

**Towards Concept Generation And Performance-Complexity Tradespace
Exploration of Engineering Systems Using Convex Hulls**

by

Narek Rouben Shougarian

MEng Aeronautical Engineering, Imperial College London 2011

SUBMITTED TO THE DEPARTMENT OF AERONAUTICS AND ASTRONAUTICS IN
PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY IN AERONAUTICS AND ASTRONAUTICS
AT THE

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

January 2017

[February 2017]

© 2017 Massachusetts Institute of Technology. All Rights Reserved.

Signature of Author: Signature redacted

Department of Aeronautics and Astronautics
January, 2017

Certified by: Signature redacted

Professor Olivier de Weck
Professor of Aeronautics and Astronautics
and Engineering Systems, Thesis Advisor

Certified by: Signature redacted

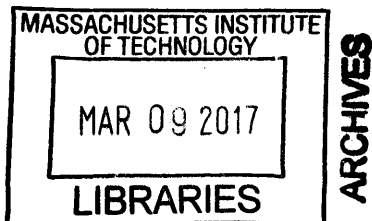
Professor Edward M. Greitzer
Professor of Aeronautics and Astronautics
Thesis Committee Member

Certified by: Signature redacted

Dr. Lawrence Zeidner
United Technologies Research Center
Thesis Committee Member

Accepted by: Signature redacted

Professor Youssef M. Marzouk
Associate Professor of Aeronautics and Astronautics
Chair, Graduate Program Committee



Towards Concept Generation And Performance-Complexity Tradespace Exploration of Engineering Systems Using Convex Hulls

by

Narek Rouben Shougarian

Submitted to the Department of Aeronautics and Astronautics
on February 1, 2017, in partial fulfillment of the
requirements for the degree of
DOCTOR OF PHILOSOPHY

Abstract

This thesis examines tradeoffs between performance and complexity in the context of future aircraft engine architectures and a reconfigurable mobile device. Traditionally aircraft/engine system architecture selection has been conducted via expert suggestion, evaluation and down-selection of alternatives. This work proposes a convex hull-based computational approach to generating architectures that uses flows at subsystem interfaces to enforce feasibility of connections between components. Convex hulls at component interfaces are used to represent the entire space of possible flows that they can exhibit. This is combined with automatic nonlinear model synthesis, optimization and complexity quantification to create an approach that allows the designer to trade the number of false negatives and the size of the design space. A false negative is an architecture that is not generated or once generated is discarded from consideration erroneously as an infeasible one, even though once implemented it would perform the intended function at the intended level of performance adequately.

71 aircraft engine architectures were generated and optimized for minimum uninstalled thrust specific fuel consumption, including families of distributed/non-distributed turboelectric architectures. Families of distributed turboelectric propulsion systems in which some fans are driven electrically and some are driven mechanically were found to exhibit lower uninstalled fuel consumption than current designs due to high efficiency & power to weight ratio of mechanical power transfer and the propulsive efficiency improvements despite losses in electrical power transfer. A simplified correction for the weight penalty associated with these architectures however shows that to gain overall benefit in fuel consumption, boundary layer ingestion will be required. While the literature typically examines individual concepts, in this case a portfolio of concepts was generated and can be leveraged in new aircraft configuration design studies in the future. The second major finding of this work is that the set of generated distributed turboelectric architectures are approximately 1.2 to 2 times as complex as current engine architectures using a complexity metric from the literature that takes into account complexity of components, interfaces and the network of connections between them.

21,168 reconfigurable mobile device architectures were generated and simulated. This number is larger than that for engines due to the presence of more optional components for the

reconfigurable mobile device and the fact that this analysis was carried out at only one level abstraction, whereas two were used for the engine. The first finding was that only approximately 2% of possible architectures were feasible. The second finding was that, while no individual architecture was exceedingly complex compared to current devices, the aggregate complexity of the full set of 21,168 architectures was very high. Moreover, the sparse space of feasible architectures meant that in addition to conducting virtual/physical verification and validation of predicted feasible device configurations, a platform owner would have to guide users through a vast architectural space to their desired configuration.

Thesis Supervisor: Olivier L. de Weck
Professor of Aeronautics and Astronautics and Engineering Systems

Acknowledgments

I would first of all like to express my gratitude to my parents, Rouben Shougarian and Lilit Karapetyan and siblings, Tigran Shougarian and Haik Shougarian, for their unwavering support through this uncertain and challenging endeavor, for being accommodating of my occasional (sometimes chronic) absentmindedness and for the many delicious late night dinners that I have enjoyed over the last 5 years.

I am profoundly thankful for the support and guidance of my advisor Professor Olivier de Weck. His advice and criticism combined with the creative freedom he provided enabled me to take risks, grow as a researcher and ultimately make a contribution to my field.

Thank you also to my committee members and readers: Dr. Lawrence Zeidner who to a large extent inspired this work and guided me throughout, Prof. Edward Greitzer for his encouragement and criticism, Dr. Jayant Sabnis for his expertise and advice and Prof. Daniel Selva for his support and input from the very beginning.

I would also like to acknowledge the wonderful support of my fellow researchers and friends in the Strategic Engineering Research Group and the wider MIT community. In particular, Dr. Kaushik Sinha, who mentored and supported me from my first day as an undergraduate researcher at MIT to my last as a doctoral candidate, Dr. Sydney Do for his expert advice, friendship and support at every turn, Marc Sanchez for his invaluable advice that always had an uncanny way of making things better, Andrew Owens, for technical as well as non-technical discussions and a willingness to help regardless of inconvenient timing, Veronica Foreman, for helping me see my research from a different perspective and making the lab a happier place for everyone, Sam Wald, for his creative advice and help, Olivier Cornes for his support in the lab and in the Charles, Bastien Gorret for his input and excellent fondue,

Dr. Edoardo Colombo for his expert advice, insights and cooking, Demetrios Kellari, for being an exceptional colleague and friend whom I will never be able to thank enough along with Vivian Diep and Bianca Datta for their friendship and support throughout. I am very fortunate to have had the honor to know you. It is my hope that our paths will continue to cross in the future.

I would also like to thank Nouné Grantovna, who first got me interested in math and physics. Many thanks to Randy Skelding for his energetic support and advice leading up to and during my time at MIT and Dr. Robert Mears for his encouragement throughout. Many thanks also to all the teachers at the Aregnazan Educational Complex-Waldorf School of Yerevan, where I built and tested my first rockets, not all of which were successful, leading me to embark on a path of “doing the math” first to avoid injury. In particular I extend my gratitude to Mr. Ara Atayan and Mr. Rafael who both played major role in my development as a young student.

Finally I would like to express my gratitude to Carolyn Mugar, without whom my undergraduate education at Imperial College London would have been impossible, and Andrei Shugarov, Ruzan Mesropyan, Vahram Atayan and Sophie Wollstein for their ongoing support without which I would have been able to study abroad.

Funding sources that enabled this work to be completed include Vanderbilt University/DARPA (contract VU-DSR #21806-S7 and VU-DSR #22666-S3), Google (contract PILOT MULTI-UNIVERSITY SRA EFF 5/31/2013, SOW #3 and MUSRA dated 5/31/13 SOW #4 PO288160), the United States Navy (contract Base TO Agreement 2015-461), Pratt & Whitney (contract 2010101) and NASA (contract MIT-CJ60D)

For my grandparents

Contents

1	Introduction	35
1.1	Motivation	35
1.1.1	Air-Breathing Propulsion System Architecture	39
1.2	Outline of Thesis	49
2	Literature Review	53
2.1	Importance of Considering Broad Sets of Architectures	55
2.2	Different Approaches to Architecture Generation	56
3	Thesis Statement	69
3.1	Research Questions	70
3.2	Approach	70
4	Approach	71
4.1	Magellan Architecture Explorer	71
4.2	Representation and Visualization	74
4.3	Structural Reasoning	78
4.3.1	Pairwise Feasibility (Convex Hulls)	79
4.3.2	Network Feasibility Using Resource Envelopes (Convex Hulls)	83
4.3.3	Satisfying Requirements	89

4.4	Multiple Layers of Abstraction	90
4.4.1	Architecture Evolution	93
4.4.2	New Components: Application to Technology Roadmapping	94
4.5	Search Algorithm	95
4.6	Performance Evaluation	96
4.6.1	Simulation	96
4.6.2	Multidisciplinary Optimization	99
4.7	Complexity Evaluation	100
4.8	End To End Architecture Generation Example: Mass Spring Damper	102
4.8.1	Phase One DS2M Generation	102
4.8.2	Phase Two DS2M Generation	102
4.9	Architecture Generation Results	103
4.10	High Level Summary	105
4.11	Limitations of Approach	107
4.11.1	1D Network Flow Representation	107
4.11.2	State Dependency during Architecture Generation	108
4.11.3	Abstraction	108
4.11.4	Decoupling of Feasible Architecture Generation and Evaluation and Optimization of those Architectures	110
4.11.5	Robustness of Models and Optimization	110
5	Air Breathing Propulsion Case Study	113
5.1	Historical Data	114
5.2	Engine Reconfigurable Model	119
5.2.1	Abstraction Level Three: Reconfigurable Engine Model	120
5.2.2	Abstraction Level Two: Reconfigurable Engine Model Modelica	155

<i>CONTENTS</i>	9
5.2.3 Abstraction Level One: Reconfigurable Engine Model	166
5.2.4 Model Validation	172
5.3 Architecture Generation	175
6 Reconfigurable Mobile Device Case Study	205
6.1 Modeling	209
6.1.1 Battery Life Model	209
6.1.2 Benefit and Price Utility Model	211
6.2 Configuration Generation	213
7 Summary, Conclusions and Areas of Future Work	225
7.1 Methodological Contributions	226
7.2 Domain Specific Contributions	227
7.2.1 Air Breathing Hybrid Electric Propulsion	227
7.2.2 Reconfigurable Mobile Device	231
7.3 Future Work	232
7.3.1 Methodological Future Work	233
7.3.2 Engine Case Study Future Work	237
7.3.3 Reconfigurable Mobile Device Future Work	238

Glossary

ATAG: Air Transport Action Group

ICAO: International Civil Aviation Organization

Concept: A concept is a product or system vision, idea, notion, or mental image that maps function to form. It is a scheme for the system and how it works. It embodies a sense of how the system will function and an abstraction of the system form. It is a simplification of the system architecture that allows for high-level reasoning.

Architecture: The embodiment of concept, and the allocation of physical/informational function to elements of form, and definition of interfaces among the elements and with the surrounding context.

Configuration: An arrangement of existing elements of form [1].

Feasible Concept, Architecture or Configuration: A concept, architecture or configuration that is able to meet requirements.

False Negative: Feasible concept, architecture or configuration that is removed from consideration.

False Positive: Infeasible concept, architecture or configuration that is kept in consideration.

Nomenclature

Air Breathing Propulsion Case Study

c_{fuel}	Specific Energy of Fuel ($42MJkg^{-1}$)
c_p	Specific Heat At Constant Pressure of Air ($1005Jkg^{-1}Kelvin^{-1}$)
γ_{cold}	Ratio of Specific Heats of Air At Room Temperature (1.4)
γ_{hot}	Ratio of Specific Heats of Air At High Temperature (1.3)
η_s	Isentropic Efficiency
η_p	Polytropic Efficiency
T_{0in}	Stagnation Temperature In [<i>Kelvin</i>]
T_{0out}	Stagnation Temperature Out [<i>Kelvin</i>]
T_{in}	Stagnation Temperature In [<i>Kelvin</i>]
T_{out}	Stagnation Temperature Out [<i>Kelvin</i>]
p_{0in}	Stagnation Pressure In [<i>Pa</i>]
p_{0out}	Stagnation Pressure Out [<i>Pa</i>]
p_{in}	Static Pressure In [<i>Pa</i>]
p_{out}	Static Pressure Out [<i>Pa</i>]
T_{ref}	Reference Temperature Sea Level Standard Day Conditions [<i>Kelvin</i>]
p_{ref}	Reference Pressure Sea Level Standard Day Conditions [<i>Pa</i>]
ω	Angular Velocity [<i>rads⁻¹</i>]
<i>torque</i>	Torque [<i>Nm</i>]
\dot{m}	Mass Flow [<i>kgs⁻¹</i>]
<i>TSFC</i>	Thrust Specific Fuel Consumption [<i>kgN⁻¹s⁻¹</i>]
<i>Thrust</i>	Forward Force Provided By Engine [<i>N</i>]
<i>BPR</i>	Ratio of Bypass Flow to Core Flow
<i>OPR</i>	Overall Pressure Ratio of Engine

\dot{Q}	Heat Transferred Per Unit Time [<i>W</i>]
U	Overall Heat Transfer Coefficient [$Wm^{-2}Kelvin^{-1}$]
A	Overall Heat Transfer Area [m^2]
ΔT_m	Mean Temperature Difference [<i>Kelvin</i>]
ΔT_{lm}	Log Mean Temperature Difference [<i>Kelvin</i>]
T_1	Hot Fluid Temperature Inlet [<i>Kelvin</i>]
T_2	Hot Fluid Temperature Outlet [<i>Kelvin</i>]
t_1	Cold Fluid Temperature Inlet [<i>Kelvin</i>]
t_2	Cold Fluid Temperature Outlet [<i>Kelvin</i>]

Reconfigurable Mobile Device Case Study

Benefits Utility ^{<i>h</i>} _{<i>j</i>}	Benefits Utility of Configuration <i>j</i> to User <i>h</i>
Price Utility ^{<i>h</i>} _{<i>j</i>}	Price Utility of Configuration <i>j</i> to User <i>h</i>
<i>b</i> _{<i>i</i>} ^{<i>h</i>}	Part-Worth Utility of Module <i>i</i> to User <i>h</i>
<i>b</i> ^{<i>h</i>}	Part-Worth Utility of Price of User <i>h</i>

List of Figures

1.1	Increasing cost of aerospace and defense systems taken taken from keynote address at the 2013 Conference for Systems Engineering Research given by Paul Eremenko, the former deputy director of the DARPA Tactical Technology Office [2]. Original source: former CEO of Lockheed Martin N. Augustine [3].	36
1.2	Correlation of a measure of structural and software complexity with aerospace development time and cost. Taken from keynote address at the 2013 Conference for Systems Engineering Research, the former deputy director of the DARPA Tactical Technology Office [2].	37
1.3	Contributors to Price Escalation from the F-15A (1975) to the F-22A (2005). Taken from keynote address at the 2013 Conference for Systems Engineering Research given by Paul Eremenko, the former deputy director of the DARPA Tactical Technology Office [2]. Original source: RAND corporation [4]. . . .	38
1.4	Performance S-curves at the system and subsystem levels. Taken from Dr. Carlos Gorbea’s PhD thesis [5]. Original Source: [6].	39
1.5	Lycoming XR-7755-3, Radial 36 Engine Displayed at Smithsonian Institution, the most powerful radial piston engine ever built (1946) [7]	41
1.6	JT3 Turbojet Engine [8].	43

1.7	JT9D [8]. Two Shaft Turbofan. Fan pressure ratio (FPR) 1.6, Bypass ratio (BPR) 5.15, Overall Pressure Ratio (OPR) 33. See nomenclature section for definitions.	44
1.8	Rolls-Royce RB211 524G [8]. Three Shaft Turbofan. Fan pressure ratio (FPR) Not Available, Bypass ratio (BPR) 4.3, Overall Pressure Ratio (OPR) 22.2. See nomenclature section for definitions.	45
1.9	Pratt & Whitney Geared Turbofan Architecture [9]. Fan pressure ratio (FPR) unavailable, Bypass ratio (BPR) 12, Overall Pressure Ratio (OPR) unavailable. See nomenclature section for definitions.	46
1.10	Kuznetsov NK93/94. Diagram from Jane's Aero-Engines [8]. Fan pressure ratio (FPR) unavailable, Bypass ratio (BPR) 16.6, Overall Pressure Ratio (OPR) 37. See nomenclature section for definitions.	47
1.11	Takeoff Thrust Specific Fuel Consumption vs. Year of First Flight. Note that the Pratt & Whitney Geared Turbofan is not listed here since there is no publicly available data regarding its TSFC at takeoff [8].	48
1.12	NASA N+3: Breakdown of the relative benefits to fuel consumption, noise and LTO NOx emissions from different sources of design changes. The D8 (double bubble) configuration provides the greatest improvement [10].	49
2.1	Concept generation techniques (Ideation Techniques). Taken from [11]	59
4.1	Convention for flow type and interface flow direction. Four primary flow types are used [12]. Each type has associated with it a color, number and location within a cell in the DS2M. Three different interface flow directions are represented. These include inputs, outputs and input/output interfaces. As the name suggests input/output interfaces have no inherent direction associated with them.	76

<i>LIST OF FIGURES</i>	17
4.2 Automatically Generated Design Space Structure Matrix (a) Block Diagram Representation (b) Matrix Representation	78
4.3 Resource Constrained Design Space Structure Matrix (a) Block Diagram Representation (b) Matrix Representation	78
4.4 Performance envelopes of KC10 and F18 must intersect for them to be able to exchange fuel in flight. Images from left to right taken from [13], [14] and [15] respectively.	79
4.5 Resource envelope intersection constraint. To reduce the number of constraints we approximate the true intersection of the convex hulls representing shaft power with one that contains fewer linear constraints. For this reason the depicted intersection (highlighted in red) does not exactly match the true intersection of the convex hulls.	80
4.6 Intersection of true envelopes based on data from [16].	81
4.7 Function and form view of multiple low power turbines driving a single compressor.	84
4.8 Notional architecture.	85
4.9 Notional architecture.	86
4.10 (a) Architecture example 1 (b) Architecture example 2	89
4.11 Imaginary Component Requirements Satisfaction	90
4.12 Abstraction from level N+1 to level N	91
4.13 Architecture evolution using resource envelopes	93
4.14 New components: First resource flow envelopes at abstraction layer N are defined. This is followed by an attempt to generate these flows with architecture generation at a deeper level of abstraction.	95
4.15 Modelica model of simplified turbojet. Run time approximately 2 seconds.	97
4.16 Modelica model of simplified turbofan. Run time approximately 2 seconds.	97

4.17 Modelica model of simplified geared turbofan. Air-oil (AOHEX) and fuel-oil (FOHEX) heat exchangers are used to cool oil circulating through and lubricating gearbox. Run time approximately 2 seconds.	98
4.18 Modelica model of unconventional architecture consisting of multiple fans powered by a single turbine. One of the fans is geared. Air-oil (AOHEX) and fuel-oil (FOHEX) heat exchangers are used to cool oil circulating through and lubricating gearbox. Run time: approximately 2-3 seconds.	99
4.19 Automatically Generated DS2M Example (Mass Spring Damper)(a) Matrix Representation (b) Block Diagram Representation	103
4.20 Interface Resource Constraints	104
4.21 Convex Hull Filtered DS2M Example. Grey entries indicate partially overlapping convex hulls. (Mass Spring Damper)	104
4.22 Architecture 1	105
4.23 Architecture 2	105
4.24 Magellan flow chart	106
4.25 Abstraction from level N+1 to level N. Module boundary crossing connections limited by description of subsystem at Level N. Difference between union and intersection generates an optimistic component at Level N whose behavior is the convex hull of the union of convex hulls at level N+1	109
5.1 Generic gas turbine engine diagram adapted from [17]	115
5.2 Graphical depiction of data missing from [8]. Each row represents data available for an engine. Missing data is highlighted in red.	117
5.3 Correlation between dependent variables (19 engines with both cruise and takeoff thrust specific fuel consumption)	118

5.4	Correlation between design variables and dependent variables. 9 engines out of the 62 in the dataset have all of these present.	119
5.5	Control Volume Representation of Flow in Intake	122
5.6	Simplified Intake Side View	123
5.7	Fan performance map from [16]	125
5.8	Fan Simplified Side View	128
5.9	Low (a), Intermediate (b), High (c) Pressure Compressor Maps [16]	133
5.10	Compressor Simplified Side View	134
5.11	Burner Simplified Side View	136
5.12	Low (a), Intermediate (b), High (c) Pressure Turbine Maps [16]	138
5.13	Turbine Simplified Side View	139
5.14	Simplified Bypass Nozzle Side View	141
5.15	Heat transfer between two fluids in concentric pipes. Adapted from [18]	142
5.16	Counter flow heat exchanger. Adapted from [18]	143
5.17	Electric Motor/Generator with Cooling Fins in Bypass Nozzle	150
5.18	Cooling Fin	151
5.19	Control Volume for In	151
5.20	Fuel, thrust, gas, oil, shaft power and electrical power interface Modelica models.	156
5.21	Level 2 Atmospheric Source Requirement	157
5.22	Level 2 Atmospheric Sink Requirement	158
5.23	Level 2 Fuel Requirement	158
5.24	Level 2 Thrust Requirement	159
5.25	Level 2 Fan	160
5.26	Level 2 Compressor	160
5.27	Level 2 Burner	161
5.28	Level 2 Turbine	162

5.29	Level 2 Core Nozzle	162
5.30	Level 2 Bypass Nozzle	163
5.31	Level 2 Oil Pump	163
5.32	Level 2 Air-Oil Heat Exchanger Internal Block Diagram	164
5.33	Level 2 Air-Oil Heat Exchanger	164
5.34	Level 2 Gearbox	165
5.35	Level 2 Electric Generator	165
5.36	Level 2 Electric Generator	166
5.37	Level 1 Shaft Power Stagnation Pressure Increasing Internal Block Diagram .	167
5.38	Level 1 Shaft Power Stagnation Pressure Increasing	167
5.39	Level 1 Brayton Shaft Power Providing Increasing Internal Block Diagram .	168
5.40	Level 1 Brayton Shaft Power Providing Increasing	168
5.41	Level 1 Cold Flow Expanding	169
5.42	Level 1 Cold Flow Expanding Internal Block Diagram	169
5.43	Level 1 Hot Flow Expanding	170
5.44	Level 1 Hot Flow Expanding Internal Block Diagram	170
5.45	Level 1 Mechanical Shaft Power Transferring with Air Cooling	171
5.46	Level 1 Mechanical Shaft Power Transferring with Air/Oil Cooling Internal Block Diagram	171
5.47	Level 1 Electrical Shaft Power Transferring	172
5.48	Level 1 Electrical Shaft Power Transferring with Air/Oil Cooling Internal Block Diagram	172
5.49	Validation of level 2 model against GE90-85B. Agreement in fuel consumption was approximately 10%.	173
5.50	Normalized Thrust Specific Fuel Consumption vs. Normalized Complexity .	174

5.51 Abstraction Layer 1 Engine DS2M. Gas connections: **red**, shaft power connections: **blue**, mechanical connections: **black**, fuel connections: **green**, oil connections: **cyan**, electrical connections: **magenta**. 178

5.52 Abstraction Layer 1 Engine DS2M Filtered With Convex Hulls (87% reduction). Gas connections: **red**, shaft power connections: **blue**, mechanical connections: **black**, fuel connections: **green**, oil connections: **cyan**, electrical connections: **magenta**. 179

5.53 Normalized Thrust Specific Fuel Consumption vs. Normalized Complexity. Note that components were sized in a manner that made all architectures ranging from turbojets to 7 fan distributed hybrid electric turbofans mathematically feasible. 181

5.54 Abstraction Layer 2: Engine DS2M. Gas connections: **red**, shaft power connections: **blue**, mechanical connections: **black**, fuel connections: **green**, oil connections: **cyan**, electrical connections: **magenta**. 184

5.55 Abstraction Layer 2: Engine DS2M Filtered With Convex Hulls (88.7% reduction). Gas connections: **red**, shaft power connections: **blue**, mechanical connections: **black**, fuel connections: **green**, oil connections: **cyan**, electrical connections: **magenta**. 185

5.56 Normalized Thrust Specific Fuel Consumption vs. Normalized Complexity . 186

5.57 Generic Turbojet Architecture. Gas connections: **red**, shaft power connections: **blue**, mechanical connections: **black**, fuel connections: **green**, oil connections: **cyan**, electrical connections: **magenta**. 189

5.58 Generic Turbojet Architecture DSM. Gas connections: **red**, shaft power connections: **blue**, mechanical connections: **black**, fuel connections: **green**, oil connections: **cyan**, electrical connections: **magenta**. 189

5.59	Generic Turbofan Architecture. Gas connections: red , shaft power connections: blue , mechanical connections: black , fuel connections: green , oil connections: cyan , electrical connections: magenta	190
5.60	Generic Turbofan Architecture DSM. Gas connections: red , shaft power connections: blue , mechanical connections: black , fuel connections: green , oil connections: cyan , electrical connections: magenta	191
5.61	Generic Geared Turbofan Architecture. Gas connections: red , shaft power connections: blue , mechanical connections: black , fuel connections: green , oil connections: cyan , electrical connections: magenta	192
5.62	Generic Geared Turbofan Architecture DSM. Gas connections: red , shaft power connections: blue , mechanical connections: black , fuel connections: green , oil connections: cyan , electrical connections: magenta	192
5.63	Generic Distributed Geared Turbofan Architecture. Gas connections: red , shaft power connections: blue , mechanical connections: black , fuel connections: green , oil connections: cyan , electrical connections: magenta	193
5.64	Generic Distributed Geared Turbofan Architecture DSM. Gas connections: red , shaft power connections: blue , mechanical connections: black , fuel connections: green , oil connections: cyan , electrical connections: magenta	193
5.65	Normalized Thrust Specific Fuel Consumption vs. Normalized Complexity. Normalized w.r.t generic geared turbofan architecture.	194
5.66	Generic Distributed Hybrid Electric Architecture. Gas connections: red , shaft power connections: blue , mechanical connections: black , fuel connections: green , oil connections: cyan , electrical connections: magenta	195

5.67	Generic Distributed Hybrid Electric Turbofan Architecture DSM. Gas connections: red , shaft power connections: blue , mechanical connections: black , fuel connections: green , oil connections: cyan , electrical connections: magenta	196
5.68	Generic Distributed Geared Hybrid Electric Architecture. Gas connections: red , shaft power connections: blue , mechanical connections: black , fuel connections: green , oil connections: cyan , electrical connections: magenta	197
5.69	Generic Distributed Geared Hybrid Electric Turbofan Architecture DSM. Gas connections: red , shaft power connections: blue , mechanical connections: black , fuel connections: green , oil connections: cyan , electrical connections: magenta	197
5.70	Normalized Thrust Specific Fuel Consumption vs. Normalized Complexity. Normalized w.r.t generic geared turbofan architecture.	198
5.71	Normalized Thrust Specific Fuel Consumption vs. Normalized Complexity. Normalized w.r.t generic geared turbofan architecture.	199
5.72	Normalized Thrust Specific Fuel Consumption vs. Normalized Mass. Normalized w.r.t generic geared turbofan architecture.	200
5.73	Normalized Thrust Specific Fuel Consumption vs. Normalized Mass. Normalized w.r.t generic geared turbofan architecture.	201
5.74	Normalized Thrust Specific Fuel Consumption vs. Normalized Mass. Normalized w.r.t generic geared turbofan architecture.	202
5.75	Normalized Thrust Specific Fuel Consumption vs. Normalized Mass. Normalized w.r.t generic geared turbofan architecture.	203
6.1	Google ARA Spiral 2 Diagram. Image source: [19]	206
6.2	Shifting industrial paradigms. Source: [20]	207

6.3	Dynamic use profile [21]	210
6.4	Use profile (0 off, 1 standby, 2 on) [21]	210
6.5	Power consumptions in each state (Watts) from [21]	210
6.6	Dendrogram representing clustering of preferences (Ward's method)	212
6.7	Part-worth utilities (preferences for different modules, prices and battery lives), [21]	213
6.8	List of 21 modules and the platform which they can occupy. Basic modules consist of existing technologies repackaged to fit within a reconfigurable mobile device. Intermediate modules represent technology with functionality found only in the most advanced integral devices. Advanced modules may not have any analog in integral smartphones. Module prices are based on data collected in [21].	216
6.9	Google ARA DS2M. Energy connections: green , mechanical connections: black , information connections: blue	217
6.10	Google ARA DS2M Filtered (82.5 % Reduction in Number of Connections). Energy connections: green , mechanical connections: black , information connections: blue	217
6.11	Trade-space utility analysis for ARA architectures evaluated according to all 5 clusters of users. Note this figure was generated during a team project and already appears in [22]. Note that this was generated via experience based rules. Some of these rules governing pairwise interaction between components were replaced via convex hulls after generation of this results.	218
6.12	Trade-space utility analysis for ARA architectures evaluated according to Cluster 4 part-worth utilities. Note this figure was generated during a team project and already appears in [22].	219

6.13	Architecture 5506 (Left) Benefit utility (1.57) Price utility (0.0034), Architecture 3184 (Middle) Benefit utility (2.52) Price utility (-1.82), Architecture 17397 (Right) Benefit utility (3.28) Price utility (-4.99). Basic modules are green, Advanced modules are red, Modules with only one type not highlighted.	220
6.14	Possible configurations.	222
6.15	Histogram of feasible configurations normalized by the most complex, using notional component and connection complexities. Higher complexities, broadly corresponding to price of modules were set for more advanced modules.	223
7.1	235
7.2	Object-Process Methodology to explicitly capture function and form	236
7.3	Object-Process Methodology to explicitly capture function and form	237
7.4	Architecture 1 (Turboelectric). Gas connections: red , shaft power connections: blue , mechanical connections: black , fuel connections: green , oil connections: cyan , electrical connections: magenta	252
7.5	Architecture 2 (Turboelectric). Gas connections: red , shaft power connections: blue , mechanical connections: black , fuel connections: green , oil connections: cyan , electrical connections: magenta	253
7.6	Architecture 3 (Turboelectric). Gas connections: red , shaft power connections: blue , mechanical connections: black , fuel connections: green , oil connections: cyan , electrical connections: magenta	254
7.7	Architecture 4 (Turboelectric). Gas connections: red , shaft power connections: blue , mechanical connections: black , fuel connections: green , oil connections: cyan , electrical connections: magenta	255

- 7.8 Architecture 5 (Turboelectric). Gas connections: **red**, shaft power connections: **blue**, mechanical connections: **black**, fuel connections: **green**, oil connections: **cyan**, electrical connections: **magenta**. 256
- 7.9 Architecture 6 (Turboelectric). Gas connections: **red**, shaft power connections: **blue**, mechanical connections: **black**, fuel connections: **green**, oil connections: **cyan**, electrical connections: **magenta**. 257
- 7.10 Architecture 7 (Turboelectric). Gas connections: **red**, shaft power connections: **blue**, mechanical connections: **black**, fuel connections: **green**, oil connections: **cyan**, electrical connections: **magenta**. 258
- 7.11 Architecture 8 (Turboelectric). Gas connections: **red**, shaft power connections: **blue**, mechanical connections: **black**, fuel connections: **green**, oil connections: **cyan**, electrical connections: **magenta**. 259
- 7.12 Architecture 9 (Turboelectric). Gas connections: **red**, shaft power connections: **blue**, mechanical connections: **black**, fuel connections: **green**, oil connections: **cyan**, electrical connections: **magenta**. 260
- 7.13 Architecture 10 (Turboelectric). Gas connections: **red**, shaft power connections: **blue**, mechanical connections: **black**, fuel connections: **green**, oil connections: **cyan**, electrical connections: **magenta**. 261
- 7.14 Architecture 11. Gas connections: **red**, shaft power connections: **blue**, mechanical connections: **black**, fuel connections: **green**, oil connections: **cyan**, electrical connections: **magenta**. 262
- 7.15 Architecture 12 (Turboelectric). Gas connections: **red**, shaft power connections: **blue**, mechanical connections: **black**, fuel connections: **green**, oil connections: **cyan**, electrical connections: **magenta**. 263

- 7.16 Architecture 13 (Turboelectric). Gas connections: **red**, shaft power connections: **blue**, mechanical connections: **black**, fuel connections: **green**, oil connections: **cyan**, electrical connections: **magenta**. 264
- 7.17 Architecture 14 (Turboelectric). Gas connections: **red**, shaft power connections: **blue**, mechanical connections: **black**, fuel connections: **green**, oil connections: **cyan**, electrical connections: **magenta**. 265
- 7.18 Architecture 15 (Turboelectric). Gas connections: **red**, shaft power connections: **blue**, mechanical connections: **black**, fuel connections: **green**, oil connections: **cyan**, electrical connections: **magenta**. 266
- 7.19 Architecture 16. Gas connections: **red**, shaft power connections: **blue**, mechanical connections: **black**, fuel connections: **green**, oil connections: **cyan**, electrical connections: **magenta**. 267
- 7.20 Architecture 17 (Turboelectric). Gas connections: **red**, shaft power connections: **blue**, mechanical connections: **black**, fuel connections: **green**, oil connections: **cyan**, electrical connections: **magenta**. 268
- 7.21 Architecture 18 (Turboelectric). Gas connections: **red**, shaft power connections: **blue**, mechanical connections: **black**, fuel connections: **green**, oil connections: **cyan**, electrical connections: **magenta**. 269
- 7.22 Architecture 19 (Turboelectric). Gas connections: **red**, shaft power connections: **blue**, mechanical connections: **black**, fuel connections: **green**, oil connections: **cyan**, electrical connections: **magenta**. 270
- 7.23 Architecture 20 (Turboelectric). Gas connections: **red**, shaft power connections: **blue**, mechanical connections: **black**, fuel connections: **green**, oil connections: **cyan**, electrical connections: **magenta**. 271

7.24	Architecture 21. Gas connections: red , shaft power connections: blue , mechanical connections: black , fuel connections: green , oil connections: cyan , electrical connections: magenta	272
7.25	Architecture 22. Gas connections: red , shaft power connections: blue , mechanical connections: black , fuel connections: green , oil connections: cyan , electrical connections: magenta	273
7.26	Architecture 23 (Turboelectric). Gas connections: red , shaft power connections: blue , mechanical connections: black , fuel connections: green , oil connections: cyan , electrical connections: magenta	274
7.27	Architecture 24 (Turboelectric). Gas connections: red , shaft power connections: blue , mechanical connections: black , fuel connections: green , oil connections: cyan , electrical connections: magenta	275
7.28	Architecture 25 (Turboelectric). Gas connections: red , shaft power connections: blue , mechanical connections: black , fuel connections: green , oil connections: cyan , electrical connections: magenta	276
7.29	Architecture 26 (Turboelectric). Gas connections: red , shaft power connections: blue , mechanical connections: black , fuel connections: green , oil connections: cyan , electrical connections: magenta	277
7.30	Architecture 27 (Turboelectric). Gas connections: red , shaft power connections: blue , mechanical connections: black , fuel connections: green , oil connections: cyan , electrical connections: magenta	278
7.31	Architecture 28 (Turboelectric). Gas connections: red , shaft power connections: blue , mechanical connections: black , fuel connections: green , oil connections: cyan , electrical connections: magenta	279

7.32 Architecture 29 (Turboelectric). Gas connections: **red**, shaft power connections: **blue**, mechanical connections: **black**, fuel connections: **green**, oil connections: **cyan**, electrical connections: **magenta**. 280

7.33 Architecture 30 (Turboelectric). Gas connections: **red**, shaft power connections: **blue**, mechanical connections: **black**, fuel connections: **green**, oil connections: **cyan**, electrical connections: **magenta**. 281

7.34 Architecture 31. Gas connections: **red**, shaft power connections: **blue**, mechanical connections: **black**, fuel connections: **green**, oil connections: **cyan**, electrical connections: **magenta**. 282

7.35 Architecture 32 (Turboelectric). Gas connections: **red**, shaft power connections: **blue**, mechanical connections: **black**, fuel connections: **green**, oil connections: **cyan**, electrical connections: **magenta**. 283

7.36 Architecture 33 (Turboelectric). Gas connections: **red**, shaft power connections: **blue**, mechanical connections: **black**, fuel connections: **green**, oil connections: **cyan**, electrical connections: **magenta**. 284

7.37 Architecture 34 (Turboelectric). Gas connections: **red**, shaft power connections: **blue**, mechanical connections: **black**, fuel connections: **green**, oil connections: **cyan**, electrical connections: **magenta**. 285

7.38 Architecture 35. Gas connections: **red**, shaft power connections: **blue**, mechanical connections: **black**, fuel connections: **green**, oil connections: **cyan**, electrical connections: **magenta**. 286

7.39 Architecture 36 (Turboelectric). Gas connections: **red**, shaft power connections: **blue**, mechanical connections: **black**, fuel connections: **green**, oil connections: **cyan**, electrical connections: **magenta**. 287

- 7.40 Architecture 37 (Turboelectric). Gas connections: **red**, shaft power connections: **blue**, mechanical connections: **black**, fuel connections: **green**, oil connections: **cyan**, electrical connections: **magenta**. 288
- 7.41 Architecture 38 (Turboelectric). Gas connections: **red**, shaft power connections: **blue**, mechanical connections: **black**, fuel connections: **green**, oil connections: **cyan**, electrical connections: **magenta**. 289
- 7.42 Architecture 39. Gas connections: **red**, shaft power connections: **blue**, mechanical connections: **black**, fuel connections: **green**, oil connections: **cyan**, electrical connections: **magenta**. 290
- 7.43 Architecture 40. Gas connections: **red**, shaft power connections: **blue**, mechanical connections: **black**, fuel connections: **green**, oil connections: **cyan**, electrical connections: **magenta**. 291
- 7.44 Architecture 41 (Turboelectric). Gas connections: **red**, shaft power connections: **blue**, mechanical connections: **black**, fuel connections: **green**, oil connections: **cyan**, electrical connections: **magenta**. 292
- 7.45 Architecture 42 (Turboelectric). Gas connections: **red**, shaft power connections: **blue**, mechanical connections: **black**, fuel connections: **green**, oil connections: **cyan**, electrical connections: **magenta**. 293
- 7.46 Architecture 43 (Turboelectric). Gas connections: **red**, shaft power connections: **blue**, mechanical connections: **black**, fuel connections: **green**, oil connections: **cyan**, electrical connections: **magenta**. 294
- 7.47 Architecture 44 (Turboelectric). Gas connections: **red**, shaft power connections: **blue**, mechanical connections: **black**, fuel connections: **green**, oil connections: **cyan**, electrical connections: **magenta**. 295

7.48 Architecture 45 (Turboelectric). Gas connections: red , shaft power connections: blue , mechanical connections: black , fuel connections: green , oil connections: cyan , electrical connections: magenta	296
7.49 Architecture 46 (Turboelectric). Gas connections: red , shaft power connections: blue , mechanical connections: black , fuel connections: green , oil connections: cyan , electrical connections: magenta	297
7.50 Architecture 47. Gas connections: red , shaft power connections: blue , mechanical connections: black , fuel connections: green , oil connections: cyan , electrical connections: magenta	298
7.51 Architecture 48 (Turboelectric). Gas connections: red , shaft power connections: blue , mechanical connections: black , fuel connections: green , oil connections: cyan , electrical connections: magenta	299
7.52 Architecture 49 (Turboelectric). Gas connections: red , shaft power connections: blue , mechanical connections: black , fuel connections: green , oil connections: cyan , electrical connections: magenta	300
7.53 Architecture 50. Gas connections: red , shaft power connections: blue , mechanical connections: black , fuel connections: green , oil connections: cyan , electrical connections: magenta	301
7.54 Architecture 51 (Turboelectric). Gas connections: red , shaft power connections: blue , mechanical connections: black , fuel connections: green , oil connections: cyan , electrical connections: magenta	302
7.55 Architecture 52 (Turboelectric). Gas connections: red , shaft power connections: blue , mechanical connections: black , fuel connections: green , oil connections: cyan , electrical connections: magenta	303

7.56	Architecture 53. Gas connections: red , shaft power connections: blue , mechanical connections: black , fuel connections: green , oil connections: cyan , electrical connections: magenta	304
7.57	Architecture 54. Gas connections: red , shaft power connections: blue , mechanical connections: black , fuel connections: green , oil connections: cyan , electrical connections: magenta	305
7.58	Architecture 55 (Turboelectric). Gas connections: red , shaft power connections: blue , mechanical connections: black , fuel connections: green , oil connections: cyan , electrical connections: magenta	306
7.59	Architecture 56 (Turboelectric). Gas connections: red , shaft power connections: blue , mechanical connections: black , fuel connections: green , oil connections: cyan , electrical connections: magenta	307
7.60	Architecture 57 (Turboelectric). Gas connections: red , shaft power connections: blue , mechanical connections: black , fuel connections: green , oil connections: cyan , electrical connections: magenta	308
7.61	Architecture 58 (Turboelectric). Gas connections: red , shaft power connections: blue , mechanical connections: black , fuel connections: green , oil connections: cyan , electrical connections: magenta	309
7.62	Architecture 59. Gas connections: red , shaft power connections: blue , mechanical connections: black , fuel connections: green , oil connections: cyan , electrical connections: magenta	310
7.63	Architecture 60 (Turboelectric). Gas connections: red , shaft power connections: blue , mechanical connections: black , fuel connections: green , oil connections: cyan , electrical connections: magenta	311

7.64 Architecture 61. Gas connections: **red**, shaft power connections: **blue**, mechanical connections: **black**, fuel connections: **green**, oil connections: **cyan**, electrical connections: **magenta**. 312

7.65 Architecture 62 (Turboelectric). Gas connections: **red**, shaft power connections: **blue**, mechanical connections: **black**, fuel connections: **green**, oil connections: **cyan**, electrical connections: **magenta**. 313

7.66 Architecture 63. Gas connections: **red**, shaft power connections: **blue**, mechanical connections: **black**, fuel connections: **green**, oil connections: **cyan**, electrical connections: **magenta**. 314

7.67 Architecture 64. Gas connections: **red**, shaft power connections: **blue**, mechanical connections: **black**, fuel connections: **green**, oil connections: **cyan**, electrical connections: **magenta**. 315

7.68 Architecture 65 (Turboelectric). Gas connections: **red**, shaft power connections: **blue**, mechanical connections: **black**, fuel connections: **green**, oil connections: **cyan**, electrical connections: **magenta**. 316

7.69 Architecture 66 (Turboelectric). Gas connections: **red**, shaft power connections: **blue**, mechanical connections: **black**, fuel connections: **green**, oil connections: **cyan**, electrical connections: **magenta**. 317

7.70 Architecture 67. Gas connections: **red**, shaft power connections: **blue**, mechanical connections: **black**, fuel connections: **green**, oil connections: **cyan**, electrical connections: **magenta**. 318

7.71 Architecture 68. Gas connections: **red**, shaft power connections: **blue**, mechanical connections: **black**, fuel connections: **green**, oil connections: **cyan**, electrical connections: **magenta**. 319

- 7.72 Architecture 69. Gas connections: **red**, shaft power connections: **blue**, mechanical connections: **black**, fuel connections: **green**, oil connections: **cyan**, electrical connections: **magenta**. 320
- 7.73 Architecture 70. Gas connections: **red**, shaft power connections: **blue**, mechanical connections: **black**, fuel connections: **green**, oil connections: **cyan**, electrical connections: **magenta**. 321
- 7.74 Architecture 71. Gas connections: **red**, shaft power connections: **blue**, mechanical connections: **black**, fuel connections: **green**, oil connections: **cyan**, electrical connections: **magenta**. 322

Chapter 1

Introduction

1.1 Motivation

In 1992, the United Nations Intergovernmental Panel on Climate Change Special Report on Aviation and the Global Atmosphere stated that 2% of all anthropogenic carbon dioxide emissions were due to aircraft [23]. The report predicted that by the year 2050 aviation would account for 3% of all anthropogenic carbon dioxide emissions. To limit the impact of aviation on the climate, in 2008, the Air Transport Action Group (ATAG), representing aircraft manufacturers, engine manufacturers and a wide range of organizations from the aviation industry, set aspirational goals to improve aircraft fleet fuel efficiency by 1.5% annually [24]. The long term goal of ATAG was to improve fuel efficiency by 50% relative to 2005 levels by the year 2050 [24]. In 2010, the United Nations International Civil Aviation Organization (ICAO) set goals to improve fuel efficiency by 2% per year until 2020 [25]. A growing body of work, however, continues to suggest that these long term fuel efficiency goals can only be met, if there are changes in aircraft and engine architecture (e.g. subsystems and the way they are connected to one another) [10, 26].

Changes in architecture to improve performance have been associated with increased complexity and development effort [27]. As we have demanded greater performance from aerospace and defense systems their unit cost, which is related to their development cost, has increased. Figure 1.1 from the former CEO of Lockheed Martin, graphically depicts this trend for fixed wing aircraft from the Wright Model A to the F35 [3]. In figure 1.1 these trends were extrapolated to show that if they continued, the entire defense budget would be consumed by a single aircraft by the late 21st century.

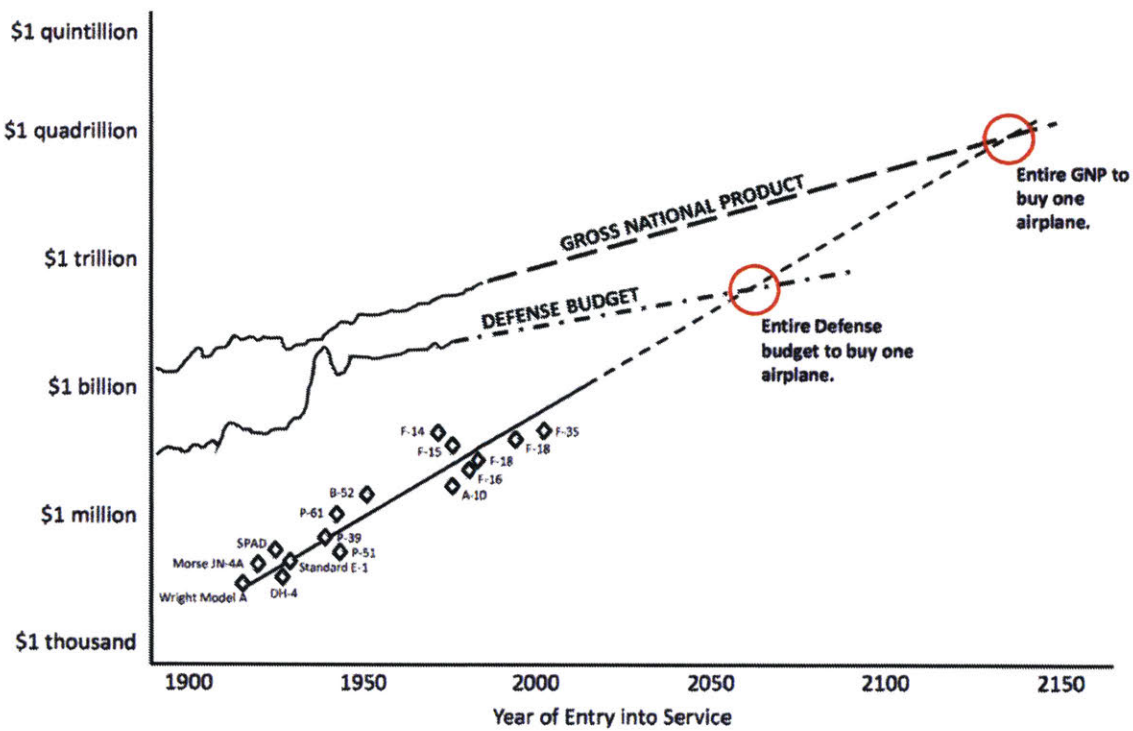


Figure 1.1: Increasing cost of aerospace and defense systems taken taken from keynote address at the 2013 Conference for Systems Engineering Research given by Paul Eremenko, the former deputy director of the DARPA Tactical Technology Office [2]. Original source: former CEO of Lockheed Martin N. Augustine [3].

Figure 1.2 from the former Deputy Director of the DARPA Tactical Technology Office depicts the correlation between development time (a measure of development effort) and complexity for aerospace and defense systems shown in blue. Development time for systems like the

JSF (F35) and F22 have exceeded 10 years. Increases in expected development time result in increased requirements uncertainty. This can lead to systems being designed to meet a more demanding set of requirements that change over time, complexity growth, and rework.

