Architecture Generation and Performance Evaluation of Aircraft Thermal Management Systems Through Graph-based Techniques

Daniel R. Herber*, James T. Allison†
University of Illinois at Urbana-Champaign, Urbana, IL 61801

Robert Buettner‡
UES Inc., Dayton, OH 45432

Philip Abolmoali§ and Soumya S. Patnaik¶
Air Force Research Laboratory, Dayton, OH 45433

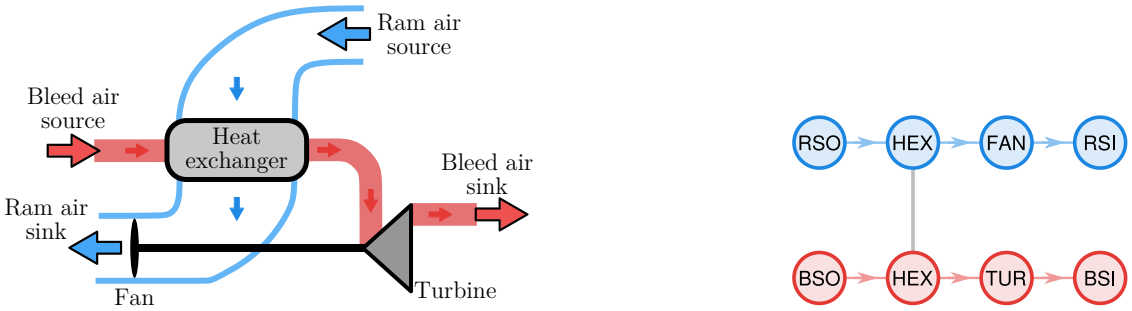
In this article, we are investigating aircraft thermal management system (TMS) architecture design problems. Here we present initial work on a conceptual-level tool that enables the generation of novel system architecture concepts through graph-based methods. Architectures are represented by a particular class of directed graphs which are generated through an enumerative procedure from a component catalog and set of network structure constraints that define the architecture feasibility. While directed labeled graphs can be used to represent the TMS architectures, evaluating their performance requires a nontrivial physics-based model derived from the graph representation. Such software tools are limited, so a custom interface between the chosen graph representation and Modelica was developed to construct the models and perform the required simulations. A case study based on an unmanned aerial vehicle (UAV) equipped with air cycle machine (ACM) based thermal management system (TMS) is shown and demonstrates that both novel and common architectures can be generated, modeled, and evaluated in this framework. The results from this work can be applied across a range of aircraft sub-systems and platforms, including manned and unmanned air vehicles.

I. Introduction

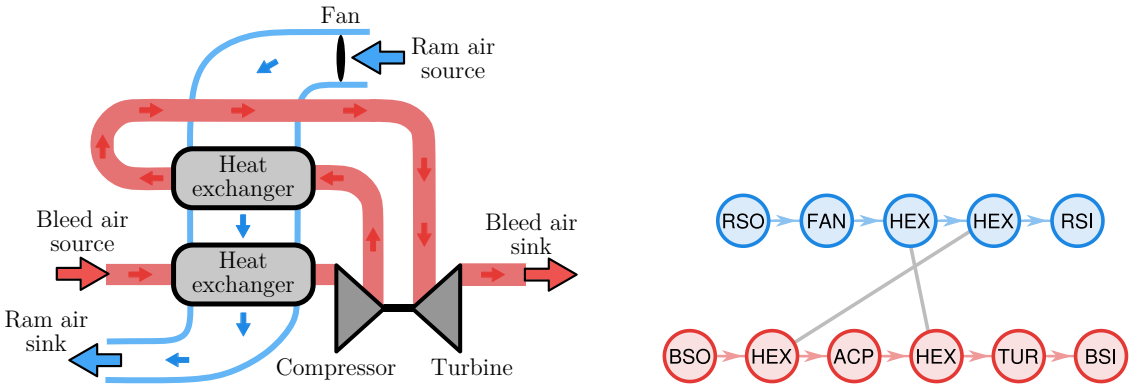
Modern aircrafts have seen increasing internal heat loads due to an increase in electronic components [1–5]. At the same time more efficient engines and use of composite frames have led to a reduction in available heat sinks thus increasing the thermal management challenges. With the rise of the impact of the thermal management system (TMS) performance on the overall performance of the aircraft, there has been an increase in the complexity in its design and many TMS architectures are possible based on the types of components and their relationships which include their relative positions and connectivity. Here we seek to capitalize on recent developments in system architecture design and integrated design optimization methods to tackle pressing challenges facing aircraft TMS design. This work focuses on bringing rigor and flexible exploration to early-stage conceptual design studies by systematically exploring different TMS architecture candidates. Modifications to system architecture (both the included components and their relationships) has the potential to effect significant system performance improvements. Architectural changes, however, lead to fundamentally different designs, which may have fundamentally different design properties compared to existing systems. We have focused our current study on TMS systems with Air Cycle Machines (ACMs) based TMS. Directed labeled graph representation of widely used ACMs such as the simple cycle, two-wheel bootstrap, and three-wheel bootstrap architectures [6–8] are shown in Fig. 1.

Graph-based methods can be used to represent and generate new architecture candidates. Here we use an enumeration-based approach that generates all possible candidate graphs under some general specifications because of

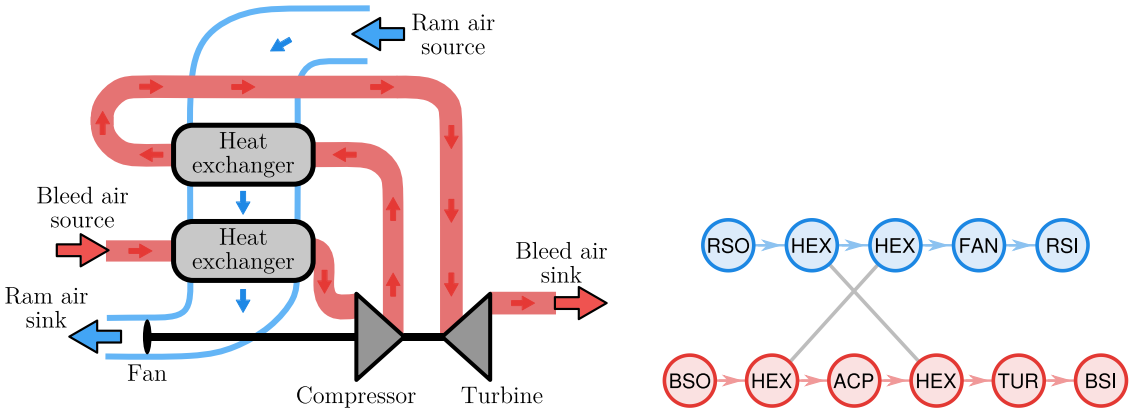
*Postdoctoral Research Associate, Department of Industrial and Enterprise Systems Engineering, 104 S Mathews Ave, AIAA Member.
†Associate Professor, Department of Industrial and Enterprise Systems Engineering, 104 S Mathews Ave, AIAA Senior Member.
‡Research Scientist, Aerospace Power & Propulsion Division, 4401 Dayton-Xenia Rd, AIAA Non-member.
§Aerospace Engineer, Aerospace Systems Directorate, 1950 5th Street, Area B, Wright Patterson AFB, AIAA Non-member.
¶Senior Engineer, Aerospace Systems Directorate, 1950 5th Street, Area B, Wright Patterson AFB, AIAA Non-member.



(a) Simple cycle.



(b) Two-wheel bootstrap.



(c) Three-wheel bootstrap.

Fig. 1 Three common aircraft ACM architectures represented as graphs.

the specific structure of the ACM graph representation and the multiobjective, complex parametric design problem associated with different architectures. The theory and tools developed in Refs. [9, 10] are used to represent and generate the candidate graphs with suitable problem-specific modifications. Alternative graph design methods include evolutionary-based algorithms [11] and graph grammars [12]. These other approaches can have issues with finding low complexity solutions, tuning algorithm hyperparameters, addressing problems with multiple objectives, and requiring significant expert knowledge [13]. The obvious downside of enumeration-based methods is the combinatorial growth of the number of potential solutions that need to be considered so it is only suitable for certain graph design problems. Enumeration-based design has been used successfully in several challenging engineered systems including single-split fluid-based thermal systems [14], vehicle suspensions [15], and synthetic biology [16]. Graph-based design has been used in other domains including electric circuits [11] and hybrid powertrains [17].

Once a candidate graph is available, there are still typically many design decisions that need to be made such as component sizing. This task is sometimes termed the parametric design problem (or inner-loop problem where the outer loop is selecting the optimal architecture). Performance evaluation of the different candidate architectures is essential for making fair comparisons between them, and for understanding the associated performance trade-offs. Recent work in Ref. [18] supports graph-based models of aircraft electrical, thermal, and turbomachinery components, but is limited in both what components can be included and model fidelity. Therefore, a custom model library is developed that supports graph-based construction. It is important to note that the purpose of the early-stage studies proposed in this article is not to supplant the detailed, rigorous previous research, but rather seeks to identify new architectures that could be investigated in the same level of detail that the few canonical architectures have received.

