2022 IEEE Workshop on Design Automation for CPS and IoT (DESTION) | 978-1-6654-7040-7/22/$31.00 ©2022 IEEE | DOI: 10.1109/DESTION56136.2022.00012

Combinatorial and Parametric Gradient-Free Optimization for Cyber-Physical System Design

Hongrui Zheng†, Johannes Betz†, Arun Ramamurthy‡, Hyunjee Jin ‡, and Rahul Mangharam†

***Abstract*—The design and evaluation of cyber-physical systems are complex as it includes mechanical, electrical, and software components leading to a high dimensional space for architectural search and parametric tuning. For each new design, engineers need to define performance objectives, capture data from previous designs, make a model-based design, and then develop and enhance each system in each iteration. To address this problem, we present a combinatorial and parametric design space exploration and optimization technique for automatic design creation. We leverage gradient-free methods to jointly optimize the multiple domains of the cyber-physical systems. Finally, we apply this method in a DARPA design challenge where the goal is to create new designs for unmanned aerial vehicles. We evaluate the new designs on performance benchmarks and demonstrate the effectiveness of gradient-free optimization techniques in automatic design creation.**

***Index Terms*—architectural design, design exploration, design optimization, modelling, simulation**

I. Introduction

Over the past few decades, there has been a drastic change to how complex systems, such as aircraft, automobiles, gas turbines, etc., are designed. The integration of computational, control, and software in the design process has resulted in products that demonstrate highly interrelated subsystems and components, resulting in what is known as Cyber-Physical Systems (CPS). Although the complexity of systems has grown exponentially, the way we design them is still fundamentally the same. The design processes still rely on traditional systems engineering techniques following the iterative process of requirements elicitation, requirements decomposition, design synthesis, fabrication and finally followed by verification and validation.

While recent developments [1] [2] have attempted to address some of these CPS design process issues, there are still some open and unsolved challenges. For example, currently, good CPS designs are mainly created by the expertise of human engineers. This expertise is acquired by working in a certain engineering domain space for multiple years, the knowledge of previous designs, and the ability to exchange knowledge with other domains. In addition, current CPS designs are unable to automatically discover new, innovative, and high performance design solutions due to cost and time constraints.

To overcome these challenges, techniques for machine-based design creation and optimization are developed where the design process gets largely automated. We demonstrate the applications of these automated exploration techniques in a DARPA-organized CPS Design Challenge (hackathon). The goal was to automatically create new designs of an unmanned aerial vehicle (UAV) that is tasked with the successful completion of a set of missions that stress the agility, endurance and accuracy of the UAV designs (Figure 1). The purpose of the design optimization is to maximize the performance of the UAV by exploring different architectures, configurations, parametric designs and control settings.

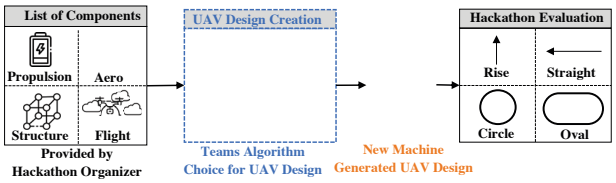

Fig. 1. Design challenge hackathon overview

This paper introduces the solutions, methods, and results utilized to solve the DARPA design challenge. This work has three primary contributions:

1) We provide three different strategies for multi-domain design exploration using a gradient-free, deterministic, parallel optimization pipeline to derive new and innovative designs for CPS.
2) Based on a set of input seed designs we apply these strategies to infer and optimize design spaces of UAVs to create designs that fulfill mission-specific objectives.
3) We provide detailed insights in both the use of the gradient-free optimization across discrete mechanical and electrical components and continuous control choices as well as the final output of the new UAV designs.

II. Related Work

Generative Design The topics of this paper are related to the field of *Generative Design*. This describes a design exploration process where the design goals and constraints are defined by engineers and a software [3]. In the field of *topology optimization* Than et al. [4] proposed an iterative six-step generative procedure that allows designing the upper arm of a Delta robot using the topology optimization method. Similarly for medical devices, [5] presented a 3D structural topology optimization to create a new and innovative design of an aneurysm implant Herath et al. [6] displayed an accelerated

†University of Pennsylvania, School of Engineering and Applied Science, 19106 Philadelphia, PA, USA hongruiz, joebetz, rahulm@seas.upenn.edu

‡Siemens Corporate Research, Princeton, PA, USA arun.ramamurthy, hyunjee.jin@siemens.com

978-1-6654-7040-7/22/$31.00 ©2022 IEEE
DOI 10.1109/DESTION56136.2022.00012

topology optimization technique based on deep learning methods. With the usage of Conditional Generative Adversarial Networks (cGANs), the authors predict the optimal topology of a given structure subject based on a given set of input parameters. In order to speed up the creation of new designs Usama et al. [7] proposed a data-driven design pipeline for fast design exploration to design new vehicles. The authors structured the generative design method in a *qualitative search*, that first builds up a lower-dimensional representation of a given design space, and a *quantitative search*, that explores the design space to find an optimal design in terms of performance criterion (e.g. drag coefficient). Similar efforts can be observed in the works of [8], [9], [10], [11] among a host of other publications in the use of generative models (GANs) or other AI-methods (convolutional neural networks).

