Title
Symbiotic CPS Design-Space Exploration through Iterated Optimization

Authors
Yu, Sheng-Jung
Incer, Inigo
Prabhu, Valmik
et al.

Publication Date
2023-05-09

DOI
10.1145/3576914.3587525

Peer reviewed

Symbiotic CPS Design-Space Exploration through Iterated Optimization

Sheng-Jung Yu[1], Inigo Incer[1], Valmik Prabhu[1], Anwesha Chattoraj[2], Eric Vin[3], Daniel Fremont[3], Ankur Mehta[2], Alberto Sangiovanni-Vincentelli[1], Shankar Sastry[1], and Sanjit Seshia[1],
[1]Univeristy of California, Berkeley, [2]University of California, Los Angeles, [3]University of California, Santa Cruz
{shengjungyu, inigo, valmik, alberto, shankar_sastry, sseshia}@berkeley.edu
{anwchatto, mehtank}@ucla.edu
{evin, dfremont}@ucsc.edu

ABSTRACT

Cyber-physical systems (CPSs) are complex systems comprised of computational processes, communication networks, and elements interacting with the physical world. The design of the CPSs involves many domain-specific tools and design flows created by engineers with diverse domain knowledge. As the scale of the systems increases, the heterogeneity nature of CPS design prolongs the CPS design process, making exhaustive design-space exploration infeasible. The symbiotic design methodology, in which the designers interact with optimization tools during the design process, is therefore promising to facilitate the design process by performing design exploration in a properly restricted design space. We present a symbiotic design methodology, which explores the design space iteratively and optimizes the system by exploiting the collaboration between designers and tools. The optimization tools perform the design space exploration, while the human designers use their expertise to guide the exploration by restricting the design space. Experimental results based on a robot car configuration problem and an unmanned aerial vehicle design problem show that the methodology can efficiently and effectively discover unconventional designs while optimizing the design objectives.

CCS CONCEPTS

• **Computer systems organization** → **Embedded and cyber-physical systems**; • **Human-centered computing** → **User centered design**.

ACM Reference Format:
Sheng-Jung Yu[1], Inigo Incer[1], Valmik Prabhu[1], Anwesha Chattoraj[2], Eric Vin[3], Daniel Fremont[3], Ankur Mehta[2], Alberto Sangiovanni-Vincentelli[1], Shankar Sastry[1], and Sanjit Seshia[1], . 2023. Symbiotic CPS Design-Space Exploration through Iterated Optimization . In *Cyber-Physical Systems and Internet of Things Week 2023 (CPS-IoT Week Workshops '23), May 09–12, 2023, San Antonio, TX, USA.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3576914.3587525

CPS-IoT Week Workshops '23, May 09–12, 2023, San Antonio, TX, USA

ACM ISBN 979-8-4007-0049-1/23/05.
https://doi.org/10.1145/3576914.3587525

1 INTRODUCTION

Cyber-physical systems (CPSs) are complex systems comprised of computational processes, communication networks, and elements interacting with the physical world. The design of CPSs makes use of techniques applied in various engineering fields, depending on the application, e.g., a power distribution system, a space shuttle, or a chemical plant. With such a diversity, CPS design meets with various challenges, such as component heterogeneity and system integration, as elaborated in [10, 12]. To cope with these challenges, several CPS design methodologies have been proposed [2, 4, 8, 9, 13, 14].

One of the preliminary stages in CPS design is design-space exploration. At this stage, one wishes to generate multiple solutions to a design problem and understand their trade-offs with respect to the design objectives. Automated tools for CPS design have been developed to efficiently explore the design space. Finn et al. [5] developed a CPS architecture exploration algorithm using a combination of discrete and continuous optimization. Discrete methods are used to connect and select components; continuous optimization is used to size parameters. Bakakeu et al. [1] studied the design space of the integration of analytic routines in a manufacturing process for CPSs. For the case of unmanned aerial vehicle (UAV) design, Krishnan et al. [7] proposed Autopilot, a method to search the design space of autonomy algorithms and their hardware accelerators for various classes of UAVs. The designs provided by Autopilot were reported to outperform industry-standard accelerators.

As the scale of the systems increases, the prohibitively large design space caused by the complexity and heterogeneous nature of CPS design prolongs the design process and even makes exhaustive and fully automatic design exploration infeasible. In our view, human designers and automated tools should be complementary in CPS design space exploration. Experienced human designers, leveraging their underlying domain knowledge and previous development experience, can direct the tools to focus the exploration on a restricted design space thus avoiding endless searches in inferior design sub-spaces.

A symbiotic design methodology, in which the designer and automated tools collaborate, is therefore a promising solutions to streamline the design process. In this methodology:

- the automated tools explore designs, using provably-good optimization and statistical learning algorithms, to suggest promising designs for exploration in the restricted design space provided by the designers.

- The designers use their domain knowledge to interpret the current state of the exploration carried out by the tools, state choices of optimization preferences, and impose restrictions on the design space for additional exploration by the tools.

To this end, the Defense Advanced Research Projects Agency launched a project for symbiotic design for CPS [3], aiming to accelerate exploration of CPS *via* tight interaction of designers and tools.

Fitzgerald et al. [6] proposed a SysML profile-based language for the creation of design space exploration experiments. They explore the use of genetic algorithms to carry out the actual search. However, the interaction with automated tools is not explored further and manual conversion from the language to an experiment is still needed.

