Title

A Mixed Discrete-Continuous Optimization Scheme for Cyber-Physical System Architecture Exploration

ISBN

9781467383882

Authors

Finn, John
Nuzzo, Pierluigi
Sangiovanni-Vincentelli, Alberto

Publication Date

2015-11-01

DOI

10.1109/iccad.2015.7372573

Peer reviewed

A Mixed Discrete-Continuous Optimization Scheme for Cyber-Physical System Architecture Exploration

John Finn, Pierluigi Nuzzo, Alberto Sangiovanni-Vincentelli
EECS Department, University of California at Berkeley, email: {jbfinn,nuzzo,alberto}@eecs.berkeley.edu

***Abstract*—We propose a methodology for architecture exploration for Cyber-Physical Systems (CPS) based on an iterative, optimization-based approach, where a discrete architecture selection engine is placed in a loop with a continuous sizing engine. The discrete optimization routine proposes a candidate architecture to the sizing engine. The sizing routine optimizes over the continuous parameters using simulation to evaluate the physical models and to monitor the requirements. To decrease the number of simulations, we show how balance equations and conservation laws can be leveraged to prune the discrete space, thus achieving significant reduction in the overall runtime. We demonstrate the effectiveness of our methodology on an industrial case study, namely an aircraft environmental control system, showing more than one order of magnitude reduction in optimization time.**

I. Introduction

The recent advancements in embedded processor technology have pushed system designers to replace several mechanical, pneumatic or hydraulic components with electronic components interacting with the physical world via sensors and actuators. This trend towards higher "electrification," particularly evident in the automotive and aerospace domains, has enabled designers to drastically reduce system cost and weight, while increasing performance and energy efficiency [1], finally leading to the realization of complex cyber-physical systems (CPS) that are characterized by the tight integration of physical processes with the "cyber" world of computation, communication and control.

By pushing such a tight integration to an increasingly larger scale, CPS are capable of interconnecting the world around us and making it "smarter," thus offering very promising solutions to societal and environmental problems. However, as system complexity and heterogeneity increase, current design practices no longer scale to meet the demanding cost, performance, or time-to-market constraints of these systems. A major bottleneck is the inability to foresee the impact of design decisions made early in the design process on the final implementation because of the lack of comprehensive frameworks for scalable, system-level architecture exploration under a set of heterogenous, possibly conflicting constraints, and with tight safety, reliability and performance guarantees [2].

The rigorous definition of a cyber-physical system architecture, including the number and type of system components, their dimensions, and their interconnections, tends to generate intractable mathematical problems. Often, designers are expected to simultaneously reason about many discrete alternatives, in conjunction with a large, continuous design space, which requires expensive, high-fidelity simulations to achieve the required accuracy in performance and cost estimations. System-level design exploration is the domain of experienced architects, often relying on their accrued knowledge and a set of heuristic evaluations to take risky decisions. In fact, the result of *ad hoc* design practices is often at the origin of unsustainable delays, lengthy redesign cycles and severe financial consequences.

This paper addresses the challenges above by proposing a rigorous and efficient optimization scheme for the exploration of large, mixed discrete-continuous design spaces. Our contribution is threefold:

- We propose a general formulation of the architecture design problem in terms of a hybrid discrete-continuous optimization problem, where the continuous design space can be captured by nonlinear, non-convex expressions, which may be, in general, not available in closed analytic form.
- We propose an efficient, iterative algorithm that can approximate the solution of the above optimization problem by leveraging a discrete architecture selection engine placed in a loop with a continuous sizing engine. The discrete optimization routine proposes a candidate architecture to the sizing engine. The sizing routine optimizes over the continuous parameters using simulation to evaluate black-box physical models and monitor the design requirements. To decrease the number of simulations, fundamental axioms based on conservation laws are leveraged to prune the discrete space at each iteration, thus achieving significant reduction in the overall runtime and correctness of the final result.
- We demonstrate the effectiveness of our methodology on an industrial case study: an aircraft environmental control system (ECS). Our results show more than one order of magnitude reduction in optimization time with respect to full enumeration of optimal candidate configurations from the discrete selection engine.

The remainder of this paper is organized as follows. After an overview of related works in Sec. II, we formulate the CPS architecture design problem in Sec. III and present our optimization-based methodology in Sec. IV. Sec. V introduces the aircraft environmental control system case study and summarizes our optimization results. Finally, concluding remarks follow in Sec. VI.

II. Related Work

Our work builds upon a series of recent results advocating a formalization of the design process and the support of design automation to address the challenges associated with the next generation of CPS [3]–[6]. Specifically, we have recently proposed a methodology for system-level design of an aircraft electric power system, which relies on a formalization of the design requirements using contracts expressed in terms of temporal logic constructs and mixed integer-linear constraints [3]. While architecture selection was previously based on steady-state, discrete system abstractions [3], [6], the focus of this paper is instead on providing an efficient procedure for architectural exploration that can also capture continuous dynamics in the plant and the controller. In addition

This work was partially supported by IBM and United Technologies Corporation (UTC) via the iCyPhy consortium, and by TerraSwarm, one of six centers of STARnet, a Semiconductor Research Corporation program sponsored by MARCO and DARPA.

to determining the system structure, we also support component sizing and continuous-valued decision variables. The approach proposed in this paper is therefore complementary to our previous one [3], [6], and can be incorporated in our methodology.

Very few papers in the literature address the class of nonlinearly mixed discrete-continuous optimization-based design problems, such as the one in this paper. A possible approach to these difficult problems is to formulate simpler, *monolithic* problem instances using a combination of approximations [7], such as continuous relaxations of discrete constraints followed by rounding methods, or analytical (e.g. linear, convex) approximations of nonlinear and dynamic constraints [8], [9] that can be handled by mixed integer-linear or semidefinite programming solvers. By building on our previous results [3], [6], we propose instead an iterative combination of two solvers that efficiently breaks a complex task into two simpler ones at the discrete and continuous levels. Differently than previous methods [7], we make no specific assumptions on the structure of the continuous design space (e.g. monotonicity of cost or constraints, knowledge of their derivatives) and the differential equations of the underlying physical system. We are then able to support black-box models that can be accurately evaluated by simulation, while leveraging fundamental axioms based on conservation laws to effectively discard conflicting discrete design choices when a certain set of continuous constraints is violated. This iterative scheme allows smoothly integrating heterogeneous modeling formalisms and distinct, possibly domain-specific, optimization and simulation frameworks.

A set of model-based approaches to aircraft ECS design have also appeared over the years, which are related to the design example in this paper. However, they mostly entail *ad hoc* procedures, characterized by a poor automation level and largely neglect the interactions between architecture selection and the control algorithm. Some of these works seek to optimize the heat exchanger size for fuel efficiency, but pay little attention to control design [10]; other approaches only focus on the design of the control protocol, by either evaluating the tradeoffs between a finite number of candidate ECS architectures with respect to temperature control and efficiency [11], or by exploring control strategies based on fuzzy logic [12]. A correct-by-construction supervisory control protocol synthesized from linear temporal logic (LTL) requirements, along with a state-estimation algorithm has also been recently reported [13]; however, the architecture sizing problem is not taken into account. More closely related to our work is the procedure to co-design the heat exchanger (a plant component) along with the bypass ratio (a control variable) to optimize the overall efficiency and volume [14]. However, as for the majority of the ECS design approaches above, the proposed technique heavily relies on domain-specific knowledge to take important decisions and lacks a rigorous formalization of the design requirements.

