Towards a Generic Optimal Co-Design of Hardware Architecture and Control Configuration for Interacting Subsystems

Michiel Haemers[a,b,c,*], Stijn Derammelaere[a,b,c], Albert Rosich[c], Clara M. Ionescu[a,c,d], Kurt Stockman[a,c]

[a] *Ghent University, Department of Electrical Energy, Metals, Mechanical Constructions and Systems, Belgium*
[b] *University of Antwerp, Department of Electromechanics, CoSys-Lab, Belgium*
[c] *Flanders Make, The Strategic Research Centre for the Manufacturing Industry, Belgium*
[d] *Technical University of Cluj Napoca, Department of Automatic Control, Romania*

Abstract

In plants consisting of multiple interacting subsystems, the decision on how to optimally select and place actuators and sensors and the accompanying question on how to control the overall plant is a challenging task. Since there is no theoretical framework describing the impact of sensor and actuator placement on performance, an optimization method exploring the possible configurations is introduced in this paper to find a trade-off between implementation cost and achievable performance.

Moreover, a novel model-based procedure is presented to simultaneously co-design the optimal number, type and location of actuators and sensors and to determine the corresponding optimal control architecture and accompanying control parameters. This paper adds the optimization of the control architecture to the current state-of-the-art. As an optimization output, a Pareto front is presented, providing insights on the optimal total plant performance related to the hardware and control design implementation cost.

The proposed algorithm is not focused on one particular application or a specific optimization problem, but is instead a generally applicable method and can be applied to a wide range of applications (e.g., mechatronic, electrical, thermal). In this paper, the co-design approach is validated on a mechanical setup.

Keywords: Multiobjective optimizations, Control system design, Genetic Algorithms, Systems design, Global optimization

*Corresponding author
Email address: Michiel.Haemers@UGent.be (Michiel Haemers)

1. Introduction

1.1. Different co-design components

Many industrial plants can be represented by a set of subsystems, connected through physical interactions. The overall design of these systems involves two main challenges, hence the term 'combined design' or 'co-design'. This involves the selection of the optimal hardware architecture (being sensor & actuator selection & placement) and the optimal control configuration (being control architecture and controller tuning parameters), as can be seen in Figure 1. This terminology is widely used in literature ([1, 2, 3, 4, 5]) and is consistently used throughout this paper.

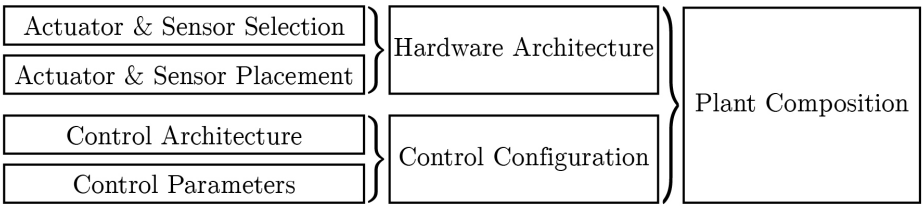

Figure 1: Overview of the nomenclature used for the different parts of the total plant composition optimization.

1.2. Need for simultaneous co-design

Traditionally, the hardware architecture and control configuration designs are treated sequentially and are therefore entirely separated from each other [6]. First, mechanical engineers design a physical setup, corresponding with the structural objectives on, for instance, weight, inertia or strength. Next, the position of the actuators and sensors in the system is determined, and these actuators and sensors are dimensioned. Subsequently, control engineers design a control system for the fixed physical structure based on the given inputs and outputs of actuators and sensors, satisfying a different set of objectives, such as settling time, reference tracking properties or robustness to disturbances. This sequential design approach is intuitive to implement but suffers from several problems when applied to more sophisticated applications.

First, there is usually a tight interdependency between the control configuration and sensor & actuator selection & placement parameters. Moreover, the hardware selection & placement can limit the design space of the controller and hence the optimal achievable control performance. In many cases, it is hard to predict the distinct impact of these design properties on the overall plant performance. It is, for example, often unclear what the effect is on the overall plant performance when a specific actuator or sensor is left out or sized differently on a subsystem level. Therefore, the hardware engineers have to make assumptions regarding the control configuration, after which the control engineers have to

stick to the hardware architecture created by the hardware engineers. This sequential approach might not lead to the most efficient or optimal design. Since modern plants are becoming increasingly more complex and the resources are always constrained by cost and space, a resource-efficient design has become an increasingly important issue [7].

Second, in the case of sequential design, the incorporation of the needs of both mechanical and control engineers becomes even more complicated as the size of the system gets larger. This results in sub-optimal designs and considerable integration, test and validation efforts. Therefore, there is a rising demand for systematic and comprehensive co-design methods to accomplish both the optimal hardware architecture and control configuration [7]. This means that a generically applicable procedure has to be developed in which both the placement and selection of the actuators and sensors have to be carried out, as well as the simultaneous determination of the control architecture and associated control parameters. This will enable the end user to gain a better understanding of the inevitable trade-off between cost and performance.

1.3. Multi-objective optimization

It is clear that a simultaneous co-design of both hardware architecture and control configuration can result in a more efficient design process, taking into account multiple (conflicting) objectives. These objectives are always related to a trade-off between cost and performance. To a limited degree, efforts have been made to implement a multi-objective co-design of control configuration and hardware architecture in electromechanical systems, but in these cases, it is always assumed that the control architecture is fixed. For example, an optimization problem with a small number of design variables describing the suspension geometry of a hard disc and its corresponding control parameters was formulated in [8]. The result is an optimization of the design variables of a hard disc suspension geometry and the controller feedback gains through the use of the Linear Quadratic Regulator (LQR) method. Similarly, the co-design of control parameters and the geometrical properties of machine tools (using a multilevel decomposition), parallel robots (using a differential evolution algorithm) and a four-bar system (using a Genetic Algorithm) was detailed in [6], [9] and [10], respectively. A design method for the sequential optimization of the mechanical structure and the control parameters for a two-link high-speed robot is developed in [11]. In that paper, optimal feedback gains minimizing the settling time are obtained as functions of the structural parameters describing the arm link geometry. These structural parameters are then optimized using a gradient projection method in order to acquire an overall optimal performance. A co-design optimization for a combined passive/active automotive suspension for a quarter-car model was executed in [12]. In that case, the controller feedback gains, as well as the physical variables (passive stiffness and damping coefficients), were optimized for an objective incorporating sprung mass acceleration, tire deflection, suspension stroke and maximum active

control force. More recently, research on the multi-objective optimization of a product's disassembly sequence planning has been conducted. For this purpose, a hybrid intelligent algorithm integrating fuzzy simulation and an artificial bee colony optimization is proposed in [13]. Moreover, an artificial bee colony algorithm is also used in [14] to maximize the profit and minimize the energy consumption of a product's disassembly sequence.

Other examples of multi-objective optimizations can be found in [15, 16, 17, 18, 19, 20]. All of the co-design examples mentioned above are limited in the sense that they can only deal with a fixed control architecture. Furthermore, they are focused on one particular application or a specific optimization problem and therefore lack universal applicability. In contrast, the approach described in this paper is a generally applicable method, capable of optimizing the control architecture itself as part of the control configuration optimization.

1.4. Different control architectures

Concerning the control configuration for systems consisting of multiple subsystems, there are three general control architecture methods, being centralized, decentralized and distributed control (see Figure 2). With centralized control, all subsystems are manipulated using only one central controller. The use of a completely centralized controller becomes infeasible as the number of subsystems increases. Decentralized control uses an individual controller unit for every subsystem without interaction between these controllers. In between centralized and decentralized control, distributed control can be implemented. Here, the control tasks are mapped on different processing units while control data is transferred between the controllers using a communication system [7]. [21] provides an overview and classification of decentralized, distributed and hierarchical control architectures for large-scale systems. Comparison of these methods has been extensively described in [22]. The conclusions tend to favour distributed control, which involves local information exchange between subsystem controllers [23].

1.5. Control structure optimization (without hardware architecture optimization)

In the following examples, the control configuration is optimized, but no hardware architecture optimization is performed. In [24], the differences between centralized and decentralized control for the Airbus A320 longitudinal and lateral dynamics are investigated. The optimal control architecture (centralized $\leftrightarrow$ decentralized $\leftrightarrow$ distributed) was also examined for a spatial six-degree-of-freedom electro-hydraulic parallel robot [25]. In [26], a distributed Model Predictive Control (MPC) method was presented and compared with centralized and decentralized MPC for a sextuple water tank system. Results show that the optimal control method depends on the extent to which the model is identified and whether reference tracking or disturbance rejection is preferred.

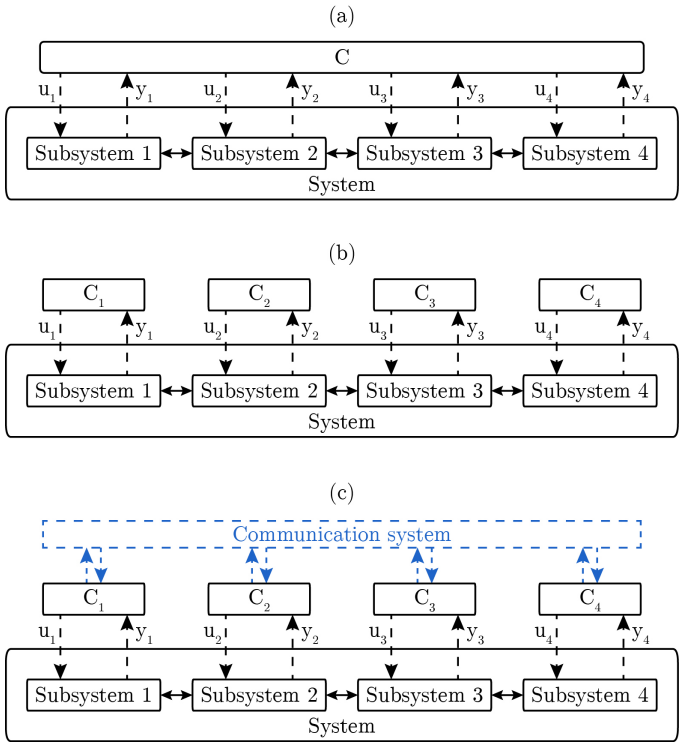

Figure 2: Schematic overview of a system consisting of four interacting subsystems (a) with one centralized controller, (b) with four decentralized controllers, one for each subsystem, (c) with distributed controllers. The communication system used for the information exchange between the controllers is highlighted in blue.

