Feedback-control operators for improved Pareto-set description: Application to a polymer extrusion process

Eduardo G. Carrano^a, Dayanne Gouveia Coelho^b, António Gaspar-Cunha^c, Elizabeth F. Wanner^d, Ricardo H.C. Takahashi^{e,*}

^a Department of Electrical Engineering, Universidade Federal de Minas Gerais, Brazil

^b Department of Computing, Universidade Federal de Ouro Preto, Brazil

^c Institute for Polymer and Composites – I3N, University of Minho, Campus de Azurém, 4800-058 Guimarães, Portugal

^d Department of Computing, Centro Federal de Educação Tecnológica de Minas Gerais, Brazil

^e Department of Mathematics, Universidade Federal de Minas Gerais, Brazil

ARTICLE INFO

Article history:

Received 4 September 2013

Received in revised form

7 July 2014

Accepted 21 October 2014

Available online 26 November 2014

Keywords:

Evolutionary computation
Multiobjective optimization
Genetic algorithms
Polymer extrusion
Local search

ABSTRACT

This paper presents a new class of operators for multiobjective evolutionary algorithms that are inspired on feedback-control techniques. The proposed operators, the *archive-set reduction* and the *surface-filling crossover*, have the purpose of enhancing the quality of the description of the Pareto-set in multi-objective optimization problems. They act on the Pareto-estimate sample set, performing operations that eliminate archive points in the most crowded regions, and generate new points in the less populated regions, leading to a dynamic equilibrium that tends to generate a uniform sampling of the efficient solution set. The internal parameters of those operators are coordinated by feedback-control inspired techniques, which ensure that the desired equilibrium is attained. Numerical experiments in some benchmark problems and in a real problem of optimization of a single screw extrusion system for polymer processing show that the proposed methodology is able to generate more detailed descriptions of Pareto-optimal fronts than the ones produced by usual algorithms.

© 2014 Elsevier Ltd. All rights reserved.

1. Introduction

One of the main concerns in the design of multi-objective evolutionary algorithms (MOEAs) has been to ensure the quality of the sample set of Pareto-optimal estimates that is generated by the algorithm. The quality measure is, by itself, multi-dimensional, and there are not, up to now, any definitive standards that represent it (Zitzler et al., 2003; Auger et al., 2012). A high-quality solution set can be defined as a set of samples that reach, as long as possible, the exact Pareto-set, and is representative of the whole extension of the Pareto-set (Silva et al., 2007; Zitzler et al., 2010). It should be noticed that a MOEA can be built with the purpose of describing a subset of the Pareto set, in the cases in which some *a priori* or *online* decision information is available (Kim et al., 2012; Sinha et al., 2013). In those cases, the quality measures should refer to the representation of such subsets (Auger et al., 2012).

This paper introduces new operators that can be used to enhance the ability of a MOEA to represent the Pareto set in detail, with a uniform spread of the samples: the *sphere-control operators*. Such operators are based on the information about the distances between every pair of solution samples in a set – this motivates the denomination of “sphere” operators. The key idea is the usage of a *feedback-control* inspired scheme (Ogata, 2001) in order to establish a dynamic equilibrium along the algorithm iterations, associated to the high-quality description of the Pareto-set. While such a high-quality description is not reached, some measured variables do not reach an equilibrium, causing a control action that will enhance a quality metric.

The proposed methodology is applied to an optimization problem of the operational and screw geometrical parameters of a single screw polymer extrusion system. In this problem, a modeling routine describing properly the complex process developed is necessary. This involves the mathematical description of the different phases suffered by the polymer inside the machine (Gaspar-Cunha, 2009). This practical problem of strong industrial relevance concerns the employment of multiobjective optimization on a simulation model for the purpose of gaining knowledge about possible efficient operating modes to be implemented in a real machine. The main purpose of such a multiobjective optimization study is to obtain

* Corresponding author.

E-mail addresses: carrano@cpdee.ufmg.br (E.G. Carrano), dayagc@gmail.com (D. Gouveia Coelho), agc@dep.uminho.pt (A. Gaspar-Cunha), efwanner@dppg.cefetmg.br (E.F. Wanner), taka@mat.ufmg.br (R.H.C. Takahashi).

Fig. 1. Diagram of a mechanism, inspired on a PI feedback control loop, that controls the number of points stored in the archive set.

different optimal solutions that will help the polymer engineers in selecting the best operating conditions and/or screw geometry. Further discussions about the application of evolutionary computation methods for problems in materials science and engineering are reported in Paszkowicz (2013).

An instance of application of a feedback-control inspired scheme, as proposed here, is represented in Fig. 1. In this figure, the measured variable is the error $e = q - a$, which feeds a mechanism that is inspired in the proportional-integral (PI) controller, which in turn determines the value of the control variable ρ . ρ , which represents the radius of the spheres associated to the points in the archive file, plays the role of the control input variable, while the variable a , which represents the number of points in the archive file, plays the role of the controlled variable. The pre-established reference number of points in the archive, denoted by q , will be attained in an equilibrium¹ situation that will be reached due to the feedback mechanism.

In the equilibrium, $e=0$ (which means the desired result of $a=q$). As in other contexts of application of feedback-control techniques, the role of the feedback-control inspired scheme here is to induce an overall system behavior that presents low sensitivity to variations in the initial conditions and in the algorithm parameter values, delivering stable results, with repeatability in the reach of high-quality solution sets (Ogata, 2001). The error variables are defined here such that the feedback loop reaches an equilibrium only when a good description of the Pareto-set is attained.

Specifically, two sphere-control operators are presented here: an *archive-set reduction* operator, which controls the number of non-dominated solutions that are stored, and a crossover operator that is applied in the archive set, the *surface-filling crossover*. The archive-set reduction plays the role of solution density reduction in the most crowded regions of the Pareto-set. The surface-filling crossover is to be applied in the less crowded regions of the Pareto-set, in order to fill eventual gaps in its description. The equilibrium between the opposite actions of surface-filling and archive-reduction, attained by a feedback-control inspired scheme, leads to an ultimate description of the Pareto-set that is composed of samples that are evenly distributed in the space of objectives.

Feedback-control inspired mechanisms, based on the principles of a switched controller and of a proportional-integral (PI) controller, are employed in the surface-filling operators and in the archive-set reduction operator, in order to enhance the distribution of solutions along the Pareto surface. This motivates the denomination of “sphere-control” operators. These operators are to be employed together, since their effects are complementary, and their dynamic interaction is necessary in order to achieve the desired behavior.

It should be noticed here that the idea of feedback-control inspiration constitutes a further step beyond some studies that have considered the theme of *parameter adaptation* in evolutionary computation. Examples of those studies can be found for

instance in Vasconcellos et al. (2001), McGinley et al. (2011), Lin and Chen (2013), and Jebari et al. (2013). The introduction of feedback control concepts in order to state the parameter adaptation procedures allows the recovery, within the field of evolutionary computation, of some well-established results from control theory concerning closed-loop system dynamics and stability. Some preliminary studies concerning the proposed feedback-control scheme have been presented by the authors in Takahashi et al. (2009). The ideas presented here also have connection with the ones presented in Takahashi et al. (2004) and Silva et al. (2007) which employ “sphere” operations which are similar to the archive-set reduction operator presented here, yet without any feedback adaptation scheme.

The basic idea, both in that references and here, is that a “sphere” means roughly a domain in which the information gained by a solution point in its center would be representative – with no need of further function evaluations inside such a sphere. Bui et al. (2008, 2009) also employ the concept of “spheres” for construction of an MOEA (multi-objective evolutionary algorithm), with a dual meaning: in that cases, the “sphere” is the domain in which a local search is conducted, with sub-populations assigned to perform searches inside each sphere.

In the specific formulation that is presented here, the proposed operators are structured for continuous-variable spaces. However, the adaptation for discrete-variable problems can be performed directly, provided that some distance metric becomes defined in the discrete-variable space. This can be performed according to the guidelines presented by Carrano et al. (2010).

It should be noticed that the proposed methodology may be supplemented by the usage of local search operators, that can enhance the precision of solutions if applied along the algorithm iterations (Wanner et al., 2008), or even provide a certificate of optimality in some cases (Takahashi et al., 2011). Those hybridizations are not discussed further, in order to privilege the presentation of the main aspects of the methodology proposed here.

This paper is structured as follows. A discussion about multi-objective evolutionary algorithms is provided in Section 2. The proposed feedback-control inspired operators are presented in Section 4. Section 3 presents a modified version of the classical NSGA-II algorithm, in which the proposed sphere-control operators are included. The results of numerical tests on some benchmark problems are presented in Section 5. The numerical tests show that the proposed methodology leads to a significant enhancement of the ability of the algorithm for reaching a fine-grained description of the Pareto-optimal set. Finally, in Section 7, the proposed algorithm is employed in the problem of polymer extrusion process design. The results are compared with the ones obtained using the basic NSGA-II and with another algorithm, the RPSGA, that was employed formerly for dealing with the same problem. The results support the conclusion that the proposed methodology is able to significantly enhance the description of the efficient solution set in this practical problem too.

2. Multiobjective evolutionary algorithms

Consider $f(\cdot) : \mathbb{R}^n \mapsto \mathbb{R}^m$ a vector-valued real function. Let $f_i(\cdot)$ denote the i -th coordinate of the function in the image space. The multiobjective problems appear from the partial ordering induced by the relation of *dominance*:

$$u < v \Leftrightarrow \begin{cases} f_i(u) \leq f_i(v) & \forall i = 1, \dots, m \\ \text{and} \\ \exists i \in \{1, \dots, m\} \text{ such that } f_i(u) < f_i(v) \end{cases} \quad (1)$$

¹ According to the analogy employed here, the *equilibrium* refers to the situation in which the algorithm variables converge to values that remain fixed along the algorithm iterations.

Consider the Pareto-set $\mathcal{P}$ defined by

$$\mathcal{P} \triangleq \{x^* \in \Omega \mid \nexists x \in \Omega \text{ such that } x \prec x^*\} \quad (2)$$

in which $x \in \mathbb{R}^n$ is the decision variable vector, and $\Omega \subseteq \mathbb{R}^n$ is the feasible set. A multiobjective optimization problem is defined as the task of generating samples of the set $\mathcal{P}$. A multiobjective evolutionary algorithm (MOEA) is an evolutionary algorithm which is intended to produce a set of samples of $\mathcal{P}$. At the end of the execution, the archive set A_k contains the algorithm outcome, which constitutes an estimate of the Pareto-set $\mathcal{P}$.

3. MOEA with sphere-control operators

This paper presents new evolutionary operators which are not intended to be algorithm-specific. Instead, they should be suitable to be inserted in any MOEA that works with archive solutions. For this reason, in this paper the proposed operators are inserted within a standard MOEA, the Non-Dominated Sorting Genetic Algorithm II (NSGA-II) (Deb et al., 2002), in order to define a working algorithm. The resulting algorithm, named Sphere-Control Multiobjective Genetic Algorithm (SCMGA), is compared here with the basic NSGA-II over a set of benchmark problems, in order to establish some figures about the effect of introducing the sphere-control operators inside a known MOEA. The basic NSGA-II and the proposed SCMGA are described briefly in this section.

