Applied Soft Computing 12 (2012) 589–599

Contents lists available at SciVerse ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

A multi-objective approach for the motion planning of redundant manipulators

Maria da Graça Marcos[a,b,*], J.A. Tenreiro Machado[a], T.-P. Azevedo-Perdicoúlis[b]

[a] Department of Electrical Engineering, Institute of Engineering, Polytechnic Institute of Porto, Porto, Portugal
[b] Department of Mathematics, University of Trás-os-Montes and Alto Douro, Vila Real, Portugal

ARTICLE INFO

Article history:
Received 5 March 2010
Received in revised form 5 April 2011
Accepted 1 November 2011
Available online 16 November 2011

PACS:
02.30.Nw
45.40.Ln
02.30.Zz

Keywords:
Redundant manipulators
Robots
Kinematics
Multi-objective genetic algorithms
Trajectory planning

ABSTRACT

Kinematic redundancy occurs when a manipulator possesses more degrees of freedom than those required to execute a given task. Several kinematic techniques for redundant manipulators control the gripper through the pseudo-inverse of the Jacobian, but lead to a kind of chaotic inner motion with unpredictable arm configurations. Such algorithms are not easy to adapt to optimization schemes and, moreover, often there are multiple optimization objectives that can conflict between them. Unlike single optimization, where one attempts to find the best solution, in multi-objective optimization there is no single solution that is optimum with respect to all indices. Therefore, trajectory planning of redundant robots remains an important area of research and more efficient optimization algorithms are needed. This paper presents a new technique to solve the inverse kinematics of redundant manipulators, using a multi-objective genetic algorithm. This scheme combines the closed-loop pseudo-inverse method with a multi-objective genetic algorithm to control the joint positions. Simulations for manipulators with three or four rotational joints, considering the optimization of two objectives in a workspace without and with obstacles are developed. The results reveal that it is possible to choose several solutions from the Pareto optimal front according to the importance of each individual objective.

1. Introduction

Kinematic redundancy occurs when a manipulator possesses more degrees of freedom than the required to execute a given task. In this case the inverse kinematics admits an infinite number of solutions, and a criterion to select one of them is required. Most of the research on redundancy deals with the use of these extra degrees of freedom and is referred to in the literature as the resolution of redundancy [1].

A class of techniques for solving the kinematics of redundant manipulators that was suggested controls the end-effector indirectly, through the rates at which the joints are driven, using the pseudo-inverse of the Jacobian (see, for instance, [2]). The pseudo-inverse of the Jacobian matrix guarantees an optimal reconstruction of the desired end-effector velocity – in the least-squares sense – with the minimum-norm joint velocity. However, even though the joint velocities are instantaneously minimized, there is no guarantee that the kinematic singularities are avoided [3]. Moreover, this method has the generally undesirable property that repetitive end-effector motions do not necessarily yield repetitive joint motions. Klein and Huang [4] were the first to observe this phenomenon for the case of the pseudo-inverse control of a planar three-link manipulator. Baillieul [5] proposed a modified Jacobian matrix called the extended Jacobian matrix. The extended Jacobian is a square matrix that contains the additional information necessary to optimize a certain function. The inverse kinematic solutions are obtained through the inverse of the extended Jacobian. The algorithms, based on the computation of the extended Jacobian matrix, have a major advantage over the pseudo-inverse techniques, because they are locally cyclic [6]. A large volume of research has been produced in the last few years in this topic [7–10]. For example, Zhang et al. [11] solve the joint angle drift problem by means of a dual-neural-network based quadratic-programming approach.

One optimization method that is gaining popularity for solving complex problems in robotics is the Genetic Algorithm (GA) approach. GAs are population-based stochastic and global search methods. Their performance is superior to that revealed by classical optimization techniques [12] and has been used successfully in robot path planning.

GAs was first introduced in robot motion planning by Parker et al. [13] that used GAs to position the end-effector of a robot at a target location, while minimizing the largest joint displacement. This method has some shortcomings, such as the lack of precision and is affected by the values of the weights. Arakawa et al. [14] developed a virus-evolutionary GA with subpopulations (VEGAS)

* Corresponding author at: Department of Mathematics, Institute of Engineering, Polytechnic Institute of Porto, Rua Dr. Antonio Bernardino de Almeida 431, 4200-072 Porto, Portugal. Tel.: +351 22 8340500; fax: +351 22 8321159.
E-mail address: mgm@isep.ipp.pt (M.d.G. Marcos).

1568-4946/$ – see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.asoc.2011.11.006

for the trajectory generation, composed of a host population and a virus population that minimizes total energy. The operators of crossover, mutation, virus infection and selection are executed in each subpopulation independently. Kubota et al. [15] studied a hierarchical trajectory planning using a virus-evolutionary GA and running simultaneously two processes. One process calculates some manipulator collision-free positions and the other generates a collision-free trajectory by combining these intermediate positions. de la Cueva and Ramos [16] proposed a GA for planning paths without collisions for two robots, both redundant and non-redundant, sharing the same workspace. The GA works directly over the task space adopting the direct kinematics. Each robot is associated with one population and each string of a population represents a complete robot path. Nishimura et al. [17] developed a motion planning method using an artificial potential field and a GA for a hyper-redundant manipulator whose workspace includes several obstacles. The motion planning is divided into two sub problems. The first is the "Path planning" that generates a trajectory leading the tip of manipulator to the goal without collisions, using the artificial potential field concept. The second consists in the "Collision-free sequence generation" that generates a sequence of movements by which distinct parts of the manipulator can avoid collisions with the obstacles. McAvoy and Sangolola [18] proposed an approach with GAs for optimal point-to-point motion planning of kinematically redundant manipulators. Their approach combines B-spline curves with GAs, for obtaining the optimal solution. Peng and Wei [19] presented the ASAGA trajectory planning method of redundant manipulators by combining a stochastic search algorithm (simulated annealing algorithm) and a GA. In the ASAGA the selection, crossover and mutation operators are adjusted by using an adaptive mechanism based on the fitness value. Zhang et al. [20] developed an algorithm to solve the inverse kinematics of a flexible macro–micro manipulator system which combines a GA and a neural network. Pires et al. [21] proposed a multi-objective GA, when considering up to five simultaneous objectives, to generate manipulator trajectories and for obstacle avoidance. Castillo et al. [22] applied single-objective and multi-objective GAs to the problem of offline point-to-point autonomous mobile robot path planning. The multi-objective problem, solved based on Pareto optimality, optimizes two criteria: the length and the difficulty of the path. Saravanan et al. [23] used NSGA-II and MODE to obtain optimal trajectory planning in the presence of obstacles for an industrial robot. To select the best optimal solution from the Pareto optimal front two methods are used: normalized weighted objective functions and average fitness factor method. In the first, the multiple objective functions are combined into one scalar value using a weight vector. In the second, a fitness factor is calculated for each objective function and the solution that has the highest average fitness factor value is selected. Saravanan et al. [24] applied MOGA, NSGA-II and MODE to obtain optimum geometrical dimensions of three types of robot gripper mechanisms. We can find considerable information in the work of Masehian and Sedighizadeh [25] that made a chronological review of classic and heuristic approaches in robot motion planning.

In this paper, we propose a modified multi-objective GA to solve the inverse kinematics of redundant manipulators, while considering the problems of repeatability, precision and obstacles in the workspace. Having these ideas in mind, the paper is organized as follows. Sections 2 and 3 introduce the fundamentals of the kinematics of redundant manipulators and the main concepts supporting the GAs, adopted in the rest of the paper. Based in these concepts, Section 4 presents the new closed-loop multi-objective GA (CLMOGA) and the open-loop GA (OLGA). Section 5 analyzes the simulation results in a workspace without and with obstacles. Finally, Section 6 draws the main conclusions.

2. Kinematics of redundant manipulators

We consider a manipulator with n degrees of freedom whose joint variables are denoted by $\mathbf{q} = [q_1, q_2, \ldots, q_n]^T$. We assume that the class of tasks we are interested in can be described by m variables, $\mathbf{x} = [x_1, x_2, \ldots, x_m]^T$, $m < n$, and that the relation between $\mathbf{q}$ and $\mathbf{x}$ is given by the direct kinematics:

$$\mathbf{x} = f(\mathbf{q}) \tag{1}$$

The differential kinematics was introduced by Whitney [26] that proposed the use of differential relationships to solve for the joint motion from the Cartesian trajectory of the end-effector. Differentiating (1) with respect to time yields:

$$\dot{x} = \mathbf{J}(\mathbf{q})\dot{q} \tag{2}$$

where $\dot{\mathbf{x}} \in \mathbb{R}^m$, $\dot{\mathbf{q}} \in \mathbb{R}^n$, and $\mathbf{J}(\mathbf{q}) = \partial f(\mathbf{q})/\partial \mathbf{q} \in \mathbb{R}^{m \times n}$. Hence, it is possible to calculate a path $\mathbf{q}(t)$ in terms of a prescribed trajectory $\mathbf{x}(t)$ in the operational space.

Eq. (2) can be inverted to provide a solution in terms of the joint velocities:

$$\dot{\mathbf{q}} = \mathbf{J}^{\#}(\mathbf{q})\dot{\mathbf{x}} \tag{3}$$

where $\mathbf{J}^{\#}$ is the Moore–Penrose generalized inverse of the Jacobian $\mathbf{J}$ [2,27].