CPS Design Space Exploration Bradley and Atkins [12] surveyed the run-time cooperative optimization and prove that considering both cyber and physical components in a co-optimization and co-regulation schemes has an advantage for the design of mobile robotic and vehicle systems. In the work of [13], the authors showed how domain knowledge can be used to guide the design-space exploration process for an advanced control system and its deployment on embedded hardware. In [14] the authors presented a design space exploration where they choose a real non-linear inverted pendulum as a demonstration example that has a wide range of physical and cyber settings that can be adjusted. The authors present a method that has parameterizable physical models and automatic recalculation of control algorithm parameters for the explored systems. [15] presented an approach to automatically synthesize both the hardware and the controller parameters for modular robots. This synthesis is mainly based on the task the robot should fulfill. The results showed that such a synthesis can outperform genetic algorithm optimizations. Finally, Lin et al. [16] presented a formalization of the design constraints of building an autonomous driving system in terms of performance, predictability, storage, thermal, and power. This work was presented with a focus on the computation hardware only (ECU implemented with CPU, GPU, or FPGA). The same task was displayed in [17] where computation platforms are optimized under latency and cost.

CPS Design Space Optimization The use of model-based systems engineering (MBSE) methods for the design of CPS saw fruition with the DARPA AVM [1] and the META program. The program resulted in the OpenMETA design framework [2] that offers capabilities to not only model the CPS domains using MBSE techniques, but also explore these disparate design spaces jointly. To reduce the design space and improve the time of the optimization, in [18] an effective design space ascertained algorithm based on design optimization parameter steering mechanism is proposed. Ren et al. [19] approach the problem of designing aircraft and wind turbines. In this work an iterative updating and re-optimization of a fast physics-based replacement model, following a surrogate-based optimization paradigm is present. Lei et al. [20] presented a comparative study on different types of robust design optimization methods for the creation of new electrical machines. Finally, the optimization techniques used in this paper is an effort to generalize the *TUNERCAR* toolchain presented in [21].

III. Methodology

A. Problem Statement

We first define the design of a UAV as a directed multi-graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. Where $\mathcal{V}$ is the set of vertices in the graph representing the components of the UAV and $\mathcal{E}$ is the set of edges in the graph which represent the connections between the components. In our experiments, the nodes in graph $\mathcal{G}$ could be one of these components: batteries, motors, propellers, wings, servos, tubes, flanges, plates or hub connectors. The edges in graph $\mathcal{G}$ could be mechanical assemblies (concentric or coincident joints), electrical connections or logical links such as a software socket that transmits data from a sensor to a computer (logical connection) or an assembly mating joint (physical connection).

A multi-directed graph is chosen as the representation of the design due to the possibility of the existence of multiple connections between a set of components. This representation follows the concepts introduced in [2]. We define the objective function $f : \mathcal{G} \rightarrow \mathbb{R}$ that maps the input graph $\mathcal{G}$ to a real numbered score representing the design's performance on designated benchmarks. The problem objective is then formulated as,

$$\arg\max_{\mathcal{G}} \left(f(\mathcal{G})\right) \tag{1}$$

To solve this problem, we propose two functions. Firstly, a function $g(\mathcal{G}_{\text{init}}, \{s_i\}) \rightarrow \mathcal{G}$. g takes an initialized graph $\mathcal{G}_{\text{init}}$, and a set of specifications $\{s_i\} \in S$, where S is the set of all valid specifications, as input, and outputs a fully defined graph $\mathcal{G}$ as output. Specifications is a mapping that assigns an available component, or a parameter, to every node and edge in the graph $\mathcal{G}$. The overall structure, including the number of nodes and edges, and the connections between them are defined in $\mathcal{G}_{\text{init}}$. The set of specifications fully defines the type and adjustable parameters of each vertex and edge. g then builds upon the input graph by fully defining the graph with the set of specifications. In this case, the objective of the problem then becomes:

$$\arg\max_{\{s_i\} \in S} \left(f(g(\mathcal{G}_{\text{init}}, \{s_i\})\right) \tag{2}$$

Secondly, a function $h(\mathcal{V}_0, \{a_i\}, m) \rightarrow \mathcal{G}$ that generates a graph iteratively. $\mathcal{V}_0$ is the root node of the graph. $\{a_i\} \in A$ is a set of actions that describes adding a specific node (alongside an edge) to the graph, where A is all allowed actions and $m \in \mathbb{Z}^+$ is the number of iterations allowed. Thus, in this case, the objective of the problem becomes:

$$\arg\max_{\{a_i\} \in A} \left(f(h(\mathcal{V}_0, \{a_i\}, m))\right) \tag{3}$$

A set of seed designs, $\mathcal{G}_{\text{init}}$, created by human designers are provided to our algorithms to warm-start the learning process. Each seed design completely specifies the design, i.e., the

location, number, and type of nodes and edges in the graph are fully defined.