3.1. The NSGA-II

The Non-dominated Sorting Genetic Algorithm (Deb et al., 2002), or simply NSGA-II, has been used for dealing with the multiobjective optimization problems considered here. For a full description of that algorithm, the reader should refer to the original reference. The following procedures, which are part of NSGA-II, will be employed in the next section in the construction of the SCMGA (the Sphere-Control Multiobjective Genetic Algorithm):

- $A \leftarrow \text{non_dominated}(P)$ returns the individuals that lie in the first front of the population P ;
- $\mathcal{F} \leftarrow \text{fast_non_dominated_sorting}(P)$ sorts the individuals of population according to fronts, the solutions in each front dominated only by solutions in the former ones;
- $C_i \leftarrow \text{crowding_distance_assignment}(\mathcal{F}_i)$ employs a *crowding distance assignment* procedure to estimate how the solutions of front i are spread in the objective space;
- $Q \leftarrow \text{selection}(P, N)$ uses binary stochastic tournaments with replacement for performing selection of a new population Q , composed of N individuals, from population P .

3.2. The SCMGA

The structure of the Sphere-Control Multiobjective Genetic Algorithm (SCMGA), is similar to the one of the original NSGA-II. Basically, this algorithm presents three major differences in relation to the original one:

- *Archive size control scheme*: In the original NSGA-II, an elitist selection scheme allows the current population to serve as an archive, which is built upon the crowding distance measure. However, while the control of the archive size based on crowding distance is efficient for two objectives, it is not so reliable when more objective functions are considered (Kukkonen and Deb, 2006). In the proposed algorithm, an external archive uses

the sphere-based archive control procedure (Section 4.1). The PI controller that is used for performing the archive-size control adjusts the radius ρ in order to keep the q most representative solutions (or a number of solutions that is close to q). As this method employs spheres for controlling the archive size, it is suitable for dealing with any number of objective functions, since spheres can be fully parametrized by the radius only, in spaces of any dimension. The archive, in SCMGA, interacts with the population on the end of each iteration.

- *Insertion of surface-filling crossover*: The surface-filling crossover, described in Section 4.2, is executed at the end of each iteration in order to fill the blank spaces of the archive. The number of surface-filling crossover operations which are performed depends on the measured efficiency of this operator in the previous generations.
- *Auto-adaptation of the algorithm parameters*: The controller parameters ρ , β and r are adjusted inside the SCMGA. The ρ parameter is adjusted based on the current number of solutions in the archive and the desired one. The β parameter is adjusted in order to obtain solutions which are not excessively near the parents in surface-filling crossover. Finally, the r parameter is adjusted based on the efficiency of surface-filling crossover: if the number of successful operations is higher than the unsuccessful ones r decreases; otherwise r increases.

The details of these operations are described in the next section. Algorithm 1 shows the pseudo-code for the SCMGA algorithm.

Algorithm 1. Pseudocode for SCMGA.

```

Inputs:  $N, q$                                 ▷  $N$ : population size;
                                                 $q$ : archive size
                                                 $A$ : archive set

Output:  $A$ 
 $t \leftarrow 0$ 
 $\rho, \beta, r \leftarrow \text{start\_}\rho\_ \beta\_ r$ 
 $P_t \leftarrow \text{new\_population}(N)$                 ▷ generates random
                                                population of size  $N$ 

 $Q_t \leftarrow \emptyset$ 
 $A \leftarrow \text{non\_dominated}(P_t)$ 
while not stop criterion do
     $R_t \leftarrow P_t \cup Q_t$ 
     $\mathcal{F} \leftarrow \text{fast\_non\_dominated\_}$ 
    sorting( $R_t$ )
     $P_{t+1} \leftarrow \emptyset$ 
     $i \leftarrow 1$ 
    while  $|P_{t+1}| + |\mathcal{F}_i| \leq N$  do
         $C_i \leftarrow \text{crowding\_distance\_}$ 
        assignment( $\mathcal{F}_i$ )
         $P_{t+1} \leftarrow P_{t+1} \cup \mathcal{F}_i$ 
         $i \leftarrow i + 1$ 
    end while
     $\mathcal{F}_i \leftarrow \text{sort}(\mathcal{F}_i, C_i, \text{'descending'})$     ▷ sorts front  $i$ , descending
                                                order of  $C_i$ 

     $P_{t+1} \leftarrow P_{t+1} \cup \mathcal{F}_i[1 : (N - |P_{t+1}|)]$ 
     $Q_{t+1} \leftarrow \text{selection}(P_{t+1}, N)$ 
     $Q_{t+1} \leftarrow \text{crossover}(Q_{t+1})$ 
     $Q_{t+1} \leftarrow \text{mutation}(Q_{t+1})$ 
     $t \leftarrow t + 1$ 

     $A \leftarrow \text{non\_dominated}(A \cup Q_t)$ 
     $(T, k_1, k_2, k_3, k_4, k_5, k_6) \leftarrow \text{Surf\_Fill\_XO}(A, \rho, r, 4, \beta)$ 
     $(r, \beta) \leftarrow \text{Update\_r\_}\beta$ 
     $(r, \beta, k_1, k_2, k_3, k_4, k_5, k_6)$ 
     $Q_t \leftarrow Q_t \cup T$ 
     $A \leftarrow \text{non\_dominated}(A \cup T)$ 

```

```

 $A \leftarrow \rho\_Archive\_Control(A, \rho)$ 
 $\rho \leftarrow Update\_ \rho(A, q)$ 
end while

```

In this algorithm, $\rho, \beta, r \leftarrow start_ \rho_ \beta_ r$ sets the initial values of the controlled parameters ρ, β and r . ρ is set in such a way that q spheres of dimension $(m-1)$ occupy a volume equivalent to the unitary simplex of dimension $(m-1)$. β and r are started with the mean value of the intervals which are defined by the lower and the upper bounds respectively.

4. The sphere operators

Consider the sets A and P respectively meaning the archive and the current population:

$$\begin{aligned} A &\triangleq \{\tilde{x}_1, \dots, \tilde{x}_a\} \\ P &\triangleq \{x_1, \dots, x_p\} \end{aligned} \quad (3)$$

with $|A| = a$ and $|P| = p$. The images of such sets in the objective space are denoted by

$$\begin{aligned} A_y &= \{\tilde{y}_1, \dots, \tilde{y}_a\} \triangleq \{f(\tilde{x}_1), \dots, f(\tilde{x}_a)\} \\ P_y &= \{y_1, \dots, y_p\} \triangleq \{f(x_1), \dots, f(x_p)\} \end{aligned} \quad (4)$$

In this paper it is considered that both sets A_y and P_y are normalized between 0 and 1, according to extreme values of A_y (0 and 1 are assigned to the minimum and maximal value in each dimension respectively). These sets are re-normalized every time A_y is updated.

The main idea which is behind the “sphere operators” is to enhance representativity of the archive A by evaluating its regularity. If the solution samples regularly cover the set $\mathcal{P}$, they should be located in relation to their nearest neighbors such that the distances to them become of the same order of magnitude. Therefore, a parameter ρ , which has the meaning of a *reference domain radius* for each point, must be defined.

This parameter is employed in order to guide the algorithm operations, with the intent to generate points which approximately “represent” the region inside the sphere of such radius centered in that point. Any two neighbors should be separated, therefore, by at least 2ρ . In the regions in which the points are too densely distributed, an operation of decimation should be performed. In the regions in which the points are too sparsely distributed, new solution points should be found and included in the archive. The indication of what operation should be performed in each region is obtained via the comparison of the distances between the points with this parameter. The parameter ρ is dynamically adjusted during the algorithm execution, in order to reach a good dispersion of the sample points along the Pareto-set estimate, considering a reference value of the number of sample points that is to be found.

The sphere-control operators to be presented in this section are: (i) the archive size control, and (ii) the surface-filling crossover. The surface filling mutation operator should also be mentioned, presented in Takahashi et al. (2009), which is not discussed in detail here. The effect of those operators is illustrated in Fig. 2.

4.1. Archive size control

Consider the set $A = \{x_1, x_2, \dots, x_m, x_{m+1}, \dots, x_a\}$ in which the individual minima of the m objective functions have been put in the first m positions, and the remainder $a-m$ points have been ordered randomly. The set A_y is ordered correspondingly. The archive set size is controlled by Algorithm 2. After this operation, there will be no two points in A_y with pairwise distances smaller

than ρ , with the individual minima receiving precedence for being kept in the archive.

Algorithm 2. Pseudocode for archive size control [$\rho_Archive_Control$].

```

Inputs:  $A, A_y, \rho$ 
Outputs:  $A, A_y$ 
 $i \leftarrow 1$ 
while  $i \leq |A|$  do
   $A \leftarrow A - \{x_j \mid \|y_i - y_j\| < \rho, i \neq j\}$ 
   $A_y \leftarrow A_y - \{y_j \mid \|y_i - y_j\| < \rho, i \neq j\}$ 
   $i \leftarrow i + 1$ 
end while

```

Algorithm 3. Pseudocode for ρ Control [$Update_ \rho$].

```

Inputs:  $A, q, \rho$ 
Output:  $\rho$ 
Parameters:  $s, K_p, K_n$ 
 $a \leftarrow |A|$ 
if  $a > q$  then
   $e \leftarrow (a - q) / q$ 
  if  $e > s$  then
     $\Delta \leftarrow K_p \times s$ 
  else
     $\Delta \leftarrow K_p \times e$ 
  end if
   $\rho \leftarrow (1 + \Delta) \times \rho$ 
else
   $\rho \leftarrow K_n \times \rho$ 
end if

```

If this operation was executed with a pre-defined value of ρ , this could cause the size a of set A to become too large or too small, since the exact extension of the set $\mathcal{P}$ is not known *a priori*. On the other hand, the number of efficient solutions which are kept in the archive is a variable that can be measured easily, and this information can be used in order to adjust ρ , in order to reach a coverage of the set $\mathcal{P}$ with a “resolution” ρ .

The adjustment of parameter ρ takes into account a reference number of points that should be stored in the archive set A . Let such a reference number be denoted by q , and let $a = |A|$. The adjustment procedure is described in Algorithm 3. Default values can be indicated for the control parameters: $K_p = 0.6$, $s = 0.1$, $K_n = 0.9$. The initial value of ρ is defined such that q spheres of dimension $(m-1)$ occupy a volume equivalent to the unitary simplex of dimension $(m-1)$.