When the manipulator is redundant, $m < n$, the Jacobian $\mathbf{J}$ is not a square matrix. In order to obtain a square matrix, we will define in Section 4 an extended Jacobian matrix, $\mathbf{J}^{+} \in \mathbb{R}^{n \times n}$.

In the closed-loop pseudo-inverse (CLP) method the joint positions can be computed through the time integration of the expression:

$$\Delta\mathbf{q} = \mathbf{J}^{\#}(\mathbf{q})\,\Delta\mathbf{x} \tag{4}$$

where $\Delta\mathbf{x} = \mathbf{x}_r - \mathbf{x}$ and $\mathbf{x}_r$ is the vector of reference (desired) position in the operational space. Nevertheless, in a previous study, addressing the CLP method [28], it was concluded that this method leads to unpredictable, not repeatable, arm configurations and reveals properties resembling those that occur in chaotic systems. As a consequence, the motion in joint space becomes unpredictable for subsequent cycles.

3. Genetic algorithms

GAs constitute a popular heuristic approach to multi-objective design and optimization [29]. GAs are a method for solving both constrained and unconstrained optimization problems, based on the mechanics of natural genetics and selection that was first introduced by Holland [30]. A GA allows a population composed of many individuals to evolve under specified selection rules to a state that maximizes the fitness function. The GA modifies repeatedly the population of individuals (possible solutions). At each step, the GA selects individuals at random, from the current population, to be parents, and uses them to produce the offspring for the next generation. Over successive generations, the population evolves towards an optimal solution. The GAs can be applied to solve a variety of optimization problems that are not well suited for standard optimization algorithms, including problems in which the objective function is discontinuous, not differentiable, stochastic, or is highly nonlinear.

The GA operates with a population of chromosomes normally randomly initialized. The GA uses two operators to generate new solution from existing ones: crossover and mutation. In the crossover operator, generally two chromosomes (the parents) are combined together to form new chromosomes (the offspring). The mutation operator introduces random changes into characteristics

of chromosomes and reintroduces genetic diversity into the population.

3.1. Multi-objective optimization

In many applications the fitness function involves multiple, often conflicting, objectives. Unlike single objective optimization, where one attempts to find the best solution, in multi-objective optimization there is usually no single solution that is optimum with respect to all objectives. There are two general approaches to multi-objective optimization [29]. One is to combine the individual objectives into a single composite function, or to move all but one objective to the constraint set. In both cases, an optimization method would return a single solution rather than a set of solutions that can be examined for trade-offs. The second general approach is to determine an entire set of Pareto optimal solutions or a representative subset. Consequently, there is one set of optimal solutions known as the Pareto-optimal set, the set of non-dominated solutions that represents the trade-off between the objectives. Without additional information, all these solutions are equally satisfactory, in the sense that no improvement can be achieved in one objective without the degradation in, at least, one of the remaining objectives.

If all objective functions are for minimization, a feasible solution s_1 dominates another feasible solution s_2 ($s_1 \succ s_2$) in the Pareto-optimality sense if, and only if, s_1 performs better in at least one objective and, at least, as good as s_2 in the rest:

$$s_1 \succ s_2 \Leftrightarrow \forall \beta \in \{1, \ldots, n_{obj}\} : f_\beta(s_1) \leq f_\beta(s_2) \text{ and } \exists \lambda \in \{1, \ldots, n_{obj}\} : f_\lambda(s_1) < f_\lambda(s_2) \quad (5)$$

where n_{obj} is the number of objective functions f. If none of the two feasible solutions, s_1 and s_2, dominates the other, then they are said to be non-dominated. A solution is classified to be Pareto-optimal if it is non-dominated by any other solution in the solution space:

$$s_1 \text{ is Pareto-optiomal} \Leftrightarrow s_2 \in S : s_2 \succ s_1 \quad (6)$$

where S is the solution space.

Generally, multi-objective GAs differ based on their fitness assignment procedure, elitism, or diversification approaches [29]. The first multi-objective GA was proposed by Schaffer [31] that considered the extension of the simple GA to accommodate vector-valued fitness measures, denoted as Vector Evaluated Genetic Algorithm (VEGA). Since then, other researchers developed new methods, such as Multi-objective Genetic Algorithm (MOGA) [32], Niched Pareto Genetic Algorithm (NPGA) [33], Strength Pareto Evolutionary Algorithm (SPEA) [34], Fast Non-dominated Sorting Genetic Algorithm (NSGA-II) [35], Dynamic Multi-objective Evolutionary Algorithm (DMOEA) [36], Multi-objective differential evolution (MODE) [37], and several others [38,39].

In this paper, it is adopted the NSGA-II where the crowded tournament selection and the crowding distance operators are replaced by the MaxiMin sorting scheme, which will be described in the sequel.

3.2. NSGA-II

The NSGA-II is a procedure for finding multiple Pareto optimal solutions that uses an elite-preserving mechanism. Moreover, NSGA-II uses a fast non-dominated sorting procedure and adopts an explicit diversity preserving mechanism.

Initially, at generation $T = 0$, a random parent population $P(T)$, of size $n_P = N$, is created. The population is sorted based on the non-domination. Each solution is assigned to a fitness equal to its non-domination level. Then, the operators of crossover and mutation are applied to create an offspring population $Q(T)$, of size $n_Q = N$. Thereafter, in generation T, the following procedure is applied. First, a combined population $R(T) = P(T) \cup Q(T)$, of size $n_R = 2N$, is formed. Then, non-dominated fronts $F_1, F_2, \ldots, F_k$, are identified in the population $R(T)$, where F_1 is the best non-dominated set, F_2 is the second best non-dominated set, and so on. The new parent population $P(T+1)$, is filled starting from solutions in F_1, then F_2, and so on, until the size exceeds $n_P = N$. To choose exactly $n_P = N$ elements in $P(T+1)$, the solutions of the last accepted front are sorted in descending order according to a crowded comparison criterion (a measure of density of solutions in the neighborhood) and the first best individuals needed to complete the parent population are selected. After, the new population $P(T+1)$ is used for selection, crossover and mutation to create a new offspring population $Q(T+1)$, of size $n_Q = N$. It is used a tournament selection based on the crowding distance to select parents from $P(T+1)$.

3.3. MaxiMin sorting scheme

The MaxiMin sorting scheme [40] is a modified version for the elitist selection operator used in NSGA-II. The crowding distance method is replaced by a MaxiMin technique leading to more uniformly distributed solutions than those achieved by the original crowding distance sorting scheme.

Similarly to the NSGA-II, the new parent population $P(T+1)$ is filled starting from solutions in F_1, then F_2, and so on, until the size exceeds $n_P = N$. To choose exactly $n_P = N$ solutions in $P(T+1)$, a maximum function is called to select the solutions of the last accepted front. To do this, the distance between each non-dominated solution and the set of solutions already selected in the new population $P(T+1)$ is evaluated. The solution, whose distance to the set is greater, is selected. Every time a solution enters to the population $P(T+1)$ the fitness of non-dominated solutions is reevaluated. This process continues until the population size is $n_P = N$.

3.4. Preferences in multi-objective evolutionary optimization

Many situations and problems in real world commonly involve the optimization of two or more conflicting objectives at once. A consequence of this state of affairs is that it is not possible to reach an optimal solution with respect to all the objectives evaluated individually. A solution that is better with respect to one objective requires normally a compromise with respect to the other objectives.

From a practical point of view the user needs only one solution; therefore, choosing a particular solution for implementation is the remaining decision-making task. Thus, the final solution of a multi-objective optimization problem results not only from the optimization process, but also from the decision process. Incorporating preference information in Evolutionary Multi-objective Optimization Algorithms (EMOA) can help with the problem of the exponential explosion of the number of solutions required for approximating the entire Pareto front as the number of objectives grows. Preference information can be used to concentrate on a small region of the Pareto front, while the EMO algorithms are used to find solutions in such a small region [41]. Usually, a decision is made based on the preferences of the decision-maker. Such preferences can be incorporated before, during, or after the optimization takes place. Rachmawati and Srinivasn [42] classified these algorithms as *a priori*, interactive and *a posteriori* methods, respectively.

A priori methods involve preference specification before the optimization process takes place. A popular approach to specify the decision-maker preferences in an *a priori* method is to form a weighted sum or some other scalarization of the multiple objectives, creating in this way a single-objective function to optimize, or by converting the objectives into restrictions imposed on the

optimization problem. However, this normally addresses another problem since the chosen alternative may be highly sensitive to the nature of scalarization used [43]. Moreover, for problems with more than three objectives, the process of specifying the right parameters becomes very difficult as the number of objectives increases.

Interactive methods allow preference specification during the optimization process, generating, in this way, better alternative accordingly with the information received from the decision maker. The interactive approaches allow decision makers to alter parameters during the search and effectively influencing the direction of the search.

A posteriori methods involve preference specification after the optimization process. Thus, the effort must be developed towards finding a set of well distributed Pareto-optimal solutions, by considering all objectives to be important. The decision maker must then use high-level information to select his preferred solution.