In addition to the seed designs, a corpus of components that can be used to construct any subsequent design is provided. The components are tagged with their characteristics (e.g., batteries are tagged with their capacity, mass, discharge rates, chemistry, etc.). In our experiments, a specification set $\{s_i\}$ explicitly defines which component is selected out of all valid components at each node. The action sequence $\{a_i\}$ defines the action of adding a component to the graph and establishing a connection to the existing incomplete gr

B. Proposed Solutions

We propose three different strategies to tackle search and the optimization problem, each using gradient-free optimization techniques to automatically create new UAV designs.

1) **Holistic optimization** from seed design on the entire parameter space as displayed in Figure 2. In this approach, we use (1+λ)-EAs (Evolutionary Algorithm). This algorithm follows the basic sketch of EAs and is applied with different mutation rates to perform the optimization on the entire search space of $\{s_i\}$. The EAs serve as the function g where by iteratively creating new specification vectors using mutation and evaluating them. A vector that maximizes the score within the given computational budget is eventually found. The objective used in this setting is the sum of all four scores provided by the benchmarks in the simulation.
2) **Sequential optimization** from seed design separating the combinatorial and parametric portions of the design space as displayed in Figure 3. With this strategy, we first split the search space into $\{s_{i,\text{comb}}\}$ and $\{s_{i,\text{control}}\}$. We then optimize the discrete parameters (propulsion, aerodynamic,and structural systems) $\{s_{i,\text{comb}}\}$ by using the aforementioned (1+λ)-EAs. For the continuous parameters (autopilot flight system) $\{s_{i,\text{control}}\}$, we use Differential Evolution (DE), which also lies under the EA category of algorithms, with TwoPoint crossover [22], [23]. Here the function g is a combination of both the (1+λ)-EA and the DE. In the first stage, g is the (1+λ)-EA. Then after finding the best specification set $\{s_{i,\text{comb}}\}$, a new $\mathcal{G}_{\text{init}}$ with specifications $\{s_{i,\text{comb}}\}$ is used as the seed design for the second stage. In the second stage, g is the DE, where the final specification including the controller parameters are found.
3) **Architecture topology generation** where certain rules on connections and composition of an UAV system are enforced as displayed in Figure 4. This approach tries to directly create the function h. We assign an integer to an allowed action and cap the length of the vector. Thus optimizing this action sequence becomes a similar problem of finding the combinatorial vector in solution (2). Similarly, we first use the (1+λ)-EAs to optimize the fixed-length vector $\{a_i\}$ to find a sequence of actions that achieve high scores on the trim response objective. Then we use the DE method to find a set of control parameters for the design on the four benchmark scores. The function h is then again a weighted combination of the EA and the DE.

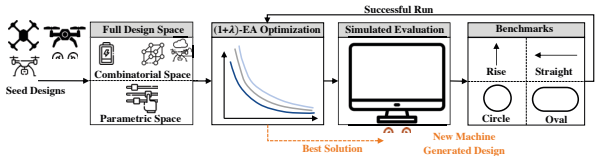

Fig. 2. Optimization pipeline Solution 1: Holistic optimization

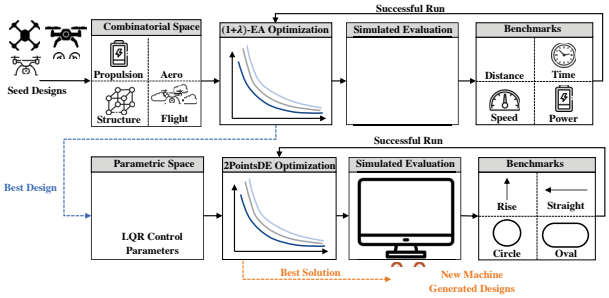

Fig. 3. Optimization pipeline Solution 2: Sequential optimization

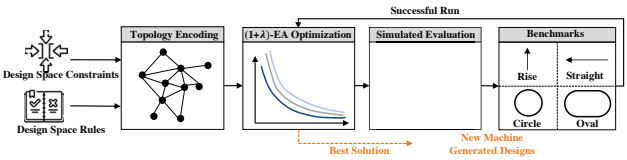

Fig. 4. Optimization pipeline Solution 3: Architecture topology generation

IV. CPS Design Challenge Experiments

A. Simulation Setup

To evaluate and benchmark the newly created designs we use a flight dynamics simulation developed in Fortran that implements a full 6 DOF flight physics based on rigid-body dynamics [24], [25]. The simulation runs in 2 flight modes: the *Trim* mode computes the steady-flight cruise state of the input design; the *Benchmark* mode computes a flight along with a set of defined paths.

An input file generated by the OpenMETA [2] toolchain with a Creo [26] CAD model of the design is used by the flight dynamics simulation. To circumvent the large overhead generated by the OpenMETA toolchain, we use the Python API provided by FreeCAD [27] to efficiently generate different designs and provide necessary information to the simulation (overall mass, moments of inertia, projected area, etc.). We then call the flight dynamics simulation directly as a subprocess as part of our optimization pipeline.

B. Search Space

Next, we define the design space that we are searching in. The algorithms are tasked with searching for UAV designs that score high on the provided benchmarks. Each UAV is defined to have the following CPS components:

- Propulsion: Batteries, ESCs, motors, propellers
- Aerodynamic: Wings, servos
- Structural: Tube, flange, plate, hub connectors
- Flight: Parameters for the autopilot

The overall architectural design follows several simple rules on what components are must-haves in a UAV, and rules on which components can connect, but the number of components on any design is unlimited. For the propulsion, aerodynamic, and structural components, the design space is combinatorial. For the flight system, the design space is parametric. For each selected structural component, parametric design parameters related to the sizes, shape, and positioning of the component can be altered continuously within a predefined design limit. The exact size of the design space and its components is specified in Appendix Table III.