Algorithm 3 performs the adjustment of the value of ρ in the following way. If the reference number of points q is greater than the current number of points in the archive, a , then the value of ρ should be increased, in order to decrease the archive size. The percentage increase in ρ is given by Δ . Δ is calculated as a constant K_p times the relative error e between a and q . However, if e is greater than a threshold s , the value of Δ is saturated as K_p times this threshold s . This saturation is introduced in order to avoid a very fast change in the value of ρ , which could cause oscillations in the number a of points in the archive, possibly harming its convergence.

On the other hand, if the reference number of points q is smaller than the number a of points in the archive, then ρ should be decreased, in order to cause the archive size to increase. For the purpose of this reduction in ρ , a constant factor K_n is employed, instead of employing a factor proportional to the relative error e . This procedure avoids a very fast decreasing of the size of ρ in the

Fig. 2. The effect of sphere-control operators. (a) A set of samples of the Pareto-optimal set is unevenly distributed. (b) The archive size control eliminates points in the most crowded regions. (c) The surface-filling mutation and the surface-filling crossover operators generate new points that are located in the less crowded regions. (d) After a number of iterations, the Pareto-optimal sample set becomes evenly distributed, with a pre-specified number of samples. In this figure, the axis represents internal coordinates of the Pareto-set manifold.

first steps of the algorithm, when the number of archive solutions is much smaller than the reference number q , and also causes a small step decrease of ρ on any time that a decrease becomes necessary when the system is near the equilibrium. A diagram of such a closed-loop feedback control scheme is presented in Fig. 1. This algorithm resembles the PI (proportional-integral) controllers with control signal saturation, which are employed in industrial control systems.

It should be noticed that, as the control action over variable ρ is incremental, the net effect has the form of an *integral* control. This integral term in the PI controller is necessary in order to induce an error-less steady behavior in the closed-loop system. Indeed, if a simple P (proportional only) controller were adopted, the steady state value of the controlled variable a would not become equal to the reference value q . Explaining this point: consider the situation in which there is a positive error between a and q , meaning that ρ should be increased in order to make that error decrease. If the value of ρ was itself proportional to the error e then there would be a value of that error that, when multiplied by the constant K_p , would give rise to a value of ρ which would cause exactly that value of error to appear – causing an equilibrium situation that would not lead to further error reductions. However, if the value of the error e is used to calculate an increment Δ proportional to it, by adding this increment to the value of ρ will make the value of ρ to always increase for any positive e , eventually leading the error e to converge to zero. In this way, the PI controller structure is the simplest one that fulfills the requirement of producing exactly a evenly spaced solutions (most textbooks on classic control theory, for instance Ogata (2001), will present a detailed discussion about such effects).

The adoption of such a closed-loop control scheme makes the size of ρ to reach an equilibrium point by itself, avoiding the need of an *a priori* knowledge about such a parameter, and rendering the multiobjective genetic algorithm robust in relation to this parameter. Notice that even in the degenerate case of the Pareto-set surface being of dimension less than $m-1$ (one or more objectives being redundant), the algorithm still works as expected, forming an archive set of Pareto estimates which will still have q evenly spaced elements.

4.2. Surface-filling crossover

Define the function $v(\cdot)$ as

$$v(y_i) = |\{y_j | \|y_i - y_j\| < 3\rho\}| \quad (5)$$

which means the cardinality of the set of points from the archive set A_y which is inside a ball of radius 3ρ around the function argument y_j . Without loss of generality, consider that the set A_y is ordered in increasing order of $v(y_i)$:

$$A_y = \{y_i | v(y_i) \leq v(y_{i+1})\} \quad (6)$$

The set A receives the corresponding ordering, and $|A| = a$. After receiving the ordering as defined by (6), the archive A becomes arranged in such a way that the first solutions are the ones which have less neighbors in the objective space.

The parameter r , which is between 0 and 1, represents the proportion of elements in archive that will be submitted to the *surface-filling crossover* (SFXO) operation. This parameter is one of the control variables that will be adjusted dynamically by Algorithm 5. The $r \cdot |A|$ first solutions of A (after sorting) are chosen to perform the SFXO. The ordering is chosen such that a random solution is initially chosen to be the first parent p_1 . Afterward, the solution which is nearest to the current parent p_i is chosen, among the solutions which have not been selected yet. In the end, the procedure delivers a set of parents $\{p_1, p_2, \dots, p_{r \cdot |A|}\}$ which will suffer crossover. The SFXO will perform the crossover operations of $(p_1 \text{ vs. } p_2)$, $(p_2 \text{ vs. } p_3)$, $(p_3 \text{ vs. } p_4)$, ..., $(p_{r \cdot |A|-1} \text{ vs. } p_{r \cdot |A|})$. Therefore, $r \cdot |A| - 1$ crossover operations are performed each time. The SFXO operator is defined in Algorithm 4.

Algorithm 4. Pseudocode for SFXO [Surf_Fill_XO].

Inputs: A	▸ archive set
Inputs: r, β	▸ controlled parameters
Outputs: T, F	▸ set of enhanced solutions (decision and objective space)
Outputs: $k_1, k_2, k_3, k_4, k_5, k_6$	▸ counters used for parameter adaptation

Fig. 3. Different shapes of Beta distribution, for $\alpha=4$ and different β values. (a) $\beta=1.5$, (b) $\beta=4$, (c) $\beta=15$.

```

 $k_1, k_2, k_3, k_4, k_5, k_6 \leftarrow 0$ 
 $T, F \leftarrow \emptyset$ 
 $p \leftarrow \text{Sequence\_SFXO}(A, r)$ 
for  $i = 1$  to  $(r \cdot |A| - 1)$  do
   $b = \text{Beta}(\alpha, \beta)$ 
   $\hat{x}_i \leftarrow (1 - b) \cdot p_i + b \cdot p_{i+1}$ 
   $\hat{y}_i \leftarrow f(\hat{x}_i)$ 
   $vf \leftarrow \text{FALSE}$ 
  if  $\|\hat{y}_i - f(p_i)\| < \rho$  then
     $k_5 \leftarrow k_5 + 1$ 
     $vf \leftarrow \text{TRUE}$ 
  end if
  if  $\|\hat{y}_i - f(p_{i+1})\| < \rho$  then
     $k_6 \leftarrow k_6 + 1$ 
     $vf \leftarrow \text{TRUE}$ 
  end if
  if  $\hat{y}_i < f(p_i)$  or  $\hat{y}_i < f(p_{i+1})$  then
     $k_1 \leftarrow k_1 + 1$ 
     $T \leftarrow T \cup \hat{x}_i$ 
     $F \leftarrow F \cup \hat{y}_i$ 
  else if  $\hat{y}_i > f(p_i)$  or  $\hat{y}_i > f(p_{i+1})$  then
     $k_2 \leftarrow k_2 + 1$ 
  Else
    if  $vf = \text{FALSE}$  then
      if  $\nexists j | f(A^j) < \hat{y}_i$  then
         $k_3 \leftarrow k_3 + 1$ 
         $T \leftarrow T \cup \hat{x}_i$ 
         $F \leftarrow F \cup \hat{y}_i$ 
      else
         $k_4 \leftarrow k_4 + 1$ 
      end if
    end if
  end if
end for

```

In this algorithm, the command $p \leftarrow \text{Sequence_SFXO}(A, r)$ returns the sequence of parents $\{p_1, p_2, \dots, p_{r \cdot |A|}\}$ on which the SFXO operator will be executed.

The main idea of the SFXO is to interpolate solutions which have less neighbors in the objective space (i.e., less other points at a distance smaller than 3ρ). These individuals are subjected to crossovers which are intended to fill the gaps on the archive set, in order to generate a better description of the Pareto set. The parameter α of the Beta distribution has been kept fixed in $\alpha=4$, in order to generate concave probability distribution

functions (PDFs). On the other hand, the parameter β has been modeled as a controlled variable, which can vary dynamically inside the algorithm in order to adjust the Beta distribution: lower values of β lead the median of the PDF to approach 1 and higher values of Beta lead the median of the PDF to approach 0. The variable β therefore can be used to approximate the offspring solution to either p_1 or p_2 . The shapes of the PDF of Beta distribution for $\alpha=4$ and different values of β are shown in Fig. 3. Each new solution obtained by the surface-filling crossover is analyzed, with the results driving the following counters:

Counter k_1 : $\hat{x}_i$ dominates p_i or p_{i+1} ;

Counter k_2 : $\hat{x}_i$ is dominated by p_i or p_{i+1} ;

Counter k_3 : $\hat{x}_i$ is not dominated by neither p_i , p_{i+1} nor by any other solution in A ;

Counter k_4 : $\hat{x}_i$ is not dominated by neither p_i nor p_{i+1} , but it is dominated by some other solution in A ;

Counter k_5 : The distance between $\hat{x}_i$ and p_i is lower than ρ ;

Counter k_6 : The distance between $\hat{x}_i$ and p_{i+1} is lower than ρ .

This means that the counter k_i is increased each time a solution makes the respective condition to become true. These six counters are used to estimate the efficiency of the operator and to update r and β values. The pseudo-code for r and β tuning is shown in Algorithm 5.

Algorithm 5. Pseudocode for r and β control in SFXO [Update r, β].

```

Inputs:  $r, \beta$  ▷ controlled parameters
Inputs:  $k_1, k_2, k_3, k_4, k_5, k_6$  ▷ counters used for parameter adaptation

Outputs:  $r, \beta$ 
 $b_{exp} \leftarrow \log_{10}(\beta)$ 
 $r_{inv} \leftarrow \frac{1}{r}$ 
 $n_{SFXO} \leftarrow k_1 + k_2 + k_3 + k_4$  ▷ number of SFXO operations performed
 $S_{SFXO} \leftarrow k_1 + k_3$  ▷ number of successful SFXO operations
 $U_{SFXO} \leftarrow k_2 + k_4$  ▷ number of unsuccessful SFXO operations

if  $k_5 \geq 0.25 \cdot n_{SFXO}$  or  $k_6 \geq 0.25 \cdot n_{SFXO}$  then ▷  $\beta$  control
  if  $k_5 > k_6$  then
     $b_{exp} \leftarrow 0.90 \cdot b_{exp}$  ▷ approach the median to  $p_2$ 
  else
     $b_{exp} \leftarrow 1.10 \cdot b_{exp}$  ▷ approach the median to  $p_1$ 
  end if
end if

```

```

if  $S_{SFXO} \geq U_{SFXO}$  then                                ▷  $r$  control
     $r_{inv} \leftarrow 0.90 \cdot r_{inv}$                             ▷ increase the number of
                                                                SFXO operations
else
     $r_{inv} \leftarrow 1.10 \cdot r_{inv}$                         ▷ decrease the number of
                                                                SFXO operations
end if
if  $b_{exp} < 0.25$  then                                    ▷  $\beta$  controller saturation -
                                                                lower bound
     $b_{exp} \leftarrow 0.25$ 
else if  $b_{exp} > 1.25$  then                                ▷  $\beta$  controller saturation -
                                                                upper bound
     $b_{exp} \leftarrow 1.25$ 
end if
if  $r_{inv} > 10$  then                                       ▷  $r$  controller saturation -
                                                                lower bound
     $r_{inv} \leftarrow 10$ 
else if  $r_{inv} < 2$  then                                   ▷  $r$  controller saturation -
                                                                upper bound
     $r_{inv} \leftarrow 2$ 
end if
 $r \leftarrow \frac{1}{r_{inv}}$ 
 $\beta \leftarrow 10^{b_{exp}}$ 

```

The parameters β and r are updated according to the results observed in SFXO operations:

- When a significant part (more than 25%) of the offspring solutions are concentrated near p_1 , the Beta distribution is adjusted in such a way that, in next generation, the operator will have higher chance of generating solutions which are more far away from it. The same holds for the case of p_2 . In this way, the operator tends to generate new solutions which are located not too close to any parent solution.
- When the SFXO presents more successful than unsuccessful cases in one generation, the number of expected crossover operators is increased for the next iteration. On the other hand, when more unsuccessful operations are performed this number is decreased.