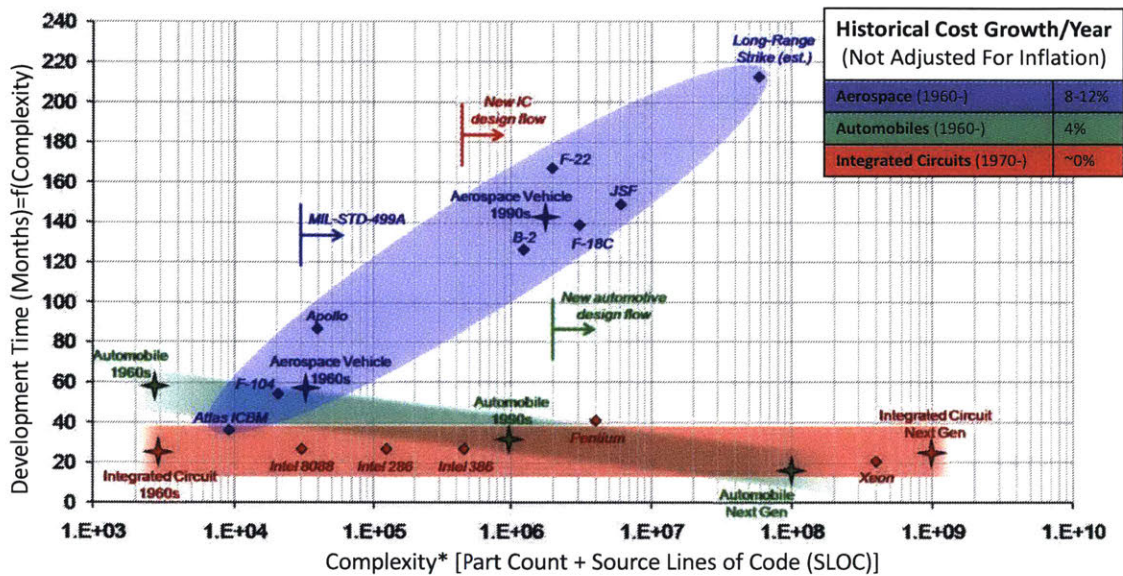
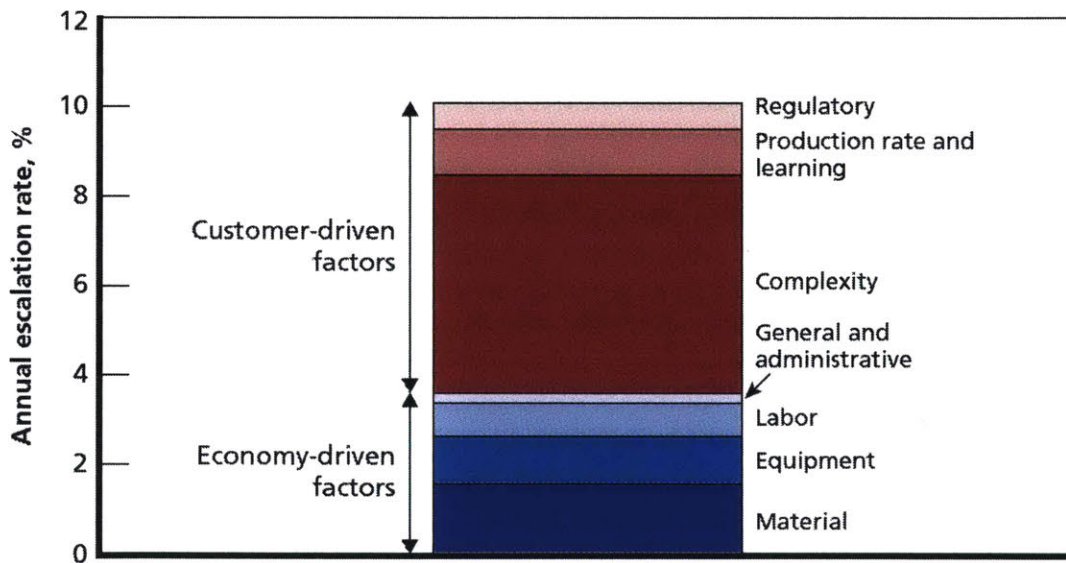


Figure 1.2: Correlation of a measure of structural and software complexity with aerospace development time and cost. Taken from keynote address at the 2013 Conference for Systems Engineering Research, the former deputy director of the DARPA Tactical Technology Office [2].

There is also evidence of a causal relationship between development effort and complexity both through human experiments reported in [27] and in a report by the RAND corporation [4]. The example in figure 1.3 from [4] depicts the causes of price escalation for the F22A compared to the F15A fighter aircraft. While a number of different causes are depicted including regulatory reasons and material costs, around half of annual price escalation rate is attributed to complexity driven by the customer (the United States Air Force).

The increasing demands on the performance of commercial aviation, the relationship between complexity, performance and development effort and the potential for major aircraft

architectural changes in the first half of the 21st century motivate exploration of the set of possible architectures during conceptual design in a manner that explicitly considers trade-offs between complexity and performance. The focus of this research is the development of a computational approach to carrying out this exploration. Previous work has suggested that new propulsion architectures may drive configuration change in aircraft. We delve more deeply into the history of air-breathing propulsion in the context of performance and complexity in the remainder of this chapter.



RAND MG696-S.1

Figure 1.3: Contributors to Price Escalation from the F-15A (1975) to the F-22A (2005). Taken from keynote address at the 2013 Conference for Systems Engineering Research given by Paul Eremenko, the former deputy director of the DARPA Tactical Technology Office [2]. Original source: RAND corporation [4].

1.1.1 Air-Breathing Propulsion System Architecture

The trend in many facets of the aerospace industry and engineering systems in general, has been to converge on a system architecture, develop and refine the subsystems within the selected system architecture and migrate to new architectures only when performance gains through refinements at the subsystem level yield diminishing returns [28], [6]. This is depicted in figure 1.4 in which performance benefits with development effort/time at both the system and subsystem levels follow S-curves [6]. The rate of change of performance with development effort is small, then it accelerates and finally plateaus, necessitating changes at the system level. Note also that notionally improvements in performance at the subsystem level yield diminishing returns at the system level. Dr. Carlos Gorbea's PhD thesis was motivated by this trend for road vehicles [5]. It argued that subsystem level improvements in internal combustion engines were yielding diminishing returns at the system level (notionally depicted in figure 1.4) and that we had entered a period of "architectural competition" in which high level architectural changes would be the driver of improvements in performance metrics. This served to motivate Gorbea's work on architecture generation and evaluation for automotive hybrid electric powertrains.

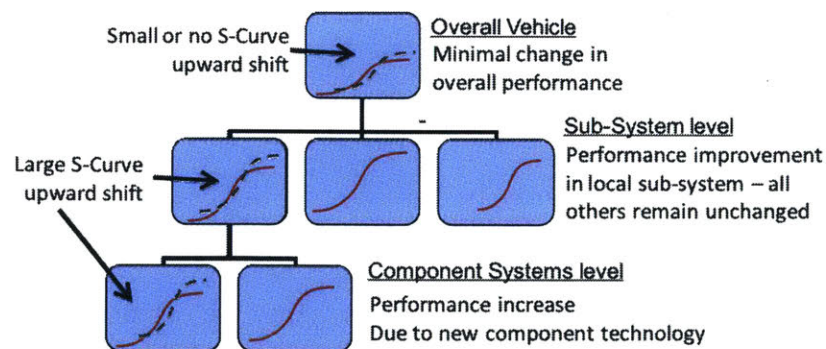


Figure 1.4: Performance S-curves at the system and subsystem levels. Taken from Dr. Carlos Gorbea's PhD thesis [5]. Original Source: [6].

Departures from legacy architectures for gas turbine engines, i.e. major changes in the components within a system or the manner in which they are connected, are discrete and infrequent events [29, 8, 30]. While architectural changes have been one option for generating improvements in performance metrics like fuel consumption, they have often resulted in higher complexity and development cost [27] and one may argue that we are currently nearing the performance plateaus described in [6].

The earliest work on gas turbines dates back to 1791 and a patent by John Barber in the United Kingdom [31]. Successful electric power generation from gas turbines was first achieved by a team led by Aurel Stodola at the Brown Boveri Company in Switzerland in 1939 [32]. While Sir Frank Whittle is generally credited with developing and building the first gas turbine engine propulsion systems for aircraft, three different individuals in Europe contributed to their journey from conception to operational flight. The first gas turbine patent for aircraft propulsion was filed in Paris, France on the May 3, 1921 by Maxime Guillaume (patent number 534.801) [33], the first gas turbine engine was designed and built by Sir Frank Whittle in 1937, and the first operational engine was designed and built by Hans von Ohain in Germany in 1939 [31].

Gas turbine aircraft propulsion represented a architectural and technological paradigm shift from the piston powered designs that dominated the first half of the 20th century. The change from piston powered to gas turbine architectures allowed higher levels of performance to be achieved with what was arguably a less complex design (fewer subsystems, independently moving parts) [34]. It therefore may be described as the beginning of a new S-curve in performance. With demands for greater performance (greater power, increased flight speed), radial piston engines increased in complexity until they were superseded by gas turbine engines in the 1940s. The most powerful and likely most complex radial piston engine was the Lycoming XR-7755-3 (see figure 1.5), which had 36 cylinders and “featured nine dual-lobe

overhead camshafts, which shifted axially for takeoff and cruising efficiency, and a two-speed, geared, dual-rotation propeller drive” [7]. This engine was designed during WWII for “high takeoff power and low fuel consumption ” for a long range bomber commissioned by the US Air Force but was “obsolete” by 1946 when it was first tested due the emergence of gas turbine engines [7].

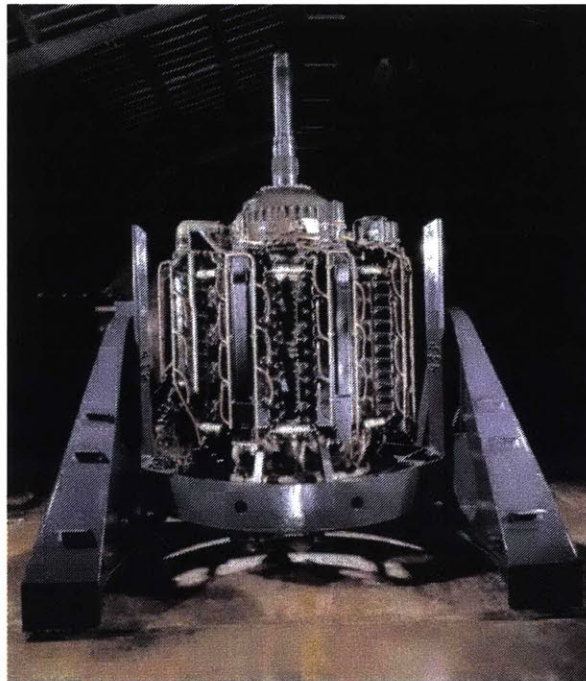


Figure 1.5: Lycoming XR-7755-3, Radial 36 Engine Displayed at Smithsonian Institution, the most powerful radial piston engine ever built (1946) [7]

According to [34] and [31], when referring to gas turbine engines “Whittle . . . stressed the great simplicity of his engine [turbojet]. Hives [Director of Rolls Royce] commented, ‘We’ll soon design the bloody simplicity out of it.’ ”. Modern gas turbine engines are far more complex than their first instantiations in the 1930s and 1940s [8], [35]. The gas turbine engine architectural paradigm has followed a qualitatively similar evolution in complexity and performance to piston powered paradigm.

Gas turbine engines in commercial aviation have gone through a few major architectural changes, evolving from single spool turbojets, to multi-spool turbojets, later into medium bypass ratio turbofans, high bypass ratio turbofans, very high bypass ratio turbofans and geared turbofans [8]. Air-breathing gas turbine engines operate via the Brayton thermodynamic cycle which involves adiabatic compression, heat addition at constant pressure, adiabatic flow expansion and cooling of air to ambient temperature at constant pressure [36].

Turbojets, have a single gas path in which gas passes through all stages of the Brayton cycle. For high bypass ratio (see nomenclature section) turbofans is that the majority of the mass flow through the engine bypasses the core (consisting of compressors, burners and turbines) via a fan which is typically situated upstream of the core. Thrust is proportional to the rate of change of momentum of gas (conservation of momentum). Rate of change of momentum, in turn, is proportional to the product of mass flow and change in velocity. The kinetic energy imparted to the flow is proportional to the difference between squares of velocities of the inlet and outlet flow. By passing a larger mass flow and imparting a lower change in velocity (i.e. by using a low pressure ratio fan), the same thrust can be achieved with lower kinetic energy addition. The ratio of the thrust power needed to propel the aircraft and the kinetic energy added to the flow is called propulsive efficiency [36]. A high bypass ratio engine increases propulsive efficiency by increasing total mass flow and reducing the total kinetic energy imparted to the flow for a given thrust level; lowering fan pressure ratio increases propulsive efficiency by lowering the mean exhaust velocity for a given thrust.

An early turbojet produced in the United States is the JT3 from Pratt & Whitney from 1950, shown in figure 1.6 from [8]. This engine was later developed into the JT3D used in the Boeing 707 and the TF33 used in the B52 [8].

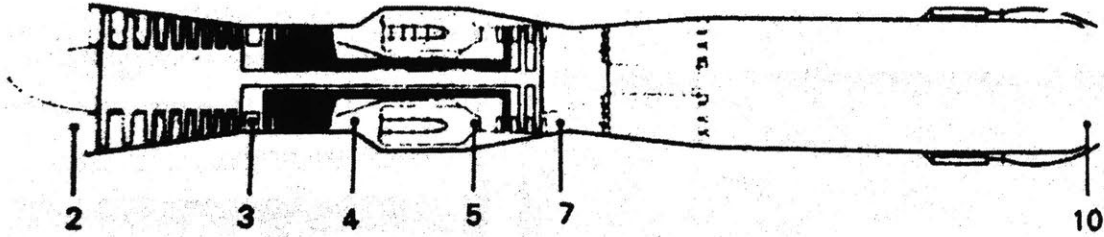


Figure 1.6: JT3 Turbojet Engine [8].

Perhaps the first major architectural change for gas turbine engines was the change from turbojets and low bypass ratio turbofans to high bypass ratio turbofans with the advent of the JT9D by Pratt & Whitney in 1966 (see Figure 1.7) which was the launch engine of the 747 [8]. The JT9D fundamentally changed engine architecture by introducing a large fan in front of the core of the engine. This architectural change increased the number of turbine stages from 4 to 6, increased the number of compression stages from 13 to 14 and introduced turbine air-cooling along with a much larger fan and bypass nozzle compared to the JT3D. The result was an approximately 26% reduction in fuel consumption compared to the JT3, but higher part count, a larger number of interfaces and therefore greater complexity [8].

The Rolls-Royce RB211 is another example of architectural change in the pursuit of performance leading to increased complexity. The RB211 (see figure 1.8) was Rolls-Royce's first three spool (three independently rotating shafts) engine [8]. Three spools allowed rotational speed in compression and expansion stages to be decoupled. By relaxing constraints from two values of rotational speed to three, attainable engine performance given subsystem level technology may be improved as per the "*Lower Bounding Principle of Optimization*", which states that relaxing constraints in any optimization problem allows as good or better

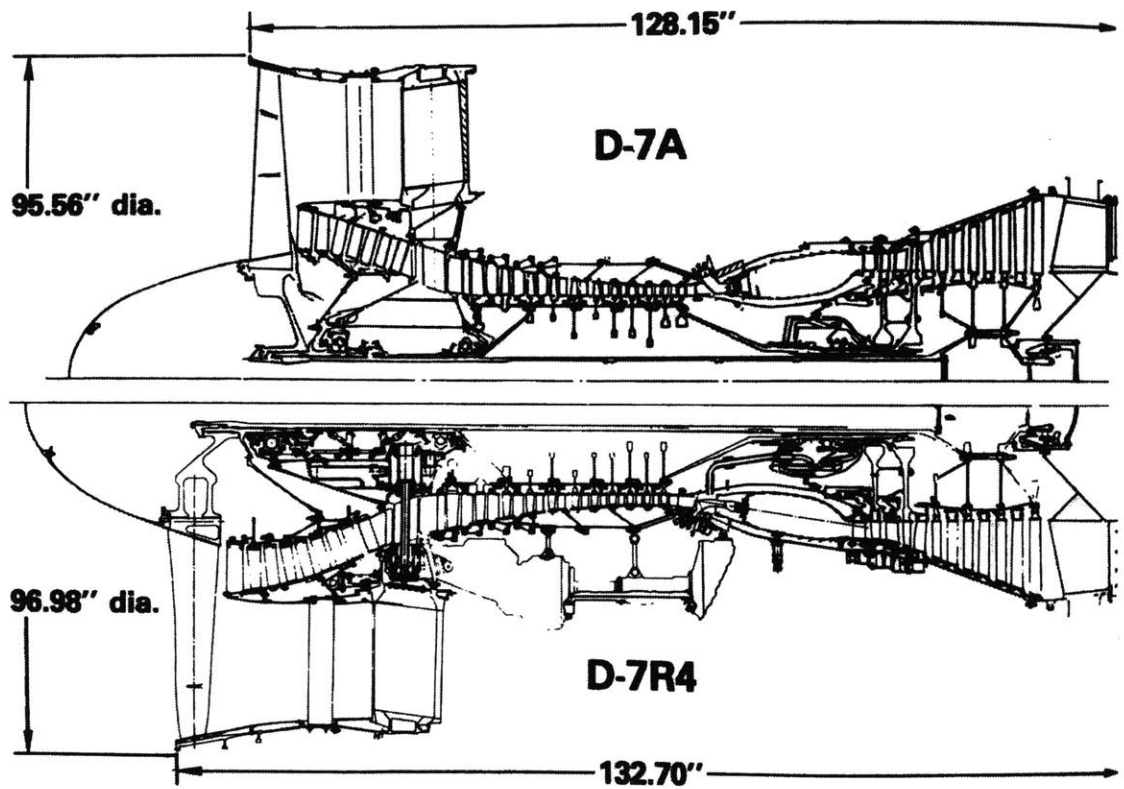


Figure 1.7: JT9D [8]. Two Shaft Turbofan. Fan pressure ratio (FPR) 1.6, Bypass ratio (BPR) 5.15, Overall Pressure Ratio (OPR) 33. See nomenclature section for definitions.

solutions to be found [37].

According to [8] Rolls-Royce experienced significant cost and schedule overruns, ultimately resulting in insolvency in 1971 primarily related to new fan blade material selection. Rolls-Royce's three spool architecture has commercially fared very well since the budget/schedule overruns that the first of its kind experienced. The Trent series of three spool engines now are widely used in commercial aviation (e.g. on the Boeing 787, 777 as well as the A380, A350 [38]).

Another more recent example of architectural change is the advent of very high bypass ratio

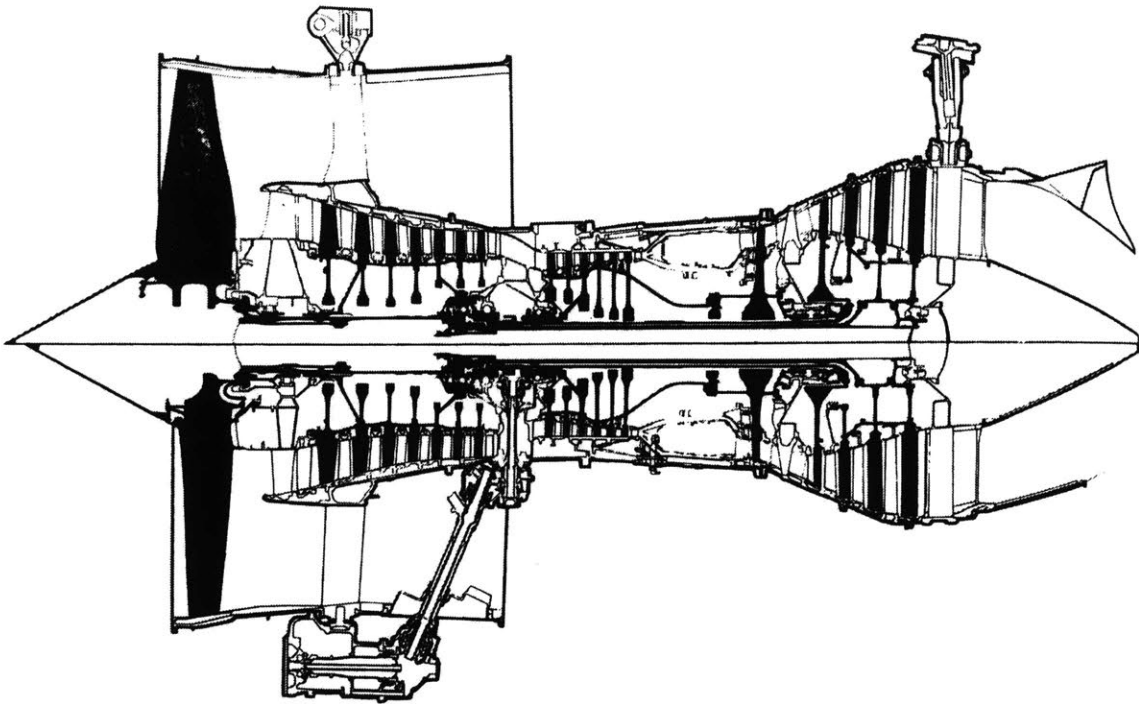


Figure 1.8: Rolls-Royce RB211 524G [8]. Three Shaft Turbofan. Fan pressure ratio (FPR) Not Available, Bypass ratio (BPR) 4.3, Overall Pressure Ratio (OPR) 22.2. See nomenclature section for definitions.

geared turbofans in the 20,000 pound and above thrust class shown in figure 1.9. In a geared turbofan the turbine's angular velocity is decoupled from the fan's angular velocity via a gearbox. This means that fan angular velocity and pressure ratio can be lower while at the same time allowing the turbine to have fewer and smaller stages, reducing noise and increasing propulsive efficiency [29]. The addition of a gearbox fundamentally changes engine architecture enabling a 20dB reduction in noise while improving fuel consumption by 12% and reducing weight. This allows lower fuel consumption operation of aircraft with lower noise levels at airports near heavily populated areas [29]. While the geared turbofan provides a number of performance metric benefits they come at the cost of a 40% increase in complexity [27] and took almost 30 years to make its way from conception to flight for the 20,000 lbf thrust class, due in part to difficulties scaling gearboxes from smaller engines

[39].

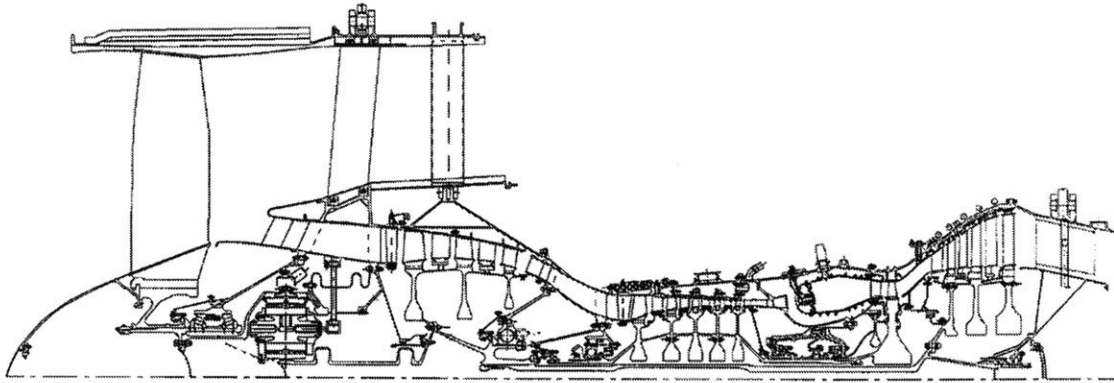


Figure 1.9: Pratt & Whitney Geared Turbofan Architecture [9]. Fan pressure ratio (FPR) unavailable, Bypass ratio (BPR) 12, Overall Pressure Ratio (OPR) unavailable. See nomenclature section for definitions.

Along with high level architectural changes increasing engine complexity and performance, subsystems have also become more complex, particularly the mechanical, supporting air and cooling/lubrication systems which provide for turbine disk and blade cooling as and the active tip clearance control of turbine blades via cooling flows to the turbine case [35].

There are other examples of changes in architecture providing performance benefits in air-breathing propulsion systems while increasing complexity. One case is the Kuznetsov NK-93/94 counter-rotating propfan shown in figure 1.10 which was developed in the 1980s but never entered service [8]. The Kuznetsov NK-93/94 has the greatest bypass ratio (16.6) of any engine in the [8] dataset but was never certified and put into production. The fan pressure ratio of this engine was not available, but given the high bypass ratio it was on the order of if not lower than existing and proposed ultrahigh bypass ratio geared turbofans. Reference [8] indicated that this engine has a large number of subsystems, with variable geometry, multiple gearboxes and counter-rotating fans with the option for cryogenic propellants. It is also quoted as being “the most efficient” gas turbine engine ever tested. This is depicted

graphically in figure 1.11 in which Thrust Specific Fuel Consumption (TSFC) is plotted against time of first flight. The Kuznetsov NK-93/94 is an example of how architectural change delivered radical performance improvements while increasing complexity to the point where it may have contributed to preventing certification. It is important to note however that other factors were at play also, certification of this engine was occurring at a similar time as the collapse of the USSR, due to the very high bypass ratio low wing aircraft would likely suffer difficulty with engine integration, and while it was the most efficient engine at sea level static conditions, there was no data to describing its performance at cruise conditions.

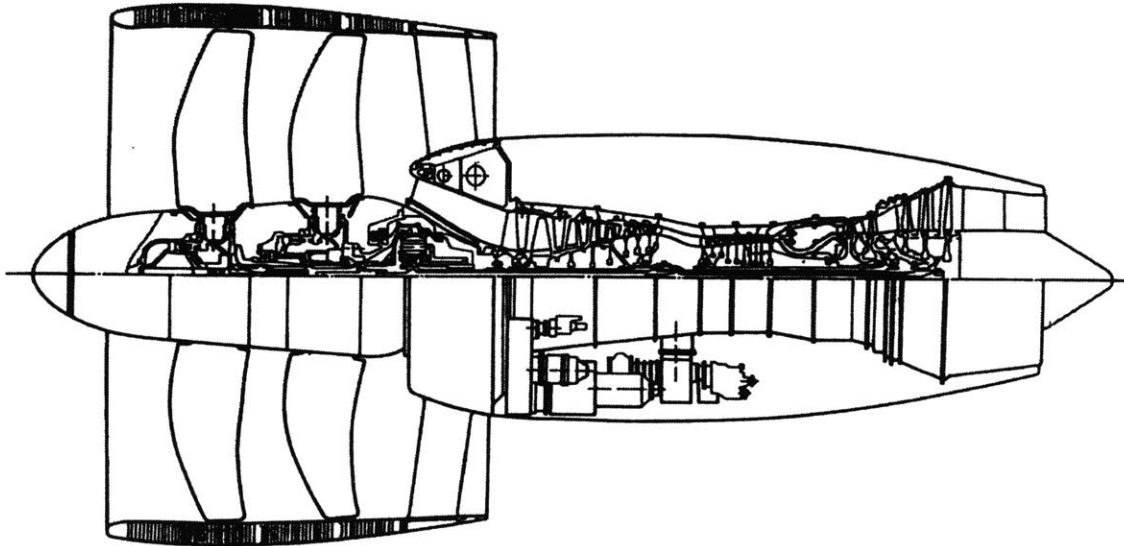


Figure 1.10: Kuznetsov NK93/94. Diagram from Jane's Aero-Engines [8]. Fan pressure ratio (FPR) unavailable, Bypass ratio (BPR) 16.6, Overall Pressure Ratio (OPR) 37. See nomenclature section for definitions.

The literature suggests that long term performance improvements in commercial aircraft will be achieved to large degree through architectural change [10]. One example of possible future architectural change can be found in the [10], which investigated aircraft concepts in the 2030-2035 time frame and targeted a 70% reduction in fuel burn. One of the three major findings was that aircraft and engine configuration change were among the most important

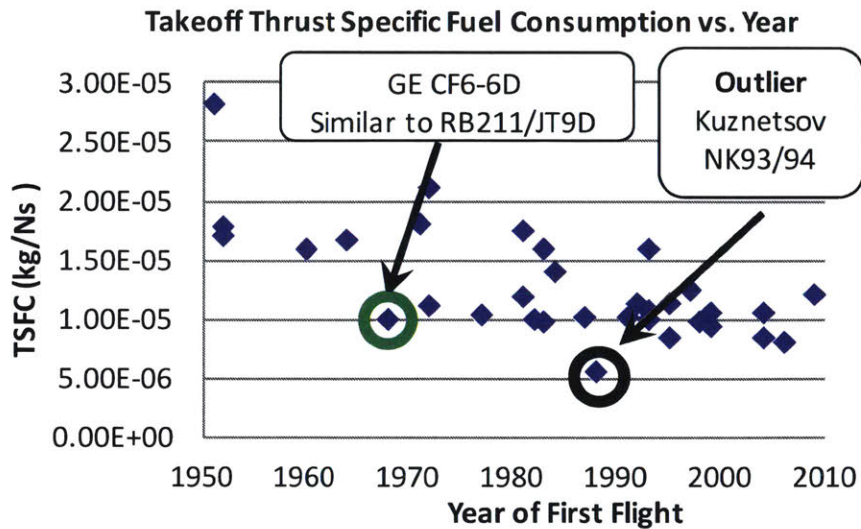


Figure 1.11: Takeoff Thrust Specific Fuel Consumption vs. Year of First Flight. Note that the Pratt & Whitney Geared Turbofan is not listed here since there is no publicly available data regarding its TSFC at takeoff [8].

sources of fuel burn savings. This is illustrated graphically in figure 1.12 which shows a breakdown of the relative benefits from different design decisions on fuel consumption, noise and LTO NO_x emissions [10].

In summary, changes in architecture can and have led to performance improvements, they have also been associated with increases in complexity [27]. This is particularly true when confined to a particular technological paradigm (like radial piston engines for example). It is important to note that radical architectural changes have also reduced complexity while improving performance (move to turbojets) [6]. The literature suggests that engine and aircraft architecture is likely to undergo major architectural change in the first half of the 21st century. The potential benefits of this change must be carefully balanced against increases in complexity. This motivates consideration of broad sets architectures at the conceptual design stage which is the focus of this thesis.

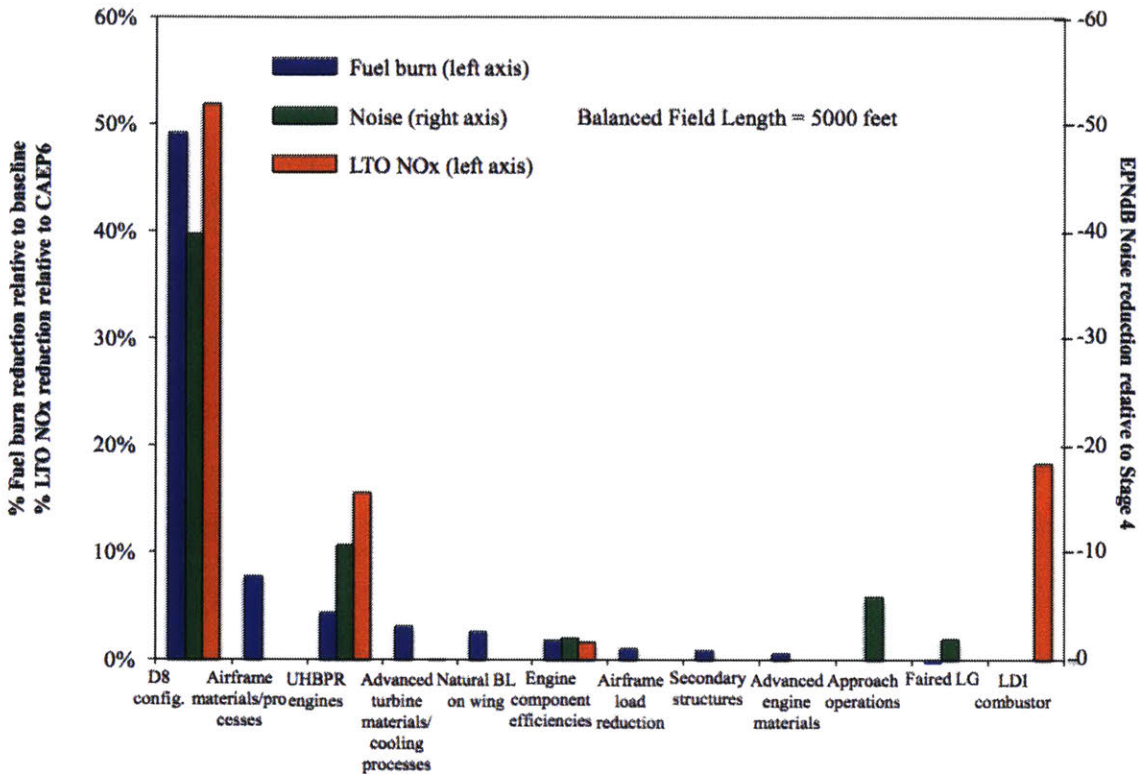


Figure 1.12: NASA N+3: Breakdown of the relative benefits to fuel consumption, noise and LTO NOx emissions from different sources of design changes. The D8 (double bubble) configuration provides the greatest improvement [10].

1.2 Outline of Thesis

The first three chapters of this work are broadly structured according to G. H. Heilmeier’s research catechism which consists of the following 7 questions given below [40]. The Heilmeier questions *H5-H7* are tailored towards research at an earlier phase of progress and are not explicitly addressed here.

H1: What are you trying to do? Articulate your objectives using absolutely no jargon.

H2: How is it done today and what are the limits of current practice?

H3: Who cares? If it is successful, what difference will it make?

H4: What is new in your approach and why do you think it will be successful?

H5: What are the risks and the payoffs?

H6: How much will it cost? How long will it take?

H7: What are the “midterm” and “final” exams to check for its success?

We summarize the contents of the chapters below:

- **Chapter 1: Introduction:** The *Introduction* touched on questions *H1* and *H3*. The high level objective of this work (*H1*) is to develop a performance-complexity tradespace exploration framework of engineering systems in light of trends of complexity/development effort growth in aerospace and defense systems (*H3*).
- **Chapter 2: Literature Review:** The *Literature Review* focuses in more detail on the limits of current practice to performance-complexity tradespace exploration (*H2*) with a focus on computational approaches to generating architectures. It outlines the gap in current published work, laying the groundwork for a formal problem statement and research question. It also discusses why addressing the gaps in the research help achieve our objectives (*H4*).
- **Chapter 3: Thesis Statement:** The *Problem Statement* summarizes the gap in current practice, defines the research questions and summarizes the approach to answering the research question (*H1-H4*).
- **Chapter 4: Approach:** This chapter focuses on the proposed approach combining architecture exploration, complexity quantification and performance calculation (*H4*).
- **Chapter 5: Air-Breathing Propulsion Case Study:** The *Air-Breathing Propulsion Case Study* applies the approach to the exploration of different architectures of an aircraft engine.

- **Chapter 6: Reconfigurable Mobile Device Case Study:** The *Reconfigurable Mobile Device Case Study* applies the approach to the exploration of different architectures of a reconfigurable mobile device.
- **Chapter 7: Summary, Conclusions and Future Work:** This chapter outlines the methodological and domain specific contributions of the thesis. It then summarizes opportunities for future work.

Chapter 2

Literature Review

In Chapter 1 we discussed ICAO/ATAG fuel efficiency goals for aircraft as well as research suggesting that aircraft/engine architecture will likely change to meet those goals. We described how architectures have evolved in the aerospace industry and motivated the need for using computational approaches to consider broad sets of architectures at the conceptual design phase considering both complexity and performance. The discussion in Chapter 1 therefore primarily focused on Heilmeier questions *H1: objective of research* and *H3: to whom this research is relevant*. Having established the utility of generating broad sets of concepts computationally, taking into account both complexity and performance in Chapter 1, Chapter 2 provides a mathematical grounding of why considering a broad set of architectures is important based on the literature and surveys existing approaches to architecture generation and evaluation. It then outlines the gap in the literature and summarizes where the methodological contributions of this work are in relation to that gap. Prior to delving into the literature review we define the following terms which can also be found in the glossary:

Concept: A concept is a product or system vision, idea, notion, or mental image that

maps function to form. It is a scheme for the system and how it works. It embodies a sense of how the system will function and an abstraction of the system form. It is a simplification of the system architecture that allows for high-level reasoning [41].

Architecture: The embodiment of concept, and the allocation of physical/informational function to elements of form, and definition of interfaces among the elements and with the surrounding context [42].

Configuration: An arrangement of existing elements of form [1].

Feasible Concept, Architecture or Configuration: A concept, architecture or configuration that is able to meet requirements.

False Negative: Feasible concept, architecture or configuration that is removed from consideration.

False Positive: Infeasible concept, architecture or configuration that is kept in consideration.

Note that the definitions of concept, architecture and configuration are related. One way to think about these three ideas is in terms of abstraction. A concept is an abstract mapping of function to form, an architecture is a granular mapping of function to form whereas a configuration is an ensemble of existing elements of form. Since this work focuses on system architecture generation and evaluation at the conceptual design stage, we will use the term “concept” and “architecture” interchangeably in the remainder of this chapter.

2.1 Importance of Considering Broad Sets of Architectures

According to the NASA Systems Engineering Handbook, perhaps “*the most creative*” and one of the most important activities in systems engineering is the generation of alternative concepts [43]. Literature suggests that 70%-80% of life-cycle cost is set by the selected concept [44, 45], though there is some debate regarding the precise fraction [46]. Once fixed, a concept or high level system architecture constrains the design space of the optimization problem to be solved during preliminary/detailed design. It therefore bounds attainable performance with available subsystem technology.

This can be seen analytically from the “*Lower Bounding Principle of Optimization*” [37]. Consider an optimization problem in which the domain of the design variables is in some set $D \in \mathbb{R}^n$. The lower bounding principle of optimization states that if there exists a larger set $E \in \mathbb{R}^m$ such that $D \subset E$ and if x_E, x_D are optimal solutions of $\min_{x \in E} f(x), \min_{x \in D} f(x)$ respectively then:

$$\min\{f(x): x \in E\} \leq \min\{f(x): x \in D\} \quad (2.1)$$

In other words, if we optimize an objective function over a broader design space $E \in \mathbb{R}^m$ such that $D \subset E$, solutions will exist which are as good or better than those in the more constrained problem. Therefore if we examine many concepts, rather than parametrically sizing a single concept, we will find designs which at the very least perform as well as current designs in addition to possibly finding better solutions.

The “*Lower Bounding Principle of Optimization*” has perhaps found its practical embodiment in the Set-Based-Design literature. Set-based design emphasizes considering broad

sets of concepts in parallel and delaying convergence to a single point design late into the design process [47]. Considering large “sets” of designs reduces the likelihood that changes in requirements or availability of new information will make the sole selected concept infeasible or difficult to realize causing rework. It also builds knowledge about the design space [47]. Another key point in set-based design is that it allows stakeholders to filter concepts based on additional considerations (e.g. objectives or constraints) that were not explicitly considered in the analysis that generated the feasible set. For example if when generating a feasible set of concepts, fuel consumption is considered but maintainability, reliability, complexity or some other objective is not considered explicitly, presenting a broad set of concepts can allow stakeholders to eliminate infeasible concepts using their subject matter expertise. Set-based design has been applied successfully in automotive [47] (Toyota) as well as naval [48],[49] applications. However, set-based design principles have not gained as much attention in aerospace applications [50] until relatively recently [51, 52].

Due to the afore mentioned benefits of considering and presenting broad sets of architectures throughout the design process both for theoretical (“*Lower Bounding Principle of Optimization*”) and practical (Set-Based Design) reasons, the approach proposed in this thesis to architecture exploration considers sets of feasible architectures rather than focusing on a single concept.

2.2 Different Approaches to Architecture Generation

Concept generation in the aerospace industry has historically been accomplished via subject matter expert (SME) suggestion, evaluation and down-selection of alternative concepts. These approaches rely on human intuition and therefore, are in principle, very open to the inception of new architectures [12]. In practice, however, unstructured intuitive approaches

tend to generate a relatively modest number of concepts and can bias results towards legacy designs in mature industries [53] [11].

There is a formidable body of work examining different approaches to generating concepts. In [54] broadly categorizes different ways of generating concepts into three categories as shown in figure 2.1: Intuitive, Mixed and Logical. As the name suggests, the intuitive category relies on lightly structured techniques to aid intuition (e.g. brainstorming or analogies/metaphors). On the other side of the spectrum is the logical category of ideation techniques that encompasses highly structured methods, which, typically rely on storage and reuse of knowledge about how design problems have been solved in the past. Four different metrics for measuring the effectiveness of each of the techniques described in [54] are proposed in [55]. These metrics are:

- Quantity of Concepts: the number of concepts that a given ideation technique generates.
- Quality of Concepts: fraction of concepts which meet requirements.
- Variety of Concepts: measures how similar concepts are to one another.
- Novelty of Concepts: measures how similar concepts are to existing concepts.

Intuitive techniques generate a wide variety of novel concepts but tend to suffer from low quantity and quality [53], [11]. This is due to the fact that unaided human intuition can only consider and evaluate a relatively small set of concepts at one time. The effectiveness of intuitive techniques depends on the design team's ability to simultaneously have a deep grasp of the underlying a physical system, gained through experience, while at the same time not letting their creativity be hostage to that experience.

Logical techniques are more structured and often formulate concept generation problems

via a sequence of design decisions [56]. Perhaps the simplest of these is the morphological matrix in which all combinations of design decisions are allowed [53]. Related computational approaches constrain the design space using architectural rules elicited from subject matter expert (SME) interviews [57]. A common thread in many logical techniques is that they typically rely on subject matter expert experience, which has a tendency to be biased towards legacy concepts. For this reason logical techniques tend to suffer from low variety and novelty of concepts [11].

The set of possible concepts grows exponentially with the number of design decisions. For most practical problems, unless the design space is tightly constrained through use of historical data or expert interviews, or a linear relaxation of the problem is solved, logical techniques can generate too many concepts to examine within a reasonable amount of time. One way of reducing the size of the design space is to conduct analyses at multiple levels abstraction. At higher levels of abstraction, one has a smaller numbers of architectural decision variables which describe system level design. Once a concept at a higher level of abstraction has been selected, it constrains the design problem at a lower level of abstraction exponentially reducing the number of alternatives. One key consideration, however, is ensuring that simplified analyses don't eliminate potentially promising concepts (false negatives) while not generating too many infeasible concepts (false positives) [56].

The research presented here investigates an approach to generating system architectures from libraries of components and conducting complexity-performance tradespace exploration. As such it has its roots in a number of different domains including multidisciplinary design optimization, design automation, constraint satisfaction and automated configuration problem solving. We therefore examine literature from all of these fields to outline the current state of the art. We start our review with multidisciplinary design automation.

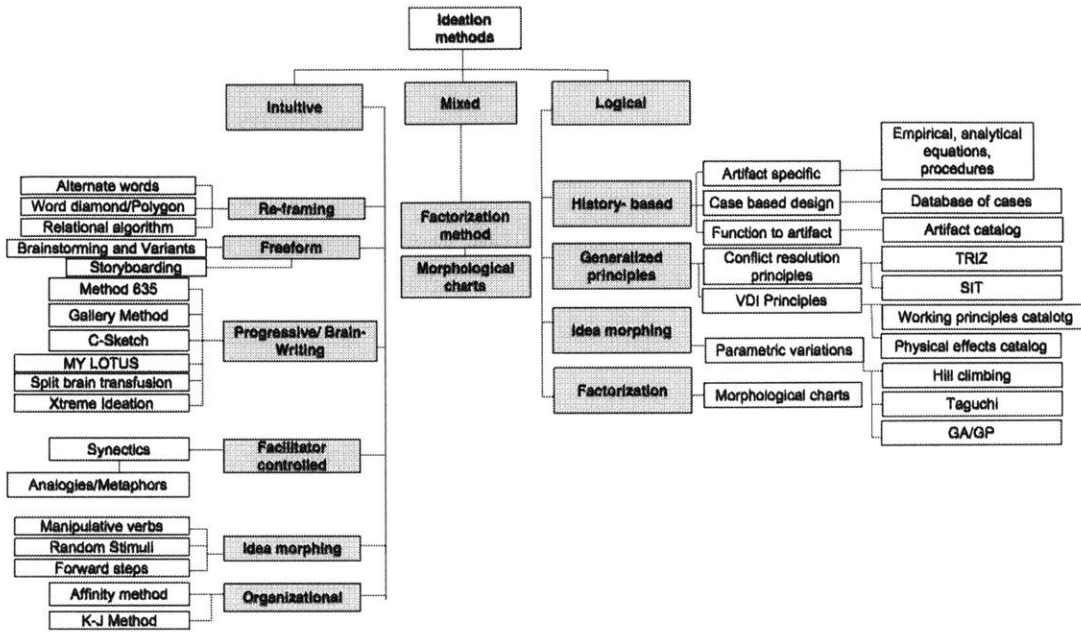


Figure 2.1: Concept generation techniques (Ideation Techniques). Taken from [11]

During the last 40 years the field of design optimization has matured and expanded from its beginnings in structural optimization to the wide range of gradient based and heuristic multi-disciplinary techniques that exist today [58]. Among the developed techniques are methods that handle uncertainty and optimize for expected performance [58], biologically inspired methods like Particle Swarm Optimization and Genetic Algorithms as well as gradient-based methods. A white paper by a panel of experts in the field provides a review of existing techniques and identifies areas where future work is necessary [58]. Two major areas of future research in MDO are outlined. The first is broadly described as an extension of current techniques. This includes further work in the use of “Massively Concurrent Computing” (MCC) in the MDO field, standardization of the way in which MDO problems are presented to highly parallel computer architectures as well as research in model decomposition using metrics for “coupling strength” and “coupling breadth” between modules. The second direction identified for future work in the MDO field is described as the introduction of new

techniques which include redefinition of the design space, inclusion of cognitive sciences in MDO, and changes in system configuration after deployment. Recent work in the aerospace context has been directed towards the application of multidisciplinary optimization to new configurations of aircraft [28], [10]. This, and all architectural exploration in the context of MDO may be described as falling in the redefinition or expansion of design space category.

Research has also been done into expanding the objectives traditionally considered during design optimization. One such example is introducing uncertainty into MDO and optimizing for system availability over its lifespan [59, 60]. One of the primary findings of this research was that system availability and expected performance are sensitive to static design variables in addition to the traditionally examined mean times between failure of individual components and subsystems [59]. When optimizing for expected performance over long deployment times, the set of “optimal designs” which define the Pareto frontier no longer correspond to what would be optimal at deployment. For example when sizing control surfaces of ultra-long endurance aircraft, one can optimize not just for maximum endurance or range in the nominal state but also for performance in degraded states where some of the control surfaces are malfunctioning. This is related to the discussion around complexity and performance in Chapter 1. Considering a traditional performance objective like weight or fuel consumption in isolation without considering a broader set of objectives can lead to unrealizable designs or designs which do not meet stakeholder needs in the field. This motivates expansion of the design space, expansion of the objectives that are considered as well as consideration of broad sets of concepts.

Generating feasible configurations of components that meet requirements can be framed as a constraint satisfaction problem. Constraint satisfaction problems are ones in which we seek to find design variables that meet constraints. Much work has been done on making constraint satisfaction problem solvers more efficient [61, 62, 63, 64, 65, 66, 67, 68]. Due

to the maturity of the constraint satisfaction problem solving field, the work presented here will focus on a way of writing constraints rather than making constraint satisfaction solvers more efficient.

Research has been done in the exploration of different system architectures during conceptual design [69, 28, 57, 51, 52]. Much of this work formulates system architecting problems as decision trees that determine the presence or absence of components and the connections between them. One such approach takes as an input a set of interconnected “design containers” from the designer [69]. These “design containers” are automatically populated with instances of components from a user defined library subject to constraints. Because the way in which these design containers are connected to one another is defined by the designer, results in practice often generate existing architectures with different instances of components. We can describe this approach in terms of the metrics for assessing effectiveness of ideation techniques discussed earlier in this chapter [55]. While the quantity and quality of configurations may be high, their novelty and variety tends to be limited.

There has also been work that incorporates expert experience into interaction rules between components for satellite systems [57]. While such approaches can be powerful, for problems in which system architecture changes significantly, commonly used heuristics may no longer be useful. For example in civil aircraft design, one common performance metric is carbon dioxide emissions which are related to total fuel consumption which for fossil fuel powered aircraft is directly related to total energy consumption. For future aircraft, which may be electric or hybrid electric, minimizing energy consumption may be less important than minimizing emissions, meaning that performance metrics and associated engineering “rules of thumb” will have to be amended.

Previous work has used multi-domain mapping matrices (MDM) to map functions to form to

help examine automotive hybrid-electric power train architectures [5]. Related research has also been done that focuses on function and form decomposition and the use of Object Process Methodology (OPM) to frame systems architecting problems [70, 71]. Object-Process Methodology is a system modeling language that treats systems as collections of objects that can be transformed by processes [72]. It enables a precise delineation between what is an element of form and what is a function that may be achieved by that element of form. Do's work presents a "Systems Architecting Matrix" [71] which leverages OPM and MDM to frame and solve system architecting problems. In MDM as well as OPM approaches, however interactions between components are typically classified by type and not expressed via a mathematical construct that describes their nature in more detail. The work presented in this thesis will propose one such mathematical construct.