A considerable number of papers appeared in the literature defining methodologies that allow the incorporation of the decision maker's preferences into the search procedure. Fonseca and Fleming [32] used MOGA with interactive goal attainment method. There are two kinds of parameters in that method, namely: the goal vector and the weight coefficient for each objective, all of them specified by the designer before starting the optimization. Jin and Sendhoff [44] developed a method for converting fuzzy preferences into interval-based weights, which are incorporated in the EMOA using random weighted aggregation and dynamic weighted aggregation techniques [45]. The method is able to find a number of solutions instead of a single one, given a set of fuzzy preferences over different objectives. Cvetkovic and Parmee [46] used a method where the decision maker adopts linguistic variables and modifiers to define the relative importance between two objectives that are transformed into a fuzzy preference matrix to establish an order between the objectives. The method is integrated into multi-objective optimization methods that use weights: weighted sum-based method, weighted Pareto optimization, weighted coevolutionary optimization, and weighted scenario and constraint handling. Ishibuchi et al. [47] implemented a hybrid algorithm of NSGA-II to incorporate *a priori* information about the decision's maker preference into EMOAs, modifying only the parent selection phase of NSGA-II. It is adopted a scalarizing function defined by the given preference information. Ferreira et al. [48] proposed a methodology, based on a weight stress function approach, to select a single solution (or a set of solutions) from the set of non-dominated solutions, taking into account the preferences of the decision-maker. Kao and Jacobson [49] studied the problem of post-optimality selection by providing a framework to obtain a preferred subset of solutions from a very large set of solutions. A value function represents the preferences of a decision-maker across the objective functions. The subset of preferred Pareto optimal solutions can be calculated by solving a discrete optimization problem, PPOSP, which allows the decision-maker to obtain a desirable subset of size N, based on threshold values for each objective function. Deb and Gupta [50] developed an EMOA to find the robust optimal frontier instead of the global Pareto-optimal front. Robustness is a measure of the sensitivity of a solution performance in objective functions against perturbations in the decision variables. Thus, robustness of a solution means the insensitivity of that solution when disturbance or noise existed in decision variables. Two measures of robustness are presented: in the first approach, the original objective functions are replaced by the mean effective objective values computed at a point; in the second approach, a constraint limiting the change in function values due to local perturbations, is added, to the original objectives. Other research approaches can also be found in [51–55].

In this paper, the decision maker preferences are incorporated *a posteriori*.

4. Proposed method for robot trajectory control

In this section we formulate a new method for the robot trajectory control of a redundant planar manipulator in periodic trajectories. The proposed method combines the CLP method with an EMOA, namely the closed-loop multi-objective genetic algorithm (CLMOGA).

With the CLMOGA the joint positions can be calculated using the closed-loop structure. With this algorithm the configurations for a given instant t are calculated using the configurations obtained at the instant $t-\Delta t$, where Δt is the sampling time. The inverse kinematics is solved without the use of the pseudo-inverse matrix. For this purpose, the Jacobian matrix is extended in order to form a square matrix.

The algorithm aims finding a set of non-dominated solutions to the inverse kinematics, when considering two objectives simultaneously. From this set is then selected the solution that satisfies certain criteria predefined by the user. This is an *a posteriori* method.

In order to find an initial robot joint configuration it is used a simple GA to be defined in Section 4.2. This method denoted as OLGA-Open Loop Genetic Algorithm adopts the direct kinematics.

4.1. The CLMOGA formulation

The CLMOGA takes advantage of the closed-loop structure without requiring the calculation of the pseudo-inverse. The CLMOGA uses an extended Jacobian matrix, $\bar{J} \in \mathbb{R}^{n\times n}$, and an extended vector, $\overline{\Delta x} \in \mathbb{R}^n$, as a strategy to obtain the joint configurations for a given end-effector position.

The matrices $\bar{J}$ and $\overline{\Delta x}$ take the form:

$$\bar{J} = \begin{bmatrix} \mathbf{J} \\ \mathbf{J}^* \end{bmatrix} \qquad \overline{\Delta x} = \begin{bmatrix} \Delta \mathbf{x} \\ \Delta \mathbf{x}^* \end{bmatrix} \tag{7}$$

where $\mathbf{J} \in \mathbb{R}^{m\times n}$ is the Jacobian of a n-link planar manipulator (i.e., $m=2$). The Jacobian $\mathbf{J}$ has a recursive nature according with the expression:

$$\mathbf{J} = \begin{bmatrix} -\sum_{k=1}^{n} l_k S_{1\ldots k} & \cdots & -l_n S_{1\ldots n} \\ \sum_{k=1}^{n} l_k C_{1\ldots k} & \cdots & l_n C_{1\ldots n} \end{bmatrix} \tag{8}$$

where l_i is the length of link i, $q_{i\ldots k} = q_i + \ldots + q_k$, $S_{i\ldots k} = \sin(q_{i\ldots k})$ and $C_{i\ldots k} = \cos(q_{i\ldots k})$, $i, k \in \mathbb{N}$, and

$$\Delta \mathbf{x} = \begin{bmatrix} \Delta x_1 \\ \Delta x_2 \end{bmatrix} \tag{9}$$

The matrices $\boldsymbol{J}^* \in \mathbb{R}^{(n-m)\times n}$ and $\Delta \boldsymbol{x}^* \in \mathbb{R}^{n-m}$ can be calculated using expressions:

$$\mathbf{J}^* = \begin{bmatrix} j_{(m+1)1} & \cdots & j_{(m+1)n} \\ \cdots & j_{ik} & \cdots \\ j_{n1} & \cdots & j_{nn} \end{bmatrix} \qquad \Delta \mathbf{x}^* = \begin{bmatrix} \Delta x_{m+1} \\ \vdots \\ \Delta x_i \\ \vdots \\ \Delta x_n \end{bmatrix} \tag{10}$$

where the matrix elements j_{ik} and Δx_i, $i = m+1, \ldots, n$ and $k = 1, \ldots, n$, are values generated by the GA in the intervals $[j_{\min}, j_{\max}]$ and $[\Delta x_{\min}, \Delta x_{\max}]$.

The joint positions, using the square matrix $\bar{\mathbf{J}}$, are computed through the time integration of the joint velocities given by the expressions:

$$\Delta \mathbf{q} = (\bar{\mathbf{J}})^{-1} \overline{\Delta \mathbf{x}} \tag{11}$$

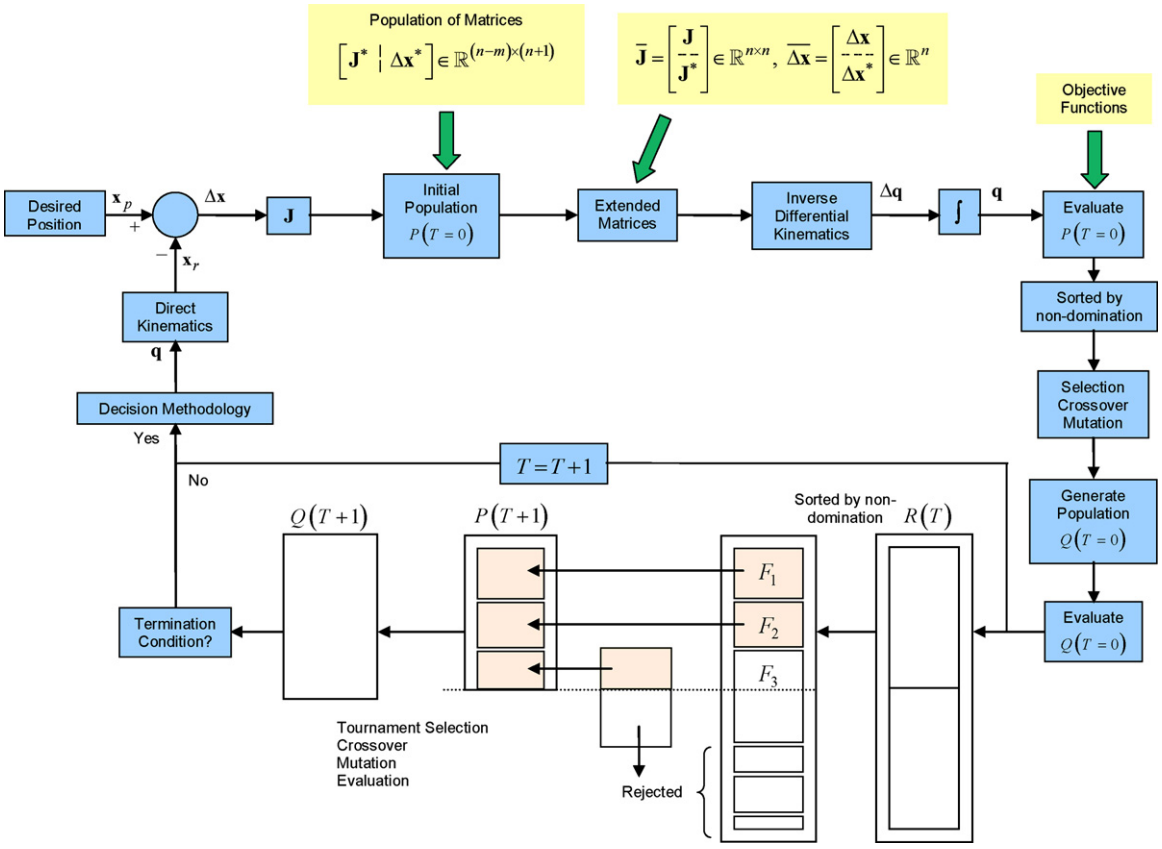

Fig. 1. Diagram of the CLMOGA method.

The CLMOGA procedure is shown in Fig. 1, where $\mathbf{x}_p$ is the vector of reference (desired) position in the operational space and $\mathbf{x}_r$ is a vector representing the current position of the end-effecter in the operational space.