The adjustment of variables is always performed in small steps, by increasing or reducing 10% of the current value. This is adopted in order to avoid abrupt variations which could cause instability in the algorithm. Both controllers also have lower and upper bounds in the control action.

After the execution of SFXO, all the solutions which are not dominated by any solution of A and whose distance to p_1 and p_2 are higher than ρ are delivered by the SFXO as a set T , which is inserted in the population Q of the GA, in order to improve the quality of the current population.

Assuming that solutions which are near on the parameter space are usually not so far away on the objective space, it is reasonable to admit that a crossover operation would be suitable for filling the blank spaces within archive, performing the interpolation of solutions. It should be noticed that the surface-filling crossover operator is likely to generate new points that are located nearby the same manifold in which the parent points lie. This means that, especially in the case of Pareto-set manifolds of relatively low dimension embedded in high-dimensional decision variable spaces, the surface-filling crossover tends to be efficient in the task of finding new Pareto-optimal points. Indeed, in the tests conducted within this study, the surface-filling crossover has been found to be more efficient than the surface-filling mutation which

was proposed in Takahashi et al. (2009). Due to this, the surface-filling mutation has not been considered in the present work.

4.3. Time complexity

With regard to computational complexity, the procedure employed to perform archive size control is built upon three major (and more costly) operations: (i) extreme solution identification: before performing archive size control, it is necessary to identify the extreme points of the current Pareto-front approximation, in order to ensure that these points are not displaced during the operator execution. It can be performed in $\mathcal{O}(|A|)$ time, in which A is the archive that stores the current Pareto-set approximation; (ii) solution distance evaluation: the archive size control algorithm requires that the distances between all points in the archive are known. It is possible to accomplish such a calculation in $\mathcal{O}(|A|^2)$ time; (iii) Pareto filtering: the while-loop of Algorithm 2 can, in the worst case, compare each solution with all the $|A| - 1$ remaining ones. Therefore, the execution of such a loop is bounded to $\mathcal{O}(|A|^2)$ time complexity.

Since these three operations are the most costly in the procedure, it is possible to conclude that the archive size control can be executed in $\mathcal{O}(|A|^2)$ time. It should be noticed that the desired number of solutions in the archive is a user-defined parameter and, therefore, it is not affected neither by the problem dimension nor by the number of objectives.

The SFXO² executes, for a $r \cdot |A|$ number of times, a crossover operator that is, usually, very similar to the one employed in the original algorithm. Therefore, the complexity of the SFXO can be estimated as $r \cdot |A|$ times the computational complexity of the crossover operator. Since the size of $|A|$ is controlled, and r is bounded by a maximum value, the SFXO computational complexity is then controlled by the algorithm parameters.

However, it must be emphasized that, in most part of cases, the function evaluations are the main responsible for the algorithm processing time. In those cases, the additional time required to perform SFXO calculations (excluding function evaluation) could be ignored, since it would be insignificant when compared to the time required to evaluate the solutions generated by the operator itself. The number of solutions evaluated inside SFXO is counted as normal algorithm evaluations, in order to keep algorithm comparisons fair.

5. Numerical tests

In order to assess the suitability of the proposed methodology, a series of numerical tests has been conducted over a set of 18 benchmark functions. The description of those benchmark multi-objective optimization problems is presented in the Appendix.

In order to support the comparisons, the NSGA-II implementation found in PISA (Zitzler, 2010) has been translated to Matlab 7,³ in order to be used as benchmark. This algorithm employs exactly the same operators (simulated binary crossover and polynomial mutation, Deb and Agrawal, 1995) and structure of the original one. Such as in the PISA implementation, the probability distribution function of the operators are adjusted in order to generate real values only in the interval $\{0, 1\}$. A scheme that controls the size of the archive has been included in order to keep it at a pre-defined size. This scheme is based on the crowding distance assignment, and it is referred as **crowding_archive_reduction** in Section 3.1.

² The complexity of the procedure adopted to adjust ρ and β is not discussed here because it requires $\mathcal{O}(1)$ time.

³ Matlab is a registered mark of Mathworks.

Fig. 4. KUR10 results. (a) CM, (b) ΔM , (c) ISC.

This algorithm has been used also as the basis for the implementation of the sphere-control hybrid algorithm presented in this paper (SCMGA). In the SCMGA, the control of the archive size is performed using ρ -Archive_Control (Section 4.1) instead of crowding_archive_reduction. The surface filling crossover is also executed at each algorithm generation, as shown in Algorithm 1. The procedures that adjust ρ , β and r are also executed in the SCMGA variant.

The performances of these two algorithms on the benchmark problems considered here have been compared in order to estimate the effect of the sphere-control operators on the algorithm convergence. The performances of the algorithms, in each test problem, are compared through the following steps:

Performance comparison procedure.

1. Each algorithm is executed k times for the same parameter set and stop criterion;
2. Based on the minimum and maximum values achieved for each objective, considering all algorithms and runs, normalize the fronts between 0 and 1 (the values 0 or 1 are not necessarily present on each Pareto-front approximation);
3. For each run i of the k runs:
 - (a) Join the archive obtained by the original NSGA-II ($A_i^{NSGA-II}$) and the archive obtained by the SCMGA (A_i^{SCMGA}) into a single archive A ;
 - (b) Perform dominance on A and create A' with the non-dominated points of A ;
 - (c) Make $A_i^{NSGA-II} = A_i^{NSGA-II} \cap A'$ and $A_i^{SCMGA} = A_i^{SCMGA} \cap A'$;
 - (d) Employ some Pareto quality metric PQM for evaluating the convergence performances of the NSGA-II and SCMGA algorithms:
 - $q_i^{NSGA-II} = PQM(A_i^{NSGA-II}, A_i^{SCMGA})$;
 - $q_i^{SCMGA} = PQM(A_i^{SCMGA}, A_i^{NSGA-II})$.
4. Sort the $q^{NSGA-II}$ and q^{SCMGA} vectors in ascending order.

This process returns k quantiles associated with the convergence performance of the algorithms that are being considered. These quantiles can be used for performing a stochastic dominance analysis, as discussed in Carrano et al. (2010). From the stochastic dominance principle, it is possible to say that a multi-objective algorithm A_1 stochastically dominates another algorithm A_2 ($A_1 \succ A_2$) if, and only if

$$A_1 \succ A_2 : \begin{cases} q_i^{A_1} \geq q_i^{A_2} \quad \forall i \in 1, \dots, k \\ \exists i | q_i^{A_1} > q_i^{A_2} \end{cases} \quad (7)$$

Therefore, A_1 dominates A_2 if all k quantiles of A_1 are higher or equal the quantiles of A_2 and at least one quantile of A_1 is higher than the same quantile of A_2 . This kind of analysis provides an accurate comparison of the algorithm outcomes.

Although this process is considerably better than single run comparison or visual analysis, it still has a problem: for a moderate number of algorithm executions, the order in which the archives are compared can effect the vectors q , because the result of the dominance analysis is dependent on the sets that are being compared. This order dependence can give rise to inaccurate results, since two different executions of the proposed scheme could lead to different (or even opposite) results. In order to avoid this inconvenient, permutation test procedures have been used for performing algorithm analysis. In each permutation test, random permutations of the indexes of the archives obtained by the algorithms are generated. The archives are compared following such indexes, and the process is repeated for a pre-determined number of times P . Each permutation test provides a set of quantiles q for each algorithm. The final value of the quantile q_i is obtained by determining the median of all P values assigned to the quantile q_i in the permutation tests. This procedure avoids order biasing of the solutions, and reduces the variability between the responses provided by the method when comparing the same algorithms.

5.1. Pareto quality metrics

In this paper, three Pareto quality metrics (PQM) have been used to estimate the quality of the Pareto approximations delivered by the algorithm: the C metric (Zitzler and Thiele, 1998) for set coverage, the Δ metric (Deb et al., 2002) for set spacing and the integrated sphere counting metric (Silva et al., 2007), which tries to handle with both aspects at once.

C Metric (CM): The C metric, proposed by Zitzler and Thiele (1998), estimates the coverage of two Pareto set approximations, based on the dominance principle. Let $\mathcal{X}$ be the set of candidate decision vectors for a given problem and let $\mathcal{A} \subseteq \mathcal{X}$, $\mathcal{B} \subseteq \mathcal{X}$ be any two sets of decision vectors. The coverage of $\mathcal{A}$ with regard to $\mathcal{B}$, $C(\mathcal{A}, \mathcal{B})$, can be evaluated as follows:

$$C(\mathcal{A}, \mathcal{B}) = \frac{|\{a \in \mathcal{A} | \exists b \in \mathcal{B} : b \leq a\}|}{|\mathcal{A}|} \quad (8)$$

In this expression, the fraction gives the ratio of solutions from $\mathcal{A}$ that are dominated by solutions from $\mathcal{B}$. If $C(\mathcal{A}, \mathcal{B}) = 0$, then $\mathcal{A}$ does not have any dominated solutions when compared to $\mathcal{B}$; on the other hand, if $C(\mathcal{A}, \mathcal{B}) = 1$, then all solutions of $\mathcal{A}$ are dominated by at least one solution of $\mathcal{B}$. In this way, a lower value of C means a more suitable Pareto front approximation.

