Robotica (2016) © Cambridge University Press 2016
doi:10.1017/xxxx

Evaluating Concurrent Design Approaches for a Delta Parallel Manipulator

Salvador Botello-Aceves†, S. Ivvan Valdez†, Héctor M. Becerra† and Eusebio Hernandez*§

†*Center for Research in Mathematics (CIMAT), Guanajuato, Mexico*
§*Instituto Politecnico Nacional, SEPI ESIME Ticoman, Mexico*

(Accepted MONTH DAY, YEAR. First published online: MONTH DAY, YEAR)

SUMMARY
This paper addresses the problem of optimal mechanisms design, for the geometric structure and control parameters of mechanisms with complex kinematics, which is one of the most intricate problems in contemporary robot modeling. The problem is stated by means of task requirements and performance constraints which are specified in terms of the end-effector's position and orientation to accomplish the task. Usually, this problem does not fulfill the characteristics needed to use gradient-based optimization algorithms. In order to circumvent this issue, we introduce case studies of optimization models using evolutionary algorithms (EAs), which deal with the concurrent optimization of both: structure and control parameters. We define and review several optimization models based on the workspace, task and dexterity requirements, such that they guarantee an adequate performance under a set of operating and joint constraints, for a Delta parallel manipulator. Then, we apply several methodologies that can approximate optimal designs. Additionally, we compare the EAs with a quasi-Newton method (the BFGS), in order to show that the last kind of methods is not capable of solving the problem if the initial point is not very close to a local optimum. The results provide directions about the best state-of-the-art EA for addressing different design problems.

KEYWORDS: Design; Dimensional synthesis; Evolutionary algorithms; Mechanisms with complex kinematics; Delta robot; Concurrent design, CMA-ES, GA, BUMDA.

1. Introduction

Optimal mechanism design can be defined as finding design parameters which performs the best for a given task. The parameters to be optimized can be type synthesis, dimensional synthesis, masses, actuator specifications, control parameters, etcetera. A concurrent design optimization problem for actuated mechanisms is to simultaneously find two or more sets of optimal parameters which belong to different types of parameters, by instance, in this paper, we tackle the problem of concurrently finding control gains and geometric parameters that best perform on a desired task. This kind of methodologies have been applied to high-speed mechanisms for position approaching using dynamic models.[1,2] The performance can be measured by computing the error with respect to a given path, or the percentage of the reached volume in a given workspace, or the widest regular region which can be covered by the manipulator. In addition, the mechanism movements are, usually, constrained by joint limits and dexterity constraints.

Optimal design methodologies could be considered as consolidated for serial mechanisms,[3] which are nowadays exploited in numerous successful industrial applications.[4–6] Even though, parallel manipulators exhibit inherent desirable features

* Corresponding author. E-mail: euhernandezm@ipn.mx

as lower inertia, lower moving masses and high velocity and acceleration capabilities, they have not reached the same total market as their serial counterparts. In general, parallel manipulators are devoted to few applications, namely for positioning in machine tools and telescopes, or for pick-and-place operations. A possible reason is that they do not perform the desired tasks as expected, because the design is not well suited for such particular task, hence, a crucial issue for an adequate performance of the parallel manipulators is the optimal choice of the geometric design parameters,[7] this problem is referred as dimensional synthesis. In designing a robotic mechanism in a sequential classical way, even with fixed structure parameters, a large number of design variables can be involved. In a classical design methodology, the mechanical structure and the control parameters are designed separately. It is desirable that the control algorithm can be implemented by using an automatic tuning of control gains, without a tedious and time-consuming tuning procedure. This suggests that if the robotic structure parameters and control parameters are simultaneously designed using an optimization criterion, then a better overall performance of the robotic system can be obtained. The need of this type of concurrent design methodology is therefore highlighted into the robotics area.

1.1. Literature review

Searching for the best robot design and verifying that such design fulfills the system requirements is not a simple task for a human designer. As a consequence, there is a wide set of approaches, in the specialized literature, that intend to circumvent this issue by proposing algorithmic methodologies for automatic optimization for representative parallel kinematic structures like planar manipulators,[8,9] and spatial (3D) manipulators.[10–13] Design problems could be categorized by means of the order of the differential equations that have to be solved for simulating the robotic task, since it is directly related with the complexity of the problem and the task requirements. First, let us introduce the *static design problem*. This kind of problems can be described as optimization problems where, actually, there is no differential equation, but an algebraic equation (or a differential equation of order 0) must be solved. Therefore, the task only depends on the kinematic equations, commonly used for maximizing a desirable workspace, considering joint and dexterity constraints, and collision avoidance. In this paper, the static design problem of a Delta parallel manipulator is defined as the maximization of a regular workspace. This design problem can be formulated from kinetostatic performance indices used to bound the serviceable workspace. A regular workspace is a reachable volume inside a regular 3D figure, which has some faces equal and all internal angles are equal. For the particular case of our experiments we use a parallelepiped. Recently, some optimization problems were formulated taking into account these quality indices, such as dexterity,[14,15] stiffness[16] and manipulability.[17]

The static problem is of interest for robotics researchers considering the effective regular workspace, by instance, a research aimed to find optimal design parameters of a Delta robot and a Gough-Stewart platform.[14] The authors used the controlled random search (CRS) method, introducing a collision avoidance constraint, which is simplified by adjusting the limits of the joints. More recently, a sequential quadratic programming (SQP), a CRS, a genetic algorithm (GA), a differential evolution (DE) and a particle swarm optimization (PSO) were applied to optimization of the geometric parameters of a Delta robot and a Stewart-Gough platform.[18] The algorithms were compared without a statistical test. In other work, a simulated annealing procedure that generates optimal approximations to Delta-manipulator structures is defined, developed and tested.[19] A similar approach is implemented by applying the previous process to optimize a Delta robot, using a randomized linear search method.[20] With the same aim, other researchers have used a GA[21] or a quasi-Newton algorithm.[22]

Regarding on a second class of design problems, namely the *kinematic design problem*, the basic idea is to solve the first-order differential equations that represent the differential kinematics of the robot. In this case, a typical problem is the tracking of a trajectory imposed by the user, which seeks to minimize the error between the desired path and the position and orientation of the end-effector of the mechanism. This type of problem

typically involves solving the kinematics of a redundant mechanism, applying collision avoidance, position constraints, and speed constraints. To perform trajectory tracking, it is necessary to use a control technique which feeds back the actuators according to the tracking error. The control law is a function of the tracking error and a set of parameters named control gains. A classical control technique in robotics is the proportional-integral-derivative (PID)[23,24] controller and its variants as only proportional (P) or proportional-derivative (PD) controllers.[25] Therefore, the geometric parameters, as well as the control gains of the mechanisms are optimizable variables, a simpler case is to consider constant geometric parameters.[24,26]

Finding adequate control gains to achieve an optimum performance is not straightforward, thus it is important to use computational optimization tools to determine the best values under certain criteria. In a related work, the optimum design of the control parameters of a serial manipulator is approached.[27] Only kinematics is considered in a task-based problem. A memetic algorithm which uses a tunneling algorithm for local search is used to approximate the solution of this problem. A similar scheme has been proposed in other work, where simultaneous design optimization of a planar manipulator and a nonlinear gain for a PD controller is presented.[24] A hybrid evolutionary algorithm (GAES), which is formulated by combining the concepts from two well-known members of evolutionary algorithms family, GA and evolution strategies (ES), is applied as the optimization method. The same procedure had been used to optimize a redundant 7-DOF spatial manipulator.[23] A task-priority redundancy resolution technique is developed and sequential quadratic programming is applied to optimize the control parameters of the mechanism. It is worth noting that few researches have addressed concurrent optimization for both geometric parameters and control gains to track a desired trajectory. In addition, to the best of our knowledge, this kind of design problem has just been formulated for serial mechanisms or simpler than the one studied herein.

In this paper, two different mathematical models for the optimal design of parallel mechanisms are presented, each one of the models results in an optimization problem. These models are presented in two categories: the first one is a static design problem that consists in the maximization of a regular workspace, and the second one is a kinematic design problem that combines the concurrent optimization of both: structure geometry and control parameters for a task-based problem.

We approach the solution of such problems using different evolutionary algorithms in the state-of-the-art. The combination of the models for optimal design with evolutionary algorithms delivers a set of methods for automated optimal design of mechanisms. In this approach, our case study (the Delta parallel manipulator) is considered a kinematically complex mechanism. We consider that a mechanism is kinematically complex when due to a large number of DOF, redundancy or multiple kinematic chains, its kinematic model is preferably solved in a numeric fashion rather than in a closed form.

The organization of this article is as follows: Section 2 is devoted to present the kinematic analysis of a Delta robot. Section 3 describes the proposed evolutionary design methodology. In Section 4, we introduce the two objective models for the design of the Delta parallel manipulator along with the problem formulation of applying the different evolutionary algorithms. Results, a comparative analysis and discussion are presented in Section 5. Finally, concluding remarks are presented in Section 6.