The reminder of this article is organized as follows. In Sec. II, the physics-based component models and automated model creation from a graph representation are described. Then in Sec. III the graph-based approach for representing and generating new candidate TMS graphs is outlined. The main case study is then described in Sec. IV and its corresponding results shown in V. Finally, the conclusions are expressed in Sec. VI.

II. Generating Physics-based Models from TMS Graphs

While directed labeled graphs can be used to represent the TMS architectures as shown in Fig. 1, evaluating their performance requires a nontrivial physics-based model derived from the graph representation. Available software tools and component models are limited, so a custom interface between the chosen graph representation and Modelica was developed to construct the total model and perform the required simulations. Modelica is a multi-domain modeling language for component-oriented modeling of complex systems [19]. Here the vertices in the graph are mapped to a specific physics-based component model and the edges are energy-based connections between the individual component model instances. First, the physics-based component models, written in the Modelica language, are briefly outlined. Then the automated creation of the required Modelica files constituting the total model is described.

A. Component Models

In this work, we consider eight distinct model types in the custom library:

- **Turbine** (*TUR*). The turbine model used in this work is based on the steady-state performance maps of an automotive turbocharger. The maps provide the corrected mass flow rate as a function of the total pressure ratio and the efficiency as a function of the blade speed ratio. While these maps are slightly unconventional in structure, they are suitable for investigations into conceptual-level TMS designs. We note that more conventional turbine maps would give the total pressure ratio and the efficiency as functions of total corrected mass flow rate and corrected rotational speed. Although these maps have been hard coded, resizing of the turbine is possible through several parameters at the best efficiency point (BEP) including the design corrected flow rate and design pressure ratio. Finally, the overall mechanical efficiency and efficiency at the BEP of the turbine can be prescribed. In the graph representation, *TUR* is a two-port component.
- **Compressor** (*ACP*). Similar to the turbine model, the compressor model used in this work is based on the steady-state performance maps of an automotive turbocharger. The maps provide the total pressure ratio and efficiency as functions of corrected rotational speed and corrected mass flow rate. Again these maps are slightly unconventional in structure, but are suitable for investigations into conceptual-level TMS designs. More conventional compressor maps would provide the corrected mass flow rate and efficiency as functions of total pressure ratio and corrected speed. Although these maps have been hard coded, resizing of the compressor is possible through several parameters at the BEP including design corrected speed, design corrected flow rate, and design pressure ratio. Finally, the overall mechanical efficiency and efficiency at the BEP of the compressor can be

prescribed. In the graph representation, *ACP* is a two-port component.

- **Heat Exchanger** (*HEX*). The (inter-subgraph) heat exchangers are modeled as a counterflow offset strip fin heat exchangers that are discretizable in the flow direction. The heat exchanger model uses Colburn (j) and Fanning (f) friction factor correlations developed by Manglik and Bergles to compute the convection heat transfer coefficient [20]:

$$j = 0.6522\,(Re)^{-0.5403}\left(\frac{s}{h'}\right)^{-0.1541}\left(\frac{\delta}{\ell_s}\right)^{0.1499}\left(\frac{\delta}{s}\right)^{-0.0678}\left[1 + 5.269\times10^{-5}\,(Re)^{1.340}\left(\frac{s}{h'}\right)^{0.504}\left(\frac{\delta}{\ell_s}\right)^{0.456}\left(\frac{\delta}{s}\right)^{-1.055}\right]^{0.1} \tag{1a}$$

$$f = 9.6243\,(Re)^{-0.7422}\left(\frac{s}{h'}\right)^{-0.1856}\left(\frac{\delta}{\ell_s}\right)^{0.3053}\left(\frac{\delta}{s}\right)^{-0.2659}\left[1 + 7.669\times10^{-8}\,(Re)^{4.429}\left(\frac{s}{h'}\right)^{0.920}\left(\frac{\delta}{\ell_s}\right)^{3.767}\left(\frac{\delta}{s}\right)^{0.236}\right]^{0.1} \tag{1b}$$

  where Re is the Reynolds number, s/h' is the ratio of the free flow width between fins and the free flow height between plates, δ/ℓ_s is the ratio of fin thickness to the length of a single offset strip in the flow direction, and δ/s is the ratio of fin thickness to the free flow width between fins. This model also assumes incompressible flow, Reynolds numbers between 120–10000, Prandtl numbers between 0.5–15, Aluminum 2024-T6 construction, and identical geometry for both the hot-side and cold-side flow passages. The restrictions are imposed to maintain ±20% accuracy of the heat transfer and friction factor calculations (and the accuracy outside of this range is not well reported). It should also be noted that the thermal mass and heat transfer surface area calculations neglect the terminal plates and the end caps on the flow passages; these are assumed to be perfectly insulated. The geometric parameters include the number of passages per fluid, number of fins along width, number of offset strips per passage, length of passage in flow direction, width of the passage, height between plates, fin thickness, and plate thickness. In the graph representation, a complete heat exchanger is represented by 2 two-port *HEX* components with one in each thermal loop and a connection between the two *HEX* completing a single heat exchanger realization (see Fig. 2a for an example with a single *HEX*).
- **Heat Loads** (*HLx*). The heat load is modeled as an offset strip fin cold plate that is discretizable in the flow direction with an ideal, time-varying thermal power source. Most of the equations and parameters are the same between the heat load models and the *HEX* component type. In the graph representation, *HLx* is a two-port component and the *x* suffix will label distinct heat loads (e.g., *HLR* for a radar-based heat load).
- **Water Separator** (*WSP*). Here we use a simple model of a high pressure water separator based on axial flow cyclone separator technology. The model makes use of the proportionality resistance coefficient [21, 22] (often called the Euler coefficient) to calculate the pressure drop (Δp) across the separator as:

$$\Delta p = (Eu)\frac{\rho v_i^2}{2} \tag{2}$$

  where Eu is the Euler number (typically between 15–30 [23]), ρ is the density of the gas at the inlet, v_i is the velocity at the inlet as calculated by dividing the inlet mass flow rate by the density and effective cross sectional area of the flow inlet. The inlet of the water separator should be kept above freezing at all times and furthermore, should also be kept well below the dew point for maximum effectiveness. However, no attempt has been made to implement actual water separation and removal at this time, therefore, the water separator simply acts as a flow resistance. Additionally, the effective diameter of the flow inlet can be modified. In the graph representation, *WSP* is a two-port component.
- **Pressure Regulating Valve** (*PRV*). The model for the pressure regulating valve follows industrial standards for valve sizing assuming a compressible fluid [24]. The default valve characteristic is of the equal-percentage type. This valve characteristic is commonly used for pressure regulation and also closely approximates the operation of butterfly-type valves which are commonly used in ACMs for their compactness. Both fixed and time-dependent valve opening commands can be provided. Additional tunable parameters include the nominal values for the inlet pressure, pressure drop, mass flow rate, and density, as well as the product of the specific heat ratio factor and pressure drop ratio factor at full-open conditions. In the graph representation, *PRV* is a two-port component.
- **Sinks and Sources** (*BSO*, *BSI*, *RSO*, *RSI*). These components are modeled as stagnant reservoirs with a fluid inlet (for sources *xSO*) or outlet (for sinks *xSI*) and are the interfaces to the external environment. The prefixes *Bxx* and *Rxx* indicate the bleed and ram loops, respectively. Various properties such as temperature, pressure, and

composition can be made either constant or time-dependent. In the graph representation, all sinks and sources are one-port components.

- **Shaft** (*SFT*). A simple rigid shaft model was created for connecting turbomachinery components in the rotational mechanical domain. The equations contained in the shaft model are those for rigid bodies rotating about their center of gravity. In the graph representation, *SFT* is a one-port component.

When defining the composition of the individual component models, it was important to consider the trade-offs in physical accuracy versus computational expense because of the considerable number of simulations are required. These medium-level fidelity models were then defined based on expert intuition and available data, attempting to balance these trade-offs. Other important modeling issues such as media model specification for every component (to specify the fluid properties) and robust state initialization are not discussed in detail but are important for accurate model use. Additional component models are under development including fan and ducts, as well as cooling technologies such as liquid cooling and vapor cycle systems [25].

B. Automatic Creation and Execution of Modelica Files for a TMS Graph

While graphical interfaces exist for constructing these models based on adding and connecting component models, requiring an engineer to manually create each total model can be laborious and restricting, especially when numerous models need to be created and simulated. The key tasks that need to be automated included 1) model file creation, 2) model compilation, 3) parameter updates, 4) simulation, and 5) result extraction. The code for automating these tasks was written in Matlab and * will be used to indicate an arbitrary model name.