III. Problem Formulation

In this section, we formulate the CPS architecture selection problem as a mixed discrete-continuous optimization problem. The discrete decision variables encode component selection from a pre-characterized library, while the continuous variables encode sizing alternatives for the selected components. Our goal is to find an optimal assignment for all the design variables to satisfy a set of composition constraints and design requirements while minimizing a cost function. To support both continuous and discrete models within a unifying framework, we need to enrich our previous formulation [6] as follows.

A. Component Library and Architecture

We assume that a design is assembled out of a *library* (collection) $\mathcal{L}$ of *components* and *contracts*. Each component is associated with a *type*, defining its functionality (role or task) in a system, and a tuple of *attributes* including: terminals, variables, parameters, behavioral and extra-functional models. A *behavioral model* is represented, in general, by a set of differential algebraic equations (DAEs) $\mathcal{F}(u, x, y, \kappa) = 0$ determining the values of the component *output variables* y and *internal variables* x (including state variables) over time given the values of the *input variables* u and the *configuration parameters* κ. We also use $[\![\mathcal{F}]\!]$ to denote the set of behaviors of a component, i.e. the solutions of $\mathcal{F}(.) = 0$, and the notation $x(t)$ to emphasize that these solutions are traces over time, $t \in \mathbb{R}_{\geq 0}$. Moreover, we assume vectors κ of the form (d, p) where d encodes a set of (finite) discrete design choices, while p represents a set of continuous design parameters. *Extra-functional models* are compact maps that allow directly estimating non-functional properties of a component, such as energy, performance, and cost, as a function of (some of) the component variables without expensive evaluations of the full behavioral model.

Components can be connected via *terminals* and terminal *variables*. Terminals can be *logical* (i.e. input or output) or *physical* (e.g. hydraulic, thermal, electrical, mechanical) in nature. *Input* terminals are used when a signal or the value of a variable are imposed to the component by the external environment; *output* terminals are used when a signal or a value of a variable are imposed by the component. Physical terminals are instead associated with *effort* and *flow* variables subject to conservation laws. These variables are shared, rather than imposed, in an interconnection. Examples of effort and flow variables include, respectively, voltage and current for an electrical terminal, pressure and mass flow for a hydraulic terminal, and temperature and heat flow for a thermal terminal. Examples of inputs and outputs include variables that are, respectively, sensed and set by a controller, for example, a valve controller's mass flow and valve opening. While physical terminals and variables cannot be univocally classified as inputs and outputs *a priori*, they can still be labelled as such *a posteriori*, based on the specific function a component performs within the system and the overall interconnection structure.

Components are associated to contracts, offering an abstraction for their behaviors. A *contract* $\mathcal{C}$ for a component M is a triple (T, A, G), where T is the set of component variables (including parameters), and A and G are assertions representing sets of behaviors over T. A represents the *assumptions* that M makes on its environment and G represents the *guarantees* provided by M under the environment assumptions. A component M *satisfies (implements)* a contract $\mathcal{C}$, i.e. $M \models \mathcal{C}$, whenever M and $\mathcal{C}$ are defined over the same set of variables and all the behaviors of M satisfy the guarantees of $\mathcal{C}$ in the context of the assumptions, i.e. when $[\![\mathcal{F}_M]\!] \cap A \subseteq G$. We say that a component N is a legal *environment* of $\mathcal{C}$, i.e. $N \models_E \mathcal{C}$, whenever N and $\mathcal{C}$ have the same variables and $[\![\mathcal{F}_N]\!] \subseteq A$. Assumptions and guarantees can be concretely expressed by DAE-based behavioral models or signal temporal logic constructs. Contracts and their algebra can then be used to formalize system-level requirements and verify the correctness of the compositions and refinements between components [3].

We then define a system *architecture* as a graph $\mathcal{G} = (V, E)$, where V is a set of nodes, while an edge $e_{ij} \in E$ represents the interconnection between nodes v_i and v_j ($i, j \in \{1, \ldots, |V|\}$, $|V|$ being the cardinality of V). Each node and edge in an abstract architecture can be mapped to a library

element that *implements* it. Both nodes and edges can then be labelled with the attributes of the associated library elements. A *template* is an architecture in which the number and types of nodes are fixed, while the interconnection structure is variable and can be reconfigured. In a template, edges can be represented by a set of Boolean variables $E = \{e_{ij}\}$, each denoting the presence or absence of an interconnection. An assignment over E defines an architecture *topology*. We use the edge variables E of a template to formulate a *topology* selection problem and the configuration parameters $K = \{\kappa_i\}_{i \in V \cup E}$ to formulate a *sizing* problem, where V and E are the sets of nodes and edges of the template.

While the topology selection problem was the objective of previous work [6], the focus of this paper is on the sizing problem. In particular, given a topology $\mathcal{T} = (V, E)$ and a library $\mathcal{L}$, we aim to map a set of design requirements into a selection of library components that implement each node and edge in $\mathcal{T}$, while minimizing an overall cost. In our contract-based framework, the architecture is specified as an aggregation of contracts from the library $\mathcal{L}$, composed according to the interconnection structure in $\mathcal{T}$, which we denote as *architecture contract* $\mathcal{C}_\mathcal{T}$. The top-level requirements are specified by a system-level *application contract* $\mathcal{C}_\mathcal{A}$. The refinement (mapping) between $\mathcal{C}_\mathcal{A}$ and $\mathcal{C}_\mathcal{T}$ is then modeled as the *vertical contract* $\mathcal{C}_\mathcal{A} \wedge \mathcal{C}_\mathcal{T}$ given by the *conjunction* of the architecture and application contracts [3]. We are interested in an optimal parameter configuration κ^* subject to the constraint that $\mathcal{C}_\mathcal{A} \wedge \mathcal{C}_\mathcal{T}$ is *consistent*, i.e. there exists an implementation satisfying both the guarantees of $\mathcal{C}_\mathcal{A}$ and $\mathcal{C}_\mathcal{T}$ in the context of their assumptions as further detailed below.

B. System Requirements and Mapping

Our framework supports the specification of contracts using algebraic constraints on integer and real variables, e.g. to capture steady-state requirements, as well as signal temporal logic constructs, to capture more complex real-time performance requirements, including transient behaviors, which can be monitored during simulation [3]. In the following, we discuss an instance of the mixed discrete-continuous architecture selection problem in the case of steady-state requirements expressed as inequality constraints on the system variables.