2. The Delta parallel manipulator

2.1. Kinematic structure of a Delta robot

The Delta robot was introduced by Clavel[28] as a 3-*DOF* parallel manipulator, dedicated to high-speed pick and place applications. Fig. 1 shows a 3D structure of a parallel Delta manipulator. It consists of a base, a moving platform, and three identical kinematic chains that connect the base with the end-effector, each kinematic chain is driven by a revolute joint located at the base. The key design feature is the parallelogram link in each arm that allows maintaining the orientation of the end-effector.

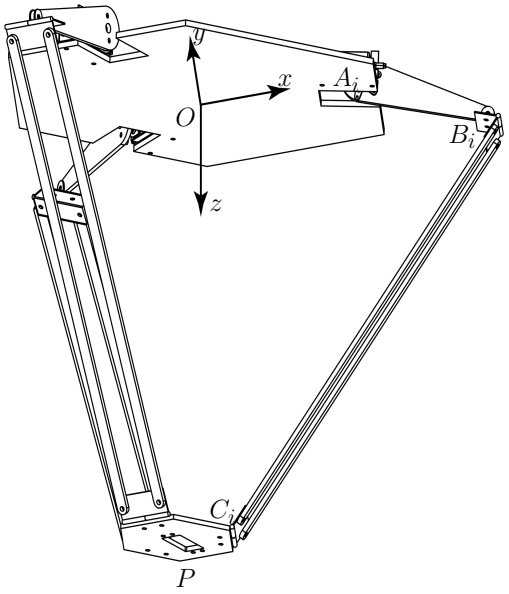

Fig. 1. 3D structure of a Delta parallel manipulator.[29]

The kinematic parameters are depicted in Fig. 2, where a denotes the length of arms $\overline{A_iB_i}$, b the length of the parallelogram $\overline{B_iC_i}$, $R = \overline{OA_i}$ and $r = \overline{PC_i}$, with O and P being the centers of the base and the moving platform, respectively. Three revolute actuated joints on the base at points A_i are arranged symmetrically at three vertices of an equilateral triangle. There are three revolute passive joints on the moving platform at points C_i.

In the sequel of this section, we present two modeling components that are needed to solve the addressed optimization problems for the Delta robot.

2.2. Inverse Kinematics

The coordinate-based framework of the manipulator angles of the $i-th$ *arm* is shown in Fig. 2, where $\theta_{1,i}$ is the angle between the x_i-axis to link $\overline{A_iB_i}$, $\theta_{2,i}$ is the complementary angle between $\overline{A_iB_i}$ and the projection of link $\overline{B_iC_i}$, $\theta_{3,i}$ is the angle between the y_i-axis and the link $\overline{B_iC_i}$, ϕ_i is the angle between the global x-axis and the x_i-axis, and $\mathbf{x} = (P_x, P_y, P_z)^T$ is the vector from the center of the fixed base (point O) to the center of the mobile base (point P) expressed in the global reference frame and represents the end-effector's position.

Let us move the origin reference framework to the $A_{i_{x_i,y_i,z_i}}$ framework, the closed loop chain for each *arm* is defined as:

$$\mathbf{p} - \mathbf{d} = \mathbf{a}_i + \mathbf{b}_i, \tag{1}$$

where $\mathbf{p} = R_z(\phi_i)\mathbf{x}$ with $R_z(\phi_i)$ a rotation matrix with respect to z-axis, $\mathbf{d}$ is the difference between the center of the mobile base and the fixed based relative to the reference framework, given as $\mathbf{d} = (R - r)[1, 0, 0]^T$, $\mathbf{a}_i$ is the distance from point A_i to B_i, relative to the reference framework, and finally, $\mathbf{b}_i$ is the distance from point B_i to C_i.

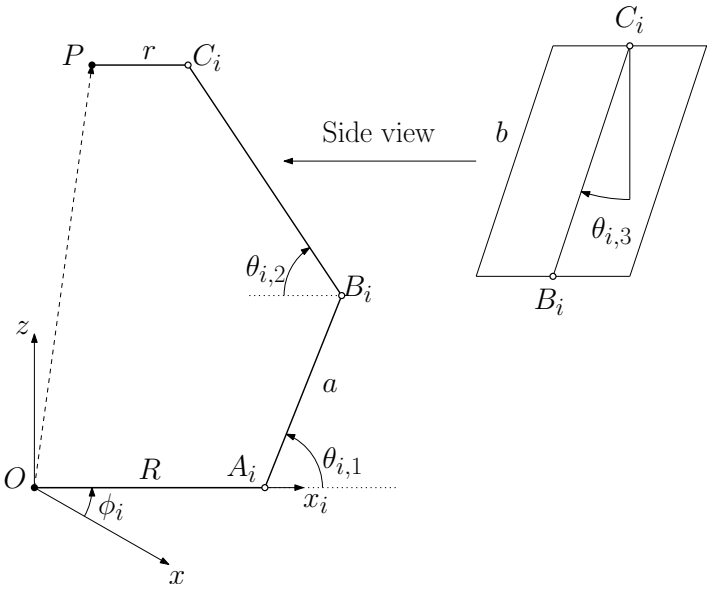

Fig. 2. Definition of variables on the $i-th$ *arm* over point A_i.

By manipulating Eq. (1), angles $\theta_{1,i}$, $\theta_{2,i}$, $\theta_{3,i}$ can be computed as in Eqs. (2-4):

$$\theta_{3,i} = \cos^{-1}\left(\frac{c_{y,i}}{b}\right) \tag{2}$$

$$\theta_{2,i} = \cos^{-1}\left(\frac{c_{x,i}^2 + c_{y,i}^2 + c_{z,i}^2 - a^2 - b^2\sin^2(\theta_{3,i})}{2ab\sin(\theta_{3,i})}\right) \tag{3}$$

$$\theta_{1,i} = \tan^{-1}\left(\frac{\frac{c_{z,i}}{c_{x,i}}\frac{g_1}{g_2} - 1}{\frac{g_1}{g_2} + \frac{c_{z,i}}{c_{x,i}}}\right) \tag{4}$$

where $c_{x,i} = P_x\cos(\phi_i) + P_y\sin(\phi_i) - (R-r)$, $c_{y,i} = -P_x\sin(\phi_i) + P_y\cos(\phi_i)$ and $c_{z,i} = P_z$, $g_1 = a - b\sin(\theta_{3,i})\cos(\theta_{2,i})$ and $g_2 = b\sin(\theta_{3,i})\sin(\theta_{2,i})$. Refer to Clavel[28] for a detailed procedure to obtain the inverse kinematics of the Delta manipulator.

2.3. Jacobian matrix

In this section, the relation between the angular joints velocities and the end-effector velocity of the Delta manipulator is presented. We are looking for a expression in the form:

$$\mathbf{J}_v\dot{\mathbf{x}} = \mathbf{J}_\theta\dot{\boldsymbol{\theta}}, \tag{5}$$

where $\mathbf{J}_v \in \mathbb{R}^{3\times3}$ and $\mathbf{J}_\theta \in \mathbb{R}^{3\times3}$ are the Cartesian and angular Jacobian, respectively. $\dot{\mathbf{x}} = (\dot{P}_x, \dot{P}_y, \dot{P}_z)^T$ and $\dot{\boldsymbol{\theta}} = (\dot{\theta}_{11}, \dot{\theta}_{12}, \dot{\theta}_{13})^T$ are the Cartesian velocities of the end-effector and angular velocities from the actuated joints, respectively.

Lopez et al.[30] introduced a closed-form equation for the required Jacobians in Eq. (5), as follows:

$$\mathbf{J}_v = \begin{pmatrix} j_{v_{1,x}} & j_{v_{1,y}} & j_{v_{1,z}} \\ j_{v_{2,x}} & j_{v_{2,y}} & j_{v_{2,z}} \\ j_{v_{3,x}} & j_{v_{3,y}} & j_{v_{3,z}} \end{pmatrix}; \quad \mathbf{J}_\theta = \mathrm{diag}\begin{pmatrix} j_{\theta_1} & j_{\theta_2} & j_{\theta_3} \end{pmatrix}, \tag{6}$$

where

$$
\begin{aligned}
j_{v_{i,x}} &= \sin(\theta_{3,i})\cos(\theta_{1,i}+\theta_{2,i})\cos(\phi_i) - \cos(\theta_{3,i})\sin(\phi_i),\\
j_{v_{i,y}} &= \sin(\theta_{3,i})\cos(\theta_{1,i}+\theta_{2,i})\sin(\phi_i) + \cos(\theta_{3,i})\cos(\phi_i),\\
j_{v_{i,z}} &= \sin(\theta_{3,i})\sin(\theta_{1,i}+\theta_{2,i}),\\
j_{\theta_i} &= -a\sin(\theta_{3,i})\sin(\theta_{2,i}).
\end{aligned}
$$

The relationship between the Cartesian reference frame and the reference frame of each joint is computed by $\dot{\mathbf{x}} = \mathbf{J}\dot{\boldsymbol{\theta}}$, where $\mathbf{J}$ is the manipulator Jacobian matrix. In this case, the Jacobian matrix is computed as $\mathbf{J} = \mathbf{J}_v^{-1}\mathbf{J}_\theta$.