A number of other approaches to generating configurations and architectures have treated components as sinks and sources of different types of flows [73, 51]. These approaches have abstracted components into resource sinks and sources, simplifying rule sets and linking them directly to component behavior [73]. In "resource-based" approaches components are connected to one another by flows of resources and an architecture or configuration is feasible only when all required flows between components are present. When component sizing is not defined a-priori, architectures are generated. When sized components configurations are generated [1]. We can describe these approaches in terms of the metrics for assessing effectiveness of ideation techniques discussed in Chapter 1 [55]. For resource-based approaches, the quality, variety and novelty of configurations can be very high. The quantity, however, is often too high to exhaustively search the design space. As mentioned in Chapter 1, one way of reducing the size of the design space is to conduct analyses at multiple levels of abstraction or fidelity. At higher levels of abstraction, one has smaller numbers of decision variables which describe system level design. Once a concept at a higher level of abstraction

has been selected, it constrains the design problem at a lower level of abstraction exponentially reducing the number of alternatives. One key consideration however is ensuring that simplified analyses don't eliminate potentially promising concepts while not generating too many infeasible concepts. While this has been explored theoretically, not much work has been done in achieving it in a domain independent way [56]. The work presented here therefore develops an architecture generation framework, using resource-based constraints at more than one level of abstraction.

Other resource-based approaches have dealt with the combinatorial size of system architecting problems by linearizing component behavior [74, 75, 76], or using domain/problem specific bounding functions [56] that insure simpler models in multi-fidelity analyses that don't eliminate promising designs (no false negatives). Formulating bounding functions that remove infeasible designs from consideration (few false positives or good discriminatory power) which also don't suggest infeasible designs (no false negatives) is non-trivial. Using simplified models often can lead to promising designs being overlooked and infeasible designs being suggested [74, 75]. Not much work has been done on linking multiple levels of abstraction. Control over false positives and false negatives during abstraction has not received much attention either in the broader engineering context. While there has been significant work examining configuration sensitivity to changes in shape in the structural domain [77] not much work has been done generalizing this to heterogeneous nonlinear systems in a way that explicitly links resource flows to detailed nonlinear system behavior.

We now summarize the relevant gap in the literature for automated architecture generation and evaluation using the table below. The table divides the relevant literature into 5 categories. The first category denotes whether an approach generates a feasible set of concepts or individual "optimal solutions". The second category indicates whether complexity and performance tradeoffs are explicitly considered. The third category denotes whether an ap-

proach is resource-based and therefore simplifies the rule set, reducing bias towards legacy architectures [73] as well as indicating whether resource flows are explicitly linked to non-linear heterogeneous system dynamics. The fourth category denotes whether the approach provides a mathematical mechanism to control the number of false negatives in exchange for reducing the size of the combinatorial design space. In the case of the “partially addressed” evaluations for this column, false negatives can be deliberately generated with additional constraints, but no specific approach is proposed. The fifth category indicates whether domain independent abstraction layers are used to reduce the size of the design space. We see that none of the approaches found in the literature address all of these direction from table 2.1.

	Feasible set of concepts, architectures, configurations rather than individual solutions	Performance and Complexity Considered	Rules on flows between nonlinear components (no prescribed direction)	Control over False Negatives Based on Matching of Resource Flows	2 Levels of Abstraction Linked in Domain Independent Way
(Murray B. et al. 2011) (Zeidner L. 2010a) (Zeidner L. 2010b)	Yes	Yes (limited application)	Partially Addressed	Partially Addressed	Partially Addressed
(Selva D. 2012) (Neema S. et al. 2003)	Yes	No	Partially Addressed	Partially Addressed	No
(Simmons 2008) (Speller 2010)	Yes	No	Partially Addressed	Partially Addressed	No
(Piacenza et al. 2015) (Ishimatsu 2013) (Ho 2015) (Peralta et al. 2003)	No	No	Yes (Linear)	Partially Addressed	Partially Addressed
(Heinrich et al. 1996)	Partially Addressed (configurations)	No	Yes	Partially Addressed	Yes
(Miller et al. 2015)	No	No	No	Partially Addressed	Partially Addressed
Benefits	Tradeoffs (stakeholder experience).	Explicit Tradeoffs.	Less bias. Direct physics.	Manages combinatorial space.	Manages combinatorial space.

Table 2.1: Gap in literature

At this stage it is useful to map the gap in the literature of automated architecture generation to the high level objective of this research which is to have conduct performance-complexity tradespace exploration of concepts which are generated with high quantity, quality, novelty and variety (metrics of ideation effectiveness) [55]. Resource-based methods to generating architectures treat components as providers and suppliers of resources. These approaches tend to have simpler rule sets that are easier to manage and reduce bias towards legacy designs [73]. This has the potential to improve the quality, novelty and variety of concepts generated. Approaches which take into account the behavior of nonlinear heterogeneous systems effect the quality of concepts generated. One of the major issues with many computational techniques to generating architectures is the large number of different combinations of components that can be computationally intractable. Carrying out analysis at multiple layers of abstraction allows a broader number of concepts to be considered. If abstraction layers are connected, such that decisions made at higher levels of abstraction don't generate infeasible configurations, quality of concepts is also increased [56]. To constrain the design space it is useful to deliberately be able to eliminate some promising designs (generate false negatives). Addressing this aspect of the gap in the literature would reduce the quantity of concepts in exchange for increasing their novelty and variety.

This work builds on previous work in resource-based architecture/evaluation generation which expresses flows at interfaces as convex hulls ("smallest convex set that contains operating points" [37]) which are explicitly linked to detailed simulation. Two layers of abstraction are linked using convex hull flow constraints to reduce the size of the design space. An approach that allows the designer to trade the number of false negatives and the size of the design space is presented. The approach is applied to an aircraft engine case study. Architecture generation is combined with automated performance optimization and evaluation and complexity quantification. A mobile electronic device case study is also presented to

	Feasible set of concepts, architectures, configurations rather than individual solutions	Performance and Complexity Considered	Rules on flows between nonlinear components (no prescribed direction)	Control over False Negatives Based on Matching of Resource Flows	2 Levels of Abstraction Linked in Domain Independent Way
(Murray B. et al. 2011) (Zeidner L. 2010a) (Zeidner L. 2010b)	Yes	Yes (limited application)	Partially Addressed	Partially Addressed	Partially Addressed
(Selva D. 2012) (Neema S. et al. 2003)	Yes	No	Partially Addressed	Partially Addressed	No
(Simmons 2008) (Speller 2010)	Yes	No	Partially Addressed	Partially Addressed	No
(Piacenza et al. 2015) (Ishimatsu 2013) (Ho 2015) (Peralta et al. 2003)	No	No	Yes (Linear)	Partially Addressed	Partially Addressed
(Heinrich et al. 1996)	Partially Addressed (configurations)	No	Yes	Partially Addressed	Yes
(Miller et al. 2015)	No	No	No	Partially Addressed	Partially Addressed
(Shougarian 2016)	Yes	Yes	Yes (Convex Hull Formulation)	Yes (Convex Hull Intersection Formulation)	Yes

Table 2.2: Contribution to gap in literature.

illustrate the generality of the approach. The specific areas of contributions are shown in table 2.2.

The primary methodological contribution of this work is therefore an architecture generation and evaluation framework which combines:

- Generation and optimization of sets of architectures rather than individual architectures taking into account both complexity and performance.
- Rules for generating architectures which rely on flows between components which are described via convex hulls (this will be described in Chapter 4) and subject matter expert experience. This builds on previous work on generating configurations [73]. The

two primary extensions here are that firstly, rather than combining pre-sized components, the approach applies to architectures (components are not sized a-priori) and flows are expressed via convex hulls around detailed simulation of sets of components.

- Control over false negatives using normalized intersection volume of convex hulls that express resource flows between components.
- More than one layer of abstraction.

The domain specific contributions are confined to the aircraft engine and reconfigurable mobile device case studies and are based on the insights gained from their respective performance-complexity tradespaces.

Chapter 3

Thesis Statement

To meet ICAO/ATAG fuel efficiency goals, aircraft and engine architecture are likely to change. We established that system architecture impacts both performance and complexity which is related to development effort. This motivates exploring the tradespace performance of architectures during conceptual design. Referring back to the metrics for ideation effectiveness discussed in Chapter 2 [55], the problem we address is that quantity, quality, variety and novelty of architectures generated during conceptual design are limited by:

1. Subject matter expert bias towards legacy architectures.
2. Limited number of alternative concepts generated through purely intuitive concept generation techniques.
3. Size of combinatorial space for computational techniques.
4. Elimination of potentially feasible designs and suggestion of infeasible designs when using simplified models.

Based on these observations, we formulate the following research questions.

3.1 Research Questions

1. *How can we automatically generate and evaluate air-breathing propulsion and other cyber-physical system concepts in the performance-complexity design space?*
2. *What is the performance-complexity tradespace of concepts in air-breathing propulsion and other cyber-physical systems?*

3.2 Approach

To answer the research questions, this work contributes a convex hull-based approach to automated generation and evaluation of new architectures by:

- Generalizing resource flows into linear constraints around operating points at component interfaces (convex hulls).
- Using convex hulls to link abstraction layers in a domain independent way.
- Allowing the designer to parametrically control the number of false negatives to reduce the size of the design space.

Chapter 4

Approach

We now discuss the approach used to answer the research questions outlined in Chapter 3. We first introduce a system architecture generator that conducts search for feasible architectures (components can parametrically vary) or configurations (components parametrically fixed a-priori) given a component model library.

4.1 Magellan Architecture Explorer

As mentioned in the literature review chapter “resource-based” approaches to architecture generation treat components as resource sinks and sources [73]. An architecture is feasible when all required flows of resources between components within the system and across the system boundary are present. Since resource-based approaches rely heavily on the underlying physics of material, energy and information exchange between components, they are less prone to bias towards legacy architectures due to subject matter expert opinion [73]. The architecture generator created presented here relies on a resource-based approach to architecture generation though it does also include some architectural rules based on subject

matter expert experience. Resource-based approaches also have limitations. The primary limitation is that they typically rely on an exchange of one-dimensional lumped parameter flows in networks where components have linear or nonlinear behavior. This is a simplification of the underlying physics and its implications must be considered in the context of the problem being solved. We will discuss limitations in more detail later in this chapter.

For brevity from now on we will refer to the architecture generator and evaluator as Magellan in honor of the explorer Ferdinand Magellan who was the first explorer to circumnavigate the globe. Magellan searches for feasible ways of selecting and connecting components together from predefined libraries such that they form an architecture or configuration which does not violate constraints. The primary contribution to the literature on architecture generation is that resource flows are directly based on nonlinear model behavior through the use of convex hulls around operating points (smallest convex set that contains all operating points). This will be discussed in more detail later in this chapter.

Libraries of components can exist at multiple levels of abstraction or granularity. At higher levels of abstraction multiple components or subsystems are grouped into subsystem types. More generally, groups of interacting components at lower levels of abstraction can be treated as a single component at higher levels of abstraction. The primary reason for using abstraction is rooted in scalability. We can think of the selection of a particular architecture in terms of making a set of design decisions (for example the presence or absence of a gearbox). The design space grows exponentially with the number of design decisions that need to be made to define an architecture. Grouping lower level components together into subsystems and making design decisions at the more abstract subsystem level reduces the number of design decisions that need to be made to define an architecture. Once an architecture at a higher level of abstraction has been selected, it constraints the design problem at a lower level of abstraction exponentially reducing the number of alternatives. This approach can

eliminate potentially feasible concepts at the more detailed level of analysis since subsystem boundaries have now been defined limiting interaction of components across those boundaries. It does however, make the design space smaller by reducing the number of design variables. The decision of how to abstract groups of interacting components into component types is non-trivial. A method for grouping different formulae using linear constraints on their outputs [78]. We adopt a similar approach for nonlinear systems here.

The primary function of Magellan is to generate sets architectures from libraries of components and compute their complexity and performance in order to allow explicit tradeoffs to be made between these two objectives. It therefore may be categorized as a decision support tool. Reference [79] outlines a list of elements that decision support tools must have that are summarized below:

- **Representation:** A way of describing the design space mathematically. This is the vector of design variables and the set of values each element can take. In systems architecting problems one commonly used way of formulating design variables is a binary vector of design decisions where “1” indicates the presence of a component or connection and “0” indicates its absence. This approach is referred to as a binary decision tree and is used here.
- **Search Algorithm:** An algorithm that searches for assignments of values to elements of the design vector that do not violate any constraints.
- **Structural Reasoning:** A way of writing constraints that can be interpreted by the algorithm which searches for feasible architectures.
- **Evaluation:** A way of computing objective functions to compare architectures.
- **Visualization:** A way of visualizing the results.

Each of the requirements outlined in [79] are addressed in the development of the architecture generation approach. In addition, [57] underlines reusability and the reduction of problem computational complexity as desirable attributes of architecture of decision support tools which will be addressed here.

4.2 Representation and Visualization

We introduce the Design Space Structure Matrix (DS2M) for the purposes of representation and visualization of the design space. The DS2M is an extension of the well known Design Structure Matrix (DSM) that is used to express connections between components via ones and zeros in an adjacency matrix [80]. Like the Design Structure Matrix, the rows and columns of the Design Space Structure Matrix represent components or functions. A non-zero entry in any location indicates a flow from the column component into the row component. In contrast to the Design Structure Matrix, which describes the components and connections within a single architecture, the Design Space Structure Matrix contains all possible architectures given a library of components. Every feasible architecture can be expressed as a subset of the components and connections within a single DS2M given a component library and bounds on the maximum number of components.

The DS2M builds on previous work using Multi-Domain Mapping Matrices (MDM) which allows a DS2M like matrix can be generated from matrices mapping components/functions to input flows and output flows [5, 81, 51]. There are however, a few important differences, both when compared to MDM approaches and compared to the design structure matrix itself.

- **Flow Properties Captured As Well as Flow Type:** Design Structure Matrices and Multi-Domain Mapping matrices typically only define flow type and do not dis-

tinguish between multiple interfaces which have the same flow type. For example, if a single component has high and low temperature gas interfaces, a DSM does not contain the information necessary to distinguish between them unless a new flow type is defined. Every location in the DS2M contains a data structure rather than a binary variable. This data structure encodes pairs of interfaces that are connected by instance, type and a set of linear constraints representing the flow envelopes at each interface. As will be described in the Structural Reasoning section, a single component may have n different gas interfaces each with an associated flow envelope. Flow envelopes capture the detailed physics behind each of the connections between components, in addition to showing their existence/absence and type.

- **No Flow Direction Assumption:** Previous work has generally assumed user defined flow direction while the baseline DS2M represents all connections given a component library of interfaces of the same type with no assumptions on flow direction. The direction is determined later by examining flow envelopes at interfaces.
- **Multiple Instances of Functionally Identical Components:** The DS2M explicitly shows the minimum and maximum number of instances of functionally duplicate components. Because of this fact, every feasible architecture will be strictly a subset of the components and connections in the DS2M. This is not the case with previous work [81].
- **Elimination of Infeasible Connections:** The final difference is that, pairwise infeasible connections (this includes removal of possible flows in one or both directions between components) are eliminated using resource flow envelopes at component interfaces represented by convex hulls.

The DS2M can also be converted into a binary matrix for visualization. While relatively small DS2M-s and DSM-s (less than 100 rows or columns) are human readable, they are most

useful when used in conjunction with algorithms which analyze them to help the designer improve understanding of the system. Clustering algorithms, for example, find subsets of components which have many connections between one-another compared to connections to the remainder of the system. Clustering algorithms assist with subsystem definition [27]. Figure 4.1 describes the convention for different types of flows and their direction. An example of a generic DS2M for a library of motors, gearboxes and propellers is given in figure 4.2. In figure 4.2 only pairwise feasible connections are allowed to exist.




Flow Type		Interface Flow Direction	
1	Mechanical Power (Translational, Rotational etc.)		Input/Output
2	Mass Flow (Gas, Oil, Fuel etc.)		Input
3	Energy Flow (Electric, Heat etc.)		Output
4	Information Flow (Control, Sensing)		
	Compatibility Scored Connection Complete (Black), None (White)		

Figure 4.1: Convention for flow type and interface flow direction. Four primary flow types are used [12]. Each type has associated with it a color, number and location within a cell in the DS2M. Three different interface flow directions are represented. These include inputs, outputs and input/output interfaces. As the name suggests input/output interfaces have no inherent direction associated with them.

The DS2M defines set of components and connections that can exist. Prior to searching for feasible architectures, we must therefore generate the DS2M. DS2M generation occurs in two phases. These are summarized below and provide a detailed discussion later in this chapter.

- *Phase One:* In the first phase all interfaces of the same type (e.g. shaft power) are connected bi-directionally (e.g. Component A to B and Component B to Component A flows exist) to create an initial DS2M.

A simple example of phase one DS2M generation is shown in figure 4.2. The example consists of a Design Space Structure Matrix generated from a notional library of

propellers and motors which exchange energy. All interfaces of the same type are connected to one another.

- *Phase Two*: In the second phase connections are removed from the initial DS2M if the set of demanded resources does not intersect with the set of supplied resources. This is based on previous work done by [51] and [73]. The primary difference is the generalization of flows at component interfaces into resource envelopes expressed as convex hulls. The direction of flow is determined implicitly using the sign of the “quantity” variables (e.g. mass flow) of the convex hulls.

A simple example of phase 2 of DS2M generation is shown in figure 4.3. The example consists of a Design Space Structure Matrix generated from a notional library of propellers and motors which exchange energy. All interface pairs for which for example angular velocity ranges are incompatible are eliminated. We will discuss how exactly these constraints are enforced in the next section. The thrust requirement is represented by a component in a similar manner to [73]. Since we are now examining only pairwise compatibility, the thrust requirement component is only described via “quality” variable convex hull (e.g. Mach number and Altitude for example). This ensures that thrust can be provided at the correct flight condition.

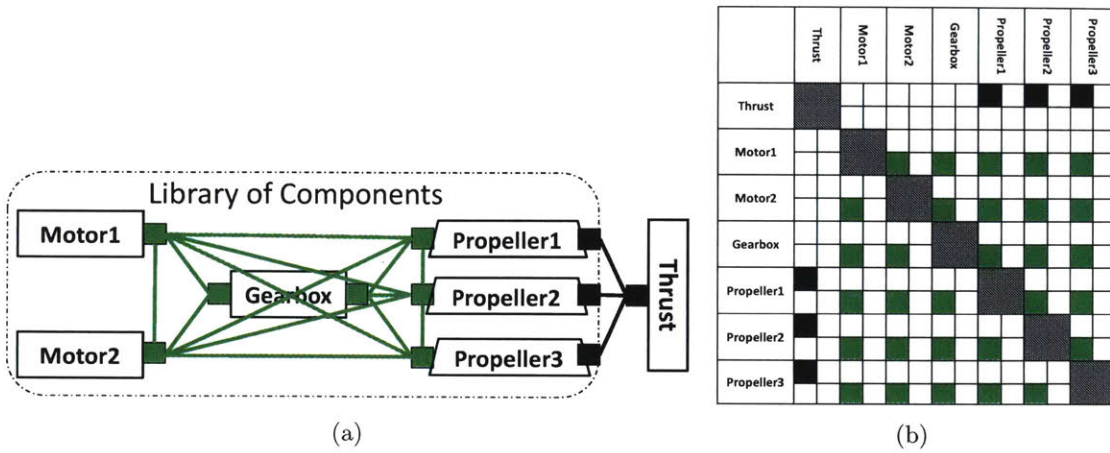


Figure 4.2: Automatically Generated Design Space Structure Matrix (a) Block Diagram Representation (b) Matrix Representation

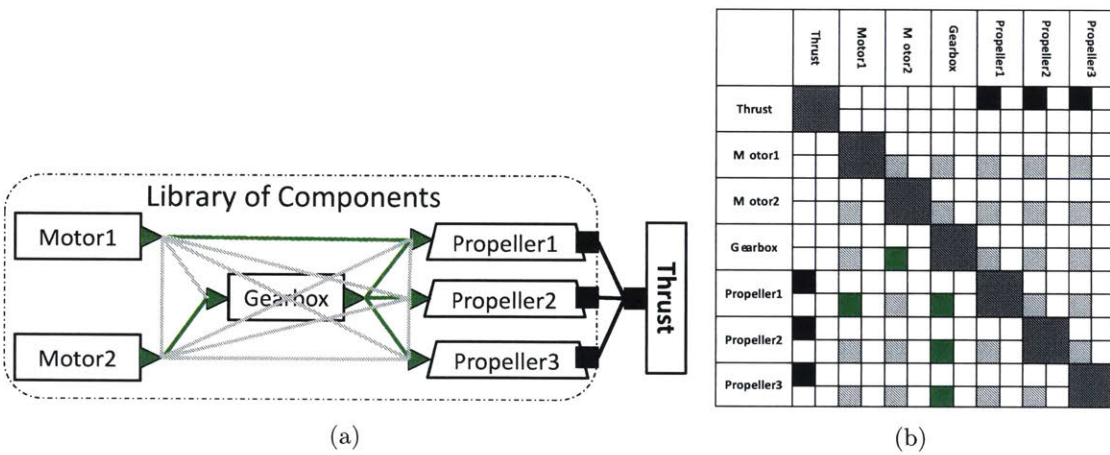


Figure 4.3: Resource Constrained Design Space Structure Matrix (a) Block Diagram Representation (b) Matrix Representation

4.3 Structural Reasoning

This section discusses an approach to assessing the feasibility of architectures that relies on constraints on resources produced and consumed by individual components.

4.3.1 Pairwise Feasibility (Convex Hulls)

We propose a resource envelope based approach to assessing whether individual connections in the DS2M are possible. Resource-based approaches to architecture generation treat components as producers and consumers of resources [73] and only connections for which resources provided can equal resources consumed are allowed to exist. The primary contribution to resource-based approaches in this work is the generalization of resource flows to convex hulls (smallest convex set around points) which encompass all possible resource flows.

A useful analogy to help illustrate this concept is aerial refueling. Figure 4.4 notionally depicts the operating envelopes of a KC10 tanker and F/A-18. A necessary condition for the KC10 to be able to refuel the F/A-18 is that that the two aircraft are able to fly at the same Mach number and same altitude. Put differently, their operating envelopes need to intersect.

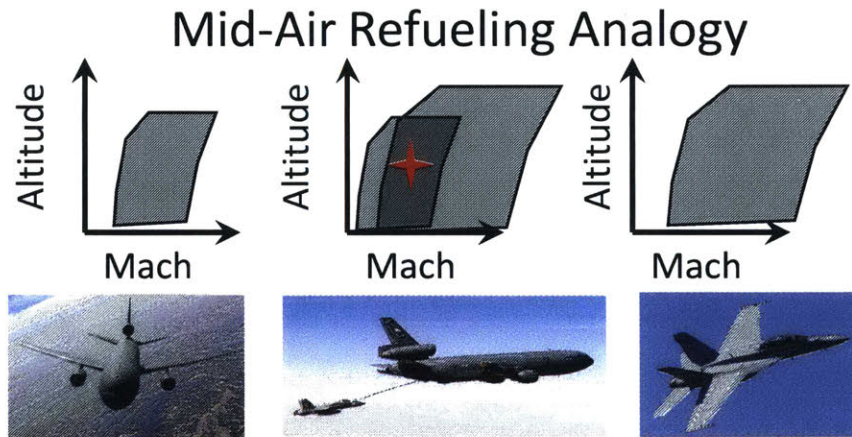


Figure 4.4: Performance envelopes of KC10 and F18 must intersect for them to be able to exchange fuel in flight. Images from left to right taken from [13], [14] and [15] respectively.

The same concept can be used to check if a turbine can power a compressor. In figure 4.5

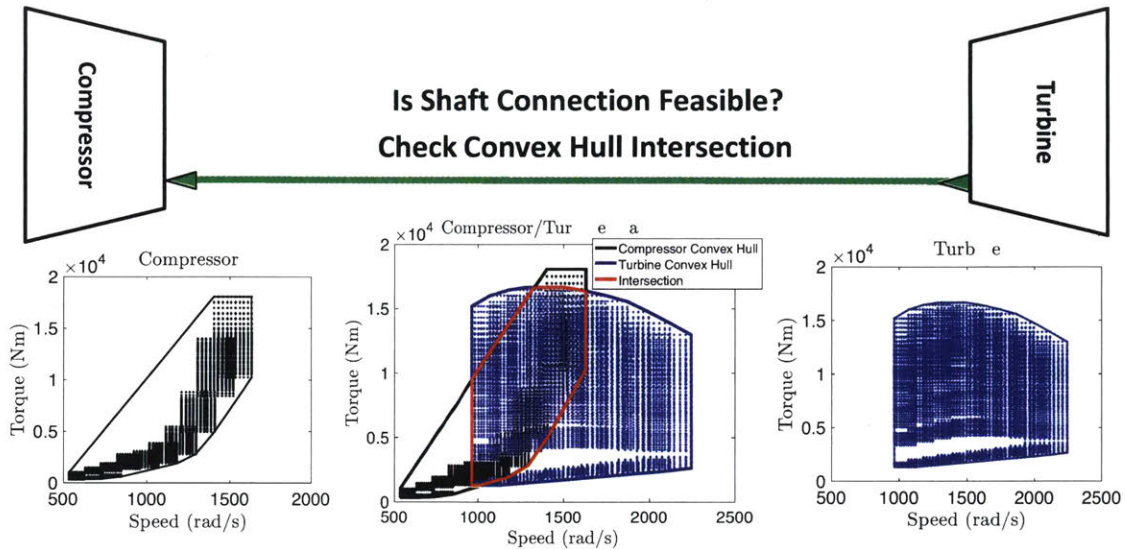


Figure 4.5: Resource envelope intersection constraint. To reduce the number of constraints we approximate the true intersection of the convex hulls representing shaft power with one that contains fewer linear constraints. For this reason the depicted intersection (highlighted in red) does not exactly match the true intersection of the convex hulls.

we consider a shaft power connection between the two components. The resource envelopes in this case express the set of possible torques and angular velocities that can be produced by the turbine and consumed by the compressor. Since the intersection of the resource envelopes is non-zero (shown in red) there exist some angular velocities and torques which match. Therefore this connection is possible. Resource envelopes can be expressed as sets of linear constraints that form convex hulls (smallest convex set which contains all operating points). We can then use these to formulate a condition for feasibility in equations (2) and (3) which state that a connection is only feasible if the normalized volume of intersection of the convex hulls describing resource flows is greater than a user defined tolerance.

$$\frac{\text{Intersection Volume}}{\text{Union Volume}} = \frac{V(\text{conv}(C) \cap V(\text{conv}(T)))}{V(\text{conv}(C) + V(\text{conv}(T))) - V(\text{conv}(C) \cap V(\text{conv}(T)))} \quad (4.1)$$

$$\text{Connection feasible iff } \frac{\text{Intersection Volume}}{\text{Union Volume}} > \textit{tolerance} \quad (4.2)$$

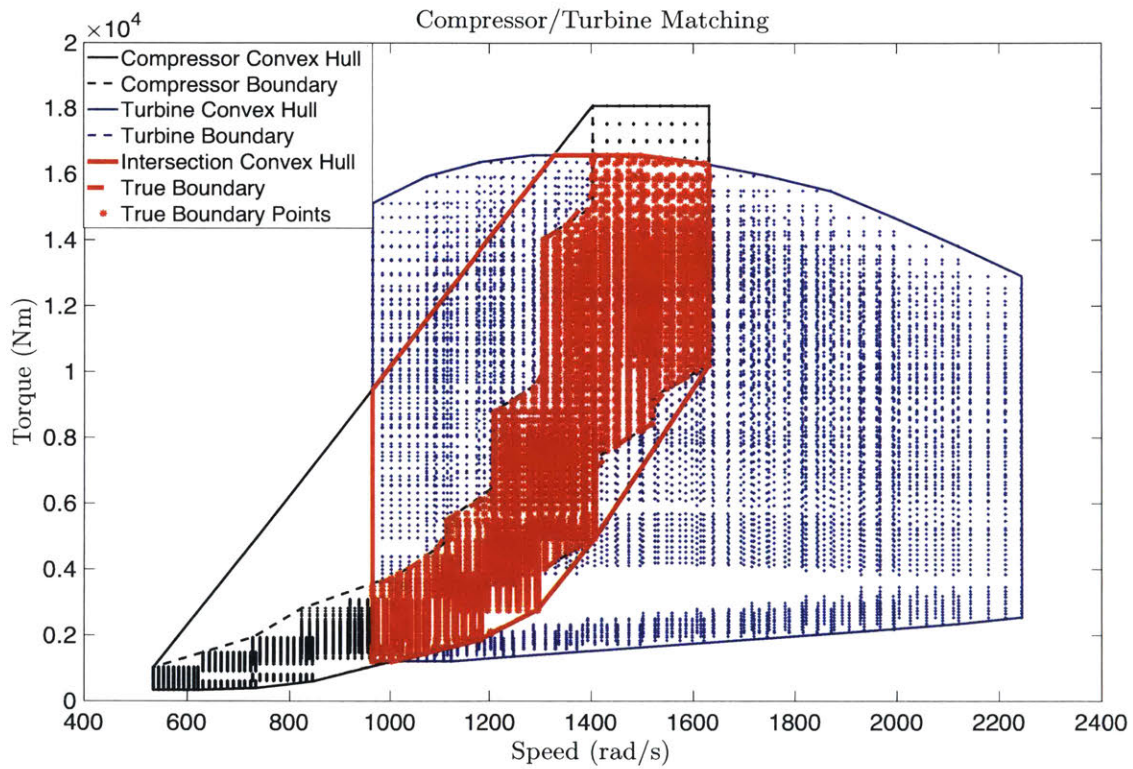


Figure 4.6: Intersection of true envelopes based on data from [16].

Upon careful inspection of the intersection of the of compressor torque and angular velocity, we notice that the actual envelope shown in red in figure 4.6 is not convex. In all cases the convex hull around operating points will be a larger set than the underlying performance envelope. This means that the convex hull around operating points will be an “optimistic

assessment” of the possible flows that a given interface can provide, generating false positives, especially when the underlying performance envelope is not convex as shown in figure 4.6. This is the primary disadvantage of the convex hull formulation.

The advantages of using the convex hull intersection approach to resource-based architecture generation may be summarized as follows:

- **Convex Hulls Expressed As Linear Constraints:** One of the primary advantages of using convex hulls around operating points to describe component interface behavior is that convex hulls can be represented via as a set of linear constraints. This simplifies the underlying mathematics of finding volumes of intersection of convex hulls since intersection regions can be computed by combining the linear constraints of different interfaces together. In addition linear constraints can be more easily integrated into existing constraint satisfaction tools and be used to formulate linear programming relaxations of architecture generation/evaluation/optimization problems [74].
- **Smaller design space:** Using the presented resource envelope approach, infeasible connections can be eliminated, reducing the size of the design space.
- **Reusability (resource envelopes defined in library):** Convex hulls which describe resource flows are associated with a component model library rather than objectives of the architecting problem. Once a library of components is defined and the convex hulls describing possible resource flows of components are known, they do not need to be updated in order to generate architectures that satisfy new requirements. This renders convex hull constraints reusable for different problem instances.
- **No false negatives (don't eliminate potentially feasible designs):** If the user chooses to set the minimum acceptable intersection volume of performance envelopes to be zero, only those connections which are always pairwise infeasible will be eliminated.

Before continuing our discussion we emphasize that flows between components can be described by two types of variables which are treated differently. The first type describes flow “quality” (temperature, pressure, voltage) and the second expresses flow “quantity” (mass flow, energy flow, torque, power) [73]. For flow variables which describe flow quantity, Kirchhoff’s current law applies (conservation of mass, energy and momentum) and feasibility will therefore depend on the network of connections in a given configuration (components defined a-priori). For the general system architecting problem in which component sizes are not defined in advance, but are determined during optimization, conservation laws are enforced in the mathematical model of the system during optimization.

The DS2M represents pairwise feasible connections. If component A has a flow to B which in turn is connected to C the compatibility between A and B is not affected by the presence of component C. For this reason only those variables are considered which describe resource “quality” like temperature, pressure, voltage are used during DS2M generation. If the minimum threshold for convex hull intersection is infinitesimally small the DS2M definition does not eliminate any potentially feasible designs (no false negatives during DS2M generation).

4.3.2 Network Feasibility Using Resource Envelopes (Convex Hulls)

In the previous section we discussed feasibility of connections in isolation. Constraints which apply to individual decision variables (the presence or absence of individual connections) are referred to in the literature as node constraints [82]. While these constraints are useful in reducing the number of possible connections prior to architecture generation, they do not address the feasibility of networks of interfaces. We now introduce resource envelope based network constraints.

Consider a similar situation to the compressor-turbine example in the previous described

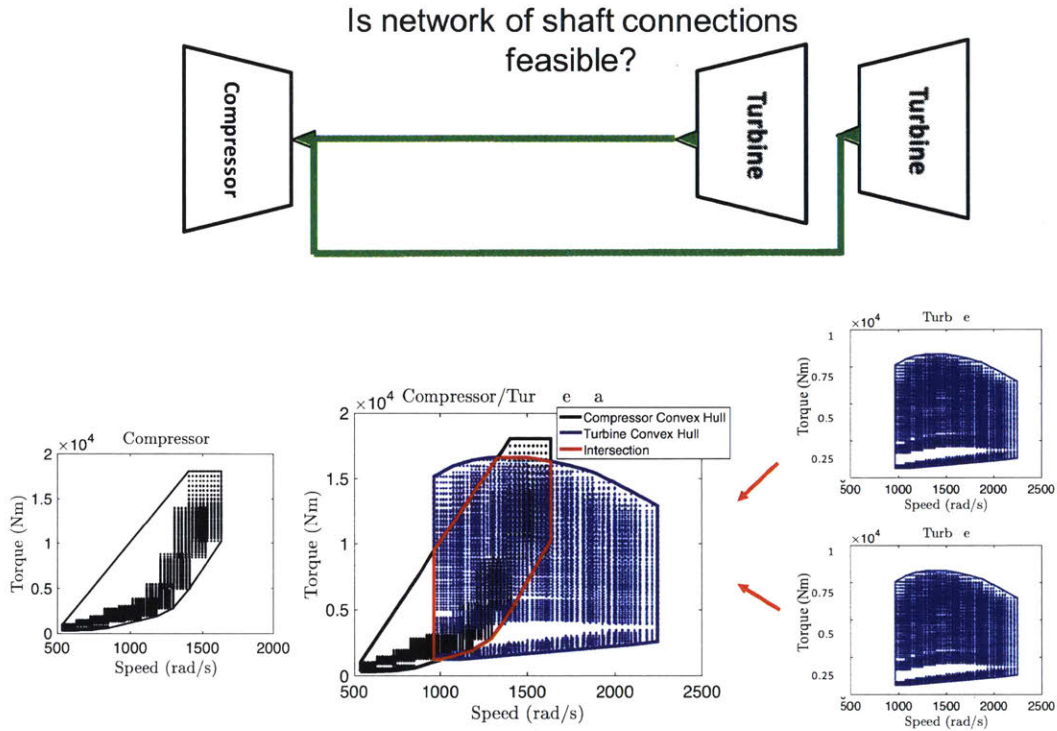


Figure 4.7: Function and form view of multiple low power turbines driving a single compressor.

earlier. In this case we have two identical low power turbines shown in figure 4.7 which have exactly half the torque of the turbine in figure 4.5. Torque is a quantity variable, meaning that the flow of torque into the compressor must equal the flow of torque out of the turbines. The flow of torque to the compressor from the two turbines is therefore the same as the flow of torque from a single turbine that has twice as much torque. This is shown graphically in figure 4.7. Resource flows describing the two turbines sum along the quantity variable axis. The result is the same as in the previous section. There are steady state operating points for which the two turbines together can power the compressor.

Taking into account only the pairwise feasibility of connections results in a combination of false positives since the problem is underconstrained and false negatives if additional,

manually defined heuristics are used to constrain the problem. We therefore extend the resource envelope constraints to networks of connections by stipulating that only those sets of connections are feasible for which there exist operating points within all resource envelopes that satisfy conservation of resource quantity. These constraints are applied after the DS2M has been defined during search for feasible configurations. Note that if components are unsized, conservation of resource quantity is enforced during optimization and not during architecture generation.

We now briefly describe algorithmically how network constraints are enforced. Consider the notional architecture in figure 4.8 which we define interfaces A_{out} , B_{in} , C_{out} , D_{in} , D_{out} , E_{in} and edges e_1 , e_2 , e_3 and e_4 .

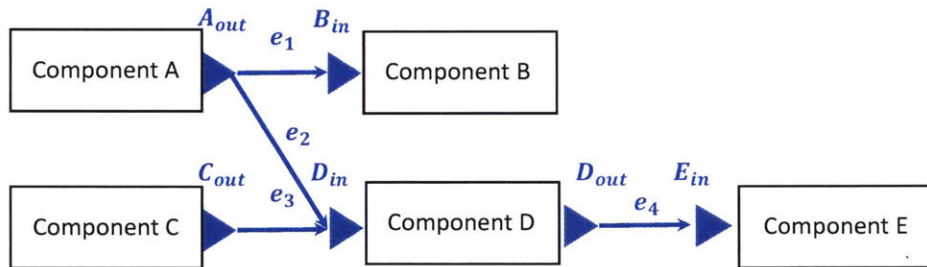


Figure 4.8: Notional architecture.

1. The feasibility of the connection between pairs of interfaces can be checked by intersecting their convex hulls. We therefore focus on the network of connections between A_{out} , B_{in} , C_{out} , D_{in} . These are depicted in figure 4.9. We have added flow variables $(e_{A_{out}}, e_{B_{in}}, e_{C_{out}}, e_{D_{in}})$ which represent the total flow through each interface. Every flow is constrained by the convex hulls associated with each interface.

A necessary condition for the feasibility of the network of connections shown in blue is that there exists a set of values of $e_1, e_2, e_3, e_{A_{out}}, e_{B_{in}}, e_{C_{out}}, e_{D_{in}}$ such that:

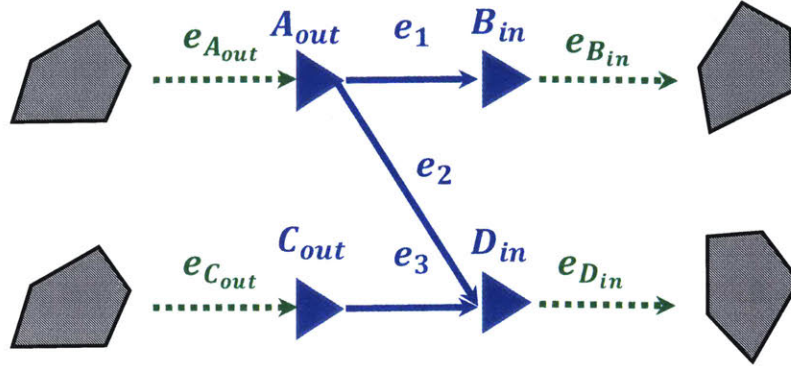


Figure 4.9: Notional architecture.

$$e_{A_{out}} \in \text{conv}(A_{out})$$

$$e_{B_{in}} \in \text{conv}(B_{in})$$

$$e_{C_{out}} \in \text{conv}(C_{out})$$

$$e_{D_{in}} \in \text{conv}(D_{in})$$

2. Since there are no flow transformations, the flows in this subgraph must have the same quality variables (similar concept to mesh voltage). We first find the convex hull describing the feasible set of quality variables for the network by intersecting all quality variable convex hulls:

$$\text{conv}_{quality}(\text{Network}) = \text{conv}_{quality}(A_{out}) \cap \text{conv}_{quality}(B_{in}) \cap \text{conv}_{quality}(C_{out}) \cap \text{conv}_{quality}(D_{in})$$

As was mentioned earlier, for problems in which components sizes are not defined

a-priori, conservation of “quantity” variables (e.g. mass, momentum and energy) is enforced during the optimization. Therefore, we only check that any connected network of interfaces can have the same “quality” variables by intersecting convex hulls defined along “quality” variable axes (e.g. temperature, pressure, Mach number or voltage). In cases where sizes of components are defined, the “quantity” variables are also defined we can perform an additional feasibility check to determine that the network is feasible according to conservation laws during architecture generation.

3. We now find the min and max values of “quantity” variables at each interface by intersecting their convex hulls with $conv_{quality}(Network)$. One example of how this can be expressed is given below:

$$A_{out_{max}} = \max(conv_{quality}(Network) \cap conv(A_{out}))_{\text{quantity variable projection}}$$

4. We now find the incidence matrix of the interfaces and use it to construct a system of linear inequalities whose intersection represents the feasible region for the entire network. In the incidence matrix -1 indicates a flow into a node and 1 indicates a flow out of a node.

Interface	e_1	e_2	e_3
A_{out}	1	1	0
B_{in}	-1	0	0
C_{out}	0	0	1
D_{in}	0	-1	-1

$$A'_{i,j} = |A_{i,j}|$$

$$A'e = b$$

$$\begin{bmatrix} -A' \\ A' \end{bmatrix} e \leq \begin{bmatrix} -b_{min} \\ b_{max} \end{bmatrix}$$

Where

$$b_{min} = \begin{bmatrix} A_{out} \\ B_{in} \\ C_{out} \\ D_{in} \end{bmatrix}_{min} \quad b_{max} = \begin{bmatrix} A_{out} \\ B_{in} \\ C_{out} \\ D_{in} \end{bmatrix}_{max}$$

A necessary condition for feasibility is that the space described by the inequalities above is finite.

Example: An example to illustrate is the notional engine configuration exploration problem with multiple thrust providers and a thrust requirement in figure 4.10. Thrust is transferred through mechanical connections given in black. We follow the DSM convention in which red denotes material flow, green denotes energy flow and blue denotes information flow. The configuration on the right is found to be infeasible since the thrust providers are unable to provide sufficient thrust through the network of mechanical connections. The primary difference between the two configurations is that the one on the left has two thrust providing components (a fan and a propeller) whereas the one on the right has only one (a fan). Note that for problems in which we are not using components of a fixed size, we ensure that an architecture provides enough thrust through sizing of propellers and fans during optimization.

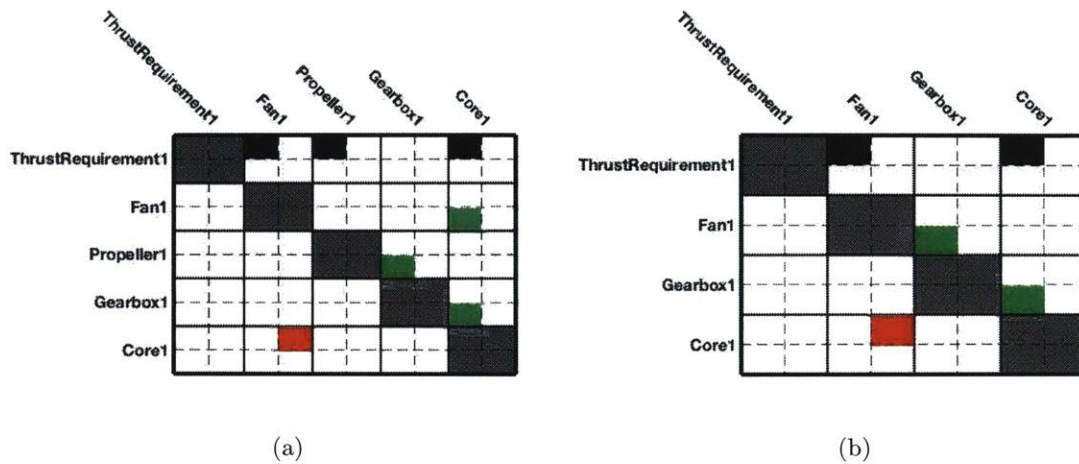


Figure 4.10: (a) Architecture example 1 (b) Architecture example 2

4.3.3 Satisfying Requirements

Ensuring design space exploration tools generate architectures that meet requirements is often done through experience based rules. The user must write component and connection compatibility rules which not only ensure that the system will generate the required types of output like thrust or power, but that it will do so subject to constraints. One way of approaching this problem for simple requirements such as providing a given torque, is to add imaginary components which enforce requirements via the same performance envelopes used to create the DS2M (see figure 4.11). These so called “imaginary components” are mathematically identical to real components. Other approaches to resource-based architecture generation have used similar concepts. The primary addition here is that resources are described via the convex hulls of component operating points [73, 1, 83]. We limit the scope of this work to consider only those requirements which can be expressed as flows of resources.

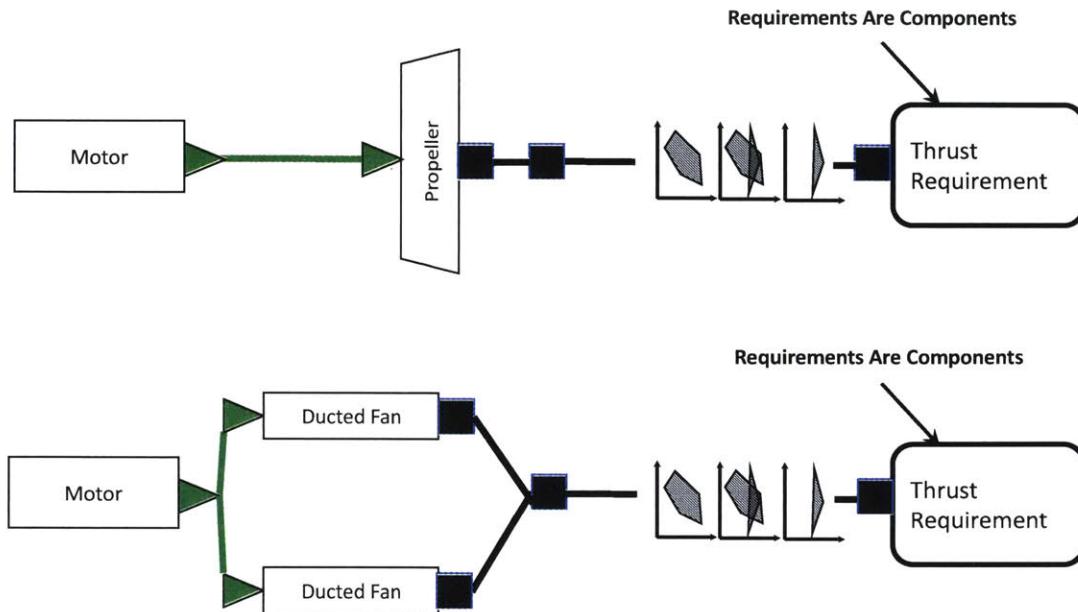


Figure 4.11: Imaginary Component Requirements Satisfaction

4.4 Multiple Layers of Abstraction

It is possible to update the resource envelopes of subsystems using simulation data from deeper abstraction layers. Resource envelopes can be mechanisms of information transfer between abstraction layers, allowing approximations made at higher layers of abstraction to be directly traceable to detailed simulation. The approach here builds on previous work done in [73]. The primary contributions are that we extend the approach in [73] from configuration problems to systems architecting problems and that resource envelopes are expressed as convex hulls.

Consider the two abstraction layers depicted in figure 4.12. At level $N+1$ we have two possible ways of generating thrust. The first uses a propeller, whereas the second uses two

ducted fans. If we consider thrust a resource which can be provided at different flight speeds the behavior of both options for generating thrust at level N+1 can be expressed via resource envelopes (expressed as convex hulls) as depicted in figure 4.12. The union of all possible behaviors at a deeper level of abstraction becomes an abstracted component at a higher level of abstraction called “electric propulsion”. By taking the union we insure that no potentially feasible architectures are disregarded given the defined subsystems at level N+1. However, having taken the union, the thrust resource envelope of electric propulsion is now overly optimistic and will likely generate infeasible architectures. If we take the intersection the opposite is true. To limit this effect, a clustering algorithm can be used to find groups of subsystems at level N+1 which are similar to each other to be used in abstraction.