To automatically generate the Modelica model file (`*.mo`), three main blocks of code need to be written: 1) initializations (such as fluid properties of the ram and bleed loops); 2) component declarations for every vertex in the graph (e.g., if an graph contains two turbines, then both TUR_1 and TUR_2 need to be declared); and 3) connect equations between different ports of the components (i.e., synonymous to the edges in the graph). Initializations are consistent between each model and may be defined a priori (see lines 2–4 in Fig. 2b). To enumerate the component declarations, first a database is defined linking the graph label to a component model definition in the custom Modelica library. Other information such as port names and parameters are also defined in this database. Then for each label in the graph, the single line component declaration is constructed including the default parameter values and appropriate media model. The media model for each component instance can be automatically determined from the graph representation. An example TMS graph (including the rotational mechanical subgraph with a *SFT* and the other turbomachinery components) is shown in Fig. 2a. The corresponding component declarations are then on lines 6–16 in Fig. 2b. Finally, the connect commands are constructed by looking at each edge in the graph (ignoring the internal *HEX* connections). Edges are defined by an ordered pair of vertices, which correspond to a defined port in each component (so that the correct directionality can be included). An example of these connect equations can be seen on lines 19–29 in Fig. 2b. With the Modelica model file created, it can be compiled for rapid simulations.

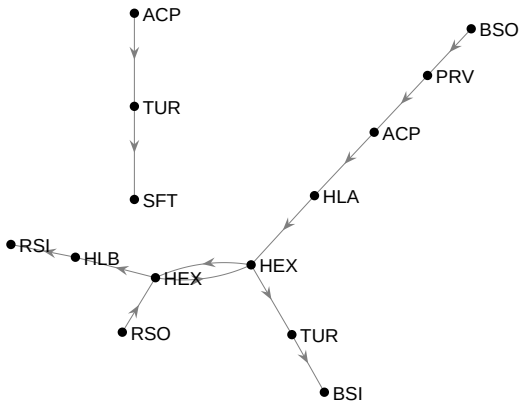
For parameter updates after compilation, the parameter information structures constructed during model file creation are updated with any necessary changes to specific parameter values. Then these updated values are written to the correct location in the appropriate `*_init.xml` file (which is created during model compilation). Now the model can be simulated with the particular set of parameters using a system call to the appropriate `*.exe` file. Results are stored in a `*_res.mat` file. An appropriate result parser is called, outputting a navigable data structure containing the simulation results. Performance measures and constraints can then be evaluated. Result extraction processes need to be written in a way such that arbitrary graphs can be readily handled because different TMS graphs may have different component frequencies and types. For example, if a particular performance measure requires all occurrences of a component type, then the code needs to be able to identify all component types and successfully execute the calculation.

III. Generating Candidate TMS Architectures through Graph-Based Methods

In this section we will describe the methods for generating and analyzing candidate TMS architectures represented by a specific type of graph which we have been referring to as TMS graphs. First, the generic graph concepts are discussed and then the TMS graph specification will be described through the graph concepts.

A. Summary of the Used Graph Concepts

A labeled directed graph G in this work is defined by an adjacency matrix A that defines edges through nonzero row-column entries and a list L that lists all vertex labels. All concepts in this section are presented in a general fashion



(a) Example TMS graph.

```
model ACM_model031
  annotation(experiment(StartTime=0,StopTime=60,Interval=0.1));
  replaceable package RamMedium = Modelica.Media.Air.DryAirNasa;
  replaceable package BleedMedium = Modelica.Media.Air.DryAirNasa;

  APTT.MAGMA.BasicComponents.ACM.BSO bso1(redeclare package Medium = BleedMedium, T = [0,297],...
  APTT.MAGMA.BasicComponents.ACM.PRV prv1(redeclare package Medium = BleedMedium, useThetaInpu...
  APTT.MAGMA.BasicComponents.ACM.BSI bsi1(redeclare package Medium = BleedMedium, T = [0,293],...
  APTT.MAGMA.BasicComponents.ACM.HL hl1(redeclare package Medium = BleedMedium, Nv = 1, nf = 1...
  APTT.MAGMA.BasicComponents.ACM.TUR t1(redeclare package Medium = BleedMedium, DsgnEta = 0.71...
  APTT.MAGMA.BasicComponents.ACM.ACP acp1(redeclare package Medium = BleedMedium, DsgnWc = 0.0...
  APTT.MAGMA.BasicComponents.ACM.HEX hx1(redeclare package Medium1 = BleedMedium, redeclare pa...
  APTT.MAGMA.BasicComponents.ACM.RSO rso1(redeclare package Medium = RamMedium, T = [0,297], p...
  APTT.MAGMA.BasicComponents.ACM.RSI rsi1(redeclare package Medium = RamMedium, T = [0,293], p...
  APTT.MAGMA.BasicComponents.ACM.HL hl2(redeclare package Medium = RamMedium, Nv = 1, nf = 112...
  APTT.MAGMA.BasicComponents.ACM.SFT shaft1(omega_init = 11000);

equation
  connect(bso1.FO1,prv1.FI1);
  connect(t1.FO1,bsi1.FI1);
  connect(acp1.FO1,hl1.FI1);
  connect(hx1.FO1,t1.FI1);
  connect(prv1.FO1,acp1.FI1);
  connect(hl1.FO1,hx1.FI1);
  connect(hl2.FO1,rsi1.FI1);
  connect(hx1.FO2,hl2.FI1);
  connect(rso1.FO1,hx1.FI2);
  connect(acp1.MO,t1.MI);
  connect(t1.MO,shaft1.MI);

end ACM_model031;
```

(b) Automatically-generated Modelica code (long lines truncated).

Fig. 2 Example TMS graph and corresponding Modelica code.

so that they may be readily applied to similar graph problems.

1. Enumerative Methods

Enumerative methods for graphs generate an explicit and exhaustive listing of *all the graphs* under some general specification [9]. This exhaustive listing is also known as a graph structure space [9, 26]. In this work, a few different enumerative methods are used for specific graph enumeration problems and are now described:

(E1) **Path Graph Enumeration.** A path graph has two terminal vertices (vertices with degree one) and all others (if any) have degree two. Given n vertices that must be between the terminal vertices (termed source and sink), then we can naively generate all possible path graphs using basic permutations. Then we would have $n!$ different path graphs to consider. However, the naive approach is lacking for two reasons: 1) typically we do not have n distinct vertices but rather replicates of certain vertex labels; 2) there are typically network structure constraints (NSCs) associated with the path graphs that define if a graph is feasible. The framework and code developed in Refs. [9, 10] provides a good foundation for efficient graph enumeration handling challenges with both NSCs and graph isomorphisms (duplicate graphs). The required information for the enumeration approach in Refs. [9, 10] are the following designer-defined elements:
- $\boldsymbol{L}$ is the label sequence representing distinct component types
- $\boldsymbol{P}$ is a vector indicating the number of ports for each component type
- $\boldsymbol{R} = \begin{bmatrix} \boldsymbol{R}_{\min} \\ \boldsymbol{R}_{\max} \end{bmatrix}$ is a matrix indicating minimum ($\boldsymbol{R}_{\min}$) and maximum ($\boldsymbol{R}_{\max}$) replicates for each component type

The collection $(\boldsymbol{L}, \boldsymbol{P}, \boldsymbol{R})$ is termed the component catalog and implicitly defines a set of graphs. (E1) is illustrated in Fig. 3a.

(E2) **Combining Disjoint Subgraphs.** If we need to combine different subgraphs into a single graph to fully define the desired graph structure space, then we need to consider all possible combinations of subgraphs. Now if there are n_1 unique variations for graph G_1 and n_2 unique variations for graph G_2, then there will be $n_1 n_2$ combinations. We note that if each graph variation is unique, then all of these combinations will result in unique graphs. (E2) is illustrated in Fig. 3b.

(E3) **Inter-Subgraph Connections.** Consider two replicates of a vertex type $\boldsymbol{L}_i$ each in a different subgraph from (E2). Here we define an edge that connects these two replicates an inter-subgraph connection because it connects the two subgraphs. Generating all possible connections between all $\boldsymbol{L}_i$ in all relevant subgraphs is equivalent to generating all perfect matchings [9]. However, we will want to ensure the replicates of $\boldsymbol{L}_i$ in a particular subgraph are not connected, reducing the number of potential connections. If there are only two subgraphs, then this task is equivalent to generating all permutations one of the subgraph's $\boldsymbol{L}_i$. (E3) is illustrated in Fig. 3c.