3. Evolutionary design methodology

Evolutionary Algorithms (EAs) are based on the principles of biological evolution, a general framework can be stated as follows: first they generate a population of candidate solutions, then some solutions are selected, usually, via a stochastic method which favors the most promising ones. By using the selected set, a variation operator is applied to generate a new set of candidate solutions, these solutions replace the current population. Usually, the best solution is preserved trough generations and the process is repeated until a stopping criterion is met.

3.1. Evolutionary Algorithms

The EAs used here for solving the concurrent design problem can be classified into three families: Estimation of Distribution Algorithms (EDAs),[31] Evolution Strategies (ESs),[32] and Genetic Algorithms.[33] Although all of them can be classified as evolutionary algorithms, each of them has a particular way of working. The Omni-optimizer[34] is the implementation of a multi-objective GA, it re-combinates some of the most promising candidates selected using simulated binary crossover (SBX) and a binary tournament selection strategy. The exploration is maintained via the polynomial mutation operator. The Boltzmann Univariate Marginal Distribution Algorithm (BUMDA) estimates a probability distribution by using the best candidates, the better a candidate is, the higher the weight of such candidate in the estimation formula. Thus, the resulting probability functions favor to sample the best regions already known. The Covariance Matrix Adaptation Evolutionary Strategy (CMA-ES) uses a reproduction operator which favors promising directions. The Omni-optimizer and BUMDA share the characteristic of sampling the regions where the best candidates are, while the CMA-ES samples in *directions* where the best candidates were generated. A brief description of these algorithms is given in the rest of this section.

3.1.1. Omni-optimizer.[34] This optimizer is a general optimization GA, which is used, in this case, to solve a single objective problem. Nevertheless, it is an optimizer that can be applied to a wide range of problems from mono-objective to multiobjective with and without constraints in discrete and continuous domains. In our problem, we used it as a simple GA with simulated binary crossover (SBX)[35] and polynomial mutation. These are the same operators than those used in the Non-dominated Sorting Genetic Algorithm II (NSGA-II).[36]

The Omni-optimizer is presented in Algorithm 1, the workflow is as follows: the initial population is randomly generated, in step 1, using a uniform distribution, then, it is evaluated in step 2. In step 3, the population is sorted and selected via Pareto ranking. Then, binary tournament selection, recombination and mutation operators are applied to generate children, as notice in step 4, and evaluated in step 5. The algorithm initialization is different than the main loop because of elitism is introduced. Inside the main loop, in step 8, parents and children are combined together. In steps 9 and 10, the combined population is sorted according to non-domination, the new parent population is selected by means of Pareto ranking. If adding all solutions with certain rank produces

Algorithm 1 Omni-optimizer (NSGA-II) Pseudocode

1: $P_0 \leftarrow \mathcal{U}(\min^{(i)}, \max^{(i)})$
2: $\mathcal{F}_0 \leftarrow f(P_0)$
3: Sort P_0 based on the non-dominated solutions
4: $Q_0 \leftarrow$ tournament selection, recombination and mutation operators(P_0)
5: $\mathcal{G}_0 \leftarrow f(Q_0)$
6: $t \leftarrow 0$
7: **while** stopCrit $\neq$ **true do**
8: $R_t = P_t \cup Q_t$
9: Sort R_t based on the non-dominated solutions
10: Compute Pareto ranking and crowding distance
11: Select P_{t+1} using Pareto ranks and Crowding distances
12: $Q_{t+1} \leftarrow$ tournament selection, recombination and mutation operators(P_{t+1})
13: $\mathcal{G}_{t+1} \leftarrow f(Q_{t+1})$
14: **end while**
15: **return** $P_t^{(1)}$

a set greater than the population size, then these solutions are discriminated using the crowding distance in step 11. Then, children are generated as usual in step 12, and evaluated as shown in step 13.

3.1.2. Boltzmann Univariate Marginal Distribution Algorithm (BUMDA).[37] This algorithm is an EDA that uses a univariate Gaussian model to approximate a Boltzmann distribution,whose energy function is related to the objective function. This means that the better a candidate is, the probability to sample in the same region increases. The mean and variance parameters of the Gaussian model are derived from the analytical minimization of the Kullback-Leibler divergence. Pseudocode of the BUMDA is shown in Algorithm 2.

Algorithm 2 Boltzmann Univariate Marginal Distribution Algorithm (BUMDA) Pseudocode

Require: D=dimensions of the problem. n_{pop} =user given population size.

1: $X \leftarrow \mathcal{U}(\min^{(i)}, \max^{(i)})$
2: $\hat{\mathcal{F}} \leftarrow f(X)$
3: Sort X according to the objective function
4: $n \leftarrow n_{pop}$, $\theta_t \leftarrow \hat{\mathcal{F}}_n$, $t \leftarrow 0$
5: **while** stopCrit $\neq$ **true do**
6: $\mu_t^{(i)} \leftarrow \frac{\sum_{j=1}^{n} x_j^{(i)} \hat{\mathcal{F}}(x_j)}{\sum_{j=1}^{n} \hat{\mathcal{F}}(x_j)}$
7: $v_t^{(i)} \leftarrow \frac{\sum_{j=1}^{n} \hat{\mathcal{F}}(x_j)(x_j^{(i)} - \mu_i)^2}{1 + \sum_{j=1}^{n} \hat{\mathcal{F}}(x_j)}$
8: x_{best}=Elitism$(X, \hat{\mathcal{F}})$
9: $X \leftarrow (\mathcal{N}(\mu_t, v_t) \cup x_{best})$
10: $\hat{\mathcal{F}} \leftarrow f(X)$
11: Sort X according to the objective function
12: $n \leftarrow \min\left(\max(k|\theta_t < \mathcal{F}_k), \frac{1}{2} \cdot n_{pop}\right)$
13: $\theta_t \leftarrow \hat{\mathcal{F}}_n$
14: $t \leftarrow t + 1$
15: **end while**
16: **return** x_1

In steps 1-2, the initial population X is sampled from a uniform distribution and it is evaluated on the objective function f. As introduced by Valdez et al.,[37] BUMDA is a

maximization algorithm, and, in order to solve a minimization problem, the vector with the original objective function values $\mathcal{F}$ is transformed to $\hat{\mathcal{F}} = 1 - (\mathcal{F} - \mathcal{F}_{max})$, where $\mathcal{F}_{max}$ is the maximum of $\mathcal{F}$. In agreement with this transformation of the objective function, for any configuration, $\hat{\mathcal{F}}$ is always greater than 1, as a consequence, the 1, in the denominator in step 7 of Algorithm 2, does not significantly impact the variance value. In addition, when the algorithm is not near to convergence different configurations have a different weight in the parameter computation, while, near to convergence any solution in the selected set has, practically, the same weight in the parameter computation. Notice that when the algorithm converges the values of the optimization variables must be almost the same for the whole population, thus, the weights used for mean and variance computation do not significantly affect their values.

The population is sorted according to the objective function in step 3. Then, using the candidates with objective function value greater or equal to θ_t, mean and variance are computed for each dimension independently in steps 6-7. In step 8-10, the best candidate x_{best} is preserved through generations and the new population is sampled from a normal distribution, which uses the computed mean and variance. The threshold θ_t is augmented each iteration to ensure the improvement of the selected set. Steps 6-14 are repeated until a stopping criterion is met.

3.1.3. Covariance Matrix Adaptation Evolutionary Strategy (CMA-ES).[38] This strategy uses a multivariate Gaussian model. Parent solutions are used to determine the size, position and orientation of a Gaussian distribution used to sample the children candidate solutions. One of the most important characteristics of the CMA-ES is that the orientation of the multivariate Gaussian model directs the search to promising regions, in a kind of descent path. The Gaussian mean is computed as the addition of a weighted sum of the difference between the current population (parents) and the current mean.