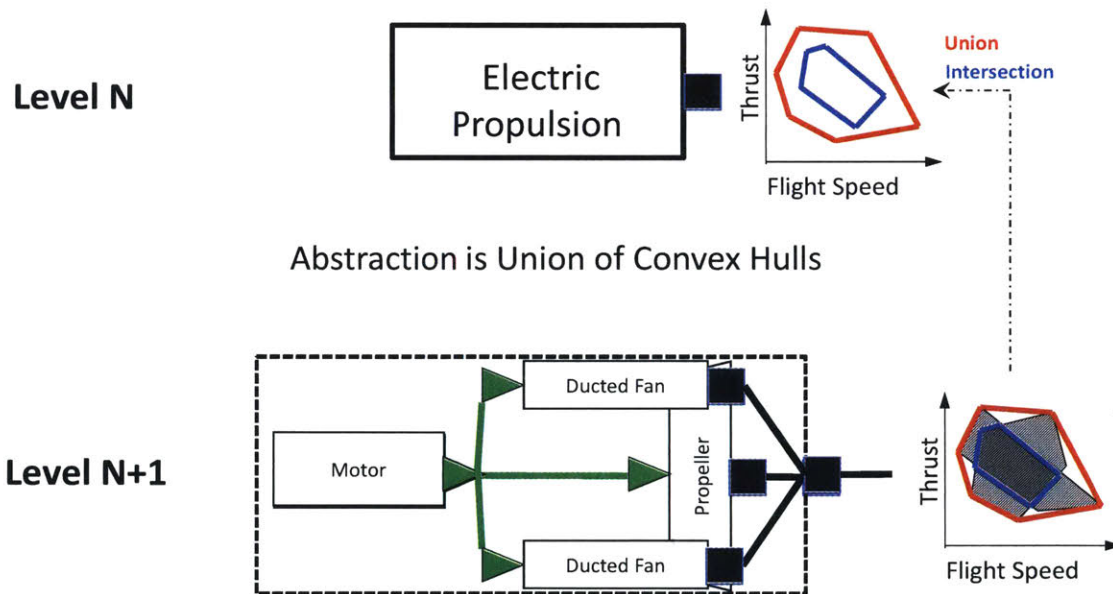


Figure 4.12: Abstraction from level N+1 to level N

Taking the union of resource envelopes at level N+1 can be overly optimistic (generate false positives), whereas taking the intersection will be overly pessimistic (generate false negatives). In figure 4.12 the union of resource envelopes shown in red intersection shown

in blue. We want to be able to create a convex hull between the union and intersection. A significant literature exists on morphing 2 and 3 dimensional polyhedral objects into one another in the computer science literature. One approach is to use the algorithm described in [84]. Implementation of this algorithm is left for future work.

Abstracting subsystems to higher levels of abstraction constrains the design space. Setting module boundaries and defining interfaces at one level of abstraction and then building such a module from more detailed components at a deeper level of abstraction we constrain the space of possible concepts. This is due to the fact that only those combinations of components which conform to the set of interfaces defined at higher level of abstraction are allowed.

Consider the electrical propulsion conceptual component at level N in figure 4.12. In this case, the electrical propulsion module only has a single interface which allows it to transfer thrust to other parts of the system. Defining the propulsion concept in this way, we eliminate all other potential connections that cross the module boundary. For example, the electrical motor component is unable to provide shaft power outside the module boundary. If exploration of the entire system were carried out at a deeper level of abstraction, where motors, propellers and fans are treated as separate components, such module boundary crossing connections would be permitted. On the other hand, if we considered a system in which propulsion were not the only module at a deeper level of abstraction, the number of possible components and connections would be greater (the motor, fan and propeller would be treated as separate components). Assuming that we had up to two fans, up to one propeller and up to one motor as is depicted in figure 4.12 we would have four components representing possible electrical propulsion architectures rather than one resulting from abstraction resulting in 2^3 times more component combinations.

Thus while taking the union of convex hulls does not generate any false negatives for the electric propulsion module itself, the definition of the electric propulsion module at level N does since it only allows interaction with that module through predefined interfaces. This is why, while abstraction is useful for reducing the combinatorial space, to avoid false negatives for the broader system architecting problem, one should model at a level of abstraction which is as deep as available computational resources allow.

4.4.1 Architecture Evolution

Another potential application of convex hull resource envelope intersection is in the redesign or retrofitting of existing architectures. Once a subsystem or set of components is selected for replacement Magellan allows users to fix all other connections or components to be always present by reducing the domain of their binary decision variables from $[0,1]$ to 1 and regenerate architectures. Imagine we only allowed part of an architecture to vary which is shown in the grayed out portion of figure 4.13.

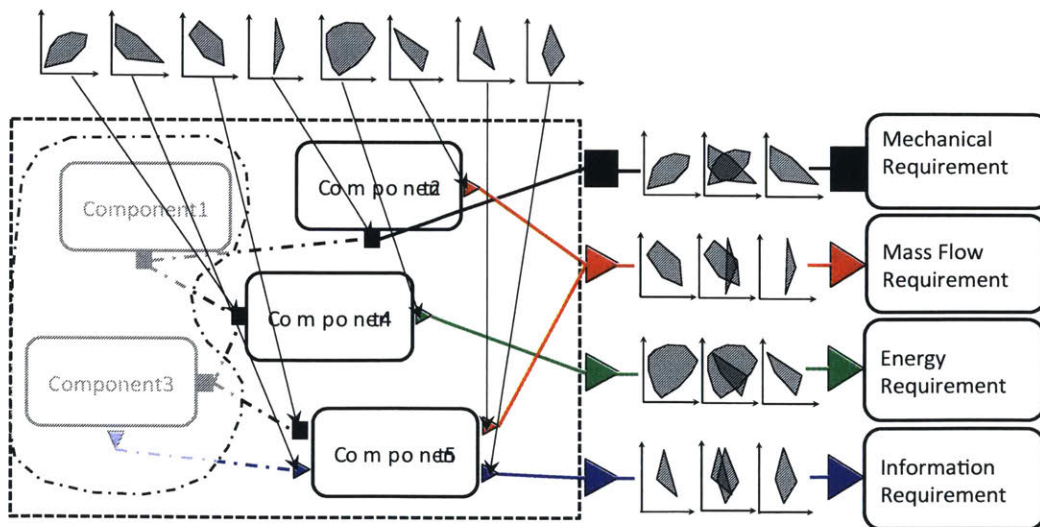


Figure 4.13: Architecture evolution using resource envelopes

We could use the convex hulls representing the interfaces of the fixed part and the requirements components to constrain what new components could be used to once again complete the architecture (i.e. ensure that all interfaces have compatible flows to them and requirements flows are met).

4.4.2 New Components: Application to Technology Roadmapping

We now consider the situation in which existing component types at a given abstraction level are not able to meet requirements. In such a situation, we define a new dummy component and conduct a sensitivity analysis to determine what resource envelopes it would need to enable new architectures to exist. Once resource envelopes are determined at level N we can conduct architecture generation again at a deeper level of abstraction using those resource envelopes as requirements. This is shown graphically in figure 4.14. Thus the first phase provides insights as to what component should be developed to enable a new architecture to exist, while the second phase searches for ways of creating that component.

Note that this proposed approach to generating components builds on previous work [73]. The primary contributions here are that resource flows are represented via convex hulls around operating points from detailed simulation, and the approach is used in architecture generation (component sizes determined during optimization) in addition to configuration generation (component sizes fixed [73]).

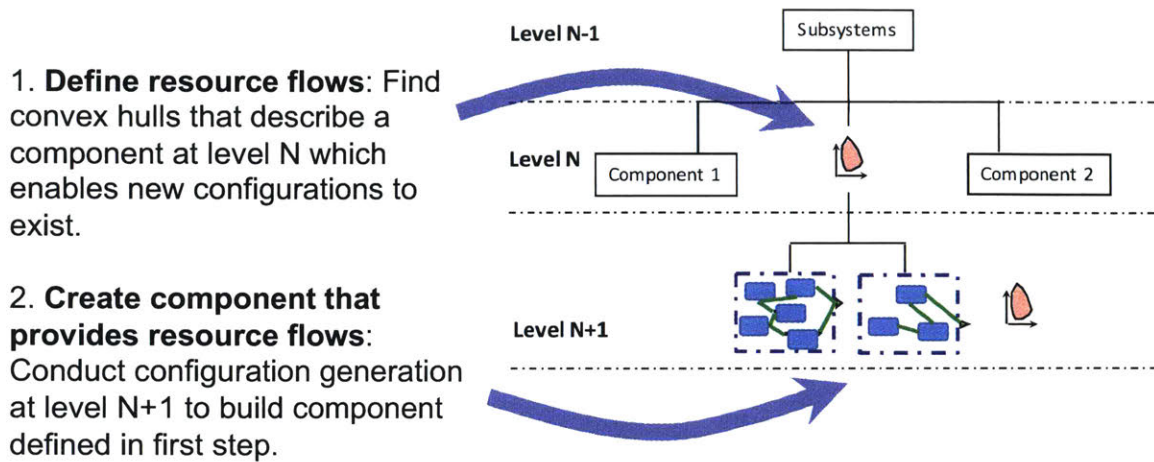


Figure 4.14: New components: First resource flow envelopes at abstraction layer N are defined. This is followed by an attempt to generate these flows with architecture generation at a deeper level of abstraction.

4.5 Search Algorithm

Once a DS2M is generated, we still need to search for feasible architectures. The DS2M is encoded into 2 design vectors. The first design vector expresses the presence or absence of components, while the second expresses the presence or absence of connections. We search for feasible groups of components subject to user defined rules using depth first search with backtracking [82]. We then search for feasible sets of connections between each feasible set of components using depth first search with backtracking. Since we have applied convex hull constraints to the DS2M, the set of possible connections is smaller by some number n connections, making the design space 2^n times smaller.

Depth first search with backtracking assigns values to each of the decision variables in the design vector. After assigning a value to each element of the design vector it checks whether a constraint has been violated. If a constraint has been violated, it “backtracks” and changes the last decision that it made [82]. The rules applied during search are either resource flow

constraints (convex hulls) or additional constraints defined by the designer's experience.

Convex hull constraints reduce the domain of the design vector by eliminating certain design decisions (connections between components) from consideration. Subject matter expert opinion informed rules can be anything from stating the minimum and maximum number of flows for an interface to something slightly more complex, like every oil loop must have exactly one oil pump in it. They can also relate numbers of components to one-another (e.g the number of gearboxes must not be larger than the number of components that can be driven by a gearbox). All of the above are used in this work.

4.6 Performance Evaluation

4.6.1 Simulation

Simulation and/or optimization of architectures occurs after a feasible concept has been generated by Magellan. The purpose of simulation is to compute metrics of interest such as fuel efficiency. The number of architectures generated by Magellan can be in the thousands even with the use of abstraction layers to reduce the size of the design space. For this reason any simulation approach that we take must run quickly (on the order of seconds) and be robust. In other words, simulations must be able to run for thousands of nonlinear heterogeneous models without human intervention. In addition, any simulation environment we select must allow libraries of components to be assembled together in new architectures automatically.

After investigating multiple languages and modeling environments we select the Modelica modeling language and OpenModelica modeling environment for simulation was selected.

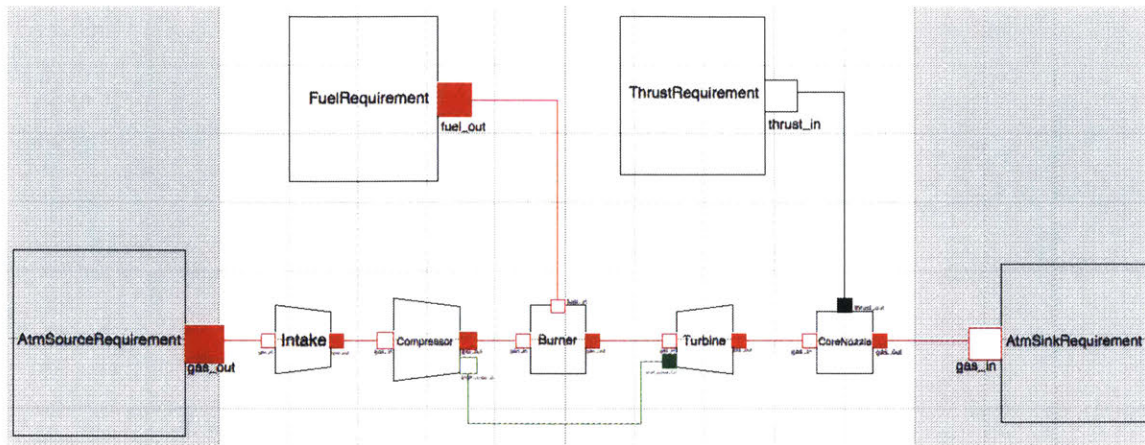


Figure 4.15: Modelica model of simplified turbojet. Run time approximately 2 seconds.

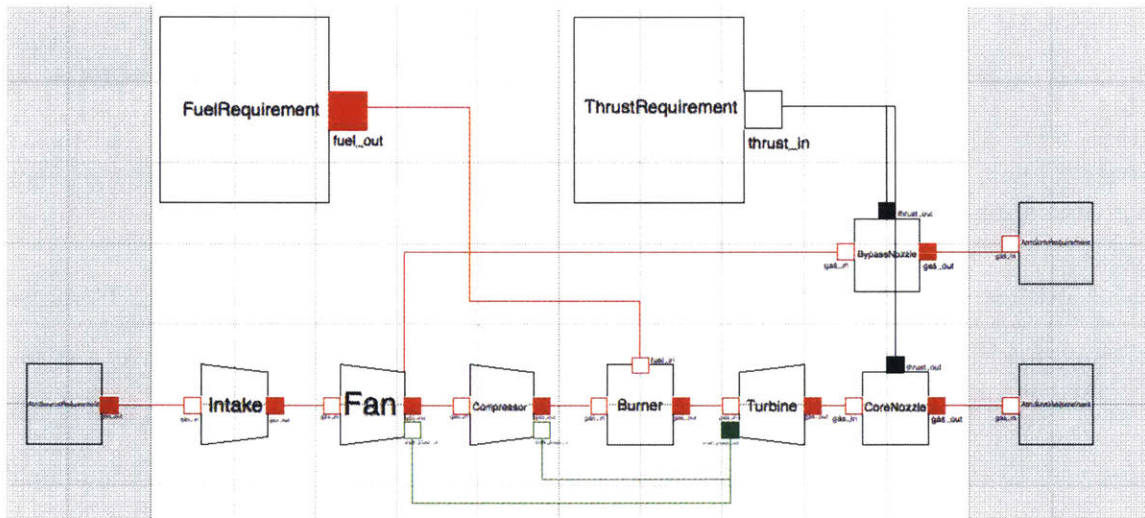


Figure 4.16: Modelica model of simplified turbofan. Run time approximately 2 seconds.

The Modelica language and the OpenModelica modeling environment have proven to be robust for modeling nonlinear heterogeneous systems [85]. The first case study in this work considers air-breathing propulsion systems with nonlinear behavior. Using Modelica models that were created for this case study we determined that even with complex architectures in which multiple fans are powered by a single core through gearboxes (109 governing equations), steady state model execution takes approximately 2-3 seconds. A screen shot of

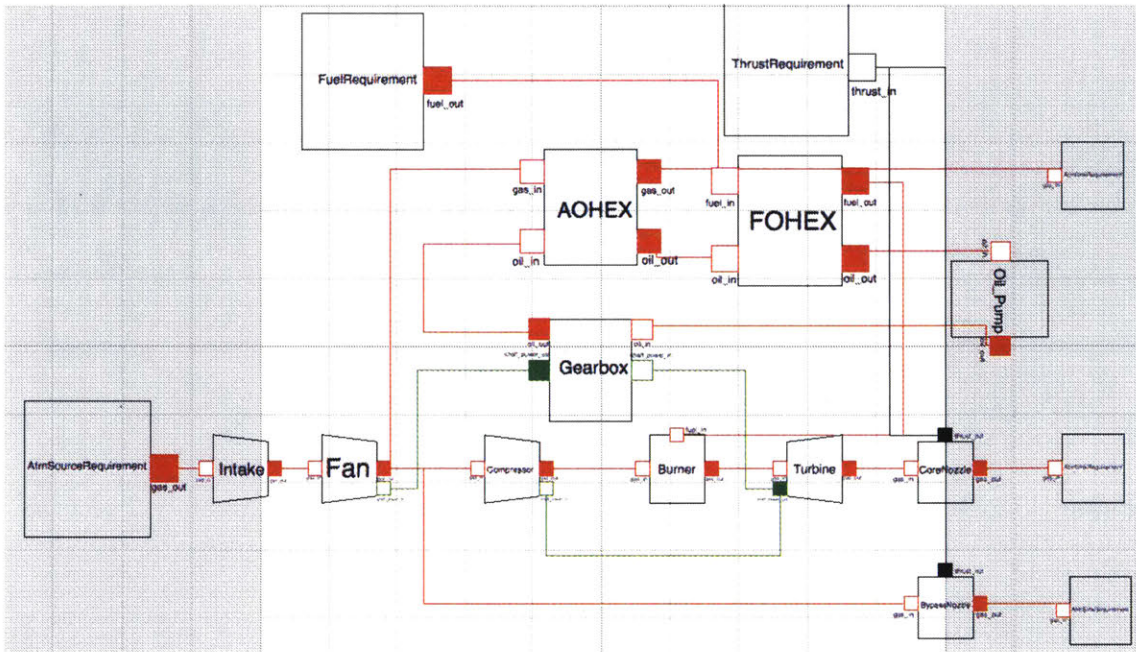


Figure 4.17: Modelica model of simplified geared turbofan. Air-oil (AOHEX) and fuel-oil (FOHEX) heat exchangers are used to cool oil circulating through and lubricating gearbox. Run time approximately 2 seconds.

OpenModelica with one such unconventional architecture is given in figure 4.18. Complex, existing engine architecture models were also examined to check that they converged to a physical solution and that run time was on the order of a few seconds. These are shown in figure 4.15 (turbojet, least complex engine architecture), figure 4.16 (turbofan, similar to most current engine architectures) and figure 4.17 (geared turbofan, similar to new engine architecture from Pratt & Whitney).

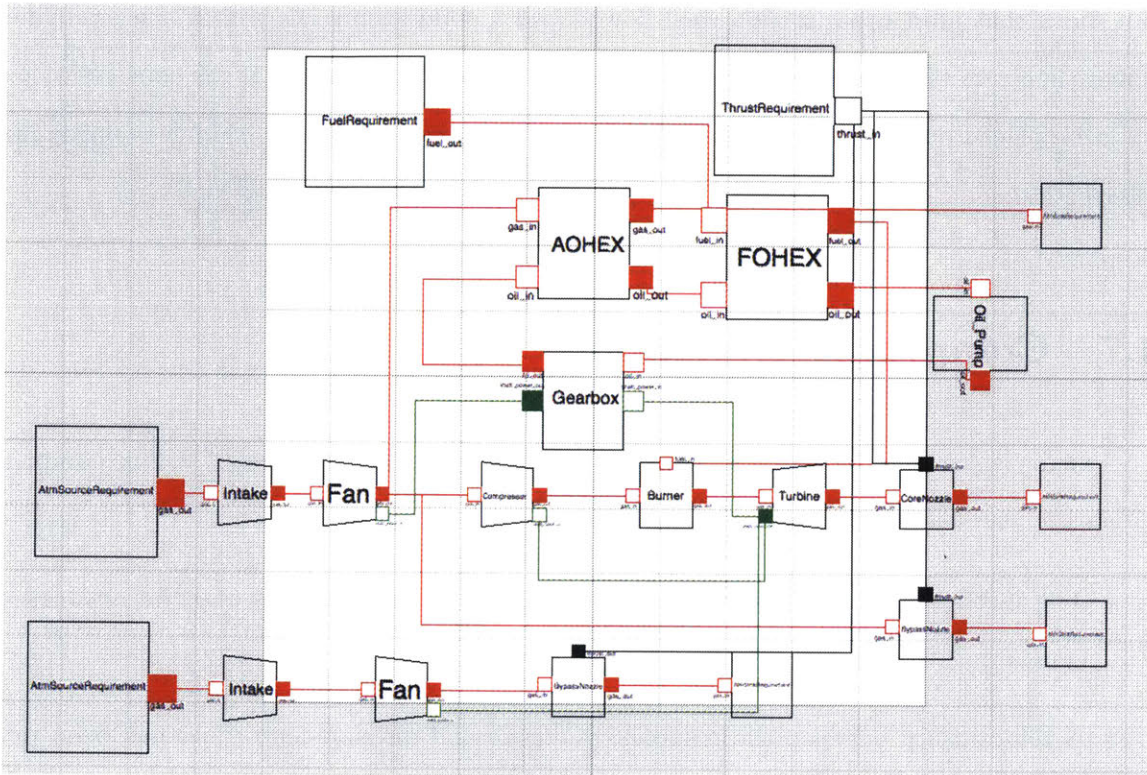


Figure 4.18: Modelica model of unconventional architecture consisting of multiple fans powered by a single turbine. One of the fans is geared. Air-oil (AOHEX) and fuel-oil (FOHEX) heat exchangers are used to cool oil circulating through and lubricating gearbox. Run time: approximately 2-3 seconds.

4.6.2 Multidisciplinary Optimization

To let each architecture put its “best foot” forward, the models allow for design parameters to be changed by an external optimizer for the purposes of multidisciplinary design optimization. Since the feasibility of a architecture depends on intersection of convex hulls, we can use them to inform initial guesses on model variables to increase the likelihood that the model will converge to a physical solution. Note that in principle the convex hulls at component interfaces could also be used to constrain optimization by for example bounding the temperatures, pressures and Mach numbers of flows between components. Alternatively, one

can check that after optimization each flow between components lies within the predefined convex hulls. Returning to the compressor-turbine example in figure 4.5, the constraint that could be placed on the optimizer is that flows between them must lie within the convex hull intersection.

4.7 Complexity Evaluation

Complexity was estimated using a complexity metric from the literature [27]. This complexity metric was selected for the following reasons:

- The metric considers both complexity of individual components and the network of connections between them.
- A relationship between the complexity metric and development effort has been proposed.
- The proposed relationship between complexity and development effort has been validated with human experiments.
- A growing body of evidence supports the proposed relationship between complexity and development effort ([27, 86]).

The complexity metric is given by the following expression:

$$C = \sum_{i=1}^n \alpha_i + \sum_{i=1}^n \sum_{j=1}^n \beta_{ij} A_{ij} \gamma E(A) \quad (4.3)$$

Where n represents the number of components, β_{ij} represents the complexity of each connection between components, A represents the design structure matrix (binary matrix encoding connections between components), $\gamma = 1/n$ is a normalization factor where n is the number

of components and $E(A)$ is the graph energy of binary design structure matrix. In other words, the first term on the right hand side represents component complexity and the second term represents complexity due to the network of connections between components.

The relationship between development effort and complexity given in [27] is of the form:

$$Development\ Effort = AC^k \quad (4.4)$$

Were A and k are constants and C is complexity. According to [27], these factors are specific to problem domain and the organization that is conducting the development effort. If we restrict ourselves to comparing similar products developed by the same organization and normalize complexity relative to a reference architecture, the relationship becomes:

$$\frac{Development\ Effort}{Development\ Effort_{Reference}} = \frac{C^k}{C_{Reference}^k} \quad (4.5)$$

Comparing to a reference architecture eliminates the need to use a specific factor A which would have to be determined from historical development effort information (e.g. cost data). Such data for gas turbine engines is typically proprietary to engine manufacturers. According to [27] and [86] the exponent k typically ranges from 1.5 to 3.5.

4.8 End To End Architecture Generation Example: Mass Spring Damper

This section describes an end to end run of the Magellan architecture generator for a mass spring damper system with up to two springs and dampers. A DS2M is generated. Infeasible connections are then removed from the DS2M using pairwise convex hull constraints. We first search for feasible groups of components subject to user defined rules. We then search for feasible sets of connections between each feasible set of components, to generate individual architectures.

4.8.1 Phase One DS2M Generation

Phase one DS2M generation connects all interfaces of the same type. Since all interfaces here are mechanical this results in a fully connected DS2M, which from a scalability perspective is not useful unless constraints are applied prior to architecture generation 4.19.

4.8.2 Phase Two DS2M Generation

In phase two DS2M generation we eliminate pairwise infeasible connections by checking for convex hull intersection for each pair of interfaces. To apply the convex hull approach we define linear constraints on flows at each interface (convex hulls around operating points are represented by linear constraints) as in figure 4.20. For this example we represent mechanical connections by power (from translational motion) and velocity. The normalized intersection volume of all pairs of interfaces is depicted in gray scale in figure 4.21 in which white indicates zero intersection and black denotes a normalized intersection volume of one (perfect overlap).

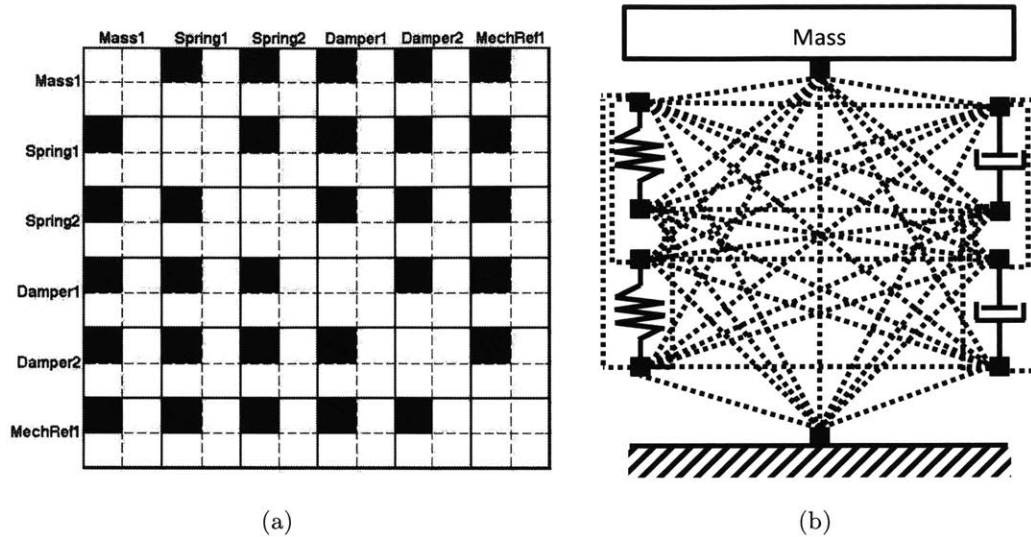


Figure 4.19: Automatically Generated DS2M Example (Mass Spring Damper)(a) Matrix Representation (b) Block Diagram Representation

Note that each connection in the DS2M may represent more than one connection in the system since each component has more than one unique interface (see network diagram in figure 4.20). Applying convex hull constraints results in a reduction of the number of possible connections from 41 to 20 representing a factor of 2^{21} reduction in the design space.

Once the DS2M has been generated the search algorithm finds feasible architectures subject to any additional constraints. In this case only pairwise feasibility resource constraints were considered. This results in the two architectures shown in figures 4.22 and 4.23.

4.9 Architecture Generation Results

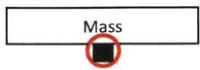





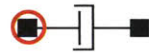

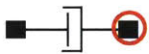
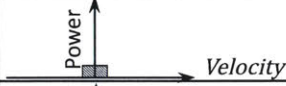


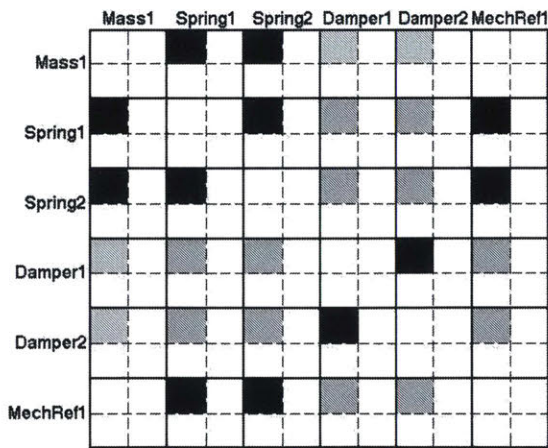
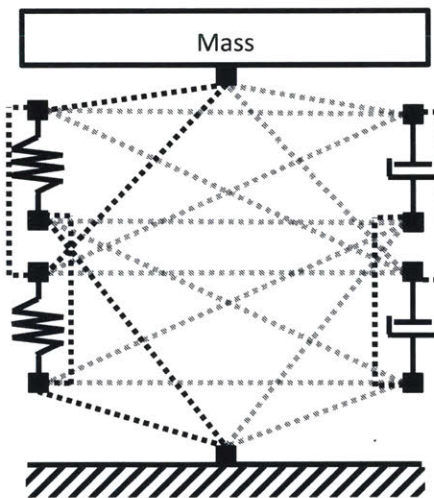
Component Interface	Resource Envelope	Resource Envelope (Graphic)
	$Power \geq 0, Power \leq 1$ $Velocity \geq -1, Velocity \leq 1$	
	$Power \geq 0, Power \leq 1$ $Velocity \geq -1, Velocity \leq 1$	
	$Power \geq 0, Power \leq 0.1$ $Velocity \geq -0.1, Velocity \leq 0.1$	
	$Power \geq 0, Power \leq 0.5$ $Velocity \geq -0.5, Velocity \leq 0.5$	
	$Power \geq 0, Power \leq 0.2$ $Velocity \geq -0.2, Velocity \leq 0.2$	
	$Power \geq 0, Power \leq 0.2$ $Velocity \geq -0.2, Velocity \leq 0.2$	

Figure 4.20: Interface Resource Constraints



(a)



(b)

Figure 4.21: Convex Hull Filtered DS2M Example. Grey entries indicate partially overlapping convex hulls. (Mass Spring Damper)

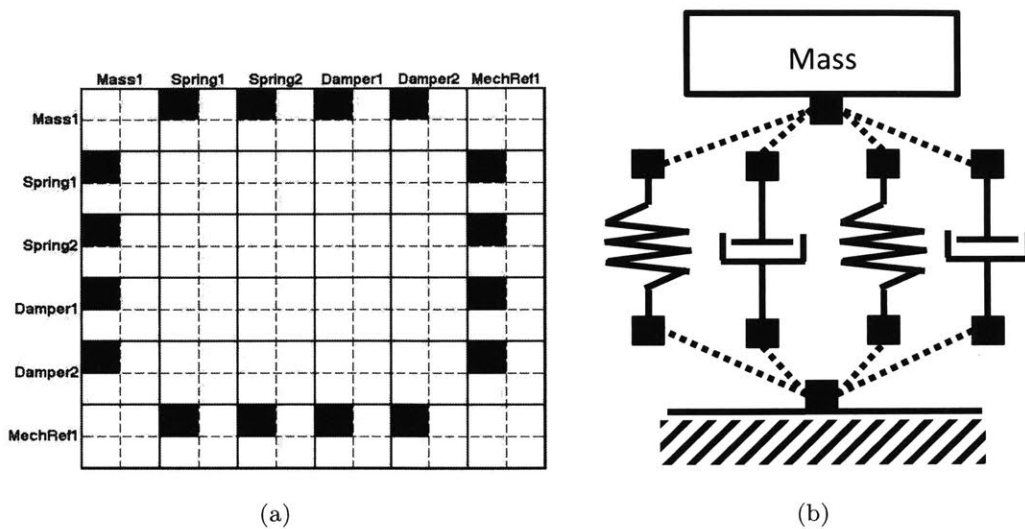


Figure 4.22: Architecture 1

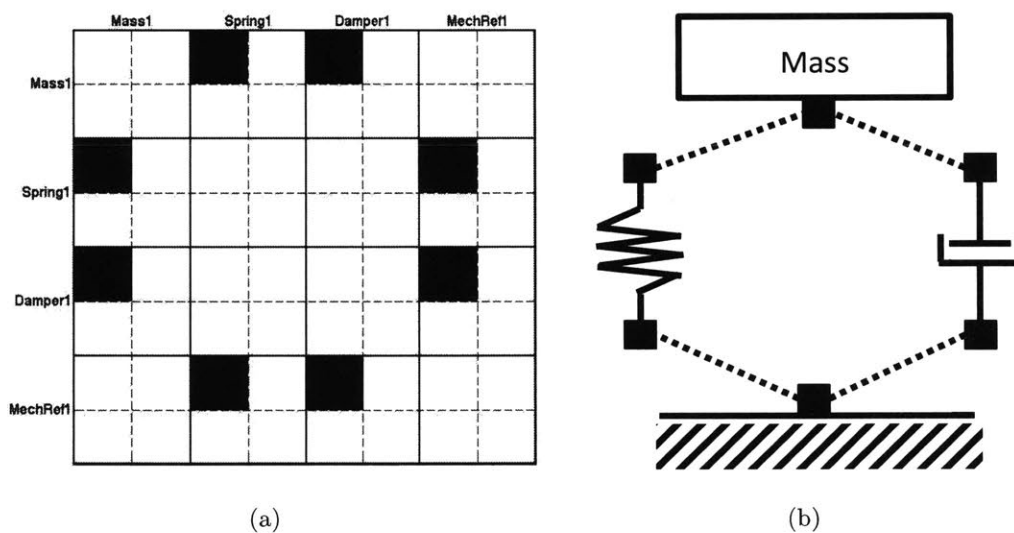


Figure 4.23: Architecture 2

4.10 High Level Summary

Figure 4.24 describes at a high level how Magellan operates. First, given a library of components and bounds on the maximum number of instances of components, all possible connec-

tions are encoded into a DS2M (network diagram view given). Every feasible architecture (DSM) will consist of a subset of the components and connections in the DS2M. Components and connections are encoded into a vector of binary design decisions (zero or one representing the presence or absence of a component or connection). Then, prior to search, infeasible connections are removed using physics based rules (convex hulls). A depth first search algorithm subject to constraints is first used to find feasible component combinations subject to expert experience-based rules. Then for each feasible set of components, the same algorithm searches for a feasible set of connections to generate an architecture or configuration. An example of how feasibility of local networks of components is enforced was given in this chapter, but was not used for the case studies. Feasible architectures are finally simulated and in the air-breathing engine case study, optimized.

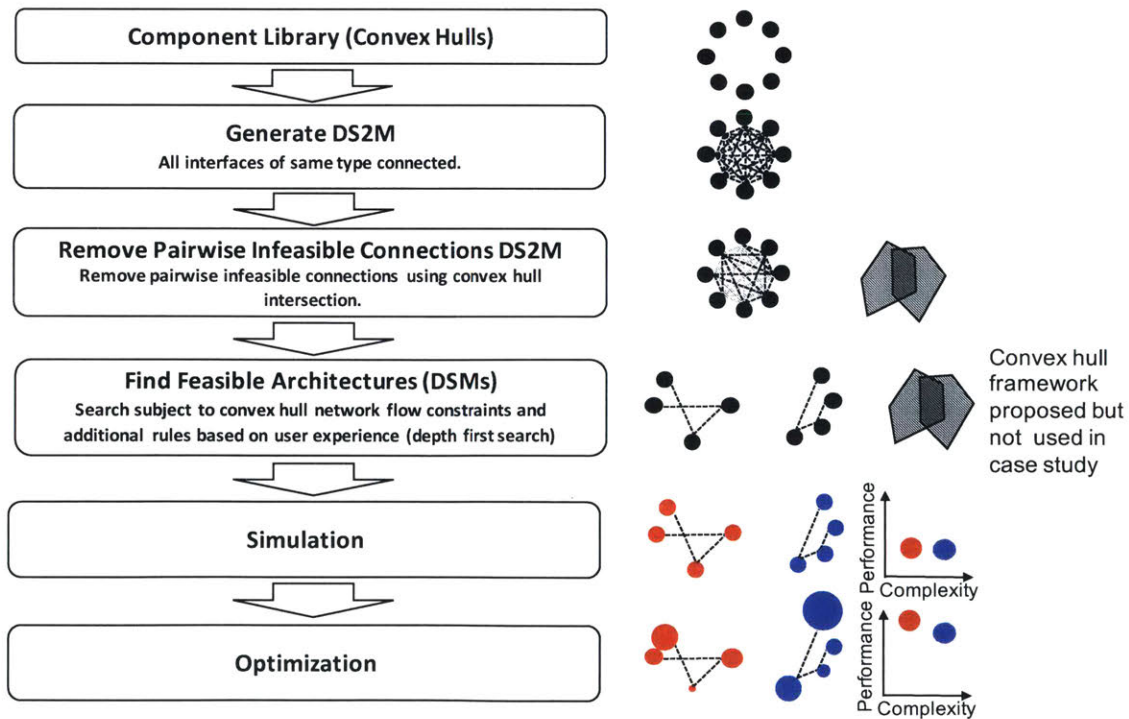


Figure 4.24: Magellan flow chart

4.11 Limitations of Approach

The limitations of the approach were mentioned in individual subsections earlier in this chapter. We now summarize and discuss them in detail.

4.11.1 1D Network Flow Representation

The first limitation of the presented approach to architecture generation, evaluation, optimization and performance-complexity tradespace exploration is that feasible architectures are generated using flow consistency constraints (convex hulls) applied to a one-dimensional network of flows between known interfaces. While a one-dimensional network flow representation allows rapid evaluation of novel architectures, it does not explicitly take into account physical layout of components or the details regarding how they are connected to one another. This means that transfer of energy between components due to anything else is not directly captured. In addition, load transfers in more than a single dimension are not captured directly. Fluid stagnation pressure losses due to 3-dimensional layout of components are also not directly captured. These limitations mean that while a particular network of connections between components may be feasible according to the constraints implemented in the architecture generation and one-dimensional models/optimization, the architecture may prove to be infeasible upon closer examination due to an absence of a feasible three-dimensional layout. The point is that a one-dimensional network representation is optimistic and therefore a useful tool for initial screening of architectures without elimination of potentially promising concepts.

4.11.2 State Dependency during Architecture Generation

Another important limitation is that component convex hulls are computed for components in isolation. As we assemble components together into an architecture the possible inputs of components is constrained by the intersection of convex hulls. One can think of every additional component as an additional set of constraints. The convex hulls are not updated through nonlinear simulation of individual components to reflect this while generating architectures. The feasibility of networks of connections *can be* considered, but this is done considering only local networks of connected interfaces. The only point at which state dependency (explicit dependence of outputs on component inputs) is captured is DS2M generation and simulation and multidisciplinary design optimization of potentially feasible architectures.

4.11.3 Abstraction

As mentioned earlier, when defining a subsystem at a higher level of abstraction (subsystem boundary and interfaces which allow it to connect to other subsystems) we necessarily limit the interactions that components within that subsystem can have with neighboring subsystems. For example, if we define an electrical propulsion subsystem at Level N which only has a mechanical thrust interface, it cannot share electrical energy with other subsystems. In exchange for reducing the combinatorial space, we therefore eliminate potentially feasible connections across a defined subsystem boundary, limiting the space of concepts that can be generated. This is a source of false negatives (elimination of potentially promising designs due to user defined module boundaries). The concept is shown graphically in figure 4.25.

Figure 4.25 also illustrates the potential for false positives if the set of convex hulls of

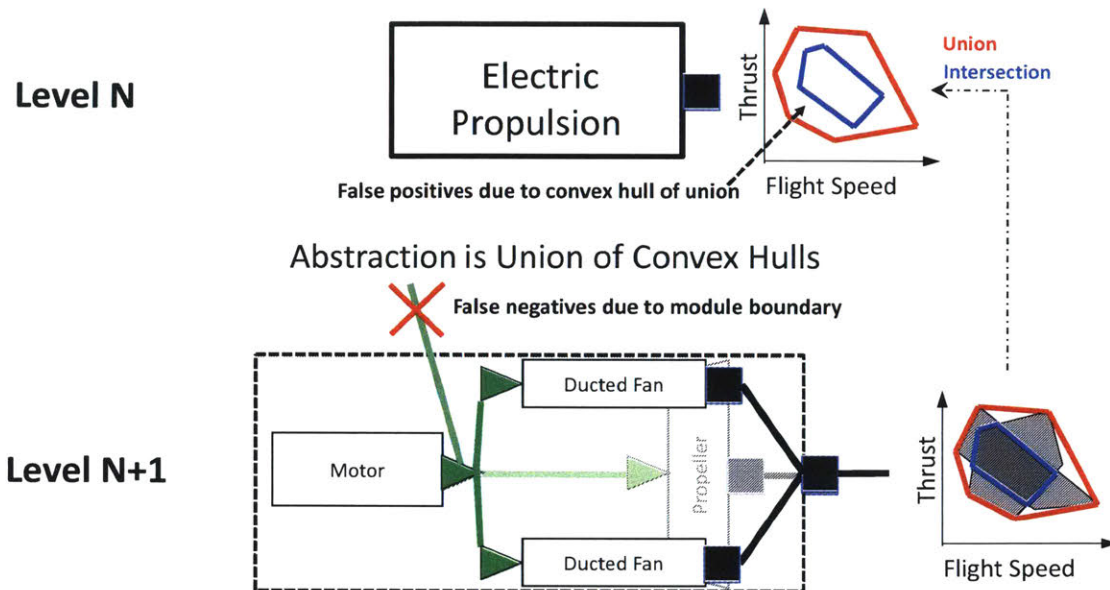


Figure 4.25: Abstraction from level N+1 to level N. Module boundary crossing connections limited by description of subsystem at Level N. Difference between union and intersection generates an optimistic component at Level N whose behavior is the convex hull of the union of convex hulls at level N+1

different instantiations of a subsystem at level N+1 has a small intersection to union ratio. For example, if one instance of an electric propulsion system is designed for flight speed A (for example a ducted fan architecture) and another is optimized for flight speed B (propeller architecture) the abstracted electrical propulsion component is represented via the convex hull of the union of the convex hulls at level N+1. This results in “optimistic” abstracted electric propulsion subsystem performance capable of efficiently providing thrust at both flight speeds, enabling architectures at level N to exist that are not necessarily realizable with current technology at level N+1.

4.11.4 Decoupling of Feasible Architecture Generation and Evaluation and Optimization of those Architectures

In the presented approach architecture generation and evaluation/optimization are decoupled. This means that when generating architectures, the approach does not have a direct way of filtering architectures based on results of evaluation and optimization. One way of filtering dominated architectures is to introduce additional rules from subject matter expert interviews. This, however, can introduce bias towards existing architectures. Another mechanism for filtering dominated concepts is conducting preliminary design space exploration at a higher level of abstraction and selecting promising architectures for further exploration. This can be done either by explicitly considering convex hulls as described earlier in the methodology section or by explicitly applying additional constraints to design space exploration that remove dominated architectures from the feasible set. For example if no electric propulsion concepts are on the Pareto frontier at Level N, all components/interfaces associated with them can be removed at level N+1. Future work will examine combining evaluation and exploration.

4.11.5 Robustness of Models and Optimization

Another limitation of the proposed approach is related to robustness of models and optimization. When component behavior is linear, robustness of models (e.g. convergence to a physical solution) is typically not an issue. However, complex electromechanical systems often are nonlinear. In such instances, initial guesses for model parameters/design variables may not provide the solver/optimizer with a feasible starting point. Typically optimization is carried out on one or a handful of selected architectures. When using computational approaches that generate hundreds or thousands of architectures, initial guesses for design

variables based on intuition may no longer yield feasible solutions. Feasible initial guesses for simulation/optimization depend on the architecture itself, resulting in a need for different starting points for different architectures.

In the engine case study that will be discussed in the next chapter the following techniques were used to increase the likelihood of convergence:

1. Simulation was carried out with different design parameters prior to optimization. Using the results of simulation runs an intuition for the underlying physics was developed. Initial constraints for optimization were defined based on this experience.
2. A genetic algorithm was used with nonlinear constraints and a penalty function for cases in which solver convergence did not occur. The genetic algorithm found “good” initial guesses for design variables which made architectures feasible.
3. Since genetic algorithms do not guarantee local optimality, the resulting design vectors for each architecture from the genetic algorithm were used as an initial guess for gradient-based optimization.

Chapter 5

Air Breathing Propulsion Case Study

In this chapter we discuss an air breathing propulsion case study for civil aircraft. The high level objectives of this chapter is the following:

- Generate and optimize the set of feasible air-breathing propulsion architectures for uninstalled fuel consumption.
- Examine the complexity-performance tradespace of these architectures.

To limit the scope of the analysis we define performance in terms of thrust specific fuel consumption (engine fuel consumption (kg/s) divided by thrust (Newtons)). We include electrical generators and motors in addition to gearboxes and turbomachinery components in the analysis to allow both mechanically driven and hybrid electric engine architectures to be assessed.

This chapter consists of three sections. We first discuss gathering of historical data for validation purposes. We then present engine models at two levels of abstraction. Finally we show and discuss results from engine architecture generation and optimization.

5.1 Historical Data

During the last 70 years civil engine performance metrics like fuel consumption, NOX emissions and noise levels have all been greatly improved due to component level changes (e.g. blade geometry and materials change) and architectural changes (changes in types of components and connections between them). We gathered engine specification data from “Jane’s Aero-Engines” [8] to:

1. Create a database of engines against which models described later in this chapter will be validated.
2. Provide intuition for the *statistical* relationships between variables that describe engine design.

We briefly discuss the primary for gas turbine engine system level design. Figure 5.1 shows a diagram of a typical turbofan engine [17]. The majority of the data used in this analysis pertain to engines of this type. For turbofan engines most of the thrust comes from the fan in the front which is powered by a core consisting of compressors, a burner, turbines and a core nozzle [36]. Most of the airflow through the fan bypasses the core.

Important design parameters include the fan pressure ratio (fpr), the overall pressure ratio of the engine (opr) and the burner exit temperature T_4 [36], [30]. A full set of variable names and descriptions is given in table 5.1.

Not all independent and dependent variables are available for all engines in [8]. Figure 5.2 graphically depicts which variables are missing for each of the 62 engines in the dataset. The variable that is missing for all engines is the burner entry temperature T_3 . Figure 5.2 also shows that a number of entries for burner exit temperature (T_4) are missing from the

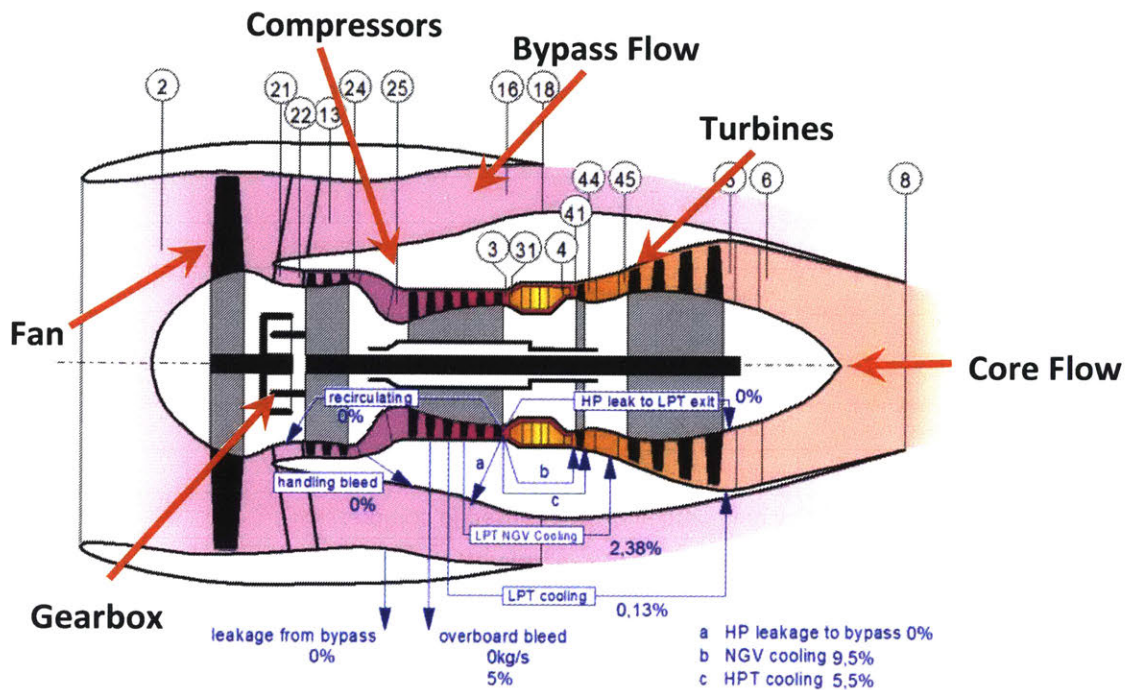


Figure 5.1: Generic gas turbine engine diagram adapted from [17]

dataset. By reducing the number of variables used in our analysis we can increase the number of engines that we are able to consider.

We first examine the correlation between variables which describe engine performance. These are thrust specific fuel consumption at cruise ($tsfcc$) and thrust specific fuel consumption at takeoff ($tsfct$). Both of these values are present for 19 engines out of the 62. Normalized fuel consumption at cruise is plotted against normalized fuel consumption at takeoff in figure 5.3. The two variables are highly correlated. We therefore use only thrust specific fuel consumption at takeoff in the remainder of this analysis of historical data.