The algorithm begins by calculating the values of $\mathbf{J}$ and $\Delta\mathbf{x}$ using expressions (8) and (9). The generation counter is initialized with $T=0$ and an initial population of chromosomes, $P(T)=[(\mathbf{J}^*, \Delta\mathbf{x}^*)^{(T,1)}, \ldots, (\mathbf{J}^*, \Delta\mathbf{x}^*)^{(T,N)}]$, with dimension $n_P=N$, is constructed randomly. For each element i of the population in generation T, the corresponding extended matrices, $(\bar{\mathbf{J}})^{(T,i)}$ and $(\overline{\Delta\mathbf{x}})^{(T,i)}$, $i=1,\ldots,N$, using expressions (7) are calculated. Then the inverse matrix, $((\bar{\mathbf{J}}))^{(T,i)}$, is calculated and the corresponding values $(\Delta\mathbf{q})^{(T,i)}$ and $\mathbf{q}^{(T,i)}$ obtained using expressions:

$$(\Delta\mathbf{q})^{(T,i)} = ((\mathbf{J}^+)^{-1})^{(T,i)}(\overline{\Delta\mathbf{x}})^{(T,i)}, \quad i=1,\ldots,N \tag{12}$$

and

$$\mathbf{q}^{(T,i)} = \Delta\mathbf{q}^{(T,i)} + \mathbf{q}_0, \quad i=1,\ldots,N \tag{13}$$

The population is sorted based on the non-dominance of solutions. After that it is adopted a binary tournament selection. The selected strings are randomly grouped together into pairs and a crossover point is randomly selected for each one of the $n-m$ lines of the parent. Then crossover is performed among pairs, generating a new offspring population $Q(T=0)$ of size $n_Q=N$. Finally, the mutation operator is used so that one variable value is replaced with a new random one. The population $Q(T=0)$ is then evaluated. Thereafter, we use the following algorithm in every generation until the maximum number of generations defined by the user is reached. First, a combined population $R(T)$, of size $n_R=2N$, is formed using populations $P(T)$ and $Q(T)$ and then the population $R(T)$ is classified using the MaxiMin sorting scheme. The new parent population $P(T+1)$ is filled by choosing N solutions from $R(T)$. The new population $P(T+1)$ is used for selection, crossover and mutation to create a new offspring population $Q(T+1)$ of size $n_Q=N$.

When the stopping criteria is satisfied, the new configuration for the joint positions is selected. At this stage it is necessary to use *a priori* information, provided by the user, to select a solution among the set of non-dominated solutions obtained by the algorithm. This method will be described in Section 4.1.3.

4.1.1. Representation in the CLMOGA

Each chromosome (string) is implemented by a matrix of $n_V(n-m)\times(n+1)$ values (genes), accordingly to expression (10). For the generation T, the pth chromosome of the population is represented as:

$$\left[(\mathbf{J}^*)^{(T,p)} \vdots (\Delta\mathbf{x}^*)^{(T,p)}\right] \tag{14}$$

where

$$(\mathbf{J}^*)^{(T,p)} = \left[j_{ik}^{(T,p)}\right] \in \mathbb{R}^{(n-m)\times n} \tag{15}$$

and

$$(\Delta\mathbf{x}^*)^{(T,p)} = \left[\Delta x_i^{(T,p)}\right] \in \mathbb{R}^{n-m} \tag{16}$$

4.1.2. Objective functions

In order to make the inverse kinematics repetitive, the following objective function minimizes the joint displacement between the current joint position and the initial joint position:

$$f_1 = \dot{\mathbf{q}}^T\dot{\mathbf{q}} + \left(\frac{\mathbf{q}-\mathbf{q}_0}{\Delta t}\right)^T\left(\frac{\mathbf{q}-\mathbf{q}_0}{\Delta t}\right) \tag{17}$$

where $\mathbf{q}$ and $\mathbf{q}_0$ represents the current and the initial joint configurations, respectively, and Δt is the step time increment.

In order to minimize the positional error, X_e, between the end-effector desired position and the obtained final position, it is considered the following objective function:

$$f_2 = X_e = \sqrt{(x_{r1} - x_{f1})^2 + (x_{r2} - x_{f2})^2} \tag{18}$$

where $\mathbf{x}_r = (x_{r1}, x_{r2})$ and $\mathbf{x}_f = (x_{f1}, x_{f2})$ are vectors representing the end-effector reference (desired) position and the obtained position, respectively.

4.1.3. Decision making methodology

The adopted methodology selects a solution from the Pareto set after the search process has been performed. Therefore, this constitutes an *a posteriori* method.

Suppose that the Pareto front obtained has M elements. First, each one of the objective functions f_β is normalized to have the same range, leading to function z_β, $\beta = 1, \ldots, n_{obj}$, i.e., $z_\beta \in [0, 1]$, through the expression:

$$z_{\beta,\gamma} = \frac{f_{\beta,\gamma} - f_{\beta,\min}}{f_{\beta,\max} - f_{\beta,\min}}, \quad \beta = 1, \ldots, n_{obj};\ \gamma = 1, \ldots, M \tag{19}$$

where $f_{\beta,\gamma}$, $f_{\beta,\min}$ and $f_{\beta,\max}$ are the actual, the minimum and the maximum values for the objective function f_β. Then, the multiple objectives are aggregated into a scalar objective through the expression:

$$g_\gamma = w_1 z_{1,\gamma} + \ldots + w_{n_{obj}} z_{n_{obj},\gamma}, \quad \gamma = 1, \ldots, M \tag{20}$$

where $w_1, \ldots, w_{n_{obj}}$ denotes a weighting factors and $\sum_{\beta=1}^{n_{obj}} w_\beta = 1$. These weights reflect the importance of each objective upon the final result and must be provided by the user in advance.

The element that has the minimum value g_γ is the selected solution.

4.2. The OLGA formulation

The OLGA trajectory planning adopts a simple open-loop structure. An initial population of strings $P(T) = [\mathbf{q}^{(T,1)}, \mathbf{q}^{(T,2)}, \ldots, \mathbf{q}^{(T,N)}]$, with dimension $n_P = N$, is constructed at random and the search is carried out among this population. Each chromosome is defined through an array of $n_V = n$ values, q_i, $i = 1, \ldots, n$, represented as floating-point numbers initialized in the range $[q_{\min}, q_{\max}]$. For the generation T, the pth chromosome of the population is represented as:

$$\mathbf{q}^{(T,p)} = \left(q_1^{(T,p)}, \ldots, q_n^{(T,p)}\right) \tag{21}$$

The end-effector position, $\mathbf{x}^{(T,p)}$, for each configuration, $\mathbf{q}^{(T,p)}$, is easily calculated using the direct kinematics:

$$\mathbf{x}^{(T,p)} = f\left(\mathbf{q}^{(T,p)}\right) \tag{22}$$

The three different operators used in the GA are reproduction, crossover and mutation. In what respecting the reproduction operator, the successive generations of new strings are reproduced on the basis of their fitness function. In this case, it is used a rank weighting to select the strings from the old to the new population. For the crossover operator, the strings are randomly grouped into pairs. Single crossover is then performed among pairs. Finally, for the mutation operator, one variable value is replaced with a new random one. The first chromosome is not mutated due to the adoption of elitism.

The fitness function is defined based on the positional error of the end-effector; therefore, the GA minimizes the function f_2, given by (18).

5. Simulation results

In this section we start by analyzing the performance of the CLMOGA for a free workspace and, in a second phase, we study the effect of including several types of obstacles in the working environment.

Without lacking of generality, in the following experiments are adopted arms having identical link lengths, $l_1 = l_2 = \ldots = l_n$.

The experiments consist in the analysis of the kinematic performance of a planar manipulator with $n = \{3, 4\}$ rotational joints, denoted as nR-robot, that is required to repeat a circular motion in the operational space with frequency $\omega_0 = 7.0$ rad s^{-1}, centre at $r = (x_1{}^2 + x_2{}^2)^{1/2}$, radius $\rho = 0.5$ and a step time increment of $\Delta t = 10^{-3}$ s. The goal here is to position the end-effector of the nR-robot at a target location while minimizing the joint angle drift using the fitness function f_1 in Eq. (12), the positional error of the end-effector using the fitness function f_2 represented in (13), while avoiding the obstacles, if these exist in the workspace. The initial joint configuration is obtained using the OLGA with the fitness function f presented in (18).

The average of the positional error for n_C cycles is given by the expression:

$$\overline{X}_e = \frac{1}{k}\sum_{i=1}^{k} X_e \tag{23}$$

where k is the number of sampling points and is defined as:

$$k = \frac{T'}{\Delta t} n_C \tag{24}$$

where $T' = 2\pi/\omega_0$ is the period (in seconds) of time to complete a cycle, Δt is the sampling time (in seconds) and n_C is the number of cycles to be executed.

The average of the total joint displacement between the initial joint configuration and the final joint configuration, $\bar{Q}_e$, for the nR-robot is given by the expression:

$$\overline{Q}_e = \frac{1}{n}\sqrt{\sum_{i=1}^{n} (q_{0i} - q_{fi})^2} \tag{25}$$

where q_{0i} denotes the initial joint configuration ($t = 0.0$), q_{fi} represents the final configuration after n_C cycles ($t = k\Delta t$), for joint i, and n is the total number of joints.

The CLMOGA adopts crossover and mutation probabilities of $p_c = 0.5$ and $p_m = 0.5$, respectively, the string population is $n_P = \{200, 400\}$ for $n = \{3, 4\}$ rotational joints, respectively, and the results are obtained for $n_G = 200$ consecutive generations. Each variable value, j_{ik} and Δx_i, $i = 1, \ldots, n-m$, $k = 1, \ldots, n$, is initialized in the range $[-1.0, 1.0]$.