Algorithm 3 Covariance Matrix Adaptation Evolutionary Strategy (CMA-ES) Pseudocode

1: Set $\lambda \leftarrow$number of samples per iteration, at least two, generally > 4
2: Initialize $m, \sigma, C = I, p_\sigma = 0, p_c = 0$
3: $t \leftarrow 0$
4: **while** stopCrit $\neq$ **true do**
5: **for** $i = 1 : \lambda$ **do**
6: $x_i \leftarrow \mathcal{N}(m, \sigma^2 C)$
7: $\mathcal{F}_i \leftarrow f(x_i)$
8: **end for**
9: $x_{1\ldots\lambda} \leftarrow x_{s(1)\ldots s(\lambda)}$ with $s(i) = argsort(f(1, \ldots, \lambda), i)$
10: $m' = m$
11: $m \leftarrow update\_m(x_1, \ldots, x_\lambda)$
12: $p_\sigma \leftarrow update\_ps(p_\sigma, \sigma^{-1} C^{-1/2}(m - m'), ||p_\sigma||)$
13: $p_c \leftarrow update\_pc(p_c, \sigma^{-1}(m - m'), ||p_\sigma||)$
14: $C \leftarrow update\_C(C, p_c, (x_1 - m')/\sigma, \ldots, (x_\lambda - m')/\sigma)$
15: $\sigma \leftarrow update\_sigma(\sigma, \ldots, ||p_\sigma||)$
16: **end while**
17: **return** x_1

In Algorithm 3, steps 1-3 initialize the multivariate Gaussian distribution and the algorithm parameters. In steps 5-7, the new candidate solutions are generated from the multivariate Gaussian distribution and evaluated on the objective function f. In step 9 the population is sorted selecting the best solutions. The previous mean is stored, then the new mean, the isotropic path p_σ and the anisotropic path p_c are computed in steps 10-13. Lastly, in steps 14-15 the new covariance matrix is computed and the isotropic and anisotropic paths are used to modify the new covariance matrix for guiding the search toward the maximum improvement directions as the maximum variance direction. Steps 5-15 are repeated until a stopping criterion is met.

3.2. Proposed evolutionary-algorithm based methodology

In this section, a general methodology for optimization of kinematically complex mechanisms is described. This methodology can be used for static and kinematic optimization problems. Indeed, the methodology can be applied for dynamic design problems where second order differential equations model the mechanism. However, such kind of optimization is left as future work in this paper. The case of kinematic optimization problems are treated as concurrent optimization problems: the optimization of geometry and control parameters simultaneously subject to a certain task.

Fig. 3 shows a graphical representation of the proposed methodology, which summarizes the process in the following main steps:

1. The mechanical model of the mechanism to be optimized must be selected together with a set of design parameters, e.g., masses, fixed lengths, inertial moments, desired workspace size and shape, etcetera, as well as a static or kinematic model.
2. At the same level, an optimization problem must be selected, such as: a) maximizing a regular workspace, or b) minimizing the integral of the absolute value of the control signal. For the first case, the output is a set of normalized lengths for the mechanism elements, which in turn can be scaled to any physical unit of length, that is to say, to meters, inches, feet or any arbitrary unit. For the second case the output is a set of lengths and control gains.
3. As a final option, an optimization algorithm must be selected, for instance one of the EAs of the previous sections, and the parameters of such algorithm must be inputted.
4. Given the mechanism, optimization problem and optimization algorithm, the overall optimization process is executed.

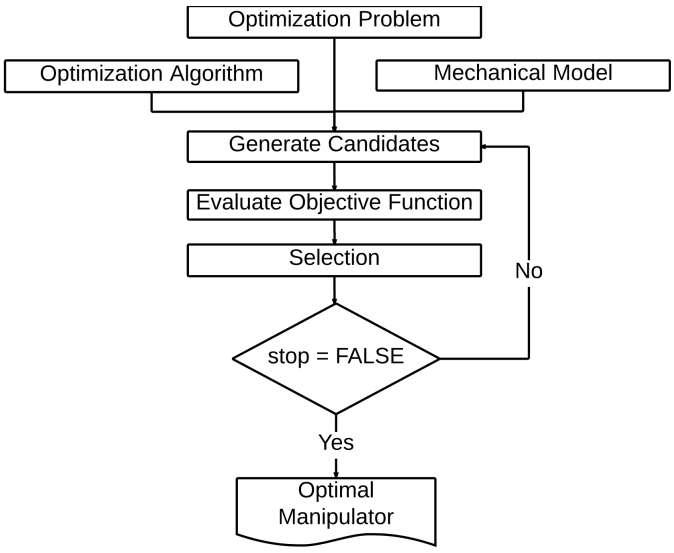

Fig. 3. Graphic representation of the concurrent optimization method for kinematically complex mechanisms.

As can be seen, this methodology could be used to optimize any mechanism under the requirement of providing an evaluator of a function which describes its quality.

4. Concurrent design of a Delta parallel manipulator

In order to compare the optimization methods described in Section 3, a formulation of the concurrent design for a Delta parallel manipulator is introduced. Let us define the general terms used in the optimization models. Considering a general mechanism with m joints, s actuators and δ degrees of freedom of the end-effector. Thus, we can define $\boldsymbol{\theta} \in \mathbb{R}^s$ as the actuated joint variables and $\boldsymbol{\phi} \in \mathbb{R}^{m-s}$ as the passive joint variables. The

mechanism is defined by a set of design parameters $\boldsymbol{\alpha} \in \mathbb{R}^p$, for instance, the lengths of the links, the relative position of each link, the relative arrangement between each axis, the size and shape of the end-effector, etcetera. Aditionally, let us define a target workspace $\mathbf{X} \in \mathbb{R}^{\delta \times n}$ as a set of Cartesian coordinates of position and orientation in a fixed reference frame, where n is the number of points. Notice that these points can represent a whole target workspace or a path which is a subset of the whole workspace.

In this paper, we consider two models for optimum design of a Delta manipulator, a) the maximization of a regular workspace subject to a constant norm of the links lengths: the goal is to cover the maximum volume of a regular workspace, usually set as a cube, sphere or cylinder. b) The minimization of the trajectory tracking error for concurrent optimal design: given a path as a set of points, each point is associated with a time coordinate to form a trajectory and the problem is to find the set of link lengths and control parameters which provide the minimum integral of the absolute value of the control signal. These models are expressed using the following objective functions.

4.1. Objective 1: Optimal mechanism design for the maximization of a regular workspace

This problem consists in finding the maximum regular workspace for a mechanism whose sum of the links lengths is the unity.[18] The problem is subject to constraints that guarantee an adequate performance, such as manipulability constraints, which aim for keeping the robot away from singular configurations. A dexterity measure used in this work, which is often referred as an intuitive quantitative measure, is the inverse of the condition number of the Jacobian matrix $\kappa(\mathbf{J})$, defined as $\kappa(\mathbf{J}) = \sigma_{\min}(\mathbf{J})/\sigma_{\max}(\mathbf{J})$,[39] where $\sigma_{\min}(\mathbf{J})$ and $\sigma_{\max}(\mathbf{J})$ are the minimum and maximum singular values of the Jacobian matrix, respectively. Notice that, $\kappa \in [0, 1]$. From now on, we refer as manipulability to this dexterity measure introduced previously. In order to decouple translational and rotational manipulability of the end-effector to guarantee position and orientation manipulability, independent constraints are imposed for these measures. In general, the Jacobian matrix $\mathbf{J}$ contains the terms of translational and rotational motion, and can be written in two separated sub-matrices as follows:

$$\dot{\mathbf{x}} = \begin{pmatrix} \dot{\mathbf{x}}_t \\ \dot{\mathbf{x}}_r \end{pmatrix} = \begin{pmatrix} \mathbf{J}_p \\ \mathbf{J}_r \end{pmatrix} \dot{\boldsymbol{\theta}}, \tag{7}$$

where $\dot{\mathbf{x}}_t$ and $\dot{\mathbf{x}}_r$ are translational and rotational velocities of the end-effector, respectively. Thus, we can compute separately manipulability measures for position $\kappa(\mathbf{J}_p)$ and orientation $\kappa(\mathbf{J}_r)$. The reader must consider that computing a dexterity measure based on singular values of independent translational and rotational Jacobians is valid if the manipulator performs only one type of motion (pure translation or rotation), if the manipulator performs a complex motion involving displacement and rotation, a different dexterity measure could be more suitable.[40,41] The Delta manipulator is considered a pure translational mechanism.