We now examine a larger group of variables (both dependent and independent). These include bypass ratio (bpr), overall engine pressure ratio (opr), fan pressure ratio (fpr) and

Variable Name	Variable Description
type	Encodes whether engine is used for civil aircraft only or if it is used by both civil and military aircraft
bpr	Bypass ratio
opr	Overall pressure ratio
mass flow	Total mass flow of air
mass	Mass of engine
length	Length of engine
diameter	Diameter of engine
n spools	Number of spools (separate rotating shafts)
n compressors	Number of compressors
n turbines	Number of turbines
n compression stages	Total number of stages used in compression
n expansion stages	Total number of expansion stages used in expansion
combustor	Combustor type
variable nozzle	Encodes whether a variable nozzle is present
variable geometry	Encodes whether variable geometry (blade pitch can be altered)
contrarotating	Encodes whether spools rotate in different directions
fuel	Encodes fuel type
open rotor	Encodes whether the engine is of open rotor type
year	Year of entry into service
afterburner	Encodes whether an afterburner is present
maxt	Maximum thrust
cthr	Cruise thrust
tsfct	Thrust specific fuel consumption at takeoff
tsfcc	Thrust specific fuel consumption at cruise
st	Specific thrust

Table 5.1: Variable names and descriptions

burner exit temperature (T_4) in figure 5.4. Fan pressure ratio is positively correlated with thrust specific fuel consumption (lower fan pressure ratios associated with lower fuel consumption). Overall pressure ratio (opr) and burner exit temperature (T_4) are negatively

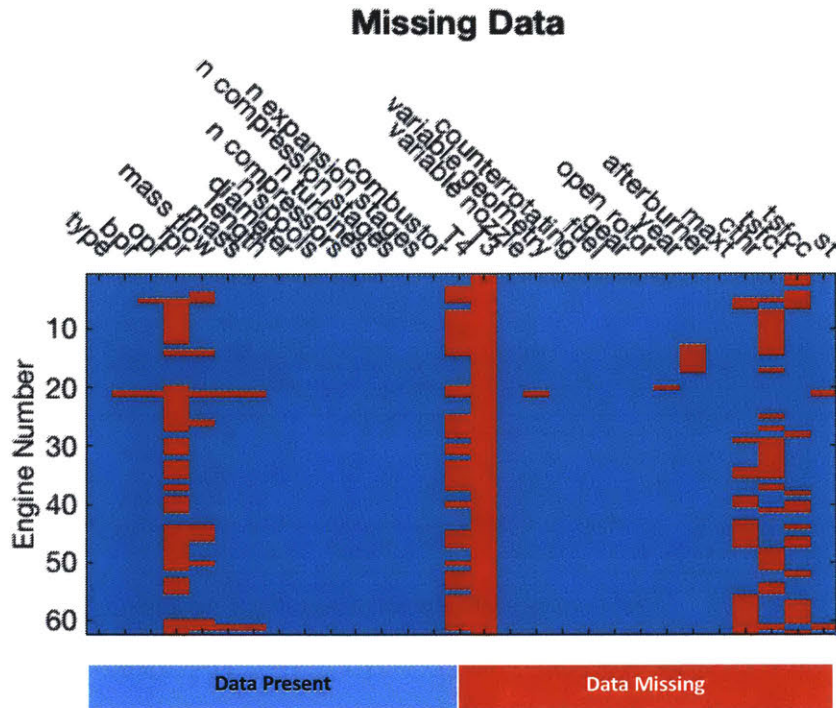


Figure 5.2: Graphical depiction of data missing from [8]. Each row represents data available for an engine. Missing data is highlighted in red.

correlated with fuel consumption. Bypass ratio (bpr) is also negatively correlated with fuel consumption. These results are consistent with literature on air-breathing propulsion systems and with basic cycle analysis [36], [30].

Finally we carry out linear regression of thrust specific fuel consumption using the design variables outlined earlier in this section. We therefore regress thrust specific fuel consumption on overall pressure ratio, fan pressure ratio and burner exit temperature. The regression diagnostics indicate a good fit ($R^2 = 0.989$ and a root mean squared error of 0.0282). The regression coefficient of opr is -0.1984 (higher opr reduces fuel consumption), that of fpr is 0.44971 (lower fpr decreases fuel consumption) and that of T_4 is -1.4392 (higher T_4 reduces fuel consumption).

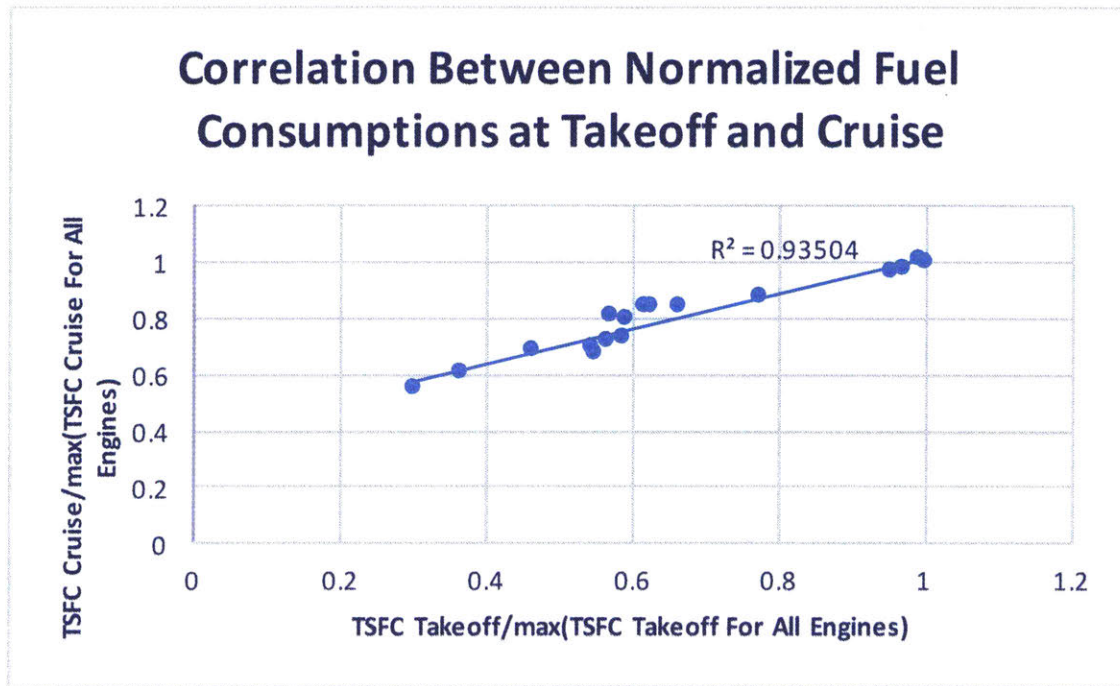


Figure 5.3: Correlation between dependent variables (19 engines with both cruise and takeoff thrust specific fuel consumption)

In summary, we have created a database of engine data which will be used for validation of engine models to be described later in this chapter. A regression analysis confirms first principles reasoning [36]. The database contains high level design variables like fan pressure ratio, overall pressure ratio and burner exit temperature and also contains information about performance at two operating points (cruise and takeoff fuel consumption) which will be useful for validation.

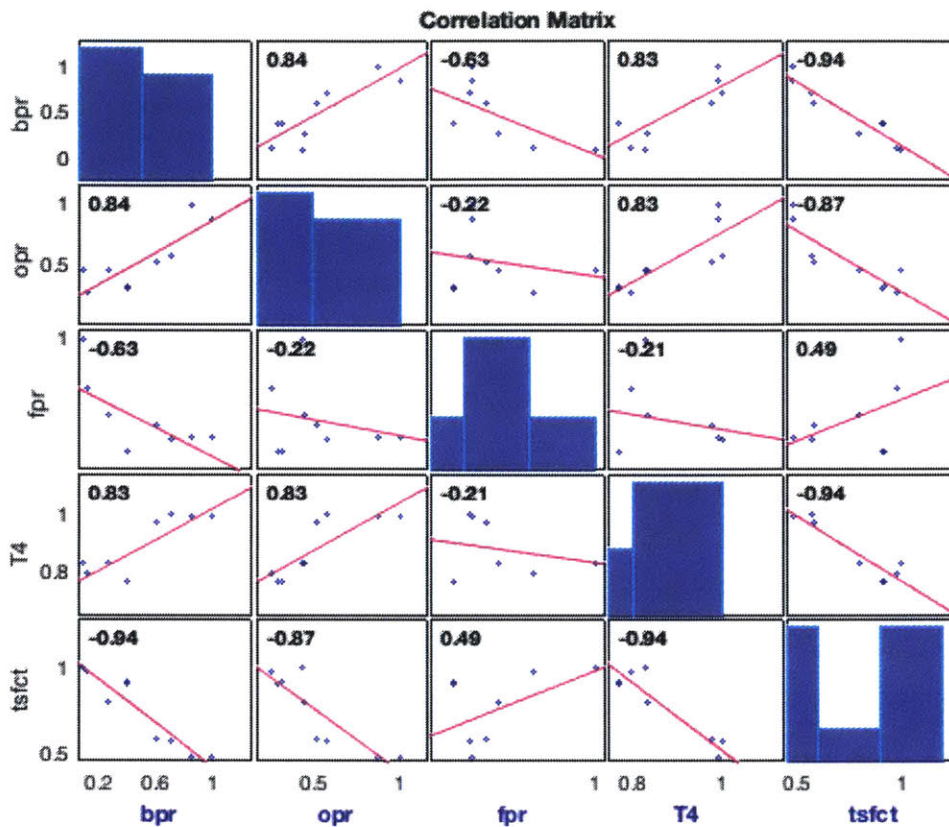


Figure 5.4: Correlation between design variables and dependent variables. 9 engines out of the 62 in the dataset have all of these present.

5.2 Engine Reconfigurable Model

The final step of the architecture generation process is evaluation. The primary objective of engine component models was to enable rapid simulation and optimization of different engine architectures for minimum thrust specific fuel consumption. To evaluate engine fuel consumption two models at two different levels of abstraction were constructed in Modelica modeling language. These are described in the sections below. A third level of abstraction was used to generate convex hulls for turbomachinery components using compressor, fan

and turbine maps from [16]. The third level of abstraction was also used to define the components used to build models at level two. We begin our discussion with the most detailed model (level three) since this was used to generate the convex hulls at higher levels of abstraction for turbomachinery components. Subsystems at level two were constructed by assembling components at level three together. Subsystems at level one in turn were constructed by combining subsystems at level two together. Turbomachinery component models (fans, compressors and turbines) at level two were simplifications of those at level three ¹.

5.2.1 Abstraction Level Three: Reconfigurable Engine Model

Gas turbine engine components at abstraction level three (the most detailed level) were modeled using fundamental thermodynamic relations from [36]. The purpose of engine component/subsystem models in the context of this work was to provide insight into trends in engine performance and complexity. The models were required to be robust, e.g. converge to physical solutions for hundreds or even thousands of different novel engine architectures with limited or no human intervention. Given the high level objective of observing trends of this work, a lumped parameter model library of components was constructed. The primary assumptions used throughout this analysis are those outlined in [36] and are as follows:

1. Flow is treated as one-dimensional.
2. Ideal gas.
3. Lumped parameter models of components.
4. Steady state operation.

¹Note that the engine component model library was built to architect engines for subsonic civil transport aircraft. For this reason afterburners (components which increase thrust by burning fuel upstream of the nozzle) were not modeled.

We now describe the underlying mathematics of each of the component models in detail starting with a generic intake for subsonic ($M \leq 0.8$) and transonic ($M < 1.0$) flight.

Intake The primary function of an intake is to recover the stagnation pressure of the flow [36]. The intake was treated as non-ideal in the sense that it did not perfectly recover the stagnation pressure of the free stream. The intake was assumed to be subsonic, and typical temperature, pressure ranges were used to define its convex hull (expressed as a set of linear constraints) based on [36]. Both the convex hull representing gas input and output of the intake were represented by 4-dimensional polytopes consisting of temperature, pressure, Mach number and normalized mass flow. The governing equations of the intake are as follows. See nomenclature section for definition of individual terms.

Conservation of Energy (Adiabatic process)

From conservation of energy and the definition of total enthalpy:

$$T_{0_{out}} = T_{in} \cdot \left(1 + \frac{(\gamma - 1)}{2} M^2 \right) \quad (5.1)$$

Stagnation Pressure Variation (Adiabatic Process)

An idealized intake recovers the stagnation pressure of the flow:

$$p_{0_{out_{isentropic}}} = p_{in} \cdot \left(1 + \frac{(\gamma - 1)}{2} M^2 \right)^{\frac{\gamma}{\gamma - 1}} \quad (5.2)$$

We model a stagnation pressure loss in the inlet as a stagnation pressure ratio PR .

$$p_{0_{out}} = PR p_{0_{out_{isentropic}}} \quad (5.3)$$

See appendix for assumed parameters.

Conservation of Mass

Conservation of mass for a control volume, where $\dot{m}_{gas_{in}}$ denotes flux of mass flow into a control volume, $\dot{m}_{gas_{out}}$ denotes mass flux of mass flow out a control volume and dm/dt denotes the rate of change of mass within the control volume depicted in figure 5.5, is given by equation 5.5. Applying conservation of mass:

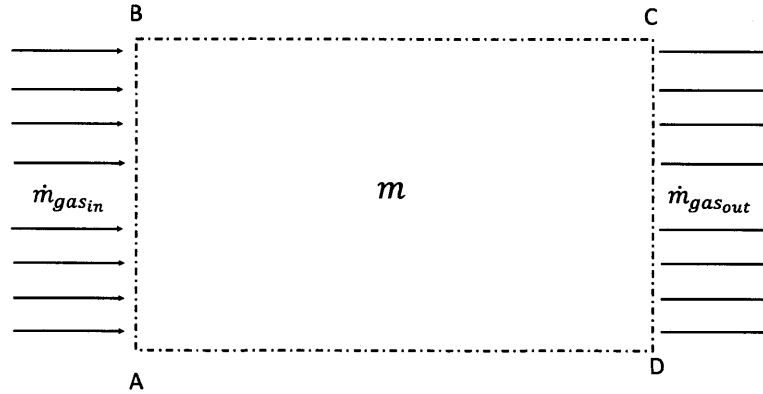


Figure 5.5: Control Volume Representation of Flow in Intake

$$\dot{m}_{gas_{in}} - \dot{m}_{gas_{out}} = \frac{dm}{dt} \quad (5.4)$$

For steady flow assumption the dm/dt is 0.

$$\dot{m}_{gas_{out}} = \dot{m}_{gas_{in}} \quad (5.5)$$

Mass Estimation

The intake was modeled as a hollow cylinder as depicted in figure 5.6 where the ratio of the outer and inner radii is given by $w = r_{inner}/r_{outer}$. The intake length L was assumed to be equal to the fan diameter based on upstream influence arguments and typical geometries from [8]. We estimate the mass of the intake based on an aggregate density $\rho_{aggregate}$ and

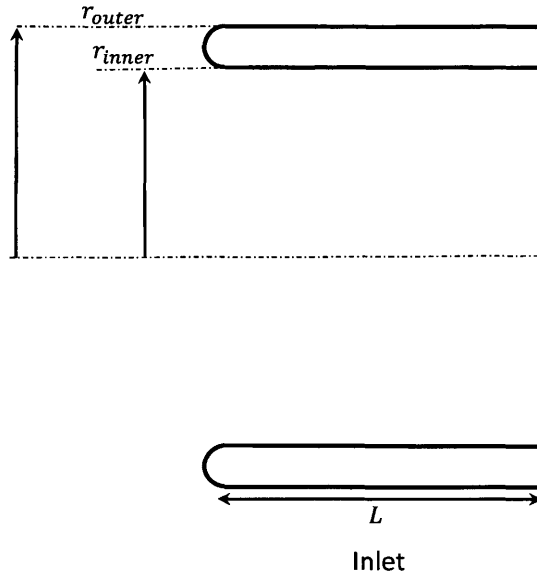


Figure 5.6: Simplified Intake Side View

the volume of the part of the intake which contains material (e.g. ignoring the empty central section). This gives the following expression for the intake.

$$m_{intake} = \pi \rho_{aggregate} (r_{outer}^2 - r_{inner}^2) r_{inner} L \quad (5.6)$$

Where we define $r_{inner}/r_{outer} = w$:

$$r_{outer} = \frac{r_{inner}}{w} \quad (5.7)$$

Substituting back into the equation:

$$m_{intake} = \pi r_{inner}^3 \left(\frac{1}{w^2} - 1 \right) \rho_{aggregate} L \quad (5.8)$$

To solve these equations one has to make assumptions on w and $\rho_{aggregate}$. These assumptions are given in the appendix and are based on/were tuned based data from [8] and [36].

Fan The primary function of a fan is to increase the stagnation pressure of a large flow volume (relative to flow moving through the core of the engine) to create high propulsive efficiency thrust. We can use either polytropic or isentropic efficiency to express irreversibility in the compression process. Polytropic efficiency represents the isentropic efficiency of an infinitesimal compression which is why it is sometimes called the “small stage” compression efficiency [87]. The upper bound on fan pressure ratio was small (less than 1.6 based on [8]), isentropic efficiency was used in the calculation, based on [16] (see figure 5.7)).

A fan has 3 primary interfaces: gas flow in, gas flow out and shaft power in. To generate the convex hulls for the fan, inputs of temperature, pressure, corrected speed and corrected mass flow were varied resulting in a shaft power convex hull represented by minimum and maximum shaft power required by the fan and four-D polytopes representing possible temperature, pressure, Mach number and mass flow outputs. The convex hull of the fan was then manually created based on the corner points for temperature, pressure, mass flow and Mach number range generated from detailed simulation. An example of a fan performance map is given in figure 5.7 from [16]. The map consists of isentropic efficiency contours (represented by the dotted lines), normalized corrected speed lines, plotted against corrected mass flow on the x axis and pressure ratio on the y axis. The governing equations of a fan are as follows from [36]. See nomenclature section for definition of individual terms.

Corrected Speed

Corrected speed is given by the following equation from [36].

$$\text{Corrected Speed} = \frac{\omega}{\sqrt{\frac{T}{T_{ref}}}} \quad (5.9)$$

Corrected Mass Flow

Corrected mass flow is given by the following equation from [36].

$$\text{Corrected Mass Flow} = \frac{\dot{m} \sqrt{\frac{T}{T_{ref}}}}{\frac{p}{p_{ref}}} \quad (5.10)$$

Stagnation Pressure Ratio (Adiabatic Process)

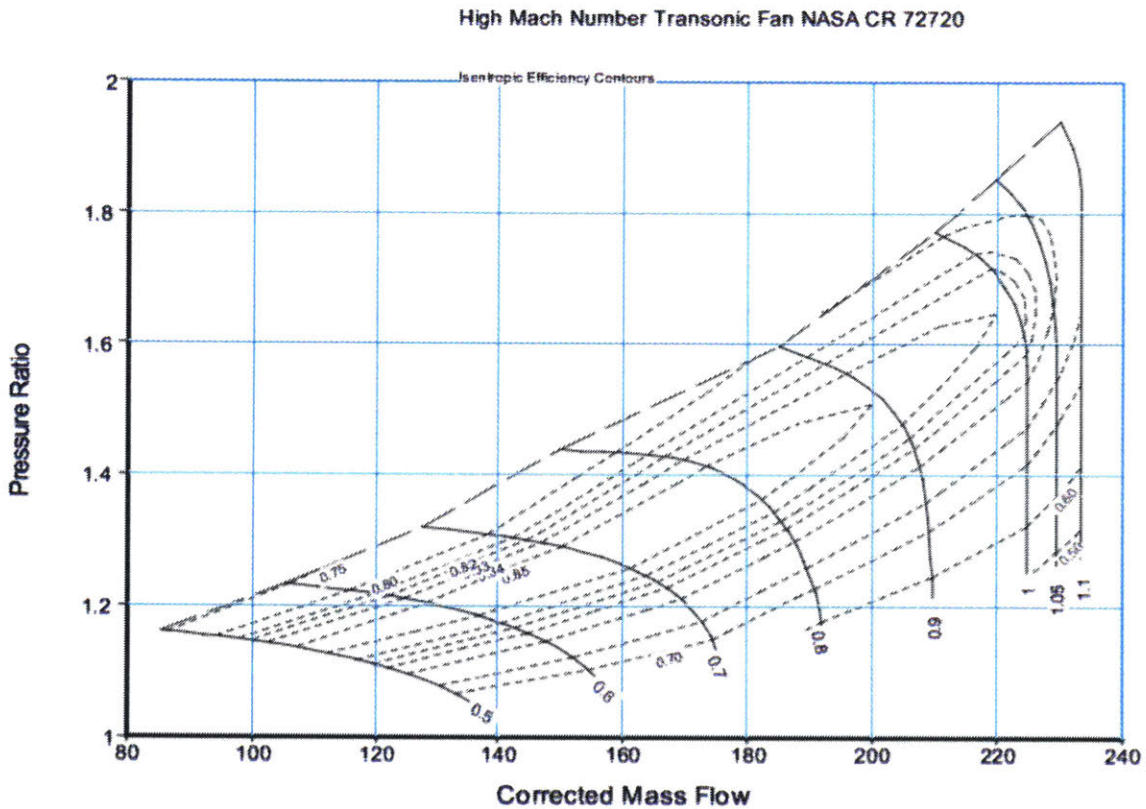


Figure 5.7: Fan performance map from [16]

$$p_{0out} = PR p_{0in} \quad (5.11)$$

Stagnation Temperature (Isentropic Process)

$$T_{0out_{isentropic}} = T_{0in} (PR)^{\frac{\gamma-1}{\gamma}} \quad (5.12)$$

$$\eta_{isentropic} = \frac{T_{0outisentropic} - T_{0in}}{T_{0out} - T_{0in}} \quad (5.13)$$

We include the polytropic efficiency formulation for completeness.

$$T_{0out} = T_{0in} (PR)^{\frac{\gamma-1}{\eta_{polytropic}\gamma}} \quad (5.14)$$

Where

$$\eta_{isentropic} = \frac{\left[1 - PR^{\frac{\gamma-1}{\gamma}}\right]}{\left[1 - PR^{\frac{\gamma-1}{\eta_{polytropic}\gamma}}\right]} \quad (5.15)$$

Conservation of Energy

The shaft power required to drive the fan is given by the following:

$$Power_{in} = cp(T_{0out} - T_{0in}) \dot{m}_{gasin} \quad (5.16)$$

Conservation of Mass

We enforce conservation of mass by stipulating that the mass flux into the component must equal the flux out.

$$\dot{m}_{gasout} = \dot{m}_{gasin} \quad (5.17)$$

Estimation of Mass

We estimate the mass of the fan component by estimating its volume and multiplying by an aggregate density. The cross-sectional area which contains flow of any turbomachinery component is given by:

$$A = \frac{\dot{m}_{gasin}}{\rho u} \quad (5.18)$$

Where:

$$\rho = \frac{P}{RT} \quad u = \sqrt{\gamma RTM} \quad (5.19)$$

From isentropic relations:

$$P = P_0 \left(1 + \frac{\gamma - 1}{\gamma} M^2\right)^{\frac{-\gamma}{\gamma - 1}} \quad (5.20)$$

From conservation of energy:

$$T = T_0 \left(1 + \frac{\gamma - 1}{\gamma} M^2\right)^{-1} \quad (5.21)$$

Rearranging and substituting into the area equation:

$$A = \dot{m} \frac{RT}{P\sqrt{\gamma RTM}} = \dot{m} \frac{\sqrt{R} \sqrt{T}}{\sqrt{\gamma} PM} \quad (5.22)$$

Substituting isentropic relations:

$$A = \dot{m} \frac{\sqrt{R} \sqrt{T_0}}{\sqrt{\gamma} P_0 M} \frac{\left(1 + \frac{\gamma - 1}{2} M^2\right)^{-\frac{1}{2}}}{\left(1 + \frac{\gamma - 1}{2} M^2\right)^{\frac{-\gamma}{\gamma - 1}}} \quad (5.23)$$

Simplifying:

$$A = \dot{m} \frac{\sqrt{R} \sqrt{T_0}}{\sqrt{\gamma} P_0 M} \left(1 + \frac{\gamma - 1}{2} M^2\right)^{\frac{\gamma + 1}{2(\gamma - 1)}} \quad (5.24)$$

Having established the area of flow we now need to make some assumptions about the geometry of turbomachinery components to estimate their mass. While in this section our objective is computing the mass of a single stage fan, some of the relationships we derive will apply to multistage axial turbomachinery components for both compression and expansion.

The fan was modeled as an annulus as shown in figure 5.8 where r_{inner} represents the distance from the rotational axis of the fan to the base of its blades (hub diameter) and r_{outer} represents the distance from the axis of rotation to the fan blade tip.

The mass of a single stage fan is given by the following:

$$m_{fan} = \pi r_{outer}^2 L \rho_{aggregate} \quad (5.25)$$

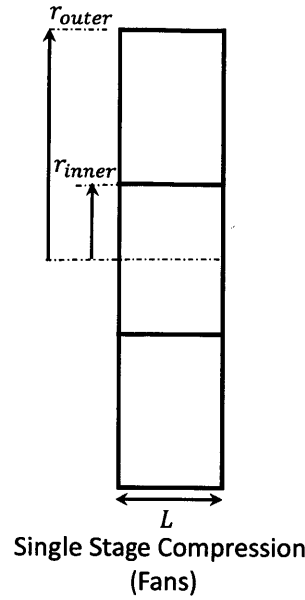


Figure 5.8: Fan Simplified Side View

Where to find L we need to have an estimate of blade size and the number of stages. We first estimate the number of stages. For the fan we assume a single stage since this is the most common configuration from historical data [8] in civil applications. For multi-stage turbomachines, focusing on axial compressors and turbines assuming an average stagnation pressure ratio for all the stages PR_{stage} which is related to the overall turbomachine pressure ratio PR by the following:

$$PR = (PR_{stage})^{n_{stages}} \quad (5.26)$$

We find:

$$n_{stages} = \frac{\ln(PR)}{\ln(PR_{stage})} \quad (5.27)$$

We now estimate the length of turbomachinery components based on the number of stages. In order to do this we define AR as the aspect ratio of blades and the hub to tip ratio of

blades w in terms of inner and outer radii of components r_{inner} , r_{outer} and blade chord:

$$w = \frac{r_{inner}}{r_{outer}} \Rightarrow r_{inner} = wr_{outer} \quad (5.28)$$

$$AR = \frac{(r_{outer} - r_{inner})}{chord} \Rightarrow chord = \frac{r_{outer}(1 - w)}{AR} \quad (5.29)$$

Where from basic geometry for a circular annulus:

$$\pi(r_{outer}^2 - r_{inner}^2) = A \quad (5.30)$$

Substituting and simplifying:

$$\pi r_{outer}^2(1 - w^2) = A \Rightarrow r_{outer} = \sqrt{\frac{A}{\pi(1 - w^2)}} \quad (5.31)$$

Assuming that each stage of turbomachinery consists of a rotor and stator:

$$L = 2n_{stages}chord = 2n_{stages}\frac{r_{outer}(1 - w)}{AR} \cos \phi \quad (5.32)$$

Where ϕ represents the angle of the blade relative to the axis of rotation of the turbomachine.

Note that the factor of two comes from the assumption that each stage has a rotor and stator of approximately equal size. Substituting and simplifying (assuming $\cos \phi \approx 1$):

$$L = 2\frac{\ln(PR)}{\ln(PR_{stage})}\frac{1 - w}{AR}\sqrt{\frac{A}{\pi(1 - w^2)}} \quad (5.33)$$

Substituting back into the mass equations from before:

$$m_{fan} = 2\pi r_{outer}^2 n_{stages} \frac{r_{outer_1}(1 - w)}{AR} \rho_{aggregate} \quad (5.34)$$

Where:

$$n_{stages} = \frac{\ln(PR)}{\ln(PR_{stage})} \quad r_{outer} = \sqrt{\frac{A}{\pi(1-w^2)}} \quad A = \dot{m} \frac{\sqrt{R} \sqrt{T_0}}{\sqrt{\gamma} P_0 M} \left(1 + \frac{\gamma-1}{2} M^2\right)^{\frac{\gamma+1}{2(\gamma-1)}} \quad (5.35)$$

In order to solve these equations one has to make assumptions on M , w and AR and $\rho_{aggregate}$. These assumptions are given in the appendix and are based on typical values from [8] and [36].

Gearbox The gearbox is modeled as a shaft power transferring device with finite efficiency. The gearbox had shaft power and oil interfaces.

Conservation of Energy

The power required to drive the gearbox is directly related to its efficiency $\eta_{gearbox}$ and the power demand on the output of gearbox $Power_{out}$. It is also assumed that all the heat generated in the gearbox \dot{Q} is transferred to the oil flow through it \dot{m}_{oil} .

$$Power_{in} = Power_{out} + \dot{Q} = \eta_{gearbox} Power_{in} + \dot{Q} \quad (5.36)$$

$$\dot{Q} = \dot{m}_{oil} c_{oil} (T_{oil_{out}} - T_{oil_{in}}) \quad (5.37)$$

Estimation of Mass

The estimation of the mass of the gearbox was conducted based on its power to weight ratio from [88] which gave a range of values for geared turbofans in the 30,000lb thrust class. A nominal value of 100 hp/pound was assumed based on the range of values given in [88].

Converting to SI units which gives:

$$m_{gearbox} = (\text{Power To Weight})Power_{in} = 100 \cdot 0.7457/0.453Power_{in} \approx 164000W/kg \quad (5.38)$$

Compressor In this analysis the term compressor was used to represent any existing combination of low, medium and high pressure compressors from [16] (see figure 5.9 for compressor maps). Put differently, the compressor component represents a compression system rather than individual compressors. The governing equations of the compressor component are identical to that of the fan. The primary difference is that compressor maps typically have higher pressure ratios as shown in the figure below [16]. Isentropic efficiency was fixed using an optimistic bound based on typical values from [16]. To generate the convex hulls for the compressor inputs of temperature, pressure, corrected speed and corrected mass flow were varied resulting in a shaft power convex hull represented by minimum and maximum shaft power required by the fan and four-D polytopes representing possible temperature, pressure, Mach number and mass flow inputs and outputs. Mass flow was normalized. The convex hull of the compressor system was then manually created based on the corner points for temperature, pressure, mass flow and Mach number range generated from detailed simulation.

Stagnation Pressure (Adiabatic Process)

$$p_{0out} = PRp_{0in} \quad (5.39)$$

Stagnation Temperature (Adiabatic Process)

$$T_{0out_{isentropic}} = T_{0in} (PR)^{\frac{\gamma-1}{\gamma}} \quad (5.40)$$

$$\eta_{isentropic} = \frac{T_{0out_{isentropic}} - T_{0in}}{T_{0out} - T_{0in}} \quad (5.41)$$

We include the polytropic efficiency formulation for completeness.

$$T_{0out} = T_{0in} (PR)^{\frac{\gamma-1}{\eta_{polytropic}\gamma}} \quad (5.42)$$

Where

$$\eta_{isentropic} = \frac{\left[1 - PR^{\frac{\gamma-1}{\gamma}}\right]}{\left[1 - PR^{\frac{\gamma-1}{\eta_{polytropic}\gamma}}\right]} \quad (5.43)$$

Conservation of Energy

The shaft power required to drive the compressor is given by the following.

$$Power_{in} = cp(T_{0out} - T_{0in}) \dot{m} \quad (5.44)$$

Conservation of Mass

We enforce conservation of mass by stipulating that the mass flow rate into the component must equal the flow rate out.

$$\dot{m}_{gas_{out}} = \dot{m}_{gas_{in}} \quad (5.45)$$

Estimation of Mass

The mass of the compressor was estimated in a similar manner to that of the fan. The primary difference was that the compressor system was treated as a truncated conical annulus as shown in figure 5.10. Given this geometry, the mass of the compressor system can be

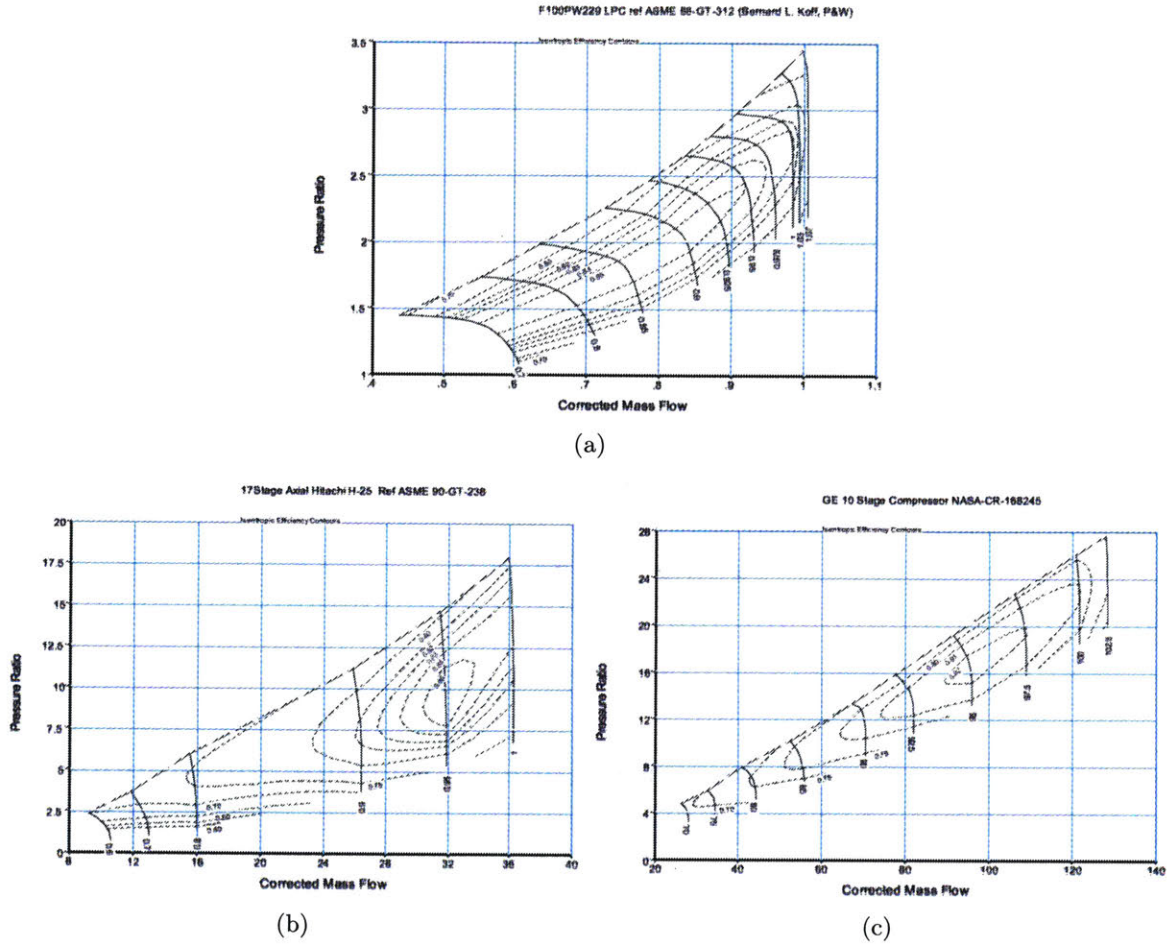


Figure 5.9: Low (a), Intermediate (b), High (c) Pressure Compressor Maps [16]

estimated via the relationship below:

$$m_{compressor} = \frac{1}{3}\pi(r_{outer1}^2 + r_{outer1}r_{outer2} + r_{outer1}^2)L\rho_{aggregate}f \quad (5.46)$$

Where:

$$n_{stages} = \frac{\ln(PR)}{\ln(PR_{stage})} \quad r_{outer} = \sqrt{\frac{A}{\pi(1-w^2)}} \quad A = \dot{m} \frac{\sqrt{R}}{\sqrt{\gamma}} \frac{\sqrt{T_0}}{P_0 M} \left(1 + \frac{\gamma-1}{2} M^2\right)^{\frac{\gamma+1}{2(\gamma-1)}} \quad (5.47)$$

To solve these equations one has to make assumptions for M , w and AR , f (representing

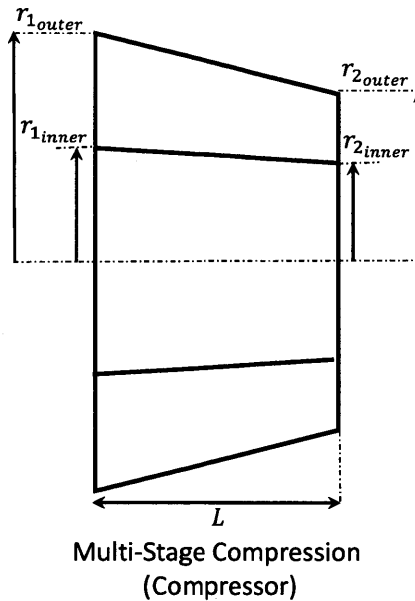


Figure 5.10: Compressor Simplified Side View

a multiplier on the volume to represent blade separation and ducting between components) and $\rho_{aggregate}$. These assumptions are given in the appendix and are based on typical values from [8].

Burner The primary function of a burner as it is modeled here, is to provide heat addition at constant pressure, as is the case for the Brayton cycle [36]. In this case the burner was assumed to have a finite stagnation pressure loss PR .

Stagnation Pressure

$$p_{0_{out}} = PR_{burner} p_{0_{in}} \quad (5.48)$$

Conservation of Energy

We make the simplifying assumption that all of chemical energy released by combustion of

fuel is transferred to the gas flow through the burner.

$$\dot{m}_{fuel} = \frac{\dot{m}_{gasin} cp (T_{0in} - T_{0out})}{c_{fuel}} \quad (5.49)$$

Conservation of Mass

We enforce conservation of mass by stipulating that the mass flow rate into the component must equal the flow rate out.

$$\dot{m}_{gasout} = \dot{m}_{gasin} \quad (5.50)$$

Estimation of Mass

The burner mass was estimated in a similar manner to the fan. In this case, however, the burner was approximated as a cylinder with outer diameter equal to the exit of compressor and a length equal to its diameter based on engine diagrams from [8]. The aggregate density used in estimating burner mass is given in the appendix and was selected along with all the other aggregate densities, based on data from [8].

$$m_{burner} = \pi r_{outer}^3 \rho_{aggregate} \quad (5.51)$$

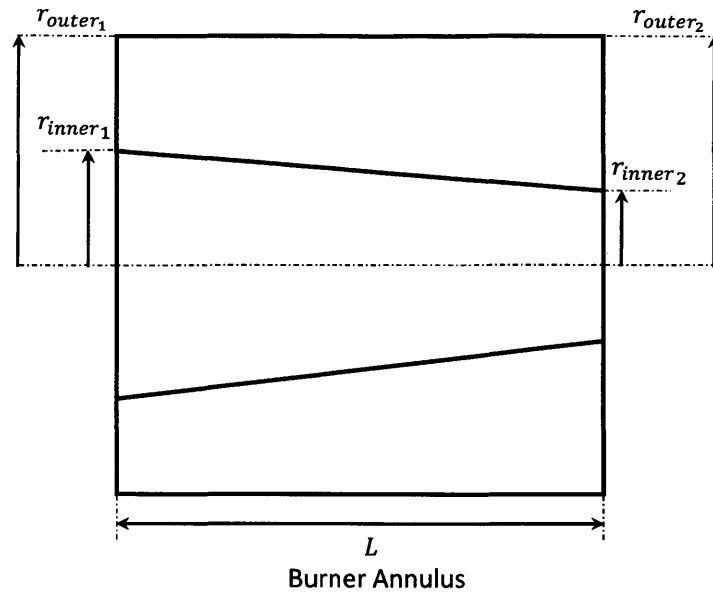


Figure 5.11: Burner Simplified Side View

Turbines The primary function of the turbine is to convert thermal energy of the working fluid into shaft power. The turbine was modeled in a similar manner to the compressor and fan in that turbine maps from [16] (see figure 5.12) were used to determine convex hulls of shaft power and gas flow interfaces. As with the compressor, the turbine component represents a turbine system (any combination of low pressure, medium and high pressure turbines). We include both the polytropic efficiency and isentropic efficiency formulation of turbine equations for completeness but use isentropic efficiency.

Stagnation Pressure (Adiabatic Process)

$$p_{0_{out}} = \frac{p_{0_{in}}}{PR} \quad (5.52)$$

Stagnation Temperature (Adiabatic Process)

$$\eta_{isentropic} = \frac{T_{0_{out_{isentropic}}} - T_{0_{in}}}{T_{0_{out}} - T_{0_{in}}} \quad (5.53)$$

$$T_{out} = T_{in} \left(\frac{p_{amb}}{p_{0_{in}}} \right)^{\frac{\gamma-1}{\gamma}} \quad (5.54)$$

$$T_{0_{out}} = T_{0_{in}} PR^{-\frac{\eta_{polytropic}\gamma-1}{\gamma}} \quad (5.55)$$

Conservation of Energy

The shaft power required to drive the turbine is given by the following.

$$Power_{out} = \dot{m}_{gas_{in}} cp(T_{0_{in}} - T_{0_{out}}) \quad (5.56)$$

Conservation of Mass

We enforce conservation of mass by stipulating that the mass flow rate into the component must equal the flow rate out.

$$\dot{m}_{gas_{out}} = \dot{m}_{gas_{in}} \quad (5.57)$$

Estimation of Mass

As was the case with the compressor the turbine was treated as a truncated conical annulus as show in figure 5.13. Using the the same approach as that used for the compressor, the turbine mass is given by the following:

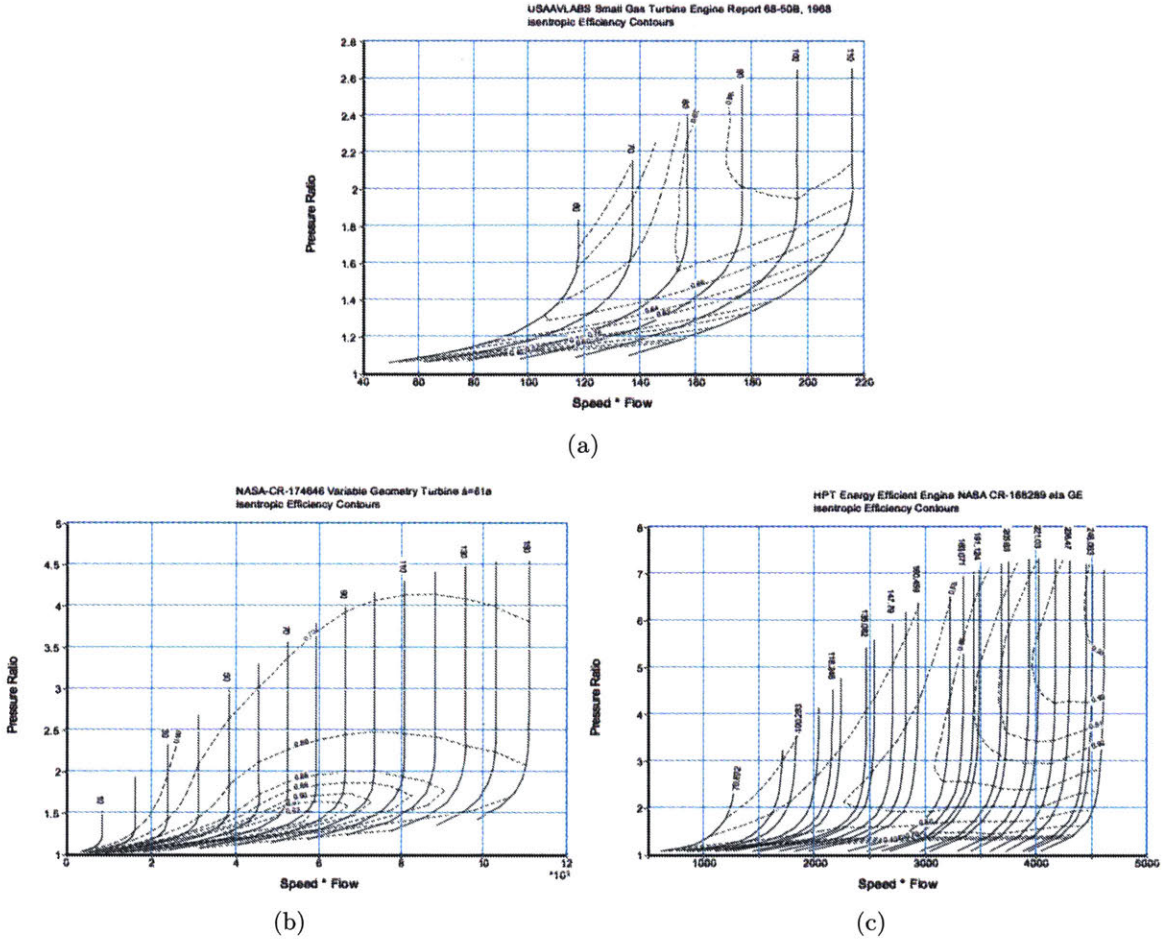


Figure 5.12: Low (a), Intermediate (b), High (c) Pressure Turbine Maps [16]

$$m_{turbine} = \frac{1}{3}\pi(r_{outer_2}^2 + r_{outer_1}r_{outer_2} + r_{outer_1}^2)L\rho_{aggregate}f \quad (5.58)$$

Where:

$$n_{stages} = \frac{\ln(PR)}{\ln(PR_{stage})} \quad r_{outer} = \sqrt{\frac{A}{\pi(1-w^2)}} \quad A = \dot{m} \frac{\sqrt{R}}{\sqrt{\gamma}} \frac{\sqrt{T_0}}{P_0 M} \left(1 + \frac{\gamma-1}{2} M^2\right)^{\frac{\gamma+1}{2(\gamma-1)}} \quad (5.59)$$

In order to solve these equations one has to make assumptions on M , w and AR , f (representing a multiplier on the volume to represent blade separation and ducting between

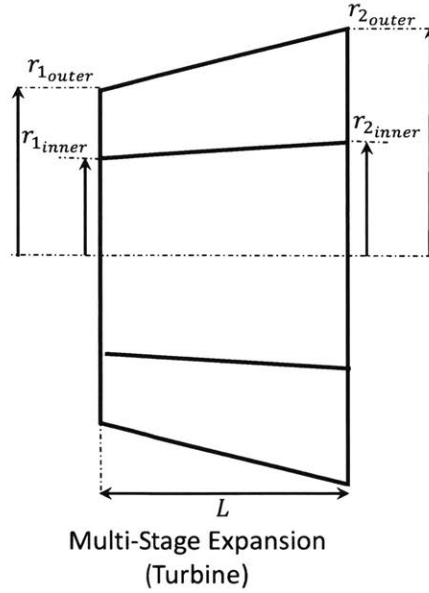


Figure 5.13: Turbine Simplified Side View

components) and $\rho_{aggregate}$. These assumptions are given in the appendix and are based on typical values from [8].

Core Nozzle The primary function of a nozzle is to expand the flow to produce thrust. We assume that the nozzle expands the flow to local atmospheric pressure. The governing equations of the core nozzle are given by the following from basic thermodynamic relations [36].

Temperature (Isentropic Expansion)

$$T_{out} = T_{0in} (PR)^{\frac{\gamma-1}{\gamma}} \quad (5.60)$$

Pressure (Isentropic Expansion)

$$p_{out} = p_{amb} \quad (5.61)$$

Conservation of Energy

$$u_e = \sqrt{2cp(T_{0_{in}} - T_{out})} \quad (5.62)$$

Conservation of Mass

$$\dot{m}_{gas_{out}} = \frac{p_{out}}{RT_{out}} u_e A \quad (5.63)$$

$$\dot{m}_{gas_{in}} = \dot{m}_{gas_{out}}$$

Conservation of Momentum (Thrust)

$$Thrust = \dot{m}_{gas_{out}} (u_e - u_{flight}) \quad (5.64)$$

Estimation of Mass

The core nozzle was assumed to have a small mass compared to other major components within the engine and was neglected.

Bypass Nozzle The bypass nozzle was modeled in a similar manner to the core nozzle. The primary difference between the two was that the permissible temperatures/pressures (convex hulls) of the bypass nozzle were lower than that of the core nozzle component.

Estimation of Mass

The bypass nozzle was modeled as a hollow cylinder. In this case the section of the nacelle which contained mass flow was the central one, as shown in the figure below. We assume based on previous designs from [8] that the length of the nacelle is approximately equal to the diameter of the fan.

We estimate the mass of the bypass nozzle based on an aggregate density $\rho_{aggregate}$ and the volume of the part of the bypass nozzle which contains material (e.g. ignoring the empty

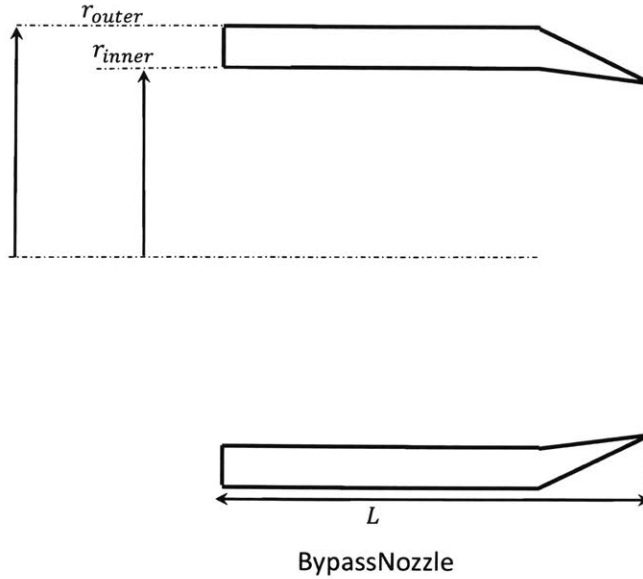


Figure 5.14: Simplified Bypass Nozzle Side View

central section). Using the same rationale as was used for the intake,

$$m_{intake} = \pi r_{inner}^3 \left(\frac{1}{w^2} - 1 \right) \rho_{aggregate} \quad (5.65)$$

In order to solve these equations one has to make assumptions on w and $\rho_{aggregate}$. These assumptions are given in the appendix and are based on typical values from [8] and [36].