C. Objective Functions

Lastly, we define the objective (introduced in Section III-A as f) of the design problem. In addition to the original scoring criteria specified in the design challenge, we propose an intermediate set of scoring criteria. These better evaluate the performance of a UAV on the dynamics level, where better scores are achieved by designs that are dynamically more capable and nimble. As per the requirements of the design challenge, all benchmarks terminate when one battery reaches 20 percent capacity. Designs are evaluated in the *Benchmark* mode of the simulation. The testing specification of each task in *Benchmark* mode and possible scores of each task are listed in Appendix Table IV. And the specific breakdown of the scoring criteria is as follows:

- For **Benchmark 1** (rise and hover), if the design can hover for 200 seconds and reach a height of 150 meters, it'll score 200 points. Additionally, each extra second the UAV is able to hover there is an additional 1 point per second, up to a maximum of 400 points in total. Failing the hover benchmark also means failing the entire suite of following benchmarks. Further, a penalty is introduced when the vehicle violates the 150-meter height requirement. Thus, the vehicle has to be stable at the requested 150-meter height for at least 400 seconds to score full points on this benchmark.
- For **Benchmark 2** (straight line), a level steady flight of over 2000 meters in a straight line is required to start scoring. Then, the design gets 1 point per 10 meters (including the 2000 meters prior) in flight. A penalty of 10 times the lateral error in meters is introduced to penalize designs that do not exhibit stable flights.
- For **Benchmark 3** (circle), the design is required to traverse a circle of diameter 1000 meters. Upon finishing flying the circle, the design gets 300 points. A penalty of 50 times the lateral error in meters is added for deviations from the circular path.
- For **Benchmark 4** (racing oval), the design is required to fly an oval consisting of two 750 meter straights, and two 600 meter diameter half circles. Finishing the course grants 200 points, and for every second less than 350 seconds taken to complete the course, an extra point is granted. Designs that exhibit poor path following capability are penalized significantly. If the deviation from the planned path exceeds 10 meters, the design gets 0 points.

To accelerate the search across the architecture space, we introduce a set of intermediate scoring criteria. The designs are evaluated in the *Trim* mode of the simulation. The testing specification of each task in *Trim* mode and scoring of each task are listed in Appendix Table V. These intermediate criteria act as proxy for the original benchmarks and has better stability in the optimization process. For example, having a high flight distance forward is a good proxy for the Straight Line benchmark, and allows for unlimited maximum scores beyond the full score of the original benchmarks.

D. Designed Experiments

We perform the following experiments to compare the performance of our proposed solutions on the design challenge and to investigate what elements of the proposed solution impacts the efficiency and performance of found solutions. The details of the gradient-free optimizers used can be found in the Appendix Section A.

1) **Experiment**: Analyze the performance of the proposed solution (1). Here we test the performance of four (1+λ)-EAs with different mutation rates on improving a quadcopter seed design.
2) **Experiment**: Analyze the performance of the proposed solution (2). Here we first test the performance of four (1+λ)-EAs with different mutation rates on improving the existing UAV design on the trim response scores and continue with TwoPoints DE to improve the continuous control parameters of each of the designs.
3) **Experiment**: Compare the performance of proposed solution (2) with different population sizes. In this experiment, we use the (1+λ)-EA with the most widely used mutation rate $1/d$.

V. Experimental Results

In the following section, we show the results of each experiment. A list of all the results as sum of individual scores with variation in the optimizer parameters is shown in Table I. Since we use a surrogate evaluation pipeline, we also evaluate the best designs through the original OpenMETA pipeline (shown in Table I in the last column).

In Figure 5, we show the progression of the best-evaluated design of each generation using Solution (1) where all parameters are optimized at the same time. Multiple designs achieve the best benchmark score of 700, by achieving full scores on Benchmarks 1 and 2. The optimizers can find solutions that can rise, hover, and fly forward, but weren't able to find any designs that could follow curves. We also show the comparison of using an EA and performing a random search. It is clear that even though the result found by this Solution does not achieve good scores on the benchmarks the algorithm demonstrates better results than random searches.