Motivated by the importance of symbiotic design methodology, this paper proposes a design methodology for CPSs based on iterative optimization. Our contributions are the following:

- We propose a symbiotic CPS design methodology which iteratively explores the design space with the collaboration of the designers and the optimization tools. To the best of our knowledge, this is the first paper that proposes a symbiotic CPS design exploration methodology in which the designers interact directly with automated tools without manual conversions.
- We identify a set of design choices that the designers can make to interact with the design tools for design space exploration. Compared with [6], no manual conversion from the design choices to set the automated tools are required.
- We apply the methodology in two CPS design problems: the design of UAVs and the design of robot configurations. The results shows that the proposed methodology is effective for restricting the design space to interesting sub-spaces and, consequently, it produces results more efficiently.
- We present a integrated-circuit (IC) design inspired methodology for UAV design consisting of the following steps: *component selection*; *placement* in free 3D space, and a *routing* step which adds a frame to the UAV.

2 PRELIMINARIES

In this section, we introduce the CPS design space exploration problem.

2.1 Elements in CPS Design Space Exploration Problem

The elements in a CPS design exploration problem consist of a library that includes the components available for composing a design, parameters that change the behavior of a component, a net-list that describes the connection between the components, a set of *rewards* that are used to assess the quality of the design, and an oracle that computes the rewards.

Components and Component Library. The components are instances of the hardware and/or of the software for control algorithms. Their composition forms a CPS design. The components communicate through ports.A component Library, denoted by $\mathcal{L}$, is the set of components that are specified to construct candidate implementations for a particular design problem. The *component types*, which partition the library $\mathcal{L}$, abstract the notion of composability for components: components having the same type can interact with other components through the same types of ports. Each component in the same component type shares the same set of parameters that affect their behaviors. Logic gates in cell-based digital circuit design flow are an example of a components library. In the case of UAV design, component types could be propellers, electronic speed control circuits (ESCs), motors, etc. at various levels of specificity.

Net-list. The net-list is an interconnection of component types. A net-list $\mathcal{T}$ can be represented as a graph whose vertices are component types and whose labeled edges tell which ports are connected between the component types. To generate a design, for every node in the net-list, the component type at the node is replaced with a "real" component of the same type.

Design parametrization. Once a net-list is chosen, all the designs having the same net-list can be described by a set of parameters from all its parameters of the component types. The set of parameters, denoted $\phi^{\mathcal{T}}$, contains all candidate designs built out of $\mathcal{T}$. We further denote $d^{\mathcal{T}}$ as the assignment of the value to the parametrization $\phi^{\mathcal{T}}$. Each candidate design consists of selections of components in each component type and assignments of the values for the parameters.

Oracle. The oracle, denoted by O, evaluates the performance of a candidate design, potentially considering interactions among components. The oracle for a design problem can be a simulator that models the system and provides estimates of system performance, or an experiment conducted on a prototype to measure performance in the real world. The computational complexity and cost for running an oracle increase the difficulty of design exploration given limited design time and resources.

Rewards. The rewards are elements in a vector representing the evaluation results of candidate designs mapped by the oracle as shown in the following relation:

$$\vec{r} = O(\mathcal{T}, d^{\mathcal{T}}), \tag{1}$$

where $\mathcal{T}$ denotes the net-list, $\vec{r}$ denotes the reward, $d^{\mathcal{T}}$ represents an assignment of value to the parametrization $\phi^{\mathcal{T}}$. Each element in the vector refers to an aspect of the design, such as cost, yield for manufacturability, energy efficiency, and functionality metrics. For ease of explanation, we can define each element in the rewards such that the higher the value is, the more desirable the design is. The optimization of the rewards is thus a multi-objective optimization problem, which many algorithms, strategies, and automated tools have been proposed to solve by finding its Pareto front or solving a single objective optimization problem.

2.2 CPS Design Space Exploration Problem

Given the elements in the design space exploration problem, the design space, denoted by $\mathcal{D}$, is defined as all possible net-lists and parameter assignments. The CPS design space exploration problem then can be defined as follows: Given the component library, oracles, and the rewards, find the net-list and assignment of parameters for the design such that the rewards are optimized.

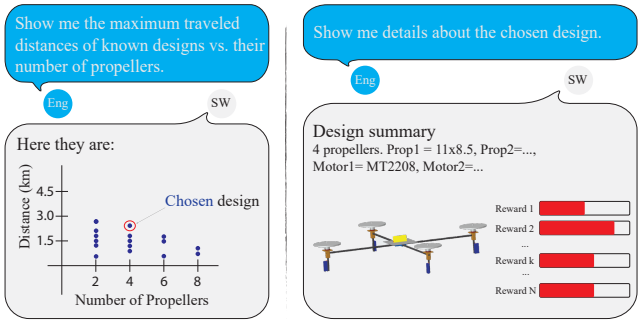


Figure 1: An initial interaction in which the engineer studies the performance of existing designs.

In this paper, we assume that a net-list is provided by the human designers, and thus the design space is limited to that net-list. Given the net-list, the assignment to the parametrization defines the design space containing the net-list, as shown by the following relation:

$$d^{\mathcal{T}} \in \mathcal{D}^{\mathcal{T}}, \tag{2}$$

where $\mathcal{T}$ is the provided net-list, $\mathcal{D}^{\mathcal{T}}$ denotes the design space limited to the net-list, and $d^{\mathcal{T}}$ is the assignment to the parametrization $\phi^{\mathcal{T}}$. We also refer to $d^{\mathcal{T}}$ as a design candidate since an assignment to the parameters uniquely determines a design.