Δ Metric (ΔM): The Δ metric, proposed by Deb et al. (2002), is used to estimate how the solutions of the approximated Pareto-set are spaced into the objective domain. Let $\mathcal{X}$ be the set of candidate decision vectors for a given problem and let $\mathcal{A} \subseteq \mathcal{X}$ be any set of decision vectors. The Δ metric for $\mathcal{A}$, $\Delta(\mathcal{A})$, can be calculated as

Fig. 5. KUR20 results. (a) CM, (b) ΔM , (c) ISC. Fig. 6. KUR30 results. (a) CM, (b) ΔM , (c) ISC. Fig. 7. QUAD2 results. (a) CM, (b) ΔM , (c) ISC. Fig. 8. QUAD4 results. (a) CM, (b) ΔM , (c) ISC.

follows:

$$\Delta(\mathcal{A}) = \frac{\sum_{i=1}^m d_i^e + \sum_{i=1}^{|\mathcal{A}|} |d_i - \bar{d}|}{\sum_{i=1}^m d_i^e + |\mathcal{A}| \cdot \bar{d}} \quad (9)$$

in which m is the number of objectives; d_i^e is the Euclidean distance between the extreme points in $\mathcal{A}$ and in $\mathcal{X}^*$; d_i is the Euclidean distance between the point $i \in \mathcal{A}$ and its closest neighbor, in the objective space; $\bar{d}$ is the average of all distances d_i ; and $\mathcal{X}^*$ is the real Pareto set of the problem. From (9), it is straightforward to note that lower values of Δ are associated to better spaced Pareto-fronts.

Fig. 9. QUAD8 results. (a) CM, (b) ΔM , (c) ISC. Fig. 10. QUAD16 results. (a) CM, (b) ΔM , (c) ISC. Fig. 11. FONSECA results. (a) CM, (b) ΔM , (c) ISC. Fig. 12. VIENNET results. (a) CM, (b) ΔM , (c) ISC.

Integrated sphere counting (ISC): The integrated sphere counting is used to estimate the quality of the approximated Pareto-sets achieved by the algorithms under analysis. The procedure for ISC calculation is presented in Algorithm 6. In this algorithm, the symbol n_r is the number of radii for which the sphere counting is applied and r_d is the vector of radii to be evaluated. In all executions here, the values $n_r=11$ and $r_d=[0.01, \dots, 0.1]$ were adopted. Notice that, when comparing Pareto-estimates coming from two algorithms, the ISC metric is applied only to the points of

an algorithm that are non-dominated with regard to the points obtained by the other algorithm. In this way, the only relevant feature to be measured is the spread of solutions in the Pareto-estimate set. The ISC metric measures the sample effectiveness of each Pareto-set estimate under different sampling distances. The more regular is the distribution of samples, and the more fine-grained is the sampling, the ISC metric will be greater. In this sense, the ISC assigns higher values to better Pareto approximations.

Fig. 13. DTLZ1 results. (a) CM, (b) ΔM , (c) ISC. Fig. 14. DTLZ2 results. (a) CM, (b) ΔM , (c) ISC. Fig. 15. DTLZ3 results. (a) CM, (b) ΔM , (c) ISC. Fig. 16. DTLZ4 results. (a) CM, (b) ΔM , (c) ISC.

Algorithm 6. Pseudocode for the ISC algorithm.

```

1: procedure ISC ( $A, n_r, r_d$ )
2:    $ISC \leftarrow 0$ 
3:   for  $i=1$  to  $n_r$  do
4:      $\mathcal{C} \leftarrow A$ 
5:      $c_i \leftarrow 1$ 
6:      $r \leftarrow r_d(i)$ 
7:     Place a sphere of radius  $r$  centered at any point of  $\mathcal{C}$ 
8:     while  $|\mathcal{C}| > 0$  do
9:       Let  $\mathcal{R}$  denote the points of  $\mathcal{C}$  inside that sphere
10:       $\mathcal{C} \leftarrow \mathcal{C} - \mathcal{R}$ 
11:      Among the remaining points, center the sphere at
        the point which is closest to the previous center;
12:       $c_i \leftarrow c_i + 1$ 
13:    end while
14:     $ISC \leftarrow ISC + c_i$ 
15:  end for
16: end procedure

```

Fig. 17. DTLZ5 results. (a) CM, (b) ΔM , (c) ISC. Fig. 18. DTLZ6 results. (a) CM, (b) ΔM , (c) ISC. Fig. 19. DTLZ7 results. (a) CM, (b) ΔM , (c) ISC. Fig. 20. TSP2 results. (a) CM, (b) ΔM , (c) ISC.

These quality metrics have been chosen due to their efficiency and low computational complexity. Note that the comparison procedure described earlier in this section can be used with any of these three metrics. However, in the specific case of ΔM , the steps 3a–3c of the performance comparison procedure should not

be executed, since this metric requires that the input sets have the same number of elements. Besides, the permutation tests are not necessary for this metric, since it does not require pairwise comparisons.

The comparison of the algorithms has been performed using stochastic dominance, as shown in Eq. (7). A visual information of

Fig. 21. TSP3 results. (a) CM, (b) ΔM , (c) ISC.

Table 1

Processing time per run (in p.u.). $\bar{x}$ is the mean computation time and s is the standard deviation of computation time in 100 algorithm runs.

Instance	NSGA-II		SCMGA		Base (s)
	$\bar{x}/base$	$s/base$	$\bar{x}/base$	$s/base$	
KUR10	1.00	0.01	1.06	0.02	11.90
KUR20	1.00	0.01	1.06	0.01	19.95
KUR30	1.00	0.01	1.06	0.00	28.06
QUAD2	1.06	0.08	1.00	0.05	43.51
QUAD4	1.00	0.05	1.01	0.08	54.44
QUAD8	1.00	0.05	1.00	0.04	91.56
QUAD16	1.11	0.01	1.00	0.01	287.65
FONSECA	1.00	0.12	1.07	0.16	36.56
VIENNET	1.00	0.00	1.00	0.00	17.72
DTLZ1	1.00	0.02	1.06	0.01	18.83
DTLZ2	1.00	0.01	1.03	0.00	18.45
DTLZ3	1.00	0.01	1.07	0.01	18.99
DTLZ4	1.00	0.00	1.03	0.00	18.42
DTLZ5	1.00	0.01	1.03	0.00	18.55
DTLZ6	1.00	0.00	1.03	0.00	18.41
DTLZ7	1.00	0.00	1.04	0.00	18.16
TSP2	1.01	0.01	1.00	0.03	56.53
TSP3	1.00	0.01	1.03	0.02	68.37

Fig. 22. Single screw extruder.

this comparison is provided using box plots, in which the horizontal lines represent (from bottom to top): quantile 0.025, quantile 0.25, quantile 0.50, quantile 0.75 and quantile 0.975.

5.2. Numerical results

This section presents the results achieved by the NSGA-II and the SCMGA algorithms in the benchmark problems that are stated in the Appendix. The comparison of the results has been performed using the stochastic dominance procedure. Finally, the results of the comparisons are illustrated using box plots.

Both algorithms have been set using the same parameters:

- Population size: 100 individuals.
- Stop criterion – maximum number of function evaluations:
 - **QUAD2**: 10,000 evaluations.
 - **KUR10** and **QUAD4**: 20,000 evaluations.

- **FONSECA**, **VIENNET**, **DTLZ1**, **DTLZ2**, **DTLZ3**, **DTLZ4**, **DTLZ5**, **DTLZ6** and **DTLZ7**: 30,000 evaluations.
- **KUR20**: 35,000 evaluations.
- **QUAD8**: 40,000 evaluations.
- **KUR30**: 50,000 evaluations.
- **TSP2**: 75,000 evaluations.
- **QUAD16**: 80,000 evaluations.
- **TSP3**: 100,000 evaluations.
- Crossover probability (per pair): 0.90.
- Crossover η : 20.
- Mutation probability (per individual): $1/n$.
- Mutation η : 20.
- Maximum archive size:
 - **QUAD2** and **TSP2**: 50 individuals.
 - **KUR10**, **KUR20**, **KUR30**, **QUAD4**, **FONSECA**, **VIENNET**, **TSP3**, **DTLZ1**, **DTLZ2**, **DTLZ3**, **DTLZ4**, **DTLZ5**, **DTLZ6** and **DTLZ7**: 100 individuals.
 - **QUAD8**: 200 individuals.
 - **QUAD16**: 400 individuals.
- Number of algorithm runs: 100 (per instance).
- Number of permutation tests: 1000 (per instance).

The Δ metric (9) depends on the extreme points of the real Pareto-fronts. The exact extreme points have been used in the QUAD, KUR, FONSECA, VIENNET and DTLZ to evaluate such a metric, since these points are known from the literature. On the other hand, the exact mono-objective optima are not known for the TSP instances, since they have been generated at random. In order to surpass such a limitation, the best solution achieved for each objective, considering all algorithm runs, has been adopted as the respective extreme point.

Figs. 4–6 show the results observed for the instances **KUR10**, **KUR20** and **KUR30** respectively. Figs. 7–10 show the results observed for the instances **QUAD2**, **QUAD4**, **QUAD8** and **QUAD16**. Fig. 11 shows the result which has been observed for the instance **FONSECA**, and Fig. 12 shows the result which has been observed for the instance **VIENNET**. Figs. 13–19 show the results observed for the seven **DTLZ** instances. Finally, Figs. 20 and 21 show the results observed for the instances **TSP2** and **TSP3**.

From the box plots, it is easy to note that the SCMGA stochastically dominates the NSGA-II algorithm in most part of cases for all performance metrics (the quantiles of the SCMGA are better than the corresponding quantiles of the NSGA-II). The only exception is the coverage metric (CM) for the instance DTLZ7, in which NSGA-II achieved better results. However, it should be noticed that the proposed algorithm outperformed the original NSGA-II for both, ΔM and ISC, in the DTLZ7 instance. The analysis of the instances of KUR problem suggest that the SCMGA is less affected by the increase in the problem dimension than the NSGA-II, since the difference between the two algorithms increases with the problem dimension. In all instances of problem QUAD, the

Fig. 23. Thermo-mechanical functional process steps.

Table 2
Optimization runs.

Case	Optimization type	Decision variables	Objectives
1	Operating conditions	$N, T_{b1}, T_{b2}, T_{b3}$	Q, L_{melt}
2	Operating conditions	$N, T_{b1}, T_{b2}, T_{b3}$	Q, T_{melt}
3	Operating conditions	$N, T_{b1}, T_{b2}, T_{b3}$	$Q, WATS$
4	Operating conditions	$N, T_{b1}, T_{b2}, T_{b3}$	All
5	Screw geometry	L_1, L_2, H_1, H_3, P, e	$Q, Power$
6	Screw geometry	L_1, L_2, H_1, H_3, P, e	$Q, WATS$
7	Screw geometry	L_1, L_2, H_1, H_3, P, e	All
8	Both	$N, T_{b1}, T_{b2}, T_{b3}$	$Q, Power$
9	Both	$N, T_{b1}, T_{b2}, T_{b3}$	$Q, WATS$
10	Both	$N, T_{b1}, T_{b2}, T_{b3}$	All

Table 3
Objectives, aim of optimization and range of variation.