We cast the architecture sizing problem as an optimization problem as follows. Let $s_{ij} = 1$ if a node or an edge $i \in V \cup E$ of $\mathcal{T}$ is implemented by a library component $j \in \mathcal{L}_i$, $\mathcal{L}_i$ being the set of library elements having the same type of element i, and 0 otherwise. Let S' be the set of all the decision variables s_{ij}. We assume that behavioral models can be composed based on the interconnection structure E in $\mathcal{T}$ via the composition operator $\wedge_E$, implemented by the conjunction of the DAEs, after extending the (local) variable alphabet of each component to a (global) common set of symbols [3]. The guarantees of the architecture contract $\mathcal{C}_\mathcal{T}$ can then be expressed by

$$\mathcal{F}(u, x, y, s, p) = \bigwedge_{i \in V \cup E, j \in \mathcal{L}_i}{}_E \; s_{ij} \mathcal{F}_j(u_j, x_j, y_j, d_j, p_j),$$

where x, y, u, s, and p are valuations over the sets $X = \cup_{i \in V \cup E} X_i$, $Y = \cup_{i \in V \cup E} Y_i$, $U = \cup_{i \in V \cup E} U_i \setminus Y$, $P = \cup_{i \in V \cup E} P_i$, $S = \cup_{i \in V \cup E} D_i \cup S'$ representing, respectively, the sets of all the internal variables, output variables, input variables, continuous parameters and discrete parameters in the system. In the expressions above, we have used a capital letter, e.g. X, to denote a set of variables, to distinguish it from a valuation x over X, or a trace $x(t)$. A first set of optimization constraints in our problem will then come from the behaviors guaranteed by the system, by requiring that the DAEs $\mathcal{F}(u, x, y, s, p, x_0) = 0$ hold $\forall u(t) \in \mathcal{U}$ and $\forall x_0 \in \mathcal{X}_0$. This constraint emphasizes the dependence on the initial condition x_0 for the state variables in $\mathcal{F}$, where $\mathcal{U}$ is the set of all admissible input behaviors (traces) for the system, and $\mathcal{X}_0$ the set of all admissible initial conditions. These legal sets of inputs and initial conditions encode the assumptions of both $\mathcal{C}_\mathcal{T}$ and $\mathcal{C}_\mathcal{A}$.

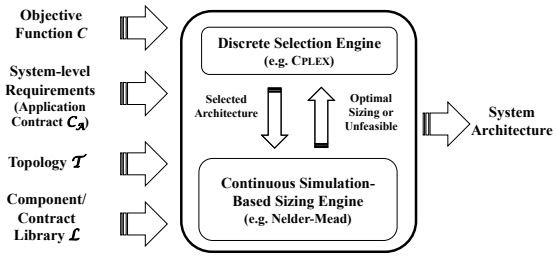

Figure 1: Architecture selection methodology.

Next, we partition the set of optimization constraints $\mathcal{R}$ into two subsets. We denote $\mathcal{R}_f$ as the set of *functional constraints*, which can only be checked by evaluating the system behavioral model, i.e. $r_k(s, p, x_\infty, y_\infty) \leq 0 \; \forall k \in \{1, \ldots, |\mathcal{R}_f|\}$. We instead denote $\mathcal{R}_{ef}$ as the set of *extra-functional constraints*, which can be checked without directly evaluating $\mathcal{F}(.) = 0$, i.e. $r_m(s, p, x_\infty, y_\infty) \leq 0 \; \forall m \in \{1, \ldots, |\mathcal{R}_{ef}|\}$. We use the notation x_∞ to denote the steady-state value of the variables X. Moreover, we assume that the domain $\mathcal{P}$ for the continuous parameters P is selected such that, for any legal discrete configuration $s \in \mathcal{S} = \{0, 1\}^{|S|}$, the steady state values $x_\infty(s, p)$ and $y_\infty(s, p)$, which are generally functions of p and s, are achieved $\forall u(t) \in \mathcal{U}$ and $\forall x_0 \in \mathcal{X}_0$. $\mathcal{R}_f$ and $\mathcal{R}_{ef}$ can both originate from the "global," system-level requirements in $\mathcal{C}_\mathcal{A}$ as well as "local," component-level requirements in $\mathcal{C}_\mathcal{T}$. In fact, the behavior of a component in $\mathcal{T}$ is guaranteed, according to its behavioral model, only under a set of constraints expressing its assumptions. Some of these constraints, e.g. including bounds on the magnitude of flows (f_{max}) and efforts (h_{max}), must be discharged by the guarantees of other components and explicitly accounted for in $\mathcal{R}$. For example, the gauge of a wire limits the amount of current that can safely pass or the material of an air duct limits the maximum temperature of the air passing through it. However, the actual values of the current or temperature can only be determined after evaluating the entire system model.

Altogether, given an objective function $C(s, p)$, the architecture sizing problem generates the following optimization problem, which searches over all the architecture configurations that make the vertical contract $\mathcal{C}_\mathcal{A} \wedge \mathcal{C}_\mathcal{T}$ consistent:

$$\min_{p \in \mathcal{P}, s \in \mathcal{S}} \quad C(s, p) \tag{1a}$$

$$\text{s. t.} \quad \mathcal{F}(u, x, y, s, p, x_0) = 0, \forall u(t) \in \mathcal{U}, \forall x_0 \in \mathcal{X}_0 \tag{1b}$$

$$r_k(s, p, x_\infty, y_\infty) \leq 0, \quad \forall k \in \{1, \ldots, |\mathcal{R}_f|\} \tag{1c}$$

$$r_m(s, p, x_\infty, y_\infty) \leq 0, \quad \forall m \in \{1, \ldots, |\mathcal{R}_{ef}|\} \tag{1d}$$

$$\sum_{j \in \mathcal{L}_i} s_{ij} = 1, \quad \forall i \in V \cup E, \tag{1e}$$

and where the last cardinality constraint ensures that one (and only one) library element is selected to implement each component in the topology.

IV. Hybrid Optimization Scheme

To find a solution for problem (1a)-(1e), we propose the iterative scheme illustrated in Fig. 1. Both the library $\mathcal{L}$ and the topology template $\mathcal{T}$ are provided as inputs to our

Algorithm 1 *Hybrid CPS Architecture Exploration*

Input: Library $\mathcal{L}$, topology $\mathcal{T}$, constraints $\mathcal{R}$, cost function C
Output: Best cost C_{best}, selection s_{best}, parameters p_{best}

$\hat{\mathcal{R}} \leftarrow \{\}$; $C_{best} \leftarrow +\infty$; $s_{best} \leftarrow \{\}$; $p_{best} \leftarrow \{\}$; $k \leftarrow 1$;
while true **do**
  $(s_k^*, p_k^*, C_k^*) \leftarrow$ OPTSEL($\mathcal{L}$, $\mathcal{T}$, $\mathcal{R}$, $\hat{\mathcal{R}}$, C);
  if $(s_k^*, p_k^*) = \{\}$ and $C_{best} = +\infty$ **then**
    return `infeasible`;
  else if $C_k^* > C_{best}$ or $(s_k^*, p_k^*) = \{\}$ **then**
    return (s_{best}, p_{best}, C_{best});
  $(p_k^{**}, C_k^{**}) \leftarrow$ OPTSIZE($\mathcal{L}$, $\mathcal{T}$, $\mathcal{R}$, C, s_k^*);
  if $C_k^{**} \leq C_{best}$ **then**
    $C_{best} \leftarrow C_k^{**}$; $s_{best} \leftarrow s_k^*$; $p_{best} \leftarrow p_k^{**}$;
  $\hat{\mathcal{R}} = \hat{\mathcal{R}} \cup$ LEARNCONS($s_k^*, p_k^{**}, \mathcal{L}, \mathcal{T}, \mathcal{R}$);
  $k \leftarrow k + 1$;

process together with the application contract $\mathcal{C}_\mathcal{A}$ and the objective function. The optimization strategy consists of a discrete selection engine placed in a loop with a continuous sizing engine. The output is an optimized system architecture $\mathcal{T}^*$.