1.6. Co-design with fixed control architecture

As far as the co-design of hardware architecture and control configuration is concerned, in current literature, only the control parameters are optimized as part of the control configuration optimization, while the control architecture is always assumed fixed. There are, however, many control architecture features which are very often used in industrial motion control and for which (to the author's knowledge) no other work is found that considers changing control architectures. This is a substantial disadvantage as the optimization of the control architecture has the potential to generate a considerable gain in performance. The optimization algorithm presented in this paper also allows to simultaneously optimize the control architecture (together with the control parameters) as part of the control configuration optimization. This results in a very powerful tool that is capable of performing a profound co-design of hardware architecture and control configuration.

1.7. A Genetic Algorithm meets the optimization problem requirements

A possible algorithm that can cope with the optimization requirements is a Genetic Algorithm (GA) [27]. As for many optimizing algorithms, a Genetic Algorithm searches for an optimal solution by minimizing a fitness value, which is calculated based on a user-specific objective function. The objective function can be programmed freely according to, for instance, settling time, tracking error, energy consumption, vibrations, or a weighted combinations of these objectives. A disadvantage of using a GA is the relatively long computational time. During the optimization, non-linear constraints can be taken into consideration. These non-linear constraints can be programmed in agreement with, for example, hardware cost, maximum actuator output, control complexity or (full) state constraints. In [28, 29, 30], the full state constraints on a class of strict-feedback non-linear systems are handled using an adaptive fuzzy control with barrier Lyapunov functions (BLF).

Genetic Algorithms have been successfully applied to determine optimal controller settings. [31] used a GA to define the optimal LQR settings to control an inverted pendulum, while [32] used the same methodology for a quarter-car MacPherson strut suspension. A GA can also be used to optimize H_∞ control, as shown for a quarter-car model in [33]. [34] used a GA to optimally tune a fractional-order PID-controller to manage an automatic voltage regulator system. Other examples of the implementation of a Genetic Algorithm to optimize feedback gains can be found for (quarter) car active suspension control [35, 36], multi-machine power systems [37], (double-parallel) inverted pendulum systems [38] or hovercraft control [39]. The papers mentioned above applied the Genetic Algorithm only to determine the optimal feedback gains. A Genetic Algorithm can also be used to optimize the trajectory for single-axis mechanisms performing repetitive tasks [40].

Moreover, [41] applied a Genetic Algorithm to define the optimal placement of an actuator/sensor pair for a simple cantilever beam, which is a common element of large flexible structures or robotic

arms. This was done by moving transmission zeros as far as possible to the left in the left-half plane. Once the Genetic Algorithm found the optimal location for the actuator/sensor pair, the closed-loop response of the system was improved by using another Genetic Algorithm to design the optimal digital controllers. In [42], a dynamic programming algorithm is used to control a hybrid power train system in a hybrid hydraulic excavator under different system parameters, after which a Genetic Algorithm is employed to acquire the power train system parameters for optimal energy consumption. The last two examples are however sequential co-design methods.

1.8. Additions to the current state-of-the-art

In this paper, a novel optimization procedure is elaborated to use a Genetic Algorithm to simultaneously determine the optimal location, type and number of sensors and actuators, together with the control loop architecture and its optimal control parameters. The greatest novelty in this work lies in the possibility of optimizing the control loop architecture as part of the simultaneous co-design. This generic optimization algorithm is demonstrated on a model of a mechanical synchronization application. It shows that the optimization algorithm is capable of simultaneously optimizing both the actuator & sensor selection & placement, as well as the control architecture and control parameters. In order to verify the control parameter optimization as part of the co-design, the comparison is made with the conventional Linear Quadratic Regulator (LQR) method [43]. With this LQR method, control parameters are obtained for which the maximum permitted actuator forces are always exceeded. One of the reasons for this is that no constraints can be defined with LQR optimization. This is not the case when optimizing the control parameters using the proposed novel optimization algorithm. This shows that the algorithm is able to make optimal use of the available actuator force. In this paper, the optimization is demonstrated on an LTI state-space system representing a mechanical synchronization model, but the proposed methodology is certainly not limited to LTI state-space systems, as also non-linear and LPV systems can be dealt with.

1.9. Section overview

The paper is organized as follows. Section 2 describes how the presence or absence of actuators and sensors can be determined in state-space systems, as part of the hardware architecture optimization. Next, section 3 details the control configuration optimization, which consists of both the control architecture and the control parameter optimization. Section 4 specifies essential differences between various optimization methods and justifies the use of a Genetic Algorithm for this multi-objective and discontinuous optimization problem. Section 5 details the specific use of a Genetic Algorithm to co-design the hardware architecture and the control configuration. This co-design is illustrated on a

synchronization application, as described in Section 6. Finally, conclusions and a discussion on future work are formulated in Section 7.

2. Hardware Architecture Optimization

A state-space representation can be used to describe a Linear Time-Invariant (LTI) system, as formulated in (1), with n being the number of states in the original state vector $\mathbf{x}$, m being the number of outputs in the initial output vector $\mathbf{y}$ and l being the maximum number of available actuator inputs from input vector $\mathbf{u}$. To define the presence or absence of an actuator, 'actuator placement binaries' $b_{act...} \in [0, 1]$ are used for every possible actuator location. $\mathbf{B}_{act}$ groups these binaries into a diagonal matrix. This can be seen in (2), where $\mathbf{I}$ is an identity matrix with dimension l, multiplied with an array of the actuator placement binaries using an element-wise array multiplication ($\circ$) (also known as a Hadamard product [44], Schur product [45] or entrywise product [46]). As a consequence, $\mathbf{B}_{act}$ is a diagonal matrix with dimensions $l \times l$. Equation 3 shows how these binaries $b_{act...}$ define whether an actuator will affect the LTI state-space system input, or not. The actuators possibly introduce mass dynamics properties into the system, which are reflected in the system matrix $\mathbf{A}$. Therefore, the system matrix $\mathbf{A}$ is adjusted to $\mathbf{A}'$ based on the actuator placement binaries and the accompanying dynamics of the included or excluded actuators.

$$\mathbf{SS}_{sys} = \begin{cases} \dot{\mathbf{x}} = \mathbf{A} \cdot \mathbf{x} + \mathbf{B} \cdot \mathbf{u} \\ \mathbf{y} = \mathbf{C} \cdot \mathbf{x} + \mathbf{D} \cdot \mathbf{u} \end{cases} \text{, with } \mathbf{x} \in \mathbb{R}^n \text{ , } \mathbf{u} \in \mathbb{R}^l \text{ , } \mathbf{y} \in \mathbb{R}^m \tag{1}$$

$$\mathbf{B}_{act} = \mathbf{I} \circ [b_{act1} \quad b_{act2} \quad ... \quad b_{actl}]^T \tag{2}$$

$$\begin{aligned} \dot{\mathbf{x}} &= \mathbf{A}'\mathbf{x} + \mathbf{B}\mathbf{B}_{act}\mathbf{u} \\ &= \begin{bmatrix} A'_{1,1} & A'_{1,2} & \cdots & A'_{1,n} \\ A'_{2,1} & A'_{2,2} & \cdots & A'_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ A'_{n,1} & A'_{n,2} & \cdots & A'_{n,n} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} + \begin{bmatrix} B_{1,1} & B_{1,2} & \cdots & B_{1,l} \\ B_{2,1} & B_{2,2} & \cdots & B_{2,l} \\ \vdots & \vdots & \ddots & \vdots \\ B_{n,1} & B_{n,2} & \cdots & B_{n,l} \end{bmatrix} \begin{bmatrix} b_{act1} & 0 & \cdots & 0 \\ 0 & b_{act2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & b_{actl} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_l \end{bmatrix} \\ &= \begin{bmatrix} A'_{1,1} & A'_{1,2} & \cdots & A'_{1,n} \\ A'_{2,1} & A'_{2,2} & \cdots & A'_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ A'_{n,1} & A'_{n,2} & \cdots & A'_{n,n} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} + \begin{bmatrix} B_{1,1} & B_{1,2} & \cdots & B_{1,l} \\ B_{2,1} & B_{2,2} & \cdots & B_{2,l} \\ \vdots & \vdots & \ddots & \vdots \\ B_{n,1} & B_{n,2} & \cdots & B_{n,l} \end{bmatrix} \begin{bmatrix} b_{act1}u_1 \\ b_{act2}u_2 \\ \vdots \\ b_{actl}u_l \end{bmatrix} \end{aligned} \tag{3}$$

Similarly, the presence or absence of a sensor is represented by using 'sensor placement binaries' $b_{sen...} \in [0, 1]$. Only the physical sensor presence or absence is used. No virtual sensing techniques

are applied in this paper. The sensor placement binaries are grouped in a matrix $\mathbf{C}_{sen}$. This can be seen in (4), where $\mathbf{I}$ is an identity matrix with dimension m, multiplied with an array of the sensor placement binaries using an element-wise array multiplication ($\circ$). Conventionally, the output vector of a state-space system is appointed as $\mathbf{y}$. The sparse rows from $\mathbf{C}_{sen}$ are deleted and therefore $\mathbf{C}_{sen}$ has dimensions $m' \times m$, where m' is variable and is depending on which sensors are active. In this way, $\mathbf{C}_{sen}$ determines which of the output signals from the original state-space system are available to be used as feedback signals, in correspondence with which sensors are active or not. As a consequence, Equation 5 shows how binaries $b_{sen...}$ ensure what signals from the output equation are selected to close the feedback loop. These feedback signals are indicated as $\mathbf{y}'$.