Air-Oil Heat Exchanger (AOHEX) The primary function of the air-oil heat exchanger is to reduce the temperature of oil by transferring energy from the oil to the air flowing through it. The air-oil heat exchanger was modeled as a general countercurrent heat exchanger in which the cooled fluid and the cooling fluid are traveling in opposite directions. We examine two different modeling approaches to generate heat exchanger governing equations. The first approach describes heat transfer from first principles whereas the second is a more heuristic approach that allows quick computation of heat exchanger capacity for

different configurations of heat exchangers. We first discuss heat exchange computation from first principles using the log mean temperature difference (LMTD) [18].

In this approach the heat transfer per unit length \dot{q} between two fluids in concentric pipes is given by the following where T_A and T_B describe fluid bulk temperature whereas T_1 and T_2 represent wall temperatures as shown in figure 5.15. The k term represents the thermal conductivity of the heat exchanger wall material and h_1/h_2 represent the heat transfer coefficients between the walls and the fluid.

$$\dot{q} = \frac{2\pi k(T_A - T_B)}{\frac{k}{r_1 h_1} + \frac{k}{r_2 h_2} + \ln\left(\frac{r_2}{r_1}\right)} \quad (5.66)$$

We define an overall heat transfer coefficient h_0 such that:

$$\dot{q} = 2\pi r_2 h_0 (T_A - T_B) \quad (5.67)$$

$$\frac{1}{h_0} = \frac{r_2}{r_1 h_1} + \frac{r_2}{k} \ln\left(\frac{r_2}{r_1}\right) + \frac{1}{h_2} \quad (5.68)$$

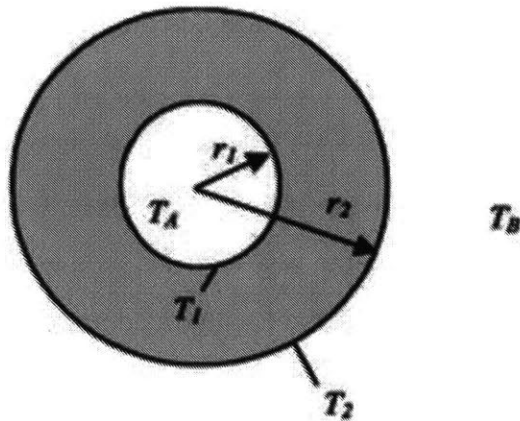


Figure 5.15: Heat transfer between two fluids in concentric pipes. Adapted from [18]

Having defined an overall heat transfer coefficient we now examine the counter flow heat exchanger shown in figure 5.16 where $T_{a1} > T_{a2}$ and $T_{b1} < T_{b2}$.

Conservation of Energy Globally

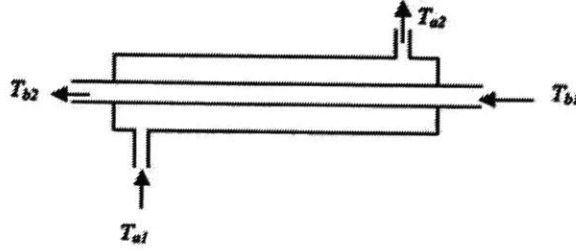


Figure 5.16: Counter flow heat exchanger. Adapted from [18]

Applying conservation of energy to the entire system:

$$\dot{Q} = \dot{m}_a c_{pa} (T_{a1} - T_{a2}) = \dot{m}_b c_{pb} (T_{b2} - T_{b1}) \quad (5.69)$$

Therefore:

$$\frac{\dot{Q}}{\dot{m}_a c_{pa}} = T_{a1} - T_{a2}, \quad \frac{\dot{Q}}{\dot{m}_b c_{pb}} = T_{b2} - T_{b1} \quad (5.70)$$

Conservation of Energy Locally

Applying conservation of energy locally:

$$-\dot{m}_a c_{pa} dT_a = -\dot{m}_b c_{pb} dT_b = \dot{q} dA = \dot{q} \pi D dx \quad (5.71)$$

Integrating from $x = 0$ to $x = L$ and simplifying:

$$\frac{T_{a2} - T_{b1}}{T_{a1} - T_{b2}} = e^{-a} \quad (5.72)$$

Where:

$$\alpha = h_0 A \left(\frac{1}{\dot{m}_a c p_a} - \frac{1}{\dot{m}_b c p_b} \right) \quad (5.73)$$

Substituting back into the equation from system level conservation of energy we find the output temperatures given input temperatures:

$$T_{b_2} = \frac{\dot{m}_a c p_a}{\dot{m}_b c p_b} \eta (T_{a_1} - T_{b_1}) + T_{b_1} \quad (5.74)$$

$$T_{a_2} = T_{a_1} (1 - \eta) + \eta T_{b_1} \quad (5.75)$$

$$\eta = \frac{1 - e^{-\alpha}}{1 - \frac{\dot{m}_a c p_a}{\dot{m}_b c p_b} e^{-\alpha}} \quad (5.76)$$

Note that $\alpha = f(h_0, A, \dots)$, and $\eta = f(\alpha, \dots)$. Given input temperatures, fluid properties and oil mass flow the output temperatures will depend on the mass flow of the cooling fluid \dot{m}_a , the overall heat exchange coefficient h_0 and the heat transfer area A . All else being equal, therefore, we can reduce the temperature of the hot fluid by increasing flow of the cooling fluid (for example by diverting more mass flow from a fan through the heat exchanger), or increase the heat exchange area A .

We now provide a highly simplified calculation of the stagnation pressure loss in the heat exchanger which is related to the wetted area of the heat exchanger. We do this by integrating skin friction experienced by the heat exchanger to compute a drag force on it and using the drag force to compute a stagnation pressure loss. Conceptually this represents a situation in which there are parallel cylinders with oil flowing through them situated such that the flow direction is parallel to their axis of rotational symmetry. Skin friction coefficient for a flat plate is given by the following for the turbulent case.

$$C_f = \frac{0.455}{(\log_{10} Re)^{2.58} (1 + 0.144 M^2)^{0.65}} \quad (5.77)$$

In all instances we place the heat exchanger inlet is downstream of the fan. Assuming a nominal length of 0.5 meters for the heat exchanger (based on sizing for 30,000lb thrust class engine), a Mach number of 0.3 and sea level standard day conditions we find that the Reynolds number is the following:

$$Re = \frac{\rho V l}{\mu} = \frac{1.225 \cdot \sqrt{\gamma R T} M \cdot 0.5}{1.81 \cdot 10^{-5}} \approx 10^6 \quad (5.78)$$

Which generates a skin friction coefficient $C_{f_{turbulent}} \approx 0.0034$. The drag then can be estimated by integrating the skin friction coefficient on the heat exchange area giving:

$$Drag = \frac{1}{2} \rho M^2 \gamma R T C_{f_{turbulent}} A_s \quad (5.79)$$

From conservation of momentum, assuming that the heat exchanger cross sectional area approximately equals the exit nozzle area of the heat exchanger $A_{AOHEX_{Nozzle}}$ we find that the stagnation pressure loss across the heat exchanger is.

$$\Delta p_0 = D / A_{AOHEX_{Nozzle}} \quad (5.80)$$

We now examine the Effectiveness-NTU Method. The advantage of this approach is that there is a wide literature that allows different heat exchanger configurations to be examined through simple changes in coefficients [89]. The Effectiveness-NTU method defines a heat exchanger effectiveness ε given by:

$$\varepsilon = \frac{\dot{Q}}{Q_{max}} = \frac{\text{Actual heat transfer rate}}{\text{Maximum possible heat transfer rate}} \quad (5.81)$$

Actual heat transfer rate is given by the following from conservation of energy:

$$\dot{Q} = \dot{m}_a c p_a (T_{a_{out}} - T_{a_{in}}) = \dot{m}_b c p_b (T_{b_{in}} - T_{b_{out}}) \quad (5.82)$$

$$\Delta T_{max} = T_{b_{in}} - T_{a_{in}} \quad (5.83)$$

$$Q_{max} = \min[\dot{m}_a c p_a, \dot{m}_b c p_b] (T_{b_{in}} - T_{a_{in}}) \quad (5.84)$$

For a counterflow heat exchanger the heat exchanger effectiveness is given by the following:

$$\varepsilon = \frac{1 - e^{-NTU(1-c)}}{1 - ce^{-NTU(1-c)}} \quad (5.85)$$

Where

$$NTU = \frac{UA_s}{(\dot{m}c_p)_{min}} \quad (5.86)$$

and

$$c = \frac{(\dot{m}c_p)_{min}}{(\dot{m}c_p)_{max}} \quad (5.87)$$

We assume that the heat exchanger has an effectiveness of 50 % which is typically the case for weight constrained applications. This gives us the following relationship for heat transfer:

$$\dot{Q} = \varepsilon \dot{Q}_{max} \quad (5.88)$$

$$T_{a_{out}} = \frac{\dot{Q}}{c p_a \dot{m}_a} + T_{a_{in}} \quad (5.89)$$

$$T_{b_{out}} = T_{b_{in}} - \frac{\dot{Q}}{c p_b \dot{m}_b} \quad (5.90)$$

We fix the temperature out $T_{b_{out}}$ based on the required temperature reduction by the heat exchanger. We also assume that we know \dot{m}_b (flow rate of fluid which we are cooling) and the overall heat transfer coefficient U . Rearranging the heat transfer effectiveness equation:

$$\varepsilon - \varepsilon c e^{[-NTU(1-c)]} = 1 - e^{[-NTU(1-c)]} \quad (5.91)$$

$$\varepsilon c e^{[-NTU(1-c)]} - e^{[-NTU(1-c)]} = \varepsilon - 1 \quad (5.92)$$

$$e^{[-NTU(1-c)]} = \frac{\varepsilon - 1}{c\varepsilon - 1} \quad (5.93)$$

$$NTU = \frac{1}{c-1} \ln \left(\frac{\varepsilon - 1}{c\varepsilon - 1} \right) \quad (5.94)$$

$$\frac{UA_s}{(\dot{m}c_p)_{min}} = \frac{1}{c-1} \ln \left(\frac{\varepsilon - 1}{c\varepsilon - 1} \right) \quad (5.95)$$

Finally we determine heat exchanger area based on thermal mass flows and the assumed heat exchanger effectiveness. Note that we know the mass flow of the hot fluid and find the mass flow of the cold fluid from conservation of energy. We then substitute into the equation below to find the area.

$$A_s = \frac{(\dot{m}c_p)_{min}}{U(c-1)} \ln \left(\frac{\varepsilon - 1}{c\varepsilon - 1} \right) \quad (5.96)$$

$$Drag = \frac{1}{2} \rho M^2 C_{f_{turbulent}} \gamma R T A_s \quad (5.97)$$

From conservation of momentum, assuming that the heat exchanger cross sectional area approximately equals the exit nozzle area of the heat exchanger $A_{AOHEX_{Nozzle}}$ we find that

the stagnation pressure loss across the heat exchanger is:

$$\Delta p_0 = D/A_{AOHEX_{Cross-Section}}, \quad (5.98)$$

where the area of the nozzle is determined based on the required mass flow.

Estimation of Mass

The estimation of mass of all cooling components was based on previous work on hybrid electric propulsion (STARC-ABL Turboelectric BLI Aircraft Concept) [90]. In that system the ratio of cooling capacity to mass was $0.68KW/kg$. For reference, assuming a 99.5% efficiency gearbox and the nominal case of 30,000hp being passed through the gearbox (22371 KW) which translates into a heat load of approximately 111 KW and a cooling system mass of approximately 164kg [88]. Thus we estimate heat exchanger size as:

$$m_{\text{heat exchanger}} = \frac{\dot{Q}}{\text{Cooling Per Kg}} \quad (5.99)$$

Oil Pump The primary function of the oil pump system is to maintain oil flow through the cooling and lubrication system. For this analysis the oil pump was treated as an oil flow source which provided fixed mass flow to the cooling system. The oil pump was assumed not to change the temperature of the oil flowing through it. The mass of the oil pump was taken into account implicitly through the cooling/mass ratio of the cooling system as a whole (part of heat exchanger estimate).

Gearbox The primary function of a gearbox is to transmit shaft power while changing angular velocity and torque. We build a highly simplified gearbox model which describes it as a device that accomplishes its function with finite losses described by an efficiency η .

Conservation of Energy

From conservation of energy and the steady state assumption the shaft power in is related to the shaft power out via the following:

$$Shaft Power_{out} = \eta Shaft Power_{in} \quad (5.100)$$

Note that from conservation of energy this will generate a finite amount of heat given by:

$$\dot{Q} = (1 - \eta) Shaft Power_{in} \quad (5.101)$$

This heat needs to be rejected in some fashion. We therefore include oil interfaces on the gearbox which facilitate heat rejection to the cooling and lubrication system. We now apply conservation of energy to the oil flowing through the gearbox.

$$\dot{m}_{oil}(T_{oil_{out}} - T_{oil_{in}}) = \dot{Q} \quad (5.102)$$

Estimation of Mass

We estimate the mass of the gearbox based on a range of power to weight ratios of planetary gearboxes for geared turbofan engines from the literature [88].

Electric Motor The primary function of an electrical motor is to convert electrical power into shaft power. In this case we build a simplified electrical motor model which describes it as a device which accomplishes its function with losses described by an efficiency η .

$$Shaft Power_{out} = \eta Electrical Power_{in} \quad (5.103)$$

Note that from conservation of energy this will generate a finite amount of heat given by:

$$\dot{Q} = (1 - \eta) \text{Shaft Power}_{in} \quad (5.104)$$

This heat will have to be rejected from the system to maintain steady state operation. One way of doing this would be to integrate cooling fins into the casing of the electrical motor (e.g. having an integrated heat sink). We model the heat sink as a collection of fins like the one shown in figures 5.17 and 5.18 based on the description in [91].

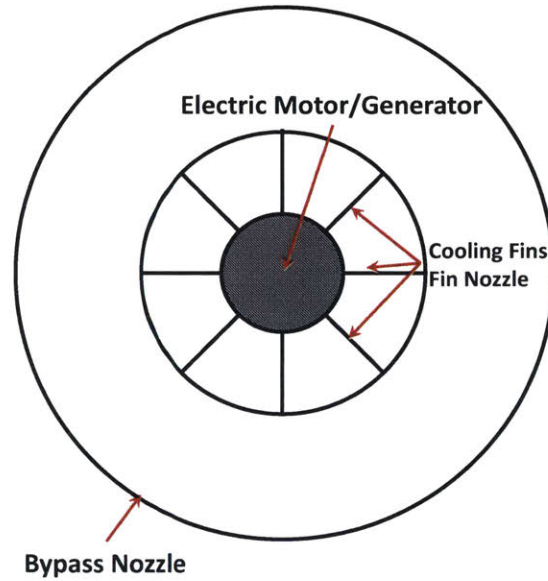


Figure 5.17: Electric Motor/Generator with Cooling Fins in Bypass Nozzle

We make the simplifying assumption that the temperature is only a function of x and not a function of y which is the case when the Biot number $\frac{hA_{cross-section}}{kPerimeter} \ll 1$.

Conservation of Energy

We can now apply conservation of energy for an infinitesimal slice of the fin dx where $A_{cross-section}$ is the cross-section of the fin normal to the x axis. A control volume view of

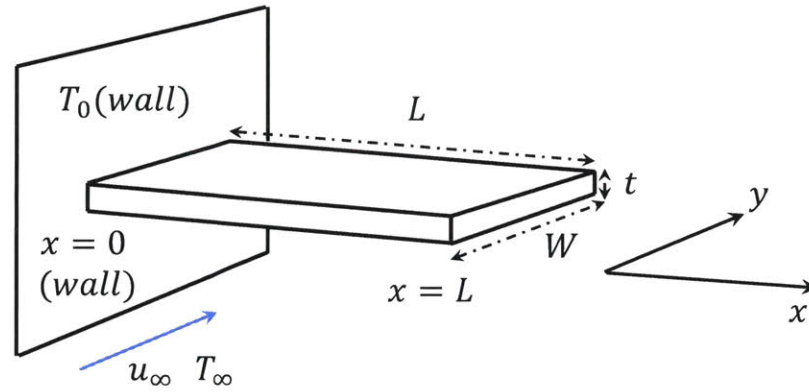


Figure 5.18: Cooling Fin

this situation is given in figure 5.19.

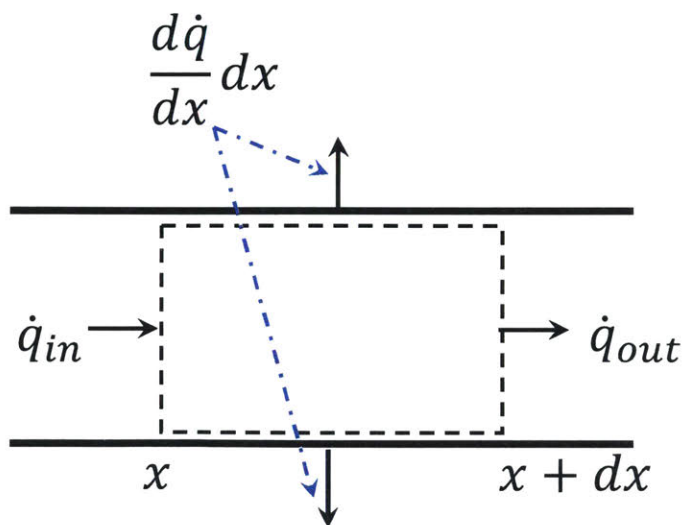


Figure 5.19: Control Volume for In

Applying conservation of energy to the control volume of length dx we find the following expression which relates heat transfer to the fluid $Perimeterh(T - T_\infty)dx$ to the net heat flux out of the control volume $A\frac{dq}{dx}dx$.

$$A_{cross-section}\frac{dq}{dx} + Ph(T - T_\infty) = 0 \quad (5.105)$$

Which is equivalent to:

$$\frac{d^2T}{dx^2}(T - T_\infty) - \frac{Perimeterh}{A_{cross-section}}(T - T_\infty) = 0 \quad (5.106)$$

Which is a second order ODE to which we must apply to boundary conditions. The first is that the temperature of the fin at the wall equals that of the wall:

$$(T - T_\infty)_{x=0} = (T_0 - T_\infty)_{x=0} \quad (5.107)$$

The second boundary condition is that heat transfer at the tip of the fin is small.

$$\frac{d}{dx}(T - T_\infty)_{x=L} = 0 \quad (5.108)$$

Integrating along the fin we find an expression of the overall heat transfer from the fin.

$$\dot{Q} = \tanh(mL)\sqrt{kA_{cross-section}Perimeter}(T_0 - T_\infty) \quad (5.109)$$

Where:

$$m = \sqrt{\frac{hPerimeter}{kA_{cross-section}}} \quad (5.110)$$

Substituting back into the heat flow equation:

$$\dot{Q} = \tanh \left(\sqrt{\frac{h \text{Perimeter}}{k A_{\text{cross-section}}}} L \right) \sqrt{k A_{\text{cross-section}} \text{Perimeter}} (T_0 - T_\infty) \quad (5.111)$$

Assuming n independent fins and a thickness t and width w .

$$\dot{Q} = n \cdot \tanh \left(\sqrt{\frac{h(2t + 2w)}{ktw}} L \right) \sqrt{ktw(2t + 2w)} (T_0 - T_\infty) \quad (5.112)$$

Thus if we know fin geometry, heat transfer coefficient, free stream flow temperature target wall temperature we can determine the number of fins n and therefore the area A_s . The heat transfer coefficient h can be estimated using the Reynolds analogy as follows,

$$h \approx \rho_\infty c_p u_\infty \frac{C_f}{2}, \quad (5.113)$$

Where the skin friction coefficient is given by the following for a turbulent boundary layer:

$$C_f = \frac{0.455}{(\log_{10} Re)^{2.58} (1 + 0.144 M^2)^{0.65}} \quad (5.114)$$

$$\text{Drag} = \frac{1}{2} \rho M^2 C_{f_{\text{turbulent}}} \sqrt{\gamma R T} A_s \quad (5.115)$$

From conservation of momentum, assuming the heat exchanger cross sectional area equals the exit nozzle area of the heat exchanger $A_{\text{MotorNozzle}}$, the stagnation pressure loss across the heat exchanger is:

$$\Delta p_0 = D / A_{\text{MotorNozzle}} \quad (5.116)$$

Electrical motors can also be cooled via oil flow in a similar fashion to gearboxes. We elect

to treat cooling of gearboxes, electrical motors and electrical generators in the same manner (e.g. oil cooling with air-oil heat exchangers) for this analysis.

Estimation of Mass

The estimation of mass of electric motors was based on previous work on hybrid electric propulsion [90]. In that system the ratio of motor power to mass was (*8hp/pound*). In SI units this is $13.2KW/kg$. For reference, assuming a nominal case of 30,000hp being passed through the motor (which would be the case for a 30,000lb thrust class engine) we find that the electric motor would have a mass of 3750lb.

Electric Generator The primary function of an electrical generator is to convert shaft power into electrical power. It accomplishes its function with losses described by an efficiency η .

Conservation of Energy

The conservation of energy relations for the electric generator are similar to that of the gearbox and electrical motor:

$$Electrical\ Power_{out} = \eta Shaft\ Power_{in} \quad (5.117)$$

$$\dot{Q} = (1 - \eta) Shaft\ Power_{in} \quad (5.118)$$

As was the case with the gearbox, we assume oil cooling (refer to gearbox governing equations).

Estimation of Mass

The mass of the electrical generator was also calculated based on previous work on hybrid electric propulsion [90]. We assume that the electrical generator has an integrated inverter

making the power to weight ratio of the system approximately $4.5hp/lb$ ($7.407KW/kg$).

Additional Components (Future Work) In order to be able to model transient operations of a larger set of solar-hybrid-electric propulsion systems we will need to add components which enable energy storage and harvesting without the use of fossil fuels. The following are some examples of such components: electrical batteries (e.g. Li-Ion), solar photovoltaics, hydrogen fuel cells etc.

5.2.2 Abstraction Level Two: Reconfigurable Engine Model Modelica

Having established the underlying mathematics of models at the deepest level of abstraction we now briefly describe the Modelica models that were used for simulation at abstraction level two. The primary difference from the previous section's description is that none of the turbomachinery components in Modelica were modeled using maps from [16].

Interfaces Each interface in Modelica defines the variables that describe flows to or from that interface (see figure 5.20). For example the fuel flow interface shown in figure 5.20a is described by mass flow rate and temperature. Mass flow rate is defined as a *flow* variable, which indicates to the solver that conservation of mass must apply to that variable. The thrust interface shown in figure 5.20b is treated as a translational mechanical interface, meaning that it is described by force (also a flow variable) and displacement. The gas interface is described by temperature, pressure and mass flow (a flow variable) as shown in figure 5.20c. The oil interface shown in figure 5.20d is treated in the same manner as the fuel interface. Finally the shaft power and electrical interfaces are depicted in figures 5.20 e and 5.20f.

These six interfaces were used to define all possible connections between components/subsystems at modeling levels 1 and 2.

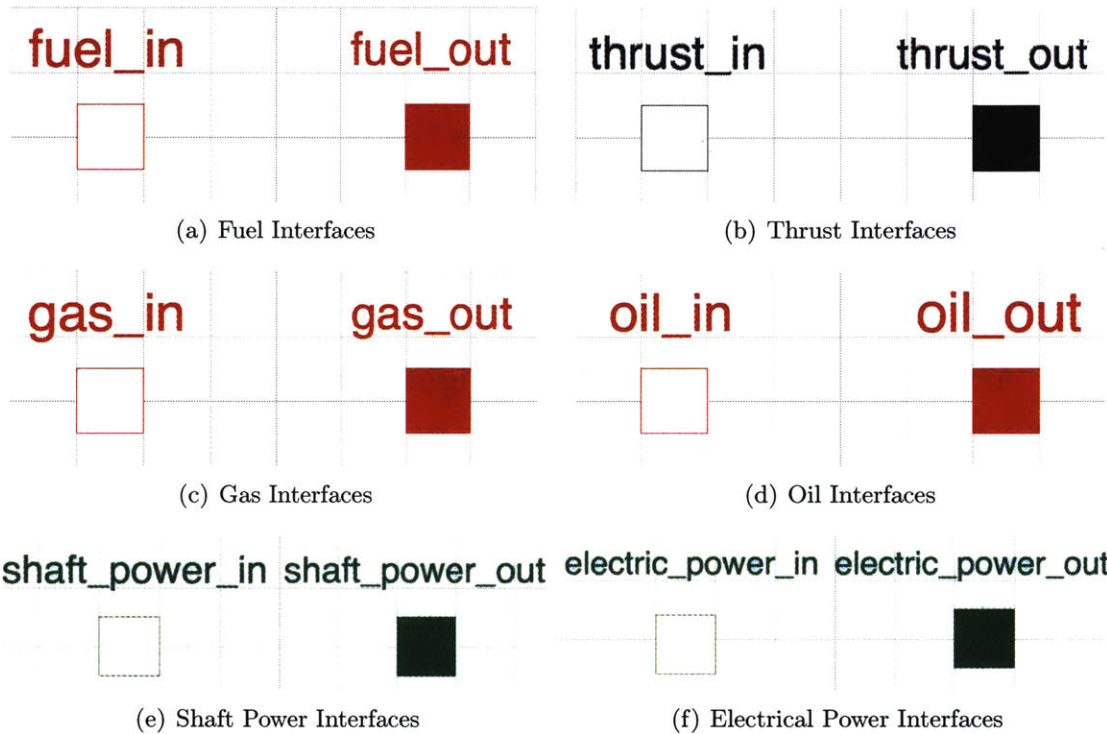


Figure 5.20: Fuel, thrust, gas, oil, shaft power and electrical power interface Modelica models.

Atmosphere Source Requirement We first define components that represent connections to the aircraft and environment. These were treated as requirements in the methodology section. The first component that we describe is the atmospheric source component which provides flow to components connecting to the atmosphere and whose gas interfaces face upstream. The equations of the atmospheric source component are based on the International Standard Atmosphere and are given below.

$$T_{0_{out}} = (T_{ref} + T_{gradient}Altitude) \left(1 + \frac{\gamma - 1}{2} M^2 \right) \quad Altitude \leq 11000meters \quad (5.119)$$

$$T_{0_{out}} = 216.7 \left(1 + \frac{\gamma - 1}{2} M^2 \right) \quad \text{Altitude} > 11000 \text{meters} \quad (5.120)$$

$$P_0 = P_{ref} e^{-\frac{g \text{Altitude}}{(RT_{ref})}} \left(1 + \frac{\gamma - 1}{2} M^2 \right)^{\frac{\gamma}{\gamma - 1}} \quad (5.121)$$

A diagram of the Modelica atmospheric source requirement component is given in figure 5.21 which depicts its interfaces.

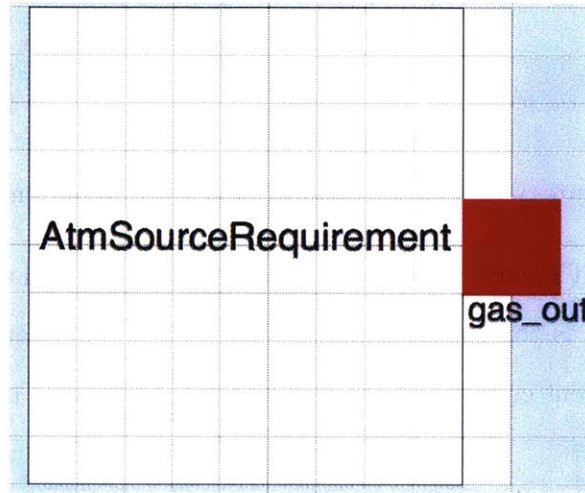


Figure 5.21: Level 2 Atmospheric Source Requirement

Atmosphere Sink Requirement The atmospheric sink component is used to provide a static pressure boundary condition for gas interfaces connecting to the atmosphere which discharge flow in downstream. The governing equations of the atmospheric sink component are based on the International Standard Atmosphere.

$$P = P_{ref} e^{-\frac{g \text{Altitude}}{(RT_{ref})}} \quad (5.122)$$

A diagram of the Modelica atmospheric sink requirement component is given in figure 5.22 which depicts its interfaces.

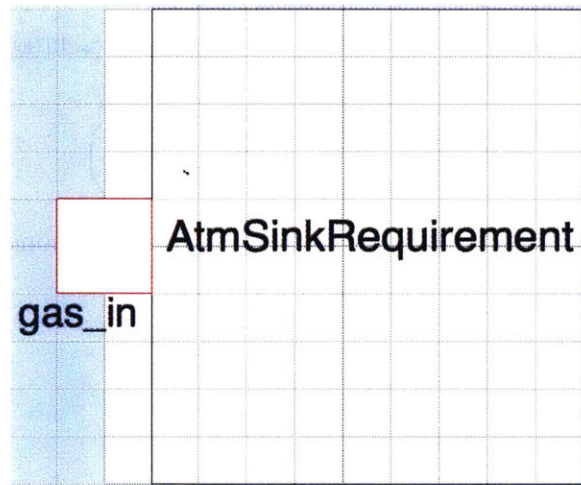


Figure 5.22: Level 2 Atmospheric Sink Requirement

Fuel Requirement The fuel requirement component was treated as a source of fuel which provided as much fuel as the engine demanded at a fixed temperature. A diagram of the Modelica fuel requirement component is given in figure 5.23 which depicts its interfaces.



Figure 5.23: Level 2 Fuel Requirement

Thrust Requirement The thrust requirement component was treated as a sink of thrust which consumed thrust from all thrust providing components in a given engine architecture.

A diagram of the Modelica thrust requirement component is given in figure 5.24 which depicts its interfaces.

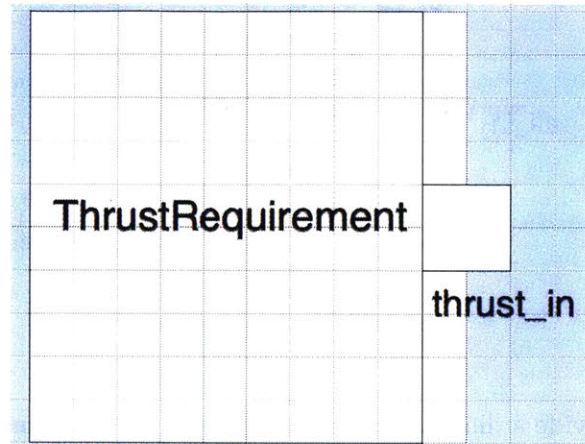


Figure 5.24: Level 2 Thrust Requirement

Fan The Modelica model for generic fan component described in the previous section for completeness in figure 5.25. The primary differences of the level two fan description from the one described in the level three, is that fan pressure ratio, isentropic efficiency and mass flow was calculated for a single operating point rather than being outputted by a fan map from [16]. Shaft power was treated as an energy flow rather than being decomposed into torque and angular velocity.

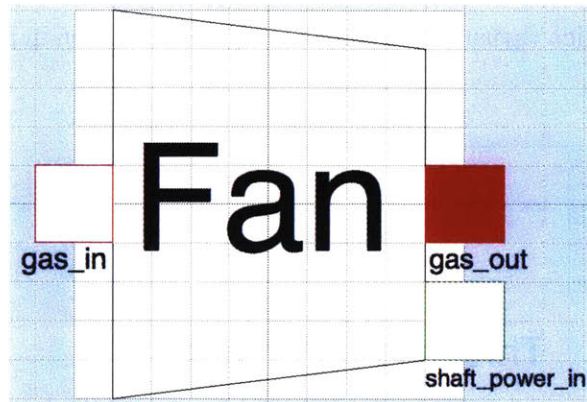


Figure 5.25: Level 2 Fan

Compressor We provide a diagram of the Modelica model for generic compressor component described in the previous section for completeness in figure 5.26. The primary difference of the level 2 compressor component from the one described in the level 3 description, is that compressor pressure ratio, isentropic efficiency and mass flow was calculated for a single operating point rather than being outputted by a compressor map from [16]. In addition only shaft power was treated as an energy flow rather than being decomposed into torque and angular velocity.

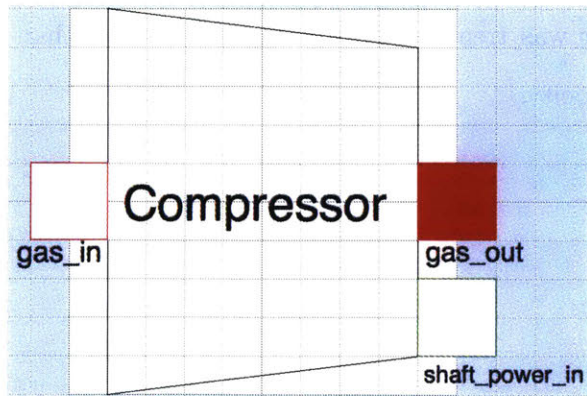


Figure 5.26: Level 2 Compressor

Burner For completeness, we provide a diagram of the Modelica model for generic compressor component described in the previous section for completeness in figure 5.27. The underlying mathematics for the burner component as it was modeled in Modelica are identical to those described in level 3.

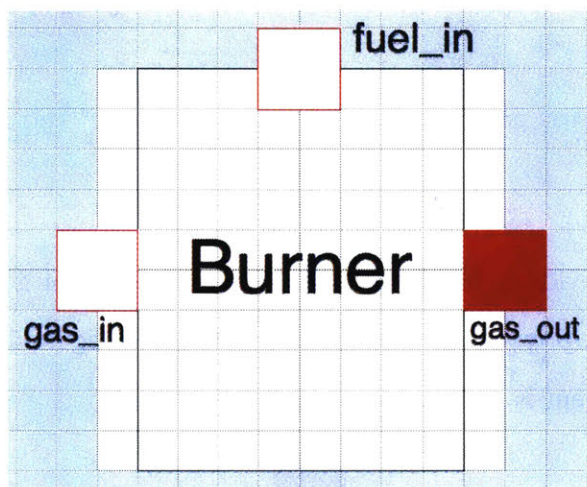


Figure 5.27: Level 2 Burner

Turbine We provide a diagram of the Modelica model for generic turbine component described in the previous section for completeness in figure 5.28. The primary differences of the level two turbine description from the one described in the level three, is that turbine pressure ratio, isentropic efficiency and mass flow was calculated for a single operating point rather than being outputted by a turbine map from [16]. Shaft power was treated as an energy flow rather than decomposed into torque and angular velocity.

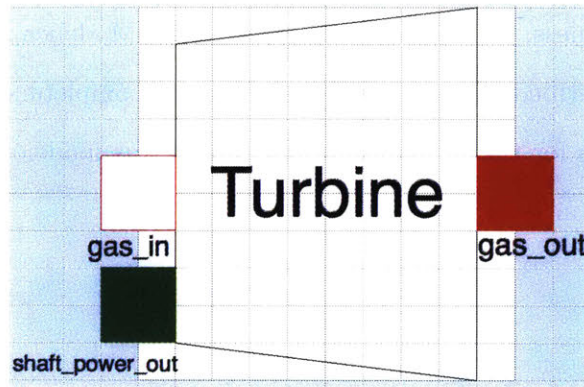


Figure 5.28: Level 2 Turbine

Core Nozzle We provide a diagram of the Modelica model for generic core nozzle component described in the previous section for completeness in figure 5.29. The description of the core nozzle component is identical to that at level three.

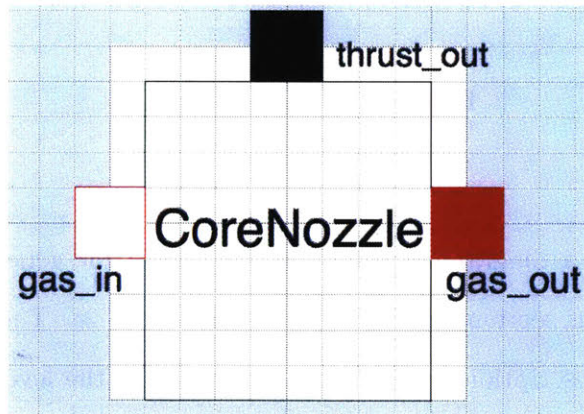


Figure 5.29: Level 2 Core Nozzle

Bypass Nozzle We provide a diagram of the Modelica model for generic bypass nozzle component described in the previous section for completeness in figure 5.30. The underlying mathematics for the bypass nozzle component as it was modeled in Modelica are identical to those described in level three.

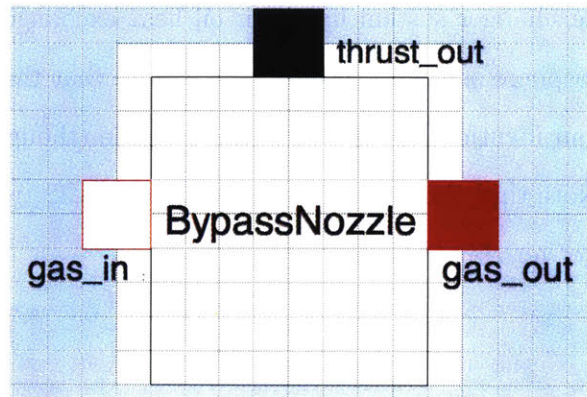


Figure 5.30: Level 2 Bypass Nozzle

Oil Pump We provide a diagram of the Modelica model for generic oil pump component described in the previous section for completeness in figure 5.31. The underlying mathematics for the oil pump component as it was modeled in Modelica are identical to those described in level three.

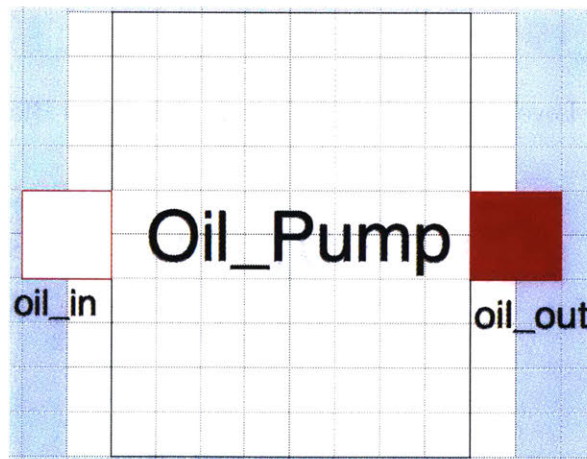


Figure 5.31: Level 2 Oil Pump

Air-Oil Heat Exchanger The air-oil heat exchanger was modeled in Modelica using the underlying governing equations described in level three. The Modelica model of the air-oil heat exchanger is shown graphically in figures 5.32 and 5.33. The primary addition here

is that the air-oil heat exchanger system has a air-oil heat exchanger nozzle with which it interfaces with the atmosphere as shown in figure 5.33. Note that the air-oil heat exchanger nozzle was modeled in an identical way to the bypass nozzle (though the areas of the two nozzles are different). Note that the thrust interface in figure 5.32 transfers the thrust from the air-oil heat exchanger nozzle to the airframe.

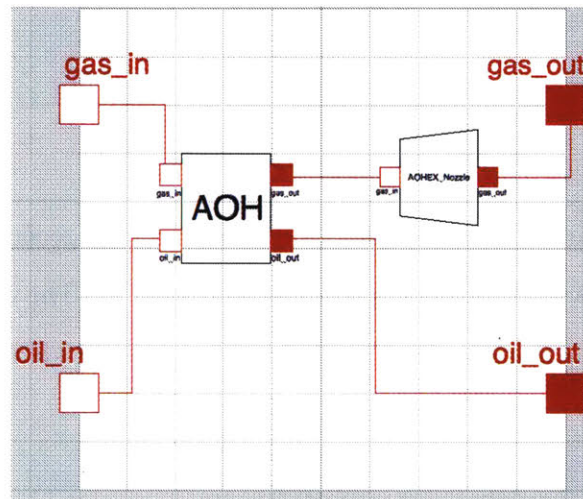


Figure 5.32: Level 2 Air-Oil Heat Exchanger Internal Block Diagram

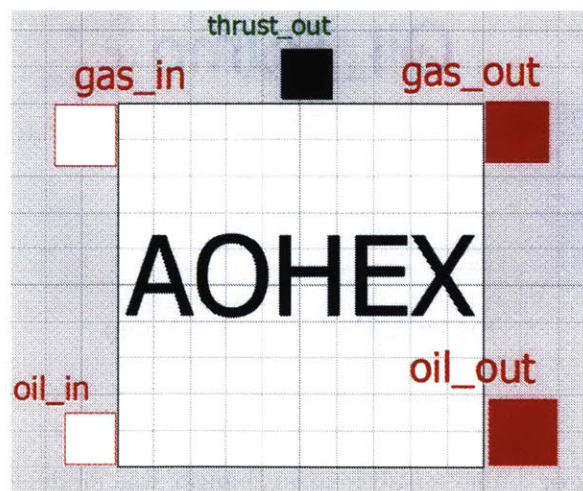


Figure 5.33: Level 2 Air-Oil Heat Exchanger

Gearbox The governing equations of the gearbox model in Modelica at level two are the same as those described in the previous section. The Modelica model of the gearbox is graphically depicted in figure 5.34.

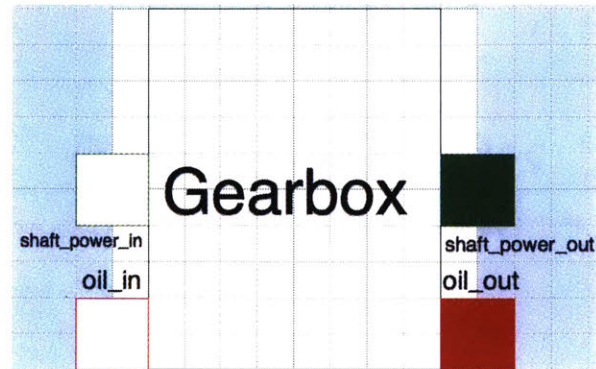


Figure 5.34: Level 2 Gearbox

Electrical Generator The governing equations of the electrical generator model in Modelica at level two are the same as those described in the previous section. The Modelica model of the electrical generator is graphically depicted in figure 5.35.

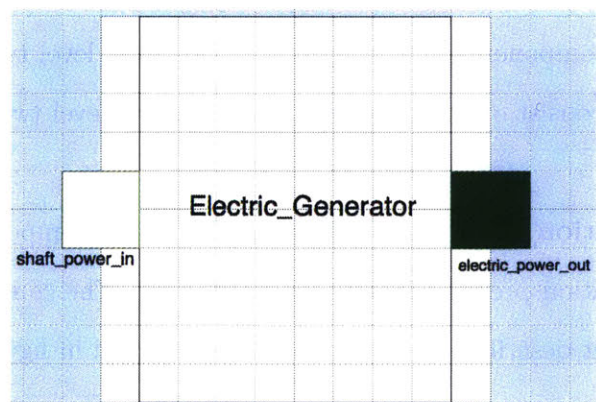


Figure 5.35: Level 2 Electric Generator

Electric Motor The governing equations of the electrical motor model in Modelica at level two are the same as those described in the previous section. The Modelica model of the electrical motor is graphically depicted in figure 5.36.

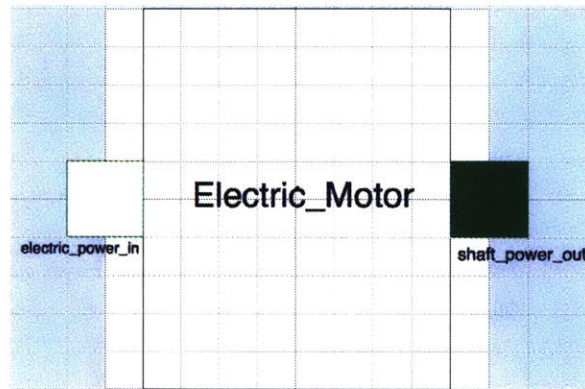


Figure 5.36: Level 2 Electric Generator

5.2.3 Abstraction Level One: Reconfigurable Engine Model

We now discuss the top-most level of abstraction (level one). For brevity we exclude the requirements components which are the same regardless of level of abstraction. All components at level one are represented by assemblies of components at level two. In some cases the representation of components at level one is the same as level two, however, in general level one components consist of ensembles of components at level two.

Shaft Power Stagnation Pressure Increasing The first component described at level one is a generic stagnation pressure increasing component. The component is composed of only the fan component described in level two and is depicted in figures 5.37 and 5.38.

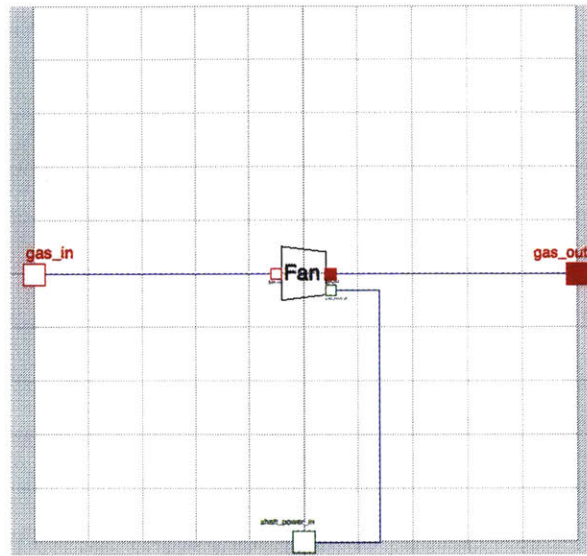


Figure 5.37: Level 1 Shaft Power Stagnation Pressure Increasing Internal Block Diagram

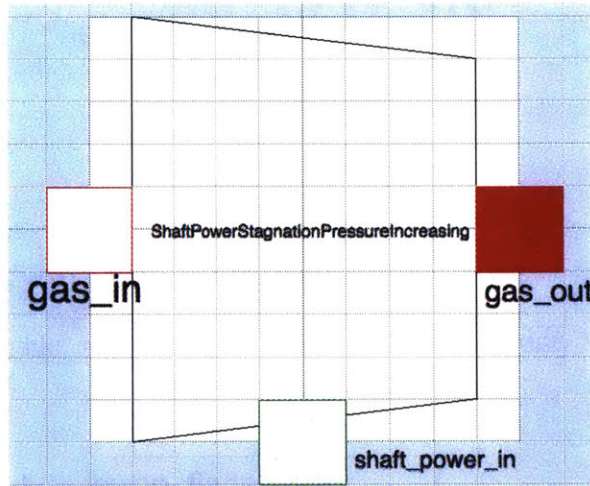


Figure 5.38: Level 1 Shaft Power Stagnation Pressure Increasing

Brayton Shaft Power Providing The Brayton shaft power providing component represents the Brayton thermodynamic cycle. The component is composed of the compressor, turbine and burner components described in level two and is graphically depicted in figures 5.39 and 5.40.

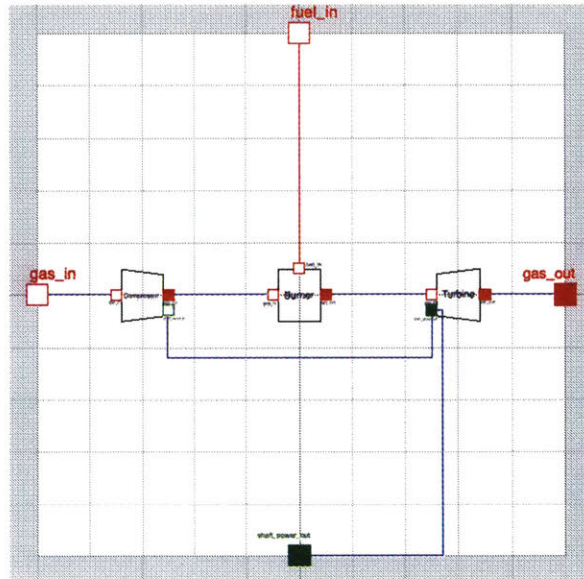


Figure 5.39: Level 1 Brayton Shaft Power Providing Increasing Internal Block Diagram

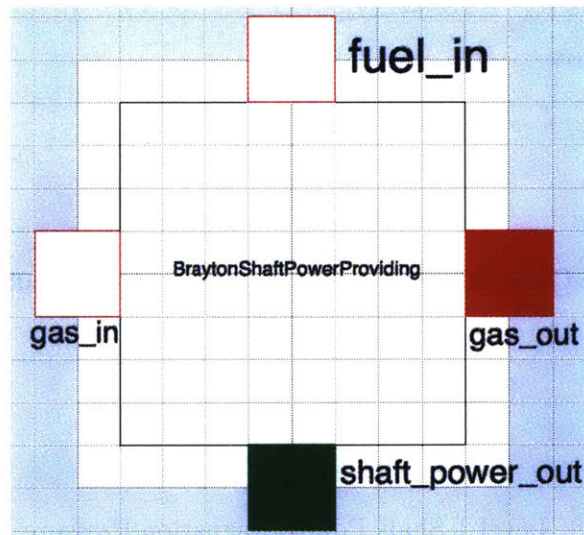


Figure 5.40: Level 1 Brayton Shaft Power Providing Increasing

Cold Flow Expanding The primary function of the cold flow expanding component is to expand the flow to ambient conditions (trade potential energy for kinetic energy) and thereby produce thrust. The cold flow expanding component was modeled with only the bypass nozzle component at level two as shown in figures 5.41 and 5.42.