Let $P_F(p, s)$ denote problem (1a)-(1e) and $P_{EF}(p, s)$ denote the *extra-functional problem* associated with $P_F(p, s)$ and consisting of just the objective function (1a) and constraints (1d) and (1e). Let C_{best} be updated, after each iteration, with the best achieved cost value. In our scheme, the discrete optimization routine OPTSEL solves a version of the extra-functional problem $P_{EF}(p, s)$ at each iteration by proposing a candidate selection (s^*, p^*) to the sizing routine OPTSIZE. Based on the nature of the cost function and the constraints, $P_{EF}(p, s)$ can be a mixed integer-linear (or integer-quadratic) program that can be efficiently solved without evaluating the system dynamics. Resting on this candidate selection, OPTSIZE attempts to find an optimal assignment p^{**} by solving the full problem $P_F(p, s^*)$ with all the design constraints. Both the cost function and the constraints are evaluated by simulating the hybrid system model. If OPTSIZE achieves a better cost solution, C_{best} is accordingly updated. Otherwise, if the newly computed cost is not optimal, or the sizing engine reports `infeasible`, the discrete engine is queried for another candidate selection. In every case, the current value s^* is excluded by the search space by augmenting $P_{EF}(p, s)$ with additional constraints. The process repeats until an optimal, feasible solution is found or OPTSEL terminates with `infeasible`. We implemented OPTSIZE using a version of the Nelder-Mead (NM) method [15], which is particularly suitable to approximate the solution of optimization problems with nonlinear, black-box constraints for which derivatives are not available and when decreasing the number of cost evaluations is more critical than achieving high accuracy on the final solution. However, the NM method can be replaced by any other method for black-box optimization.

The iterative process described above can be very inefficient if only the current discrete assignment is eliminated at each iteration, especially when simulation runs are expensive or the discrete design space is extremely large. Therefore, we develop a LEARNCONS function to help prune the discrete design space with a set of new constraints $\hat{\mathcal{R}}$ that eliminate more discrete assignments based on the gathered insight from simulations. More formally, our procedure is summarized in Algorithm 1. Let C_k^* represent the cost after solving $P_{EF}^k(p, s)$, at the kth iteration, achieved by the assignment (s_k^*, p_k^*). Similarly, let C_k^{**} denote the optimal cost value achieved by the sizing engine for the candidate selection s_k^* and the optimal point p_k^{**}. Since the current discrete variable assignment s_k^* is eliminated by the search space at each iteration, the sequence of costs C_k^* is finite, which guarantees termination, and is monotonically increasing with k, i.e. $C_k^* \geq C_{k-1}^*$. Moreover, by our assumptions, C_k^{**} can only be higher than or equal to C_k^* since OPTSIZE solves a problem that is more constrained than the one solved by OPTSEL. Therefore, Algorithm 1 can terminate as soon as $C_k^* > C_{best}$ is found without having to visit all the discrete combinations.

To further reduce the number of iterations and simulation runs, LEARNCONS implements a strategy that can leverage the results of the current optimization run and the underlying conservation laws enforced by the topology to prune a larger portion of the discrete search space whenever there is a violation of certain categories of functional constraints. To illustrate the strategy, let $\mathcal{R}'_f \subseteq \mathcal{R}_f$ be a subset of functional constraints enforcing bounds on flow or effort variables in the system. Intuitively, if component A's flow constraint in $\mathcal{R}'_f$ is violated, meaning that the current (simulated) flow through the terminals of A exceeds its upper bound, we can use this information to eliminate all the discrete configurations whose upper bound on the allowed flow is smaller than the current value for A, thus enforcing an increase in the maximum allowed flow at the next sizing iteration. Since A is the "bottleneck" in the current configuration, increasing the maximum flow allowed in any other component in series with A will not help push the search towards a feasible point. However, if component A is in parallel with another component B of the same type, increasing the maximum allowed flow through B may alleviate the burden posed on A and move faster towards a feasible sizing. In this case, we require the maximum flow requirement should be increased for at least one of the components in parallel in the next iteration.

Formally, let f_i^k be the value of the flow through the terminal of component $i \in V \cup E$ of topology $\mathcal{T}$ at iteration k, and let $f_{j,max}$ be the maximum allowed flow through component $j \in \mathcal{L}_i$ of the same type as i. Let $F_i \subseteq V \cup E$ be the set of components in parallel with component i, including i, in $\mathcal{T}$. F_i can be easily determined from the topology graph, or from the balance equations at the interconnection between terminals. For example, from a balance equation of the form $f_1 + f_2 - f_3 = 0$, we can directly infer that components 1 and 2 are in parallel, and both are in series with component 3. Then, there is a flow constraint violation at component i if we find

$$\sum_{j \in \mathcal{L}_i} s_{ij}^{k*} f_{j,max} < f_i^k, \tag{2}$$

where $s_{i,j}^{k*}$ is the assignment decided by OPTSEL at iteration k. Next, let $W_k \subseteq V \cup E$ be the set of components such that (2) holds at iteration k. We can then augment the number of constraints in the optimization problem as follows:

$$\hat{\mathcal{R}}^{k+1} = \hat{\mathcal{R}}^k \bigcup_{i \in W_k} \left\{ \sum_{l \in F_i} \sum_{j \in \mathcal{L}_l} f_{j,max}(s_{lj}^{k*} - s_{lj}^{k+1}) < 0 \right\}, \tag{3}$$

where we enforce an increase in the maximum allowed flow for at least one component in F_i for all $i \in W^k$. Finally, by exploiting the duality between flow and effort variables, we handle violations in efforts in a similar way by replacing parallel components with series components and *vice versa*.

As discussed above, the violation of a constraint involving a set of flow (or effort) variables J will mostly impact the selection of those components that enter the same set of balance equations as the variables in J. However, it is possible to generalize the above strategy to any constraint that is *monotone*, as defined below.

Definition IV.1 (Monotone Constraint)**.** *A constraint* $(r(.) \leq 0) \in \mathcal{R}_f$ *in Problem* (1a)-(1e) *is* monotone *if there exists a*

partial order $\preccurlyeq$ on the set of legal assignments $\mathcal{S}' \subset \mathcal{S}$ such that, if $r(.) \leq 0$ is violated for an assignment $s \in \mathcal{S}'$ then $r(.) \leq 0$ is also violated for any other assignment $s' \in \mathcal{S}'$ such that $s' \preccurlyeq s$.

Therefore, if a monotone constraint is violated for a configuration of library components s_k^*, we are allowed to disregard all the selections of elements in $\mathcal{L}$ that are dominated by s_k^* without the risk of cutting feasible portions from the search space. Altogether, the arguments offered above allow us to conclude that Algorithm 1 terminates and does not miss optimal points by disregarding feasible space portions. We can then state the following theorem on its correctness.