3 THE SYMBIOTIC CPS DESIGN SPACE EXPLORATION METHODOLOGY

Given a net-list $\mathcal{T}$ that defines a space of design parameters ϕ, the ideal design exploration solves the following multi-objective optimization problem:

$$d^{\mathcal{T}} \leftarrow \arg\max_{\phi^{\mathcal{T}}} \ \vec{r} = O(\mathcal{T}, \phi^{\mathcal{T}}), \tag{3}$$

where $\mathcal{T}$ denotes the net-list, $d^{\mathcal{T}}$ is the design with the optimized parameter values, $\phi^{\mathcal{T}}$ is the parametrization of the design, and $\vec{r}$ indicates the evaluation of reward provided by the oracle O.

However, solving (3) may be prohibitive, as $\phi^{\mathcal{T}}$ will be large-dimensional in complex CPSs. Therefore, in this section, we propose our symbiotic design space-exploration methodology. We abstracted them as *Design Choice Step* and *Exploration Step*. The design choice step allows designers to restrict the design space to guide the automated tools, and the exploration step is performed by the automated tools to explore the design space. Our symbiotic design space exploration is the continual process that alternates between these two steps. First, a few designs are created through traditional (manual) methods as seed designs. Designers then select an existing seed design, an objective function, a subset of the parameters over which to optimize, and constraints on the parameters to perform an iteration of optimization; to make their selections, designers make use of their experience that a design objective may be improved by altering a subset of the parameters. After the design choice is made, the designers invoke the automated tools to solve (5) and obtain all results as the potential designs. The designers then review these potential designs, make manual modifications at their discretion,

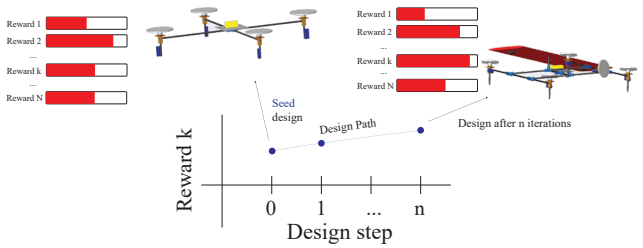


Figure 2: A design step is a modification of an existing design to produce a new one, which is added to the library of known designs. A design path connects any known design to one of the seed designs.

select new design choice and start the next iteration of optimization. The process terminates when the designer is satisfied with the performance of a design.

Thus, in this methodology, there is *symbiotic interaction between the human user and automated tool.* Figure 1 illustrates a dialog in which the engineer interrogates the software for a properties of known designs. In this methodology, every known implementation is obtained from an existing design. As a result, there is a unique path connecting every known implementation to one of the original seed designs. For each design step, we know exactly how the new design obtained, as shown in Figure 2. This provides a high degree of explainability to the design process. In the following, we detail the Design Choice Step and Exploration Step.

3.1 Design Choice Step

The designer can make the design choice to guide the optimization problem. Here we summarize the design choices that a designer can make:

- *Seed Design Setting:* The designers can provide traditional (manual) designs to the automated tool.
- *Manual Modification:* The designers make modifications, including topology changes and parameters adjustment, to a design provided by the automated tools and then sends it back to the tools.
- *Objective Function Setting:* The designer can specify the comparison metrics for the rewards, such as partial ordering, weighted sum, or priority of each element in the rewards. The objective function is denoted by F.
- *Parameter Set Restriction:* The designer can specify the subset of the parameters $\phi'^{\mathcal{T}} \subseteq \phi^{\mathcal{T}}$ for optimization.
- *Parameter Constraint Setting:* Besides the variable set, additional constraints can be set on the parameter to further restrict the design space, as shown by

$$c_i(\phi_{\mathcal{T}}) < 0, i = 0 \ldots N_c, \tag{4}$$

  where c_i denotes the i^{th} constraint functions and N_c denotes the number of constraints set by the designers.
- *Incremental Optimization:* The designers request the automated tool to solve the optimization problem with selected global optimization with a certain design in the automated tools as the initial point.

- *Local optimization:* Similar to the incremental optimization, the designers request the automated tool to solve optimization problem, while the solver only performs local optimization.

3.2 Exploration Step

As introduced, the ideal exploration is prohibitive due to the large design space. Therefore, the design space needs to be properly restricted so that the exploration can be performed by the automated tools and produce satisfying results. Given the restriction provided by the designer, as introduced in Section 3.1, here we formulate the exploration step as the following optimization problem:

$$d'^{\mathcal{T}} \leftarrow \arg\max_{\phi_{\mathcal{T}}} F(O(\mathcal{T}, \phi_{\mathcal{T}})) \quad (5)$$
$$\text{s.t. } c_i(\phi_{\mathcal{T}}) < 0, i = 0 \dots N_c,$$
$$d'^{\mathcal{T}}_j = d^{\mathcal{T}}_j, \forall j \ \phi^{\mathcal{T}}_j \notin \phi'^{\mathcal{T}}$$

where $\phi^{\mathcal{T}}_j$ is the j^{th} parameter in $\phi^{\mathcal{T}}$, $d^{\mathcal{T}}_j$ is the value of the j^{th} parameter from the existing design, $d'^{\mathcal{T}}$ is the value of the parameters of the optimized design, $d'^{\mathcal{T}}_j$ is the value of the j^{th} parameter of the optimized design, and all the other symbols are as defined in Section 2 and Section 3.1.

In this formulation of the problem, we perform optimization only over the parameters $\phi'^{\mathcal{T}}$, and the multi-objective optimization problem is also provided with the function F to reduce the number of rewards, prioritize the rewards, or even convert the multi-dimensional rewards into a single objective function. Thus, the optimization problem will in general be much more manageable than (3). The inputs to the problem are a library of components, a reward to be improved, a topology, a design to improve, and the allowed design choices or modifications that the algorithm can effect on the original design in order to improve the reward selected by the user.