Objective	Aim of optimization	Range of variation
Output (kg/h)	Maximization	[1, 20]
Length for melting (m)	Minimization	[0.2, 0.9]
Melt temperature (°C)	Minimization	[150, 210]
Power consumption (W)	Minimization	[0, 9200]
WATS	Maximization	[0, 1300]

decision variable space has dimension 20, and the dimension of the objective space varies. The results are similar in all cases, what suggests that the number of objectives does not affect the relative efficiency between the algorithms. In a wide analysis, it is possible to see that the proposed algorithm can obtain a bigger amount of efficient solutions than the NSGA-II. Besides, the solutions achieved by the SCMGA are often better spaced into the objective domain, what is also highly desirable.

The computation times required by both algorithms are shown in Table 1. The simulations have been performed on a single core of an iMac 27 (model 2011), with Core i7 3.4 GHz processor and 16 GB RAM, using Matlab 2010. In this table, it is shown the mean and the standard deviation observed amongst the runs, in p.u. values. It is possible to note that the difference between the

algorithm processing times is not significant in any of the instances considered. In the worst case, the proposed algorithm is less than 10% slower than the original NSGA-II and it is faster in some instances. It means that the overhead caused by the employment of the self-controlled operators is not significant when compared to the other operations required by the algorithm. The reader should notice that such a difference would be even smaller in problems in which the time required to evaluate the function is high, such as several real-world problems as the application example presented next in this paper.

6. Application study: polymer extrusion process

This section presents a brief explanation about the polymer extrusion process, which constitutes the system to which the proposed methodology is applied.

Single screw extrusion is one of the most important polymer processing technologies, allowing the production of products such as pipes, film, profiles, and fibers. Basically this process consists in feeding a solid polymer at the beginning of the system (in the hopper), melting and homogenizing it and forcing the melted polymer to pass through a tool called the die that gives the final shape to the product to be obtained. Fig. 22 illustrates this procedure. The extruder is constituted by: (i) a hopper, where the solid polymer with the shape of pellets is fed; (ii) a heated barrel; (iii) an Archimedes-type screw rotating inside the barrel at a given speed (N); (iv) heater bands, with temperature defined by the operator and (v) a die.

The process performance depends on three different type of parameters: the polymer properties, the system geometry and the operating conditions. The different polymers are characterized by properties such as thermal (e.g., heat conduction coefficient, melting temperature and heat capacity), physical (e.g., friction coefficients and density) and rheological (which is a measure of the resistance of the polymer to the flow). Usually, the process is optimized in order to process a single polymer, thus the properties are, in this case, constant. In the most simple case, a conventional screw with three geometrical zones, as the one shown in Fig. 22,

Fig. 24. Empirical attainment functions: comparison between the RPSGA, NSGA-II and SCMGA algorithms.

is used. First, appears a screw section with constant depth (H_1), the feed section. Then, there is a compression section where the depth varies between H_1 and H_3 . Finally, there is a metering section with a constant but smaller depth (H_3). The screw is also characterized by the pitch (P) and the flight width (e). The operating conditions are the variables that are controlled by the operator of the machine. In the present case these variables are the screw rotation speed (N) and the barrel temperature profile (T_{b1} , T_{b2} and T_{b3}).

The main functions of the extruder are to transport the solid material from the hopper to the heated barrel zone, to melt the polymer, to homogenize and mix the melted polymer with the additives usually present and to create the necessary pressure which enables the polymer to pass through the die at the desired output. As stated above, the performance of the extruder depends on the polymer properties, system geometry and operating conditions. Taking this into account a thermo-mechanical environment is developed in which the polymer passes through different thermal and physical states (see Fig. 23): (i) first, the solids are fed into the hopper where, by action of gravity, they are transported inside the barrel (solids conveying in the hopper); (ii) then, by

action of the screw rotation and due to the friction between the screw and barrel walls the solid polymer is pressurized and a solid bed is formed and, simultaneously, the polymer is transported to the heated barrel zone (solids conveying in the screw); (iii) at this point, due the heat generated by friction and the heat conducted from the barrel a melt film is formed (delay zone); (iv) then, a specific melting mechanism develops, characterized by the existence of a melt pool and melt films around the solid bed (melting zone); (v) finally, the polymer is pressurized and is transported to the die (conveying zone).

The modeling of this process involves the linkage of all those functional zones using the appropriate boundary conditions. The program developed is based on finite differences, where central and backward differences are applied for the first order derivatives and the implicit Crank–Nicolson scheme is applied for the second order derivatives. The system of equations resulting from the analysis, momentum and energy equations, was solved using the method of Gauss elimination with pivoting. The stability of the solutions was assured taking into account the convergence of the velocity field developed, simultaneously with the convergence

Fig. 25. Results obtained for the optimization of the operating conditions. (a) Run 1: NSGA-II vs. RPSGA. (b) Run 1: RPSGA vs. SCMGA. (c) Run 2: NSGA-II vs. RPSGA. (d) Run 2: RPSGA vs. SCMGA. (e) Run 3: NSGA-II vs. RPSGA. (f) Run 3: RPSGA vs. SCMGA.

in temperature. More details of the modeling routine implemented can be found elsewhere (Gaspar-Cunha, 2009). Also, due to the complexity of the process, in each functional zone (as identified in Fig. 23) the existence of multiple zones is considered and solved simultaneously, i.e., the convergence must be assured simultaneously for the different zones.

The program developed is able to compute some important performance characteristics of the process as a function of the polymer properties, system geometry and operating conditions. The process performance is characterized by the mass output of

the machine (Q), the average melt temperature of the polymer at die exit (T_{melt}), the power consumption required to rotate the screw ($Power$), the capacity of pressure generation (P_{max}), the mixing capacity measure by the average of deformation ($WATS$) and the length of screw required to melt the polymer (L_{melt}). Those are the objectives that are employed in the definition of the multiobjective optimization problem studied here.

The interested reader may find further information about the simulation of polymer extrusion processes in such as Vergnes et al. (1998) and Cassagnau et al. (2007). The topic of optimization of

Fig. 26. Results obtained for the optimization of the operating conditions: run 4.

polymer extrusion processes has been also studied in several works, for instance [Smith et al. \(1998\)](#), [Sienz et al. \(2006\)](#), and [Lebaal et al. \(2009\)](#). It is also worthy to mention that, as the simulation of the polymer extrusion process is computationally expensive, the usage of meta-models (or surrogate models) may be useful for reducing the number of calls of the system numerical model. Some examples of usage of such models may be found, for instance, in [Wanner et al. \(2008\)](#) which presents a local meta-modeling approach, and in [Giri et al. \(2013\)](#) which presents a global approach.

7. Results: application to polymer extrusion

In this section, the proposed algorithm is employed for the study of variable setup in a polymer extrusion system. In the study conducted here, three algorithms are compared: the basic NSGA-II, the proposed SCMGA, and the RPSGA ([Gaspar-Cunha and Covas, 2004](#); [Gaspar-Cunha, 2009](#)). This last algorithm is also used here because it was employed, in former studies, for the multiobjective optimization of the same polymer extrusion system studied here, and therefore it provides some reference for the evaluation of the behavior of other algorithms in this problem. All the tests reported in this section were performed with a fixed budget of 20,000 function evaluations for each algorithm.

Three different types of studies were carried out as shown in [Table 2](#). First, only the operating conditions (i.e., N , T_{b1} , T_{b2} , and T_{b3}) were considered as decision variables – in cases 1–4. In a second set of studies, the aim was to optimize the screw geometry (i.e., L_1 , L_2 , H_1 , H_3 , P , and e) – in cases 5–7. Finally, both types of decision variables were considered – in the cases 8–10. Due to the complexity of the process and with the aim of understanding

better the variable interactions in the optimization procedure, in some preliminary studies only two objectives were considered in each run. In this case the mass output was considered the most important objective, in order to increase the profits, being used in all optimization runs. [Table 3](#) lists the objectives used, the aim of optimization and their range of variation.

A statistical method based on attainment functions was applied to compare the final population for all runs. The method attributes to each objective vector z a probability that this point z is attained in one single run. It is not possible to compute the true attainment function, but it can be estimated based on an approximation to set samples, i.e., different approximations obtained in different runs, denoted as empirical attainment function (EAF) ([Fonseca and Fleming, 1996](#)). The differences between two algorithms can be visualized by plotting the points in the objective space where the differences between the empirical attainment functions of the two algorithms are significant ([López-Ibáñez et al., 2010](#)).

[Fig. 24](#) shows a comparison for case 1 between NSGA-II, RPSGA and SCMGA. Ten different runs for case 1 were performed. For this particular case, NSGA-II is slightly better than RPSGA, but SCMGA is better than the NSGA-II, since there are some black dots that indicate that the SCMGA is better in at least 80% of the runs than NSGA-II. [Figs. 25–27](#) show the results obtained by the different methods (RPSGA, NSGA-II and SCMGA) for the remaining cases studied. [Fig. 25](#) shows that for the cases where the operating conditions were optimized the better performance of the SCMGA is mainly accomplished in the extreme limits of the objective functions. In case study 4, in which all five objectives are considered, the SCMGA was able to find Pareto-optimal solutions much more spread over the search space. The results for cases 7 and 10 are very similar to results obtained for case 4. Finally, [Fig. 27](#) shows the results for cases 6 and 9 where both objectives are

Fig. 27. Results obtained for the optimization of the screw geometry and both. (a) Run 6: screw geometry. (b) Run 9: both.

to be maximized. In these cases the differences between the two methods considered were not so evident due to the fact that now, in the extrusion process, the geometry of the system is allowed to vary. The effect of geometry variation is evident if these results were compared with the results of case 3 represented in Fig. 25(e) and (f).

The outcomes of the multiobjective optimization – the Pareto-optimal solutions – may be used by a decision-maker as a means to assist the choice of the parameter configuration to be employed in the manufacturing of a specific product. The Pareto-optimal set carries the information about the limits of performance of the plant for that task, including the information about the trade-offs involved in the choice between different Pareto-optimal solutions. In this specific case study, the trade-offs are related to energy consumption, production capacity and product quality. A good description of the Pareto-set covering all its extension with a uniform sampling is important in order to provide the full set of alternatives to the decision-maker. The SCMGA method was shown to be an effective way to find out good descriptions of the Pareto-set that were not possible to obtain by other algorithms, mainly when more than two objectives are considered. This suggests that the proposed methodology is suitable for dealing with problems in application domains similar to the one studied here.

8. Conclusion

The main contribution of this paper is the proposal of feedback-control techniques for performing parameter tuning within the operators of evolutionary algorithms. This concept has been demonstrated by the construction of feedback-control-based operators that are intended to enhance the description of Pareto-sets in multiobjective evolutionary algorithms. The proposed new operators, the *archive reduction operator* and the *space-filling crossover* were shown to present the predicted behavior.