Theorem IV.2. *Let the following hold for Algorithm 1: (i)* OPTSEL *is sound and complete on any instance of the extra-functional problem $P_{EF}^k(s,p)$, i.e. it either finds the optimum or correctly concludes with* `infeasible`*; (ii)* OPTSIZE *is sound and complete on any instance of the problem $P_F(s_k^*, p)$; (iii)* LEARNCONS *augments the set of constraints $\hat{\mathcal{R}}^k$ by adding constraints that eliminate the current assignment s_k^* and, if any monotone constraints is violated, any other assignment that is dominated by s_k^*. Then, Algorithm 1 terminates and is sound and complete.* □

Unlike previously proposed monolithic optimization schemes, the above decomposition does not rely on constraint relaxations or approximation techniques. While it can be used, in general, as a heuristic method, stronger guarantees can be provided under the assumptions of the above theorem. The full enumeration of all the discrete choices may still be necessary in the worst case; however, in practice, we observed substantial reductions in the search space and in the overall execution time as reported in the following section. A rigorous investigation of the complexity of the proposed algorithm will be object of future work.

V. APPLICATION TO AIRCRAFT ENVIRONMENTAL CONTROL SYSTEM DESIGN

An aircraft Environmental Control System (ECS) is responsible for maintaining a comfortable cabin temperature, pressure and freshness for all passengers and crew [16]. To illustrate our methodology, we consider a simplified architecture, shown in Fig. 2, inspired by an industrial patent [17]. High temperature (T_e) and pressure (P_e) bleed air from the engine enters the ECS through Valve 1, which regulates the mass flow rate, and is sent to the Heat Exchanger for cooling. The Heat Exchanger cools the bleed air using ambient air with a lower temperature T_a and a controlled flow rate F_a from the RAM Air door. The cooled air from the Heat Exchanger is mixed with bypass air, which is controlled by Valve 2, and recirculated air in the Mixer. The output of the Mixer is delivered to the cabin. Control of the bypass valve and the RAM air allow the ECS to respond to disturbances and set point changes in a timely manner. Finally, air ducts connect all the components to transport gaseous air from one to another. In such a multi-physics system, the flow variables are represented by mass flow and heat flow rates, while the effort variables are temperature and pressure.

In ECS design, critical parameters are the individual duct size and material, the heat exchanger size (length, area and tube diameter) and material, the valve positions, and the RAM air flow rate. Ducts are typically selected "off-the-shelf" according to a catalog of existing choices because customization is too expensive. In contrast, optimizing the heat exchanger size, which varies on a continuous spectrum, is necessary to meet the unique requirements of a particular application or aircraft. The control variables, such as the valve positions and RAM air flow rate, can also be regulated within a continuous, predefined

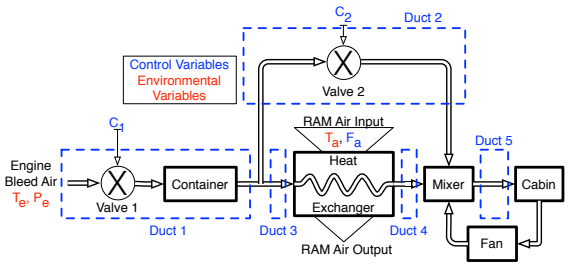

Figure 2: Example environmental control system.

range. In this scenario, a possible bottleneck stems from the need to accurately capture the coupling between duct sizing and control. Duct size, a discrete decision variable, limits the amount of allowed flow, which directly affects valve control, a continuous variable. We start our analysis by providing details on the components and contracts in our library and the models used to represent them.

A. Component and Contract Library

In the following, a subscript i denotes an input variable while o denotes an output. As mentioned in Sec. III-A, physical terminals and variables cannot be univocally classified, in general, as inputs and outputs. To do so, we consider the function associated with an ECS architecture, i.e. to condition and transport air from the engine to the cabin. Therefore, as represented by the arrows in Fig. 2, input and output terminals are labelled according to the direction of the air flow. Behavioral models capture both the *thermal* and *hydraulic viewpoints* (aspects) of each component.

Valve: The valve controls the mass flow rate through it by restricting the size of its opening, which is regulated by the valve coefficient C. The temperature of the air leaving the valve equals the temperature of the one entering the valve, i.e. $T = T_i = T_o$. The relationship between the mass flow rate F and the pressure P_i and P_o at the terminals is described by the following function:

$$F = \begin{cases} 6.67 \cdot 10^{-4} C P_i \sqrt{\frac{1}{T}} & P_o < \frac{1}{2} P_i \\ 4.72 \cdot 10^{-4} C (P_i + 2P_o) \sqrt{\frac{1}{T}\left(1 - P_o/P_i\right)} & \frac{1}{2} P_i \leq P_o < P_i \\ 0 & P_o \geq P_i \end{cases} \tag{4}$$

accounting for the absence of reverse flow and the case of choked flow. Since no mass accumulates in the valve, $F = F_i = -F_o$ hold. Finally, based on the valve contract, the behaviors above are guaranteed under the assumption that the valve coefficient satisfies $0 \leq C \leq C_{max}$.

Container: The container is characterized by a fixed volume $V = 0.0049$ m^3. The state variables describing its behaviors are the mass m [kg] of the internal air and the corresponding thermal energy Q_c [J]. The input and output terminals have the same pressure, i.e. $P = P_i = P_o$. The Ideal Gas Law relates the pressure, output temperature, mass and volume at each time instant as $P = mT_oR/V$, where R = 287.058 J/(kgK) is the gas constant for air. Similarly, the thermal energy is related to the mass and output temperature by $Q_c = C_{air} m T_o$, where C_{air} = 1003.5 J/(kgK) is the specific heat of air. The dynamic equations for the mass and thermal energy in terms of the incoming and outgoing mass flow rates and temperatures are reported below:

$$\frac{\mathrm{d}m}{\mathrm{d}t} = F_i + F_o \tag{5}$$

$$\frac{\mathrm{d}Q_c}{\mathrm{d}t} = C_{air}(F_i T_i + F_o T_o). \tag{6}$$

Heat Exchanger: In a heat exchanger two flows with different temperature and rates are allowed to transfer energy. We denote these two flows as the hot (subscript h) and cold sides (subscript c). As mentioned above, the parameters to optimize for are the cross-sectional area A_{HX} [m^2], length L_{HX} [m] and tube diameter D_{HX} [m]. The overall heat transfer area is a function of the number n and length L_{HX} of the tubes as well as the cross-sectional area A_{HX}. Since we assume no accumulation of mass, $F_{h,i} + F_{h,o} = 0$ and $F_{c,i} + F_{c,o} = 0$ hold. The available area for heat transfer is then $A_{HT} = n\pi D_{HX} L_{HX}$, where n is the number of tubes. The heat exchanger dynamics follow the equations below:

$$\frac{\mathrm{d}T_{h,o}}{\mathrm{d}t} + \frac{UA_{HT}}{M_{HX}C_m}(T_{h,o} - T_{hs}) = 0 \tag{7}$$

$$\frac{\mathrm{d}T_{c,o}}{\mathrm{d}t} + \frac{UA_{HT}}{M_{HX}C_m}(T_{c,o} - T_{cs}) = 0, \tag{8}$$