The OLGA adopts crossover and mutation probabilities of $p_c = 0.5$ and $p_m = 0.5$, respectively, a string population of $np = 1600$ and the results are obtained for $n_G = 200$ consecutive generations. Each variable value, q_i, $i = 1, \ldots, n$, is initialized in the range $[-2\pi, 2\pi]$.

5.1. The CLMOGA performance in a workspace without obstacles

The average of the positional error, $\bar{X}_e$, and the average of the total joint displacement, $\bar{Q}_e$, for $n = \{3, 4\}$ rotational joints, $n_C = 2$ cycles, radial distance $r = \{0.7, 2.0\}$ and weight $w \in [0.0, 1.0]$, with a step increment of $\Delta w = 0.1$, are depicted in Tables 1 and 2, respectively.

We observe that, in general, the positional error is worst and the drift in the joint positions is better the higher the value of w.

The positional error is, in general, good, revealing that the CLMOGA leads to good precision in the task of positioning the

Table 1
Average of the positional error, $\bar{X}_e$, after $n_C = 2$ cycles for $n = \{3, 4\}$, $w \in [0.0, 1.0]$ and (a) $r = 0.7$; (b) $r = 2.0$.

$\bar{X}_e$		
w	3R	4R
(a)		
0.0	3.36E−06	3.44E−07
0.1	4.06E−06	2.22E−06
0.2	5.56E−06	3.47E−06
0.3	6.55E−06	6.15E−06
0.4	7.00E−06	7.62E−06
0.5	7.35E−06	9.27E−06
0.6	7.35E−06	9.73E−06
0.7	7.36E−06	1.00E−05
0.8	7.36E−06	1.02E−05
0.9	7.36E−06	1.03E−05
1.0	7.36E−06	1.03E−05
(b)		
0.0	3.53E−06	3.31E−06
0.1	4.00E−06	5.22E−06
0.2	4.01E−06	5.44E−06
0.3	4.02E−06	5.61E−06
0.4	4.02E−06	5.74E−06
0.5	4.03E−06	5.76E−06
0.6	4.06E−06	5.75E−06
0.7	4.06E−06	5.74E−06
0.8	4.06E−06	5.73E−06
0.9	4.06E−06	5.72E−06
1.0	4.06E−06	5.71E−06

Table 3
Average of the positional error, $\bar{X}_e$, in a workspace of two obstacles, after $n_C = 2$ cycles for $n = \{3, 4\}$, $w \in [0.0, 1.0]$ and (a) $r = 0.7$; (b) $r = 2.0$.

$\bar{X}_e$		
w	3R	4R
(a)		
0.0	2.59E−06	7.73E−07
0.1	4.65E−06	2.00E−06
0.2	5.73E−06	2.71E−06
0.3	6.57E−06	6.05E−06
0.4	7.01E−06	7.41E−06
0.5	7.34E−06	8.80E−06
0.6	7.34E−06	9.20E−06
0.7	7.34E−06	9.66E−06
0.8	7.34E−06	9.86E−06
0.9	7.34E−06	9.99E−06
1.0	7.34E−06	1.00E−05
(b)		
0.0	3.05E−06	3.29E−06
0.1	3.29E−06	3.53E−06
0.2	3.37E−06	3.56E−06
0.3	3.39E−06	3.58E−06
0.4	3.41E−06	3.59E−06
0.5	3.41E−06	3.59E−06
0.6	3.42E−06	3.58E−06
0.7	3.42E−06	3.58E−06
0.8	3.42E−06	3.57E−06
0.9	3.42E−06	3.58E−06
1.0	3.42E−06	3.57E−06

end-effecter at the target position. The largest variation between the minimum, $\bar{X}_{e_{\min}}$, and the maximum, $\bar{X}_{e_{\max}}$, for the average of the positional error occurs for $n = 4$ and $r = 0.7$ when $\bar{X}_{e_{\min}} = 3\mathrm{E}-07$ and $\bar{X}_{e_{\max}} = 1\mathrm{E}-05$.

On the other side, the drift in the joint positions has a significant decrease as the value of w becomes higher. For small values of w we get a large drift in the joint positions, but for high values of w the drift is relatively small suggesting that the joint configurations are repetitive, which was confirmed by analyzing the joint waveforms.

So, different strategies in the selection of the solution from the Pareto front lead to different types of solutions. For example, if we select either $w = 0.0$ or $w = 1.0$ for $n = 4$ and $r = 0.7$, we get clearly different results for the repeatability as we can see in Fig. 2, where successive robot configurations during $n_C = 2$ cycles are depicted. Fig. 3 shows the time evolution of the corresponding joint positions. Moreover, the average of the positional error is $\bar{X}_e = 3\mathrm{E}-07$ for $w = 0.0$ and $\bar{X}_e = 1\mathrm{E}-05$ for $w = 1.0$. Between these extreme optimal solutions several others were found, for different values of w, that have an intermediate behaviour and which can be selected according with the importance of each objective given by function g_γ.

Table 2
Average of the joint displacement, $\bar{Q}_e$, after $n_C = 2$ cycles for $n = \{3, 4\}$, $w \in [0.0, 1.0]$ and (a) $r = 0.7$; (b) $r = 2.0$.

$\bar{Q}_e$		
w	3R	4R
(a)		
0.0	5.18E−01	2.16E−01
0.1	3.14E−02	1.69E−02
0.2	8.62E−03	5.86E−03
0.3	4.99E−03	3.80E−03
0.4	2.29E−03	4.33E−03
0.5	1.32E−03	5.02E−03
0.6	8.40E−04	2.74E−03
0.7	5.39E−04	2.51E−03
0.8	3.60E−04	2.85E−03
0.9	2.56E−04	6.94E−04
1.0	2.27E−04	1.43E−04
(b)		
0.0	8.97E−02	4.13E−02
0.1	3.33E−04	4.42E−03
0.2	6.43E−04	1.88E−03
0.3	5.40E−04	1.25E−03
0.4	4.16E−04	8.24E−04
0.5	3.12E−04	5.61E−04
0.6	2.48E−04	3.95E−04
0.7	1.99E−04	2.76E−04
0.8	1.68E−04	2.08E−04
0.9	1.51E−04	1.69E−04
1.0	1.46E−04	1.57E−04

Because for each simulation we have k Pareto fronts, it is impossible to represent all the fronts. However, in Figs. 4 and 5 are represented the Pareto fronts obtained for $t = \{0.001, 0.002\}$, $w = 0.0$ and $r = \{0.7, 2.0\}$, respectively, where the points a and b represent the best solutions found for the f_1 and f_2 objectives, respectively. In general, all the Pareto fronts have n_p points but it is verified that, when the manipulator has $n = 4$ rotational joints, the diversity of the solution front for $r = 0.7$ is not as good as it occurs for $r = 2.0$. This result reveals that, as expected, it is more difficult to find good solutions that satisfy the criterion of repeatability and precision simultaneously when the radial distance is $r = 0.7$.

5.2. The CLMOGA performance in a workspace with obstacles

This section presents the results of the CLMOGA when considering two obstacles in the workspace. For a given joint configuration, when some part of the manipulator is inside an obstacle, the CLMOGA rejects the configuration and generates a new chromosome.

For the case of $r = 0.7$ are adopted obstacles consisting of a circle with centre at (1.4, 0.5) and radius 0.2, and one rectangle, with upper left and lower right corners with coordinates (0.3, 1.7) and (0.8, 1.3), respectively. For $r = 2.0$ are considered the obstacles represented by one circle with centre at (1.6, 0.6) and radius 0.2, and one rectangle, with upper left and lower right corners with coordinates (0.4, 1.0) and (0.9, 0.6), respectively.

Tables 3 and 4 depict the average of the positional error, $\bar{X}_e$, and the average of the total joint displacement, $\bar{Q}_e$, for $n = \{3, 4\}$

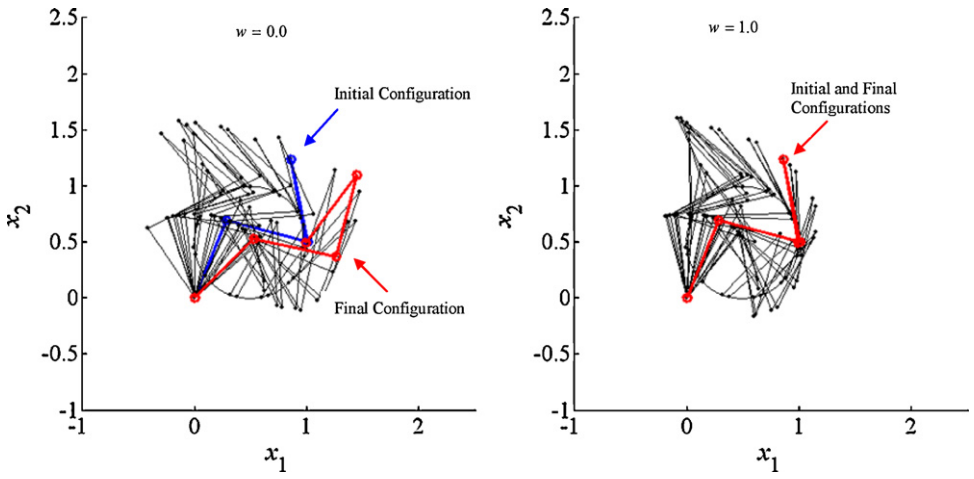

Fig. 2. Successive robot configurations during $n_C = 2$ cycles for $n = 4$, $r = 0.7$ and $w = \{0.0, 1.0\}$.

