respect to the reference algorithms. Additionally, our approaches require a lower computational time. Also, in this study we propose to use the inverted generational distance to evaluate the quality of a subset of objectives.

Categories and Subject Descriptors

I.2.8 [**Computing Methodologies**]: Artificial Intelligence—*Problem Solving, Control Methods and Search*

General Terms

Algorithms, Performance, Experimentation

Keywords

Multiobjective Optimization, feature selection, many-objective problems, objective reduction, nonessential objectives

1. INTRODUCTION

Nowadays Multi-objective Evolutionary Algorithms (MOEAs) have shown an acceptable performance in many real-world problems with their origins in engineering, scientific and industrial areas [4]. Nonetheless, most of the publications consider problems with two or three objectives, in spite of the fact that many real-world problems involve a large number of objectives (4 or more). Besides the difficulty to analyze the Pareto front when there are more than three objectives, recent studies have shown that MOEAs based on Pareto optimality have difficulties to find a good Pareto front approximation in higher dimensions [10, 9, 14]. One of the reasons for this is that the proportion of non-dominated solutions (*i.e.*, equally good solutions) in a population increases rapidly with the number of objectives. In [8] it is shown that this number goes to infinity when the number of objectives approaches to infinity. This implies that in the presence of many objectives the selection of new solutions is carried out almost at random since a large number of the solutions are equally good in the Pareto sense.

Currently, there are mainly two approaches to deal with problems involving many objectives, namely: *i*) to propose relaxed forms of Pareto optimality as in [1, 8, 7, 12], and *ii*) to reduce the number of objectives of the problem to ease the decision making or the search processes [6, 2]).

In some problems it is possible that although a conflict exists between some objectives, others behave in a non-conflicting manner. In this case, we can discard these objectives to obtain a lower-dimensional problem. The reduction of objectives can be helpful both for decision making and search. That is, the decision maker would have to analyze fewer objectives and a lower number of non-dominated solutions. On the other hand, Pareto-based optimizers can be improved if the number of objectives is reduced adaptively during the search.

In this paper we propose an algorithm to reduce the number of objectives of a given problem by identifying the non-conflicting objectives (also called, non-essential or redundant objectives). The algorithm is based on an unsupervised feature selection technique proposed by Mitra *et al.* [11] where the goal is to identify the subset of essential objectives of a problem. We developed two variants of the algorithm, namely: *i*) an algorithm that finds the minimum subset of objectives with the minimum error possible, and *ii*) an algorithm that finds a subset of objectives of a given size and that yields the minimum error possible. A comparative study shows that the algorithms achieve competitive results with respect to two algorithms recently proposed [6, 2]. Besides, the proposed approaches have a lower time complexity, which make them good candidates to be incorporated into

GECCO '08 Atlanta, Georgia USA

a multiobjective optimization algorithm.

The remainder of this paper is organized as follows. Section 2 presents two algorithms similar to our approach. In Section 3 we describe in detail the two proposed algorithms. The validation of these algorithms is presented in Section 4. Finally, in Section 5 we draw some conclusions about the proposed algorithms, as well as some possible paths for future research.

2. RELATED WORK

Deb and Saxena [6] proposed a method for reducing the number of objectives based on principal component analysis. The main assumption is that if two objectives are negatively correlated (taking the generated Pareto front as the data set), then these objectives are in conflict with each other. To determine the most conflicting (*i.e.*, the most essential) objectives the authors analyze in turn the eigenvectors (*i.e.*, the principal components) of the correlation matrix. That is, by picking the most-negative and the most-positive elements from the first eigenvector, we can identify the two most important conflicting objectives. To aggregate more objectives to the set of essential objectives the remainder of the eigenvectors are analyzed in a similar way until the cumulative contribution of the eigenvalues exceeds a threshold cut (TC). This method is incorporated into an iterative scheme which uses a multiobjective optimizer (the actual implementation uses NSGA-II [5]) to obtain a reduced objective set containing only the non-redundant objectives according to the analysis of the eigenvectors. First, the evolutionary multiobjective optimizer is run and then the correlation analysis is carried out to obtain a reduced set of objectives. This process is repeated using the new reduced set of objectives. The process stops when the current subset is equal to the subset generated in the previous iteration.

Brockhoff and Zitzler [2] defined two kinds of objective reduction problems and two corresponding algorithms to solve them. Here the conflict is defined using the change in the dominance relation induced by the set of objectives over a solution set in the objective space. That is, if the dominance relation among the vectors does not change when an objective is discarded, then that objective is not in conflict with the other objectives and therefore is considered as redundant. This way, the degree of change in the dominance relation induced on the solution set can be regarded as a measure of conflict between two sets of objectives. One of the problems defined consists of finding the minimum objective subset that yields a given error δ (degree of change in the dominance relation) (the δ-MOSS problem, which stands for Minimum Objective Subset problem). The other problem consists of finding an objective subset of size k with the minimum error possible (the k-EMOSS problem). Since both problems are $\mathcal{NP}$-*hard*, the authors proposed both an exact and a greedy algorithm for each of them. The exact algorithms for both problems have time complexity $O(m^2 s 2^s)$, where m is the size of the given non-dominated set and s is the number of objectives. On the other hand, the greedy algorithm for the δ-MOSS problem has time complexity $O(\min\{m^2 s^3, m^4 s^2\})$, while the greedy algorithm for the k-EMOSS problem has time complexity $O(m^2 s^3)$.

