THE INFLUENCE OF THE INITIAL JOINT CONFIGURATION ON THE REDUNDANCY RESOLUTION USING SEARCH METHODS

DAN N. DUMITRIU, CORNEL SECARĂ

Abstract. The redundancy problem concerns here a serial planar redundant manipulator with four degrees of freedom. More precisely, the end-effector of this redundant manipulator must achieve the imposed task of following the contour of a curve (*e.g.*, the contour of a circle), while fulfilling two other performance criteria: obstacle avoidance and minimization of a cost function, *i.e.*, a weighted sum of all joint displacements. Numerically, this is a nonlinear optimization problem with nonlinear constraints. Based on Bellman's principle of optimality, the global optimization is divided into a sequence of local optimizations. These local, sequential optimizations are performed using two search methods, *i.e.*, genetic algorithm and direct search. The influence of the initial joint configuration on the cost function minimization is analyzed, in order to find the optimal initial joint configuration.

Key words: serial planar manipulator, redundancy resolution, initial configuration, nonlinear optimization, genetic algorithm, direct search.

1. **INTRODUCTION**

Kinematic redundancy means that the robotic manipulator or other polyarticulated system has more joints than necessary for performing the end-effector task. The extra *degrees of freedom* (DOF) can be used to optimize additional performance criteria, while performing the main end-effector task. These performance criteria can be defined in terms of the kinematic or dynamic parameters and can be related to the different aspects of performance [1]. The purpose is to improve the overall performance of robots in a large variety of tasks.

Nature, in its perfection, provides us with numerous examples of redundancy, *e.g.*, the snake and the human body (composed of several redundant parts like arms and legs). Robotics is often inspired by the nature; it tries to reproduce as much as possible the technical solutions provided by the nature. For example, the snake-like robotic arm, called also the serpentine robot, is conceived as a hyper-redundant robot which can be used for bridge inspection [2, 3]. Humanoid robots are also composed of several redundant parts: the arms, the legs, a 3-DOF articulated spine [4], etc.

Institute of Solid Mechanics of the Romanian Academy, Ctin Mille 15, 010141 Bucharest, E-mail: dumitri04@yahoo.com

Rev. Roum. Sci. Techn. – Méc. Appl., Tome 55, Nº 3, P. 195–210, Bucarest, 2010

In what concerns redundant manipulators for manufacturing and industry, one can distinguish parallel manipulators (*e.g.*, planar 2-DOF redundant parallel manipulator [5]) and serial ones, such as the 4-DOF planar SCARA type manipulator considered as case study in this paper [6–8].

As already mentioned, for redundant manipulators the mapping from the task coordinates to the joint coordinates is not unique. The extra DOF supposed by redundancy can be used to fulfill and to optimize additional performance criteria: obstacle avoidance, cost function minimization, etc. Due to their extra DOF, redundant manipulators are able to ensure obstacle avoidance, while non-redundant manipulators cannot avoid possible collisions with workspace obstacles [9–10]. As for the cost function to be minimized, it is considered in this paper as the weighted sum of all joint displacements, so it is a kinematic cost function.

By introducing the additional criteria (obstacle avoidance and cost function minimization), the redundancy problem takes the form of a non-linear optimization problem with nonlinear constraints. This redundancy problem can be solved either using methods based on inverse kinematics (*e.g.*, the Gradient Projection Method and Extended Jacobian Method [9–11]), or using search methods for solving the direct kinematics (*i.e.*, the direct geometric model with obstacle avoidance and kinematic cost function minimization). This second possibility of redundancy resolution is studied here: the problem formulated as direct kinematics with obstacle avoidance and kinematic cost function minimization is solved using search methods, more precisely genetic algorithm (GA) and direct search (DS). The influence of the initial joint configuration on the cost function minimization is studied as well, in order to find the optimal initial joint configuration.

2. REDUNDANCY RESOLUTION BASED ON DIRECT KINEMATICS

The redundancy resolution proposed in this paper is based on direct kinematics, more precisely on the direct geometric model, *i.e.*, the direct relation (1), at time t_i ($i = 0 \div N_\mathrm{p}$), between the end-effector configuration vector $\mathbf{x}^{(i)} = \mathbf{x}(t_i) = [\, x_1(t_i) \;\; x_2(t_i) \;\; \ldots \;\; x_m(t_i)]^\mathrm{T}$ and the joint coordinates vector $\boldsymbol{\theta}^{(i)} = \boldsymbol{\theta}(t_i) = [\, \theta_1(t_i) \;\; \theta_2(t_i) \;\; \ldots \;\; \theta_n(t_i)]^\mathrm{T}$:

$$\mathbf{x}^{(i)} = f\left(\boldsymbol{\theta}^{(i)}\right), \tag{1}$$

where N_p is the number of imposed end-effector postures on the curve contour to be followed, n is the number of DOF and m is the workspace dimension, with $n > m$ in the case of redundancy.