The archive reduction plus the space-filling crossover were included in the canonical NSGA-II, with the hybrid algorithm (the SCMGA) used for the purpose of performing some comparisons with the original algorithm. In this way, some quantitative comparisons were performed, showing a significant enhancement of the resulting Pareto-set descriptions obtained by the hybrid algorithm, in several benchmark problems. The numerical experiments that have been conducted on those benchmark functions also showed that the computational overhead caused by the proposed operators is smaller than 10%, not considering the computational effort devoted to objective function evaluation.

The proposed SCMGA was employed in the multiobjective optimization of a polymer extrusion system. Some specific conclusions about the case study are: (i) The SCMGA was able to generate a much enhanced description of the Pareto-set of the problem, providing both a more uniform sampling and a more complete description of the whole extension of that set. (ii) It was also verified that the difference of the performance of SCMGA in comparison with the other algorithms was greater in the case of three objectives than in the case of two objectives.

Those features indicate that the proposed SCMGA may be a good choice as a tool for providing the detailed information about the Pareto-optimal set, in order to assist the decision-maker in tasks such as the choice of the configuration of a polymer extrusion machine.

Acknowledgments

This work was supported by the Brazilian agencies CNPq, CAPES and FAPEMIG. The authors also acknowledge the support by a Marie Curie International Research Staff Exchange Scheme Fellowship within the 7th European Community Framework Programme.

Appendix A. Description of benchmark problems

The benchmark multiobjective optimization problems employed in the numerical tests are described in this appendix.

Kursawe problem (Kursawe, 1991):

$$\mathcal{A}^* = \arg \min_{\mathbf{x}} \begin{cases} f_1 = \sum_{i=1}^{n-1} \left[-10 \cdot \exp \left(-0.2 \cdot \sqrt{x_i^2 + x_{i+1}^2} \right) \right] \\ f_2 = \sum_{i=1}^n [|x_i^{0.8}| + 5 \cdot \sin(x_i^3)] \end{cases} \quad (10)$$

$$\text{subject to } -5 \leq x_i \leq 5 \quad \forall i = 1, \dots, n \quad (11)$$

in which n is the number of problem variables.

Three instances of the Kursawe problem are considered here:

- **KUR10:** $n = 10$;
- **KUR20:** $n = 20$;
- **KUR30:** $n = 30$.

Quadratic problem:

$$\mathcal{X}^* = \arg \min_{\mathbf{X}} \begin{cases} f_1 = (X - C^1)' \cdot I \cdot (X - C^1) \\ f_2 = (X - C^2)' \cdot I \cdot (X - C^2) \\ \vdots \\ f_{16} = (X - C^{16})' \cdot I \cdot (X - C^{16}) \end{cases} \quad (12)$$

$$\text{subject to } -10 \leq x_i \leq 10 \quad \forall i = 1, \dots, n \quad (13)$$

in which n is the number of problem variables; I is the $n \times n$ identity matrix; C^i is the minimum of the quadratic function i :

$$C^i = \begin{cases} c_j^i = 1 & \text{if } j = i \\ c_j^i = 0 & \text{otherwise.} \end{cases}$$

Although quadratic problems are often easy, this example is useful to evaluate the performance of the proposed algorithm when the number of objective function increases. Four instances of the quadratic problem have been considered:

- **QUAD2**: Two objective problem, considering only the functions f_1 and f_2 ;
- **QUAD4**: Four objective problem, considering the functions f_1 – f_4 ;
- **QUAD8**: Eight objective problem, considering the functions f_1 – f_8 ;
- **QUAD16**: Sixteen objective problem, considering all functions.

In all instances it has been assumed $n=20$.

Fonseca and Fleming problem (Fonseca and Fleming, 1995):

$$\mathcal{X}^* = \arg \min_{\mathbf{X}} \begin{cases} f_1 = 1 - \exp \left[- \sum_{i=1}^n \left(x_i - \frac{1}{\sqrt{n}} \right)^2 \right] \\ f_2 = 1 - \exp \left[- \sum_{i=1}^n \left(x_i + \frac{1}{\sqrt{n}} \right)^2 \right] \end{cases} \quad (14)$$

$$\text{subject to } -4 \leq x_i \leq 4 \quad \forall i = 1, \dots, n \quad (15)$$

in which n is the number of problem variables.

It has been considered a single instance of the Fonseca and Fleming problem:

- **FONSECA**: $n=15$.

Viennet problem (Viennet, 1996):

$$\mathcal{X}^* = \arg \min_{\mathbf{X}} \begin{cases} f_1 = 0.5 \cdot (x_1^2 + x_2^2) + \sin(x_1^2 + x_2^2) \\ f_2 = \frac{(3x_1 - 2x_2 + 4)^2}{8} + \frac{(x_1 - x_2 + 1)^2}{27} + 15 \\ f_3 = \frac{1}{x_1^2 + x_2^2 + 1} - 1.1 \cdot \exp(-(x_1 + x_2)^2) \end{cases} \quad (16)$$

$$\text{subject to } -3 \leq x_i \leq 3 \quad \forall i = 1, 2 \quad (17)$$

A single instance of the Viennet problem has been considered, which has been labeled as

- **VIENNET**.

DTLZ problems (Deb et al., 2001):

All the seven unconstrained instances of the DTLZ family have been tested in this work, as follows.

DTLZ1: $n=7, k=5, M=3$:

$$\min \begin{cases} f_1(\mathbf{x}) = 0.5[1 + g_1(\mathbf{x})]x_1x_2 \\ f_2(\mathbf{x}) = 0.5[1 + g_1(\mathbf{x})]x_1 \cdot (1 - x_2) \\ f_3(\mathbf{x}) = 0.5[1 + g_1(\mathbf{x})](1 - x_1) \end{cases} \quad (18)$$

DTLZ2: $n=12, k=10, M=3$:

$$\min \begin{cases} f_1(\mathbf{x}) = [1 + g_2(\mathbf{x})] \cos(x_1\pi/2) \cos(x_2\pi/2) \\ f_2(\mathbf{x}) = [1 + g_2(\mathbf{x})] \cos(x_1\pi/2) \sin(x_2\pi/2) \\ f_3(\mathbf{x}) = [1 + g_2(\mathbf{x})] \sin(x_1\pi/2) \end{cases} \quad (19)$$

DTLZ3: $n=12, k=10, M=3$:

$$\min \begin{cases} f_1(\mathbf{x}) = [1 + g_1(\mathbf{x})] \cos(x_1\pi/2) \cos(x_2\pi/2) \\ f_2(\mathbf{x}) = [1 + g_1(\mathbf{x})] \cos(x_1\pi/2) \sin(x_2\pi/2) \\ f_3(\mathbf{x}) = [1 + g_1(\mathbf{x})] \sin(x_1\pi/2) \end{cases} \quad (20)$$

DTLZ4: $n=12, k=10, M=3, \alpha=100$:

$$\min \begin{cases} f_1(\mathbf{x}) = [1 + g_2(\mathbf{x})] \cos(x_1^\alpha\pi/2) \cos(x_2^\alpha\pi/2) \\ f_2(\mathbf{x}) = [1 + g_2(\mathbf{x})] \cos(x_1^\alpha\pi/2) \sin(x_2^\alpha\pi/2) \\ f_3(\mathbf{x}) = [1 + g_2(\mathbf{x})] \sin(x_1^\alpha\pi/2) \end{cases} \quad (21)$$

DTLZ5: $n=12, k=10, M=3$:

$$\min \begin{cases} f_1(\mathbf{x}) = [1 + g_2(\mathbf{x})] \cos(\theta_1(\mathbf{x})\pi/2) \cos(\theta_2(\mathbf{x})\pi/2) \\ f_2(\mathbf{x}) = [1 + g_2(\mathbf{x})] \cos(\theta_1(\mathbf{x})\pi/2) \sin(\theta_2(\mathbf{x})\pi/2) \\ f_3(\mathbf{x}) = [1 + g_2(\mathbf{x})] \sin(\theta_1(\mathbf{x})\pi/2) \end{cases} \quad (22)$$

DTLZ6: $n=12, k=10, M=3$:

$$\min \begin{cases} f_1(\mathbf{x}) = [1 + g_3(\mathbf{x})] \cos(\Theta_1(\mathbf{x})\pi/2) \cos(\Theta_2(\mathbf{x})\pi/2) \\ f_2(\mathbf{x}) = [1 + g_3(\mathbf{x})] \cos(\Theta_1(\mathbf{x})\pi/2) \sin(\Theta_2(\mathbf{x})\pi/2) \\ f_3(\mathbf{x}) = [1 + g_3(\mathbf{x})] \sin(\Theta_1(\mathbf{x})\pi/2) \end{cases} \quad (23)$$

DTLZ7: $n=22, k=20, M=3$:

$$\min \begin{cases} f_1(\mathbf{x}) = x_1 \\ f_2(\mathbf{x}) = x_2 \\ f_3(\mathbf{x}) = [1 + g_4(\mathbf{x})]h(\mathbf{x}) \end{cases} \quad (24)$$

For these problems:

$$g_1(\mathbf{x}) = 100\{|\mathbf{x}| + \sum_{i=1}^n (x_i - 0.5)^2 - \cos[20\pi(x_i - 0.5)]\} \quad (25)$$

$$g_2(\mathbf{x}) = \sum_{i=1}^n (x_i - 0.5)^2 \quad (26)$$

$$g_3(\mathbf{x}) = \sum_{i=1}^n x_i^{0.1} \quad (27)$$

$$g_4(\mathbf{x}) = 1 + \frac{9}{|\mathbf{x}|} \sum_{i=1}^n x_i \quad (28)$$

$$\theta_i(\mathbf{x}) = \frac{\pi}{4[1 + g_2(\mathbf{x})]}[1 + 2g_2(\mathbf{x})x_i] \quad (29)$$

$$\Theta_i(\mathbf{x}) = \frac{\pi}{4[1 + g_3(\mathbf{x})]}[1 + 2g_3(\mathbf{x})x_i] \quad (30)$$

$$h(\mathbf{x}) = 3 - \sum_{i=1}^2 \left\{ \frac{f_i(\mathbf{x})}{1 + g(\mathbf{x})} \{1 + \sin[3\pi f_i(\mathbf{x})]\} \right\} \quad (31)$$

$$\text{subject to } 0 \leq x_i \leq 1 \quad \forall i \in 1, \dots, n \quad (32)$$

The DTLZ problems are scalable, which means that it is possible to employ them with any number of objective functions. The number of objectives in such problems is defined by two parameters: n , which is the number of decision variables, and; k , which is a parameter chosen *a priori*. The k values have been chosen such

as recommend on the original reference and the n values have been set in order to build problems with three objective functions.