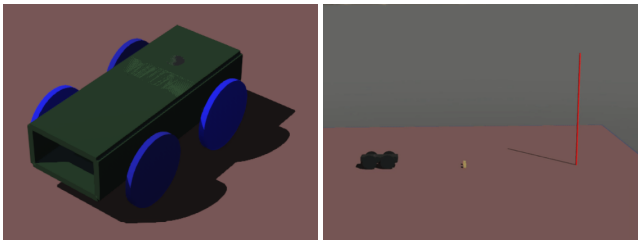
4 ILLUSTRATION: ROBOT CAR CONFIGURATION OPTIMIZATION

To show the effectiveness of our proposed symbiotic design methodology on the CPS design exploration problem, we apply our design methodology to two CPS design exploration problems. The first design problem is a robot car configuration optimization problem, through which we aim to demonstrate the interactions between designers and automated tools on a simple design exploration problem and the incremental improvement achieved by the methodology. The second one is a UAV design problem, which serves as a complex CPS design problem to show that the proposed symbiotic methodology can generate unconventional design and optimize performance.

In this section, we demonstrate how our symbiotic methodology can be applied to the robot car configuration optimization problem. The UAV design problem will be introduced in Section 5.

4.1 Design Problem Description

In the robot car configuration problem, the objective is to find configurations of the robot car such that it can reach the target with the minimized consumed energy and elapsed time in a testing environment. The configuration of the robot car is specified by four parameters: *number of wheels*, *length*, *width*, *radius*. The number of wheels is a discrete parameter that determines the number of wheels on one side of the robot car. The length and the width determine the dimension of the robot car and affect the distance between the wheels. The radius represents the size of the wheels. Figure 3a shows an example of the robot car.



(a) (b)

Figure 3: The visualization extracted by the simulator in the robot car configuration optimization problem (a) The configuration of the car. (b) The environment for testing, which contains yellow speed bumps and a red pole as the target.

The testing environment contains speed bumps and a pole that represents the target position, as shown in Figure 3b. The oracle simulates the robot car in the environment and returns the elapsed time and the consumed energy for reaching the goal as the rewards. The execution time for the oracle to run one simulation takes about 2 minutes.

4.2 Design Methodology

Using our symbiotic design methodology, experienced designers with domain knowledge can specify the parameters to optimize, set constraints on the parameters, and provide an objective function for the rewards. In this optimization problem, since the dimension of the parameters are only four, we apply exploration on the continuous variable and let the designers determine the number of wheels, the constraints, and the objective function.

The objective function is set as follows:

$$F(e, t) = e + 0.5t, \quad (6)$$

where e denotes the energy consumption to complete the goal, t is the time to complete goal, and $F(e, t)$ is the objective function, taking the rewards e and t as inputs.

In the exploration, the human designers determine the number of wheels and set the constraints for the parameters as follows:

$$L \in (80, 200)$$
$$W \in (65, 200)$$
$$R \in (36, 200), \quad (7)$$

where L denotes the length, W represents the width, and R is the radius of the robot car. The underlying exploration algorithm is simulated annealing and Bayesian optimization.

4.3 Results

Table 1 lists the design choice and the results at each iteration. In the symbiotic design process, the designers first perform the optimization with the number of wheels set to 4. Then the designers proceed

Iteration	N_w	Optimization tools	Result
1	4	Bayesian Optimization	412.485
2	4	Simulated Annealing	358.010
3	3	Bayesian Optimization	325.798
4	2	Bayesian Optimization	279.403
5	2	Simulated Annealing	277.839

Table 1: The design choice and results for each iteration of the design space exploration on the robot car configuration problem. N_w denotes the number of wheels and the Result is the objective function value.

with a different optimization tool and observe an improvement in the objective function. In iterations 3 and 4, the designers try to change the number of wheels to explore different design spaces. The results of these two iterations show that the robot car has a better performance on the objective function when the number of wheels is set to 2. Finally, the designer performs the last optimization in the design space with 2 wheels and improves the design further. The process of the iterated design and the improved objective function value shows that our symbiotic design methodology can effectively handling optimization with the interaction between the designers and the optimization tools.

5 ILLUSTRATION: UAV DESIGN

In this section, we illustrate the methodology in a UAV design problem to show that the proposed methodology can be applied to complex CPS design. Based on our symbiotic design methodology, we propose an IC design inspired methodology for UAV design consisting of the following steps: first, *component selection*; second *placement* in free 3D space, and third, a *routing* step which adds a frame to the UAV. The component selection chooses the component type for a given topology of the UAV by optimizing the discrete parameters determined by component type and continuous parameters that describe the behaviors of the components. The placement optimizes the parameters for physical configuration and generates the mechanical properties of the UAV. After the rewards are optimized, the routing connects the components to ensure that the structure of the UAV is stable. In the following, we detail the UAV design problem and describe each step.

5.1 Design Problem Description

The library of components for the UAV design include the component types of propellers, motors, ESCs, flanges, central support, mechanical connectors, and batteries. Models for the propellers come from APC Propellers; we use the motor characteristics from T-motor; our batteries are modeled after the offering of Turnigy; our structural elements follow the parameters of DragonPlate's offering.

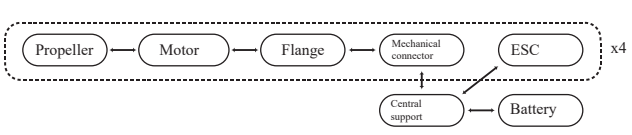


Figure 4: The topology of the quadrotor. Each box denotes a component type.