3. PROPOSED OBJECTIVE REDUCTION ALGORITHMS

In this paper, we propose an unsupervised feature selection technique to identify the most conflicting objectives in order to reduce the number of objectives of an optimization problem. The technique employed was originally proposed by Mitra *et. al.* [11] to preprocess data prior to classification by selecting a subset of the original features.

As in Deb and Saxena's approach, this technique also uses a correlation matrix to measure the conflict between each pair of objectives. This matrix is computed using an approximation set of the Pareto front generated by some optimizer, for instance, a multiobjective evolutionary algorithm as we did in this study.

The original algorithm as proposed by Mitra *et. al.*, used $1 - |\rho(x, y)|$ (where $\rho(x, y)$ is the correlation coefficient between random variables x and y) as the similarity measure between features, which only determines the degree of correlation (positive or negative) between features (objectives in our context) x and y. However, in the case of objective selection we are interested in measuring only the negative correlation between objectives in the approximation set of the Pareto Front. For this purpose, we used $1 - \rho(x, y) \in [0, 2]$ instead. Thus, a result of zero indicates that objectives x and y are completely positively correlated and a value of 2 indicates that x and y are completely negatively correlated.

A negative correlation between a pair of objectives means that one objective increases while the other decreases and vice versa. On the other hand, if the correlation is positive, then both objectives increase or decrease at the same time. This way, we could interpret that the more negative the correlation between two objectives, the more conflict between them.

We propose the following algorithms to identify the essential objectives in a multiobjective problem:

1. Algorithm 1 finds the minimum subset of non-redundant objectives with the minimum error possible.

2. Algorithm 2 finds a subset of non-redundant objectives of a given size, k, yielding the minimum error possible.

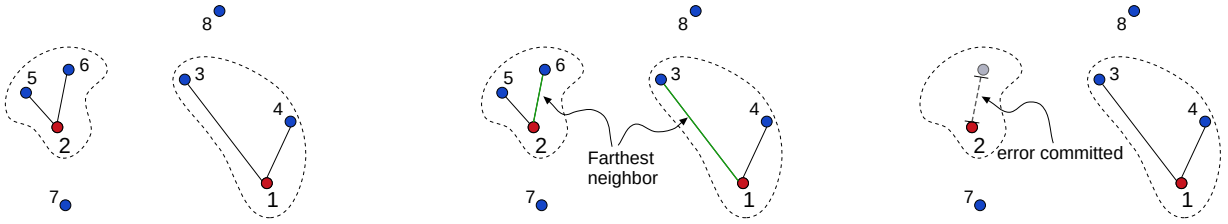
The central part of the two proposed algorithms can be divided in three steps:

1. Divide the objective set into homogeneous neighborhoods of size q around each objective. The conflict between objectives takes the role of the distance. That is, the more conflict between two objectives, the more distant they are in the objective space. Figure 1(a) shows only two neighborhoods of a hypothetical situation with eight objectives and $q = 2$.

2. Select the most compact neighborhood. That is, the neighborhood with the minimum distance to its neighbor q-th (*i.e.*, the farthest one in the neighborhood). Figure 1(b) shows the farthest neighbor for each of the two neighborhoods. As it can be seen in the example, the neighborhood on the left is the most compact one.

3. Retain the center of that neighborhood and discard its q neighbors (the objectives with least conflict in the current set). In this process, the distance to the neighbor q-th can be thought of as the error committed by removing the q objectives (see Figure 1(c)).

The process described above is repeated until some stopping criterion is met. It is important to mention that the



(a) Divide the objective set into neighborhoods around each objective.

(b) Select the most compact neighborhood.

(c) Retain the center and remove the neighbors.

Figure 1: Main scheme of the proposed objective reduction algorithms 1 and 2.

size of the neighborhoods q is a parameter that is reduced during the search.

Algorithm 1 is described in Figure 2, where r_i^q is an entry of the correlation matrix denoting the conflict between objective F_i and its q-th nearest-neighbor. Since the correlation matrix is computed at the beginning of Algorithm 1, we do not have to compute the r_i^q value each time we need it, but just take it from the corresponding entry of the correlation matrix.

Finally, we have to make some changes to the correlation matrix in order to use it to select the most conflicting objectives properly. If we use this matrix, it is possible that some important conflicting objectives are discarded. For instance, if objective F_2 is in conflict with F_3 but not with F_1, then F_1 will be very close to F_2 and, thus F_2 can be removed even if it is one of the most conflicting objectives. To overcome this problem we carried out the following process to the correlation matrix:

- Find the maximum conflict value $c_{i,max}$ of each row i in the matrix (*i.e.*, the maximum negative correlation value for each objective).
- Add the value $c_{i,max}$ to the column i. This means that we are assuming that if objective F_i is in conflict with some objectives, then it is in conflict with all the objectives.