where M_{HX} [kg] is the mass of the heat exchanger and C_m [J/(kgK)] is the specific heat of the metal separating the two flows. T_{hs} and T_{cs} are calculated from the input temperature and mass flow rates on both the cold and hot sides based on the exchanger *effectiveness* ϵ [18], and U [W/(m^{2}K)] is the heat transfer coefficient of the heat exchanger. U is a function of the input temperatures and flow rates and is linearly interpolated from experimental data. Lastly, the pressure drop across the hot side of the heat exchanger is given as a function of the *headloss* [19] h_l and air density ρ as $\Delta P_{HX} = P_{h,i} - P_{h,o} = h_l \rho$. The headloss h_l can be obtained as

$$h_l = \frac{f L_{HX} V_{h,i}^2}{2 D_{HX}}, \tag{9}$$

where f is the friction coefficient and $V_{h,i}$ is the velocity [m/s] of the hot inflow. The velocity is a function of the mass input flow rate ($F_{h,i}$), volumetric flow rate ($Q_{h,i}$) and air density, while the friction coefficient is a function of the Reynold's number. An exhaustive description of the heat exchanger behavioral model is out of the scope of this paper. We conclude here by pointing out that the specific heat, mass and air density parameters of the heat exchanger model also depend on the material used to build it. In our library, a heat exchanger can be realized using either aluminum or steel. The set of parameters related to the two materials, together with their cost and weight model, are listed in rows 1 and 2, respectively, of Tab. I.

Mixer: The mixer allows multiple input flows to be mixed into a single output flow. All terminals (ports) have the same pressure ($P_o = P_{i,k}, \forall k$). Flows and thermal energies are instead balanced as follows

$$F_o = -\sum_k F_{i,k}, \quad F_o T_o = -\sum_k F_{i,k} T_{i,k}, \tag{10}$$

where the input terminals are indexed by k.

Cabin: To simplify, we use a reduced model of the cabin in this study, characterized by a fixed volume $V = 141.58$ m^3, number of passengers $n = 200$, thermal energy per passenger $Q_{pass} = 90$ W and heat gain from the external environment $\delta Q = 0$ W. Moreover, we assume that a release valve in the cabin can provide an air flow rate F_l to maintain a constant pressure $P = P_i = P_o = 101.325$ kPa. Just like in the container, the Ideal Gas Law relates the pressure P with the cabin temperature $T_o = T$, its volume V and the mass m of the inner air. Because of our assumptions, the thermal energy of the air inside the cabin, $Q_{cab} = C_{air} m T = C_{air} PV/R$ is a constant, while the dynamic equation for Q_{cab} degenerates into the following algebraic equation:

$$0 = \frac{\mathrm{d}Q_{cab}}{\mathrm{d}t} = C_{air}\Big(F_i T_i + F_o T + F_l T\Big) + nQ_{pass} + \delta Q. \tag{11}$$

Lastly, the cabin mass m obeys the following differential equation:

$$\frac{\mathrm{d}m}{\mathrm{d}t} = F_i + F_o + F_l. \tag{12}$$

Fan: The fan enforces a flow rate $F_f = F_i = -F_o = 0.3042$ kg/s, which is assumed as fixed in this paper. Furthermore, the input and output temperatures are equal, i.e. $T_o = T_i$.

Ducts: The air ducts connect each component together and have various sizes and materials. Each duct enforces $F_i = -F_o$ at its terminals, as well as the same input and output temperatures, i.e. $T_o = T_i$. Moreover, a duct is associated with a tuple of variables (c, w, r, l, T_{max}, F_{max}), including the cost per area, weight per area, radius, length, maximum allowed temperature, maximum allowed flow rate and specific heat, respectively. To simplify, we assume that the maximum flow rate is directly proportional to the duct radius, while the pressure drop across each duct is zero. Tab. I shows 15 possible configurations for a duct in our library. Configurations 1, 4, 7, 10 and 13 relate to aluminum ducts; configurations 2, 5, 8, 11 and 14 relate to steel, and 3, 6, 9, 12 and 15 are composite.

ID	r (m)	F_{max} (kg/s)	T_{max} (K)	c (\$/m^2)	w (kg/m^2)	SP (J/(kgK))
1	0.17	0.400	455	5.50	2.17	900
2	0.17	0.400	800	3.00	3.85	450
3	0.17	0.400	330	41.50	0.75	NA
4	0.26	1.250	455	5.50	2.17	900
5	0.26	1.250	800	3.00	3.85	450
6	0.26	1.250	330	41.50	0.75	NA
7	0.23	0.870	455	5.50	2.17	900
8	0.23	0.870	800	3.00	3.85	450
9	0.23	0.870	330	41.50	0.75	NA
10	0.20	0.695	455	5.50	2.17	900
11	0.20	0.695	800	3.00	3.85	450
12	0.20	0.695	330	41.50	0.75	NA
13	0.15	0.304	455	5.50	2.17	900
14	0.15	0.304	800	3.00	3.85	450
15	0.15	0.304	330	41.50	0.75	NA

Table I: Available configurations for a duct in our library $\mathcal{L}$. Configurations 1 and 2 can also be used to implement the Heat Exchanger. The columns relate, respectively, to the radius, maximum allowed flow rate, maximum allowed temperature, cost per area, weight per area, and specific heat. The specific heat parameter SP is only used in the heat exchanger model.

Controller: A typical ECS may leverage complex, hierarchical control architectures, where a high-level supervisor provides the appropriate set points to lower-level proportional-integral-derivative (PID) controllers, which directly interact with the plant. While our simulation-based methodology can support the exploration of these architectures in closed-loop configuration, in this paper, we opt for a simplified steady-state model of the controller to formulate the architecture exploration problem. Specifically, given the current system state and input, as measured from the physical plant, the controller directly regulates the valve coefficients C_1 and C_2 with the RAM air flow-rate F_a to achieve the desired steady-state cabin temperature and air flow, while avoiding component faults. The control parameters can then be determined as part of the topology sizing problem as further detailed below.

B. System Requirements and Mapping

We cast the ECS architecture design problem as a mapping problem using the formulation in Sec. III-B. Given the library

$\mathcal{L}$ in Sec. V-A, our optimization problem is constrained by the achievable system behaviors, as obtained by the composite model $\mathcal{F}_{ECS}(.) = 0$, under the input scenario $u = (T_e, P_e, T_a)$ for the bleed air temperature and pressure, and the ambient air temperature, where $T_e = 450$ K, $P_e = 350$ kPa and $T_a = 240$ K, all constant over time. We denote as $p \in \mathcal{P}$ an assignment over the continuous parameters $P = (A_{HX}, D_{HX}, L_{HX}, F_a, C_1, C_2)$, i.e., the heat exchanger's cross-sectional area, tube diameter and length, the RAM air flow rate, and the positions of Valve 1 and Valve 2, respectively. $\mathcal{P}$ is defined by the lower and upper bounds $(0.15, 0.001, 0.2, 0.5, 0.01, 0.01)$ and $(0.30, 0.005, 0.4, 1.0, 0.10, 0.10)$, respectively, with the units given in Sec. V-A. To initialize the system state, we chose a point x_0 such that $0.5\|x_{ref}\|^2 \leq \|x_0 - x_{ref}\|^2 \leq 0.8\|x_{ref}\|^2$, where x_{ref} is a "target" reference point and $\|x\|$ denotes the 2-norm of x. We aim to minimize the overall material cost for the ducts and heat exchanger computed as follows

$$\frac{C(s,p)}{2\pi} = \sum_{i=1}^{5} l_i \sum_{j \in \mathcal{L}_i} s_{ij} c_j r_j + L_{HX} \frac{A_{HX}}{D_{HX}} \sum_{j \in \mathcal{L}_{HX}} s_{HX,j} c_j, \tag{13}$$

where the first sum in (13) ranges over the set of the five ducts in $\mathcal{T}$, $s_{ij} = 1$ if and only if the configuration $j \in \mathcal{L}_i$ is selected to implement a duct i, $s_{HX,j} = 1$ if and only if the configuration $j \in \mathcal{L}_{HX}$ is selected to implement the heat exchanger, while c_j and r_j are defined as in Tab. I.