TABLE I
FULL RESULTS OF ALL DESIGNED EXPERIMENTS

Experiment	Optimizer	Budget (# evals)	Best *Trim* Score	Best *Benchmark* Score
Holistic	Discrete (1+λ)-EA	19200	N/A	700
Holistic	Lengler (1+λ)-EA	19200	N/A	700
Holistic	Portfolio (1+λ)-EA	19200	N/A	699
Holistic	FastGA (1+λ)-EA	19200	N/A	700
Holistic	RandomSearch	19200	N/A	400
Sequential	Discrete (1+λ)-EA + 2ptsDE	9600+9600	-24937.41	1489, 1565[3]
Sequential	Lengler (1+λ)-EA + 2ptsDE	9600+9600	-32583.37	1100
Sequential	Portfolio (1+λ)-EA + 2ptsDE	9600+9600	-27999.26	0
Sequential	FastGA (1+λ)-EA + 2ptsDE	9600+9600	-26213.04	1100
Sequential	RandomSearch + RandomSearch	9600+9600	-30660.95	400
Sequential, $\lambda = 8$	Discrete (1+λ)-EA + 2ptsDE	9600+9600	-30177.3	1089
Sequential, $\lambda = 16$	Discrete (1+λ)-EA + 2ptsDE	9600+9600	-25055.92	1489
Sequential, $\lambda = 32$	Discrete (1+λ)-EA + 2ptsDE	9600+9600	-24937.41	1489, 1565[3]
Sequential, $\lambda = 64$	Discrete (1+λ)-EA + 2ptsDE	9600+9600	-24937.41	1489, 1565[3]
Sequential, $\lambda = 128$	Discrete (1+λ)-EA + 2ptsDE	9600+9600	-24937.41	1489, 1565[3]
Topology Generation	Discrete (1+λ)-EA + 2ptsDE	9600+9600	-53228.69	N/A, 1556[3]

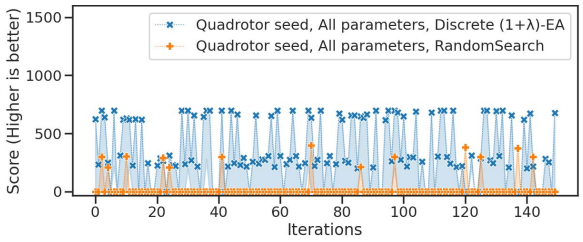

Fig. 5. Results of experiment 1: Optimizing on all parameters using discrete (1+λ)-EA ($1/d$ mutation rate). Maximum (best) score of each generation is shown with a check mark.

In Figure 6 and 7, we show the the progression of best designs on the trim response metrics, and best control parameters found for the best design using solution (2). In Figure 6, we show the designs in each generation with the minimum objective score on the trim response metrics marked by checkmarks. The error band in the figure shows the range of scores achieved by designs in the same generation.

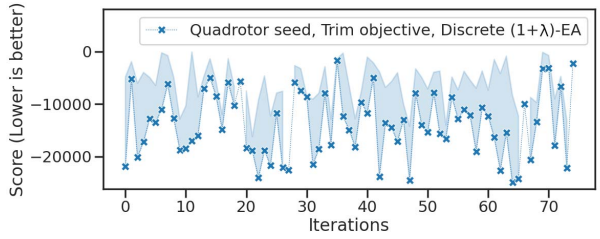

Fig. 6. Results of experiment 2: Optimizing discrete parameters using Discrete (1+λ)-EA ($1/d$ mutation rate).The Minimum (best) score of each generation is shown with a check mark.

In Figure 7, we show the performance of solution (2) in the second stage when control parameters are tuned. We also show the comparison with solution (1) and random search. Average

[3]First score: Python pipeline; Second score: OpenMETA pipeline.

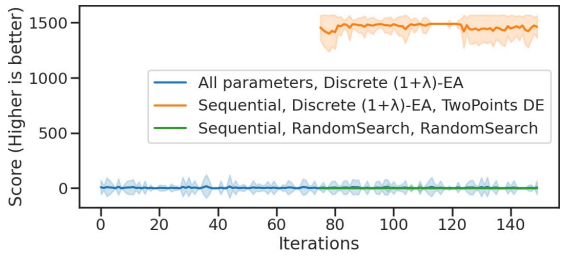

Fig. 7. Results of experiment 2: Continuous parameters using TwoPoints DE using best design from Figure 6. The figure shows the average of each generation compared to results from Experiment 1 and Random Search.

scores of each generation with standard deviation as error bands are shown. Although the all-parameter-optimization was able to find some designs that score above 600 points, the majority of designs explored are either invalid or do not score. Thus we see the average of all performances in a population hovering around zero. In comparison, the best design from the trim response optimization readily scores 1100 points (full score on the first three benchmarks) without much tuning. At the end of the optimization, a design is found to achieve the best score at 1489 points out of 1582 possible points. We also show the 3D model of the design found by this solution in Figure 8. There is a noticeable asymmetric design choice present where one of the propellers has a smaller diameter than the others.

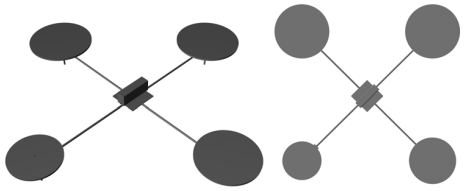

Fig. 8. 3D-model of the highest scoring design from Experiment 2.

Next, we study the effect of different population size during both stages of the optimization in solution (2). As shown in Figure 9, the best scoring design in trim responses is found when $\lambda = 8$.

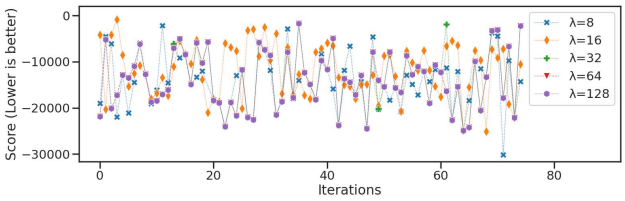

Fig. 9. Results of experiment 3: Effect population sizes in each generation in (1+λ)-EA during trim response optimization. Minimum (best) scores shown at each generation with marker.