The redundancy resolution based on the direct geometric model (1) consists of minimizing a kinematic cost function expressed as the sum of all joint displacements during the motion, while imposing two nonlinear constraints to be fulfilled: the end-effector task of following the contour of a curve and the obstacle

avoidance. This nonlinear optimization problem with nonlinear constraints can be formulated as follows:

• Minimize the global cost function J expressed as the weighted sum of all joint displacements required for accomplishing the end-effector task

$$\begin{aligned} J &= \sum_{i=0}^{N_\mathrm{p}-1} \left\| \boldsymbol{\theta}^{(i+1)} - \boldsymbol{\theta}^{(i)} \right\|_{w_1,\ldots,w_n} = \sum_{i=0}^{N_\mathrm{p}-1} \sum_{j=1}^{n} w_j \left\| \theta_j(t_{i+1}) - \theta_j(t_i) \right\| \\ &= \sum_{i=0}^{N_\mathrm{p}-1} \left(w_1 \left\| \theta_1(t_{i+1}) - \theta_1(t_i) \right\| + \ldots + w_n \left\| \theta_n(t_{i+1}) - \theta_n(t_i) \right\| \right) \end{aligned} \tag{2}$$

while

• imposing the end-effector task

$$\left\| \mathbf{x}^{(i)} - \mathbf{x}_\mathrm{d}^{(i)} \right\| \le \varepsilon_\mathrm{d}, \quad i = 1 \div N_\mathrm{p}, \tag{3}$$

• ensuring the obstacle avoidance

$$\min\left(d_{kl}^{(i)} \right) \ge d_0, \quad i = 1 \div N_\mathrm{p}, \quad k = 1 \div N_\mathrm{CCP}, \quad l = 1 \div N_0. \tag{4}$$

In relation (2), w_j ($j = 1 \div n$) are weighting coefficients. In this paper we have considered $w_j = 1, j = 1 \div n$, *i.e.*, all n joint angular displacements count as much in the overall cost function J. In relations (2)–(4), $\boldsymbol{\theta}^{(i)}$ is the joint coordinates vector corresponding to the real end-effector i-th posture $\mathbf{x}^{(i)}$ at time t_i, while $\mathbf{x}_\mathrm{d}^{(i)}$ is the desired end-effector i-th posture on the curve contour to be followed, with ε_d the admissible error between real $\mathbf{x}^{(i)}$ and desired $\mathbf{x}_\mathrm{d}^{(i)}$ end-effector postures. With respect to the obstacle avoidance constraint (4), the minimum of the $d_{kl}^{(i)}$ distances must be greater than a desired d_0 distance imposed by the user. The $d_{kl}^{(i)}$ distances are calculated between the k-th Configuration Control Point (CCP) and the l-th obstacle, where N_CCP is the number of the CCP and N_O is the number of obstacles. The CCP are imposed by the user on the manipulator structure.

3. SEQUENTIAL OPTIMIZATION STRATEGY

The redundancy problem of minimizing the global cost function (2), subject to constraints (3) and (4), represents a global nonlinear optimization problem. It is global in the sense that the minimization process concerns J, *i.e.*, the overall sum of the joint displacements for all N_p imposed end-effector postures. But this global optimization, concerning all N_p imposed end-effector postures, is almost

impossible to realize by using search methods. In fact, for each imposed end-effector posture there are *n* optimization variables (the number of DOF), and let us consider that each optimization variable is searched among N_{search} possible values (*e.g.*, $N_{\text{search}} = 20$ is quite reasonable). Thus, a search method will have to find the minimum of *J* among $N_{\text{search}}{}^{n \cdot N_{\text{p}}}$ possibilities. The case study in this paper concerns a redundant manipulator with $n = 4$ DOF. If only $N_{\text{p}} = 10$ end-effector postures would have been imposed (in fact this paper's case study considers $N_{\text{p}} = 120$, so much bigger), then a number of $N_{\text{search}}{}^{n \cdot N_{\text{p}}} = 20^{4 \cdot 10} = 20^{40} \approx 10^{52}$ computations of *J* would have to be performed, which is far too much for nowadays Central Processing Units (CPUs). In fact, the most modern parallel CPU can hardly attend 10^{12} IPS (Instructions Per Second), which is not even comparable with the global direct search need of $\approx 10^{52}$ IPS.

Faced with this situation, the solution is to replace the global minimization with a sequence of N_{p} local minimizations $J_{N_{\text{H}}}^{(i)}$, $i = 0 \div N_{\text{p}} - 1$. More precisely, at each time t_i corresponding to *i*-th imposed end-effector posture, the following local cost function $J_{N_{\text{H}}}^{(i)}$ has to be minimized:

$$\begin{aligned} J_{N_{\text{H}}}^{(i)} &= \sum_{k=i}^{i+N_{\text{H}}-1} \left\| \boldsymbol{\theta}^{k+1} - \boldsymbol{\theta}^{k} \right\|_{w_1,\ldots,w_n} = \sum_{k=i}^{i+N_{\text{H}}-1} \sum_{j=1}^{n} w_j \left\| \theta_j(t_{k+1}) - \theta_j(t_k) \right\| = \\ &= \sum_{k=i}^{i+N_{\text{H}}-1} \left(w_1 \left\| \theta_1(t_{k+1}) - \theta_1(t_k) \right\| + \ldots + w_n \left\| \theta_n(t_{k+1}) - \theta_n(t_k) \right\| \right), \end{aligned} \tag{5}$$

where N_{H} stands for the "horizon" of the local optimization, indicating how many steps forward will be considered in the local cost function to minimize. Of course, the upper summation $i + N_{\text{H}} - 1$ in (5) must not exceed N_{p}, so it would be more correct to use $\min(i + N_{\text{H}} - 1, N_{\text{p}})$ as upper summation index in (5), instead of $i + N_{\text{H}} - 1$.