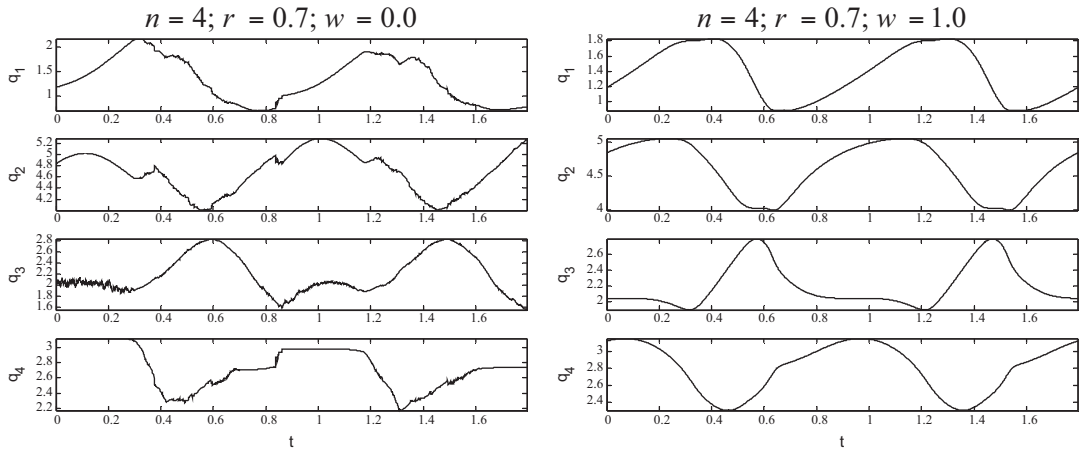

Fig. 3. The 4*R*-robot joint positions *versus* time during $n_C = 2$ cycles for $r = 0.7$ and $w = \{0.0, 1.0\}$.

rotational joints, $n_C = 2$ cycles, radial distance $r = \{0.7, 2.0\}$, weight $w \in [0.0, 1.0]$ and an increment of $\Delta w = 0.1$.

We observe that, in general, the positional error becomes higher and the drift in the joint positions results smaller the higher the value of *w*.

The positional error is good, revealing that the CLMOGA leads to good precision in the task of positioning the end-effecter at the target position while avoiding the obstacles in the workspace, whichever the value of *w*.

On the other hand, the drift in the joint positions has, in general, a significant diminishing when the value of *w* increases.

Fig. 6 depicts successive robot configurations for $n = 4$, $r = 0.7$ and $w = \{0.0, 1.0\}$, during $n_C = 2$ cycles. As we can see, the results are clearly different for the repeatability in the joint position. Fig. 7 shows the time evolution of the corresponding joint positions. Moreover, the average of the positional error is $\bar{X}_e = 8\mathrm{E}-07$ for $w = 0.0$ and $\bar{X}_e = \mathrm{E}-05$ for $w = 1.0$.

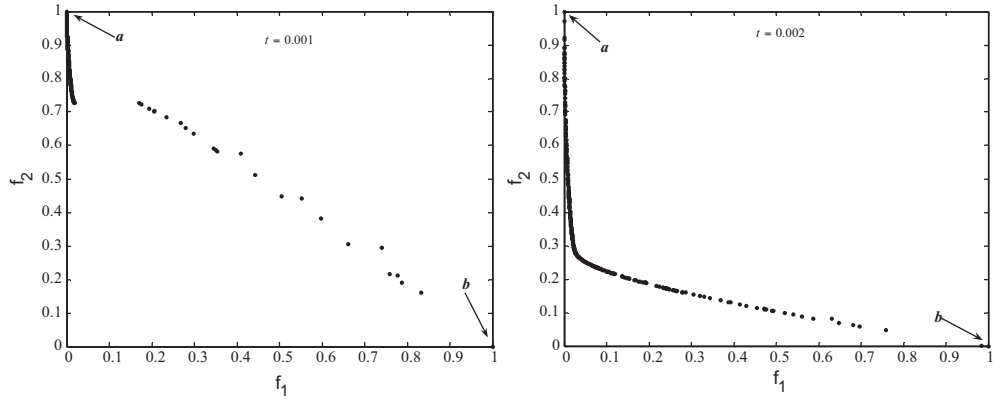

Fig. 4. Best Pareto fronts obtained for the 4*R*-robot for $t = \{0.001, 0.002\}$, $w = 0.0$ and $r = 0.7$.

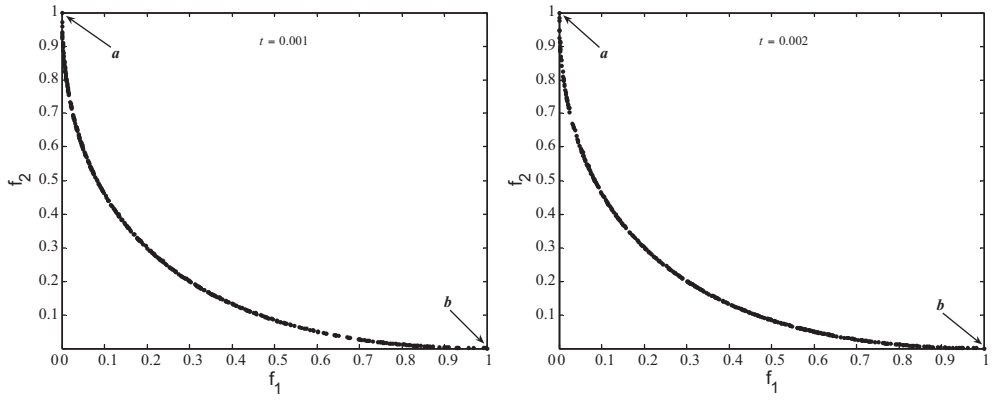

Fig. 5. Best Pareto fronts obtained for the 4*R*-robot for $t = \{0.001, 0.002\}$, $w = 0.0$ and $r = 2.0$.

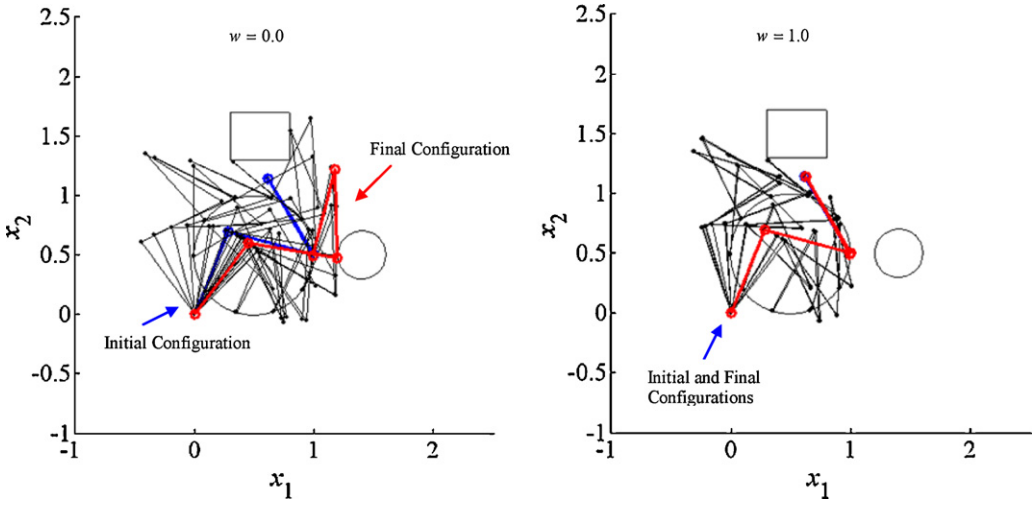

Fig. 6. Successive robot configurations in a workspace with two obstacles during $n_C = 2$ cycles for $n = 4$, $r = 0.7$ and $w = \{0.0, 1.0\}$.

In Figs. 8 and 9 are represented the Pareto fronts obtained for $t = \{0.001, 0.002\}$, $w = 0.0$ and $r = \{0.7, 2.0\}$, respectively. The points a and b represent the best solutions found for the f_1 and f_2 objectives, respectively. Similarly to what was verified for a workspace without obstacles, in general, for $n = 4$, all the Pareto fronts have n_p solutions. Moreover, the diversity of the solution front for $r = 0.7$ is not as good as it is for $r = 2.0$. This result reveals that, it is more difficult to find good solutions that satisfy the criterion of repeatability and precision simultaneously when the radial distance is $r = 0.7$, with or without obstacles in the workspace.

Therefore, the results are consistent with those of the previous section and that the presence of obstacles does not present an additional complexity for the CLMOGA to reach a good solution in accordance to the user preferences.