Let us define a measure $\Phi(\boldsymbol{\alpha})$ for the volume of the workspace W. The measure $\Phi(\boldsymbol{\alpha})$ is expressed in terms of a parameter of the regular workspace, for this case, the side length of a cube. Considering all the statements above, the optimization problem is defined as follows:

$$\max_{\boldsymbol{\alpha}} \Phi(\boldsymbol{\alpha}), \tag{8}$$

subject to

$$\begin{aligned} \kappa(\mathbf{J}_p(\mathbf{X},\theta,\boldsymbol{\alpha})) &\geq \gamma_1, \\ \kappa(\mathbf{J}_r(\mathbf{X},\theta,\boldsymbol{\alpha})) &\geq \gamma_2, \\ \theta_i^{\min} \leq \theta_i &\leq \theta_i^{\max}, \\ \phi_j^{\min} \leq \phi_j &\leq \phi_j^{\max}, \\ \sum_{k=1}^{q} l_k(\boldsymbol{\alpha}) &= \tau, \end{aligned} \tag{9}$$

where $i = 1, \ldots, s$, $j = 1, \ldots, m-s$, γ_1 and γ_2 are position and orientation manipulability bounds, respectively. $l_k(\boldsymbol{\alpha})$ is the lenght of the link k and τ is a given normalizing constant, set to $\tau = 1$ in this work. The last constraint removes the dimension effects by normalizing the design parameters $\boldsymbol{\alpha}$ of the manipulator. The joint limits are specified for actuated and passive joint variables. Additionally, we assume that joint limits are set adequately to avoid self-collisions. In the case study presented in this work, the Jacobian of the Delta parallel manipulator is only translational, but the formulation is presented in general for any mechanism. Thus, in our case study there is not a rotational Jacobian matrix.

4.1.1. Evaluation of a candidate solution. We define a cubic shape for a regular workspace W. The length of an edge of the cubic workspace is denoted by $2l$ and it is used to compute the objective function in Eq. (8). The center of the resultant maximal effective regular workspace is unknown for the evaluation of each candidate solution, but it is known that the manipulator is symmetrical with respect to the z-axis. In consequence, the coordinates of the center of the workspace have the form $(0, 0, z_c)$. In our work z_c is found by an intensive search, placing the reference frame at the origin frame as presented in Fig. 2, the z-axis is discretized from 0 to $(a + b)$ in partitions of size $1.0e$-5.

For this case study, we evaluate a candidate solution by finding the maximum cube for each z_c in the discretized segment of the z-axis, the maximum cube is found as follows:

1. Initialize $l_t = 0.0$ and $\Delta_s = 1.25(a + b)$.
2. Generate 27 equidistant points in a cube of length $l_t + \Delta_s$ centered at $(0, 0, z_c)$, in such a way that a point has the coordinate (x_i, y_i, z_c).The constraints in Eq. (10) are verified to be fulfilled for each point
3. If any point does not fulfills all the constraints, Δ_s is decreased by a factor of 0.5, otherwise the length of the side of the cube l_t is updated as $l_{t+1} = l_t + \Delta_s$.
4. The loop is repeated from step 2 until Δ_s is lower than a tolerance of $1e$-8.

4.2. Objective 2: Concurrent optimal design of kinematic control

In this problem, we consider an actuated mechanism under a proportional-derivative control action. Given a position and orientation function dependent on time, we aim to minimize the error between the desired trajectory and the actual trajectory followed by the end-effector of the mechanism. The error is a time-dependent function. Hence, we propose to model the optimization problem as the integral of the absolute value of the control signal, as follows:

$$\min_{\boldsymbol{\alpha}} \mathcal{F}(\boldsymbol{\alpha}) = k_p(\boldsymbol{\alpha}) \int_{t_0}^{t_n} ||\mathbf{e}(\boldsymbol{\alpha}, t)|| \partial t + k_d(\boldsymbol{\alpha}) \int_{t_0}^{t_n} ||\dot{\mathbf{e}}(\boldsymbol{\alpha}, t)|| \partial t, \tag{10}$$

subject to

$$\begin{aligned} \theta_i^{\min} \leq \theta_i &\leq \theta_i^{\max}, \\ \phi_j^{\min} \leq \phi_j &\leq \phi_j^{\max}, \end{aligned}$$

where $i = 1, \ldots, s,\ j = 1, \ldots, m - s,\ k_p(\boldsymbol{\alpha})$ and $k_d(\boldsymbol{\alpha})$ are proportional and derivative control gains, respectively, which are included as optimization variables in $\boldsymbol{\alpha}$ besides of the link lenghts. The functions $\mathbf{e}(\boldsymbol{\alpha}, t) = \mathbf{X}(\boldsymbol{\alpha}, t) - \mathbf{X}^D(t) \in \mathbb{R}^\delta$ and $\dot{\mathbf{e}}(\boldsymbol{\alpha}, t) = \dot{\mathbf{X}}(\boldsymbol{\alpha}, t) - \dot{\mathbf{X}}^D(t) \in \mathbb{R}^\delta$ are translation and velocity errors, respectively. $\mathbf{X}(\boldsymbol{\alpha}, t)$ and $\dot{\mathbf{X}}(\boldsymbol{\alpha}, t)$ are the translation and velocity coordinates (including orientation in both cases) of the manipulator's end-effector at an instant t. Likewise, $\mathbf{X}^D(t)$ and $\dot{\mathbf{X}}^D(t)$ are the desired translation and velocity coordinates imposed to the manipulator by the desired trajectory. In the same way that in the previous case study, the joint limits are specified for actuated and passive joint variables in such a way that self-collisions of the links are avoided.

The physical meaning of Eq. (10) is the following: the control signal is, basically, the energy required to perform the trajectory tracking, whose accuracy depends on the control gains k_p and k_d. Notice that we need to use the norm of the position and velocity errors to avoid cancellation in the integral. Moreover, the control signal includes in a single term the position and velocity error as well as the energy. Hence, this objective function intends to complete the task with the minimum energy.

The kinematic proportional-derivative control that allows tracking the desired trajectory is implemented as follows:

$$\dot{\boldsymbol{\theta}}_{t+1} = \mathbf{J}_t^{-1}(\dot{\mathbf{X}}^D(t) - k_P\mathbf{e}_t - k_D\dot{\mathbf{e}}_t), \tag{11}$$

The values of the joints variables in the next instant of time are obtained using the Euler method, so that $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \Delta t \cdot \dot{\boldsymbol{\theta}}_{t+1}$. The translation and orientation variables in the next instant of time can be computed by using the Cartesian velocity in the current instant of time as follows: $\mathbf{X}_{t+1} = \mathbf{X}_t + \Delta t\ \dot{\mathbf{X}}_{t+1}$, where $\dot{\mathbf{X}}_{t+1} = \mathbf{J}_t\dot{\boldsymbol{\theta}}_{t+1}$.

In order to find an optimal configuration, if any constraint is unfulfilled or if the angular positions are indeterminate, the simulation is stopped at time t, and the configuration is penalized by adding β, where $\beta = 1e{+}3 \cdot (t_n - t)$;

4.3. Algorithms setting and termination criteria

The stopping criteria are the following: the CMA-ES stops if the best objective function value is less than $1.0e$-9, or the difference between two consecutive generations is less than $1.0e$-20. The Omni-optimizer stops when it reaches 40 generations. The BUMDA stops if the maximum variance is less than $1.0e$-12. Additionally, all algorithms stop at a maximum of $1e4{\cdot}p$ evaluations, for p optimization variables. In order to conclude which algorithm delivers the best performance value for each design problem, we have used a non-parametric intensive sampling technique for statistical hypothesis test, the boostrap method.[42]

5. Results and comparative analysis

5.1. Results for the maximum regular workspace.

Considering the first problem addressed for the Delta robot, we aimed to find the maximum regular workspace within the total workspace of the mechanism. The set of optimization variables are the links lengths denoted as $\boldsymbol{\alpha} = [a\ b\ d]^T$, where $d = R - r$. The search range of the lengths is $[0, 1]$ for all of them. Joint limits and manipulability constraints are taken as in Lou et al.[18] and the inverse kinematics for the Delta robot is computed as described in Section 2.2. Fig. 4 (a) presents the optimal approximation of the whole workspace, as well as the enclosed regular workspace. Figs. 4 (b), (c) and (d) show a cross section of the workspace at heights of 0.4137, 0.5940 and 0.7742 units, respectively.

Table I shows the best solution found by each algorithm for the lengths $\{a, b, d\}$ and the corresponding objective function as well as the mean and standard deviation of ten

Table I . Best solution for Eq. (8), links lengths $\{a, b, d\}$ and objective value $\mathcal{F}(\boldsymbol{\alpha})$.

EA	a	b	d	$\mathcal{F}(\boldsymbol{\alpha})$
CMA-ES	0.40119	0.56786	0.03039	0.1807
OMNI	0.36434	0.56111	0.02573	0.1685
BUMDA	0.38189	0.57725	0.03678	0.1752

independent executions. As can be notice, the best solution is found by the CMA-ES, nevertheless, the lengths are similar for all the algorithms.