$$\mathbf{C}_{sen} = \mathbf{I} \circ \begin{bmatrix} b_{sen1} & b_{sen2} & ... & b_{senm} \end{bmatrix}^T \tag{4}$$

$$\begin{aligned}
\mathbf{y}' &= \mathbf{C}_{sen}\mathbf{y} \\
&= \mathbf{C}_{sen}\left(\mathbf{C}\mathbf{x} + \mathbf{D}\mathbf{B}_{act}\mathbf{u}\right) \\
&= \begin{bmatrix} b_{sen1} & 0 & \cdots & 0 \\ 0 & b_{sen2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & b_{senm} \end{bmatrix} \cdot ... \\
... &\left(\begin{bmatrix} C_{1,1} & C_{1,2} & \cdots & C_{1,n} \\ C_{2,1} & C_{2,2} & \cdots & C_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ C_{m,1} & C_{m,2} & \cdots & C_{m,n} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} + \begin{bmatrix} D_{1,1} & D_{1,2} & \cdots & D_{1,l} \\ D_{2,1} & D_{2,2} & \cdots & D_{2,l} \\ \vdots & \vdots & \ddots & \vdots \\ D_{m,1} & D_{m,2} & \cdots & D_{m,l} \end{bmatrix} \begin{bmatrix} b_{act1} & 0 & \cdots & 0 \\ 0 & b_{act2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & b_{actl} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_l \end{bmatrix} \right)
\end{aligned} \tag{5}$$

In this way, the basic state-space representation from (1) is extended to (6). Every sensor and actuator placement binary is a design parameter for the optimization procedure, as explained later in the paper. It is important to note here that the $\mathbf{B}$ and $\mathbf{C}$ matrix values are not optimization parameters for the algorithm, but are constant real values describing the initial LTI system, while the actuator & sensor placement optimization is done on $\mathbf{B}_{act}$ and $\mathbf{C}_{sen}$, respectively. This state-space system is graphically displayed in Figure 3.

$$\mathbf{SS}_{sys} = \begin{cases} \dot{\mathbf{x}} = \mathbf{A}' \cdot \mathbf{x} + \mathbf{B} \cdot \mathbf{B}_{act} \cdot \mathbf{u} \\ \mathbf{y}' = \mathbf{C}_{sen} \cdot (\mathbf{C} \cdot \mathbf{x} + \mathbf{D} \cdot \mathbf{B}_{act} \cdot \mathbf{u}) \end{cases} \text{, with } \mathbf{x} \in \mathbb{R}^n \text{ , } \mathbf{u} \in \mathbb{R}^l \text{ , } \mathbf{y} \in \mathbb{R}^m \tag{6}$$

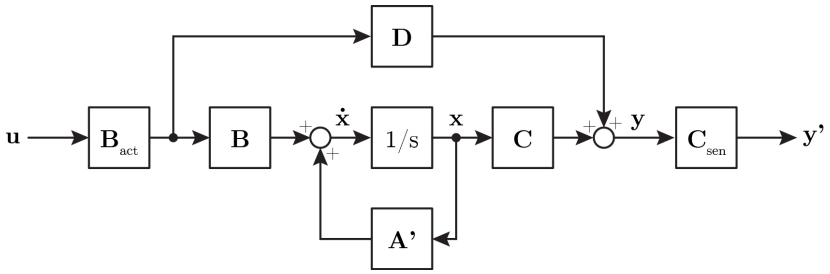

Figure 3: Graphical representation of equation (6).

Next to the presence or absence of the actuators and sensors, indicated by the accompanying binaries $b_{act...}$ and $b_{sen...}$, the optimizing algorithm should also be able to select between different types of actuators and sensors. Therefore, extra hardware selection integer values $i_{act...}$ and $i_{sen...}$ are added and grouped in vectors $\mathbf{I}_{act}$ and $\mathbf{I}_{sen}$, respectively. They correspond with the different possible types of actuators and sensors at every active actuator or sensor position. If multiple actuator types are possible, then the optimizing algorithm can effectively choose between each actuator type with a different cost and maximum actuator output. For example, a more expensive actuator can have a higher control effort. Based on this integer value, the accompanying actuator cost and maximum output are taken into account as non-linear constraints during the optimization.

3. Control Configuration Optimization

The algorithm proposed in this paper allows optimizing the control architecture together with the control parameters. This is in contrast to current literature in which co-design algorithms are found that only consider the control parameter optimization as part of the control configuration optimization. The ability to also optimize the control architecture is a significant added value since modern industrial drives used for motion control have various control architecture possibilities on which the end user has to decide whether or not to implement them. One example of these control architecture capabilities is cascaded control, where multiple SISO (Single Input, Single Output) controllers are connected one after another. In this case, the outer controller generates a control signal that serves as the setpoint for the following controller [47]. In addition to cascaded control, feedforward control can be applied to reduce the tracking error. This is done by pre-commanding the system depending on a reference signal or a disturbance signal, without using any feedback from the plant itself. As a result, reference feedforward or disturbance feedforward can be obtained [47]. Finally, also synchronizing control (also known as cross-coupling) can be used as a control architecture, referring to a control loop extension used to improve output synchronization between coupled subsystems that are both parts of a larger system [48]. Thereby, the synchronous control of subsystem outputs is improved by taking into account

the output deviations. This can be applied by adding separate controllers (e.g., P/PI/PID) acting on the relative fault signals between different actuators. This can reduce the error between the different outputs when one or more subsystems are disturbed internally or externally [48].

In order to allow the overall optimization routine to switch between different control architectures, each control architecture feature is accompanied by a binary indicating its presence or absence. As a result, every decentralized or distributed controller loop, feedforward control loop, and synchronizing control loop can be activated or deactivated by a binary in the control architecture optimization. These control architecture selection binaries (e.g., b_{dis}, b_{FF}, b_{SC} for distributed, feedforward or synchronizing control, respectively) are grouped in vector $\mathbf{B}_c$. The entire design space is thus defined by the case in which all possible control features are active. In addition to the control architecture, the control parameters are also simultaneously optimized by the overall optimization algorithm. Vector $\mathbf{R}_c$ groups all control parameter values.

4. Optimization Algorithm Selection

The co-design optimization algorithm must be able to deal with binary numbers (sensor and actuator placement, selection of the applied control loop architectures), integer values (sensor and actuator selection) and real numbers (control parameters), resulting in a mixed-integer problem [49]. This also implies that the objective functions are discontinuous and therefore non-smooth and non-convex. A schematic overview of the optimization variables is depicted in Table 1, while a mathematical formulation of the optimization problem can be seen in Equation 7.

Optimization variables	**Type**			**Grouped in**
Sensor and actuator placement	binary	$\in \mathbb{B}$	(e.g., b_{act1}, b_{sen1})	Matrixes $\mathbf{B}_{act}$ and $\mathbf{C}_{sen}$
Sensor and actuator selection	integer	$\in \mathbb{Z}^+ \mid \leq z$	(e.g., i_{act1}, i_{sen1})	Vectors $\mathbf{I}_{act}$ and $\mathbf{I}_{sen}$
Control loop architecture selection	binary	$\in \mathbb{B}$	(e.g., b_{FF}, b_{SC})	Vector $\mathbf{B}_c$
Control parameters	real	$\in \mathbb{R}$	(e.g., K_{FF}, K_{SC})	Vector $\mathbf{R}_c$

Table 1: Overview of the optimization variables, with z = number of possible actuator or sensor types.

$$\underset{b\in\mathbb{B},\, i\in\mathbb{Z}^+,\, K\in\mathbb{R}}{\textbf{min}}\ \mathit{Objective\ function}(b,i,K) \quad \text{subject to} \begin{cases} \text{non-linear constraint 1} \\ \text{non-linear constraint 2} \\ \ldots \end{cases} \tag{7}$$

It is clear that the overall co-design issue is, in fact, a multi-objective optimization problem since this optimization is executed for a combination of multiple objectives, such as energy consumption, actuator effort, reference tracking or vibration reduction. In general, one may distinguish between

derivative-free and gradient-based algorithms as the two main approaches to deal with multi-objective optimization problems.

In gradient-based optimizations, a general Linear Time-Invariant (LTI) system is represented as a Linear Matrix Inequality (LMI). From this LMI, a convex optimization problem can be expressed and mathematically solved [50]. An advantage of the gradient-based algorithms is that they often need less time to converge to an optimal solution [51]. There are also two main limitations of these methods.

First, additional constraints usually result in convergence in a local optimum, being an optimal point in a convex subset. This local optimum is not necessarily a global optimum, being the optimal point in the complete set. Second, a convexification-based approach for non-convex problems only works for some specific cases, resulting in limited applicability [20]. Because the functions for this co-design optimization problem are non-smooth and even discontinuous, derivative or gradient information generally cannot be used to determine the direction in which the function is increasing (or decreasing) [52].

In contrast to the gradient-based approach, the derivative-free algorithms do not rely on the derivatives and can, therefore, work exceptionally well when the objective function is noisy and discontinuous [53]. Derivative-free methods are often the best global-searching algorithms as they sample a large portion of the design space [54]. A drawback of these methods is that they are computationally expensive and also stochastic and thus non-deterministic. They may yield different solutions on different runs, even when started from the same point on the same model, depending on which points are randomly sampled [52]. Some examples of commonly used derivative-free algorithms are: DIRECT or Dividing Rectangles [54], Simulated Annealing (SA) [55], Genetic Algorithms (GA) [54], Surrogate Optimization [56], Particle Swarm Optimization (PSO) [57] or Artificial Bee Colony Algorithm (ABC) [13].

Concerning the optimization algorithms mentioned above, only Direct Search, Genetic Algorithm, Particle Swarm Optimization and Surrogate Optimization can be used to find a global minimum for non-smooth objective functions and constraints. From these, the Genetic Algorithm is the only optimization algorithm that also supports mixed-integer optimization [58]. Consequently, it can be concluded that at the moment and by the author's knowledge, the Genetic Algorithm is the only derivative-free algorithm available that is able to administer the requirements for the optimization problem discussed in this paper.

It is important to note that depending on the variables to be optimized, a model is obtained that is always linear time-invariant (LTI) (if the initial system was already LTI), but the overlying optimization problem is, in fact, a highly discontinuous, multi-objective and mixed-integer optimization

problem.

5. Genetic Algorithm

Genetic Algorithms (GA) are a group of optimization algorithms based on evolutionary processes and Darwin's concept of natural selection [54]. It starts by (randomly) generating an initial population of individual solutions utilizing a creation function. Every solution consists of a combination of proposed design parameters (sensor and actuator placement binaries $\in \mathbb{B}$, sensor and actuator selection integers $\in \mathbb{Z}^+$, control loop architecture binaries $\in \mathbb{B}$ and real numbers for the control parameters $\in \mathbb{R}$). The limits for the binaries are fixed at [0,1], while the limits on the integers are defined by the number of possible types of sensors and actuators. The limits for the real numbers representing the control parameters are the only design parameter limits that have to be specified by the end user. Too small limits cause the algorithm to search only over a limited design space. That is why these limits are preferably chosen relatively large. However, excessively large limits may cause the algorithm to need more iterations leading to a longer computational time.