Let us exemplify for $N_{\text{H}} = 1, 2, 3$:

- if a 1-step forward local optimization is considered, *i.e.*, the "horizon" $N_{\text{H}} = 1$, then the following local cost function $J_1^{(i)}$ will be minimized at each time t_i:

$$J_1^{(i)} = \left\| \boldsymbol{\theta}^{(i+1)} - \boldsymbol{\theta}^{(i)} \right\|_{w_1,\ldots,w_n} = \sum_{j=1}^{n} w_j \left\| \theta_j(t_{i+1}) - \theta_j(t_i) \right\|, \quad i = 0 \div N_{\text{p}} - 1; \tag{6}$$

• if a 2-steps forward local optimization is considered, *i.e.*, the "horizon" $N_{\mathrm{H}}=2$, then the following local cost function $J_2^{(i)}$ will be minimized at each time t_i, $i=0\div N_{\mathrm{p}}-2$:

$$J_2^{(i)}=\sum_{k=i}^{i+1}\left\|\boldsymbol{\theta}^{(k+1)}-\boldsymbol{\theta}^{(k)}\right\|_{w_1,\ldots,w_n}=\sum_{j=1}^{n}w_j\left(\left\|\theta_j(t_{i+1})-\theta_j(t_i)\right\|+\left\|\theta_j(t_{i+2})-\theta_j(t_{i+1})\right\|\right)$$

• if a 3-steps forward local optimization is considered, *i.e.*, the "horizon" $N_{\mathrm{H}}=3$, then the following local cost function $J_3^{(i)}$ will be minimized at each time t_i, $i=0\div N_{\mathrm{p}}-3$:

$$J_3^{(i)}=\sum_{k=i}^{i+2}\left\|\boldsymbol{\theta}^{(k+1)}-\boldsymbol{\theta}^{(k)}\right\|_{w_1,\ldots,w_n}=$$

$$=\sum_{j=1}^{n}w_j\left(\left\|\theta_j(t_{i+1})-\theta_j(t_i)\right\|+\left\|\theta_j(t_{i+2})-\theta_j(t_{i+1})\right\|+\left\|\theta_j(t_{i+3})-\theta_j(t_{i+2})\right\|\right).$$

In this paper, only a 1-step forward local optimization is considered ("horizon" $N_{\mathrm{H}}=1$), using the local cost function $J_1^{(i)}$ given by (6), at each time t_i, $i=0\div N_{\mathrm{p}}-1$. Of course, the minimization of $J_1^{(i)}$ given by (6) is performed by imposing the end-effector task and by ensuring obstacle avoidance, *i.e.*, constraints (3) and (4) must be fulfilled at time t_{i+1}, $i+1=1\div N_{\mathrm{p}}$:

$$\left\|\mathbf{x}^{(i+1)}-\mathbf{x}_{\mathrm{d}}^{(i+1)}\right\|\le\varepsilon_{\mathrm{d}}\ \text{ (constraint imposing the end-effector task),}\qquad(7)$$

$$\min\left(d_{kl}^{(i+1)}\right)\ge d_0,\ \ k=1\div N_{\mathrm{CCP}},\ \ l=1\div N_0\ \text{ (for obstacle avoidance).}\qquad(8)$$

This sequential optimization strategy is iterative; the joint configuration computed in the previous step represents the current point around which the search method will be performed again, iteratively.

Without proving it rigorously, let us explain why a sequential optimization strategy can provide a result quite close to the result of the global optimization. Our proof relies on the principle of optimality introduced by Richard Bellman as axiomatic basis for the dynamic programming [12]. According to Bellman's principle of optimality, an optimal "strategy" can only be constituted of optimal actions, implying that, for every initial state and initial action, the next actions must represent optimal actions in relation to the intermediary state that results from the first action [12, 13]. By "strategy" one understands here a certain succession of actions from the initial to the final state. This axiomatic principle of optimality of

Bellman appears almost as a truism. In fact, if an optimal strategy includes a non-optimal action, it seems possible to replace this non-optimal action with the optimal one, thus apparently improving the strategy. However, this replacement could affect the performance of neighboring actions, and finally the new strategy could turn out to be worse. Nevertheless, numerous numerical tests have proved the practical efficiency of Bellman's principle of optimality.

Of course, a successful global optimization will always be better than a sequential one, but has the drawback of requiring unaffordable computational resources/time when using search methods.

The proposed sequential optimization strategy is summarized in Fig. 1. Remark that an optimal initial joint configuration, found as indicated in §6, is used. The sequential optimization consists of optimizing each action in strict relation with the previous one and with the perspective of the actions to be taken from now on (a $N_{\mathrm{H}} = 1$-step "horizon" is considered), *i.e.*, minimizing $J_1^{(i)}$ given by (6) under constraints (7)–(8) and starting from $\boldsymbol{\theta}_i$ issued from the previous minimization under constraints of $J_1^{(i-1)}$.