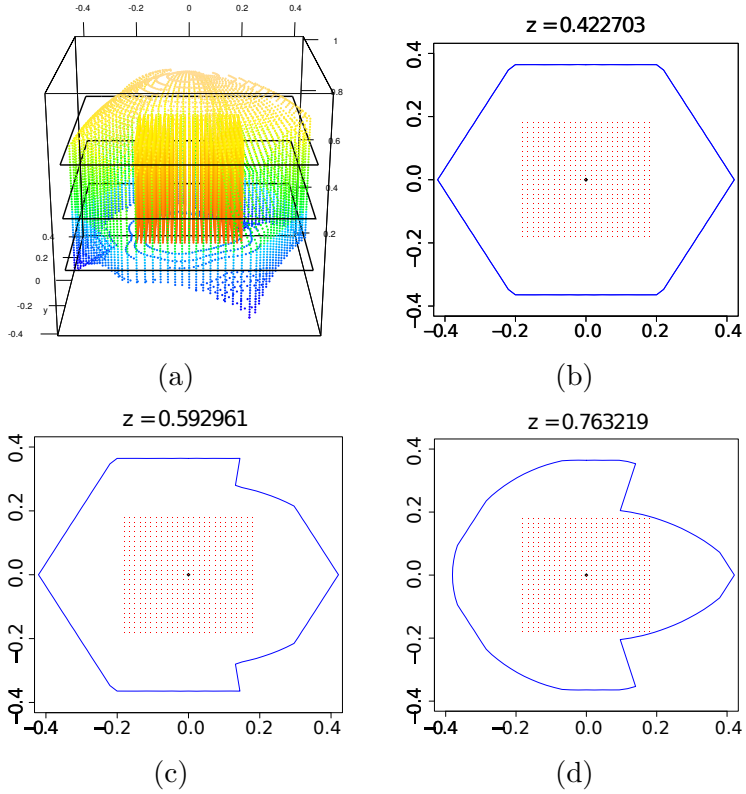

Fig. 4. Workspace visualization. (a) 3D total and regular workspace. (b) Cross-section at 0.422703. (c) Cross-section at 0.592961. (d) Cross-section at 0.763219.

5.2. Results for the concurrent optimization of the geometry and control parameters. In the second problem addressed for the Delta manipulator, we aimed to find the geometric and control parameters to minimize the tracking error of a desired trajectory. In this case, we use 100 equidistant points in time, sampled from the function $\mathbf{X}_i(t) = \langle c_2 \sin(2.0\pi \cdot c_1), c_2 \cos(\pi \cdot c_1), c_3(c_1^4) + 0.2 \rangle$, where $c_1 = \frac{2.0 \cdot t - t_f}{t_f}$, $c_2 = 0.3$, $c_3 = 0.5$, $0.0 \leq t \leq 30.0$ s and $i = 1, \ldots, 100$. The search range of the links lengths is $[0, 1]$ for lengths a, b and d, and the range for the control parameters is $[0, 100]$ for gains k_p and k_d. In order to justify the concurrent optimization model, we present a comparison between only control optimization using a fixed geometric structure versus concurrent geometry and control optimization. Table II shows results obtained by EAs for proportional (P) control and proportional-derivative control (PD). The last column presents the objective function value of the solution when optimizing only control gains, while the previous column to the last one shows the objective function value obtained by the concurrent optimization methodology. As can be seen, the signal value of the concurrent optimization is lower than the control gain optimization, meaning that the energy applied to the concurrently designed mechanism is lower than that applied to the fixed-

lengths mechanism. An additional advantage of the concurrent optimization is that the performance of a mechanism is subject to track a trajectory, thus, when considering fixed lengths, flexibility and manipulability of the manipulator are reduced. Concurrent optimization allows to find a suitable set of link lengths which maximizes flexibility and manipulability of the manipulator for each desired task.

Table II . Best solution for Eq. (10) for concurrent optimization, links lengths $\{a, b, d\}$, control gains $\{k_p^{(c)}, k_d^{(c)}\}$ and objective value $\mathcal{F}^{(c)}(\boldsymbol{\alpha})$, and a control optimization, with fixed links lengths $\{0.4, 0.5, 0.1\}$, control gains $\{k_p^{(f)}, k_d^{(f)}\}$ and objective value $\mathcal{F}^{(f)}(\boldsymbol{\alpha})$.

EA	a	b	d	$k_p^{(c)}$	$k_p^{(f)}$	$k_d^{(c)}$	$k_d^{(f)}$	$\mathcal{F}^{(c)}(\boldsymbol{\alpha})$	$\mathcal{F}^{(f)}(\boldsymbol{\alpha})$
Proportional control									
CMA-ES	4.35e-1	4.90e-1	7.39e-2	1.00e2	8.49e-2	-	-	3.71e-1	6.66e-1
OMNI	3.36e-1	5.51e-1	9.07e-2	9.99e1	8.49e-2	-	-	3.71e-1	6.66e-1
BUMDA	4.72e-1	4.61e-1	7.41e-2	1.00e2	8.54e-2	-	-	3.71e-1	6.67e-1
Proportional-derivative control									
CMA-ES	4.38e-1	4.87e-1	7.26e-2	9.99e-1	9.87e-1	8.28e-1	8.33e-1	8.74e-3	5.28e-1
OMNI	4.25e-1	5.02e-1	7.61e-2	9.99e-1	9.87e-1	8.28e-1	8.33e-1	8.76e-3	5.28e-1
BUMDA	4.55e-1	4.77e-1	7.50e-2	9.99e-1	2.29	8.28e-1	6.62e-1	8.75e-3	5.80e-1

As for the concurrent optimization, notice that for both controls, P and PD, the best structure parameters are almost the same, meaning that those parameters could be one local or global optimum, and that they are robust due to they support small changes in the links lengths. Nevertheless, the proportional gain is larger for the P than for the PD controller, meaning that the P controller is using more energy in order to achieve a reasonable performance. This can be noticed in the objective function column. In addition, notice that all algorithms reach basically the same objective function value, but the CMA-ES consistently delivers a similar value for the P and PD control gains. Fig. 5 (a) and (c) shows the logarithmic absolute End-Effector Position Error (EEPE) over time for the P and PD control, respectively. Figs. 5 (b) and (d) show in green the path undertaken by the Delta robot and the desired path in blue, for the P and PD controls, respectively. Fig. 5 (e) describes the integration of the control signal over time, in order to compare energy consumption of the P and PD controllers. As shown, the P control reaches faster the target trajectory at a higher energy consumption.

5.3. Statistical comparison of optimization algorithms

With the aim of comparing the optimization methods, we use hypothesis tests to compare pairs of them via the bootstrap methodology,[42] it is presented in Table III. Each evolutionary algorithm is executed 10 times for each optimization problem under the same conditions. We test for each optimization variable and objective function, if the value delivered by the algorithms in the first column are different (less than or greater than) each other. For the optimization parameters, that is to say lengths and control gains, if there is not statistical evidence that they are different we write $=$, if the first is greater than the second we write $>$ and the *p-value* of the corresponding hypothesis test, similarly if the second is greater than the first we write $<$. For the objective function comparison, we show in bold the name of the algorithm which performs the best according to the hypothesis test, and the *p-value* of the hypothesis test in the corresponding column. If neither algorithm can be consider the best according the statistical evidence, hence both names are in normal font, and there is an asterisk in the value of the $\mathcal{F}$ column.

According to Table III, the methods find similar geometrical as well as control parameters, nevertheless, we can see that CMA-ES consistently reports the best objective function value. The hypothesis test also can be used to infer the relationship between the optimization parameters and the objective function, by instance, for the first problem the CMA-ES is the best and it seem that the only statistical significant difference is in the d parameter. For the concurrent optimization problem it seems that, in general, the