For every individual solution, a fitness value is calculated using an objective function (often referred to as 'fitness function'), resulting in a definite score of how well the individual performs. As this is a minimization function, individuals with a lower fitness value have a better performance. When assigning a fitness value, the constraints are also taken into consideration. Individual solutions that do not meet certain constraints are penalized with a much higher fitness value as a result.

Based on these fitness values, GA then applies three different operations to create a new generation of individuals, being elite selection, crossover and mutation. Elite selection implies that individuals with the best fitness have the most significant probability of being directly selected to pass on to the next generation, unchanged. Crossover is where parts of two individuals from the initial population (sensor & actuator placement binaries, sensor & actuator selection integers, control loop architecture selection binaries or control parameters) are exchanged to get two new individuals. Mutation is where variables of one individual from the initial population are randomly changed to get a new individual. The crossover fraction indicates the ratio between crossover and mutation.

The above process is executed to generate a new generation of individuals and is continued to further improve the fitness of the best individual until a stopping criterion is met [54]. Possible stopping criteria are related to a maximum calculation time, a maximum number of generations or a minimum amount of change in the average fitness values of successive generations. The workflow of the algorithm is schematically displayed in Figure 5.

An advantage of the Genetic Algorithm is that there is a large amount of freedom in the scoring of each individual, as this can easily be programmed in the 'fitness function'. In this way, the individual's

fitness can be calculated according to (a weighted sum of) several objectives, such as settling time, reference tracking, energy consumption or robustness. This fitness function can be non-linear and even discontinuous. The implementation of discontinuous and highly non-linear constraints can easily be programmed in a 'constraint function'. This can be achieved without any convexification-based approach necessary for gradient-based optimization algorithms.

A disadvantage of the Genetic Algorithm is that a relatively long calculation time is required compared to gradient-based methods [59]. Because this is also a heuristic algorithm, it cannot be fully guaranteed that the solution found by the Genetic Algorithm is the global minimum of the optimization problem [60]. However, the randomization avoids being stuck in a local minimum as much as possible [61].

The overall optimization problem has multiple conflicting objectives and therefore has no unique optimal solution. Instead, the concept of Pareto optimality can be used to characterize the trade-off between different objectives. Pareto optimality is obtained when one objective of an individual solution cannot be improved without one or more other objectives deteriorating. Such an individual solution is then described as a Pareto point. All Pareto points collectively form a Pareto front that indicates the boundary of the Pareto optimality [62, 63]. In the context of this paper, the main results from the optimizing algorithm can be graphically shown in a Pareto front, revealing the trade-off between the total implementation cost and the optimal performance to be achieved. A general example of such a Pareto front is given in Figure 4. In this figure, every point represents the performance of a particular plant composition, being hardware architecture and control configuration (with controller tuning). As stated before, the lower the fitness value, the better the performance. The best performing plant compositions are situated at the Pareto front (in blue), and these show the maximum achievable plant performance for a given total plant cost. If, for example, a plant design is operating with a sub-optimal performance (not on the blue Pareto front), two different actions can be taken: performance can be increased without an extra cost (arrow 1) or the investment cost can be reduced without deterioration in performance (arrow 2).

It is clear that this is a very useful tool to graphically provide insights for the end user into the trade-off between performance and cost, which is otherwise very complicated to predict.

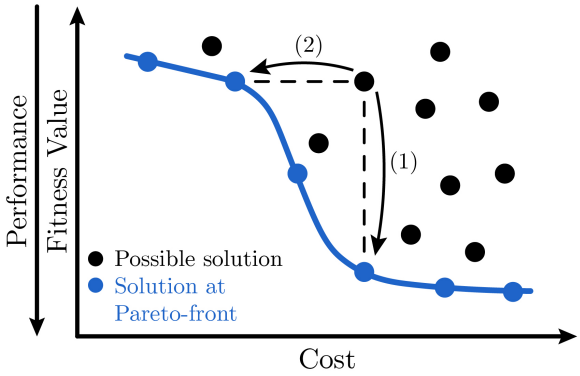

Figure 4: General overview of a Pareto front.

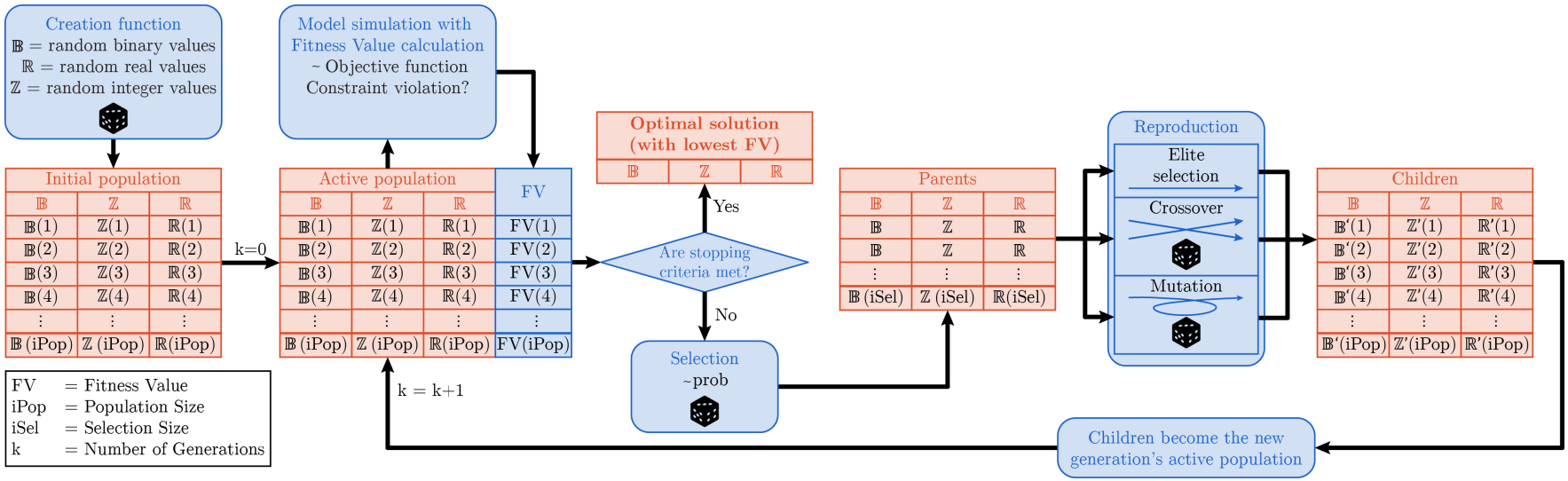

Figure 5: Schematic overview of a Genetic Algorithm's workflow.

6. Mechanical Synchronization Setup

6.1. Setup Introduction

The performance of the optimization algorithm is tested on a motion control application in which a central load inertia needs to be positioned. Common examples of this type of application are steel ladles used in the steel industry or overhead gantry cranes. In this case, the central load inertia can be driven by either one or two actuators on the sides of this load inertia. A graphical overview of the application is given in Figure 6, with θ the angular displacement [rad], T_1 the actuator 1 torque [Nm], T_2 the actuator 2 torque [Nm], T_L the load torque [Nm], k the torsion spring constant [Nm/rad] and b the angular damping constant [Nms/rad].

The positioning torque can be applied through two possible motor actuators at each side of the load inertia. Note that these motors also introduce inertia to the structure. The three bodies are dynamically coupled through rotational springs and dampers (k and b), as depicted in Figure 6. For this application, there are three possible sensor locations: one on the load inertia and one on each of the outer actuator motors. While referring to the figures in the introduction, the system can be represented schematically as in Figure 7. In this diagram, the three inertias can be seen as dynamically linked subsystems, in which the outer subsystems can be controlled with sensor information from the respective inertia or the central load inertia. The distributed control can be represented as the information exchange using a communication system between the controllers on the outermost subsystems. Table 2 shows the key parameters for the setup.

The reference trajectory r_θ to be followed by the load inertia can be seen in Figure 8 (a) and consists of a trapezoidal movement to an angle of 1.8 radians or roughly 103 degrees. The shape of the associated load torque can be seen in Figure 8 (b). Furthermore, output disturbances d are also added to create a more realistic situation where disturbances are present in the measured sensor signal. In this way, it can be verified whether the obtained controllers respond adequately to the relevant course of the output signals and not to the additional noise. This disturbance is a uniformly distributed noise with an amplitude equal to 0.05 rad.

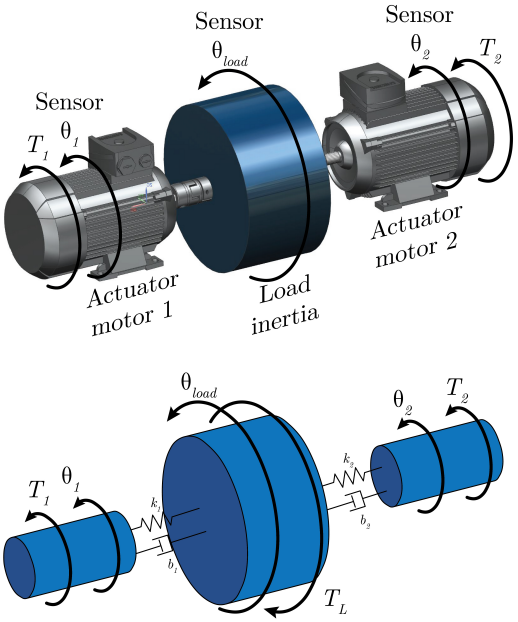

Figure 6: Motion control application in which two actuators have to position a central load inertia (top), with schematic representation (bottom).

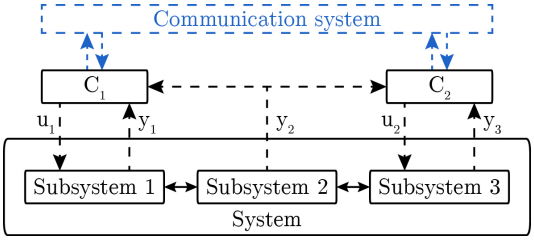

Figure 7: Subsystem devision with controller arrangement for a mechanical synchronization application.