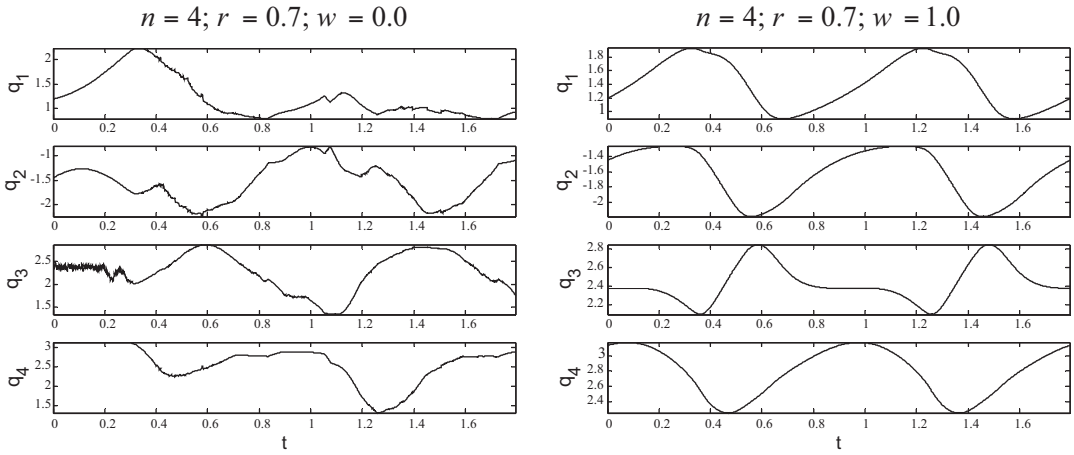

Fig. 7. The 4*R*-robot joint positions *versus* time in a workspace of two obstacles during $n_C = 2$ cycles for $r = 0.7$ and $w = \{0.0, 1.0\}$.

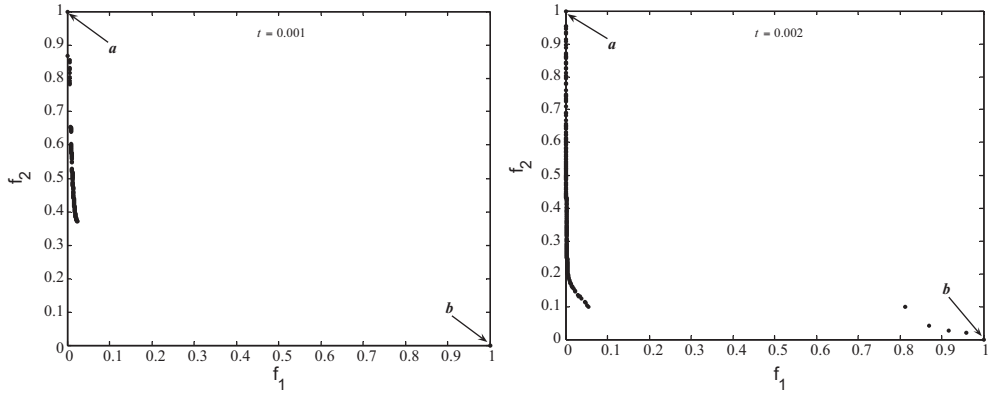

Fig. 8. The Pareto fronts obtained for the 4*R*-robot for $t = \{0.001, 0.002\}$, $w = 0.0$ and $r = 0.7$.

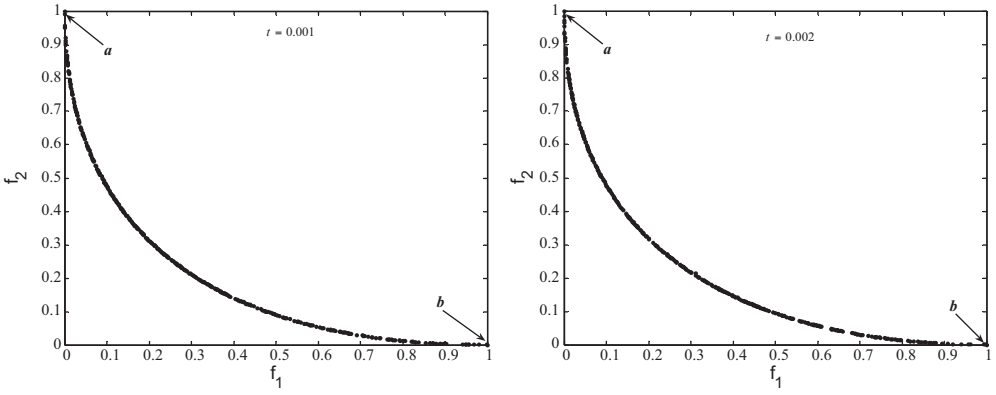

Fig. 9. The Pareto fronts obtained for the 4*R*-robot for $t = \{0.001, 0.002\}$, $w = 0.0$ and $r = 2.0$.

Table 4
Average of the joint displacement, $\bar{Q}_e$, in a workspace of two obstacles, after $n_C = 2$ cycles for $n = \{3, 4\}$, $w \in [0.0, 1.0]$ and (a) $r = 0.7$; (b) $r = 2.0$.

$\bar{Q}_e$		
w	3*R*	4*R*
(a)		
0.0	3.82E−01	2.00E−01
0.1	2.85E−02	1.26E−02
0.2	5.90E−03	4.23E−03
0.3	3.18E−03	5.55E−03
0.4	1.60E−03	3.03E−03
0.5	9.83E−04	5.95E−03
0.6	6.47E−04	5.21E−03
0.7	4.48E−04	2.49E−03
0.8	3.23E−04	3.99E−03
0.9	2.66E−04	5.83E−04
1.0	2.51E−04	1.34E−04
(b)		
0.0	1.05E−01	4.43E−02
0.1	1.20E−02	9.62E−03
0.2	4.99E−03	4.06E−03
0.3	2.60E−03	2.91E−03
0.4	1.49E−03	1.24E−03
0.5	9.26E−04	5.63E−04
0.6	6.00E−04	5.54E−04
0.7	3.86E−04	3.70E−04
0.8	2.48E−04	2.27E−04
0.9	1.63E−04	1.33E−04
1.0	1.35E−04	9.28E−05

6. Conclusions

A CLMOGA that combines the CLP with a multi-objective GA was presented. It was used the NSGA-II, where the crowding distance method adopted in the NSGA-II, is replaced by the MaxiMin sorting scheme. Two criteria were selected: the joint displacement, between the current joint position and the initial joint position, and the positional error between the desired final position and the obtained final position. The proposed methodology selects a solution from the Pareto set after the search process has been performed. First, each one of the objective functions is normalized to have the same range and then the multiple objectives are aggregated into a scalar objective using a weighting factor. The element that has the minimum value is the selected solution.

Several experiments were developed to study the performance of the CLMOGA when the manipulator is required to repeat a circular motion in the operational space, while satisfying the optimization criteria, in a workspace without and with obstacles.

The results show that the CLMOGA gives good results in the perspective of the positional error. For small values of the weighting factor *w* we get a large drift in the joint positions; however, for high values of *w* the drift is considerable inferior making the joint configurations repetitive. Between the extreme optimal solutions several others were found, for different values of *w*, that have an intermediate behaviour and which can be selected according with the importance of each objective.

The Pareto optimal fronts have a large number of solutions but, the diversity of the fronts, is in general better for the radial distance $r = 2.0$. For $r = 0.7$ and $n = 4$ rotational joints the results reveal that it is more difficult to find good solutions that satisfy the criterion of repeatability and precision simultaneously. Finally, it is also shown that the presence of obstacles does not present an additional complexity for the CLMOGA.

Some final words about the pros and cons of the proposed methodology. The first observation is that the CLMOGA requires a

considerable computational effort and, therefore, further research is necessary for implementing real-time algorithms. The second aspect is the number and type of objective functions, since expressions considering power, torque and energy may also be useful in defining new optimization criteria.