Traveling salesman problem (Schrijver, 2005):

$$\mathcal{X}^* = \arg \min_{\mathcal{X}} \begin{cases} f_1 = x_{ij} \cdot c_{ij}^1 \\ f_2 = x_{ij} \cdot c_{ij}^2 \\ f_3 = x_{ij} \cdot c_{ij}^3 \end{cases} \quad (33)$$

$$\text{subject to } \begin{cases} \sum_{i=1}^n x_{ij} = 1 \quad \forall i = 1, \dots, n \\ \sum_{j=1}^n x_{ij} = 1 \quad \forall j = 1, \dots, n \\ x_{ij} \in \{0, 1\} \quad \forall i, j = (1, \dots, n) \times (1, \dots, n) \end{cases} \quad (34)$$

in which n is the number of cities; x_{ij} is a binary decision variable which is 1 when the cities i and j are directly connected or 0 otherwise; C^k is the $n \times n$ cost matrix k in which each cell c_{ij} defines the cost of connecting the cities i and j directly.

The only difference between the three objective functions shown in (33) is the cost matrix C^k . In a practical case it could be interpreted as three cost matrices that describe different physical quantities, such as distance, time and fuel consumption.

Two instances of the traveling salesman problem (TSP) have been considered:

- **TSP2:** Two objective problem, considering only the functions f_1 and f_2 ;
- **TSP3:** Three objective problem, considering all the three functions.

In both instances it has been assumed that $n=50$ (50 cities).

Solution encoding and decoding: In all problems described above, the solutions have been encoded in real vectors $V = [1 \times n]$, in which each cell v_i can assume a continuous value in the interval $\{0, 1\}$.

In the continuous problems (Kursawe, Quadratic, Fonseca and Fleming, Viennet and DTLZs) the solutions have been decoded to the problem domain by using the following equation:

$$x_i = L_i + v_i \cdot (U_i - L_i) \quad (35)$$

in which x_i is the real value of the i -th variable in the problem domain, and L_i and U_i are the lower and upper bounds of the i -th variable respectively.

In the case of the TSP, the solutions have been decoded using the random keys encoding scheme (Bean, 1994). This procedure transforms a real vector $1 \times n$ in a permutation of n elements. This permutation is used as a candidate solution for the TSP (sequence of city visitations). With this trick, the same Euclidean distance of real variable spaces could be used in this combinatorial problem.

References

Auger, A., Bader, J., Brockhoff, D., Zitzler, E., 2012. Hypervolume-based multi-objective optimization: theoretical foundations and practical implications. *Theor. Comput. Sci.* 425, 75–103.

Bean, J.C., 1994. Genetic algorithms and random keys for sequencing and optimization. *ORSA J. Comput.* 6, 154–160.

Bui, L.T., Abbass, H.A., Essam, D., 2009. Local models—an approach to distributed multi-objective optimization. *Comput. Optim. Appl.* 42 (1), 105–139.

Bui, L.T., Deb, K., Abbass, H.A., Essam, D., 2008. Interleaving guidance in evolutionary multi-objective optimization. *J. Comput. Sci. Technol.* 23, 44–63.

Carrano, E.G., Takahashi, R.H.C., Fonseca, C.M., Neto, O.M., 2010. Nonlinear network optimization—an embedding vector space approach. *IEEE Trans. Evol. Comput.* 14 (2), 206–226.

Cassagnau, P., Bounor-Legare, V., Fenouillet, F., 2007. Reactive processing of thermoplastic polymers: a review of the fundamental aspects. *Int. Polym. Process.* 22, 218–258.

Deb, K., Agrawal, R.B., 1995. Simulated binary crossover for continuous search spaces. *Complex Syst.* 9, 115–148.

Deb, K., Pratap, A., Agarwal, S., Meyarivan, T., 2002. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Trans. Evol. Comput.* 6, 182–197.

Deb, K., Thiele, L., Laumanns, M., Zitzler, E., 2001. Scalable Test Problems for Evolutionary Multi-Objective Optimization. Technical Report TIK—Technical Report No. 112. Institut für Technische Informatik und Kommunikationsnetze—ETH Zürich, Zürich, Switzerland.

Fonseca, C.M., Fleming, P.J., 1995. An overview of evolutionary algorithms in multiobjective optimization. *Evol. Comput.* 7 (3), 205–230.

Fonseca, C.M., Fleming, P.J., 1996. On the performance assessment and comparison of stochastic multiobjective optimizers. In: *PPSN IV: Proceedings of the 4th International Conference on Parallel Problem Solving from Nature*, pp. 584–593.

Gaspar-Cunha, A., 2009. Modelling and Optimisation of Single Screw Extrusion Using Multi-Objective Evolutionary Algorithms. Lambert Academic Publishing, Köln, Germany.

Gaspar-Cunha, A., Covas, J.A., 2004. RPSGAe—reduced Pareto set genetic algorithm: application to polymer extrusion. In: *Gandibleux, X., Sevaux, M., Sörensen, K., T'kindt, V. (Eds.), Metaheuristics for Multiobjective Optimisation. Lecture Notes in Economics and Mathematical Systems*, vol. 535. Springer, Berlin, pp. 221–249.

Giri, B.K., Hakanen, J., Miettinen, K., Chakraborti, N., 2013. Genetic programming through bi-objective genetic algorithms with a study of a simulated moving bed process involving multiple objectives. *Appl. Soft Comput.* 13, 2613–2623.

Jebbari, K., Bouroumi, A., Ettouhami, A., 2013. Parameters control in GAs for dynamic optimization. *Int. J. Comput. Intell. Syst.* 6 (1), 47–63.

Kim, J.H., Han, J.H., Kim, H.Y., Choi, S.H., Kim, E.S., 2012. Preference-based solution selection algorithm for evolutionary multiobjective optimization. *IEEE Trans. Evol. Comput.* 16 (1), 20–34.

Kukkonen, S., Deb, K., 2006. Improved pruning of non-dominated solutions based on crowding distance for bi-objective optimization problems. In: *Proceedings of the IEEE International Congress on Evolutionary Computation*.

Kursawe, F., 1991. A variant of evolution strategies for vector optimization. In: *Proceedings of the Parallel Problem Solving from Nature*.

Lebaal, N., Schmidt, F., Puissant, S., 2009. Design and optimization of three-dimensional extrusion dies, using constraint optimization algorithm. *Finite Elem. Anal. Des.* 45, 333–340.

Lin, Q.Z., Chen, J.Y., 2013. A novel micro-population immune multiobjective optimization algorithm. *Comput. Oper. Res.* 40 (6), 1590–1601.

López-Ibáñez, M., Paquete, L., Stützle, T., 2010. Exploratory analysis of stochastic local search algorithms in biobjective optimization. In: *Bartz-Beielstein, T., Chiarandini, M., Paquete, L., Preuss, M. (Eds.), Experimental Methods for the Analysis of Optimization Algorithms*. Springer, Berlin, pp. 209–222.

McGinley, B., Maher, J., O'Riordan, C., Morgan, F., 2011. Maintaining healthy population diversity using adaptive crossover, mutation, and selection. *IEEE Trans. Evol. Comput.* 15 (5), 692–714.

Ogata, K., 2001. *Modern Control Engineering*, fourth ed. Prentice-Hall, Upper Saddle River, NJ.

Paszkowicz, W., 2013. Genetic algorithms, a nature-inspired tool. Part II: a survey of applications in materials science and related fields. *Mater. Manuf. Process.* 28, 708–725.

Schrijver, A., 2005. On the history of combinatorial optimization (till 1960). In: *Aardal, K., Nemhauser, G.L., Weismantel, R. (Eds.), Handbook of Discrete Optimization*. Elsevier, Amsterdam, Netherlands, pp. 1–68.

Sienz, J., Bates, S.J., Pittman, J.E.T., 2006. Flow restrictor design for extrusion slit dies for a range of materials: simulation and comparison of optimization techniques. *Finite Elem. Anal. Des.* 42, 430–453.

Silva, V.L.S., Wanner, E.F., Cerqueira, S.A.A.G., Takahashi, R.H.C., 2007. A new performance metric for multiobjective optimization: the integrate sphere counting. In: *Proceedings of the IEEE Congress on Evolutionary Computation*. Singapore.

Sinha, A., Saxena, D.K., Deb, K., Tiwari, A., 2013. Using objective reduction and interactive procedure to handle many-objective optimization problems. *Appl. Soft Comput.* 13 (1), 415–427.

Smith, D.E., Tortorelli, D.A., Tucker, C.L., 1998. Optimal design for polymer extrusion. Part I: sensitivity analysis for nonlinear steady-state systems. *Comput. Methods Appl. Mech. Eng.* 167, 283–302.

Takahashi, R.H.C., Carrano, E.G., Wanner, E.F., 2011. On a stochastic differential equation approach for multiobjective optimization up to Pareto-criticality. In: *Proceedings of the 6th International Conference on Evolutionary Multi-Criterion Optimization (EMO 2011)*. Ouro Preto, Brazil, pp. 61–75.

Takahashi, R.H.C., Guimaraes, F.G., Wanner, E.F., Carrano, E.G., 2009. Feedback-control operators for evolutionary multiobjective optimization. In: *Proceedings of the 5th International Conference on Evolutionary Multi-Criterion Optimization (EMO 2009)*. Nantes, France, pp. 66–80.

Takahashi, R.H.C., Palhares, R.M., Dutra, D.A., Gonçalves, L.P.S., 2004. Estimation of Pareto sets in the mixed $\mathcal{H}_2/\mathcal{H}_\infty$ control problem. *Int. J. Syst. Sci.* 35 (1), 55–67.

Vasconcellos, J.A., Ramirez, J.A., Takahashi, R.H.C., Saldanha, R.R., 2001. Improvements in genetic algorithms. *IEEE Trans. Magn.* 37 (5), 3414–3417.

Vergnes, B., Della Valle, G., Delamare, L., 1998. A global computer software for polymer flows in corotating twin screw extruders. *Polym. Eng. Sci.* 38, 1781–1792.

Viennet, R., 1996. Multicriteria optimization using a genetic algorithm for determining the Pareto Set. *Int. J. Syst. Sci.* 27, 255–260.

- Wanner, E.F., Guimaraes, F.G., Takahashi, R.H.C., Fleming, P.J., 2008. Local search with quadratic approximations into memetic algorithms for optimization with multiple criteria. *Evol. Comput.* 16 (2), 185–224.
- Zitzler, E., 2010. PISA—A platform and programming language independent interface for search algorithms. Available from: <http://www.tik.ee.ethz.ch/sop/pisa/> (last accessed 14.02.2010).
- Zitzler, E., Thiele, L., 1998. Multiobjective optimization using evolutionary algorithms- a comparative case study. In: Eiben, A.E., Back, T., Schoenauer, M., Schwefel, H.P. (Eds.), *Proceedings of the International Conference on Parallel Problem Solving from Nature (PPSN)*.
- Zitzler, E., Thiele, L., Bader, J., 2010. On set-based multiobjective optimization. *IEEE Trans. Evol. Comput.* 14 (1), 58–79.
- Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C.M., da Fonseca, V.G., 2003. Performance assessment of multiobjective optimizers: an analysis and review. *IEEE Trans. Evol. Comput.* 7 (2), 117–132.