We have an initial quadrotor design, as shown in Figure 4. Designs are obtained by providing specific instances for each component type together with any additional parameters needed to place these components in free space. The parametrization of this topology gives us the following design choices: positions and orientations of the four propellers, positions of the four ESCs and batteries, selection of all components, and control parameters for an LQR controller. The parametrization thus consists of the cyber aspect and the physical part of the design. As a result, The design choices on the parameter is any subset of the parametrization with at least 12 components and over 30 continuous parameters.

The rewards are computed using a flight simulator. The inputs to this simulator are the performance files for the propellers and their geometrical data, the electromechanical characteristics of the motors, and descriptions of the batteries. In addition, the simulator takes as inputs the mass properties of the UAV. We build assemblies in PTC Creo[1], a 3D CAD tool, and obtain from it these mass properties; to make programmatic calls to Creo, we used the python library Creopyson. The simulator implements several tests, detailed in our results section. Each of these tests generates rewards we can use to assess the performance of candidate designs. The UAV design challenge, together with the libraries of components, and the simulator was provided by the authors of [15].

5.2 Overview of the Proposed UAV Design Methodology

Algorithm 1 outlines our proposed IC design inspired methodology for UAV design.

Algorithm 1 UAV Design step

Require: existing design $d^{\mathcal{T}}$ with 4 propellers with orientations ϕ_o and positions ϕ_p (24 variables), 4 propeller choices ϕ_{pc}, 4 motor choices ϕ_{mc}, 4 ESC choices ϕ_{ec}, 1 battery choice ϕ_{bc} (we let $\phi^{\mathcal{T}}$ be all parameters), 4 control inputs ϕ_{cnt}, and the design choice at iteration k including the subset of the parametrization, $\phi_k'^{\mathcal{T}}$, constraints $C_k = \{c_i | i = 0 \dots N_c\}$, and objective function F_k to improve.

1: $d'^{\mathcal{T}} \leftarrow d^{\mathcal{T}}$
2: **while** NotConverged **do**
3: $d'^{\mathcal{T}}_{pc}, d'^{\mathcal{T}}_{mc}, d'^{\mathcal{T}}_{ec}, d'^{\mathcal{T}}_{bc} \leftarrow$ UpdateComponents$(\phi_k'^{\mathcal{T}}, C_k, F_k)$
4: $d'^{\mathcal{T}}_{o}, d'^{\mathcal{T}}_{p} \leftarrow$ UpdatePropellerGeometry$(\phi_k'^{\mathcal{T}}, C_k, F_k)$
5: $d'^{\mathcal{T}} \leftarrow$ MapAndPlace$(\phi_k'^{\mathcal{T}}, C_k, F_k)$
6: $d'^{\mathcal{T}}_{cnt} \leftarrow$ UpdateControl$(\phi_k'^{\mathcal{T}}, C_k, F_k)$
7: $\vec{r} \leftarrow$ ComputeReward$(F_k, d'^{\mathcal{T}})$
8: $d'^{\mathcal{T}} \leftarrow$ RouteDesign$(\phi^{\mathcal{T}})$
9: **return** improved design $d'^{\mathcal{T}}$

Lines 2–7 carry out the optimization problem (5). The convergence of the algorithm is dictated by our optimization method, which is simulated annealing. UpdateComponents carries out component selection for those components allowed by the design choice on the parametrization $\phi_k'^{\mathcal{T}}$. UpdatePropellerGeometry, which updates the geometry, make changes to those positions and

[1] https://www.ptc.com/en/products/creo

orientations of the input UAV $d_{\mathcal{T}}$, where $\mathcal{T}$ is the given topology of the UAV. UpdateControl operates similarly, updating the weights of the LQR controller. For example, $\phi_k'^{\mathcal{T}}$ may contain the parameters for the propellers but not for the batteries, thus allowing the propellers be optimized while fixing the battery. MapAndPlace builds a 3D CAD assembly for the candidate implementation using PTC Creo and extracts the mass properties of the UAV. These mass properties are used in the computation of the reward chosen by the user. The candidate implementation generated in the optimization loop only carries out component selection and component placement in free space. The output of this step goes through another step, which we call routing. Routing adds tubes and connectors to the design in order for the UAV to have mechanical integrity as it operates. We now discuss the specific techniques used in component selection and routing.

5.3 Component selection

In Algorithm 1, ϕ_{mc} represents the choice of motors. We have four motors, yielding four objects to decide in ϕ_{mc}. In other words, this parameter has datatype $\mathcal{L}_m^{\times 4}$, where m stands for the motor component type. If we perform a design step to improve reward r_k by allowing C to choose one of the motors, the component selection algorithm can proceed in three ways. (i) If the number of potential motors in the library is not large, we could test all motors and pick the one that maximizes the reward. (ii) We could cluster motors into similar groups and make a decision hierarchically. (iii) We could treat the parameters that define a specific motor as continuous quantities, optimize for those quantities, and pick the library element which is closest to the optimal values.

The third approach deserves further discussion. Many types of components are summarized by few parameters. For example, for our simulator, a motor is summarized by approximately five numbers. It is thus feasible to allow our optimizer to find a good value for the parameters; if the number of values that characterize a component is large, we allow the user to specify which subset of the parameters of the component should be used to make the choice of the component. Our component-selection optimization constantly takes steps that improve a reward based on changes to the parameters specified by the user; as there are other component parameters affected when a specific component is chosen, our algorithm often updates the values of those parameters according to the design choice made by the parameters being optimized.