(E4) **Vertex-Subgraph Distribution.** Suppose we are given a set of distinct vertex types $\boldsymbol{L}_l$ and we can place each one in one of n disjoint subgraphs. Assuming each vertex type can be placed in each subgraph, then there are $n^{|\boldsymbol{L}_l|}$ different arrangements. Each one of these arrangements specifies a different $(\boldsymbol{L}, \boldsymbol{P}, \boldsymbol{R})$ for each subgraph because the label sequence changes. (E4) is illustrated in Fig. 3d.

2. Network Structure Constraints

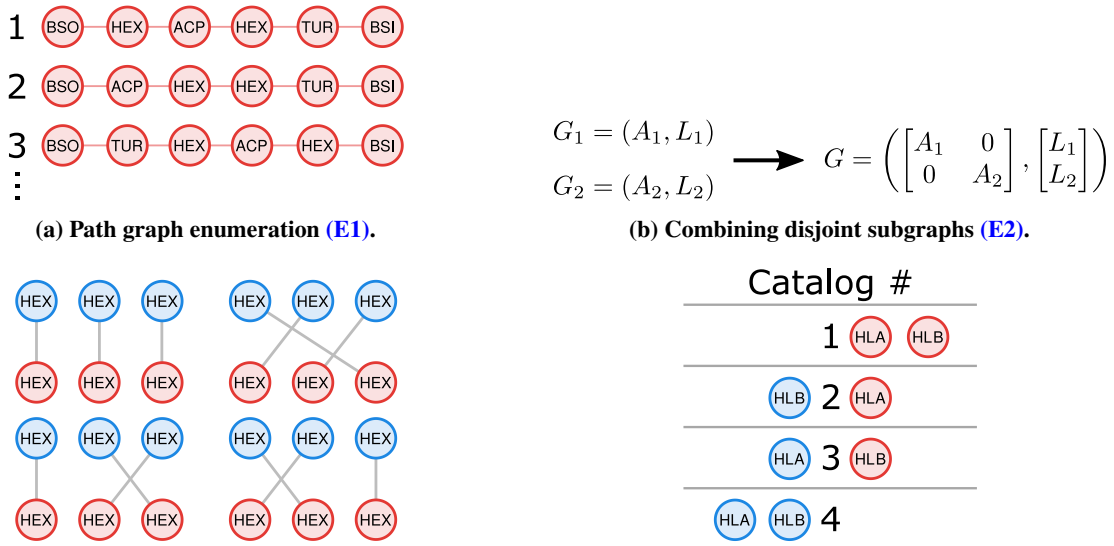
Ensuring that certain NSCs are satisfied help improve the usefulness of the set of generated graphs. We now describe a variety of NSCs that are considered:

(C1) **Direct Connection NSC.** We can directly disallow connections between vertex types L_i and L_j. The collection of these constraints can be combined into a single matrix termed the reduced potential adjacency matrix. Please see Refs. [9, 10] for more details. (C1) is illustrated in Fig. 4a.

(C2) **Line-Connectivity NSC.** If vertex type L_i is connected to vertex type L_j, we can specify if a connection between L_j and L_k is allowed. This is termed a line-connectivity constraint and is described in detail in Ref. [10]. (C2) is illustrated in Fig. 4b.

(C3) **Multi-Edges NSC.** We can require that each edge is unique, i.e., there are no multi-edges in the graph. This constraint is described in detail in Refs. [9, 10]. (C3) is illustrated in Fig. 4c.

(C4) **Intermediate Component NSC.** Given two vertex types labeled L_i and L_j in a path graph, we can determine if there exists a particular vertex type L_k between them. A constraint can be constructed to either declare a particular graph feasible or infeasible if such a condition is satisfied. To implement this constraint, we first determine the location of the two vertex types. Next, we remove all L_k from the graph, i.e., zero all connections to this vertex



(a) Path graph enumeration (E1). **(b) Combining disjoint subgraphs (E2).**

(c) Inter-subgraph connections (E3). **(d) Vertex-subgraph distribution (E4).**

Fig. 3 Illustration of the enumerative methods.

type creating a new graph G_*. Then we compute the connectivity matrix of G_*, denoted W_*. Finally, we can determine if at least one L_k exists between L_i and L_j if there is a connection directly between any L_i and L_j in W_*. (C4) is illustrated in Fig. 4d.

(C5) **Downstream Vertex NSC.** We can determine if a vertex type L_i is downstream from another vertex L_j in a path graph with a specified source vertex. First, we assume that the graph is a directed graph. Then we compute the connectivity matrix for the directed graph, denoted W_*. If there exists a connection at the row index for L_i and column index L_j in W_*, then L_i is downstream from L_j. We can then define conditions to specify if a graph is feasible or infeasible if this situation occurs. (C5) is illustrated in Fig. 4e.

(C6) **Vertex-Subgraph NSC.** We can place constraints on the vertex-subgraph distribution described in (E4). For example, we might want vertex type L_i only in subgraph G_2, but L_j could be in either G_1 or G_2. For each subgraph, we can specify a binary vector with the same number of elements as the set of distinct vertex types $\boldsymbol{L}_l$ where a zero indicates that a particular vertex type cannot be in the subgraph. These vectors are then used to directly enumerate the feasible vertex-subgraph distributions. (C6) is illustrated in Fig. 4f.

3. Graph Modifications

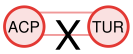
Graph modifications can be used to transform a graph from one form to another. Such modifications are typically used when it is easier to operate on a particular form during enumeration, but another form is desired when generating models or for visualization.

(M1) **Create Directed Graph.** If an undirected graph is a path graph with a specified source vertex, then it can be readily converted from an undirected graph to a directed one where the direction is defined away from the source vertex. Converting an undirected graph into the appropriate directed graph provides additional analysis tools as well as better visualizations. To perform this graph modification, we first initialize an empty adjacency matrix A_d that is the same size as the undirected adjacency matrix A_u. Then we find the location of the source vertex and select it. Then for the selected vertex, we determine the remaining connections (there should only be one in a path graph and this connection defines a downstream vertex). This connection is then added to A_d. Then the downstream vertex is selected and this process is repeated until the selected node no longer has a downstream vertex (i.e., a sink). The labels remain the same.

(M2) **Single Vertex Subgraph Insertion.** For a single selected vertex in a path graph $G = (A, L)$, we can insert a given subgraph $G_* = (A_*, L_*)$ at the selected vertex. We first determine the location of the selected vertex in the list of labels. Then we store the original connections for the selected vertex. Next we insert A_* at the correct location in A. Next we add the original connections since they have been removed. Finally, we insert L_* into the correct location of L. This graph modification can increase the number of total graphs depending on how it is applied.

(a) Direct connection NSC (C1).

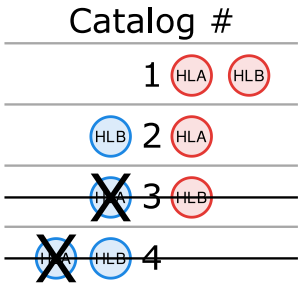
(b) Line-connectivity NSC (C2).



(c) Multi-edges NSC(C3).

(d) Intermediate component NSC (C4).

(e) Downstream vertex NSC (C5).



(f) Vertex-subgraph NSC (C6).

Fig. 4 Illustration of the network structure constraints.

(M3) **Create New Disjoint Subgraph.** Using a given graph $G = (A, L)$, create a new disjoint subgraph based on G. This graph modification then combine the two into a single graph.

B. Aircraft TMS Graph Specification

The aircraft TMS graphs are generated through a number of steps, each enabling the generation and analysis of a particular feature of the architecture. The primary characteristic of the chosen representation is the use of two distinct subgraphs to represent the ram and bleed loops. Each one of the subgraphs can be enumerated individually and then connected through inter-subgraph connections between the *HEX* components. An overview of the steps is shown in Fig. 5. This representation allows us to capture some commonly used TMS architectures shown in Fig. 1.

1. Load-Loop Distributions

Here will we consider some specified list of n unique heat loads. These heat loads can be placed in the different loops enumerated with (E4) and constrained by (C6). For each vertex-subgraph distribution, there will be two lists $\boldsymbol{L}_r$ and $\boldsymbol{L}_b$ which denote the heat loads that are placed in the ram and bleed loops, respectively.

2. Bleed Loop Specification

Using the notation from (E1), the component catalog for the bleed loop is described with:

$$\begin{aligned}
\boldsymbol{L} &= \begin{bmatrix} \mathit{BSO} & \mathit{BSI} & \mathit{TUR} & \mathit{ACP} & \mathit{HEX} & \mathit{WSP} & \boldsymbol{L}_{r,1} & \cdots & \boldsymbol{L}_{r,n} \end{bmatrix} \\
\boldsymbol{P} &= \begin{bmatrix} 1 & 1 & 2 & 2 & 2 & 2 & 2 & \cdots & 2 \end{bmatrix} \\
\boldsymbol{R}_{\min} &= \begin{bmatrix} 1 & 1 & \underline{n}_{\mathit{TUR}} & \underline{n}_{\mathit{ACP}} & 1 & 1 & 1 & \cdots & 1 \end{bmatrix} \\
\boldsymbol{R}_{\max} &= \begin{bmatrix} 1 & 1 & \overline{n}_{\mathit{TUR}} & \overline{n}_{\mathit{ACP}} & \overline{n}_{\mathit{HEX}} & 1 & 1 & \cdots & 1 \end{bmatrix}
\end{aligned} \tag{3}$$

Note that the source, sink, water separator, and loads are required components with unity min and max replicate values. The lower and upper bounds on the number of replicates of the *TUR*, *ACP*, and *HEX* component types are designer-defined parameters.