Parameter		**Value**	**Unit**
Actuator 1 inertia	$= J_1$	0.173	kgm^2
Actuator 2 inertia	$= J_2$	0.173	kgm^2
Load inertia	$= J_{load}$	3	kgm^2
Torsion spring constant 1	$= k_1$	150	Nm/rad
Torsion spring constant 2	$= k_2$	150	Nm/rad
Angular damping constant 1	$= b_1$	0.2	Nms/rad
Angular damping constant 2	$= b_2$	0.2	Nms/rad

Table 2: Key parameters for the motion control application.

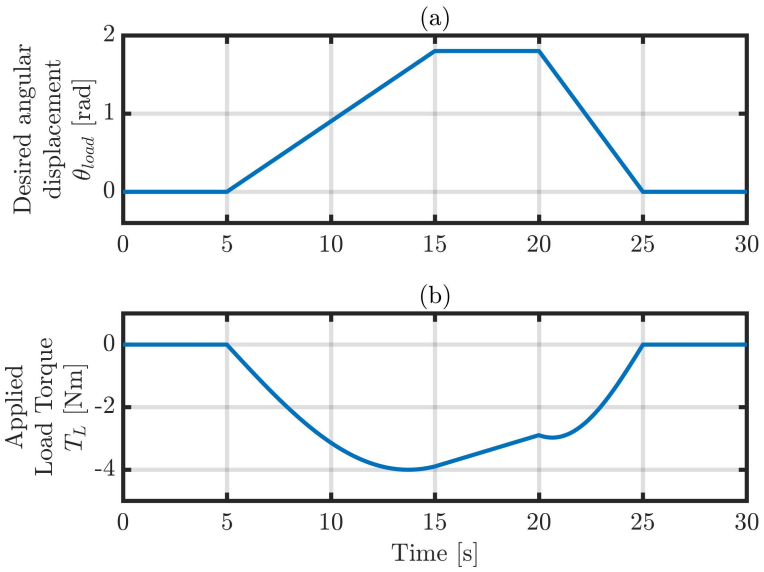

Figure 8: (a) Angular displacement trajectory to be followed by the central load inertia θ_{load}. (b) Applied load torque T_L.

6.2. Pareto front

The primary goal is to let the optimizer determine the Pareto front, graphically showing the trade-off between implementation cost and achievable performance. For this example, the optimizing algorithm can select the presence or absence and type of the two actuators at the sides of the load from a list of three possible motors, as can be seen in Table 3. Every motor has a specific cost and a corresponding maximum actuator torque. Evidently, more expensive motors have a higher maximum torque. As a result, the actuator selection integers i_{act1} and i_{act2} for the left and right actuator can have values from 0 to 3, according to the selected actuator. If the actuator selection integer $i_{act...}$ is equal to zero, no actuator is present on that location, and the corresponding actuator placement binary $b_{act...}$ is equal to 0.

In this setup, there is a maximum of three inertias. This means that there are also three possible sensor locations providing position and speed feedback from the corresponding inertia. It is more convenient to add a sensor on the actuator inertias since most actuators already provide relatively easy mounting capabilities for encoders. This is why a sensor on the load inertia is more expensive compared to a sensor on the outer inertias. Table 4 shows the possible sensor locations and the accompanying cost. Each sensor is assumed to feed back angular displacement and velocity. As a result, the sensor selection integers i_{sen1} and i_{sen2} for the left and right actuator can each have integer values from 0 to 2, depending on whether no sensor is used, a sensor on the corresponding actuator

Actuator Selection Integers $i_{act...}$		**Cost**	**Maximum Torque**	**Actuator Placement Binary $b_{act...}$**
0	No actuator	€ 0	0 Nm	0
1	Actuator type 1	€ 400	2 Nm	1
2	Actuator type 2	€ 1000	4 Nm	1
3	Actuator type 3	€ 2000	15 Nm	1

Table 3: Different actuator types with their accompanying cost, maximum torque and corresponding actuator placement binaries.

inertia or a sensor on the middle inertia is used. If the sensor selection integer $i_{sen...}$ is equal to zero, no sensor is present, and the sensor placement binary $b_{sen...}$ is equal to 0.

Section 3 deals with the control architecture possibilities that can be implemented and optimized. For this motion application, the control loop possibilities are depicted in Figure 9. For each actuator, the inner cascaded loop is a PI speed controller, while the outer loop consists of a P position controller. The algorithm can evaluate decentralized and distributed control for both the inner and outer loop controllers. In the case of decentralized control, there is no controller interaction present, as opposed to distributed control where the controllers are interconnected to each other. As this motion control application is a setup where synchronization is essential, the synchronizing control on speed and position can also be used. Moreover, this is an application in which a trajectory has to be followed. Therefore, feedforward control is also relevant.

Sensor Selection Integers $i_{sen...}$		**Cost**	**Sensor Placement Binary $b_{sen...}$**
0	No sensor data used	€ 0	0
1	Sensor on corresponding actuator	€ 200	1
2	Sensor on load inertia	€ 600	1

Table 4: Sensor selection integers for every actuator with their accompanying cost and corresponding sensor placement binaries.

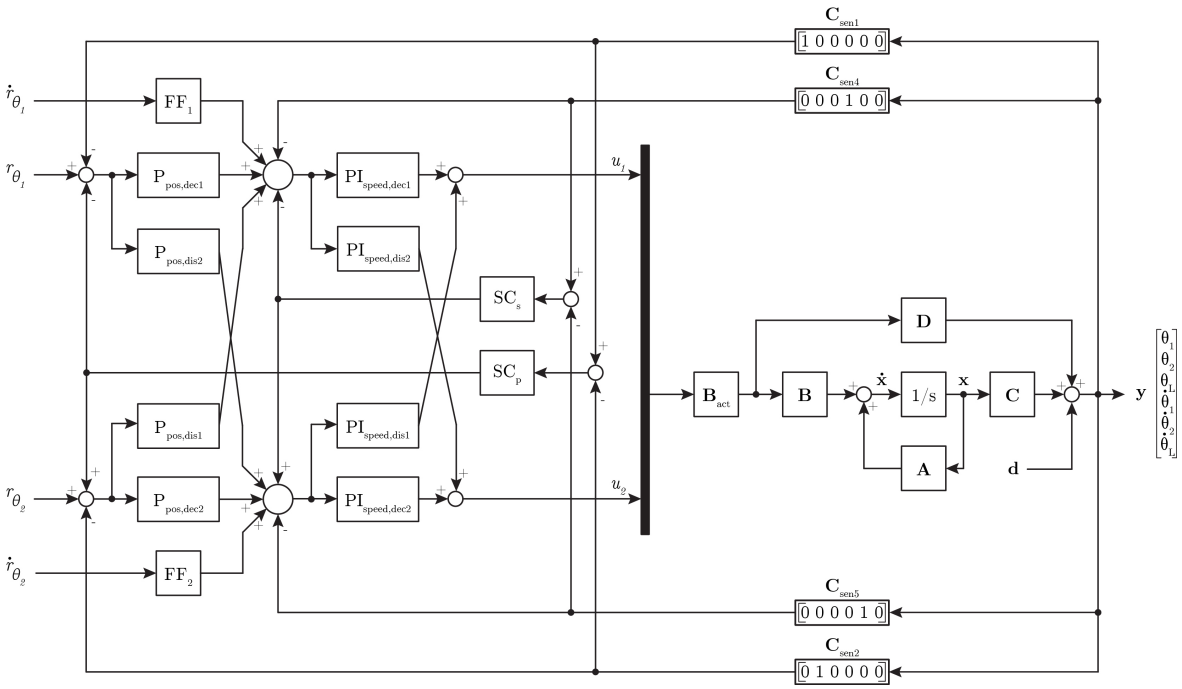

Figure 9: Control scheme of a mechanical synchronization application controlled by a distributed cascaded controller with feedforward and synchronizing control.

Table 5 shows the possible control architecture features that can be implemented, with their respective implementation costs. These costs are arbitrary and can be changed by the end user according to their estimated value. The cost estimation can be influenced by, for example, hardware cost, programming cost or estimated training cost for control technicians.

370 In order to determine the performance of an individual solution, the entire closed-loop system is simulated and the fitness value is calculated from the simulation results using (8). As specified in

Description	**Symbol**	**Cost**
Decentralized outer loop P-position control	b_{Pdec}	€ 50
Distributed outer loop P-position control	b_{Pdis}	€ 75
Decentralized inner loop PI-speed control	b_{PIdec}	€ 100
Distributed inner loop PI-speed control	b_{PIdis}	€ 140
Feedforward control	b_{FF}	€ 160
Synchronizing control on position	b_{SCp}	€ 300
Synchronizing control on speed	b_{SCs}	€ 320

Table 5: Control architecture features with their corresponding cost.

Section 5, many fitness function criteria can be applied. Here, the control objective (or fitness value) to be minimized is a weighted sum consisting of two parts. The first and most important objective is to have an optimal trajectory tracking by the load inertia. For this, the ISE (Integral of Square Error) error performance index is used which evaluates the central load reference tracking [64], as can be seen in the first part of (8). The second objective is to reduce undesirable variations in accelerations, as this nervous behaviour reduces the life span of the system. Therefore, the accelerations on all inertias are penalized by adding the RMS (Root Mean Square) value of these accelerations to the second part of the calculation of the fitness value. Furthermore in (8), T is the total simulation time for every individual solution (= 30s), r is the reference trajectory (see Figure 8a) and w_1 and w_2 correspond with the weights on reference tracking and acceleration, respectively.

$$\begin{aligned} F.V. =& w_1 \left(\int_{t=0}^{t=T} \left(r(t) - \theta_{load}(t)\right)^2 dt \right) + \\ & w_2 \left(\sqrt{\frac{1}{T} \int_{t=0}^{t=T} \ddot{\theta}_1(t)^2 dt} + \sqrt{\frac{1}{T} \int_{t=0}^{t=T} \ddot{\theta}_{load}(t)^2 dt} + \sqrt{\frac{1}{T} \int_{t=0}^{t=T} \ddot{\theta}_2(t)^2 dt} \right) \end{aligned} \tag{8}$$

Consequently, the variables to be optimized by the Genetic Algorithm for this application are 2 actuator selection integers (i_{act1} and i_{act2}), 2 sensor selection integers (i_{sen1} and i_{sen2}), 7 control architecture binaries (grouped in $\mathbf{B}_c$) and a maximum of 18 real numbers for the control gains (grouped in $\mathbf{R}_c$). These add up to 29 optimization variables in total. The Genetic Algorithm can optimize these variables according to the fitness function (see (8)), with respect to constraints on maximum actuator force and total plant cost. Similar to the application of previously mentioned constraints, constraints could also be applied to the states itself, resulting in full-state constraints.