When components are represented by a small set of numbers, the approach just described is suitable for an implementation. There are some component types, though, such as propellers, which are often characterized by large performance files. In that case, one could say that the number of parameters defining a component is very large. Optimizing over such large parameter space is unfeasible. As a result, we learn sparse representations of the performance files and carry out the optimization in the reduced parameter space. More specifically, we use the geometric data of the propellers as inputs to a neural network that is trained to generate performance metrics for the propellers. This neural network enables us to generate performance files for arbitrary propellers.

5.4 Routing

The optimization loop that chooses and places components in Algorithm 1 outputs a design with no frame to keep the UAV together. It is the role of routing to add tubes and connectors to the UAV. The goal is a solid (connected) frame that is as light as possible without deflecting overmuch during flight. While this step could perhaps be achieved more optimally by using a process like generative design [11], we made the design decision to restrict the frame to a collection of tubes and consider only static flight forces, which enabled significantly simpler calculations and a large reduction in design time.

By representing the mounting point of each motor and the battery as nodes and connecting tubes as edges, we can parametrize this problem as a topology optimization on a graph. Thus, we want to minimize total edge weight while ensuring graph connectedness and satisfying deflection constraints.

We define the position of each node $p_i \in \mathbb{R}^3$, and their displacements $dp_i \in \mathbb{R}^3$. The first l nodes are "fixed" nodes, which are fixed in space during the optimization ($d_{1..l} = 0$) and assumed to be already connected together. These represent the "main body" of the UAV, and would be secured to a fixture during a frame stiffness test. The next m nodes represent "body" nodes, the motor assemblies and other components that will be secured by the frame. And the next n nodes represent any additional "designer" nodes the designer may include, in order to allow for additional design topologies, such as the H-frame. Each node has an associated external force vector $F_i \in \mathbb{R}^3$ representing the weight of the component and the expected force of the associated propeller in flight, if present. Note that these are only predefined for "body" nodes, and must be nonzero. The external forces on fixed nodes are free variables, representing fixturing forces during a test, while the external force on "designer" nodes is zero. We define each potential tube ij to have variable outer radius $r_{ij} \in \mathbb{R}$, and constant thickness t (a tube is considered not present if $r_{ij} = 0$). We assume that each tube is composed of an isotropic material with Young's modulus E. To facilitate more readable equations we can define initial tube vector $L_{ij} = p_j - p_i \in \mathbb{R}^3$ and final tube vector $L_{ij}^* = p_j + dp_j - p_i - dp_i \in \mathbb{R}^3$. We assume that the deflection constraints $\delta_i \in \mathbb{R}$, where $\|dp_i\| < \delta_i$ are reasonably chosen such that the frame is stiffness-limited rather than stress-limited, and that deflections will be small with respect to tube lengths ($\|L_{ij}^*\| - \|L_{ij}\| \ll \|L_{ij}\|$). The tubes thus have independent longitudinal stiffness $K_{long_{ij}} \in \mathbb{R}$ lateral stiffness $K_{lat_{ij}} \in \mathbb{R}$, where

$$K_{long_{ij}} = \frac{2\pi r_{ij} t E}{\|L_{ij}\|} \quad \text{and} \quad K_{lat_{ij}} = \frac{3E\pi(r_{ij}^4 - (r_{ij} - t)^4)}{4\|L_{ij}\|^3}.$$

We can now define the force through each tube $F_{ij} = -F_{ji}, \in \mathbb{R}^3$ as

$$F_{ij} = K_{long_{ij}} \left(\left(L_{ij}^* \cdot \frac{L_{ij}}{\|L_{ij}\|} \frac{L_{ij}}{\|L_{ij}\|} \right) - L_{ij} \right) + K_{lat_{ij}} \left(L_{ij}^* - \left(L_{ij}^* \cdot \frac{L_{ij}}{\|L_{ij}\|} \frac{L_{ij}}{\|L_{ij}\|} \right) \right)$$

Using these, we can define a nonlinear optimization problem.

$$\begin{aligned} \min_{r_{ij}, dp_i, F_{1\ldots l}} \quad & \sum r_{ij} L_{ij} \\ \text{such that} \quad & F_i + \sum_j F_{ij} = 0 \\ & dp_i < \delta_i \\ & 0 \le r_{ij} \le r_{ij_{max}} \end{aligned}$$

The free variables here are the tube radii, the node displacements, and the fixturing forces. The force balance constraint is quartic (linear in dp and cubic in r_{ij}), but since the lateral stiffness is monotonic in r_{ij}, it tends to perform relatively well in practice. The cost function and other constraints are linear. Multiple force configurations may be applied in a single test by duplicating the force and displacement constraints, with fixed nodes changeable between configurations. Note that by allowing the tube radii to continuously vary from zero to some maximum, we bypass the mixed integer program that would otherwise be required to ensure graph connectedness. If a "body" node is not connected to a fixed node, the force balance will fail or the displacement will go to infinity. On the other hand, a "designer" node does not need to be connected, allowing the user to try things without forcing the optimizer to choose a suboptimal frame. Of course, tubes cannot be made with arbitrarily small radii, so all nonzero tubes will have to be increased to some minimum radius r_{min} in a post-processing step.

5.5 Results

We demonstrate the application of our iterated optimization methodology for the exploration of UAVs that perform well on various tests and which have unusual geometries. First we discuss the tests which our simulator performs on each design. We discuss our methodology and obtained designs afterwards.