There are a few NSCs defined to help ensure feasible and useful bleed graphs. First, we want to ensure that all connections are unique, i.e., (C3). Second, we want to ensure that *WSP* is connected directly downstream of a *TUR*. Line-connectivity constraints can ensure that *WSP* is always attached to a *TUR* component. Namely, we disallow allow any combinations of $\square - \mathit{WSP} - \square$ that do not include *TUR*. The downstream vertex NSC (C5) ensures *WSP* is on the correct side of a *TUR*. Next, we want to ensure that there is at least one *HEX* between every (*TUR*, *ACP*) pair. This

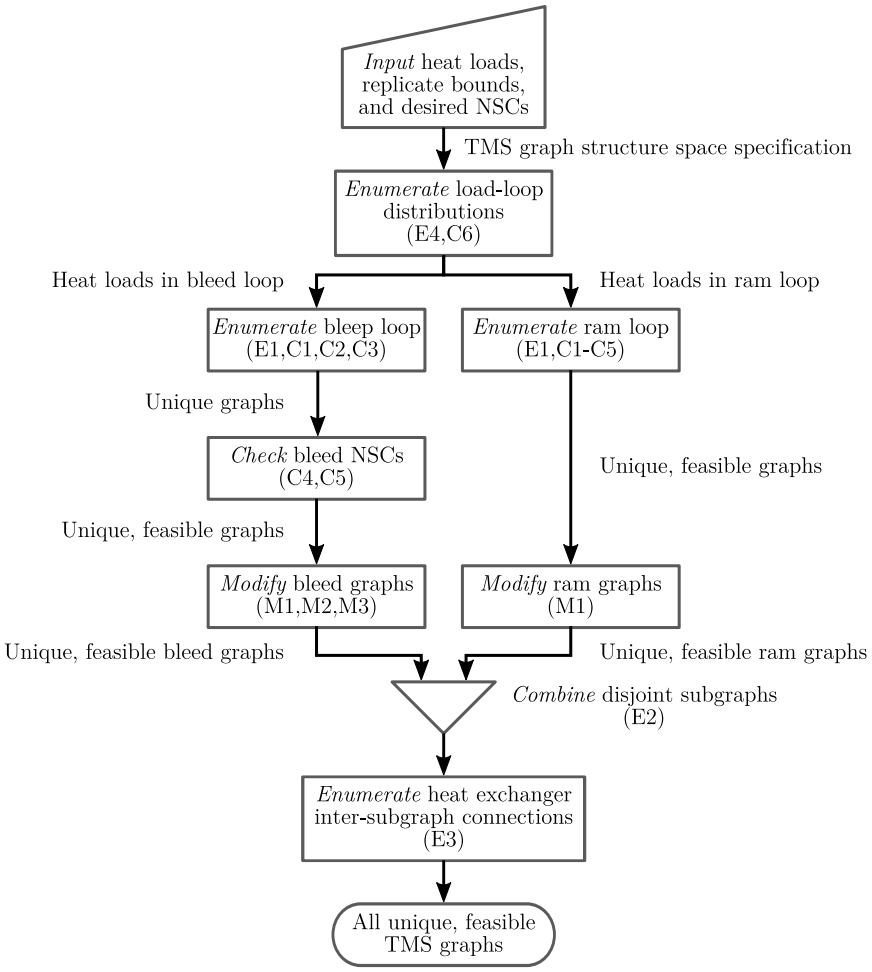


Fig. 5 Overview of the process for generating all aircraft thermal management system graphs.

constraint is an intermediate component constraint described in (C4). To improve the efficiency of the enumeration approach, we can utilize a direction connection constraint described in (C1) and line-connectivity constraints described in (C2). Since there must a *HEX* between *TUR* and *ACP*, we disallow the following connections types: *ACP* – *ACP*, *TUR* – *TUR*, and *TUR* – *ACP*. For the line-connectivity constraints, we disallow all connection types *TUR* – □ – *ACP* and *ACP* – □ – *TUR* except when the middle component is *HEX*. Note that these additional NSCs do not modify the graph structure space but rather improve the runtime of the enumerative methods.

Finally, there are a few graph modifications that are performed on the generated graphs. First, the unidirected graph is converted into a directed one, i.e., (M1), using *BSO* as the source. Next, (M2) is performed expanding *BSO*→ to *BSO*→*PRV*→. Finally, (M3) is applied to create a subgraph representing the connections between the turbomachinery and the *SFT* component. This modification is necessary to model the rotational mechanical dynamics in the ACM. Since all turbomachinery are on the same shaft, all non-(*TUR*, *ACP*) are removed from the bleed graph and then connected in series with a single *SFT* component as a new path graph. For the visualization of the TMS graphs, this subgraph will be omitted. An example of this graph modification was shown in Fig. 2a.

3. Ram Loop Specification

Using the notation from (E1), the component catalog for the ram loop is described with:

$$\begin{aligned} \boldsymbol{L} &= \begin{bmatrix} \mathit{RSO} & \mathit{RSI} & \mathit{HEX} & \boldsymbol{L}_{b,1} & \cdots & \boldsymbol{L}_{b,n} \end{bmatrix} \\ \boldsymbol{P} &= \begin{bmatrix} 1 & 1 & 2 & 2 & \cdots & 2 \end{bmatrix} \\ \boldsymbol{R}_{\min} &= \begin{bmatrix} 1 & 1 & 1 & 1 & \cdots & 1 \end{bmatrix} \\ \boldsymbol{R}_{\max} &= \begin{bmatrix} 1 & 1 & \overline{n}_{\mathit{HEX}} & 1 & \cdots & 1 \end{bmatrix} \end{aligned} \tag{4}$$

where the upper bound on the number of *HEX* replicates $\overline{n}_{\mathit{HEX}}$ should be the same as the bleed loop. The source, sink, and heat loads have similar specifications as the bleed loop. The only NSC used in the ram loop is (C3) to ensure a unique number of edges. The only graph modification performed is (M1) using *RSO* as the source.

4. Heat Exchanger Inter-Subgraph Connections

The primary purpose of the *HEX* component type is to transfer thermal energy between the ram and bleed loops. These connections can be described as the inter-subgraph connections in (E3). For each pair of disjoint candidate bleed and ram subgraphs, all possible matchings between the heat exchangers are considered.

IV. Case Study Description

A. Aircraft and Mission Specification

The generic mission specified here is for an unmanned aerial vehicle (UAV) equipped with ACM-cooled flight control and radar systems. The TMS is tasked with cooling a notational flight control heat load (assumed to be a constant 1,000 W) and a notional radar heat load (assumed a constant 6,500 W). The desired temperatures for these systems are assumed to be 343 K (69.9 °C) and 300 K (26.9 °C), respectively. The aircraft is assumed to be equipped with a turboprop engine making bleed air available. The boundary conditions are assumed to be constant during this generic mission and are shown in Table 1. The mission simulation time is selected to allow the models to reach steady state (allowing the transients induced by problem initialization to diminish).

Table 1 Case study boundary conditions.