For both actuators, there are three possible actuator types and two possible sensor types, where seven different control architecture features may or may not apply. Even if the optimization of the control parameters is not considered, there is a huge number of different possible plant compositions, each with a corresponding cost. It would certainly not be efficient to determine the optimal control parameters for each possible plant composition. Therefore, the optimal plant composition is only determined for a limited number of total plant costs. As a result, the entire plant composition is optimized several times, each time with a different maximum total plant cost. In this way, the Pareto points are determined. For this case, ten maximum plant composition costs are chosen, evenly distributed between the smallest and largest possible plant composition cost, being €750 and €6350. For every point, the maximum calculation time for the Genetic Algorithm is set to 20 minutes, leading to approximately 3 hours and 20 minutes of total calculation time. Other key settings for the Genetic Algorithm are shown in Table 6. The optimization algorithm was performed on an Intel® Xeon® CPU @ 3.10 GHz with 64 GB of RAM.

The maximum number of iterations is set to 'infinite'. In this way, no maximum number of iterations is given to the Genetic Algorithm as stopping criteria. The constraint tolerance is the tolerance on the difference between consecutive averages of fitness values of successive generations. If the difference between the average fitness value over a number of generations is smaller than the constraint tolerance, the Genetic Algorithm will assume that a global minimum has been found. For this case, the constraint tolerance is chosen small with the purpose that the GA would not conclude too quickly that a global minimum is found and in this way, the optimizing algorithm makes full use of the specified optimization time. The crossover fraction determines in which ratio crossover and mutation are applied to achieve a new generation. This value is set in such a way as to have a high mutation level when determining a new generation of possible solutions. This is desirable to keep the search field large and not to end up in a local minimum as the optimization problem is discontinuous.

Description	**Value**
Maximum number of iterations	infinite
Constraint tolerance	1e-10
Crossover fraction	0.5
Population size	100
Elite count	10

Table 6: Genetic Algorithm's key settings.

6.3. Results

Table 8 in Appendix A shows the optimization results for every point in the Pareto front. A graphical overview of the results is shown in Figure 10. Depending on the binaries that determine the control architecture, some controller gains no longer apply, shown by a '/' in Table 8. No feasible solution was found for the first Pareto point with a maximum total plant cost equal to €750. In other words, the limited amount of €750 does not allow for a useful plant arrangement. In any other case, the maximum applicable actuator force (depending on what type of actuator is applied) is not exceeded by the results.

The second Pareto point with a maximum total plant cost equal to €1380 has a relatively high fitness value, thus a very low performance. A detailed view is depicted in Figure 11 to better inspect what the Pareto front looks like besides these first two outliers. As expected, the results show that the fitness value decreases or in other words, the performance increases as a larger plant composition cost is available. Since the performance of the system improves considerably from the third Pareto point onwards, it would be wise for the end user to decide to invest at least €2010 to obtain a satisfying

operation of the setup. It is up to the end user to choose whether the gain in performance due to higher investments is actually worth it, or not.

For comparison, a conventional LQR method was also applied to obtain the control parameters. Note that this can only be done for a fixed hardware and control configuration. With this method, relative weights are assigned to the states and inputs using Q and R matrices, respectively. Hereafter, the control gains are calculated [47]. Still, finding the right weights for this LQR method turns out to be another difficulty. [43] describes an approach to determine these weights as the inverse of the square of the maximum value for the corresponding state or input [65]. This method was applied to the motion application using distributed outer loop P-position control with two actuators of type 2 and 3, successively. Not only was the method described in [43] used to obtain Q and R matrices, but also manual fine-tuning of these matrices was performed to check if the resulting controller gains could be further improved. In each case, sensors on the corresponding motor inertias are used ($i_{sen1} = i_{sen2} = 1$). Table 7 shows the results of the LQR method, and these are also shown in Figure 11 as red points. When using actuators of type 2, both the approach by [43] and manual fine-tuning lead to control parameters where the maximum actuator force of 4 Nm is exceeded, which is not the case with optimization using the Genetic Algorithm. The approach described in [43] was also applied with type 3 actuators with a maximum actuator force of 15 Nm. This also leads to a situation in which the maximum actuator force is exceeded. After further manual fine-tuning, a situation was achieved with an actuator force lower than the maximum actuator force, but still, the performance is worse than the control found by the Genetic Algorithm, as can be seen in Figure 11. From this, it can be concluded that the obtained system using the LQR method always has a lower performance compared to the optimization using the Genetic Algorithm.

Figure 12 illustrates the central load displacement for different plant compositions with a maximum total plant cost of consecutively 750, 2010 and 6420 euro. This plot demonstrates that the load is more capable of following the desired trajectory for higher plant costs, which is logical as more hardware and control architecture features become applicable.

i_{sen1}	i_{sen2}	i_{act1}	i_{act2}	**Applied method**	**Maximum actuator force [Nm]**	**Fitness Value**
1	1	2	2	Method proposed by [43]	5.2 ($> f_{max,act}$)	273
1	1	2	2	Manual fine-tuning	5.3 ($> f_{max,act}$)	290
1	1	3	3	Method proposed by [43]	55.2($> f_{max,act}$)	1365
1	1	3	3	Manual fine-tuning	6.8	162

Table 7: Results after applying the LQR method. A comparison of these results with the results obtained using the Genetic Algorithm is shown in Figure 11.

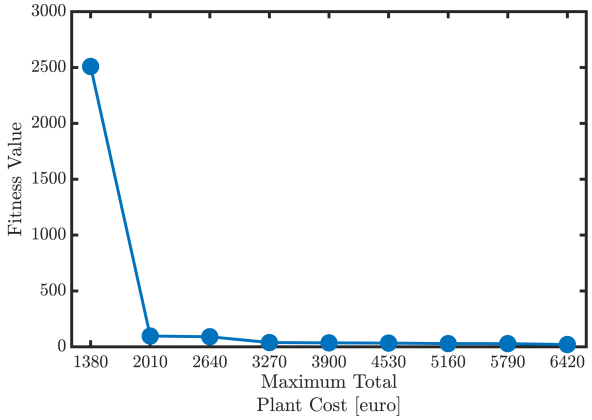

Figure 10: Resulting Pareto front (overview).

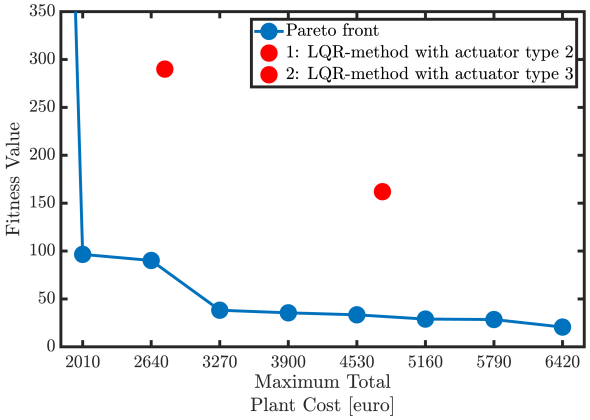

Figure 11: Resulting Pareto front (detailed view with comparison to LQR optimization).

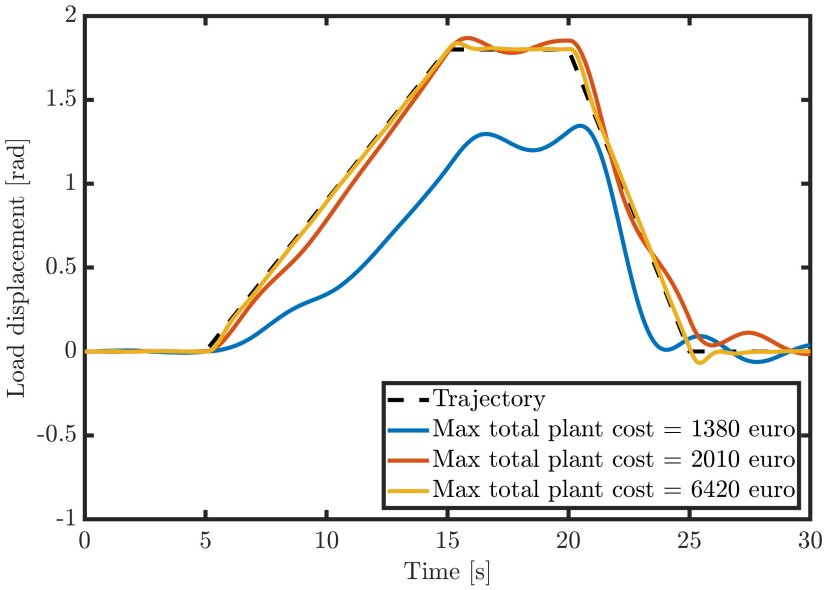

Figure 12: Resulting load displacement for different maximum total plant costs.

7. Conclusions and Future Work

Concerning the co-design of hardware architecture and control configuration, the current literature only considers the optimization of the optimal location and number of sensors and actuators, together with corresponding control parameters. The main novelty of this paper is the addition to optimize the control loop architecture as part of the simultaneous co-design. In addition, the algorithm also allows optimizing for different types of actuators and sensors. The proposed methodology to co-design the hardware architecture and control configuration provides useful insights into the trade-off between

the maximum achievable performance and the total implementation cost, graphically represented in a Pareto front.

The Pareto points found by the optimizing algorithm using a GA have a lower fitness value and thus a better performance compared to the conventional LQR method. This shows that the Genetic Algorithm successfully optimizes not only the hardware and control architecture but also the control parameters. If a complete Pareto front is to be compiled using the LQR method rather than the optimizing algorithm proposed in this paper, this should be done for every possible hardware architecture and control architecture. It is therefore easy to see that the proposed optimizing algorithm produces a Pareto front more efficiently. Moreover, the algorithm can better optimize the control parameters to obtain an even better performance in comparison to the LQR method. If desired by the end user, the cost interval between the Pareto points can be reduced to obtain a Pareto front with a higher resolution. This will, of course, increase the computational time.