However, when we move to the next stage of optimization, $\lambda = 8$ has the worst results on the final benchmark scores when trying to track the four designated paths. Upon further inspection, the design found by this setting in the trim response optimization stage has a higher maximum lateral velocity compared to others. However, finding a set of controller parameters that's able to maintain a low error while tracking an oval at the high lateral velocity becomes too challenging for the optimizer in the second stage. This is why we see designs found by $\lambda = 8$ hovers at around 1100 points in the final stage of the optimization, never achieving a non-zero score for the last benchmark. For all other values of λ, the max lateral velocity found limits the highest score achievable on the last benchmark again. 1489 is again the highest scoring design in all explored designs. Another observation is that for λ greater than 32, experiments all found the same best design.

Lastly, we show the results from solution (3), topology generation. The best design overall is found in this experiment. On the intermediate trim benchmark, it scored the highest out of all designs, with a maximum lateral velocity of 50 m/s. We show the 3D models of the 3 best designs found during the first stage of optimization in Figure 10. All three designs were able to achieve impressive intermediate benchmark scores from trim responses and exhibit interesting design choices.

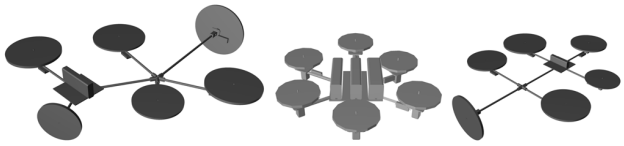

Fig. 10. Three highest scoring designs on trim responses scores from Experiment 4 (from left to right: -53228.69, -40707.43, -19572.00).

VI. Discussion

The results presented show that the proposed method of combinatorial and parametric system design optimization based on gradient-free methods is generally functional. Unfortunately, with solutions (1) and (2) and their respective experiments we can see that using a seed design has its limitations. Although we're able to evaluate tens of thousands of designs in a short amount of time and find a design that is close to achieving the design requirement, the approach still leaves a lot more to be desired. For example, the seed design's topology is purely created based on human intuition and not on specific requirements for the UAV's flight dynamics. This means that a bad design as a starting point for the optimization does not lead to better outcomes afterward. Future research could focus on how to automatically generate seed designs that take specific design requirements into account. Afterward the same optimization techniques could be applied to find the best component specifications on the seed design.

In solution (3), although we formulate the vector to be optimized as a sequence of actions, the optimization doesn't take the nature of a sequence into account. The previous action in the sequence is not influenced by its predecessors and does not influence its successors. Considering the history of actions opens up new possibilities where the intentions of human designers could potentially be modeled. The formulation of the function h in section III-A used by solution (3) provides the prime opportunity to replace the fixed length, EA-optimized vector by a sequence from a recurrent neural network or a transformer. Sequence generation is the strong suit of these approaches that intrinsically model history.

Conclusion

In this paper, we present three solutions for architectural exploration and parameter optimization to create new designs for UAVs that meet the requirements set by the DARPA CPS Design Challenge. In the first solution, all design parameters are optimized at once. In the second solution, the search is split into combinatorial and parametric sub-spaces. The combinatorial parameters are first optimized on an intermediate benchmark. Then the parametric parameters of the best design are optimized on the final overall benchmarks. In the third solution, graphs are generated from a root node using a sequence of actions. We were able to demonstrate that all solutions result in feasible designs while solution (3) provides the best overall designs that fulfill the mission goals. Finally, with this research, we were able to demonstrate that gradient-free optimization techniques can be used for the automatic design creation of CPS based on provided seed designs.

VII. Acknowledgments

This research was supported in part by the Defense Advanced Research Projects Agency (DARPA) under Contract FA8750-20-C-0542 (Systemic Generative Engineering). The views, opinions, and/or findings expressed are those of the author(s) and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government. We thank the SwRI team (Ms. Sydney Wittington, Dr. James Walker) and Vanderbilt team (Dr. Theodore Bapty) for developing the UAV simulation framework as part of the DARPA design challenge.

REFERENCES

[1] "Darpa avm," 2021. [Online]. Available: https://www.darpa.mil/program/adaptive-vehicle-make

[2] J. Sztipanovits, T. Bapty, S. Neema, L. Howard, and E. Jackson, "Openmeta: A model-and component-based design tool chain for cyber-physical systems," in *From programs to systems. The systems perspective in computing*. Springer, 2014, pp. 235–248.