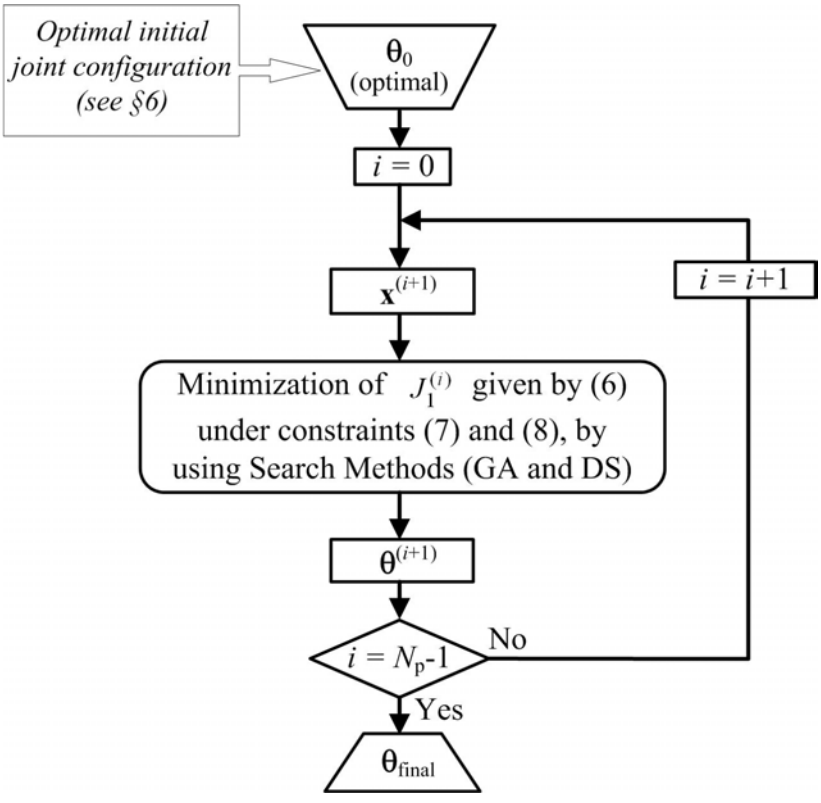

Fig. 1 – Proposed sequential optimization strategy, based on search methods.

4. SEARCH METHODS: GENETIC ALGORITHM AND DIRECT SEARCH

The sequential minimization of the kinematic cost function (6), under nonlinear constraints (7) and (8), is performed here using search methods. Several search methods are available in the literature: genetic algorithms [6–7, 14–18], direct search [1, 8, 19], neural networks [18], fuzzy techniques [20], etc. In this paper, only a genetic algorithm and direct search were used.

Among the evolutionary algorithms, the genetic algorithms (GAs) represent one of the most popular and efficient probabilistic search methods for cost function minimization, with or without constraints (additional nonlinear constraints may always be specified). GAs are able to find the global optimum in complex spaces, based on the mechanisms of natural genetics and natural selection, characteristic to biological structures. They start with an initial population of candidate solutions and then iteratively evolve towards better problem solutions, by survival of the fittest among string structures obtained by applying genetic operators like mutation and crossover [21–24]. As the other search methods, GAs need no previous experience on the problem and use only cost function information, without requiring any information about the gradient of the cost function.

A simple GA requires [21–24]:

- a genetic representation of the potential problem solutions: the parameters of the optimization problem are coded as a finite-length string over some finite alphabet (*e.g.*, binary alphabet). Such a string is called *chromosome* or *individual*. The elements of the chromosome are called *genes* (*bits* in the binary case). A *population* consists of a group of chromosomes;
- a method for generating an initial population of solutions: random generation, or more elaborated method using previous experience so that either to reduce the search domain or to place the solution nearer to the optimum, etc;
- the *fitness function*, gathering in our case the kinematic cost function (6) and the constraints (7) and (8) to be fulfilled. The fitness function determines the survival or elimination of a chromosome from the population. Through several repetitions, the evolution of the chromosomes leads to the domination of stronger ones;
- the genetic operators: *selection*, *elitism*, *crossover* and *mutation*;
- values for the parameters used by GA: population size, probabilities of applying the genetic operators, etc.

As already mentioned, the applied genetic operators in each step are [21-24]:

– *Selection.* The selection function chooses the best parents for the next generation, based on their fitness values. The usual selection functions are: *roulette* and *tournament*. The tournament function was used here: each parent is selected by choosing the best chromosome out of a randomly generated set of chromosomes. The *tournament size* parameter specifies the number of chromosomes from which only the best one is selected.

- *Elitism*. In order to preserve the best (optimum) chromosome of each generation for the next generation, the elitism operator is used. The goal is to keep the best chromosome of all previous generations in the current population and avoid the possibility of losing good chromosomes. The *elite count* parameter is a positive integer specifying how many chromosomes in the current generation are guaranteed to survive in the next generation.
- *Crossover*. This genetic operator combines two parent chromosomes, so that to form a new child chromosome for the next generation. The most common method uses *a single point crossover* operator. This operator chooses a single breaking point, i.e., a random integer number between 1 and the number of genes indicating the position of the gene where the crossover between the two parents is operated. At this breaking point, the two parent chromosomes are swapping segments of genes with same size and position. Two child chromosomes can be generated in this way: a first child with the first part of genes taken from the first parent and the second part of genes taken from the second parent, respectively a second child with the first part of genes taken from the second parent and the second part of genes taken from the first parent.
- *Mutation*. The jump mutation function makes small random changes in the chromosomes of the population, thus providing genetic diversity. It operates on each bit (gene) of each chromosome and reverses the value of 1 to 0 and conversely. The *mutation probability* operator indicates the probability that a jump mutation is operated on the current gene of a chromosome.

The results provided in §6, corresponding to the case study presented in §5, have been obtained using the Matlab 7.6 GA toolbox [25], with the following values of the GA parameters/operators:

- initial population: randomly generated;
- population size: 50;
- selection function: tournament;
- tournament size: 4;
- elite count: 5;
- crossover: single point;
- mutation probability: 0.03.

For the case study in §5, by applying the GA at time t_i, $i = 0 \div N_\mathrm{p} - 1$, the GA parameters to be found are the 4 joint coordinates $\theta_1(t_{i+1})$, $\theta_2(t_{i+1})$, $\theta_3(t_{i+1})$ and $\theta_4(t_{i+1})$. Each of the 4 parameters to be found has been coded on 8 bits, so the permitted interval between the lower and the upper bounds for each parameter was divided into $2^8 = 256$ equal angular subintervals, leading to a total chromosome length of 9×8 = 72 bits.