Component		Property	Value
Flight Controls Heat Load	*HLF*	Power	1000 W
Radar Heat Load	*HLR*	Power	6500 W
Ram Air Source	*RSO*	Temperature	256.34 K
		Pressure	34561 Pa
Bleed Air Source	*BSO*	Temperature	628.3 K
		Pressure	1002497.7 Pa
Ram Air Sink	*RSI*	Temperature	252.65 K
		Pressure	32853 Pa
Bleed Air Sink	*BSI*	Temperature	252.65 K
		Pressure	32853 Pa

B. Architecture Utility Function

An appropriate utility function is required to rank and identify the most desirable architecture candidates. For an aircraft, the largest concerns for any subsystem are weight, volume, and imposed fuel penalty; the three concerns are related. For example, assuming all other variables are held constant, an increase in the weight of a subsystem requires the aircraft to produce more lift which increases drag, and, subsequently, the thrust required. This, in turn, reduces the maximum range of the aircraft. For an ACM, the primary concern is maintaining the thermal loads within acceptable temperature limits. It is, however, also important to do so while minimizing the impact on other aircraft performance measures such as range and payload. Taking into account these objectives, the following multiobjective utility function

is utilized:

$$\text{minimize:} \quad J = w_1 J_1 + w_2 J_2 + w_3 J_3 + w_4 J_4 + w_5 J_5 \tag{5a}$$

$$= w_1 \left(\overline{T}_{HLF} - 343 \text{ K} \right) + w_2 \left(\overline{T}_{HLR} - 300 \text{ K} \right) + w_3 \left(C_{\text{total}} \right) + w_4 \left(\dot{m}_{BSl} \right) + w_5 \left(\dot{m}_{RSl} \right) \tag{5b}$$

where $\overline{T}_{HLF}$ is the flight control heat load temperature averaged across all discrete volumes, $\overline{T}_{HLR}$ is the radar heat load temperature averaged across all discrete volumes, C_{total} is the total heat capacity of all heat exchangers and heat loads in the system, $\dot{m}_{BSl}$ is the bleed loop mass flow rate, and $\dot{m}_{RSl}$ is the ram loop mass flow rate. The five objectives are discussed in order of perceived importance on overall system utility. J_2 is an indicator for the cooling effectiveness of the TMS for the radar system, while J_1 is related to the flight controls. It is assumed that the radar heat load is more temperature sensitive than the flight controls. Temperature control is the primary purpose of the TMS, so the heat load temperatures are given priority over all other measures. J_3 is an indicator of the overall system mass and the transient response time constant. J_4 is an indicator of fuel computation penalty imposed by the system as well as turbomachinery and ducting size. The bleed air flow rate is considered as being more important than the ram air flow rate because it imposes a direct fuel consumption penalty on the engine. J_5 is an indicator of ram induced drag penalty due to inlet drag as well as ducting size. Any time-dependent properties are evaluated using the values at the end of the simulation as they are the closest to the steady-state values. Nominal weighting factors were determined to provide a single value for ranking architectures, but other values of the weights are also considered:

$$w_1 = 1 \text{ 1/K}, w_2 = 1.1 \text{ 1/K}, w_3 = 10^{-5} \text{ K/J}, w_4 = 1 \text{ s/kg}, w_5 = 0.5 \text{ s/kg} \tag{6}$$

C. System Parameters

As discussed in Sec. I, to fairly compare the different architectures, optimal performance is needed with respect to the many objectives. Trade-offs between the computational expense needed to explore a candidate architecture and its optimal performance needed to be considered. Therefore, only a subset of the available component parameters listed in Sec. II.A were made tunable.

For the turbomachinery (all *ACP* and *TUR*), the design mass flow rate can vary continuously between 0.06–0.1 kg/s. For the pressure regulating valve, the nominal mass flow rate W_{nom} can vary continuously between 0.08–0.2 kg/s. For all heat loads, the geometry of the heat exchanger is selected from one of 7 predefined geometries derived using mass flow rates of $[0.05, 0.12, 0.24, 0.4, 0.6, 1, 1.9]$ kg/s. Similarly, the geometry of all *HEX* is defined by heat exchanger geometries derived using mass flow rates of $[0.05, 0.1, 0.3, 0.5, 1, 2, 3]$ kg/s. Finally, the effective diameter of the *WSP* flow inlet D_{eff} depends on the selected W_{nom} as:

$$D_{\text{eff}} = 0.188562 \sqrt{W_{\text{nom}}} \tag{7}$$

All other parameters were set to a nominal value for all architectures, and modifications to these parameters will be explored in future studies.

V. Results

In this section we present the results for TMS graph generation and the automated model construction and simulation of these TMS graphs to evaluate system performance measures. The computing environment used for all of these studies was a single workstation* where Matlab is the primary coding environment that interacts with OpenModelica when necessary to create, compile, and simulate the Modelica models.

A. Graph Generation

Since the use of an enumerative method suffers from the issue of combinatorial growth, it is important to consider the graph structure space of interest. Using the aircraft TMS graph specification described in Sec. III.B, various replicate counts for the turbomachinery and heat exchangers were considered. The number of unique, feasible graphs for each of these subcatalogs is shown in Table 2. In total, if we want to consider all potential TMS graphs with {0–1 *ACP*, 1–2 *TUR*, 1–4 *HEX*}, we would need to evaluate 503636 architectures. Unless the evaluation of each architecture is quite

*An i7-6800K CPU at 3.8 GHz, 32 GB DDR4 3200 MHz RAM, Matlab 2019a update 6, OpenModelica 1.13.2 64-bit, and Windows 10 build 18362.476.

Table 2 TMS graph generation counts.

Subcatalog	# *ACP*	# *TUR*	# *HEX*	# of Graphs
1	0	1	1	60
2	0	1	2	336
3	0	1	3	2160
4	0	1	4	15840
5	1	1	1	84
6	1	1	2	864
7	1	1	3	7920
8	1	1	4	74880
9	0	2	1	132
10	0	2	2	1056
11	0	2	3	8880
12	0	2	4	80640
13	1	2	1	56
14	1	2	2	1512
15	1	2	3	22176
16	1	2	4	287040
			(total)	503636

inexpensive or sufficient computational resources are available, there are too many candidates. It is important to note that the generation of these half a million unique graphs only required 52 seconds of computation time, which speaks to the power of the developed graph enumeration methods and chosen graph representation. The combinatorial growth can be observed by just considering the increasing number of *HEX*s where approximately a 10× increase in the number of graphs is observed with each additional *HEX* (332→3768→41136→458400). Therefore, it was decided to limit the search to TMS graphs with 1 *ACP*, 1–2 *TUR*, 1–3 *HEX*; there are now *only* 32612 TMS graphs. This new set of graphs was small enough that reasonable exploration of each architecture could be performed while still being able to explore a desirable space of TMS architectures.

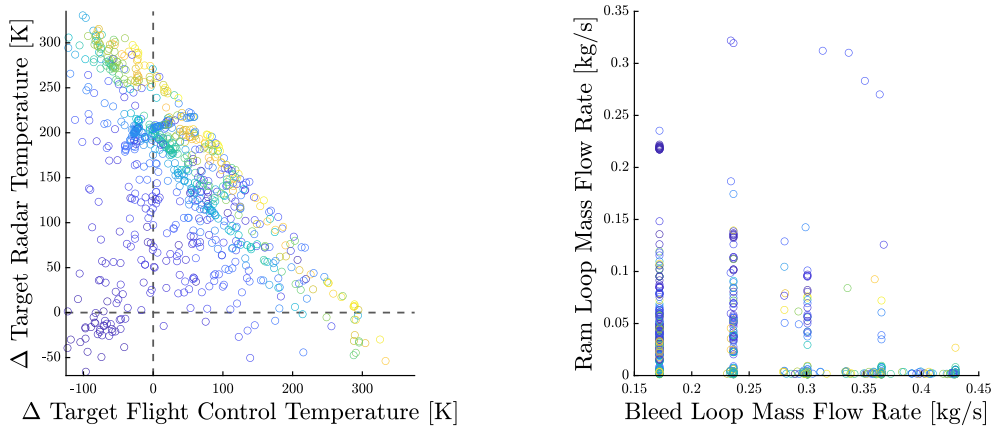
Understanding the component catalog composition and appropriate NSCs is an important task when using the enumerative methods because these decisions can greatly expand or limit the search space. To further illustrate this point, consider (C6) where we can limit what loops the heat loads are placed in. If both heat loads must be in the bleed loop, then there are 200552 graphs (60% reduction). There are 82380 graphs if they are only in the ram graphs (84% reduction). If one heat load must in the bleed loop but the other in either loop, there are 310904 graphs (38% reduction). Understanding these trade-offs is critical to the success of this type of study.

B. Performance

All 32612 TMS graphs in the specified graph structure space have been evaluated. Here evaluation was carried out using 200 randomly generated parameter sets based on the tunable system parameters described in Sec. IV.C. The total computational cost on the single workstation was approximately 9.5 days (noting that these evaluation tasks are embarrassingly parallelizable). Many TMS graphs did not compile (5585/32612 successfully compiled). Of those, only 2097/5585 had at least one successful simulation[†] (simulated the full 300 s). The termination condition could have been one of a number of conditions including the violation of a physical constraint enforced on a particular component/fluid. In total, there were 63437 successful simulations. In Fig. 6, J_2 vs. J_1 and J_5 vs. J_4 are shown for all simulations that resulted in a combined utility value $J < 300$ calculated using the weights in Eq. (6). Only a few of these points are Pareto-optimal, nondominated solutions. These points are shown in Fig. 7, sorted by J. Points in the figure with the same color have the same TMS graph but the parameter sets are different, highlighting the performance variation for a given architecture and the need for comprehensive inner-loop exploration to determine the best performance for a particular architecture.

Figure 8 focuses on the two most important objectives, cooling the heat loads. There are several solutions where

[†]This ratio could be further increased with a more comprehensive exploration of the model parameters but it is also currently affected by a known bug in OpenModelica impacting certain models where simulations cannot be properly initialized.