[3] "Generative design: Siemens," 2021. [Online]. Available: https://www.plm.automation.siemens.com/global/en/our-story/glossary/generative-design/27063

[4] T. H. T. Tran, D. S. Nguyen, N. T. Vo, and H. N. Le, "Design of delta robot arm based on topology optimization and generative design method." IEEE, Nov. 2020. [Online]. Available: https://doi.org/10.1109/gtsd50082.2020.9303083

[5] L. Jiang, S. Chen, C. Sadasivan, and X. Jiao, "Structural topology optimization for generative design of personalized aneurysm implants: Design, additive manufacturing, and experimental validation." IEEE, Nov. 2017. [Online]. Available: https://doi.org/10.1109/hic.2017.8227572

[6] S. Herath, U. Haputhanthri, and C. Mallikarachchi, "Initial design of trusses using topology optimization in a deep learning environment." IEEE, Dec. 2020. [Online]. Available: https://doi.org/10.1109/fiti52050.2020.9424894

[7] M. Usama, A. Arif, F. Haris, S. Khan, S. K. Afaq, and S. Rashid, "A data-driven interactive system for aerodynamic and user-centred generative vehicle design." IEEE, Apr. 2021. [Online]. Available: https://doi.org/10.1109/icai52203.2021.9445243

[8] Y. Wang, K. Shimada, and A. B. Farimani, "Airfoil gan: Encoding and synthesizing airfoils foraerodynamic-aware shape optimization," *arXiv preprint arXiv:2101.04757*, 2021.

[9] W. Chen, K. Chiu, and M. D. Fuge, "Airfoil design parameterization and optimization using bézier generative adversarial networks," *AIAA Journal*, vol. 58, no. 11, pp. 4723–4735, 2020.

[10] W. Chen and F. Ahmed, "Mo-padgan: Reparameterizing engineering designs for augmented multi-objective optimization," *Applied Soft Computing*, vol. 113, p. 107909, 2021.

[11] W. Jing, L. Runze, H. Cheng, C. Haixin, R. CHENG, Z. Chen, and M. ZHANG, "An inverse design method for supercritical airfoil based on conditional generative models," *Chinese Journal of Aeronautics*, 2021.

[12] J. Bradley and E. Atkins, "Optimization and control of cyber-physical vehicle systems," *Sensors*, vol. 15, no. 9, pp. 23 020–23 049, Sep. 2015. [Online]. Available: https://doi.org/10.3390/s150923020

[13] Y. Vanommeslaeghe, J. Denil, J. D. Viaene, D. Ceulemans, S. Derammelaere, and P. D. Meulenaere, "Leveraging domain knowledge for the efficient design-space exploration of advanced cyber-physical systems," in *2019 22nd Euromicro Conference on Digital System Design (DSD)*. IEEE, Aug. 2019. [Online]. Available: https://doi.org/10.1109/dsd.2019.00058

[14] H. M. Buini, S. Peter, and T. Givargis, "Including variability of physical models into the design automation of cyber-physical systems," in *Proceedings of the 52nd Annual Design Automation Conference*, ser. DAC '15. New York, NY, USA: Association for Computing Machinery, 2015. [Online]. Available: https://doi.org/10.1145/2744769.2744857

[15] T. Campos, J. P. Inala, A. Solar-Lezama, and H. Kress-Gazit, "Task-based design of ad-hoc modular manipulators," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 6058–6064.

[16] S.-C. Lin, Y. Zhang, C.-H. Hsu, M. Skach, M. E. Haque, L. Tang, and J. Mars, "The architectural implications of autonomous driving," *ACM SIGPLAN Notices*, vol. 53, no. 2, pp. 751–766, Nov. 2018. [Online]. Available: https://doi.org/10.1145/3296957.3173191

[17] A. Collin, A. Siddiqi, Y. Imanishi, E. Rebentisch, T. Tanimichi, and O. L. Weck, "Autonomous driving systems hardware and software architecture exploration: optimizing latency and cost under safety constraints," *Systems Engineering*, vol. 23, no. 3, pp. 327–337, Dec. 2019. [Online]. Available: https://doi.org/10.1002/sys.21528

[18] O. Hai-ying, L. Xiao-yu, F. Zhan-ping, and H. Wen-ting, "Design optimization parameter steering method applied in exploring an effective design space." IEEE, Nov. 2010. [Online]. Available: https://doi.org/10.1109/icalip.2010.5684976

[19] J. Ren, "Aerodynamic shape optimization by multi-fidelity modeling and manifold mapping," Ph.D. dissertation, 2016. [Online]. Available: https://doi.org/10.31274/etd-180810-4667

[20] G. Lei, G. Bramerdorfer, C. Liu, Y. Guo, and J. Zhu, "Robust design optimization of electrical machines: A comparative study and space reduction strategy," vol. 36, no. 1, pp. 300–313, Mar. 2021. [Online]. Available: https://doi.org/10.1109/tec.2020.2999482

[21] M. O'Kelly, H. Zheng, A. Jain, J. Auckley, K. Luong, and R. Mangharam, "TUNERCAR: A superoptimization toolchain for autonomous racing," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, May 2020. [Online]. Available: https://doi.org/10.1109/icra40945.2020.9197080

[22] R. Storn and K. Price, "Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces," *Journal of global optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[23] J. H. Holland *et al.*, *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. MIT press, 1992.

[24] B. L. Stevens, F. L. Lewis, and E. N. Johnson, *Aircraft control and simulation: dynamics, controls design, and autonomous systems*. John Wiley & Sons, 2015.

[25] M. E. Dreier, *Introduction to helicopter and tiltrotor flight simulation*. American Institute of Aeronautics and Astronautics, 2007.