The Genetic Algorithm is a heuristic method and computationally expensive. Calculation time can be reduced by integrating specific prior knowledge of the plant and its composition. Nevertheless, it can be concluded from the results presented in the paper that a GA is able to determine the optimal achievable performance, depending on the maximum implementation cost with respect to the selected actuators, sensors, control architectures and (discontinuous) constraints. It is clear that this can significantly help the end user to get a better understanding of the trade-off between optimal achievable performance and implementation cost. In this paper, the co-design was successfully executed on a mechanical synchronization application with a relatively low number of possible hardware combinations. For larger applications with more possibilities, it is also more difficult to intuitively estimate the impact of changing configurations on the final performance of the system. This will make the practical use of this method even more appealing. Additionally, for systems that are already operational, this tool can be a great added value to evaluate the potential for performance improvements, related to investment costs.

The method proposed in this paper is a general method applicable to a very broad range of system classes. These include, for example, mechatronic, electrical, thermal or chemical system classes. The authors encourage industrial partners and other research groups to apply this method on different systems.

Acknowledgements

This work is supported by Flanders Make: SBO ROCSIS: Robust and Optimal Control of Systems of Interacting Subsystems.

Appendix A: Optimization Results

Pareto point	**Maximum cost**	**Fitness value**	**Actuator 1 selection integer** (i_{act1})	**Actuator 2 selection integer** (i_{act2})	**Sensor 1 selection integer** (i_{sen1})	**Sensor 2 selection integer** (i_{sen2})	**Decentralized outer loop P controller binary** (b_{Pdec})	**Distributed outer loop P controller binary** (b_{Pdis})	**Decentralized inner loop PI controller binary** (b_{PIdec})	**Distributed inner loop PI controller binary** (b_{PIdis})	**Feedforward control binary** (b_{FF})	**Synchronizing control on position binary** (b_{SCp})	**Synchronizing control on speed binary** (b_{SCs})	$P_{pos,dec1}$ **controller** K_p **gain**	$P_{pos,dec2}$ **controller** K_p **gain**	$P_{pos,dis2}$ **controller** K_p **gain**	$P_{pos,dis1}$ **controller** K_p **gain**	$PI_{speed,dec1}$ **controller** K_p **gain**	$PI_{speed,dec2}$ **controller** K_p **gain**	$PI_{speed,dis2}$ **controller** K_p **gain**	$PI_{speed,dis1}$ **controller** K_p **gain**	$PI_{speed,dec1}$ **controller** T_i **gain**	$PI_{speed,dec2}$ **controller** T_i **gain**	$PI_{speed,dec2}$ **controller** T_i **gain**	$PI_{speed,dec1}$ **controller** T_i **gain**	SC_p **controller** K_p **gain**	SC_s **controller** K_p **gain**	SC_p **controller** T_i **gain**	SC_s **controller** T_i **gain**	FF_1 **controller gain**	FF_2 **controller gain**
1	750	inf	0	0	0	0	○	○	○	○	○	○	○	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/
2	1380	2.5E3	1	0	2	1	•	•	○	○	○	○	○	5	5	1	1	/	/	/	/	/	/	/	/	/	/	/	/	/	/
3	2010	97	2	0	1	0	○	○	•	○	•	○	○	/	/	/	/	5	5	/	/	1	1	/	/	/	/	/	/	1	1
4	2640	90	3	0	1	1	•	•	•	○	○	○	○	4.5	4.7	2.3	1.6	6	5	/	/	0.9	0.7	/	/	/	/	/	/	/	/
5	3270	38	3	0	1	1	○	•	•	•	•	•	○	/	/	0.9	1	5	5	1.2	1	1.1	1	1.1	0.9	1	/	1.2	/	1	1.1
6	3900	35	3	0	1	1	○	•	•	•	•	•	○	/	/	0.9	0.9	5.9	5.4	1.4	1	1	0.7	0.8	1	0.9	/	1	/	0.8	1.3
7	4530	33	3	0	1	1	○	○	•	•	•	○	○	/	/	/	/	5.3	5.1	0.9	0.3	1.8	0.9	1.3	0.9	/	/	/	/	0.9	1
8	5160	29	3	0	1	1	○	○	•	•	•	○	○	/	/	/	/	6.8	1.2	1	8.7	2.1	20	1.1	-3.8	/	/	/	/	1	0.8
9	5790	29	3	3	1	1	○	○	•	•	•	○	○	/	/	/	/	5	5	1.1	1	1	1.2	1	1.6	/	/	/	/	1.2	1
10	6420	21	3	3	1	1	○	•	•	•	•	•	•	/	/	1	0.9	4.9	4.1	1	1	1	1	1	1	0.9	1.1	1	1.1	1.2	0.8

Table 8: Pareto front results (rounded) for the mechanical synchronization application.

References

[1] K. Vanherpen, A contract-based approach for multi-viewpoint consistency in the concurrent design of cyber-physical systems, Ph.D. thesis, University of Antwerp (2018).

[2] M. Thone, M. Potters, S. Baldi, Control configurations in distillation columns: A comparative study, in: 2016 European Control Conference, ECC 2016, Aalborg, Denkmark, 2016, pp. 37–42. doi:10.1109/ECC.2016.7810260.

[3] M. H. De Queiroz, J. E. Cury, Modular control of composed systems, in: Proceedings of the American Control Conference, Vol. 6, Chicago, Illinois, 2000, pp. 4051–4055. doi:10.1109/ACC.2000.876983.

[4] J. Dong, X. Yang, Q. Liu, Z. Wang, T. Wang, Design and implementation of CNC controllers using reconfigurable hardware, in: 2009 IEEE International Conference on Control and Automation, ICCA 2009, Christchurch, New Zealang, 2009, pp. 1481–1486. doi:10.1109/ICCA.2009.5410418.

[5] G. Yong, S. Yushan, M. Yufeng, W. Lei, Design of Semi Physical Motion Simulation System of Underwater Robot, in: Proceedings of the 25th chinese control conference, Harbin, 2007, pp. 1601–1604. doi:10.1109/chicc.2006.280783.

[6] C. Chin-Yin, C.-C. Cheng, Integrated design for a mechatronic feed drive system of machine tools, Proceedings, 2005 IEEE/ASME International Conference on Advanced Intelligent Mechatronics. (2005) 24–28doi:10.1109/AIM.2005.1511046.

[7] D. Roy, L. Zhang, W. Chang, S. Chakraborty, Automated Synthesis of Cyber-Physical Systems from Joint Controller / Architecture Specifications, in: 2016 Forum on specification and design languages (FDL), no. i, Bremen, Germany, 2016.

[8] Y.-P. Yang, Y.-A. Chen, Multiobjective optimization of hard disk suspension assemblies: Part II - integrated structure and control design, Computers & Structures 59 (4) (1996) 771–782. doi:10.1016/0045-7949(95)00311-8.

[9] M. G. Villarreal-Cervantes, Approximate and Widespread Pareto Solutions in the Structure-Control Design of Mechatronic Systems, Journal of Optimization Theory and Applications 173 (2) (2017) 628–657. doi:10.1007/s10957-016-1053-4.

[10] Z. Affi, B. EL-Kribi, L. Romdhane, Advanced mechatronic design using a multi-objective genetic algorithm optimization of a motor-driven four-bar system, Mechatronics 17 (9) (2007) 489–500. doi:10.1016/j.mechatronics.2007.06.003.

[11] J.-H. Park, H. Asada, Concurrent design optimization of mechanical structure and control for high speed robots, American Control Conference (1993) 2673–2679.

[12] H. Fathy, P. Papalambros, a.G. Ulsoy, D. Hrovat, Nested plant/controller optimization with application to combined passive/active automotive suspensions, Proceedings of the 2003 American Control Conference, 2003. 4 (2003) 3375–3380. doi:10.1109/ACC.2003.1244053.

[13] G. Tian, M. C. Zhou, P. Li, Disassembly Sequence Planning Considering Fuzzy Component Quality and Varying Operational Cost, IEEE Transactions on Automation Science and Engineering 15 (2) (2018) 748–760. doi:10.1109/TASE.2017.2690802.

[14] G. Tian, Y. Ren, Y. Feng, M. Zhou, H. Zhang, J. Tan, Modeling and Planning for Dual-objective Selective Disassembly Using AND/OR Graph and Discrete Artificial Bee Colony, IEEE Transactions on Industrial Informatics 3203 (c) (2018) 1–12. doi:10.1109/TII.2018.2884845.

[15] D. Li, W. Liu, J. Jiang, R. Xu, Placement optimization of actuator and sensor and decentralized adaptive fuzzy vibration control for large space intelligent truss structure, Science China Technological Sciences 54 (4) (2011) 853–861. doi:10.1007/s11431-011-4333-0.

[16] B. Xu, J. S. Jiang, Integrated optimization of structure and control for piezoelectric intelligent trusses with uncertain placement of actuators and sensors, Computational Mechanics 33 (5) (2004) 406–412. doi:10.1007/s00466-003-0541-1.

[17] H. K. Fathy, J. A. Reyer, P. Y. Papalambros, A. G. Ulsoy, On the coupling between the plant and controller optimization problems, Proceedings of the American Control Conference 3 (2001) 1864–1869. doi:10.1109/ACC.2001.946008.

[18] A. Baheri, J. Deese, C. Vermillion, Combined Plant and Controller Design Using Bayesian Optimization: A Case Study in Airborne Wind Energy Systems, Proceedings of the ASME 2017 Dynamic Systems and Control Conference (Octobe 11-13) (2017) V003T40A003. doi:10.1115/DSCC2017-5242.
URL http://proceedings.asmedigitalcollection.asme.org/proceeding.aspx?doi=10.1115/DSCC2017-5242

[19] D. Peters, K. Kurabayashi, P. Papalambos, G. Ulsoy, Co-Design of a MEMS Actuator and its Controller Using Frequency Constraints (2008) 1–7doi:10.1115/DSCC2008-2212.

[20] Y. S. Wang, Y. Wang, A gradient-based approach for optimal plant controller co-design, Proceedings of the American Control Conference 2015-July (2015) 3249–3254. doi:10.1109/ACC.2015.7171833.