Algorithm 2 only differs from Algorithm 1 in the stopping criterion and in the input parameters. Figure 3 shows the modifications required to obtain Algorithm 2.

The main advantage of both algorithms is their low-computational complexity since unlike other algorithms (*e.g.*, Brockhoff and Zitzler's algorithm), the search for the best subset is not involved. Regarding the number of objectives the proposed algorithms have complexity $O(s^2)$, where s is the number of objectives of the given non-dominated set. The computation of the correlation for each pair of objectives has complexity $O(m)$, where m is the size of the non-dominated set. Thus, the total complexity of both algorithms is $O(ms^2)$. For comparison purposes, Table 1 summarizes the complexities of the objective reduction algorithms included in this work. The time complexity of Deb and Saxena's algorithm corresponds to only one iteration, since the number of iterations depends on the threshold cut parameter. Usually, when this parameter is near to one, more iterations are required to converge. Also, we are considering that the NSGA-II is used as the optimizer of the overall algorithm. The second term of the complexity corresponds to the NSGA-II as was published in [5], where g is the number of generations.

For practical purposes, Figure 4 shows the running times of the three algorithms implemented in MATLAB. In a first experiment we fixed the size of the input nondominated set to 240 solutions and for Deb and Saxena's algorithm (DS

Input: Set of non-dominated solutions, A
Initial objective set $O = \{F_i, i = 1, ..., s\}$.
Number of neighbors $q \leq s - 1$.
Step 0: Compute the correlation matrix using A (each entry i, j corresponds to r_i^j).
Step 1: $R \leftarrow O$.
Step 2: Find objective $F_{i,min}$ which corresponds to $r_{i,min}^q \leftarrow \min_{F_i \in R} \{r_i^q\}$.
Step 3: Retain $F_{i,min}$ and discard its q neighbors.
Let $error \leftarrow r_{i,min}^q$.
Step 4: **If** $q > |R| - 1$ **then** $q \leftarrow |R| - 1$.
Step 5: **If** $q = 1$ **then** go to **Step 8** to stop.
Compute again $r_{i,min}^q \leftarrow \min_{F_i \in R} \{r_i^q\}$.
Step 6: **While** $r_{i,min}^q > error$ **do**:
$q \leftarrow q - 1$.
$r_{i,min}^q \leftarrow \min_{F_i \in R} \{r_i^q\}$.
If $q = 1$ **then** go to **Step 8**.
Step 7: Go to **Step 2**.
Step 8: Return set R as the reduced objective set.

Figure 2: Pseudocode of the proposed objective reduction *Algorithm 1*.

Input: Set of non-dominated solutions, A.
Initial objective set $O = \{F_i, i = 1, ..., s\}$ and **Size of the desired objective subset**, t.
⋮
Step 4: **If** $|R| = t$ **then** go to **Step 8** to stop.
Compute again $r_{i,min}^q \leftarrow \min_{F_i \in R} \{r_i^q\}$.
Step 5: **While** $r_{i,min}^q > error$ and $q > 1$ **do**:
⋮

Figure 3: Modifications to obtain objective reduction *Algorithm 2*.

Table 1: Time complexity of the objective reduction algorithms considered in this study (m is the size of the nondominated set, s the number of objectives and g the number of generations for each run of NSGA-II).

Algorithm	Complexity
Brockhoff and Zitzler (δ-MOSS)	$O(\min\{m^2s^3, m^4s^2\})$
Brockhoff and Zitzler (k-EMOSS)	$O(m^2s^3)$
Deb and Saxena[†]	$O(ms^2 + s^3) + O(gm^2s)$
Algorithms 1 and 2	$O(ms^2)$

[†]Complexity of each iteration of the algorithm.

algorithm) we used 50 generations for each run of NSGA-II. As we can see in the leftmost plot of Figure 4, the running time of the DS algorithm is too much greater than those of the other two algorithms. In a second experiment we analyze only the Brockhoff and Zitzler's algorithm (BZ algorithm) and Algorithm 2 fixing m=300 (middle and rightmost plots corresponds to these two algorithms respectively). Each of these plots shows the running time for the worst, "median" and best cases, depending on the size of the final objective subset size (k). Note that the worst and best cases for the these algorithms is presented for opposite values of k. Leaving trivial cases aside, for Algorithm 2, the worst case is presented when $k = 2$, while for BZ algorithm the worst case is presented when $k = s - 1$. For both algorithms, the median case is presented when $k = s/2$ for even s.