In most cases, the GA successfully converges towards the global minimum of the cost function (6) while the expressions of the nonlinear constraints (7) and (8)

are accomplished. But this convergence to the global minimum is not guaranteed. Thus, the GAs have generally two main disadvantages: 1) they do not necessarily find the exact global minimum; 2) convergence may be slow, requiring a large, unknown number of evaluations of the cost function, thus GAs cannot assure constant optimization response times [20–24].

These two disadvantages of GAs can be eliminated by simply using the direct search (DS) method. This basic method searches a set of points around the current point, looking for one where the value of the cost function is lower than the value at the current point. Of course, as for GAs, it does not require any information about the gradient of the cost function. The use of the DS method is appropriate only if the dimension of the problem is not too big, so that to avoid the exponential increase of the computational time (its main disadvantage) [1, 8, 19].

The Matlab 7.6 DS toolbox [25] was used here, with the following parameters:

- poll method: pattern search algorithm;
- polling order: consecutive;
- initial mesh size = 1;
- maximum mesh size: infinite;
- mesh contractor factor = 0.5;
- mesh expansion factor = 2;
- mesh contraction factor = 0.5.

The results provided by the two search methods, *i.e.*, GA and DS, are compared in §6.

5. CASE STUDY: 4-DOF PLANAR REDUNDANT MANIPULATOR

The simulations were performed on a laboratory model of planar redundant manipulator, possessing $n = 4$ DOF, *i.e.*, the joint coordinates $\theta_1(t_i)$, $\theta_2(t_i)$, $\theta_3(t_i)$ and $\theta_4(t_i)$, $i = 0 \div N_{\mathrm{p}}$. The operational space dimension is $m = 3$ because the position and orientation of the end-effector are both taken into account. Thus, the degree of redundancy is $n - m = 1$.

This laboratory manipulator can be modeled as shown in Fig. 2 [6]. It has the following geometric characteristics (*i.e.*, the lengths of the four links):

$$l_1 = 0.12\,\mathrm{m};\ \ l_2 = 0.12\,\mathrm{m};\ \ l_3 = 0.10\,\mathrm{m};\ \ l_4 = 0.05\,\mathrm{m}.$$

The manipulator's base is considered fixed, with coordinates $x_0 = 0$ and $y_0 = 0$. A further work will vary also this fixed position of the manipulator base in order to obtain a further, supplementary reduction of the kinematic cost function.

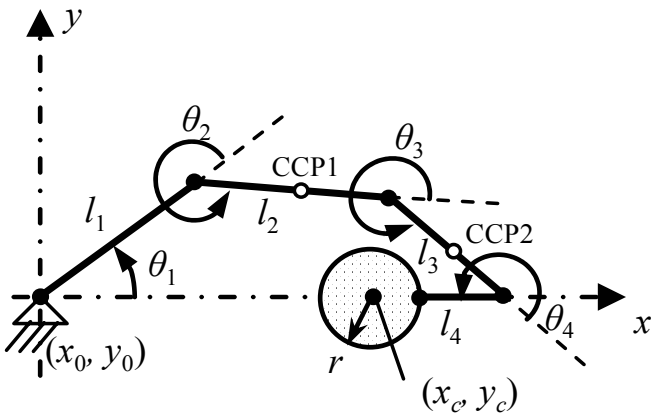

Fig. 2 – Model of the 4-DOF planar redundant manipulator.

The task of this manipulator is to generate the references (positions and orientations) of the end-effector along the contour of a circle of radius *r*, whose surface is considered to be restrictive for all four elements of the manipulator structure. The contour of the circle to be followed is defined by the radius *r* of the restriction circle and the coordinates of its center, as follows:

$$r = 0.03\,\text{m};\quad x_\text{c} = 0.2\,\text{m};\quad y_\text{c} = 0.$$

The end-effector references generation is a function of the sampling step of generation *i*, with $i = 1 \div N_\text{p}$:

$$x_\text{d}^{(i)} = x_\text{c} + r \cdot \cos(i \cdot \Delta\alpha);$$
$$y_\text{d}^{(i)} = y_\text{c} + r \cdot \sin(i \cdot \Delta\alpha);$$
$$\Sigma\theta_\text{d}^{(i)} = -\pi + i \cdot \Delta\alpha,$$

where $\Delta\alpha$ is the angular step of generation. The angular step of generation considered here is $\Delta\alpha = 3°$. In this case, the number of steps (between two consecutive imposed end-effector postures) to entirely cover the circle is N_p=120 .

The imposed positioning and orientation error of the end-effector, used in constraint (7) imposing the end-effector task, is ε_d=$\left(0.001\text{m}\quad 0.001\text{m}\quad 0.1°\right)$.