[21] R. Scattolini, Architectures for distributed and hierarchical Model Predictive Control - A review (2009). doi:10.1016/j.jprocont.2009.02.003.
URL http://dx.doi.org/10.1016/j.jprocont.2009.02.003

[22] P. Massioni, M. Verhaegen, Distributed control for identical dynamically coupled systems: A decomposition approach, IEEE Transactions on Automatic Control 54 (1) (2009) 124–135. doi:10.1109/TAC.2008.2009574.

[23] I. Tomi, G. D. Halikias, Performance analysis of distributed control configurations in LQR multi-agent system designdoi:10.1109/CONTROL.2016.7737530.

[24] C. C. Wo, Z. Q. Min, Coupling & decoupling control study on aircraft (Airbus A320), 2016 8th International Conference on Modelling, Identification and Control (ICMIC) (2016) 29–36doi:10.1109/ICMIC.2016.7804160.

[25] C. Yang, Q. Huang, J. Han, Decoupling control for spatial six-degree-of-freedom electro-hydraulic parallel robot, Robotics and Computer-Integrated Manufacturing 28 (1) (2012) 14–23. doi:10.1016/j.rcim.2011.06.002.

[26] A. Maxim, D. Copot, R. De Keyser, C. M. Ionescu, An industrially relevant formulation of a distributed model predictive control algorithm based on minimal process information, Journal of Process Control 68 (2018) 240–253. doi:10.1016/j.jprocont.2018.06.004.
URL https://doi.org/10.1016/j.jprocont.2018.06.004

[27] D. E. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, Addison-Wesley Publishing Company, Inc., 1989.

[28] W. Sun, S.-f. Su, Y. Wu, J. Xia, V.-t. Nguyen, Adaptive Fuzzy Control With High-Order Barrier Lyapunov Functions for High-Order Uncertain Nonlinear Systems With Full-State Constraints, IEEE Transactions on Cybernetics PP (2019) 1–9. doi:10.1109/TCYB.2018.2890256.

[29] W. Sun, S.-f. Su, G. Dong, W. Bai, Reduced Adaptive Fuzzy Tracking Control for High-Order Stochastic Nonstrict Feedback Nonlinear System With Full-State Constraints, IEEE Transactions on Systems, Man, and Cybernetics: Systems PP (2019) 1–11. doi:10.1109/TSMC.2019.2898204.

[30] J. Xia, J. Zhang, W. Sun, B. Zhang, Z. Wang, Correspondence With Full State Constraints, IEEE Transactions on Systems, Man, and Cybernetics: Systems 49 (7) (2019) 1541–1548. doi:10.1109/TSMC.2018.2854770.

[31] B. Wu, C. Liu, X. Song, X. Wang, Design and implementation of the inverted pendulum optimal controller based on hybrid genetic algorithm, 2015 International Conference on Automation, Mechanical Control and Computational Engineering (Amcce) (2015) 623–629.

[32] M. P. Nagarkar, G. J. Vikhe Patil, Multi-objective optimization of LQR control quarter car suspension system using genetic algorithm, FME Transactions 44 (2) (2016) 187–196.

[33] H. Du, J. Lam, K. Y. Sze, Non-fragile output feedback H∞ vehicle suspension control using genetic algorithm, Engineering Applications of Artificial Intelligence 16 (7-8) (2003) 667–680. doi:10.1016/j.engappai.2003.09.008.

[34] N. R. Raju, P. L. Reddy, Optimal Tuning of Fractional Order PID Controller for Automatic Voltage Regulator System through Genetic Algorithm, International Journal of Engineering and Technology (IJET) 8 (3) (2016) 922–927.

[35] A. Baumal, J. McPhee, P. Calamai, Application of genetic algorithms to the design optimization of an active vehicle suspension system, Computer Methods in Applied Mechanics and Engineering 163 (1-4) (1998) 87–94. doi:10.1016/S0045-7825(98)00004-8.

[36] G. Duc, Designing a Low Order Robust Controller for an Active Suspension System Thank LMI, Genetic Algorithm and Gradient Search, European Journal of Control 9 (April 2002) (2003) 29–38. doi:10.3166/ejc.9.29-38.

[37] I. Robandi, K. Nishimori, R. Nishimura, N. Ishihara, Optimal feedback control design using genetic algorithm in multimachine power system, International Journal of Electrical Power and Energy Systems 23 (4) (2001) 263–271. doi:10.1016/S0142-0615(00)00062-4.
URL http://www.sciencedirect.com/science/article/pii/S0142061500000624

[38] C. Wongsathan, C. Sirima, Application of GA to design LQR controller for an inverted pendulum system, 2008 IEEE International Conference on Robotics and Biomimetics, ROBIO 2008 (2) (2008) 951–954.

[39] M. Haris, I.-U.-H. Shaikh, H. Shoaib, Genetic Algorithm Based LQR Control Of Hovercraft, 2016 International Conference on Intelligent Systems Engineering (ICISE).

[40] N. Van Oosterwyck, F. Vanbecelaere, M. Haemers, D. Ceulemans, K. Stockman, S. Derammelaere, CAD Enabled Trajectory Optimization and Accurate Motion Control for Repetitive Tasks, IEEE International Conference on Control and Automation, ICCA.

[41] M. Borairi, M. Soufian, Genetic Sensor Placement in Active Control of a Robotic Arm, 2017 10th International Conference on Developments in eSystems Engineering (DeSE) (2017) 279–284doi:10.1109/DeSE.2017.47.
URL http://ieeexplore.ieee.org/document/8285834/

[42] Q. Chen, T. Lin, H. Ren, Parameters optimization and control strategy of power train systems in hybrid hydraulic excavators, Mechatronics 56 (668) (2018) 16–25. doi:10.1016/j.mechatronics.2018.10.003.
URL https://doi.org/10.1016/j.mechatronics.2018.10.003

[43] A. E. Bryson, H. Yu-Chi, Applied Optimal Control: Optimization, Estimation, and Control, 1979.
URL http://www.amazon.com/Applied-Optimal-Control-Optimization-Estimation/dp/0470267747

[44] E. Million, The Hadamard Product (2007).

[45] C. Davis, The norm of the Schur product operation, Numerische Mathematik 4 (1) (1962) 343–344. doi:10.1007/BF01386329.

[46] R. A. Horn, C. R. Johnson, Matrix analysis, Cambridge University Press., 2012. doi:10.1017/9781139020411.

[47] S. Skogestad, I. Postlethwaite, Multivariable Feedback Control: Analysis and Design, 2nd Edition, Building Services Engineering Research and Technology (2005) 590arXiv:0470011688, doi:978-0-470-01167-6.

[48] W. Chen, J. Liang, T. Shi, Speed Synchronous Control of Multiple Permanent Magnet Synchronous Motors Based on an Improved Cross-Coupling Structuredoi:10.3390/en11020282.

[49] K. Deep, K. P. Singh, M. L. Kansal, C. Mohan, A real coded genetic algorithm for solving integer and mixed integer optimization problems, Applied Mathematics and Computation 212 (2) (2009) 505–518. doi:10.1016/j.amc.2009.02.044.
URL http://dx.doi.org/10.1016/j.amc.2009.02.044

[50] C. Scherer, Theory of robust control, Delft University of Technology (2001) 1–160.
URL http://www.imng.uni-stuttgart.de/mst/robust/RCNotes.pdf

[51] E. Silvas, T. Hofman, N. Murgovski, P. Etman, M. Steinbuch, Review of Optimization Strategies for System-Level Design in Hybrid Electric Vehicles, IEEE Transactions on Vehicular Technology (2016) 1–1doi:10.1109/TVT.2016.2547897.

[52] Frontline Systems, Optimization Problem Types (2018).
URL https://www.solver.com/nonsmooth-optimization

[53] A. L. Custódio, J. F. A. Madeira, A. I. F. Vaz, L. N. Vicente, Direct Multisearch for Multiobjective Optimization, SIAM Journal on Optimization 21 (3) (2011) 1109–1140.

[54] W. Gao, S. K. Porandla, Design optimization of a parallel hybrid electric powertrain, 2005 IEEE Vehicle Power and Propulsion Conference (2005) 6 pp.

[55] S. Kirkpatrick, C. D. J. Gelatt, M. P. Vecchi, Optimization by simulated annealing', Science, Vol. 220pp (1983) 671–680.

[56] Y. Wang, C. A. Shoemaker, A General Stochastic Algorithmic Framework for Minimizing Expensive Black Box Objective Functions Based on Surrogate Models and Sensitivity AnalysisarXiv:1410.6271.
URL http://arxiv.org/abs/1410.6271

[57] M. Reza Bonyadi, Z. Michalewicz, Particle Swarm Optimization for Single Objective Continuous Space Problems: A Review, Evolutionary Computation (MIT) 25 (1) (2017) 1–54.

[58] The MathWorks Inc., MATLAB Global Optimization Toolbox User's Guide (Release 2018b) (2018).

[59] T. Singh, J. Swevers, G. Pipeleers, Concurrent H2 /H∞ feedback control design with optimal sensor and actuator selection, Proceedings - 2018 IEEE 15th International Workshop on Advanced Motion Control, AMC 2018 (2018) 223–228doi:10.1109/AMC.2019.8371092.

[60] S. F. Hwang, R. S. He, A hybrid real-parameter genetic algorithm for function optimization, Advanced Engineering Informatics 20 (1) (2006) 7–21. doi:10.1016/j.aei.2005.09.001.

[61] K. O. Stanley, R. Miikkulainen, Evolving neural networks through augmenting topologies, Evolutionary Computation 10 (2) (2002) 99–127.

[62] K. Deb, Multi-Objective Optimization Using Evolutionary Algorithms: An Introduction, Wiley-Interscience Series in Systems and Optimizationdoi:10.1001/jama.1943.02840160014004.

[63] Y. Censor, Pareto optimality in multiobjective problems, Vol. 4, 1977. doi:10.1007/BF01442131.

[64] M. Kishnani, S. Pareek, R. Gupta, Optimal tuning of DC motor via simulated annealing, 2014 International Conference on Advances in Engineering & Technology Research (ICAETR - 2014) (2014) 1–5doi:10.1109/ICAETR.2014.7012928.
URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=7012928

[65] R. M. Murray, Optimization-Based Control Control and Dynamical Systems California Institute of Technology (February) (2009) 0–16.