References

[1] D.N. Nenchev, Y. Tsumaki, Motion analysis of a kinematically redundant seven-DOF manipulator under the singularity-consistent method, in: Proc. of the 2003 IEEE Int. Conf. on Robotics and Automation, 2003, pp. 2760–2765.
[2] L. Keith, C. Doty, C. Melchiorri, Bonivento, A theory of generalized inverses applied to robotics, International Journal of Robotics Research 12 (1993) 1–19.
[3] J. Baillieul, J. Hollerbach, R. Brockett, Programming and control of kinematically redundant manipulators, in: Proc. of the 23rd IEEE Conf. on Decision and Control, 1984, pp. 768–774.
[4] C.A.C. Klein, C. Huang, Review of pseudoinverse control for use with kinematically redundant manipulators, IEEE Transactions on Systems, Man, and Cybernetics 13 (1983) 245–250.
[5] J. Baillieul, Kinematic programming alternatives for redundant manipulators, in: Proc. of the IEEE Int. Conf. on Robotics and Automation, 1985, pp. 722–728.
[6] D.R. Baker, C.W. Wampler I.I., On the inverse kinematics of redundant manipulators, The International Journal of Robotics Research 7 (2) (1988) 3–21, 11.
[7] R.G. Roberts, A. Maciejewski, Repeatable generalized inverse control strategies for kinematically redundant manipulators, IEEE Transactions on Automatic Control 38 (5) (1993) 689–699.
[8] C.A. Klein, S. Ahmed, Repeatable pseudoinverse control for planar kinematically redundant manipulators, IEEE Transactions on Systems, Man, and Cybernetics 25 (12) (1995) 1657–1662.
[9] Y. Zhang, H. Zhu, X. Lv, K. Li, Joint, angle drift problem of PUMA560 robot arm solved by a simplified LVI-based primal-dual neural network, in: Proc. of the IEEE Int. Conf. on Industrial Technology, 2008, pp. 1–26.
[10] Y. Zhang, X. Lv, Z. Li, Z. Yang, H. Zhu, Effective neural remedy for drift phenomenon of planar three-link robot arm using quadratic performance index, Electronics Letters 44 (6) (2008) 436–437.
[11] Y. Zhang, Z. Tan, Z. Yang, X. Lv, A dual neural network applied to drift-free resolution of five-link planar robot arm, in: Proceedings of the 2008 IEEE International Conference on Information and Automation, 2008, pp. 1274–1279.
[12] D.E. Goldenberg, Genetic Algorithms in Search Optimization, and Machine Learning, Addison-Wesley, Reading, MA, 1989.
[13] J.K. Parker, A.R. Khoogar, D.E. Goldberg, Inverse kinematics of redundant robots using genetic algorithms, in: Proc. of the 1989 IEEE Int. Conf. on Robotics and Automation, 1989, pp. 271–276.
[14] T. Arakawa, N. Kubota, T. Fukuda, Virus-evolutionary genetic algorithm with subpopulations: application to trajectory generation of redundant manipulator through energy optimization, in: Proc. of the 1996 IEEE Int. Conf. on Systems, Man, and Cybernetics, 1996, pp. 14–17.
[15] N. Kubota, T. Arakawa, T. Fukuda, K. Shimojima, Trajectory generation for redundant manipulator using virus evolutionary genetic algorithm, in: Proc. of the IEEE Int. Conf. on Robotics and Automation, 1997, pp. 205–210.
[16] V. de la Cueva, F. Ramos, Cooperative genetic algorithms: a new approach to solve the path planning problem for cooperative robotic manipulators sharing the same work space, in: Proc. of the IEEE/RSJ Int. Conference on Intelligent Robots and Systems, 1998, pp. 267–272.
[17] T. Nishimura, K. Sugawara, I. Yoshihara, K. Abe, A motion planning method for a hyper multi-joint manipulator using genetic algorithm, in: Proc. of the IEEE Int. Conf. on Systems, Man, and Cybernetics, 1999, pp. 645–650.
[18] B. McAvoy, B. Sangolola, Optimal trajectory generation for redundant planar manipulators, in: Proc. of the IEEE Int. Conf. on Systems, Man, and Cybernetics, 2000, pp. 3241–3246.
[19] Y. Peng, W. Wei, A new trajectory planning method of redundant manipulator based on adaptive simulated annealing genetic algorithm (ASAGA), in: Proc. of the IEEE Int. Conf. on Computational Intelligence and Security, 2006, pp. 262–265.
[20] Y. Zhang, Z. Sun, T. Yang, Optimal motion generation of a flexible macro–micro manipulator system using genetic algorithm and neural network, in: Proc. of the 2006 IEEE Conf. on Robotics, Automation and Mechatronics, 2006, pp. 1–6.
[21] E.J.S. Pires, J.A.T. Machado, P.B.M. OliveiraY, Manipulator trajectory planning using a MOEA, Applied Soft Computing Journal 7 (3) (2007) 659–667.
[22] O. Castillo, L. Trujillo, P. Melin, Multiple objective genetic algorithms for path-planning optimization in autonomous mobile robots, Soft Computing 11 (2007) 269–279.
[23] R. Saravanan, S. Ramabalan, C. Balamurugan, Evolutionary multi-criteria trajectory modelling of industrial robots in the presence of obstacles, Engineering Applications of Artificial Intelligence 22 (2009) 329–342.
[24] R. Saravanan, S. Ramabalan, N. Ebenezer, C. Dharmaraja, Evolutionary multi criteria design optimization of robot grippers, Applied Soft Computing 9 (2009) 159–172.
[25] E. Masehian, D. Sedighizadeh, Classic and heuristic approaches in robot motion planning—a chronological review, Proceedings of World Academy of Science, Engineering and Technology 23 (2007) 101–106.
[26] D.E. Whitney, Resolved motion rate control of manipulators and human prostheses, IEEE Transactions on Man–Machine Systems MMS-10 2 (1969) 47–53.
[27] B. Siciliano, Kinematic control of redundant robot manipulators: a tutorial, Journal of Intelligent and Robotic Systems 3 (1990) 201–212.
[28] M.G. Marcos, F.B. Duarte, J.A.T. Machado, Complex dynamics in the trajectory control of redundant manipulators, Transactions of Nonlinear Science and Complexity (2006) 134–143.
[29] A. Konak, D.W. Coit, A.E. Smith, Multiobjective optimization using genetic algorithms: a tutorial, Reliability Engineering and System Safety 91 (2006) 992–1007.
[30] J.H. Holland, Adaptation in Natural and Artificial Systems, 2nd ed., University of Michigan Press, Ann Arbor, 1992, Mit Press.
[31] J.D. Schaffer, Multiple objective optimization with vector evaluated genetic algorithms, in: L. Erlbaum (Ed.), Proceedings of the First International Conference on Genetic Algorithms, 1985, pp. 93–100.
[32] C.M. Fonseca, P.J. Fleming, Genetic algorithms for multiobjective optimization: formulation, discussion and generalization, in: Fifth International Conference on Genetic Algorithms, 1993, pp. 416–423.
[33] J. Horn., N. Nafploitis, D. Goldberg, A niched pareto genetic algorithm for multi-objective optimization, in: Proceedings of the first IEEE Conference on Evolutionary Computation, 1994, pp. 82–87.
[34] E. Zitzler, L. Thiele, Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach, IEEE Transactions on Evolutionary Computation 3 (4) (1999) 257–271.
[35] K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, A fast and elitist multiobjective genetic algorithm: NSGA II, IEEE Transactions on Evolutionary Computation 6 (2) (2002) 182–197.
[36] G.G. Yen, H. Lu, Dynamic multiobjective evolutionary algorithm: adaptive cell-based rank and density estimation, IEEE Transactions on Evolutionary Computation 7 (3) (2003) 253–274.
[37] B.V. Babu, B. Anbarasu, Multi-objective Differential Evolution (MODE): An Evolutionary Algorithm for Multi-objective Optimization Problems (MOOPs), 2005, http://discovery.bits-pilani.ac.in/discipline/chemical/BVb/publications.
[38] H. Lu, G.G. Yen, Rank-density-based multiobjective genetic algorithm and benchmark test function study, IEEE Transactions on Evolution Computation 7 (4) (2003) 325–343.
[39] B. Luo, J. Zheng, A new methodology for searching robust Pareto optimal solutions with MOEAs, in: IEEE Congress on Evolutionary Computation, 2008, pp. 580–586.
[40] E.J.S. Pires, P.B.M. Oliveira, J.A.T. Machado, Multi-objective MaxiMin sorting scheme, in: Conference on Evolutionary Multi-criterion Optimization—EMO, Lecture Notes in Computer Science, 3410, Springer-Verlag, 2005, pp. 165–175.
[41] H. Ishibuchi, N. Tsukamoto, Y. Nojima, Evolutionary many-objective optimization: a short review, in: Proc. of 2008 IEEE Congress on Evolutionary Computation, 2008, pp. 2424–2431.
[42] L. Rachmawati, D. Srinivasan, Prefernces incorporation in multi-objective evolutionary algorithms: a survey, in: IEEE Congress on Evolutionary Computation, 2006, pp. 962–968.
[43] I. Das, A preference ordering among various Pareto optimal alternatives Structural Optimization, 18, Springer-Verlag, 1999, pp. 30–35.
[44] Y. Jin, B. Sendhoff, Incorporation of fuzzy preferences into evolutionary multiobjective optimization, in: Proceedings of the 4th Asia Pacific Conference on Simulated Evolution and Learning, 1, 2002, pp. 26–30.
[45] Y. Jin, T. Okabe, B. Sendhoff, Adapting weighted aggregation for multiobjective evolution strategies, in: Proceedings of First International Conference on Evolutionary Multi-criteria Optimization, Lecture Notes in Computer Science, 2001, pp. 96–110.
[46] D. Cvetkovic, I.C. Parmee, Preferences and their application in evolutionary multiobjective optimization, IEEE Transactions on Evolutionary Computation 6 (1) (2002) 42–57.
[47] H. Ishibuchi, Y. Nojima, K. Narukawa, T. Doi, Incorporation of decision maker's preference into evolutionary multiobjective optimization algorithms, in: Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation, 2006, pp. 741–742.
[48] J.C. Ferreira, C.M. Fonseca, A. Gaspar-Cunha, A new methodology to select the preferred solutions from the Pareto-optimal set: application to polymer-extrusion, AIP Conference Proceedings 907 (2007) 861–866.
[49] G.K. Kao, S.H. Jacobson, Finding preferred subsets of Pareto optimal solutions, Journal of Computational Optimization and Applications 40 (1) (2008) 73–95.
[50] K. Deb, H. Gupta, Searching for robust Pareto-optimal solutions in multi-objective optimization, in: C.A. Coello Coello, et al. (Eds.), EMO 2005, LNCS 3410, Springer-Verlag, Berlin/Heidelberg, 2005, pp. 150–164.
[51] B. Luo, J. Zheng, A new methodology for searching robust Pareto optimal solutions with MOEAs, in: IEEE World Congress on Computational Intelligence, 2008, pp. 580–586.
[52] H. Zhao, A multi-objective genetic programming approach to developing Pareto optimal decision trees, Decision Support Systems 43 (2007) 809–826.
[53] J. Sanchis, M.A. Martínez, X. Blasco, Integrated multiobjective optimization and a priori preferences using genetic algorithms, Information Sciences 178 (2008) 931–951.
[54] K. Miettinen, J. Molina, M. González, A. Hernández-Díaz, R. Caballero, Using box indices in supporting comparison in multiobjective optimization, European Journal of Operational Research 197 (2009) 17–24.
[55] J.A. Tenreiro Machado, Calculation of fractional derivatives of noisy data with genetic algorithms, Nonlinear Dynamics 57 (1–2) (2009) 253–260.