At each time t_i, $i = 1 \div N_\text{p}$, GA or DS is searching for the values $\boldsymbol{\theta}^{(i+1)} = [\,\theta_1^{(i+1)}\ \ \theta_2^{(i+1)}\ \ \theta_3^{(i+1)}\ \ \theta_4^{(i+1)}\,]^\text{T}$ of the joint coordinates vector at time t_{i+1}, by minimizing the cost function $J_1^{(i)}$ given by (6) under nonlinear constraints (7) and (8). In practice, this search is performed only between some lower and upper bounds with respect to the previous $\boldsymbol{\theta}^{(i)}$, more precisely one searches:

$$\boldsymbol{\theta}^{(i+1)} \in \left[\boldsymbol{\theta}^{(i)} - \Delta\boldsymbol{\theta};\ \boldsymbol{\theta}^{(i)} + \Delta\boldsymbol{\theta}\right],$$

where $\boldsymbol{\theta}^{(i)} - \Delta\boldsymbol{\theta}$ defines the lower bound, while $\boldsymbol{\theta}^{(i)} + \Delta\boldsymbol{\theta}$ defines the upper bound of the domain in which the solution of the nonlinear optimization with nonlinear constraints problem is searched using GA or DS. In this paper, it was considered a constant vector $\Delta\boldsymbol{\theta} = [4°\ \ 7°\ \ 8°\ \ 4°]$, for $\forall\, i = 1 \div N_{\mathrm{p}}$, used to define the lower and upper bounds of the search method at time t_i.

The end-effector coordinates (position and also orientation) are obtained from the direct geometric model (1), which in our case study is as follows:

$$x_1(t_i) = x^{(i)} = \sum_{j=1}^{4} l_j \cdot \cos\left(\sum_{k=1}^{j} \theta_k^{(i)}\right);\quad x_2(t_i) = y^{(i)} = \sum_{j=1}^{4} l_j \cdot \sin\left(\sum_{k=1}^{j} \theta_k^{(i)}\right) - \text{for position,}$$

$$x_3(t_i) = \Sigma\theta^{(i)} = \sum_{j=1}^{4} \theta_j^{(i)} \quad - \text{for orientation.}$$

Two CCP are considered in our case study ($N_{\mathrm{CCP}} = 2$), placed in the middle of second element and, respectively, in the middle of the third element of the manipulator. The desired distance imposed by the user is $d_0 = 0.015\,\mathrm{m}$ (used in constraint (8) imposing the obstacle avoidance).

6. RESULTS: INFLUENCE OF THE INITIAL JOINT CONFIGURATION

The simulations were performed for the case study detailed in §5. Both GA and DS were used to minimize, at each time t_i, $i = 0 \div N_{\mathrm{p}} - 1$, the kinematic cost function $J_1^{(i)}$ given by (6), under nonlinear constraints (7) and (8), imposing the end-effector task and respectively the obstacle avoidance.

An optimization study was realized, involving the initial joint configuration. Since the manipulator has one degree of redundancy, one has to choose among a variety of initial configurations. Let us consider the initial first joint coordinate $\theta_1^{(0)} = \theta_1(t_0 = 0)$ as variable corresponding to one initial configuration or another. The goal of our optimization study is to find the optimal $\theta_{1,\mathrm{opt}}^{(0)}$ for which corresponds the lowest kinematic cost function $J_1^{(i)}$. This study concerning the influence of the initial joint configuration on $J = \sum_{i=0}^{N_{\mathrm{p}}-1} J_1^{(i)}$ is performed using both GA and respectively DS. The initial first joint coordinate is varied between –25°

and $50°$; for each $\theta_1^{(0)}$ in this interval, the corresponding initial configuration is shown in Fig. 3.

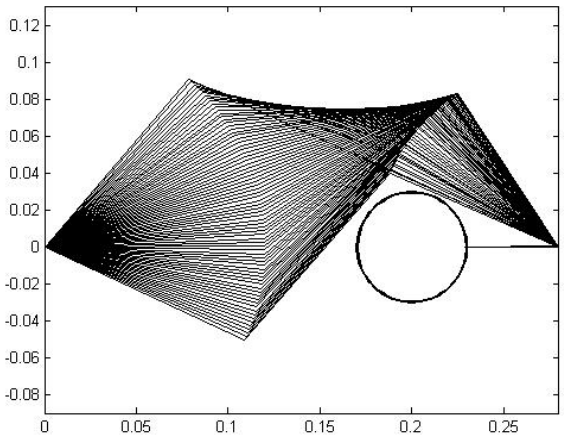

Fig. 3 – Different initial configurations for different $\theta_1^{(0)} \in [-25°; 50°]$, in *x*-*y* plane.

So, the optimization study consists to find $\theta_{1,\text{opt}}^{(0)} \in [-25°; 50°]$ minimizing $J = \sum_{i=0}^{N_p-1} J_1^{(i)}$, under the nonlinear constraints imposing the end-effector task and respectively the obstacle avoidance. Fig. 4 shows the results of this optimization study above.

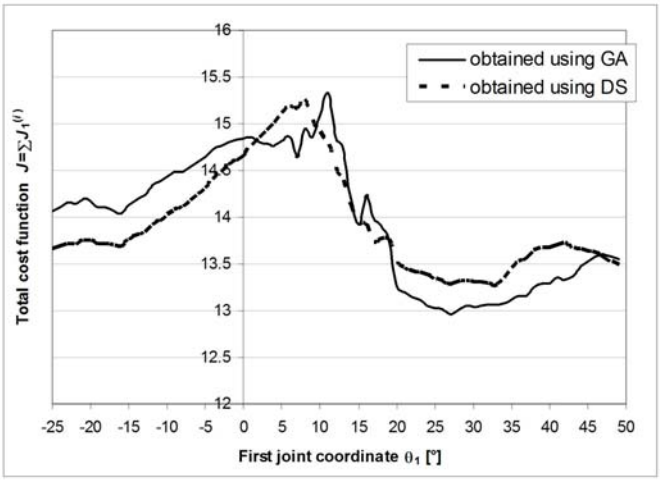

Fig. 4 – Influence of the initial joint configuration on the total cost function *J*.