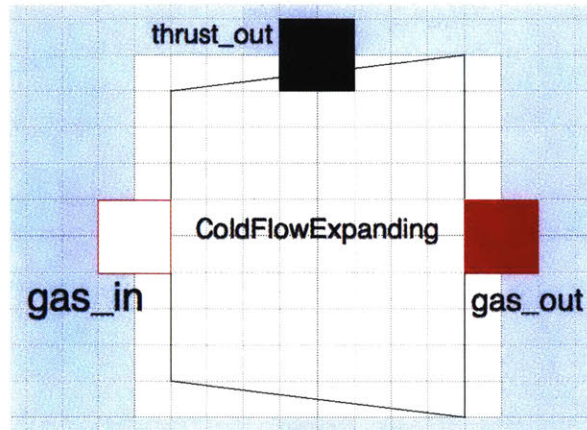


Figure 5.41: Level 1 Cold Flow Expanding

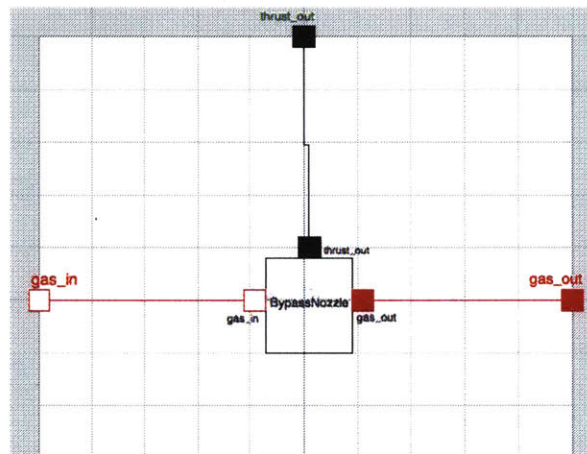


Figure 5.42: Level 1 Cold Flow Expanding Internal Block Diagram

Hot Flow Expanding The hot flow expanding component was modeled in a similar manner to the cold flow expanding component. The primary difference is that the core nozzle was used rather than the bypass nozzle from level one as shown in figures 5.43 and 5.44.

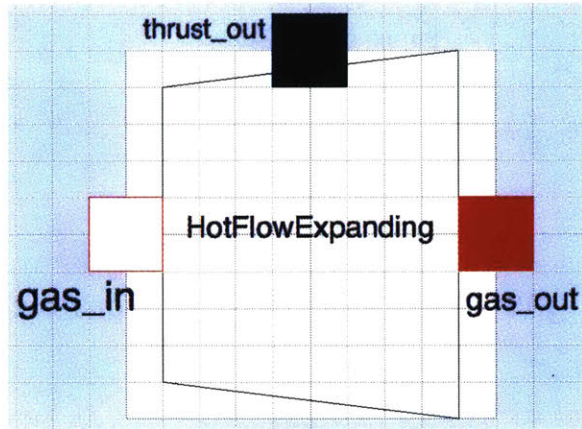


Figure 5.43: Level 1 Hot Flow Expanding

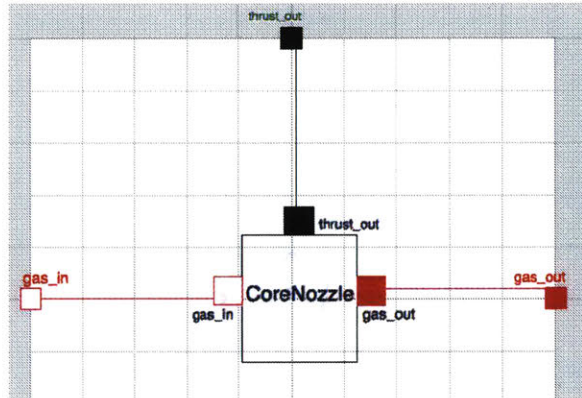


Figure 5.44: Level 1 Hot Flow Expanding Internal Block Diagram

Mechanical Shaft Power Transferring with Air/Oil Cooling The mechanical shaft power transferring component at level one was created via an ensemble of the gearbox and cooling system components in level two and is depicted in figures 5.45 and 5.46.

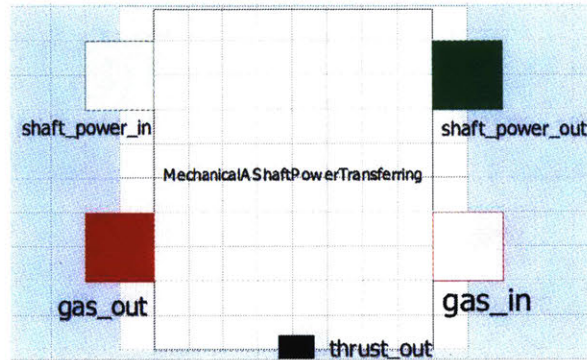


Figure 5.45: Level 1 Mechanical Shaft Power Transferring with Air Cooling

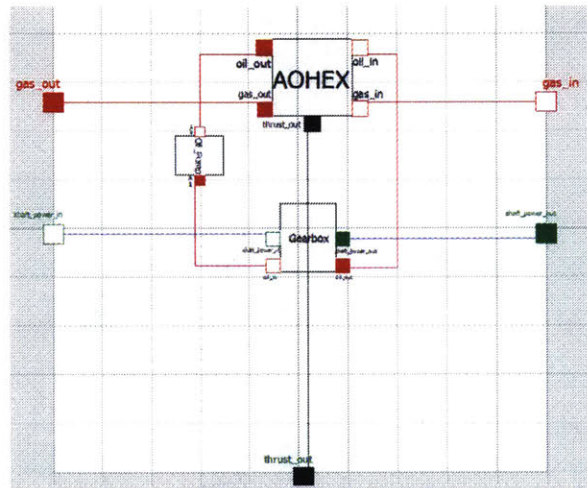


Figure 5.46: Level 1 Mechanical Shaft Power Transferring with Air/Oil Cooling Internal Block Diagram

Electrical Shaft Power Transferring with Air/Oil Cooling The electrical shaft power transferring component at level one was treated in a similar manner to the mechanical shaft power transferring component. The electrical shaft power transferring component is represented by a motor and generator along with their supporting cooling systems as depicted in figures 5.47 and 5.48.

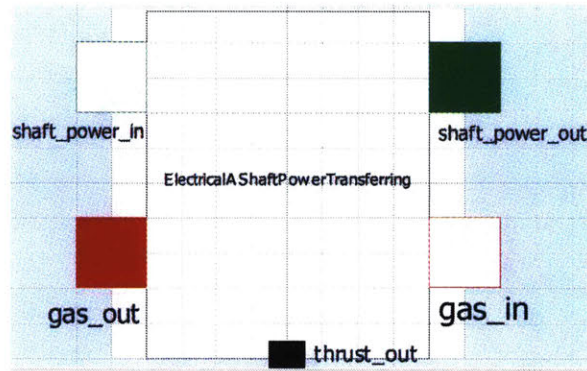


Figure 5.47: Level 1 Electrical Shaft Power Transferring

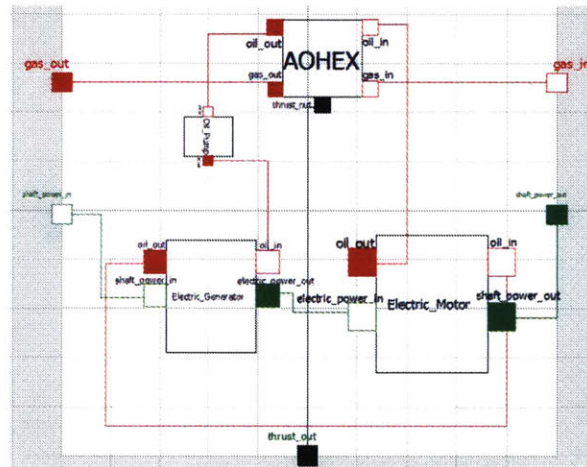


Figure 5.48: Level 1 Electrical Shaft Power Transferring with Air/Oil Cooling Internal Block Diagram

5.2.4 Model Validation

There were three primary requirements which were taken into consideration for the models created in this work. The first requirement was that the models be simple enough to run in novel configurations on a desktop machine with little or no intervention from the user. The second was that the models predict performance (fuel consumption in this case), with sufficient accuracy compared to test data of existing engines, so that broad categories of architectures can be ranked correctly. To satisfy the first requirement, the level of fidelity

chosen for the models was as “as simple as possible” but not any simpler. The models were

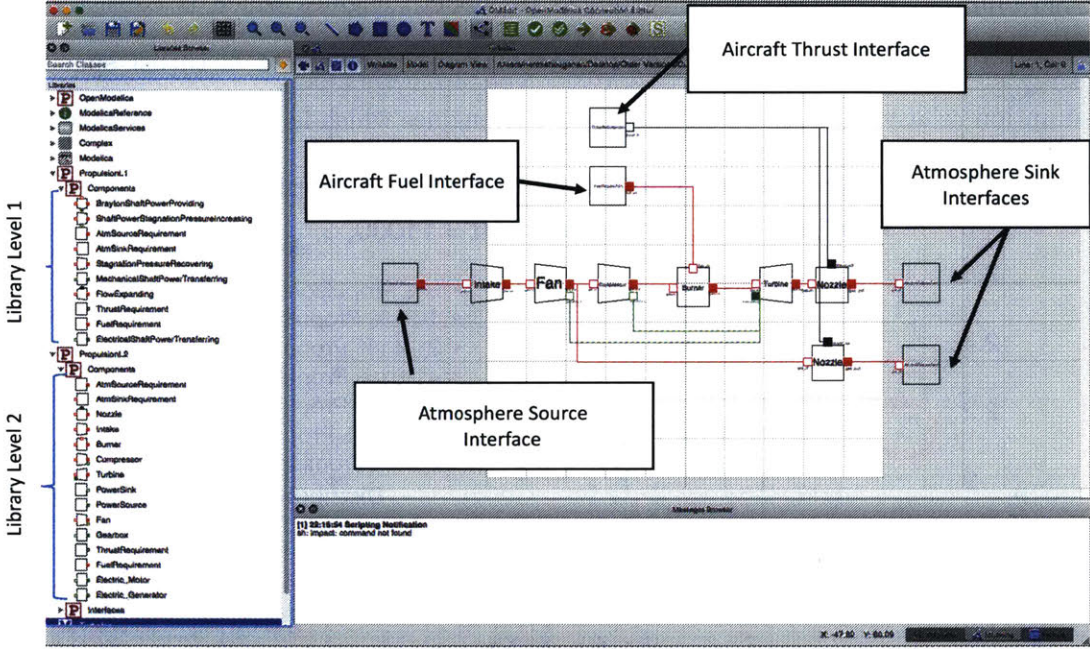


Figure 5.49: Validation of level 2 model against GE90-85B. Agreement in fuel consumption was approximately 10%.

validated against the GE90-85B engine from [8]. It was found that fuel consumption was within approximately 10% of the measured value. Since we are considering major architectural changes in which changes in performance as well as complexity are significant this was deemed adequate given previous changes in performance with architecture discussed in the introduction. Figure 5.49 shows the GE90-85B model. Model parameters were assigned values based on the dataset from [8].

Figure 5.50 depicts uninstalled thrust specific fuel consumption normalized relative to a generic geared turbofan architecture for a number of generic engine architectures that will be discussed in detail later. The geared turbofan in this case is limited to a bypass ratio of

12 (based on current designs [29]) and the direct drive turbofan is limited to 9 (based on typical values from [8]). We see on the order of a factor of two improvement in uninstalled fuel consumption when comparing turbojets to modern engines which is consistent with the literature [8]. We also see on the order of 5-10 % improvement in thrust specific fuel consumption relative to current high bypass ratio engines which is also consistent with [29].

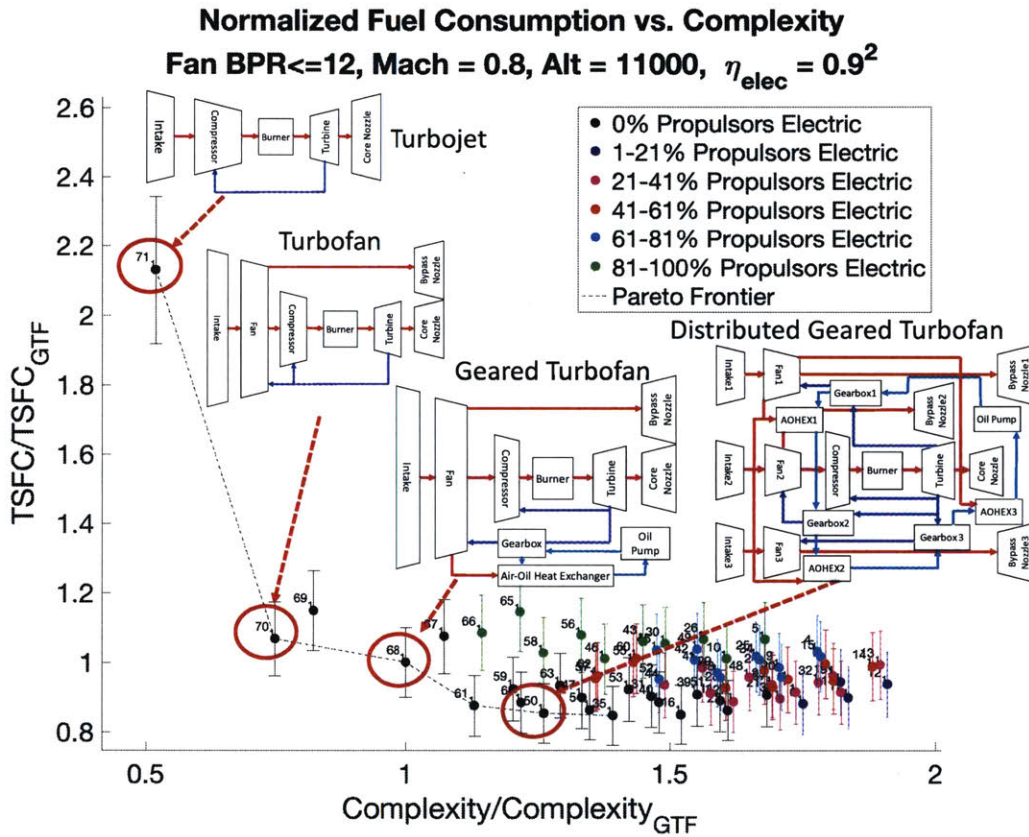


Figure 5.50: Normalized Thrust Specific Fuel Consumption vs. Normalized Complexity

Parameters in the the engine mass model were manually tuned using five relatively modern turbofan engines from [8]. These were the Rolls-Royce Trent 1000, General Electric GE90-85B, Engine Alliance GP7200, Pratt & Whitney PW4098 and Rolls-Royce Trent 875. The percentage root-mean squared error of the engine mass model was 12.9%. The data is given below:

Engine	Mass Estimate	True Mass	Percentage Error
Rolls-Royce Trent 1000	6680 <i>kg</i>	5409 <i>kg</i>	23.5%
General Electric GE90-85B	8881 <i>kg</i>	7825 <i>kg</i>	13.5%
Engine Alliance GP7200	6090 <i>kg</i>	6085 <i>kg</i>	0.1%
Pratt & Whitney PW4098	8243 <i>kg</i>	7484 <i>kg</i>	10.1%
Rolls-Royce Trent 875	5954 <i>kg</i>	5942 <i>kg</i>	0.2%

Table 5.2: Geometric and efficiency parameters based primarily on [8].

It is important to note, however, that engine mass is a function of detailed design and should in general be investigated at the preliminary design phase. In this work, assumptions were made on non-dimensional geometric parameters which allowed some inferences to be made about some aspects of engine preliminary design. This enabled engine mass estimates that were closely related to the underlying physics driving component volumes, stage count etc. The reason that this approach was chosen, rather than employing linear regression on the historical dataset from [8] is that it is more generalizable to radically new concepts.

5.3 Architecture Generation

We now examine generated engine architectures given a library of intakes, fans, core nozzles, bypass nozzles, compressors, burners, turbines, gearboxes, electric motors, electric generators, heat exchangers and oil pumps. The scope of this analysis was limited to aircraft engines in isolation. Thus values for thrust specific fuel consumption (fuel flow/thrust) were computed for all architectures for engines in an uninstalled state. There are a number of reasons for this limitation in scope. Perhaps the most important one is that with unconventional engine configurations aircraft configuration is likely to change and be co-optimized with propulsion architecture in order to take advantage of boundary layer ingestion opportunities along with other system integration benefits. Each unconventional engine architecture will perform differently depending on the configuration of aircraft and its positioning on

the aircraft. This work presents a baseline performance which can be used in future work with novel aircraft configurations. Performance results can relatively easily be augmented by changing inlet conditions, representing local flow conditions at different locations on the aircraft. This work therefore provides a broad catalog of engine architectures that can be drawn on during conceptual design of aircraft which no longer necessarily represent the traditional tube and low-wing architecture [28].

This case study was conducted at two levels of abstraction. The primary purpose of the first level of abstraction was to rapidly create a set of concepts for consideration and check their feasibility through simulation. The second purpose of the analysis at level one was to eliminate concepts from consideration due to excessive complexity or other factors based on the subject matter expertise. The second layer of abstraction employed both heuristic and gradient based engine optimization for minimum uninstalled fuel consumption. A detailed examination of distributed and hybrid electric architectures was carried out at level two including investigation of tradeoffs between fuel consumption and complexity as well as fuel consumption and engine mass.

We first examine the feasibility of architectures at abstraction layer one. As was discussed in the previous chapter, the first steps of the architecture generation process involve the generation of the DS2M. The DS2M without convex hull constraints is given in figure 5.51. The DS2M applying convex hull constraints (eliminating pairwise infeasible connections) is given in figure 5.52. The number of possible connections is reduced by 87 % after applying convex hull constraints.

Since the size of the design space is proportional to 2^n where n represents the number of design decisions (e.g. the presence or absence of a component or connection) this exponentially reduces the size of the search problem. The next step is to generate potential component

combinations subject to additional rules and then search for feasible sets of connections between components. The maximum number of “BraytonShaftPowerProviding” components (e.g. engine cores) was limited to one. The maximum number of “ShaftPowerStagnation-PressureIncreasing” components (e.g. fans) was limited to seven for this analysis to bound the size of the problem. In addition the maximum number of flows to each interface was in general limited to one. Only those architectures where there were no unconnected interfaces were deemed to be feasible. For example, if a “ShaftPowerStagnationPressureIncreasing” component in an architecture did not have a source of shaft power, the architecture would be deemed infeasible ².

²Note that there are multiple “AtmSinkRequirement” components in the DS2M. Each of these components represents a downstream connection to the atmosphere. As such it imposes a pressure boundary condition (AtmSinkRequirement pressure is equal to local atmospheric pressure). The reason for having a different AtmSinkRequirement for each flow to the atmosphere is related to the way that Modelica treats situations where multiple flows converge to a single interface. All flows in Modelica consist of “potential” variables (like voltage, temperature or pressure) that must be the same at any continuously connected part of the network and “flow” variables which obey Kirchoff’s current law (conservation of mass). If two flows were connected to the same AtmSinkRequirement block in Modelica, the underlying OpenModelica interpreter would set their pressures as well as their temperatures to be the same. There is no physical reason for temperature from disparate sources to be the same as it is ejected into the atmosphere. For this reason the pressure boundary condition for each flow into the atmosphere was set with separate “AtmSinkRequirement” components.

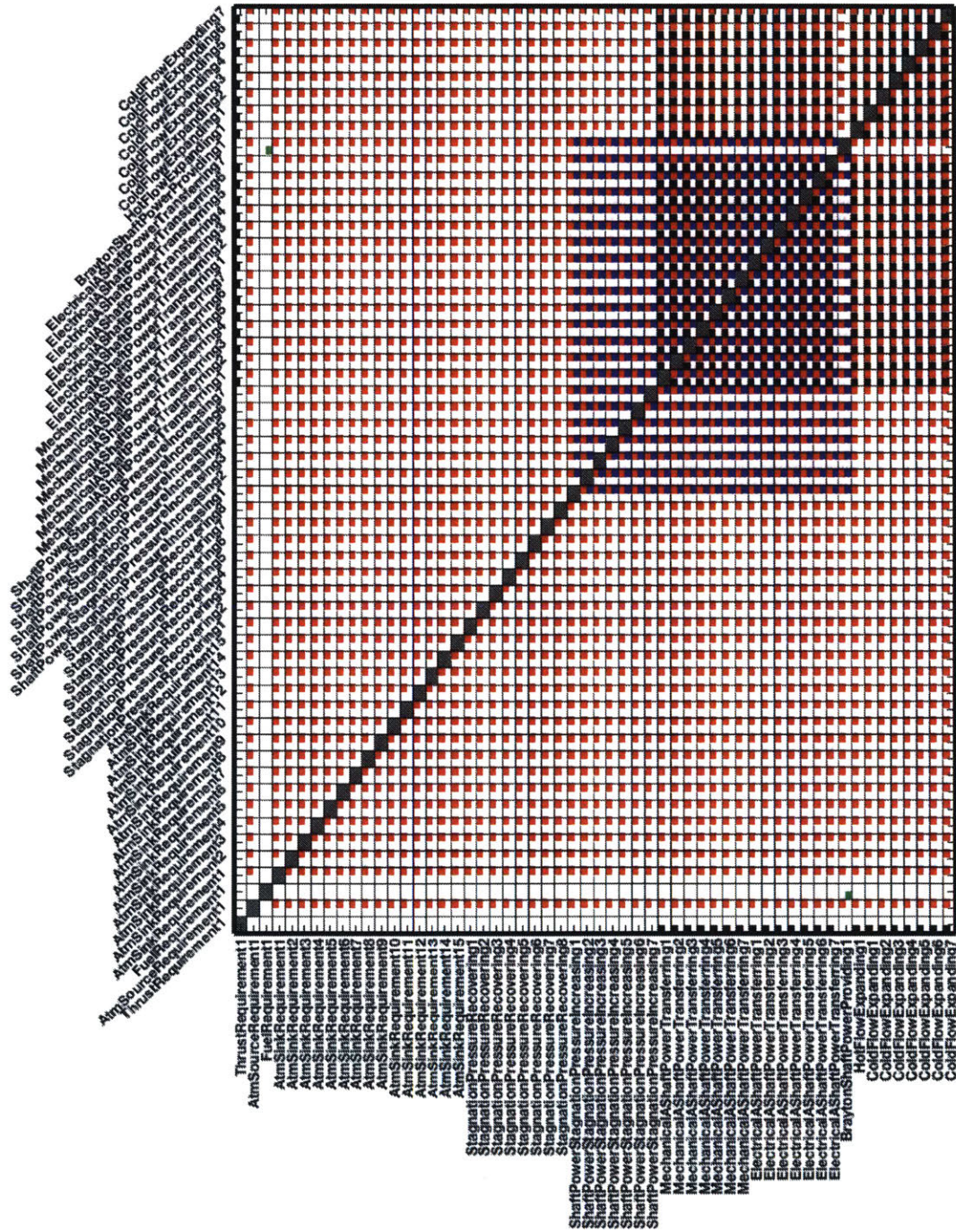


Figure 5.51: Abstraction Layer 1 Engine DS2M. Gas connections: **red**, shaft power connections: **blue**, mechanical connections: **black**, fuel connections: **green**, oil connections: **cyan**, electrical connections: **magenta**.

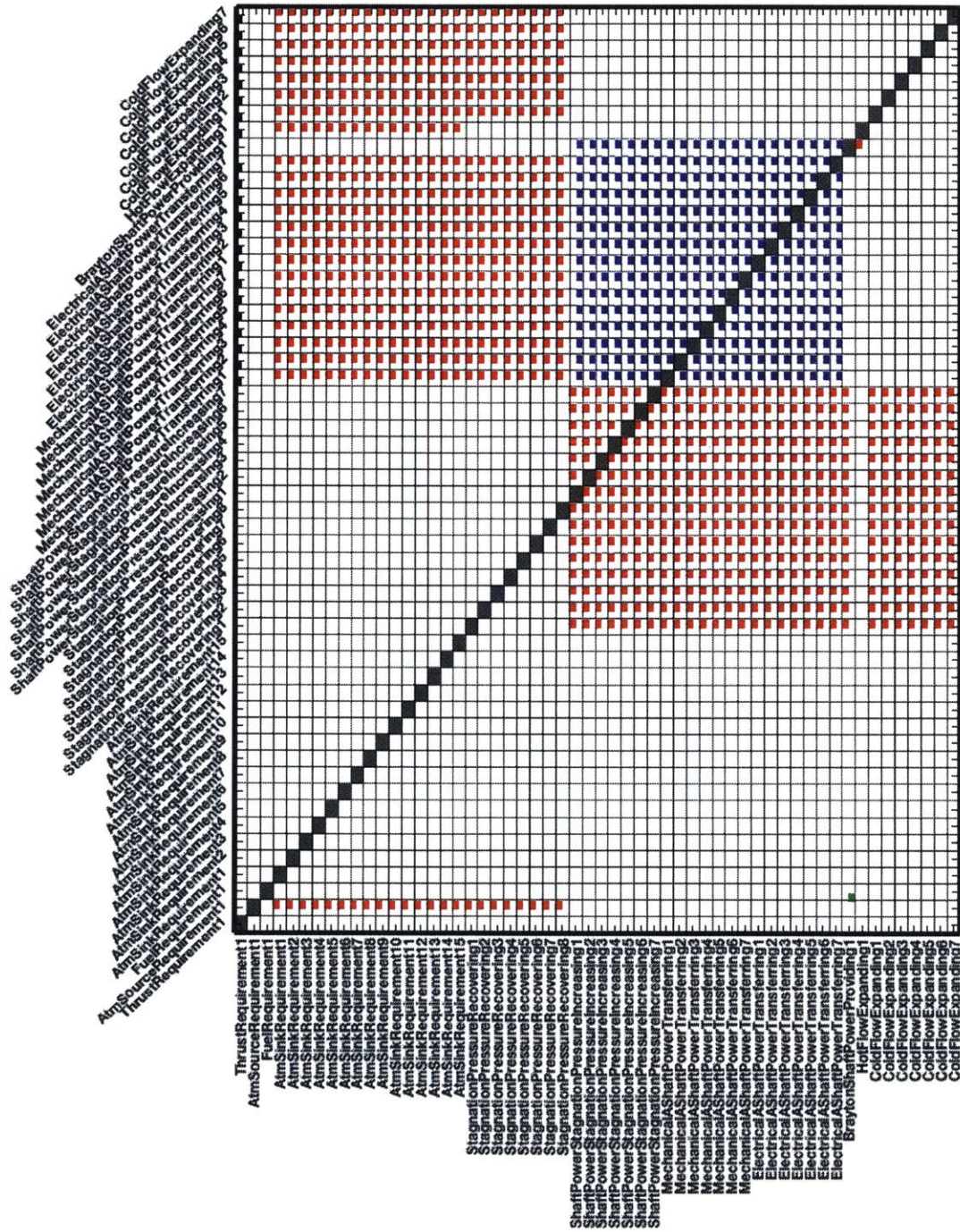


Figure 5.52: Abstraction Layer 1 Engine DS2M Filtered With Convex Hulls (87% reduction). Gas connections: **red**, shaft power connections: **blue**, mechanical connections: **black**, fuel connections: **green**, oil connections: **cyan**, electrical connections: **magenta**.

A total of 127 architectures were generated. These included the well known turbojet, turbofan and geared turbofan architectures which represent the majority of civil aircraft engines in [8]. In addition, hybrid electric architectures in which power is transferred electrically from the “BraytonShaftPowerProviding” component (e.g. engine core) to the “ShaftPowerStagnationPressureIncreasing” components (e.g. fan) were generated consisting of up to seven fans situated in parallel. In order to carry out an additional feasibility check, a model of each of these architectures was automatically synthesized and simulated in OpenModelica as shown in figure 5.53. The primary objective of this step was to check via simulation that the generated architectures were feasible (represented a set of simultaneous equations that had a solution). Since the purpose of the analysis carried out at abstraction layer one served to check potential feasibility of architectures and not to conduct detailed comparison of fuel consumption, no optimization was carried out.

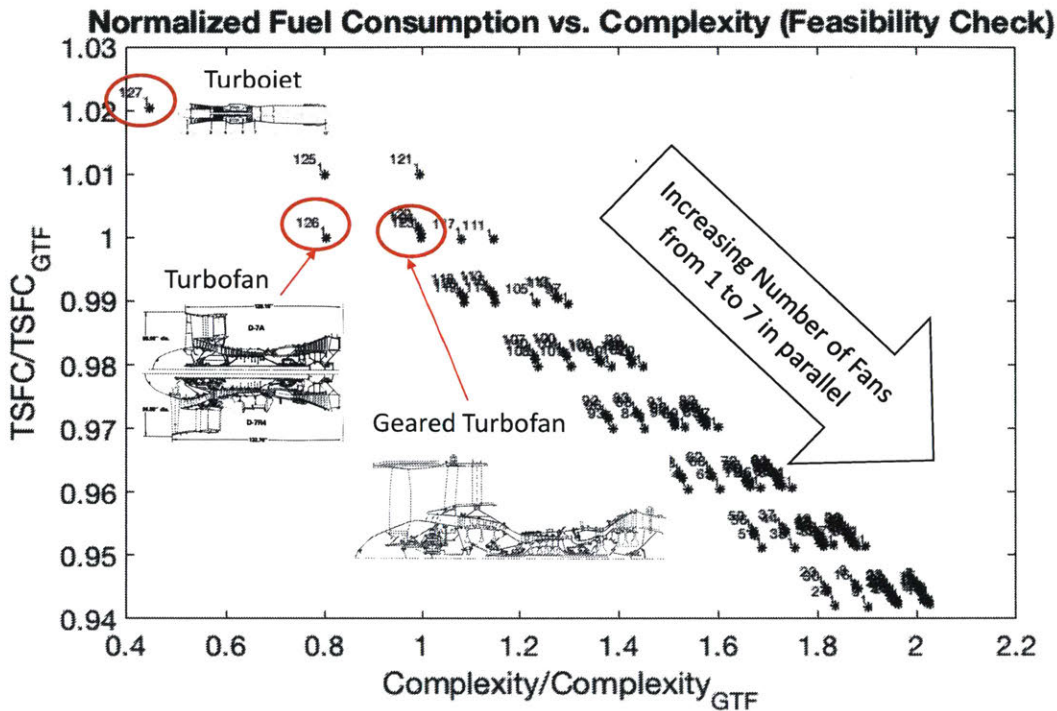


Figure 5.53: Normalized Thrust Specific Fuel Consumption vs. Normalized Complexity. Note that components were sized in a manner that made all architectures ranging from turbojets to 7 fan distributed hybrid electric turbofans mathematically feasible.

To ensure that all potentially feasible architectures converged to a physical solution, pressure ratios of “ShaftPowerStagnationPressureIncreasing” components (e.g. fans) were set to 1.05 (typical fan pressure ratios for turbofans are on the order of 1.5). This was done to ensure both single fan architectures and highly distributed seven fan architectures could be powered by the same “BraytonShaftPowerProviding” component. As can be seen from figure 5.53 all 127 architectures did indeed converge to physical solutions. It can also be seen that fuel consumption is reduced with increasing number of fans due to propulsive efficiency benefits. This is consistent with physical intuition and historical data. Figure 5.53 also shows an increase in complexity with the number of fans in an architecture. The top left hand cluster of architectures in figure 5.53 represents single fan architectures, whereas the bottom right

hand corner represents seven fan distributed architectures. Having established the potential feasibility of all of the generated architectures, we now move on to the second more detailed abstraction layer. In view of the increased complexity associated with highly distributed architectures consisting of 6 or more fans and available computational resources, we limit the number of fans in the second stage of our analysis to five.

Figure 5.54 portrays the DS2M prior to applying convex hull constraints. The color convention used for populating the DS2M is given within the description of each figure. The primary reasons for not strictly adhering to the color convention described in the methodology section is that the number of types of flows is greater than four and using a single color for all mass flows would make distinguishing between oil, gas and fuel mass flows difficult. The DS2M represents all possible components and connections given a library of components. Every architecture that can exist in this problem formulation is composed of a subset of the components and connections in the DS2M. Notice that the number of instances of components in the DS2M is bounded. This was done to keep the problem tractable while still allowing for a wide range of architectures to be generated. There can be up to 6 intakes, up to 5 fans, up to 5 electric motors, up to one generator and a single instance of each of the Brayton cycle core components (compressor, burner, turbine, core nozzle). The total number of possible connections in the DS2M is 6016. Each one of these connections is treated as a binary design variable (value of 0 representing a connection that does not exist and one representing a connection that does). Thus the unconstrained space of possible connections in the DS2M is 2^{6016} .

We now apply the convex hull constraints described in the methodology section. The number of possible connections is reduced from 6016 to 680 meaning that the size of the search space is reduced by a factor of 2^{5336} as shown in figure 5.55. The reason that the number of connections is larger than that depicted explicitly in the DS2M is that each component has

multiple unique interfaces and therefore each entry in the DS2M can represent different pairs of interfaces being connected between the same components. We then apply backtracking search to find feasible component combinations subject to user defined rules. For each feasible component combination we then apply backtracking search to find which set of connections from the DS2M satisfies all constraints. Most constraints simply state the minimum and maximum number of flows to and from each interface. For this problem the minimum number of flows was always one for all interfaces, meaning that an architecture would only be feasible if there were no interfaces with no connection to them (e.g. if an air-oil heat exchanger is not receiving oil or air, or if the air/oil going into it is not leaving it). In addition, the maximum number of connections was bounded at each interface. In general the number of connections was limited to one, but in some cases, the fan being one example, it was limited to five, to facilitate transfer of gas to heat exchangers. In addition, a constraint stating that there could be no cooling oil flow loops without the presence of exactly one oil pump were applied.

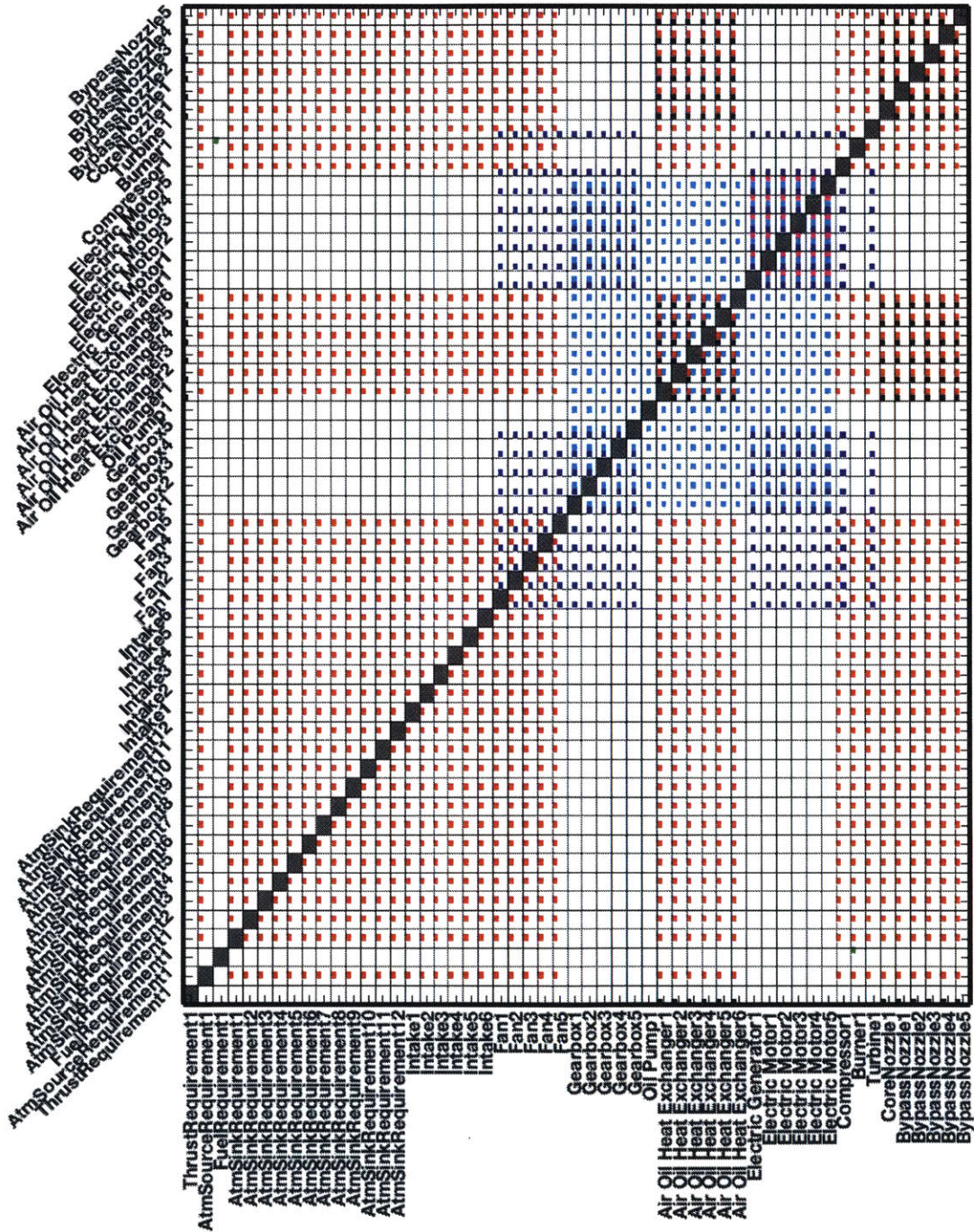


Figure 5.54: Abstraction Layer 2: Engine DS2M. Gas connections: **red**, shaft power connections: **blue**, mechanical connections: **black**, fuel connections: **green**, oil connections: **cyan**, electrical connections: **magenta**.

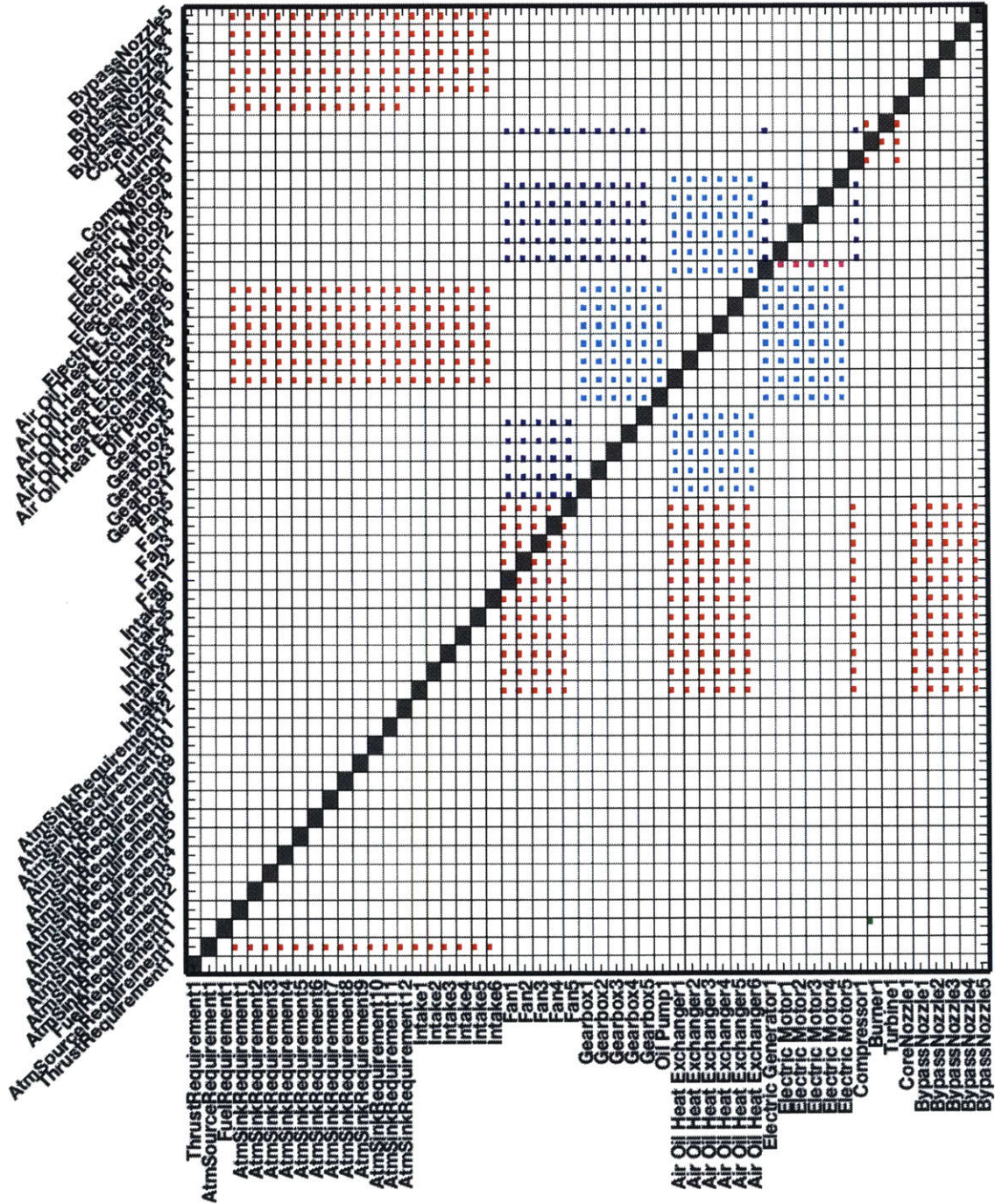


Figure 5.55: Abstraction Layer 2: Engine DS2M Filtered With Convex Hulls (88.7% reduction). Gas connections: red, shaft power connections: blue, mechanical connections: black, fuel connections: green, oil connections: cyan, electrical connections: magenta.

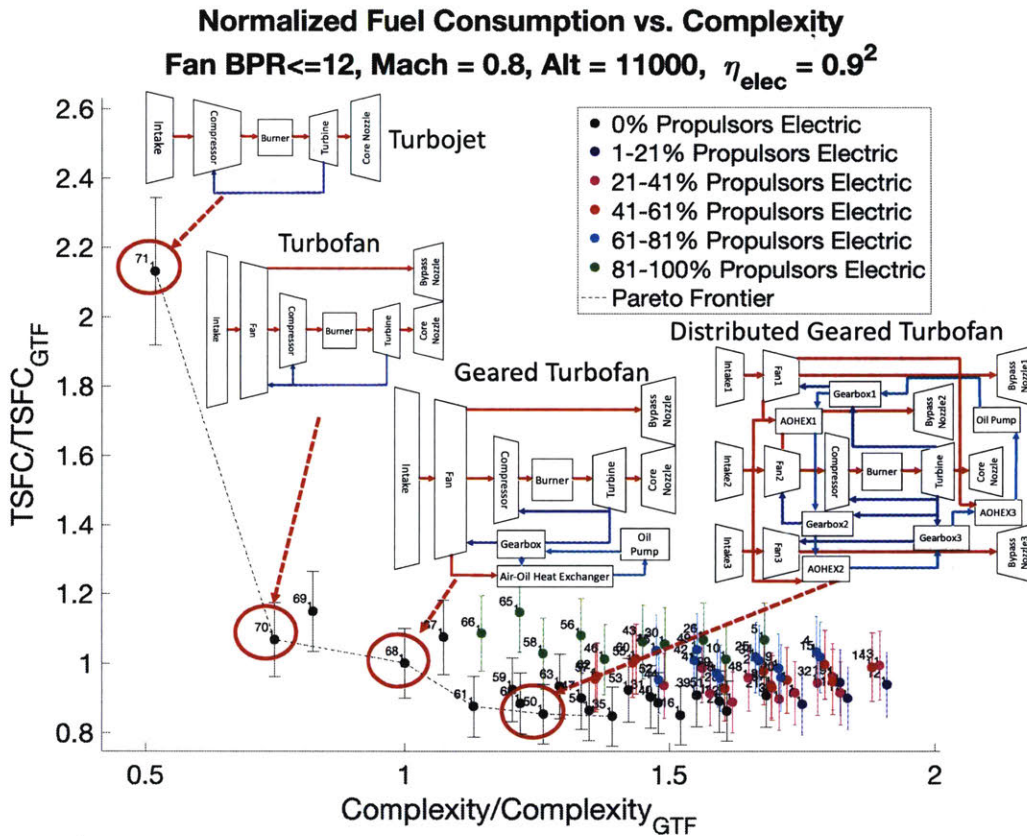


Figure 5.56: Normalized Thrust Specific Fuel Consumption vs. Normalized Complexity

Figure 5.56 depicts performance-complexity tradespace for the resulting 71 aircraft engine architectures from architecture exploration. Figure 5.70 depicts the same analysis relaxing bypass ratio constraints for geared/electric fans to 24 representing future ultra-high bypass ratio engines. All engine complexities and fuel consumptions are normalized to a geared turbofan architecture. The chart was generated by optimizing each architecture for minimum fuel consumption at Mach number 0.8 and an altitude of 11000 meters. In this case it was assumed that the electrical efficiency of all motors and electrical generators was 90%. Oil temperatures in all gearboxes, electrical motors and electrical generators were constrained to 120 Celsius. Bypass ratio constraints for figure 5.56 reflect what is typically achieved for geared/direct drive architectures [8, 29]. It was assumed that geared/electrically driven fans

could have a bypass ratio of up to 12 relative to the core mass flow. Direct drive fans were limited to a bypass ratio of nine relative to the core. Figure 5.70 depicts the same analysis but with a relaxed bypass ratio constraint for geared/electric fans (up to 24) representing future ultra-high bypass ratio engines. Isentropic efficiencies were assumed to be 0.9 throughout the turbomachinery (based on typical value from performance map data from [16]) and overall pressure ratio of the engine was assumed to be 50 (based on typical values from [8]) in cases where there was a fan stage in front of the core and 35 when there was not (assuming a fan pressure ratio of approximately 1.4 based on [8] and removing it from the stagnation pressure increasing process).

The design variables used in the optimization were pressure ratios and mass flows through all propulsors (fans) as well as mass flows of air and oil through all heat exchangers. To reduce the number of design variables the ratio of thermal mass flow (heat capacity multiplied by mass flow) of air and oil was fixed for all heat exchangers.

Due to the variety of different architectures and the nonlinear nature of the thermodynamics of the engine, selecting an initial guess for gradient based optimization that ensured model/optimization convergence was non-trivial. Each architecture was then optimized via genetic algorithm [92]. The design vector from the genetic algorithm was then used as an initial guess for gradient-based optimization.

The results presented here represent a sample of feasible architectures rather than the entire feasible set. We have included one feasible set of connections for every set of feasible components. Each feasible component combination can have more than one feasible set of connections. Only one set of possible connections was found for each set of feasible components since it provided a broad range of architectures (set of feasible architectures can be generated, evaluated and optimized on a desktop computer within hours).

In figure 5.56 going from left to right we see that *major existing architectures have been generated*, starting from a turbojet in the top left hand corner (architecture 71), a turbofan (architecture 70), a geared turbofan at the knee of the curve (architecture 68) and a distributed geared turbofan similar to the one proposed for the Cambridge-MIT silent aircraft initiative (architecture 50) [93]. These architectures are shown in detail via their Design Structure Matrices and block diagrams in the figures 5.57, 5.58 (turbojet), 5.59, 5.60 (turbofan), 5.61,5.62 (geared turbofan), 5.63, 5.64 (distributed geared turbofan).

In addition to the existing and proposed architectures, we have generated and optimized a broad range of other concepts, including distributed (more than one fan), distributed turboelectric (fans driven by electrically) and distributed partially turboelectric (some fans driven mechanically and some electrically) architectures. While individual points in this space have been discussed in the literature, as a group, these architectures have not gained much attention in the literature.