Our flight simulator implements four tests: hover, straight-line, circle, and oval. The hover test verifies whether the UAV can rise from the ground to a height of 150 m at 2 m/s. The test provides a score equal to the amount of time during which the UAV can rise and hover, capped at both 200 s and 400 s. The straight line test verifies that the UAV can travel horizontally, following a straight line; the test provides a score equal to the distance traveled in meters divided by 10. The minimum score is 200 and the maximum is 400. The circle test requires the UAV to complete a circular course of 1 km diameter; completing the course yields 300 points. Finally, the oval test requires the UAV to complete a path of two straight lines connected by two semicircles; the total length of the path is 3.38 km. Completing the path yields 200 points; points are added by completing it faster. Points are subtracted if the designs deviate from the path they are to follow.

To show the effectiveness of our methodology on design space exploration, we implemented an optimization tool for performing the UAV design steps discussed in Section 5.2, and conducted a series of experiments to find UAVs with unusual geometries, but still displaying acceptable performance in all tests. Our optimization tool is implemented in python, using the dual-annealing algorithm in the SciPy library as the underlying optimization mechanism. The optimization tool is capable of returning all designs that are local minima for the specified reward during design exploration. The flight simulator is relied upon as the ground tool for the design process, i.e., the tool can automatically detect whether a design is valid or not. After the tool provides a set of results, human designers can pick a set of solutions for further design steps.

Now we discuss the iterated design steps we followed to obtain geometrically interesting designs that perform well in all tests. We will specify the reward that is being improved and the design decisions passed as input to the optimization tool at each design step. Table 2 summarizes the design steps we carried out in our search for geometrically-interesting and well-performing designs. For each design step, we use as the reward a metric that indicates the performance of the UAV in one of the tests. To have better optimization results, the reward function in each design step is not necessarily the score of the chosen test, even though the objective of each design step is to maximize the score for the chosen test. Some score functions do not provide local information of the function, as they are not continuous and clip the output between zero and the full score, and thus using the score as a reward may affect optimization efficiency, make it difficult for the tool to find local minima. We use modified rewards in some design steps to facilitate the optimization process.

Step	Design decision	Test	Reward
1	Battery type	Hover	t
2	Propeller positions and orientations	Straight-line	$d/10 - 10e$

Table 2: Design steps for obtaining geometrically-interesting designs. In the reward column, t denotes the flight time, d represents the flight distance, and e is the maximum lateral error from the designated path.

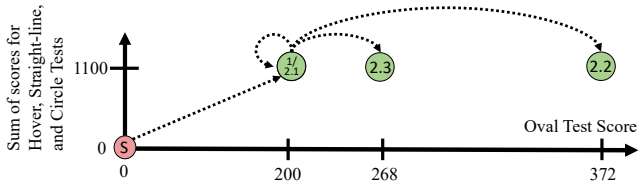
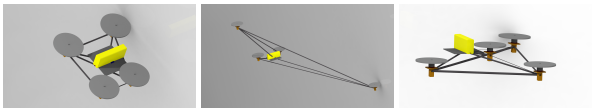


Figure 5: Design path of sample UAV design. The node S is the seed design, and the numbers on each node indicate the design step in which the node is obtained.

The seed design for our first design step is a symmetric quadcopter, similar to the seed design shown in Figure 2. In the first design step, we perform battery selection by exhaustively searching for a battery that optimizes the reward of hovering time in the hovering test. After this, we perform propeller placement and orientation. The reward used in this case if the distance traveled by the UAV in a straight line. We look attentively at local minima in this optimization step in order to obtain geometrically-interesting designs.

Figure 5 shows the resulting design path of our optimization process, together with the scores obtained at each step. The battery selection returns a design that can pass all tests with scores of 400 on the straight-line test, 300 on the circle test, 400 on the hover test, and 200 on the oval test. By optimizing straight-line flight over the placement and orientation of the propellers, we obtain three geometrically-interesting designs, as shown in Figure 6. The first design differs from the seed design by the position of one propeller; this design obtains scores of 400 on the straight-line test, 300 on the circle test, 400 on the hover test, and 200 on the oval test. The second design is asymmetric, with two propellers far away from the center of the UAV. This design achieves 396 on the straight-line test, 300 on the circle test, 400 on the hover test, and 372 on the oval test. The third design has a propeller above the center plate that holds the battery. It obtains scores of 400 on the straight-line test, 300 on the circle test, 400 on the hover test, and 268 on the oval test.



(a) (b) (c)

Figure 6: Three interesting design geometries obtained in design step 2.

The designer may choose the second design as the final result due to its interesting geometry, better overall performance, and the potential to be a successful design for real implementation. In terms of geometry, the first design is not too geometrically interesting, as it only differs in the position of one propeller when compared with the seed design. Regarding performance, the second design indeed outperforms all three designs, as it achieves a higher score on the oval test and almost full scores for all the other tests. The third design has a potential difficulty regarding a physical implementation since the propeller which is placed close to the center could exert a force on the center place. Our simulator does not consider such interaction between components. This point reinforces the notion that the interaction between designers and design tools is crucial. The designer with domain knowledge knows the limitation of the oracle and can foresee the possible failure of the design, while the design tools are able to explore vast regions of the design space, sometimes yielding surprising designs. Through the interaction between designer and the optimization tools, we can reject some design that an expert could consider problematic but which the simulator accepts, and then prioritize the exploration effort for promising designs.