We want to end this section with two important remarks. First, it should be noted that although Algorithm 2 is intended to solve the k-EMOSS problem, Algorithm 1, as shown, does not solve the δ-MOSS problem, but a slightly different problem. Besides the difference between the semantic of the error involved, the δ-MOSS problem asks for the minimum subset with a given δ error, while Algorithm 1 finds the minimum subset with the minimum error possible. Second, we have to note that both algorithms proposed follow a top-down approach instead of a bottom-up approach like in the Brockhoff and Zitzler's algorithm or in Deb and Saxena's algorithm. That is, our algorithm starts with the whole set of objectives and iteratively removes some objectives until the minimum non-redundant objective set is obtained. In contrast, the other algorithms start with an empty set to which some objectives are aggregated at each iteration.

4. COMPARISON STUDY

4.1 Evaluation of Algorithm 1

To evaluate the effectiveness of the proposed Algorithm 1, we compare its results against those obtained by the approach proposed by Deb and Saxena (DS algorithm) and the greedy algorithm proposed by Brockhoff and Zitzler to solve the δ-MOSS (with $\delta = 0$). In this experiment, we employed a variation of the well-known DTLZ5 problem defined in [6]. This variation, denoted by DTLZ5(I,M), allows to fix *a priori* the number of essential objectives, I, from the total number of objectives, M.

We apply the three algorithms to four instances of the DTLZ5(I,M) problem, namely: DTLZ5(2,3), DTLZ5(2,5), DTLZ5(2,10), DTLZ5(3,10). For the proposed approach and for Brockhoff and Zitzler's algorithm we use as input data a non-dominated set of 500 solutions generated by the NSGA-II. We use an implementation of Deb and Saxena's algorithm following the instructions in [6] and for each run of the NSGA-II we use a population size of 500 and 300 generations (*i.e.*, a total of 150 000 evaluations).

Table 2: Essential objectives identified by the proposed Algorithm 1, Deb and Saxena's algorithm and Brockhoff and Zitzler's algorithm.

Problem	Reduced set of objectives
DTLZ5(2,3)	f_1, f_2
DTLZ5(2,5)	f_1, f_5
DTLZ5(2,10)	f_1, f_{10}
DTLZ5(3,10)	f_1, f_9, f_{10}

In all problems, the three algorithms were able to identify the essential objectives (which are the same as reported in [6]). Table 2 shows the essential objectives identified in each case.

4.2 Evaluation of Algorithm 2

The validation of Algorithm 2, intended to solve problem k-EMOSS (see Section 2), was carried out comparing its results with respect to the greedy algorithm proposed by Brockhoff and Zitzler (BZ algorithm) to solve the same problem. We also used Deb and Saxena's algorithm in this problem, however, since it does not address the k-EMOSS problem directly, we tried different values of the threshold cut (TC) to obtain different sizes of the final reduced objective set. In this comparison we considered two problems. First, two instances with 10 and 20 objectives of the 0/1 knapsack problem with 100 items. Second, two instances with 10 and 20 objectives of a variation, proposed in [3], of the well-known problem DTLZ2, which is denoted here by DTLZ2$_{BZ}$.

For the proposed approach and the BZ algorithm we used as input a non-dominated set with 500 solutions generated by the NSGA-II, while for Deb and Saxena's approach we employed a population of 500 individuals and 400 generations (*i.e.*, a total of 200 000 evaluations) for each run of the NSGA-II. We had to use this higher number of objectives in order to obtain a good approximation of the true Pareto front and therefore achieve a good effectiveness of the algorithm.

In this evaluation, we adopted two different measures. First, we used the δ-error defined by Brockhoff and Zitzler [2]. This error measures the degree of change between the dominance relation induced by a subset of objectives $\mathcal{F}'$ and the whole set of objectives, $\mathcal{F}$, with respect to a given solution set. Thus, a value of 0 in this metric means that the subset $\mathcal{F}'$ contains only essential objectives since the dominance relation does not change.

Ultimately, when we remove some objectives from a problem, we would want that the obtained Pareto front by using the objective subset is as near as possible to the Pareto front of the original problem (*i.e.*, that obtained considering all the objectives). This way, we can evaluate the quality of two reduced subsets using some quality indicators usually used to measure the convergence of a generated Pareto front to

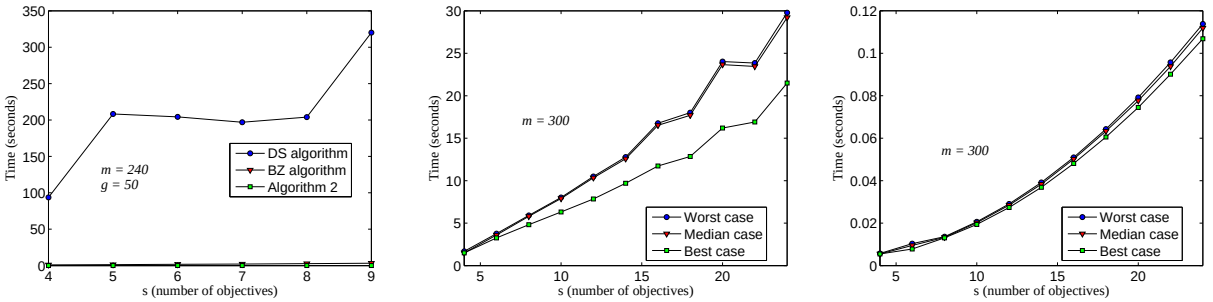


Figure 4: Running times (in seconds) of the three objective reduction algorithms. Leftmost plot shows the running times of the three algorithms using $k = s/2$, $m = 240$ and, for the DS algorithm, 50 generations for NSGA-II. Middle and rightmost plots show the running times for the worst ($k = s - 1, 2$), median ($k = s/2$) and best ($k = 2, s - 1$) cases for the BZ algorithm and Algorithm 2 respectively.