(a) J_2 **vs.** J_1**.** **(b)** J_5 **vs.** J_4**.**

Fig. 6 Results for all architectures instances with $J < 300$ **(different colors indicate different architectures, 341 in total).**

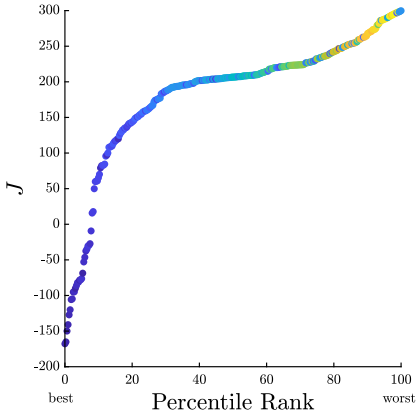


Fig. 7 Sorted results with all 163 Pareto-optimal architectures with $J < 300$ **(different colors indicate different architectures).**

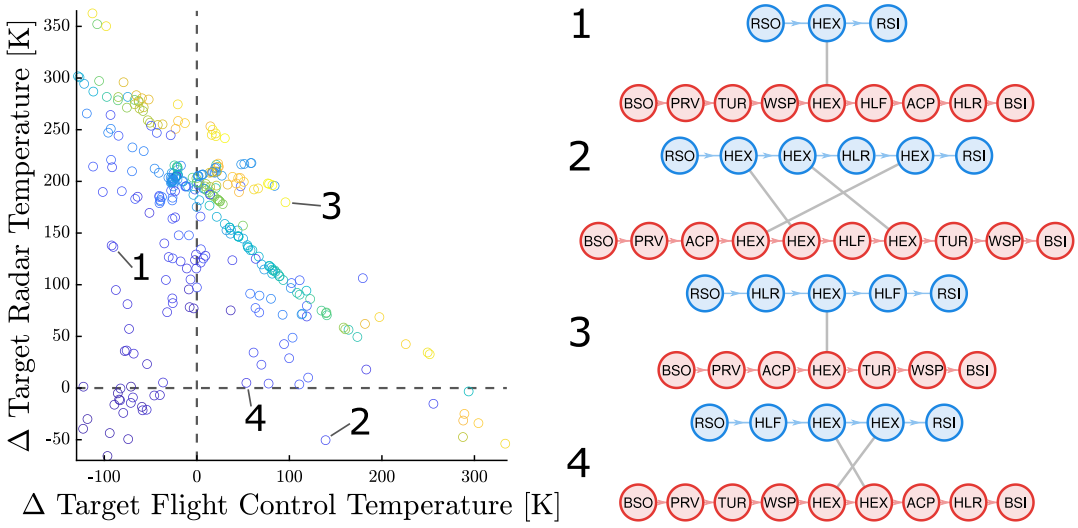


Fig. 8 J_2 **vs.** J_1 **with only the Pareto-optimal architecture instances (different colors indicate different architectures, 163 in total).**

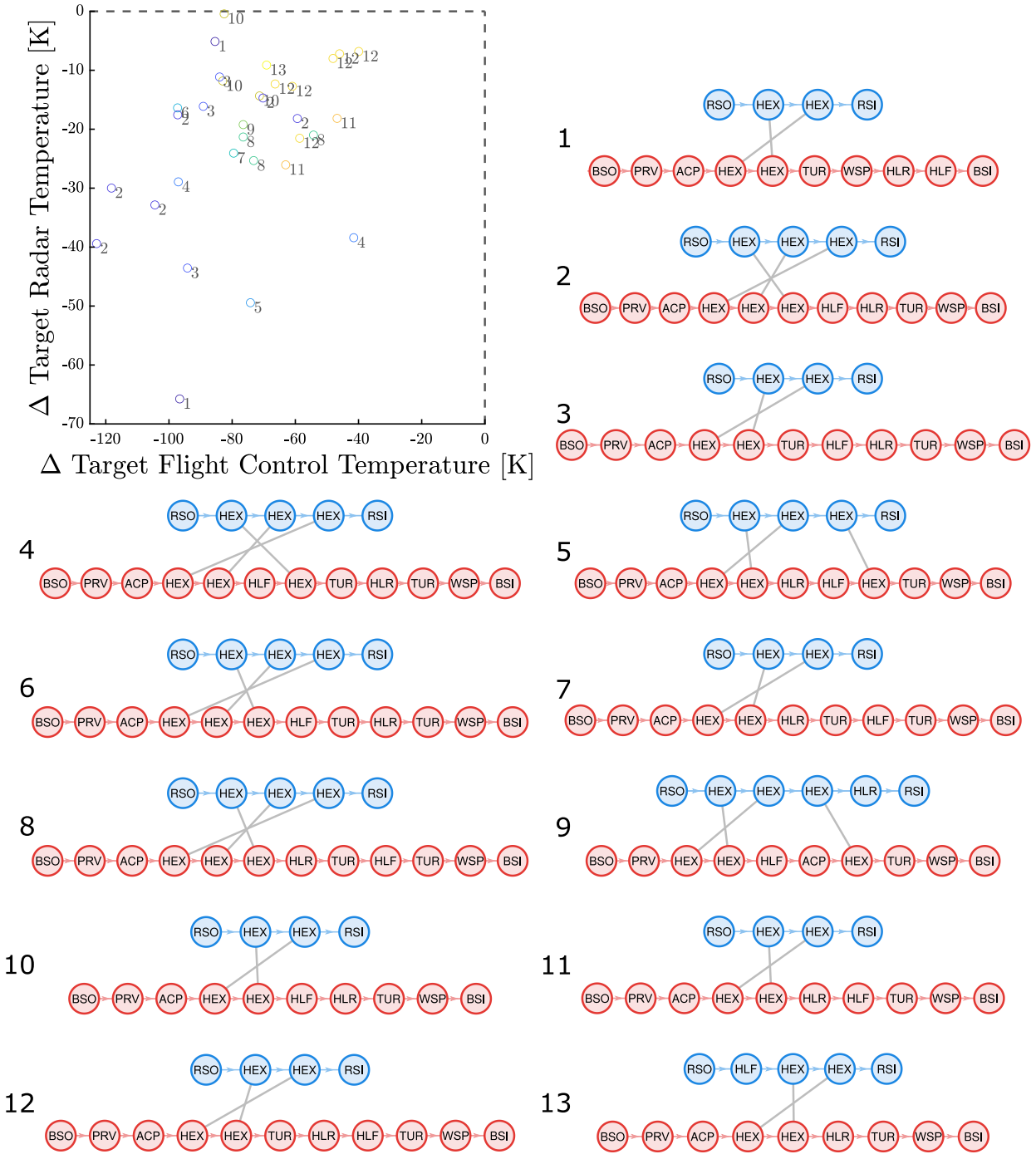


Fig. 9 Pareto-optimal architectures that can sufficiently cool both heat loads (different colors indicate different architectures, 13 in total).

the thermal targets are satisfied. Furthermore, it is observed that there are more results sufficiently cooling the flight control heat load than the radar, which may be expected because the radar has a higher thermal power output. The best weighted solution with one *HEX* is #1 in Fig. 8 which can cool the flight controls but not the more thermal demanding radar. Contrasting with the common ACMs shown in Fig. 1, the single *TUR* is upstream from *ACP*. This collection of TMS graphs highlight the different possible TMS architecture compositions with heat loads distributed between the two loops and differing number of replicates for the component types.

Figure 9 shows all 13 architectures that have both heat loads below their temperature targets. They are numbered with respect to their best weighted objective result with #1 having the lowest value. Many of these architectures share similar features. All of these TMS graphs have the single *ACP* upstream from all *TUR*, similar to the common two-wheel and three-wheel bootstrap architectures. Additionally, all except #1 have *TUR* – *WSP* at the end of the bleed loop. Most have the first *HEX* in the bleed loop connected to the last *HEX* in the ram loop (the exceptions are #5 and #9). The pair #3 and #12 simply have the heat load locations swapped and other pairs are overall quite similar. Further investigation is needed to determine if these architectures are in fact as desirable as these results indicate.

The closest architecture in Fig. 9 to a design in common use is #1. This architecture closely resembles a two-wheel bootstrap system except for the positioning of the first *HEX* in the bleed loop. In the two-wheel bootstrap, this *HEX* would be placed before the *ACP* to pre-cool the bleed air before entering the compressor. Closer examination of architecture #1 would likely show that a pre-cooler is necessary before the *ACP* in order to prevent temperatures from exceeding material limits, but that is beyond the scope of this initial study. In consideration of the temperatures being encountered in the bleed loop, it is likely that the *WSP* would need to come before the TUR instead of the positioning shown here. This may be considered in future studies.