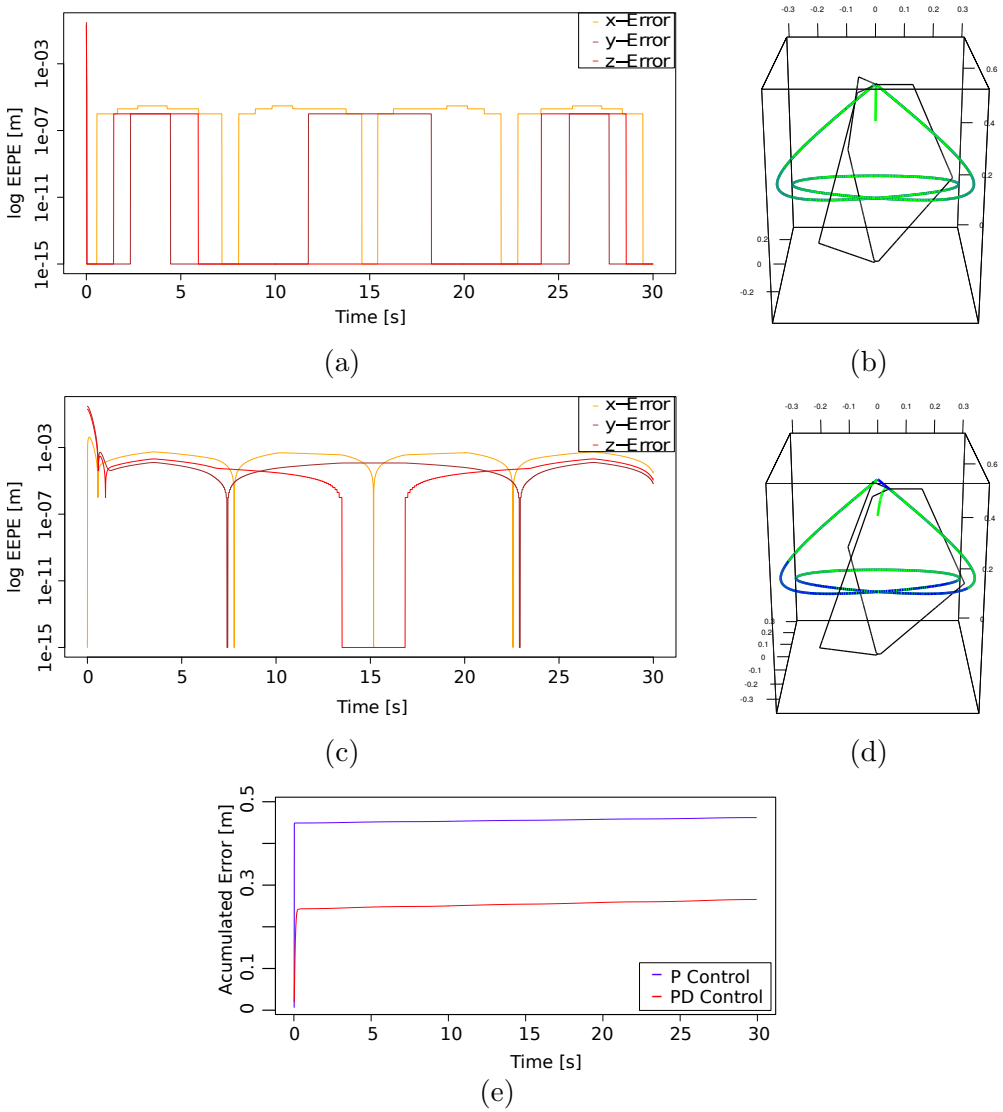

Fig. 5. Trajectory tracking control for the Delta robot. (a) Logarithmic absolute EEPE over time for P control. (b) Trajectory tracking for P. (c) Logarithmic absolute EEPE over time for PD control. (d) Trajectory tracking for PD. (e) Absolute EEPE integral for P and PD control.

difference is not significant for the parameters while the objective function actually is. A possible explanation is that there is not a high correlation between the objective function and the parameters even if they are dependent, in other words, it is not straightforward to statically predict the objective function by means of the parameters, even though the objective function is a function of the parameters.

5.4. Addressing the problem using a quasi-Newton method

In order to empirically demonstrate how difficult the problem is, we present a comparison with the BFGS[43] implemented in the `minfunc` function in Matlab, and the Nelder-Mead Simplex[44] in the `fminsearch` function.

The results are show in Table IV. The first method can not find a solution, the message from minfunc is: *stopped because it cannot decrease the objective function along the current search direction.* While the second successfully terminates after 142 iterations. The initial parameters are randomly drawn in a vicinity of the best known solution. The CMA-ES and actually any of the EAs deliver a greater workspace than that delivered by these methods.

Table III . Results of the hypothesis test ($\mu_A < \mu_B$) with a 5% significance for the optimization of a Delta parallel manipulator, with the objective functions described in Eq. (8) and Eq. (10).

EA_1 vs EA_2	a	b	d	k_p	k_d	$\mathcal{F}(\boldsymbol{\alpha})$
1.- Optimal maximum regular workspace mechanism design						
CMA-ES vs BUMDA	=	=	$<(0.02)$	-	-	$6.00e$-4
CMA-ES vs OMNI	=	=	$<(0.02)$	-	-	0.000
BUMDA vs OMNI	$>(0.032)$	=	=	-	-	$8.200e$-4
2.- Optimal concurrent mechanism design with a kinematic proportional control						
CMA-ES vs BUMDA	=	=	$<(0.016)$	=	-	0.000
CMA-ES vs OMNI	=	=	=	=	-	0.000
BUMDA vs OMNI	=	=	=	=	-	0.110 *
3.- Optimal concurrent mechanism design with a kinematic proportional-derivative control						
CMA-ES vs BUMDA	=	=	=	=	=	0.033
CMA-ES vs OMNI	=	=	=	$>(0.000)$	=	$2.100e$-4
BUMDA vs OMNI	=	=	=	$>(0.009)$	=	0.134 *

Table IV . Comparison of BFGS and simplex vs CMA-ES Eq. (8), links lengths $\{a, b, d\}$ and objective value $\mathcal{F}(\boldsymbol{\alpha})$.

EA	a	b	d
BFGS Initial parameters	0.4364588	0.5151318	0.0484094
BFGS Final parameters	0.4863362	0.5136638	0.0000
Nelder-Mead Initial parameters	0.4093702	0.5469331	0.0436966
Nelder-Mead Final parameters	0.4550914	0.5062968	0.0386118
Objective function BFGS	0.1297084		
Objective function simplex	0.1205316		
Objective function CMA-ES	**0.1807**		

The very same procedure is applied to the concurrent design with the P and PD controls. The results are shown in Table V, as can be seen, even if the initial values are not so far away from the solution, the BFGS is incapable of finding an adequate result, the simplex, finds a similar objective function value (P control) for the first case than the EAs, but for the second case it can not find a closed approximation to the EAs solution. One can infer from the second case (PD control) that both algorithm intend to minimize the function by minimizing the control gains, that is reasonable, because the objective function is linear and convex with respect to the control gains, but they, actually, can not optimize the lengths, because the function is not convex with respect to these parameters.

6. Conclusions

In this article we propose a categorization for the optimum design of robots as follows: static (model of zero order differential equations), kinematic (model of first order differential equations), dynamic (model of second order differential equations). Two instances of these problems are addressed, the first is a static problem for approximating the optimum lengths which maximize a regular workspace, while the second is a kinematic problem that concurrently approximates the optimum control and geometrical design of a Delta parallel manipulator using P and PD controllers. On the one hand, we have shown that these problems are not solvable using gradient base methods such as BFGS or greedy-like methods such as the Nelder-Mead simplex, hence, it is adequate to approximate the solutions using evolutionary algorithms. On the other hand, all EAs deliver similar optimum approximations in the parameters and objective function, note in Tables I and Table II, that the difference among objective functions and parameters

Table V . Comparison of BFGS and simplex vs CMA-ES Eq. (10), links lengths $\{a, b, d\}$, control gains $\{k_p, k_d\}$ and objective value $\mathcal{F}(\boldsymbol{\alpha})$.

EA	a	b	d	k_p	k_d
Proportional control					
BFGS Initial parameters	0.4329	0.4882	0.0279	98.9	
BFGS Final parameters	0.4329	0.4882	0.0279	100	
Nelder-Mead Initial parameters	0.4166	0.4449	0.0339	96.73	
Nelder-Mead Final parameters	0.4802	0.411	0.03166	100	
Objective function BFGS	18820.4				
Objective function simplex	0.371816				
Objective function CMA-ES	**0.371262**				
Proportional-derivative control					
BFGS Initial parameters	0.4149	0.4802	0.0547	2.584	5.579
BFGS Final parameters	0.6346	0.4802	0.2713	1.0	1.0
Nelder-Mead Initial parameters	0.4117	0.4609	0.0391	4.83	4.8
Nelder-Mead Final parameters	0.4351	0.4613	0.0543	1.0	1.0
Objective function BFGS	20598.5				
Objective function simplex	19769.6				
Objective function CMA-ES	**0.00874**				

are less than 1%, in spite of the fact that all of the compared EAs work different and belong to different families. Hence, we have shown that the problems can be addressed under a unified framework, although they are different in the goal, optimization variables and complexity.

The optimization methods are compared via non-parametric hypothesis tests. This allows us to determine which optimization method is most suitable for each optimization problem. According to the results, in most of the cases, the CMA-ES consistently delivers the best solution and in 90% of the cases it statistically outperforms the other algorithms. Thus, we suggest using CMA-ES as a basic optimization method to solve the optimization problems of kinematically complex mechanisms. Table III can be used to determine relations between optimization variables and the corresponding objective function, for example, for the maximum workspace, it could be inferred that a small change in d slightly but consistently cause a change in the objective function, an it is the cause of the difference of the CMA-ES with other algorithms. Considering that the values in the objective function and parameters delivered by the CMA-ES are quite similar to the ones delivered by the other algorithms, we can conclude that CMA-ES can refine better the solution, that is to say, it performs a better exploration in a closed vicinity at the end of the search process which consistently improves the objective function value. There are limitations on the CMA-ES, for instance, the estimation of the parameters uses more memory and computation than the other algorithms. Meanwhile, the BUMDA uses less memory and it is not as computationally expensive as the CMA-ES. Therefore, it can be executed in situ with low computational resources, in addition, it is shown the difference in the objective function is less than 1% in most of the cases.