the true Pareto front (PF_{true}). The detailed process to make such a comparison is the following. First, we have to obtain the Pareto optimal sets generated by using the two objective subsets. Then, we evaluate the two corresponding sets of Pareto optimal solutions using all the objectives to obtain its known Pareto fronts (PF_{known}). The closeness of these known Pareto fronts to the true Pareto front can be used as a measure of quality of the reduced objective subsets. Besides the closeness to the true Pareto front we are interested in how well it is covered by the Pareto fronts obtained with the subsets of objectives. These two objectives are taken into account by the inverted generational distance, so this is a natural candidate to evaluate the quality of two objective subsets.

The *inverted generational distance* (IGD), which is a variantion of a metric proposed in [13], is defined by $IGD = \left(\sqrt{\sum_{i=1}^{n} d_i^2}\right)/n$, where $n = |PF_{true}|$ and d_i is the Euclidian distance between each vector of PF_{true} and the nearest member of PF_{known}. In addition, this metric measures the spread of PF_{known} onto PF_{true}. That is, a non-dominated set whose vectors are located on a reduced area of the Pareto-optimal set, will be penalized in the value of this metric even though its vectors belong (or are near) to PF_{true}. Lower values are preferred for this metric.

4.2.1 Evaluation Using the Change in the Dominance Relation

The results corresponding to the 0/1 knapsack problem with a total of 10 objectives and using the δ-error measure, are shown in Table 3. The first column indicates the size of the obtained reduced objective set, while the last line considers all the objectives, therefore, the error in this case is zero. Additionally, the best result of each row are emphasized with boldface.

The results show that the proposed algorithm obtained its best results when the number of objectives removed is small (from 1 to 5 objectives). In this problem, the BZ algorithm achieved the best result in 5 of the 8 cases, our approach obtained the best result in 3 cases, and the DS algorithm obtained the best result in only 1 case. Thus, with respect to the δ-error, we can consider that the BZ algorithm had the best performance on this particular instance, while the DS algorithm had the worst performance.

In the instance with 20 objectives (see Table 4), Algorithm 2 obtained the best result in 12 of the 18 cases, while BZ algorithm achieved the best result in 7 cases. On the other hand, DS algorithm obtained the worst result in 14 cases. We conclude that, in this instance, the proposed algorithm had the best performance and the DS algorithm the worst performance.

Table 5 shows the results for the problem DTLZ2$_{BZ}$ using a total of 10 objectives. In this instance, the proposed algorithm achieved the best results in 5 of the 8 cases, the BZ algorithm obtained the best results in 3 cases, and the DS algorithm obtained the best result only once. From this results, we can say that the proposed algorithm had the best performance on this problem. However it is not clear which algorithm had the worst performance. On the one hand, the DS algorithm obtained the worst result in 2 cases and, on the other, although the BZ algorithm achieved the best result in 3 cases, it showed the worst result in 4 cases. Additionally, it is interesting to note that for a subset of 9 objectives, unlike the two reference algorithms, Algorithm 2 was unable to find a subset with a δ-error equal to zero. That is to say, a subset that does not change absolutely the dominance relation.

With respect to the instance DTLZ2$_{BZ}$ with a total of 20 objectives (see Table 6), the BZ algorithm obtained the best performance achieving the best result in 12 cases, while the proposed algorithm achieved the best result in 10 of the 18 cases. One more time, the DS algorithm was the worst algorithm since it obtained the worst results in 10 cases. In this problem we can observe that 4 objectives are completely redundant, since the dominance relation does not change absolutely from 16 to 20 objectives. Brockhoff and Zitzler's approach was the only algorithm that reflected this fact, while the DS algorithm and Algorithm 2 found a subset with zero δ-error until sizes of 19 and 18, respectively.

4.2.2 Evaluation Using the Inverted Generational Distance

To evaluate the convergence of the obtained Pareto front by using an objective subset we used the inverted generational distance (IGD). This performance measure takes into account the convergence of the known Pareto front and how well it covers the extension of the true Pareto front. In some way, it measures how similar is the Pareto front obtained by using a subset of the objectives with respect to the "original"

Table 3: Comparison with respect to the δ-error using the 0/1 knapsack problem with 10 objectives.

	δ-error		
# obj	BZ	DS	Algorithm 2
2	**1458**	1466	1595
3	**1412**	1513	1596
4	**1331**	1533	1466
5	1158	1362	**1148**
6	954	**859**	988
7	881	1148	**822**
8	**822**	1148	**822**
9	**569**	859	614

Table 4: Comparison with respect to the δ-error using the 0/1 knapsack problem with 20 objectives.