6 CONCLUSIONS

We introduced a symbiotic methodology for the design-space exploration of CPS design. To the best of our knowledge, this is the first work on the design methodology that allows direct interaction between the designers and the automated tools. With the close interaction between designers and automated tools, the design exploration is iteratively guided by the domain knowledge of the designers. The design choice allows the designer to guide the exploration to discover unconventional designs and optimize the system performance. We illustrate the methodology by applying it to a robot car configuration optimization problem and a UAV design problem. The results show that our symbiotic methodology can efficiently and effectively discover unconventional design and optimize the objective for the design problem, and thus is promising for reducing the design cycle for complex CPS design.

ACKNOWLEDGMENTS

This work is supported by the DARPA LOGiCS project under contract FA8750-20-C-0156.

REFERENCES

[1] J. Bakakeu, J. Fuchs, T. Javied, M. Brossog, J. Franke, H. Klos, W. Eberlein, S. Tolksdorf, J. Peschke, and L. Jahn. 2018. Multi-Objective Design Space Exploration for the Integration of Advanced Analytics in Cyber-Physical Production Systems. In *2018 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)*. 1866–1873. https://doi.org/10.1109/IEEM.2018.8607483

[2] Albert Benveniste, Benoît Caillaud, Dejan Nickovic, Roberto Passerone, Jean-Baptiste Raclet, Philipp Reinkemeier, Alberto Sangiovanni-Vincentelli, Werner Damm, Thomas A. Henzinger, and Kim G. Larsen. 2018. Contracts for System Design. *Foundations and Trends in Electronic Design Automation* 12, 2-3 (2018), 124–400. https://doi.org/10.1561/1000000053

[3] DARPA [n. d.]. Symbiotic Design for Cyber Physical Systems. https://www.darpa.mil/program/symbiotic-design-for-cyber-physical-systems. Accessed: 2022-05-20.

[4] Patricia Derler, Edward A. Lee, Stavros Tripakis, and Martin Törngren. 2013. Cyber-Physical System Design Contracts. In *Proceedings of the ACM/IEEE 4th International Conference on Cyber-Physical Systems* (Philadelphia, Pennsylvania) *(ICCPS '13)*. Association for Computing Machinery, New York, NY, USA, 109–118. https://doi.org/10.1145/2502524.2502540

[5] John Finn, Pierluigi Nuzzo, and Alberto Sangiovanni-Vincentelli. 2015. A mixed discrete-continuous optimization scheme for Cyber-Physical System architecture exploration. In *2015 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. 216–223. https://doi.org/10.1109/ICCAD.2015.7372573

[6] John Fitzgerald, Carl Gamble, Richard Payne, and Benjamin Lam. 2017. Exploring the Cyber-Physical Design Space. *INCOSE International Symposium* 27, 1 (2017), 371–385. https://doi.org/10.1002/j.2334-5837.2017.00366.x arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/j.2334-5837.2017.00366.x

[7] Srivatsan Krishnan, Zishen Wan, Kshitij Bhardwaj, Paul Whatmough, Aleksandra Faust, Sabrina Neuman, Gu-Yeon Wei, David Brooks, and Vijay Janapa Reddi. 2021. AutoPilot: Automating SoC Design Space Exploration for SWaP Constrained Autonomous UAVs. arXiv:2102.02988 [cs.RO]

[8] Edward A. Lee. 2015. The Past, Present and Future of Cyber-Physical Systems: A Focus on Models. *Sensors* 15, 3 (2015), 4837–4869. https://doi.org/10.3390/s150304837

[9] Edward A. Lee. 2016. Fundamental Limits of Cyber-Physical Systems Modeling. *ACM Trans. Cyber-Phys. Syst.* 1, 1, Article 3 (Nov. 2016), 26 pages. https://doi.org/10.1145/2912149

[10] Azad M. Madni and Michael Sievers. 2014. Systems Integration: Key Perspectives, Experiences, and Challenges. *Systems Engineering* 17, 1 (2014), 37–51. https://doi.org/10.1002/sys.21249 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/sys.21249

[11] Sangeun Oh, Yongsu Jung, Seongsin Kim, Ikjin Lee, and Namwoo Kang. 2019. Deep generative design: Integration of topology optimization and generative models. *Journal of Mechanical Design* 141, 11 (2019).

[12] Alberto Sangiovanni-Vincentelli. 2007. Quo Vadis, SLD? Reasoning About the Trends and Challenges of System Level Design. *Proc. IEEE* 95, 3 (2007), 467–506. https://doi.org/10.1109/JPROC.2006.890107

[13] Alberto Sangiovanni-Vincentelli, Werner Damm, and Roberto Passerone. 2012. Taming Dr. Frankenstein: Contract-Based Design for Cyber-Physical Systems*. *European Journal of Control* 18, 3 (2012), 217–238. https://doi.org/10.3166/ejc.18.217-238

[14] Janos Sztipanovits, Xenofon Koutsoukos, Gabor Karsai, Nicholas Kottenstette, Panos Antsaklis, Vijay Gupta, Bill Goodwine, John Baras, and Shige Wang. 2012. Toward a Science of Cyber–Physical System Integration. *Proc. IEEE* 100, 1 (2012), 29–44. https://doi.org/10.1109/JPROC.2011.2161529

[15] James D. Walker, F. Michael Heim, Bapiraju Surampudi, Pablo Bueno, Alexander Carpenter, Sidney Chocron, Jon Cutshall, Richard Lammons, Theodore Bapty, Brian Swenson, and Sydney Whittington. 2022. A Flight Dynamics Model for Exploring the Distributed Electrical eVTOL Cyber Physical Design Space. In *2022 IEEE Workshop on Design Automation for CPS and IoT (DESTION)*. 7–12. https://doi.org/10.1109/DESTION56136.2022.00008