While only a few of the TMS graphs are directly shown, there are 163 Pareto-optimal architectures that can be explored (and 341 in total with $J < 300$). This extremely rich data set and the identification of many promising candidates can be used to explore system architecture trade-offs in a more comprehensive manner.

VI. Conclusion

Here we explored methods for representing, generating, modeling, and evaluating aircraft thermal management system architectures. The conceptual-level framework used graph-based methods for generating all possible candidate TMS graphs under defined graph structure space specifications. It was shown that existing and new TMS graphs could be quickly generated using this approach (503636 unique, feasible graphs in under a minute) but combinatorial issues quickly appear when using enumerative methods. Component models suitable for this study were created in Modelica and an automated procedure was developed to translate the labeled directed graph into an executable Modelica model. Simulations were performed on 32612 TMS graphs for an UAV equipped with ACM-cooled flight control and radar systems. These results indicated that new TMS architectures can be identified. Furthermore, the enumeration-based approach provided many alternative architectures that have diverse trade-offs in the five considered objectives.

There are many potential future work items. This study is considered only an initial filtering process to identify promising TMS architectures that should be investigated further under additional mission profiles and more thorough parametric design optimization. The implementation of a more comprehensive inner-loop optimization approach for each candidate architecture could be included but must be able to handle the multiobjective, mixed discrete/continuous nature of the problem (and an algorithm such as the nondominated sorting genetic algorithm may be appropriate). Additional component models can be included to expand what can be modeled through TMS graphs including the fan and ducts, as well as additional cooling technologies such as liquid cooling and vapor cycle systems.

Acknowledgments

This material is based upon work supported by the National Science Foundation Engineering Research Center (NSF ERC) for Power Optimization of Electro-Thermal Systems (POETS) with cooperative agreement EEC-1449548 and funding support from Aerospace Systems Directorate, AFRL under Contract No FA8650-12-D-2225.

References

[1] Brelje, B. J., and Martins, J. R. R. A., "Electric, Hybrid, and Turboelectric Fixed-wing Aircraft: A Review of Concepts, Models, and Design Approaches," *Prog. Aerosp. Sci.*, Vol. 104, 2019, pp. 1–19. doi: 10.1016/j.paerosci.2018.06.004

[2] de Bock, P., "Thermal Challenges in Today's Commercial and Military Aviation," Online (Accessed on 2019/11/18), Mar. 2011. url: https://www.electronics-cooling.com/2011/03/thermal-challenges-in-todays-commercial-and-military-aviation

[3] Sarlioglu, B., and Morris, C. T., "More Electric Aircraft: Review, Challenges, and Opportunities for Commercial Transport Aircraft," *IEEE Trans. Transport. Electrific.*, Vol. 1, No. 1, 2015, pp. 54–64. doi: 10.1109/TTE.2015.2426499

[4] Wall, T. J., and Meyer, R., "A Survey of Hybrid Electric Propulsion for Aircraft," *AIAA/SAE/ASEE Joint Propulsion Conference*, Atlanta, GA, USA, 2017. doi: 10.2514/6.2017-4700

[5] Gerstler, W. D., and Bunker, R. S., "Aircraft Engine Thermal Management: The Impact of Aviation Electric Power Demands," *Mechanical Engineering-CIME*, Vol. 130, No. 12, 2008.

[6] Kim, J., Kwon, K., Roy, S., Garcia, E., and Mavris, D. N., "Megawatt-class Turboelectric Distributed Propulsion, Power, and Thermal Systems for Aircraft," *AIAA Aerospace Sciences Meeting*, Kissimmee, FL, USA, 2018. doi: 10.2514/6.2018-2024

[7] Santos, A. P. P., Andrade, C. R., and Zaparoli, E. L., "A Thermodynamic Study of Air Cycle Machine for Aeronautical Applications," *Int. J. Thermodyn.*, Vol. 17, No. 3, 2014, pp. 117–125. doi: 10.5541/ijot.538

[8] Conceição, S. T., Zaparoli, E. L., and Turcio, W. H. L., "Thermodynamic Study of Aircraft Air Conditioning Air Cycle Machine: 3-wheel × 4-wheel," *SAE Technical Paper Series*, SAE International, 2007. doi: 10.4271/2007-01-2579

[9] Herber, D. R., Guo, T., and Allison, J. T., "Enumeration of Architectures with Perfect Matchings," *J. Mech. Des.*, Vol. 139, No. 5, 2017, p. 051403. doi: 10.1115/1.4036132

[10] Herber, D. R., and Allison., J. T., "Enhancements to the Perfect Matching-based Tree Algorithm for Generating Architectures," Tech. Rep. UIUC-ESDL-2017-02, Engineering System Design Lab, Urbana, IL, USA, Dec. 2017.

[11] Das, A., and Vemuri, R., "Topology Synthesis of Analog Circuits Based on Adaptively Generated Building Blocks," *ACM/IEEE Design Automation Conference*, 2008, pp. 44–49.

[12] Chakrabarti, A., Shea, K., Stone, R., Cagan, J., Campbell, M., Hernandez, N. V., and Wood, K. L., "Computer-Based Design Synthesis Research: An Overview," *J. Comput. Inf. Sci. Eng.*, Vol. 11, No. 2, 2011. doi: 10.1115/1.3593409

[13] Herber, D. R., "Advances in Combined Architecture, Plant, and Control Design," Ph.D. thesis, University of Illinois at Urbana-Champaign, Urbana, IL, USA, Dec. 2017.

[14] Peddada, S. R. T., Herber, D. R., Pangborn, H. C., Alleyne, A. G., and Allison, J. T., "Optimal Flow Control and Single Split Architecture Exploration for Fluid-Based Thermal Management," *J. Mech. Des.*, Vol. 141, No. 8, 2019, p. 083401. doi: 10.1115/1.4043203

[15] Herber, D. R., and Allison, J. T., "A Problem Class with Combined Architecture, Plant, and Control Design Applied to Vehicle Suspensions," *J. Mech. Des.*, Vol. 141, No. 10, 2019, p. 101401. doi: 10.1115/1.4043312

[16] Ma, W., Trusina, A., El-Samad, H., Lim, W. A., and Tang, C., "Defining Network Topologies that can Achieve Biochemical Adaptation," *Cell*, Vol. 138, No. 4, 2009, pp. 760–773. doi: 10.1016/j.cell.2009.06.013

[17] Silvas, E., Hofman, T., Murgovski, N., Etman, P., and Steinbuch, M., "Review of Optimization Strategies for System-Level Design in Hybrid Electric Vehicles," *IEEE Trans. Veh. Technol.*, Vol. 66, No. 1, 2016. doi: 10.1109/TVT.2016.2547897

[18] Williams, M. A., Koeln, J. P., Pangborn, H. C., and Alleyne, A. G., "Dynamical Graph Models of Aircraft Electrical, Thermal, and Turbomachinery Components," *J. Dyn. Syst. Meas. Contr.*, Vol. 140, No. 4, 2017, p. 041013. doi: 10.1115/1.4038341

[19] Fritzson, P., and Bunus, P., "Modelica - A General Object-oriented Language for Continuous and Discrete-event System Modeling and Simulation," *Annual Simulation Symposium*, San Deigo, CA, USA, 2002. doi: 10.1109/simsym.2002.1000174

[20] Manglik, R. M., and Bergles, A. E., "Heat Transfer and Pressure Drop Correlations for the Rectangular Offset Strip Fin Compact Heat Exchanger," *Exp. Therm Fluid Sci.*, Vol. 10, No. 2, 1995, pp. 171–180. doi: 10.1016/0894-1777(94)00096-q

[21] Dickson, P. J., "Gas/liquid Separation within a Novel Axial Flow Cyclone Separator," Ph.D. thesis, Cranfield University, Cranfield, England, 1998.

[22] Funk, P. A., Elsayed, K., Yeater, K. M., Holt, G. A., and Whitelock, D. P., "Could Cyclone Performance Improve with Reduced Inlet Velocity?" *Powder Technol.*, Vol. 280, 2015, pp. 211–218. doi: 10.1016/j.powtec.2015.04.026

[23] Behrouzi, P., "Performance of Multicell, Axial-entry Cyclones for Industrial Gas Cleaning," Ph.D. thesis, University of London, London, England, Nov. 1988.

[24] *Control Valve Handbook*, Emerson Automation Solutions, 5th ed., 2017.

[25] Doman, D. B., "Fuel Flow Topology and Control for Extending Aircraft Thermal Endurance," *J. Thermophys. Heat Transfer*, Vol. 32, No. 1, 2018, pp. 35–50. doi: 10.2514/1.T5142

[26] Wyatt, D. F., Wynn, D. C., and Clarkson, P. J., "A Scheme for Numerical Representation of Graph Structures in Engineering Design," *J. Mech. Des.*, Vol. 136, No. 1, 2014. doi: 10.1115/1.4025961