	δ-error		
# obj	BZ	DS	Algorithm 2
2	**1467**	1599	1589
3	1434	1580	**1304**
4	1388	1433	**1304**
5	**1286**	1406	1304
6	**1243**	1405	1304
7	1164	1233	**1100**
8	1158	1225	**1003**
9	1149	1164	**993**
10	1075	1075	**918**
11	1056	1055	**804**
12	993	1153	**804**
13	937	918	**804**
14	871	1036	**804**
15	793	843	**785**
16	785	805	**785**
17	**699**	793	785
18	**359**	586	785
19	**41**	104	785

Table 5: Comparison of the three algorithms with respect to the δ-error, using the DTLZ2$_{BZ}$ problem with 10 objectives.

	δ-error		
# obj	BZ	DS	Algorithm 2
2	**0.6959**	0.7353	0.7353
3	**0.6688**	0.7051	0.7051
4	0.6610	0.6239	**0.5455**
5	0.6051	0.5890	**0.4908**
6	0.5916	0.5799	**0.4682**
7	0.5707	0.5799	**0.3787**
8	0.5151	0.5130	**0.3054**
9	**0.0000**	**0.0000**	0.3054

Table 6: Comparison of the three algorithms with respect to the δ-error, using the DTLZ2$_{BZ}$ problem with 20 objectives.

	δ-error		
# obj	BZ	DS	Algorithm 2
2	**0.5931**	0.7442	0.7326
3	0.5925	0.7326	**0.5761**
4	**0.5570**	0.6275	0.5761
5	**0.4818**	0.5059	**0.4818**
6	0.4441	0.5171	**0.3706**
7	0.3257	0.4799	**0.2877**
8	0.3056	0.3257	**0.2877**
9	**0.2877**	0.4532	**0.2877**
10	**0.2461**	0.3036	0.2877
11	0.0569	0.3706	**0.0279**
12	0.0532	0.4125	**0.0216**
13	**0.0068**	0.4125	0.0216
14	**0.0038**	0.3056	0.0216
15	**0.0001**	0.1014	0.0216
16	**0.0000**	0.3056	0.0216
17	**0.0000**	0.0038	0.0068
18	**0.0000**	0.0038	**0.0000**
19	**0.0000**	**0.0000**	**0.0000**

Pareto front of the problem which includes all the objectives.

The objective subsets to assess the three algorithms with the IGD metric are those used in the previous section to compute the δ-error. Likewise, in the present study the known Pareto fronts corresponding to each objective subset were obtained using the NSGA-II. For each objective subset the following parameters were used: 500 individuals and 1000 generations. The large number of generations was intended to produce Pareto fronts with a small standard deviation of IGD for each subset. The IGD values shown in this section are the average of 20 runs for each objective subset. Moreover, instead of using the true Pareto front, we used the non-dominated set resulting from the union of the known Pareto fronts generated using the objective subsets of the three algorithms, as well as the known Pareto front used as input in the objective reduction algorithms.

Table 7 shows the results for the 0/1 knapsack problem with 10 objectives with respect to the IGD metric. As we can see, in 4 of the 8 cases, the proposed algorithm achieved the best results, while the BZ and DS algorithms obtained the best result in 3 and 1 cases, respectively. With respect to the IGD metric the proposed approach had the best performance on this problem. It is worthwhile to remember that regarding the δ-error, in this instance the BZ algorithm obtained the best performance. This shows a clear inconsistency between the results based on the performance measure adopted. In particular, the DS algorithm obtained the worst result for the subset of 4 objectives using the δ-error, while using the IGD metric it achieved the best result.

Regarding the 0/1 knapsack instance with 20 objectives (see Table 8), we can clearly see that the proposed algorithm had the best performance, which agrees with the conclusion obtained using the δ-error. However, the DS algorithm had the second best performance in this instance, while it was the worst algorithm with respect to the δ-error.

With respect to the DTLZ2$_{BZ}$ with 10 objectives, both

Table 7: Comparison of the three algorithms with respect to the inverted generational distance (IGD), using the 0/1 knapsack problem with 10 objectives.

	IGD		
# obj	BZ	DS	Algorithm 2
2	5.7834	7.5759	**5.7452**
3	**7.5212**	8.9904	7.5425
4	5.8694	**5.4537**	5.8445
5	3.7525	5.2474	**3.7406**
6	3.4633	3.6170	**3.4318**
7	3.2607	3.6615	**3.2473**
8	**3.1893**	3.7855	3.2001
9	**3.1521**	3.3678	3.1541

Table 8: Comparison of the three algorithms with respect to the inverted generational distance (IGD), using the 0/1 knapsack problem with 20 objectives.