A set of optimization constraints for the objective in (13) derives from the spatial arrangement of the ducts (*mechanical viewpoint*), specifying that the lengths of ducts 1, 2 and 5 are fixed ($l_1 = 0.1$ m, $l_2 = 0.5$ m and $l_5 = 0.25$ m), while the others are linked to the heat exchanger length via the constraints $l_3 = l_4 = (l_2 - L_{HX})/2$. These length constraints can be directly incorporated into the cost function (13) by adequately rearranging its terms. The other optimization constraints will instead enforce that the component-level contracts are compatible (i.e. $\mathcal{C}_\mathcal{T}$ is compatible and therefore $\mathcal{F}_{ECS}(.) = 0$ provides the correct system behaviors), and their aggregation is consistent with the application contract, i.e. $\mathcal{C}_\mathcal{A} \wedge \mathcal{C}_\mathcal{T}$ is consistent. Therefore behaviors obtained by solving $\mathcal{F}_{ECS}(.) = 0$ refine the top-level requirements captured by $\mathcal{C}_\mathcal{A}$. As discussed in Sec. III-B, we distinguish between functional and extra-functional constraints.

In addition to the constraints generated by the DAEs of the model $\mathcal{F}_{ECS}() = 0$, other *functional constraints* enforce that the cabin reaches a comfortable steady-state temperature ($T_{c,\infty} \in [290, 300]$ K) and maintains a desirable flow of fresh air ($F_{c,\infty} \in [0.8, 1.2]$ kg/s). Furthermore, for safety, the heat exchanger should not freeze ($T_{x,\infty} > 273$ K). Finally, the temperature and flow rate of the air through the ducts must not exceed their maximum value, based on the selected duct configuration, i.e.,

$$T_{i,\infty} \leq \sum_{j \in \mathcal{L}_i} T_{j,max} s_{ij}, \;\; F_{i,\infty} \leq \sum_{j \in \mathcal{L}_i} F_{j,max} s_{ij}, \tag{14}$$

for all $i \in \{1, \ldots, 5\}$. These constraints assure that the assumptions of the duct contracts are discharged by the guarantees of the other contracts in the composition $\mathcal{C}_\mathcal{T}$. The *extra-functional constraints* include the box constraints imposed by $\mathcal{P}$ on p, the sum (cardinality) constraints on the $\{s_{ij}\}$ and $\{s_{HX,j}\}$ Boolean variables and a maximum weight constraint requiring the weight of the ECS be less than W_{max} = 44.5 kg. The overall weight can be computed with a similar expression as the one used for the objective function in (13), where c_j is replaced with w_j.

We observe that both the objective function and the weight constraint can be linearized by using auxiliary variables and constraints to eliminate the products of continuous and Boolean variables in (13). Therefore, the extra-functional optimization problem can be formulated as a mixed integer-linear program (MILP) and efficiently solved by a commercial solver. Moreover, the duct configurations in the library form an ordered set with respect to their temperature and flow constraints in (14), which allows eliminating all the dominated configurations whenever one of the constraints in (14) is violated. Therefore, by Theorem IV.2, we can leverage our hybrid optimization scheme to solve the ECS architecture exploration problem.

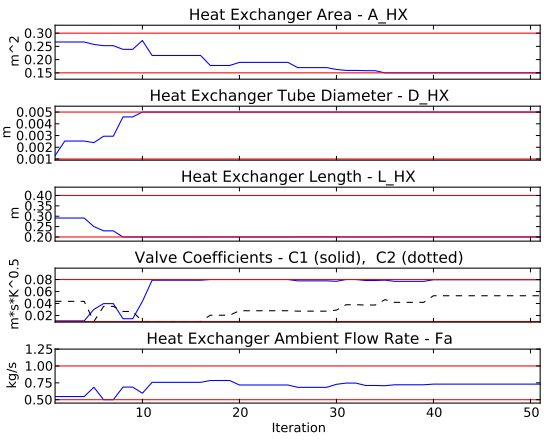

Figure 3: Single Nelder-Mead optimization trace: continuous configuration parameters p for heat exchanger and controller.

C. Optimization Results

We implemented the full ECS model using MODELICA[1], an object-oriented modeling language for acausal systems based on DAEs, and JMODELICA[2], an open source, PYTHON toolbox for the simulation of MODELICA models. The discrete optimization routine leverages CPLEX[3] via the PYTHON interface PULP[4]. The continuous optimization routine utilizes the version of the NR method that is built into JMODELICA. All tests were executed on an Intel Xeon 3.59-GHz processor with 24-GB memory.

A single NM optimization trace is shown in Fig. 3 and Fig. 4 for a randomly selected initial guess in $\mathcal{P}$. As apparent from Fig. 3, OPTSIZE aims to minimize L_{HX} and A_{HX} and maximize D_{HX}, while determining the steady-state control values of the valve coefficients and the RAM air flow rate to satisfy the system requirements. The red lines in Fig. 3 indicate the parameter bounds. In practice, to evaluate the objective function and the optimization constraints for each parameter configuration, we run a transient simulation for enough time to guarantee that changes in the system state and output variables become negligible, hence the steady-state is achieved. Optimization constraints other than bounding boxes are incorporated in the cost function using a ramp penalty function. As shown in Fig. 4, although the cabin temperature (T_c), cabin flow rate (F_c) and heat exchanger temperature (T_x) requirements may be initially violated, the optimization trace is eventually driven into the feasible region, denoted by the thresholds in red, in a way that is largely independent of the specific slope selected for the ramp penalty function.

We then executed the hybrid optimization scheme under the assumption that only the first six duct configurations in Tab. I

[1] https://www.modelica.org/
[2] http://www.jmodelica.org/
[3] www.ibm.com/software/integration/optimization/cplex-optimizer/
[4] https://pypi.python.org/pypi/PuLP

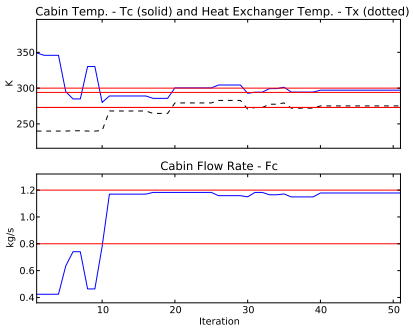

Figure 4: Single Nelder-Mead optimization trace: evaluation of system level requirements on cabin temperature (T_c) and flow rate (F_c), and heat exchanger temperature (T_x).