[26] PTC, "Creo," 2021. [Online]. Available: https://www.ptc.com/en/products/creo

[27] FreeCAD, "Freecad." [Online]. Available: https://www.freecadweb.org/

[28] J. Lengler and A. Steger, "Drift analysis and evolutionary algorithms revisited," *Combinatorics, Probability and Computing*, vol. 27, no. 4, pp. 643–666, 2018.

[29] D.-C. Dang and P. K. Lehre, "Self-adaptation of mutation rates in non-elitist populations," in *International Conference on Parallel Problem Solving from Nature*. Springer, 2016, pp. 803–813.

[30] B. Doerr, H. P. Le, R. Makhmara, and T. D. Nguyen, "Fast genetic algorithms," in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2017, pp. 777–784.

APPENDIX

A. Optimizers

In the following section, we introduce the two gradient free optimizers mentioned in III-B to jointly explore the discrete and the continuous portions of the parameter space.

For the discrete parameters, we use a (1+λ) Evolutionary Algorithm (EA). This algorithm follows the basic sketch of EAs. First, an initial population is randomly generated and evaluated on the given metric. Then the fittest individuals are selected as parents of following generations. New individuals are then generated through mutation and sometimes crossover operations to form a new generation. Then parts of the population, usually the least-fit ones, are discarded and replaced by the new individuals. The (1+λ)-EA we utilize is a variant of (1+1)-EA. We generate λ mutants to compete with the parent. The best mutant becomes the parent of the next generation while the current parent is always discarded. In our experiments, we evaluate four different variations of (1+λ)-EA with different mutation rates. Mutation rate refers to the proportion of the genome vector that's randomly changed in one step of mutation. Although designed to target discrete parameters, these EAs can also optimize continuous parameters, albeit not achieving high accuracy on them. A summary of the variants can be found in Table II.

For the continuous parameters, we use Differential Evolution (DE), which also lies under the EA category of algorithms. In its simplest form, a population is first sampled uniformly according to the constraints of the parameters. Then after evaluations, the genome goes through mutation and recombination. In this step, the genomes are moved around in the search space

TABLE II
MUTATION RATES OF DIFFERENT VARIANTS OF (1+λ)-EA USED. [2]

Name	Mutation Rate
Discrete (1+λ)-EA	$1/d$
Lengler [28] (1+λ)-EA	$\texttt{int}(\max(1, d(\frac{a\log(i)}{i})))/d$
Portfolio [29] (1+λ)-EA	$\{\chi\}/d$
FastGA [30] (1+λ)-EA	α/d

by combining positions of existing genomes in the population. Then if a new genome improves in performance compared to the previous population, it is accepted as part of the new population. The algorithm then iterates until a satisfactory solution is found. We use a specific version of DE called TwoPoints DE (two-point crossover) in our experiments. The mutation step performs a crossover to move genomes in the search space. For each individual in the current population, three individuals, one fittest, and two random, are first selected. A donor genome is then created by adding the weighted difference between the two random individuals to the fittest individual. Then two crossover points (indices in the candidate vectors) are specified to split the genome vector equally on both the donor genome and the target genome to specify which portion of the vectors will be exchanged between the two. And whether the vector between the two points or outside the two points are exchanged is decided randomly. Then the new generated genomes are evaluated again, the better performing new genomes will replace the worst performing ones in the previous population. This process is repeated until the desired solution is found.

TABLE III
DETAILED SPECIFICATION OF DESIGN SPACE

Subsystem	Component	Type	Size
Propulsion	Battery	Discrete	34
	Battery Position	Continuous	3×#Battery
	ESC	Discrete	20
	ESC Offset	Continuous	3×#ESCs
	Motor	Discrete	83
	Propeller	Discrete	417
Aero	Left Wing	Discrete	68
	Right Wing	Discrete	68
	Servo	Discrete	27
Structure	Tube Connectors	Discrete	1
	Tube Length	Continuous	# arm
	Flange Connectors	Discrete	2
	Plate Connectors	Discrete	1
	Hub Connectors	Discrete	6
	Connection Angles	Continuous	#Connection
Flight	Autopilot (LQR-Controller)	Continuous	20

[2] d is the dimension of the search space, i is the iteration index, $a = 1.54468$, $\{\chi\}$ is a set of selected numbers between 1 and d, where the specific number is adaptively chosen randomly at every generation, and α is chosen randomly from a power-law distribution. For Lengler, the mutation rate is not fixed, meaning that mutation is randomly decided but on average at the specified rate.

TABLE IV
OVERALL BENCHMARKS FROM DESIGN CHALLENGE

Type	Specification	Score
Rise and Hover	Vertical from 0 to 150m then hover	400
Straight Line	North in a straight line	300
Circle	1km diameter circle clockwise	400
Racing Oval	Racing oval with 2x 750 m straights and 2x 600 m diameter half circles	482
Total	Sum of all Benchmarks	1582

TABLE V
INTERMEDIATE BENCHMARKS FROM TRIM RESPONSES

Type	Specification	Score
Distance	Max. flight distance forward	`target_dist` - *dist*
Time	Max. flight time forward	`target_time` - *time*
Speed	Max. lateral speed	-300 × *max_latvel*
Power	Within max. power limit	*pow* ¡ `pow_lim`