	IGD		
# obj	BZ	DS	Algorithm 2
2	**7.7657**	8.0942	10.0344
3	6.7232	**6.4659**	6.9511
4	5.8777	**5.1232**	5.4373
5	4.8191	4.4417	**4.1027**
6	5.0627	4.8973	**4.7977**
7	5.1815	4.9218	**4.4888**
8	4.8017	5.0338	**4.7557**
9	4.8552	4.6459	**4.6217**
10	5.0189	5.0132	**4.9704**
11	4.7852	4.6779	**4.6470**
12	4.8673	4.8582	**4.7358**
13	4.9812	**4.7996**	4.8880
14	5.2370	5.0018	**4.9121**
15	5.1767	5.1215	**4.9753**
16	5.1166	5.0791	**4.9389**
17	**5.0209**	5.0534	5.0534
18	5.2423	**5.1553**	5.1639
19	5.1714	**5.1661**	5.2538

Algorithm 2 and the DS algorithm obtained the best IGD values in 5 of the 9 cases (see Table 9). Algorithm BZ obtained the best result in 3 cases, although it presented the worst result in 5 cases.

Nonetheless, we can observe a discrepancy between some results obtained by the two performance measures. On the one hand, using the δ-error, the DS algorithm showed the worst performance, but in the other hand, it is the best algorithm regarding the IGD metric.

In the DTLZ2$_{BZ}$ instance using 20 objectives (see Table 10), the BZ algorithm showed the best performance. It obtained the best results in 13 cases. The proposed problem obtained the second best performance achieving the best results in 5 cases. Finally, the DS algorithm obtained the worst results in 13 cases. Therefore it was the worst algorithm in this instance considering the IGD metric.

4.3 Final Remarks

Since Algorithm 2 follows a top-down approach one would

Table 9: Comparison of the three algorithms with respect to the inverted generational distance (IGD), using the DTLZ2$_{BZ}$ problem with a total of 10 objectives.

	IGD		
# obj	BZ	DS	Algorithm 2
2	0.00226	**0.00089**	**0.00089**
3	0.00243	**0.00071**	0.00080
4	**0.00082**	0.00089	**0.00082**
5	0.00115	0.00092	**0.00090**
6	0.00115	**0.00099**	0.00102
7	**0.00093**	0.00102	0.00114
8	0.00093	**0.00082**	**0.00082**
9	**0.00077**	**0.00077**	**0.00077**

Table 10: Comparison of the three algorithms with respect to the inverted generational distance (IGD), using the DTLZ2$_{BZ}$ problem with a total of 20 objectives.

	IGD		
# obj	BZ	DS	Algorithm 2
2	0.00235	**0.00115**	**0.00115**
3	0.00144	0.00253	**0.00078**
4	0.00104	0.00233	**0.00094**
5	**0.00094**	0.00110	0.00103
6	**0.00115**	0.00124	0.00148
7	0.00126	**0.00115**	0.00128
8	0.00115	**0.00104**	0.00110
9	**0.00103**	0.00123	0.00120
10	**0.00100**	0.00104	0.00102
11	**0.00097**	0.00104	0.00099
12	**0.00094**	0.00108	0.00097
13	**0.00094**	0.00104	0.00095
14	**0.00092**	0.00107	0.00097
15	**0.00091**	0.00095	0.00092
16	**0.00094**	0.00101	0.00098
17	**0.00095**	0.00096	**0.00095**
18	**0.00092**	0.00093	0.00093
19	**0.00092**	0.00093	**0.00092**

expect that it would achieve its best results when the size of the objective subset is close the total number of objectives, specially when $k = s - 1$. However for this case, in 3 of the problems considered (both knapsack instances and DTLZ2$_{BZ}$ with 10 objectives), Algorithm 2 was unable to obtain the least δ-error. Although in general, the IGD tends to decrease as the size of the objective subset increases, in some particular cases the IGD values increase when an objective is added. One possible reason for this behavior is that the number of objectives affects the optimizer degrading its performance as the number of objectives is increased.

5. CONCLUSIONS AND FUTURE WORK

This paper presented two algorithms to identify the most-conflicting objectives of a problem so that we can obtain a reduced set objectives that makes the search (or the decision making process) easier. Both algorithms are based on the correlation between each pair of objectives to detect the essential objectives. One algorithm finds the minimum subset of objectives with the minimum error possible (Algorithm 1), and the other finds a subset of objectives with a given size yielding the minimum error possible (Algorithm 2).

The results showed that the proposed algorithms are very competitive with respect to other two similar algorithms recently proposed [6, 2]. One advantage of the proposed algorithms over the two reference algorithms is their low computational time. To compare the effectiveness of the algorithms we adopted the δ-error defined in [2]. Additionally, we proposed the use of the inverted generational distance to measure the quality of the obtained reduced set of objectives. Algorithm 1 was able to identify all the essential objectives of the problem considered. With regard to the δ-error, Algorithm 2 achieved the best performance in 2 of the 4 problem instances included (0/1 knapsack with 20 objectives and DTLZ2$_{BZ}$ with 10 objectives), while it showed the best performance in 3 of the 4 instances regarding the IGD metric. In general, the DS algorithm presented the worst performance in the 4 instances considered.