The two search methods, GA and DS, provide quite similar results: $\theta_{1,\text{opt}}^{(0)} \cong 32°$ is the optimal initial first joint coordinate, for which a minimum $J_{\min} \cong 13$ rad of the kinematic cost function is found. Since the average of the total cost function is $\overline{J} \cong 14$ rad, this means that by optimizing the initial joint configuration one obtains a reduction of the total cost function J of $\cong \frac{14-13}{14} \cong 7\%$. For this $\theta_{1,\text{opt}}^{(0)} \cong 32°$, the initial posture of the manipulator is given by the following initial joint coordinates vector:

$$\boldsymbol{\theta}^{(0)} = \left[32° \ -23.49° \ -61.61° \ -126.46°\right] =$$
$$= \left[0.5585 \ -0.4177 \ -1.0753 \ -2.2071\right] \text{rad}.$$

Starting from this initial posture, the kinematic redundancy resolution using GA and DS is performed, considering at each time t_i, $i = 0 \div N_\text{p} - 1$, the minimization of the kinematic cost function (6), the achievement of the end-effector task (7) and the obstacle avoidance (8). The kinematics of the manipulator is presented in Fig. 5, showing the evolutions of the joint coordinates $\theta_1(t_i)$, $\theta_2(t_i)$, $\theta_3(t_i)$ and $\theta_4(t_i)$, $i = 0 \div N_\text{p}$. One can remark that the results obtained using GA are similar with the ones obtained using DS. Fig. 6 presents the motion of the redundant manipulator, when achieving the end-effector task, with obstacle avoidance and kinematic cost function minimization ($J_{\min} \cong 13$ rad is obtained).

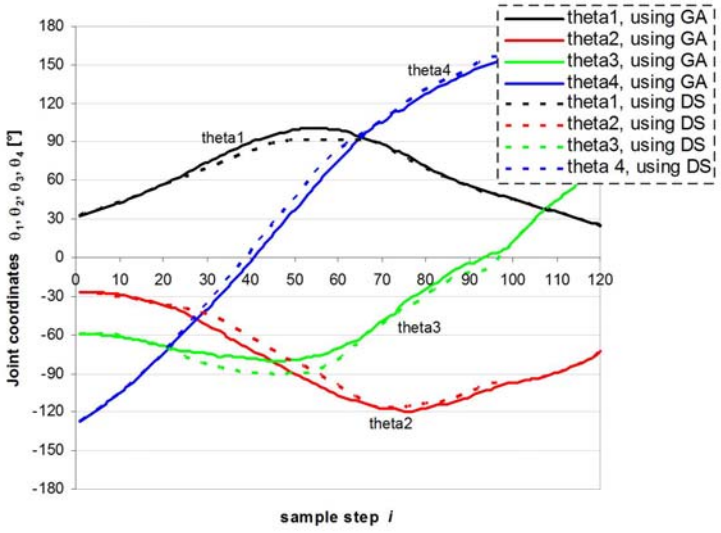

Fig. 5 – Evolution of the joint coordinates for end-effector task achievement, with obstacle avoidance and kinematic cost function minimization.

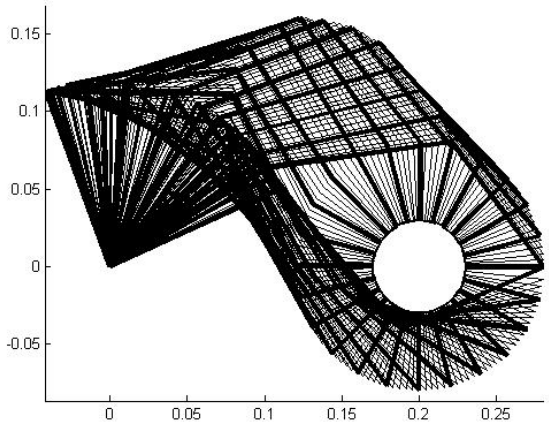

Fig. 6 – Motion of the redundant manipulator in *x-y* plane, when achieving the end-effector task, with obstacle avoidance and kinematic cost function minimization.

7. **CONCLUSIONS**

The goal of the kinematic redundancy resolution is to accomplish the imposed end-effector task of following the contour of a curve, while ensuring obstacle avoidance and minimizing the sum of the joint displacements required for accomplishing the end-effector task. This is a nonlinear optimization problem with nonlinear constraints.

This paper presents a sequential optimization strategy for kinematic redundancy resolution, using search methods, more precisely GA and respectively DS. The joint configuration computed in the previous step represents the current point around which the search method will be performed again, iteratively. Thus, the global optimization problem can be successfully decomposed into a sequence of local optimization sub-problems.

The results obtained in this paper show insignificant differences between the two search methods tested, so we conclude that both GA and DS are satisfactory to solve the sequential optimization problem.

The influence of the initial joint configuration on the total cost function is studied as well. This optimization study led to a reduction of $\cong 7\%$ of the total kinematic cost function.

So far, the manipulator's base is considered as fixed. Further work will vary also the position of the manipulator base, in order to obtain further reductions of the total kinematic cost function. The goal is to optimize as much as possible the redundant manipulation tasks.

Acknowledgements. The authors gratefully acknowledge the financial support of the National Authority for Scientific Research (ANCS, UEFISCSU), Romania, through PN-II research project no.

106/2007. Dan Dumitriu gratefully acknowledges the financial support of UEFISCDI for PN-II grant no. 72194/2008.