Library		Runtime Performance			
Size	**Discrete Choices**	**Cost**	**Discrete Iterations**	**Simulations**	**Run Time (h)**
FULL ENUMERATION					
6	15,552	112.39	17	62,496	3.72
9	118,098	112.03	72	257,141	15.42
12	497,664	18-h Timeout			
15	1,518,750	21-h Timeout			
LEARNCONS					
6	15,552	112.39	2	5,812	0.36
9	118,098	112.03	4	13,329	0.83
12	497,664	111.63	6	21,418	1.34
15	1,518,750	111.14	9	31,874	1.91

Table II: Runtime *versus* library size (number of available duct configurations).

are available in our library. To decrease the chance of getting trapped in a local minimum, we also ran the sizing problem separately using three different initial points for each discrete selection. The initial guess consists of $A_{HX} = A_{HX,min}$, $L_{HX} = L_{HX,min}$ and $D_{HX} = D_{HX,max}$, while F_a, C_1 and C_2 are randomly assigned within $\mathcal{P}$. When only one discrete assignment is eliminated at each iteration, which we call the *full-enumeration* operating mode, the optimization converges after 17 selections and 51 sizing iterations for an overall execution time of 3.72 hours to explore 62,496 total configurations. The optimum material cost of 112.39 is achieved by asserting the Boolean variables $\{s_{1,4}, s_{2,1}, s_{3,4}, s_{4,4}, s_{5,4}, s_{HX,1}\}$ and for the following sizing parameters: $\{A_{HX} = 0.15\ \text{m}^2, D_{HX} = 0.005\ \text{m}, L_{HX} = 0.2\ \text{m}, F_a = 0.66\ \text{kg/s}, C_1 = 0.07\ \text{m·s·K}^{0.5}, C_2 = 0.07\ \text{m·s·K}^{0.5}\}$. When LEARNCONS is instead used to eliminate a larger set of conflicting assignments based on the topology balance equations, the same optimum can be obtained using only 2 discrete iterations and a total of 22 minutes, i.e. more than one order of magnitude improvement in execution time. This shows that our algorithm is indeed able to drastically reduce the search space and the number of calls to OPTSIZE. The average execution time of OPTSEL is on the order of a few milliseconds.

Finally, we ran the proposed algorithm with an increasing number of allowed duct configurations in Tab. I to test its scalability as the size of the library $\mathcal{L}$ increases. As shown in Tab. II, design spaces including up to 1.5 million discrete configurations could be explored in less than 2 hours, while the full-enumeration method would timeout after 18 and 24 hours, respectively, for a library including 12 and 15 duct configurations. For a library of 9 configurations, LEARNCONS almost shows a 20× improvement in runtime performance with respect to full enumeration, while achieving the same optimum.

VI. CONCLUSIONS

We proposed an optimization-based methodology for cyber-physical system architecture selection and sizing, which uses a two-level iterative scheme to traverse a hybrid discrete-continuous design space in a scalable way. Our algorithm can substantially reduce the optimization time when applied to an industrial case study such as an aircraft environmental control system. As future work, we would like to further investigate the complexity of the proposed scheme and its integration with our framework for optimized selection of topologies subject to reliability constraints [6].

Acknowledgments. The authors wish to acknowledge Jeff Ernst and Earl Lavallee (UTC) for helpful discussions.

REFERENCES

[1] I. Moir and A. G. Seabridge, *Aircraft Systems: Mechanical, Electrical and Avionics Subsystems Integration. Third Edition*. Chichester, England: John Wiley and Sons, Ltd, 2008.

[2] A. Sangiovanni-Vincentelli, W. Damm, and R. Passerone, "Taming Dr. Frankenstein: Contract-Based Design for Cyber-Physical Systems," *European Journal of Control*, vol. 18-3, no. 3, pp. 217–238, 2012.

[3] P. Nuzzo, H. Xu, N. Ozay, J. Finn, A. Sangiovanni-Vincentelli, R. Murray, A. Donzé, and S. Seshia, "A contract-based methodology for aircraft electric power system design," *IEEE Access*, vol. 2, pp. 1–25, 2014.

[4] A. Canedo, J. Wan, and M. A. Al Faruque, "Functional modeling compiler for system-level design of automotive cyber-physical systems," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design*, 2014, pp. 39–46.

[5] A. Rajhans, A. Bhave, I. Ruchkin, B. Krogh, D. Garlan, A. Platzer, and B. Schmerl, "Supporting heterogeneity in cyber-physical systems architectures," *IEEE Trans. Automatic Control*, vol. 59, no. 12, pp. 3178–3193, Dec. 2014.

[6] N. Bajaj, P. Nuzzo, M. Masin, and A. Sangiovanni-Vincentelli, "Optimized selection of reliable and cost-effective cyber-physical system architectures," in *Proc. Design, Automation and Test in Europe*, 2015.

[7] S. Praharaj and S. Azarm, "Two-level nonlinear mixed discrete-continuous optimization-based design: An application to printed circuit board assemblies," *ASME J. Electron. Packag.*, vol. 114, no. 4, pp. 425–435, 1992.

[8] G. Box and N. Draper, *Empirical Model-Building and Response Surfaces*. John Wiley, 1987.

[9] M. Hershenson, S. Boyd, and T. H. Lee, "Optimal Design of a CMOS Op-Amp via Geometric Programming," *IEEE Trans. on CAD*, vol. 20, pp. 1–21, January 2001.

[10] J. V. C. Vargas and A. Bejan, "Thermodynamic optimization of finned crossflow heat exchangers for aircraft environmental control systems," vol. 22, pp. 657–665, 2001.

[11] L. Shang, G. Liu, and P. Hodal, "Development of high performance aircraft bleed air temperature control system with reduced ram air usage," *IEEE Trans. Control Systems Technology*, vol. 18, no. 2, pp. 438–445, March 2010.

[12] Y. Tu and G. P. Lin, "Dynamic Simulation of Aircraft Environmental Control System Based on Flowmaster," *Journal of Aircraft*, vol. 48, no. 6, pp. 2031–2041, Nov. 2011.

[13] O. Mickelin, N. Ozay, and R. M. Murray, "Synthesis of correct-by-construction control protocols for hybrid systems using partial state information," in *Proc. American Control Conference*, June 2014, pp. 2305–2311.

[14] I. Pérez-Grande and T. J. Leo, "Optimization of a commercial aircraft environmental control system," *Applied Thermal Engineering*, vol. 22, no. 17, pp. 1885–1904, Dec. 2002.

[15] J. A. Nelder and R. Mead, "A simplex method for function minimization," *Computer Journal*, vol. 7, pp. 308–313, 1965.

[16] M. Dechow and C. Nurcombe, "Aircraft environmental control systems," in *Air Quality in Airplane Cabins and Similar Enclosed Spaces*, ser. The Handbook of Environmental Chemistry, M. Hocking, Ed. Springer Berlin Heidelberg, 2005, vol. 4H, pp. 3–24.

[17] J. Warner, "Environmental control system condensing cycle," *European Patent*, vol. EP 0 542 909 B1, 1994.

[18] D. Dias, E. L. Zaparoli, M. E. Gomes, W. H. L. Turcio, E. Brasileira, and B. F. Lima, "Dynamic modeling of an aeronautical air conditioning simple air cycle machine," no. 1995, 2010.

[19] R. Fox and A. McDonald, *Introduction to fluid mechanics*. Wiley, 1985.