Although, in general, the performance measures yielded consistent results, in some specific cases they yielded completely contradictory results. This calls for the need to determine which performance measure is better to evaluate the quality of a reduced set of objectives.

As part of our future work it would be interesting to study, in more depth, the behavior of the two performance measures considered in this article to discover the source of the discrepancy observed here. Also, we are interested in adopting another performance measure, for instance the hypervolume indicator [15], to evaluate the quality of two objective subsets. Finally, given the low complexity of the proposed algorithms, we intend to develop a multiobjective evolutionary algorithm that integrates the proposed Algorithm 2 and assess its performance in a real-world application having a high number of objectives.

6. REFERENCES

[1] P. J. Bentley and J. P. Wakefield. Finding Acceptable Solutions in the Pareto-Optimal Range using Multiobjective Genetic Algorithms. In P. K. Chawdhry, R. Roy, and R. K. Pant, editors, *Soft Computing in Engineering Design and Manufacturing*, Part 5, pages 231–240, London, June 1997. Springer Verlag London Limited. (Presented at the 2nd On-line World Conference on Soft Computing in Design and Manufacturing (WSC2)).

[2] D. Brockhoff and E. Zitzler. Are All Objectives Necessary? On Dimensionality Reduction in Evolutionary Multiobjective Optimization. In *Parallel Problem Solving from Nature IX*, pages 533–542. Springer-Verlag, 2006.

[3] D. Brockhoff and E. Zitzler. Offline and Online Objective Reduction in Evolutionary Multiobjective Optimization Based on Objective Conflicts. TIK Report 269, Institut für Technische Informatik und Kommunikationsnetze, ETH Zürich, Apr. 2007.

[4] C. A. Coello Coello, G. B. Lamont, and D. A. Van Veldhuizen. *Evolutionary Algorithms for Solving Multi-Objective Problems*. Springer, New York, second edition, September 2007. ISBN 978-0-387-33254-3.

[5] K. Deb, S. Agrawal, A. Pratab, and T. Meyarivan. A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II. In M. Schoenauer, K. Deb, G. Rudolph, X. Yao, E. Lutton, J. J. Merelo, and H.-P. Schwefel, editors, *Parallel Problem Solving from Nature VI*, pages 849–858, Paris, France, 2000. Springer. Lecture Notes in Computer Science No. 1917.

[6] K. Deb and D. Saxena. On finding pareto-optimal solutions through dimensionality reduction for certain large-dimensional multi-objective optimization problems. *Kangal report*, 2005.

[7] F. di Pierro. *Many-Objective Evolutionary Algorithms and Applications to Water Resources Engineering*. PhD thesis, School of Engineering, Computer Science and Mathematics, UK, August 2006.

[8] M. Farina and P. Amato. On the Optimal Solution Definition for Many-criteria Optimization Problems. In *Proceedings of the NAFIPS-FLINT International Conference'2002*, pages 233–238, Piscataway, New Jersey, June 2002. IEEE Service Center.

[9] E. J. Hughes. Evolutionary Many-Objective Optimisation: Many Once or One Many? In *2005 IEEE Congress on Evolutionary Computation (CEC'2005)*, volume 1, pages 222–227, Edinburgh, Scotland, September 2005. IEEE Service Center.

[10] V. Khare. Performance Scaling of Multi-Objective Evolutionary Algorithms. Master's thesis, School of Computer Science, The University of Birmingham, Edgbaston, Birmingan, UK, September 2002.

[11] P. Mitra, C. Murthy, and S. Pal. Unsupervised feature selection using feature similarity. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(3):301–312, 2002.

[12] A. Sülflow, N. Drechsler, and R. Drechsler. Robust Multi-objective Optimization in High Dimensional Spaces. In S. Obayashi, K. Deb, C. Poloni, T. Hiroyasu, and T. Murata, editors, *Evolutionary Multi-Criterion Optimization, 4th International Conference, EMO 2007*, pages 715–726, Matshushima, Japan, March 2007. Springer. Lecture Notes in Computer Science Vol. 4403.

[13] D. A. V. Veldhuizen. *Multiobjective Evolutionary Algorithms: Classifications, Analyses, and New Innovations*. PhD thesis, Department of Electrical and Computer Engineering. Graduate School of Engineering. Air Force Institute of Technology, Wright-Patterson AFB, Ohio, May 1999.

[14] T. Wagner, N. Beume, and B. Naujoks. Pareto-, Aggregation-, and Indicator-Based Methods in Many-Objective Optimization. In S. Obayashi, K. Deb, C. Poloni, T. Hiroyasu, and T. Murata, editors, *Evolutionary Multi-Criterion Optimization, 4th International Conference, EMO 2007*, pages 742–756, Matshushima, Japan, March 2007. Springer. Lecture Notes in Computer Science Vol. 4403.

[15] E. Zitzler and L. Thiele. Multiobjective Optimization Using Evolutionary Algorithms—A Comparative Study. In A. E. Eiben, editor, *Parallel Problem Solving from Nature V*, pages 292–301, Amsterdam, September 1998. Springer-Verlag.