Future work contemplates approaching other optimization problems, like dynamic design problems, including different mechanisms, requirements, and using more realistic simulations, which could consider stiffness as part of the optimization model.

Acknowledgments
The first and third authors were supported in part by Conacyt [grant 220796].

References

1. J.H. Park and H. Asada. Concurrent design optimization of mechanical structure and control for high speed robots. In *American Control Conference, 1993*, pages 2673–2679. IEEE, 1993.
2. G.H.T. Navajas, J.A.P. Raad, and S. R. Prada. Concurrent design optimization and control of a custom designed quadcopter. In *16th International Conference on Research and Education in Mechatronics*, pages 63–72. IEEE, 2015.
3. V.C. Moulianitis, A. I Synodinos, C.D. Valsamos, and N.A. Aspragathos. Task-based optimal design of metamorphic service manipulators. *Journal of Mechanisms and Robotics*, 8(6):061011, 2016.
4. A. Borboni, R. Bussola, R. Faglia, P.L. Magnani, and A. Menegolo. Movement optimization of a redundant serial robot for high-quality pipe cutting. *Journal of Mechanical Design*, 130(8):082301, 2008.
5. R. Vijaykumar, K.J. Waldron, and M.J. Tsai. Geometric optimization of serial chain manipulator structures for working volume and dexterity. *The International Journal of Robotics Research*, 5(2):91–103, 1986.
6. M.J.H. Lum, J. Rosen, M.N. Sinanan, and B. Hannaford. Optimization of a spherical mechanism for a minimally invasive surgical robot: theoretical and experimental approaches. *IEEE Transactions on Biomedical Engineering*, 53(7):1440–1445, 2006.
7. J.P. Merlet and D. Daney. Appropriate design of parallel manipulators. In *Smart Devices and machines for advanced manufacturing*, pages 1–25. Springer, 2008.
8. M. Ganesh, B. Bihari, V. Rathore, D. Kumar, C. Kumar, A.Sree, K. Sowmya, and A. Dash. Determination of the closed-form workspace area expression and dimensional optimization of planar parallel manipulators. *Robotica*, pages 1–20, 2016.
9. J. Angeles and C. Gosselin. The optimum kinematic design of a planar three-degree-of-freedom parallel manipulator. *ASME Journal of Mechanisms, Transactions and Automation in Design*, 110:35–41, 1988.
10. E. Ottaviano and M. Ceccarelli. Optimal design of CaPaMan (Cassino Parallel Manipulator) with a specified orientation workspace. *Robotica*, 20(02):159–166, 2002.
11. D. Zhang, Z. Xu, C. Mechefske, and F. Xi. Optimum design of parallel kinematic toolheads with genetic algorithms. *Robotica*, 22(1):77–84, 2004.
12. J.A. Carretero, R.P. Podhorodeski, M.A. Nahon, and C.M. Gosselin. Kinematic analysis and optimization of a new three degree-of-freedom spatial parallel manipulator. *ASME. Journal of Mechanical Design*, 122(1):17–24, 1999.
13. K.H. Pittens and R.P. Podhorodeski. A family of stewart platforms with optimal dexterity. *Journal of Field Robotics*, 10(4):463–479, 1993.
14. Y. Lou, G. Liu, and Z. Li. Randomized optimal design of parallel manipulators. *IEEE transactions on automation science and engineering*, 5(2):223–233, 2008.
15. M. Armada, A. Sanfeliu, and M. Ferre. Dexterity optimization of a three degrees of freedom delta parallel manipulator. *Advances in Intelligent Systems and Computing*, 253:719–726, 2014.
16. E. Courteille, D. Deblaise, and P. Maurine. Design optimization of a delta-like parallel robot through global stiffness performance evaluation. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pages 5159–5166. IEEE, 2009.
17. K. Miller. Optimal design and modeling of spatial parallel manipulators. *The International Journal of Robotics Research*, 23(2):127–140, 2004.
18. Y. Lou, Y. Zhang, R. Huang, X. Chen, and Z. Li. Optimization algorithms for kinematically optimal design of parallel manipulators. *IEEE Transactions on Automation Science and Engineering*, 11(2):574–584, 2014.
19. C. Riaño, C. Peña, and A. Pardo. Approach in the optimal development of parallel robot for educational applications. In *Proceedings of the WSEAS international conference on Recent Advances in Intelligent Control, Modelling and Simulation (ICMS)*, volume 145, 2014.
20. X.J. Liu and J. Wang. A new methodology for optimal kinematic design of parallel mechanisms. *Mechanism and Machine Theory*, 42(9):1210–1224, 2007.
21. M.A. Laribi, L. Romdhane, and S. Zeghloul. *Advanced Synthesis of the DELTA Parallel Robot for a Specified Workspace*, pages 207–224. INTECH, Rijeka, 2008.
22. Y. Lou, G. Liu, J. Xu, and Z. Li. A general approach for optimal kinematic design of parallel manipulators. In *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, volume 4, pages 3659–3664. IEEE, 2004.
23. S. Chiaverini. Singularity-robust task-priority redundancy resolution for real-time kinematic control of robot manipulators. *IEEE Transactions on Robotics and Automation*, 13(3):398–410, 1997.
24. T. Ravichandran, G.R. Heppler, and D.W.L. Wang. Task-based optimal manipulator/controller design using evolutionary algorithms. Technical report, University of Waterloo, 2004.
25. G. Reynoso-Meza, J. Sanchis, x. Blasco, and M. Martínez. Algoritmos evolutivos y su empleo en el ajuste de controladores del tipo pid: Estado actual y perspectivas. *Revista Iberoamericana de Automática e Informática Industrial RIAI*, 10(3):251–268, 2013.
26. Y. Xia and J. Wang. A dual neural network for kinematic control of redundant robot manipulators. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 31(1):147–154, 2001.

27. R.R. dos Santos, V. Steffen, and S.F. Saramago. Optimal task placement of a serial robot manipulator for manipulability and mechanical power optimization. *Intelligent Information Management*, 2(9):512–525, 2010.
28. R. Clavel. Delta, a fast robot with parallel geometry. In *Proc. 18th Int. Symp. on Industrial Robots, Lausanne, 1988*, pages 91–100, 1988.
29. S. Mohamed. Delta robot. `https://grabcad.com/library/delta-robot--2`, 2012. [Online; upload 2012-12-04].
30. M. López, E. Castillo, G. García, and A. Bashir. Delta robot: inverse, direct, and intermediate jacobians. *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, 220(1):103–109, 2006.
31. P. Larranaga and J.A. Lozano. *Estimation of distribution algorithms: A new tool for evolutionary computation*, volume 2. Springer Science & Business Media, 2002.
32. H.P.P. Schwefel. *Evolution and optimum seeking: the sixth generation*. John Wiley & Sons, Inc., 1993.
33. L. Davis. *Handbook of genetic algorithms*. R. Van Nostrand, 1991.
34. A. Klanac and J. Jelovica. A concept of omni-optimization for ship structural design. *Advancements in Marine Structures, Proceedings of MARSTRUCT*, pages 473–481, 2007.
35. K. Deb. *Multi-objective optimization using evolutionary algorithms*, volume 16. John Wiley & Sons, 2001.
36. N. Srinivas and K. Deb. Muiltiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary computation*, 2(3):221–248, 1994.
37. S.I. Valdez, A. Hernández, and S. Botello. A Boltzmann based estimation of distribution algorithm. *Information Sciences*, 236:126–137, 2013.
38. N. Hansen, A. Niederberger, L. Guzzella, and P. Koumoutsakos. A method for handling uncertainty in evolutionary optimization with an application to feedback control of combustion. *IEEE Transactions on Evolutionary Computation*, 13(1):180–197, 2009.
39. C.A. Klein and B.E. Blaho. Dexterity measures for the design and control of kinematically redundant manipulators. *The International Journal of Robotics Research*, 6(2):72–83, 1987.
40. G. Pond and J.A. Carretero. Formulating jacobian matrices for the dexterity analysis of parallel manipulators. *Mechanism and Machine Theory*, 41(12):1505–1519, 2006.
41. C.M. Gosselin. Dexterity indices for planar and spatial robotic manipulators. In *Robotics and Automation, 1990. Proceedings., 1990 IEEE International Conference on*, pages 650–655. IEEE, 1990.
42. B. Efron and R.J. Tibshirani. *An introduction to the bootstrap*. CRC press, 1994.
43. J. Nocedal and S. Wright. *Numerical optimization*. Springer Science & Business Media, 2006.
44. J. Nelder and R. Mead. A simplex method for function minimization. *The computer journal*, 7(4):308–313, 1965.