Received on October 15, 2010

REFERENCES

1. KAPOOR, C., CETIN, M., TESAR, D., *Performance Based Redundancy Resolution with Multiple Criteria*, Proceedings of the 1998 ASME Design Engineering Technical Conference (DETC'98), Georgia, USA, September 13-16, 1998.
2. SOFGE, D., CHIANG, G., *Design, Implementation, and Cooperative Coevolution of an Autonomous/Teleoperated Control System for a Serpentine Robotic Manipulator*, Proceedings of the American Nuclear Society Ninth Topical Meeting on Robotics and Remote Systems, Seattle, Washington, March 2001.
3. CHOSET, H., *Bridge Inspection with Serpentine Robots*, Final Report for Highway IDEA Project 56, Carnegie Mellon University, 2000.
4. ROOS, L., GUENTER, F., GUIGNARD, A., BILLARD, A., *Design of a Biomimetic Spine for the Humanoid Robot Robota*, IEEE/RAS-EMBS International Conference on Biomedical Robotics and Biomechatronics, Pisa, Italy, February 2006.
5. YIU, Y.K., LI, Z.X., *Dynamics of a Planar 2-dof Redundant Parallel Robot*, International Conference on Mechatronics Technology, Singapore, June 2001.
6. SECARA, C., CHIROIU, V., DUMITRIU, D., *Obstacle avoidance by a laboratory model of redundant manipulator using a genetic algorithm based strategy*, Proceedings of IV-th National Conference THE ACADEMIC DAYS of the Academy of Technical Sciences in Romania, November 19-20, 2009, Iaşi, pp. 199-204, AGIR Publishing House, 2009.
7. SECARĂ, C., VLĂDĂREANU, L., *Iterative strategies for obstacle avoidance of a redundant manipulator*, WSEAS Transactions on Mathematics, **9**, *3*, pp. 211-221, 2010.
8. SECARĂ, C., DUMITRIU, D., *Direct Search based strategy for obstacle avoidance of a redundant manipulator*, Analele Universităţii "Eftimie Murgu" Reşiţa, **XVII**, *1*, pp. 11-20, 2010.
9. SEZGIN, U., SENEVIRATNE, L.D., EARLES, S.W.E., *Collision Avoidance in Multiple-Redundant Manipulators*, International Journal of Robotics Research, **16**, *5*, pp. 714-724, 1997.
10. PERDEREAU, V., DROUIN, M., PASSI, C., *Real-Time Collision Avoidance for Redundant Manipulators*, Proceedings of the IASTED International Conference Robotics and Applications 2000, Honolulu, Hawaii, USA, August 14-16, 2000.
11. CHIACCHIO, P., CHIAVERINI, S., SCIAVICCO, L., SICILIANO, B., *Closed-Loop Inverse Kinematics Schemes for Constrained Redundant Manipulators with Task Space Augmentation and Task Priority Strategy*, International Journal of Robotics Research, **10**, *4*, pp. 410-425, 1991.
12. BELLMAN, R., *Dynamic Programming*, Princeton University Press, Princeton, NJ, 1957.
13. CHEVALIER, A., *La Programmation Dynamique*, Dunod, Paris, 1977.
14. MITSI, S., BOUZAKIS, K.-D., SAGRIS, D., MANSOUR, G., *Determination of Optimum Robot Base Location considering Discrete End-effector Positions by means of Hybrid Genetic Algorithm*, Robotics and Computer-Integrated Manufacturing, **24**, *1*, pp. 50-59, 2008.
15. MITSI, S., BOUZAKIS, K.-D., SAGRIS, D., MANSOUR, G., *A Multi-objetive Optimum Robot Base Placement using a Hybrid Genetic Algorithm*, The Ninth IFToMM International Symposium on Theory of Machines and Mechanisms, Bucharest, September 2005.
16. NEARCHOU, A., *Solving the Inverse Kinematics Problem of Redundant Robots Operating in Complex Environments via a Modified Genetic Algorithm*, Mech. Mach. Theory, **33**, *3*, pp. 273-292, 1998.

17. TIAN, L., COLLINS, C., *Motion Planning for Redundant Manipulators using a Floating Point Genetic Algorithm*, Journal of Intelligent and Robotic Systems, **38**, pp. 297-312, 2003.
18. DU, X., CHEN, H.-H., GU, W.-K., *Neural Network and Genetic Algorithm Based Global Path Planning in a Static Environment*, Journal of Zhejiang University Science, **6A**, *6*, pp. 549-554, 2005.
19. ATA, A.A., MYO, R.T., *Optimal Point-to-Point Trajectory Tracking of Redundant Manipulators using Generalized Pattern Search*, International Journal of Advanced Robotic Systems, **2**, *3*, pp. 239-244, 2005.
20. HAM, C., QU, Z., JOHNSON, R., *Robust Fuzzy Control for Robot Manipulators*, IEEE Proc.-Control Theory Applications, **147**, *2*, pp. 212-216, 2000.
21. GOLDBERG, D.E., *Genetic Algorithms in Search, Optimization & Machine Learning*, Addison Wesley, Reading, MA, 1989.
22. COLEY, D., *An Introduction to genetic algorithms for scientists and engineers*, World Scientific Press, 1999.
23. DEB, K., *Multi-Objective Optimization using Evolutionary Algorithms*, John Wiley & Sons Ltd., 2001.
24. AGAPIE, A., *Evolutionary Algorithms – Modeling and Convergence*, Romanian Academy Publishing House, Bucharest, 2007.
25. * * *, MATLAB 7.6 (Release 2008a), *Genetic Algorithm and Direct Search Toolbox 2.3*, The MathWorks, 2008.