The color coding of each of the points in figure 5.56 represents the fraction of propulsors (fans) which are driven electrically. Architectures shown in green represent engine configurations in which 100% of fans are electrically driven, whereas those in black represent configurations for which there is no electric component. Improvements in performance are achieved via propulsive efficiency (greater mass flow and lower pressure ratio fans). The reason for the banded structure is the fact that transfer losses in electrical components are much higher than those in mechanical shaft power interfaces. Electrical transfer of power results in approximately a 20% loss ($1 - 0.9^2$ if the motor and generator have an efficiency of 90%) whereas mechanical transfer of power results in 0.5% loss when a gearbox is present. Thus the greater the fraction of electrical fans the greater the losses in power transfer from the turbine to the fans. The relationship between engine performance and transfer losses is somewhat more complicated since transfer losses are removed from the system in the form

of heat via heat exchangers which require air flow from the bypass streams and have finite stagnation pressure losses themselves along with a finite thrust.

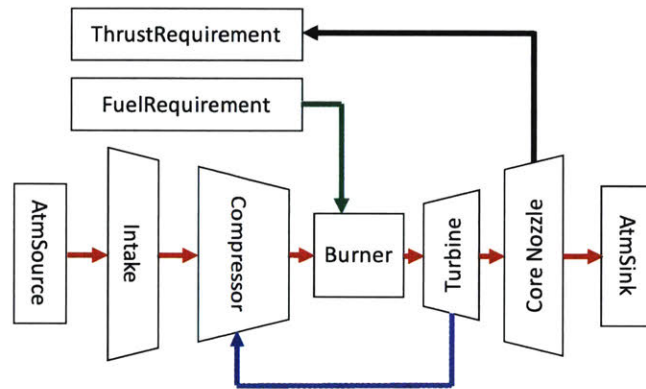


Figure 5.57: Generic Turbojet Architecture. Gas connections: **red**, shaft power connections: **blue**, mechanical connections: **black**, fuel connections: **green**, oil connections: **cyan**, electrical connections: **magenta**.

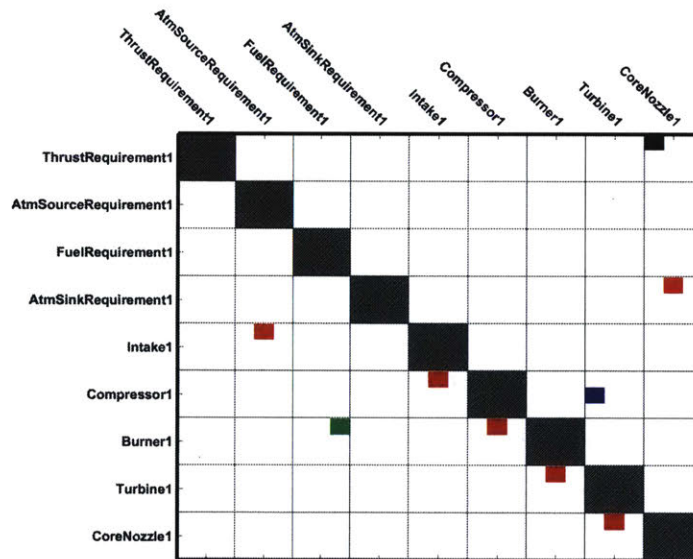


Figure 5.58: Generic Turbojet Architecture DSM. Gas connections: **red**, shaft power connections: **blue**, mechanical connections: **black**, fuel connections: **green**, oil connections: **cyan**, electrical connections: **magenta**.

Figures 5.57, 5.58 depict a generic turbojet architecture. Due to the relatively small number

of components the requirements components are also shown here. The model is simplified such that only the thrust requirement and nozzle components have mechanical interfaces. For more complex engines we will from now on only depict the architectures themselves without the requirements blocks. There is only one shaft power connection present (turbine to compressor) and one gas path traveling from the intake through the core of the engine to the core nozzle.

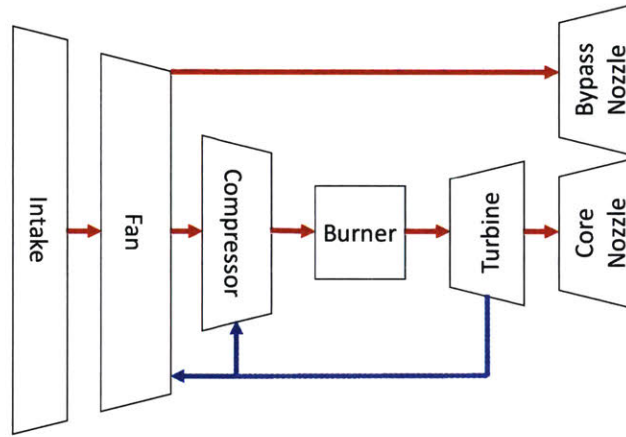


Figure 5.59: Generic Turbofan Architecture. Gas connections: **red**, shaft power connections: **blue**, mechanical connections: **black**, fuel connections: **green**, oil connections: **cyan**, electrical connections: **magenta**.

Figures 5.59, 5.60 depict a generic turbofan architecture. In this case there are two shaft power connections present (turbine to compressor and turbine to fan). There are two gas paths. One gas path is through the core (compressor, burner and turbine) of the engine and another bypasses the core. The propulsive efficiency of a gas turbine engine is given by the following:

$$\eta_{propulsive} = \frac{Thrust \cdot u_0}{\frac{\dot{m}_e u_e^2}{2} + \frac{\dot{m}_f u_e^2}{2} - \frac{\dot{m}_e u_0^2}{2}} \quad (5.123)$$

Where u_0 is the flight velocity \dot{m}_e is the mass flow of air through the engine, \dot{m}_f is the mass flow of fuel through the engine and u_e is the exhaust velocity. The propulsive efficiency conceptually represents the ratio of power to drive the aircraft and the rate of kinetic energy

addition to the flow passing through the engine. Thrust is proportional to the rate of change momentum of the flow (in the numerator). The fan bypass stream provides thrust at higher propulsive efficiency thrust (large \dot{m}_e , small u_e) improving the performance of the engine since overall efficiency is proportional to propulsive efficiency.

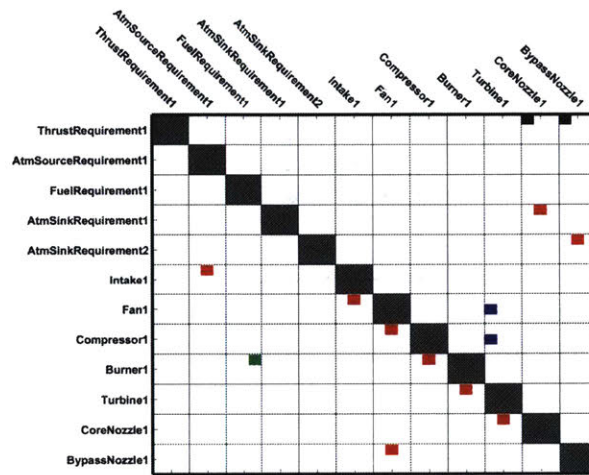


Figure 5.60: Generic Turbofan Architecture DSM. Gas connections: **red**, shaft power connections: **blue**, mechanical connections: **black**, fuel connections: **green**, oil connections: **cyan**, electrical connections: **magenta**.

The geared turbofan architecture depicted in figures 5.61 and 5.62 decouples the rotational speed of the turbine via a gearbox. This enables reduction of fan rotational speed and pressure ratio (and therefore noise) *while* allowing the turbine to provide a similar amount of power with fewer and smaller turbine stages. In practice, this enabled geared turbofans to be designed with lower pressure fan pressure ratios/higher bypass ratios improving propulsive efficiency and noise. As mentioned earlier two cases were run for all architectures. The first constrained individual fan bypass ratios to 12, based on what has been achieved to date, while the second relaxed this constraint allowing geared/electric fans to have bypass ratios of 24 (see figure 5.70).

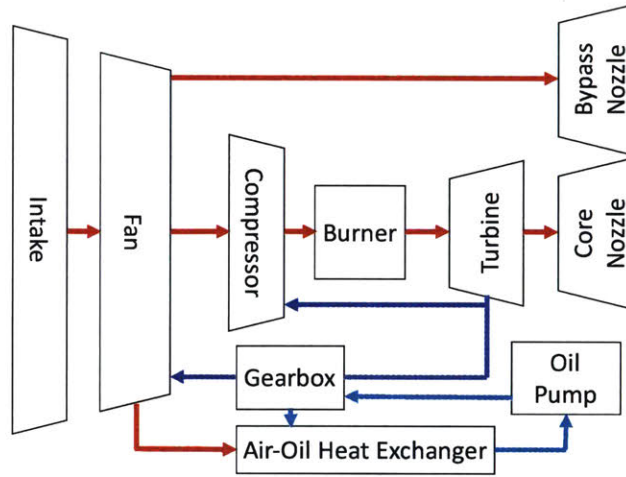


Figure 5.61: Generic Geared Turbofan Architecture. Gas connections: **red**, shaft power connections: **blue**, mechanical connections: **black**, fuel connections: **green**, oil connections: **cyan**, electrical connections: **magenta**.

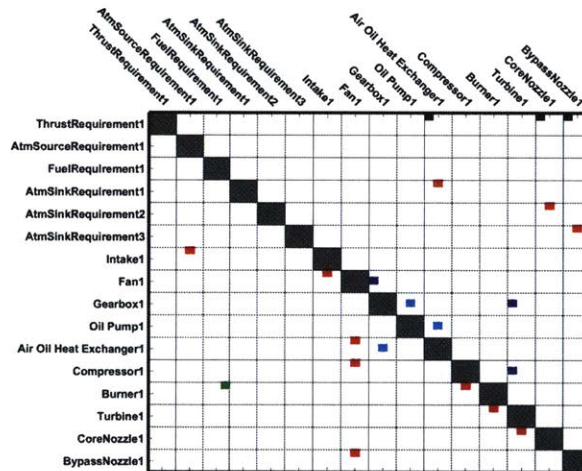


Figure 5.62: Generic Geared Turbofan Architecture DSM. Gas connections: **red**, shaft power connections: **blue**, mechanical connections: **black**, fuel connections: **green**, oil connections: **cyan**, electrical connections: **magenta**.

Another architecture that was generated was a distributed geared turbofan configuration similar to the one in [93] shown in figures 5.63 and 5.64. This architecture has four primary gas paths: one through the core and 3 bypass paths through three bypass nozzles.

Its increased complexity is due in large part to the complex cooling and lubrication system consisting of three heat exchangers that needed to discard waste heat from the three gearboxes.

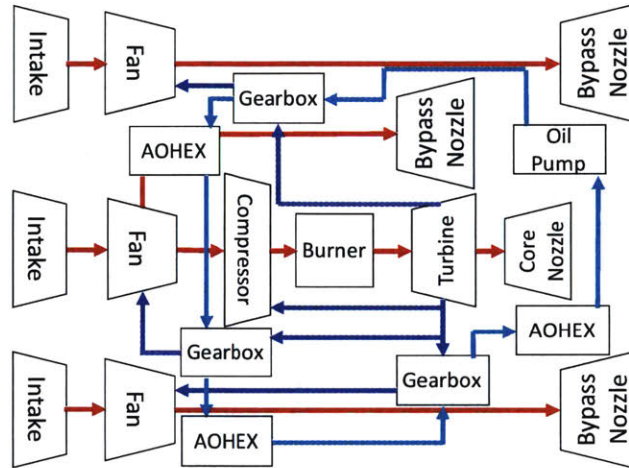


Figure 5.63: Generic Distributed Gearing Turbofan Architecture. Gas connections: **red**, shaft power connections: **blue**, mechanical connections: **black**, fuel connections: **green**, oil connections: **cyan**, electrical connections: **magenta**.

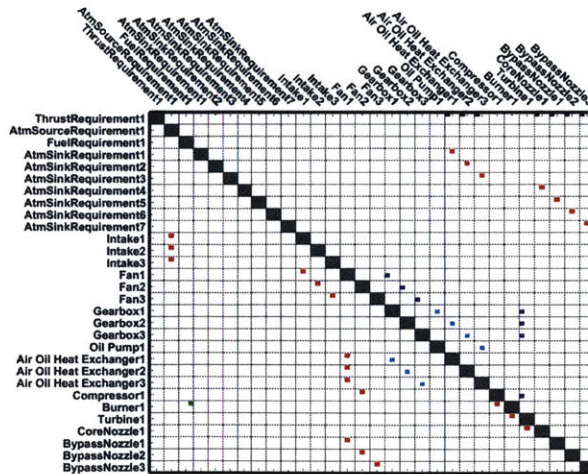


Figure 5.64: Generic Distributed Gearing Turbofan Architecture DSM. Gas connections: **red**, shaft power connections: **blue**, mechanical connections: **black**, fuel connections: **green**, oil connections: **cyan**, electrical connections: **magenta**.

Figure 5.65 depicts the magnified version of figure 5.56, and highlights architecture 46 (see figures 5.66, 5.67) which utilizes 100% electric fans, and architecture 44 (see figure 5.68, 5.69) in which two of the fans are geared and one is electrically driven. The larger the fraction of electrically driven fans the greater the losses associated with power transfer from the core along with the associated cooling requirements. This can clearly be seen in the banded structure of the of the results in both figure 5.56 and figure 5.65. We now examine the

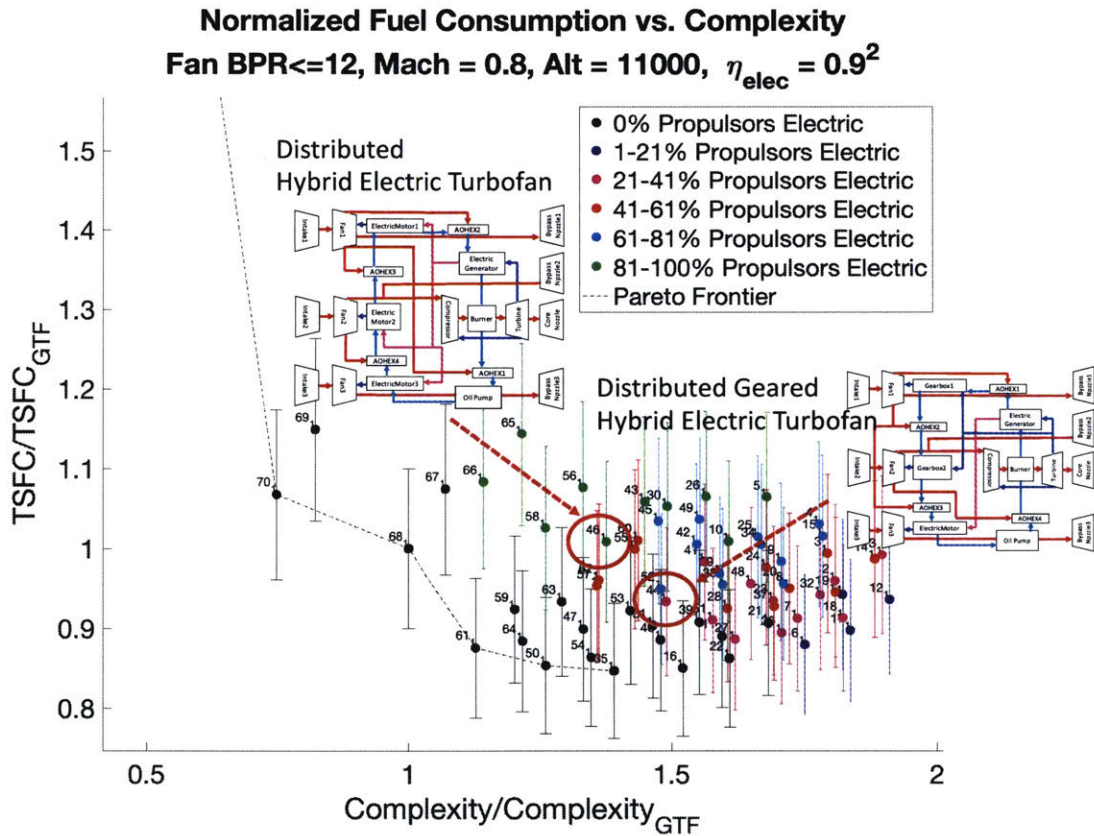


Figure 5.65: Normalized Thrust Specific Fuel Consumption vs. Normalized Complexity. Normalized w.r.t generic geared turbofan architecture.

distributed hybrid electric architecture shown in figures 5.66 and 5.67 in more detail. This architecture is similar to the distributed geared architecture shown in figures 5.63 and 5.64 that was discussed earlier. The primary difference is that all 3 fans in this case are driven

by electric motors which are powered via an electric generator powered by the turbine. The improvements in thrust specific fuel consumption achieved via the very high propulsive efficiency are completely offset by losses in the electrical system. Furthermore the hybrid architecture was found to be 40% more complex than the geared turbofan architecture. In addition it is important to note that the power to weight ratio of non-superconducting electrical motors is on the order of 20 times poorer than that of a gearbox, which would result in a large additional weight penalty. This will be discussed in more detail later in this chapter. While these results suggest that fully electric hybrid architectures are strictly dominated, it is important to note that some of these losses may be offset in hybrid architectures through fan placement in boundary layer ingesting locations and more flexible airframe integration since electrical transfer of power to fans weakens coupling between core and fan placement. This was the case in [90]. Since such an analysis requires knowledge about airframe integration, it is left for future work.

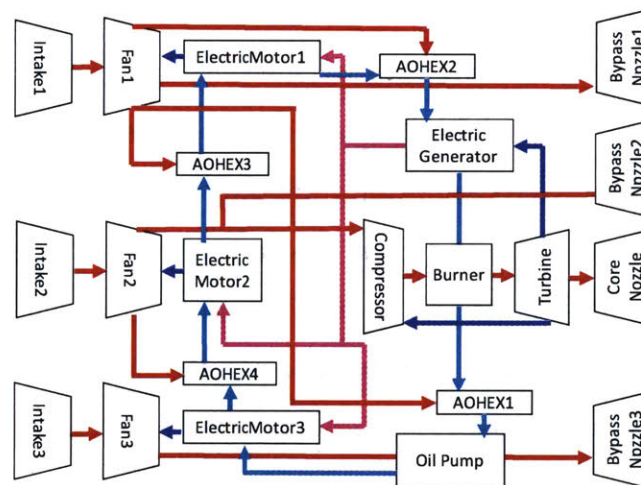


Figure 5.66: Generic Distributed Hybrid Electric Architecture. Gas connections: **red**, shaft power connections: **blue**, mechanical connections: **black**, fuel connections: **green**, oil connections: **cyan**, electrical connections: **magenta**.

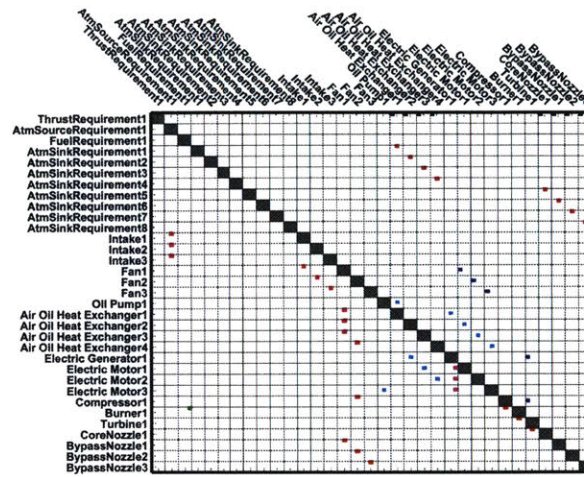


Figure 5.67: Generic Distributed Hybrid Electric Turbofan Architecture DSM. Gas connections: **red**, shaft power connections: **blue**, mechanical connections: **black**, fuel connections: **green**, oil connections: **cyan**, electrical connections: **magenta**.

Perhaps the most interesting family of architectures are the ones in which some of the fans are mechanically driven and some are driven by electric motors. These are shown in figure 5.65 in magenta, cyan, blue and red. These architectures provide a way of taking advantage of both the higher mechanical transfer efficiency of geared fans and the freedom of placement of electrically driven fans. One such architecture is depicted in figures 5.68 and 5.69. This is yet another distributed architecture. In this case, however, two of the fans are driven mechanically via gearboxes and one electrically driven fan. As such it is in between the two previously described distributed architectures and provides an uninstalled thrust specific fuel consumption improvement compared to a generic geared turbofan architecture.

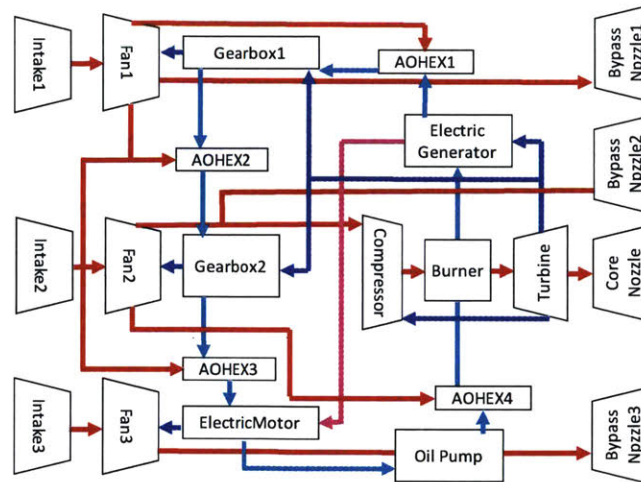


Figure 5.68: Generic Distributed Geared Hybrid Electric Architecture. Gas connections: **red**, shaft power connections: **blue**, mechanical connections: **black**, fuel connections: **green**, oil connections: **cyan**, electrical connections: **magenta**.

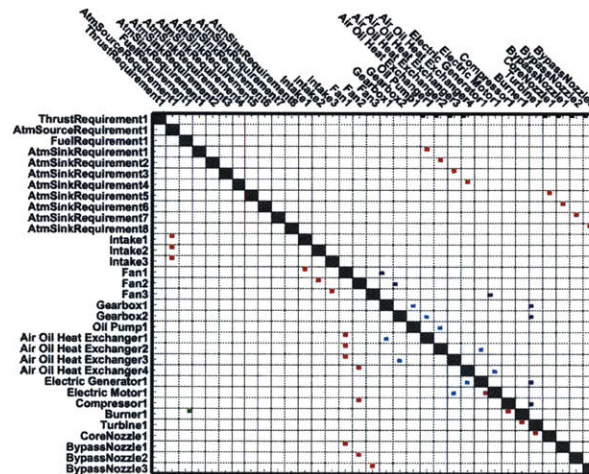


Figure 5.69: Generic Distributed Geared Hybrid Electric Turbofan Architecture DSM. Gas connections: **red**, shaft power connections: **blue**, mechanical connections: **black**, fuel connections: **green**, oil connections: **cyan**, electrical connections: **magenta**.

Figures 5.70 and 5.71 provide a similar analysis to the one presented in figures 5.56 and 5.65 but relaxes the constraint on geared/electric drive fans to 24 (double what has currently commercially been achieved). In this case, as expected improvements in propulsive efficiency

from more fans yield diminishing returns and going beyond a single fan and an ultra-high bypass ratio geared architecture is positioned at the knee of the Pareto frontier.

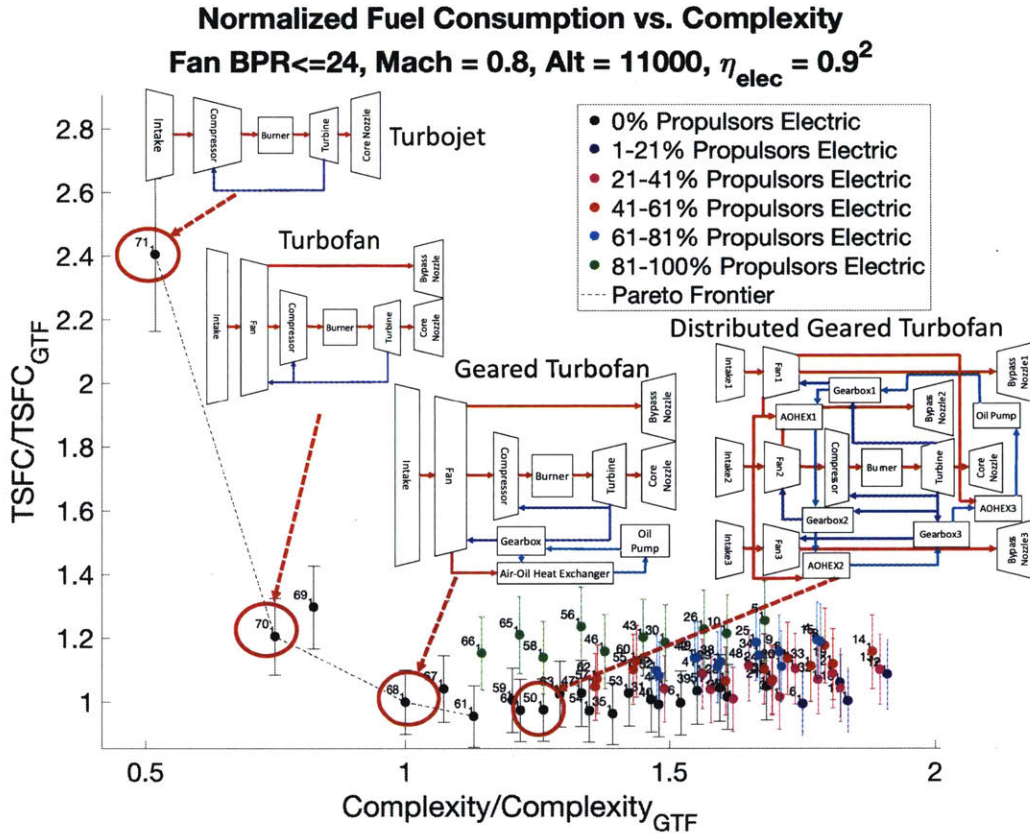


Figure 5.70: Normalized Thrust Specific Fuel Consumption vs. Normalized Complexity. Normalized w.r.t generic geared turbofan architecture.

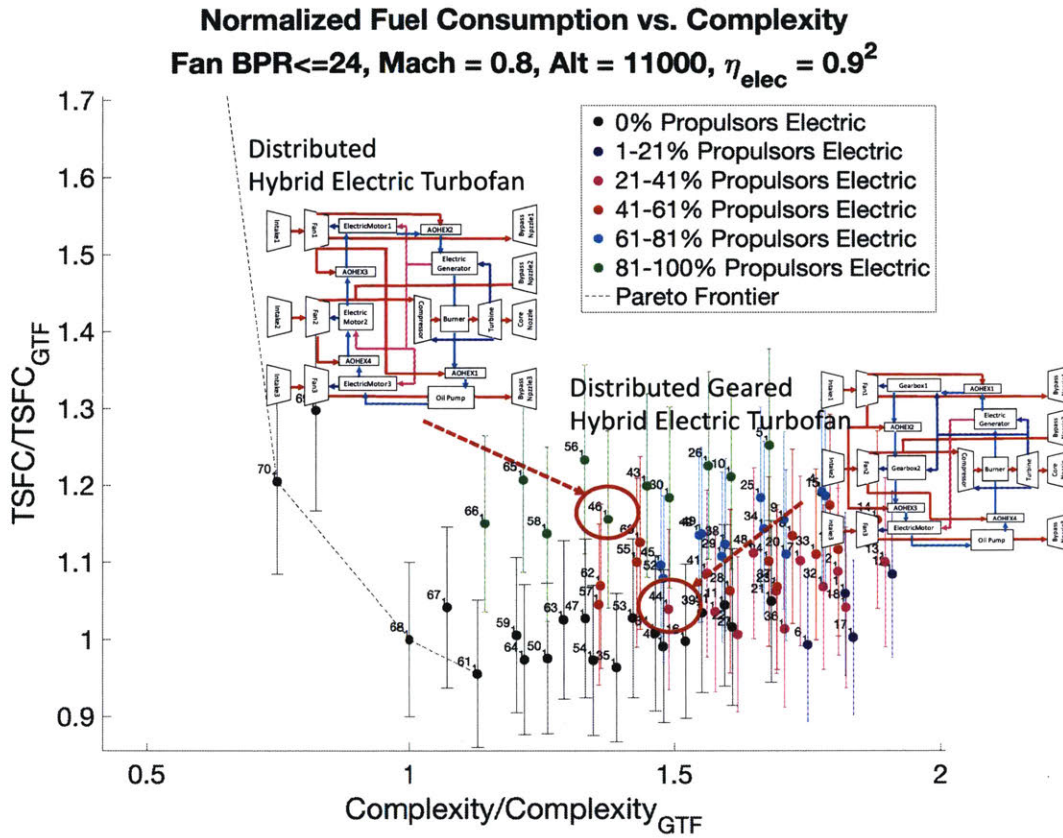


Figure 5.71: Normalized Thrust Specific Fuel Consumption vs. Normalized Complexity. Normalized w.r.t generic geared turbofan architecture.

We now examine fuel consumption-mass tradeoffs for the 70 turbofan architectures that were generated in this analysis. Four different cases are considered. These consist of high and ultrahigh bypass ratio constraints as well as 90% and 95% assumptions on transfer efficiency of electrical motors and generators and are depicted in figures 5.72, 5.73, 5.74 and 5.75. Figure 5.72 depicts the nominal case where bypass ratio is limited to 12 and motor/generator efficiencies are 90%. It is immediately clear that distributed architectures are as expected more massive than traditional single fan architectures. Architecture 50 for example which was discussed in more detail earlier (see figure 5.64), is on the order of 1.6 times as massive as a traditional geared turbofan.

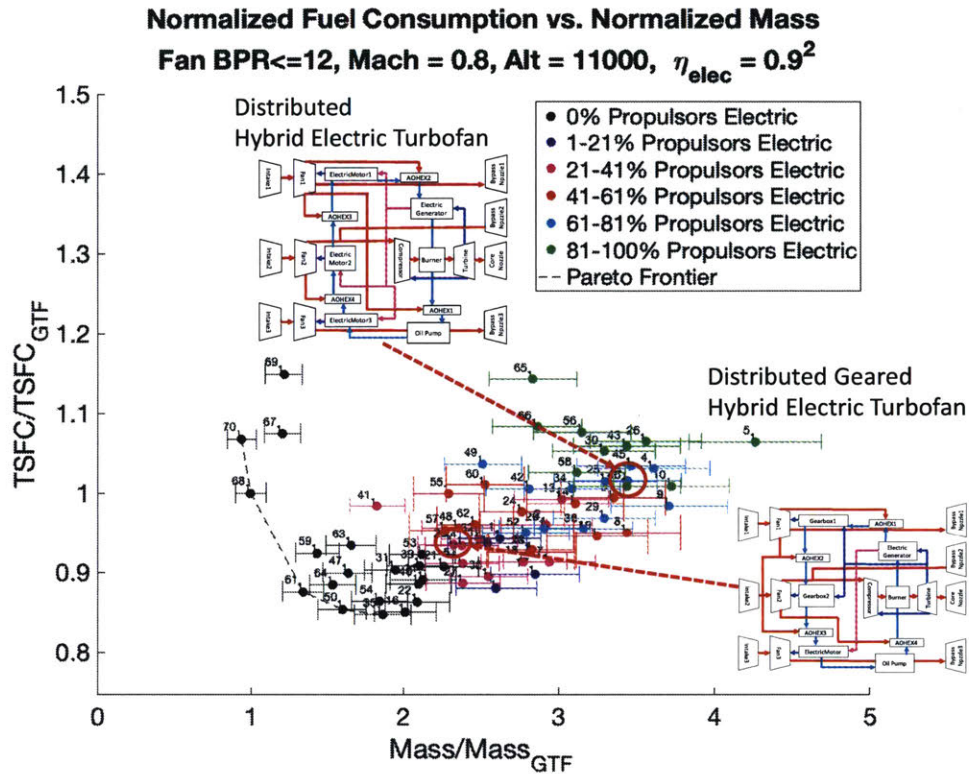


Figure 5.72: Normalized Thrust Specific Fuel Consumption vs. Normalized Mass. Normalized w.r.t generic geared turbofan architecture.

This is due to the mass of additional gearboxes, fans, and their fairings. Hybrid electric architectures have yet greater mass, primarily due to the larger cooling/lubrication systems needed to remove heat from electrical components. The heaviest architectures are highly distributed systems in which all fans are driven electrically (shown in green). Distributed architectures in which only some of the fans are driven electrically have lower mass while at the same time providing fuel consumption improvements compared to the geared turbofan. This is the case with architecture 44 highlighted in figure 5.72.

Optimization of uninstalled fuel consumption was also carried out for three other cases: ultra-high bypass ratio-nominal electrical transfer efficiency (figure 5.73), high bypass ratio-

high electrical transfer efficiency (figure 5.74), ultra-high bypass ratio-high electrical transfer efficiency (figure 5.75). For the high bypass ratio cases increasing electrical transfer efficiency generally decreased overall engine weight due to downsizing of the cooling/lubrication systems. For example, the 100 % propulsors electric distributed architecture (number 46) depicted in all figures reduced mass by on the order of 10% when electrical efficiency was increased by approximately %10.

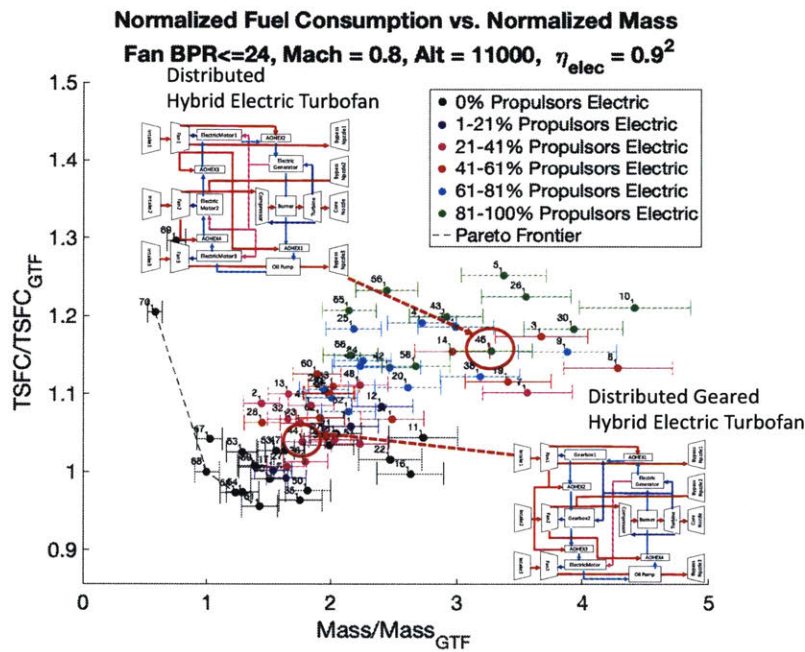


Figure 5.73: Normalized Thrust Specific Fuel Consumption vs. Normalized Mass. Normalized w.r.t generic geared turbofan architecture.

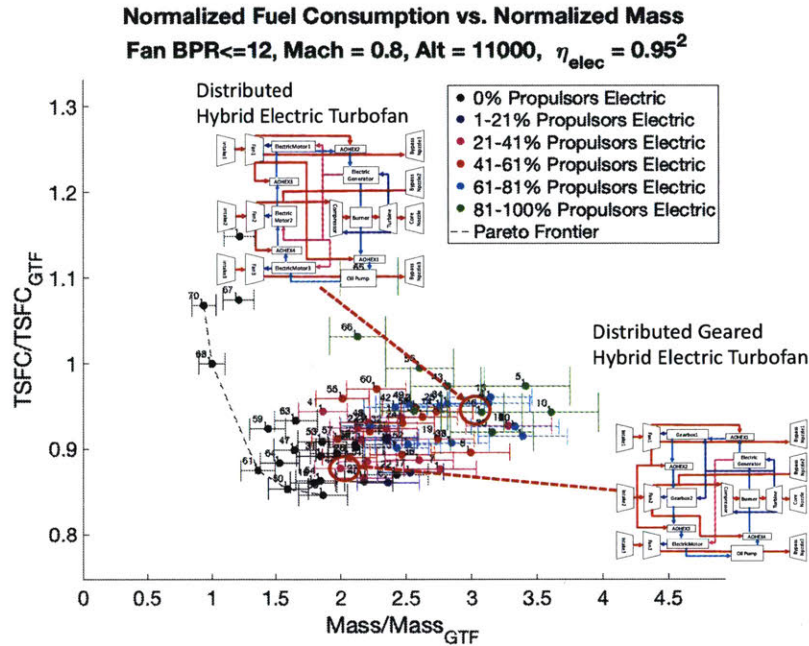


Figure 5.74: Normalized Thrust Specific Fuel Consumption vs. Normalized Mass. Normalized w.r.t generic geared turbofan architecture.

A similar trend was observed for architecture 44 which is only partially electric both in the high bypass ratio case shown in figures 5.72 and 5.74 and the ultra-high bypass ratio case shown in figures 5.73 and 5.75.

To summarize, in this chapter we generated engine architectures from libraries of components at two layers of abstraction and examined their performance-complexity and performance-mass tradespace for uninstalled operation. At the first level of abstraction no optimization was conducted, while at the second, architectures were optimized first via genetic algorithm and then via a gradient based approach. Distributed architectures in general were found to be up to twice as complex as conventional ones. A major contributor of this was increased complexity in cooling and lubrication systems associated with more gearboxes as well as electrical motors/generators. Distributed engine architectures in which all propulsors were electric were found to be up to 3.5 times as heavy as a generic geared turbofan architecture,

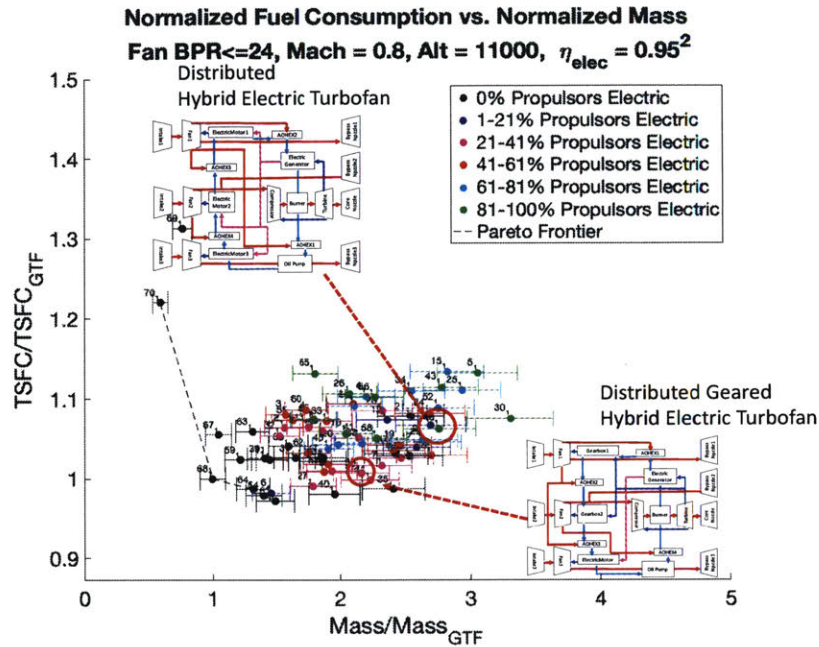


Figure 5.75: Normalized Thrust Specific Fuel Consumption vs. Normalized Mass. Normalized w.r.t generic geared turbofan architecture.

as well as being more complex and consuming more fuel. This is primarily due to losses in electrical power transfer which necessitated larger cooling systems and the power to weight of electrical components compared to gearboxes. Distributed hybrid electric architectures were found to have lower uninstalled fuel consumption than a generic geared turbofan architecture, but were somewhat heavier and less efficient than fully mechanically driven distributed architectures. These partially electric propulsor architectures, in which some fans are driven electrically while others are driven by gearboxes, may prove to be attractive if electrically driven fans can be placed in boundary layer ingesting locations which may have been difficult to realize through mechanical power transfer. Furthermore, such architectures may also provide to be a useful testbed for improved electric motor/generator technology at power levels below what is required for fully electric flight at Mach 0.8, but high enough so that they could be used on smaller electric regional aircraft. This is consistent with recent work

[90]. Future work will examine different ways of optimally integrating the set of engine architectures with novel aircraft configurations as well as optimization for full flight profiles and electric energy storage.

Chapter 6

Reconfigurable Mobile Device Case Study

We now consider a modular reconfigurable mobile device based on the Google ARA reconfigurable phone concept [94]. The objective of this device was to increase user satisfaction by:

- Allowing users to customize their phone by adding or removing hardware modules purchased from a “module store”. The vision was to treat customization of the hardware of the device in a similar manner as we have become used to seeing software reconfiguration via applications [20].
- Allowing users to reconfigure their phone over time to reflect changing preferences and obsolescence/degradation of individual modules [20].

A diagram of the second iteration of this device (Spiral 2) is given in figure 6.1. The device consists of an “Endoskeleton”, shown in gray, which allows consumers to attach 3rd party modules like cameras, batteries and environmental sensors to the phone. The front of the

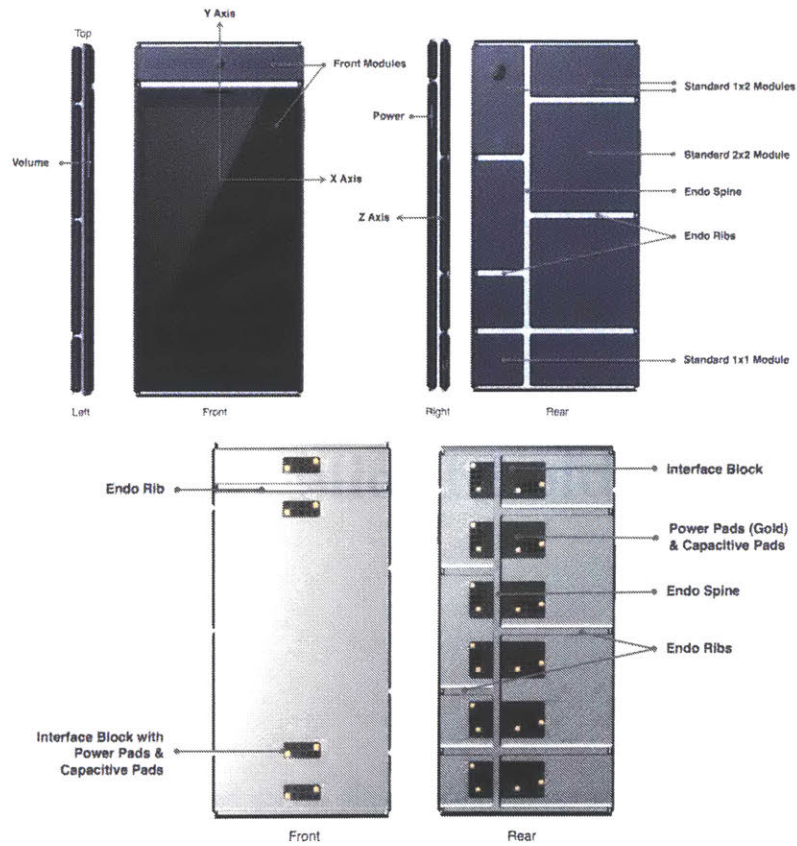


Figure 6.1: Google ARA Spiral 2 Diagram. Image source: [19]

of the device consists of a large slot intended for screens and a smaller slot intended for audio/video modules. The back of the device consists of 8 additional slots of varying sizes intended for adding anything from battery modules, processors and environmental sensors to blood glucose sensors.

We can think of this device in the context of manufacturing paradigms from the 1850s until the present day as shown in figure 6.2 from [20]. Figure 6.2 shows the total number of variants of a product divided by the number of products manufactured on the y axis against the total number of products manufactured divided by the time required to manufacture each product on the x axis. At the top left hand corner we see craft manufacturing (prior

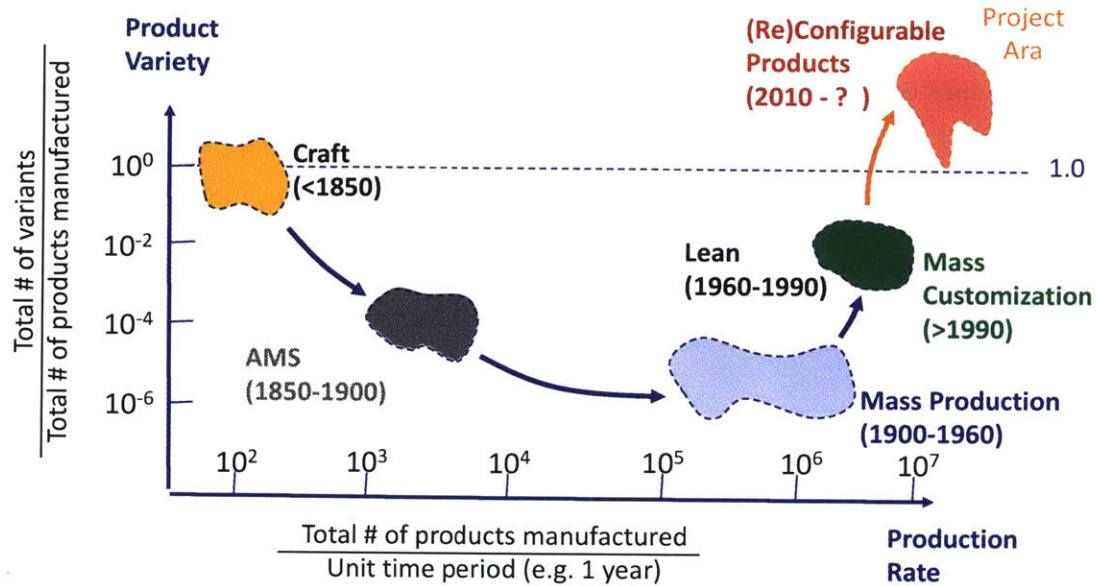


Figure 6.2: Shifting industrial paradigms. Source: [20]

to 1850) in which the number of variants was on the order of the number of products manufactured. In other words, each product was unique and the total number of products manufactured was small. Craft manufacturing was followed by the American System of Manufacturing, which focused on the improving technology and processes, reduced product variety and increased the number of products manufactured. During the industrial revolution the total number of variants became smaller relative to the total number of products manufactured. This trend continued until the 1960s through to lean manufacturing and mass production. Mass customization increased the variety of products in the second half of the 20th century. Since the Google ARA concept allows reconfiguration of an existing product after it has been purchased as well as customization at the point of purchase, the ratio of variants to the number of functioning phones is in principle greater than one. In other words, a consumer could purchase a customized device and reconfigure it over time. This means that the number

of variants is greater than the number of “Endoskeletons” sold. This has already been done in the software arena (software changes are made after purchase of a phone, tablet or computer). It has to a limited extent also been done with hardware upgrades for computers (users can change rapid access memory, graphics cards or storage on their computers after purchase). The ARA concept is unique in that functional reconfiguration of a device is possible and encouraged not just in terms of improving performance with which functions are achieved but in terms of which functions are present (e.g. addition of blood glucose sensor).

To summarize, the Google ARA reconfigurable mobile device concept was designed to allow hardware and software customization and reconfiguration. In this chapter we examine the tradeoffs between cost and user benefit utility (a measure of performance related to user preferences) as well as the tradeoffs between user benefit utility and configuration complexity. In the air-breathing propulsion case study we applied the Magellan architecture generator/evaluator to an architecting/general design problem (sizes of components were not fixed a-priori). In this chapter we apply it to a configuration problem (sizes of components are fixed a-priori). The first purpose this case study is to illustrate the generality of the approach by showing that it can be applied to both general design (system architecting) and configuration problems. The second purpose of this case study is to generate the architectural tradespace of the ARA platform. Due to the fact that model behavior was linear in this case, and the fact that all modules connected to a single platform which mediates information and energy transfer between them, only 1 level of abstraction was deemed necessary for this case study.

6.1 Modeling

The value perceived by potential users of device configurations is assumed to depend on the modules that are present, battery life and the cost of a configuration. We start our modeling description with a battery life model base on the one presented in [21].

6.1.1 Battery Life Model

We construct a battery life model for the device based on one validated against existing smartphones from [21]. Battery life of each configuration depends on how it is used. A typical use profile is depicted in figure 6.3 and consists of different states that a typical device is over time, including “off”, “standby”, “phone call” etc. Figure 6.4 depicts the state of modules for each part of the use profile. Figure 6.5 depicts the power consumption of modules in different states. Battery life may be calculated by dividing the total energy available on the device by the average power consumption as shown in the equations below. Total power consumption is calculated for each operating regime k (e.g. state of the device). Average power is then computed based on the use profile.

The governing equations and parameter assumptions of the model are the same as the ones validated against existing smartphone data in [21] but are implemented in Matlab rather than Simulink. This leverages the fact that the only source of dynamics are the different states that the device can be in (off, standby, talk, web, video, music) which allows the use of a linear model that can be implemented via Matlab script that runs in seconds rather than hours.

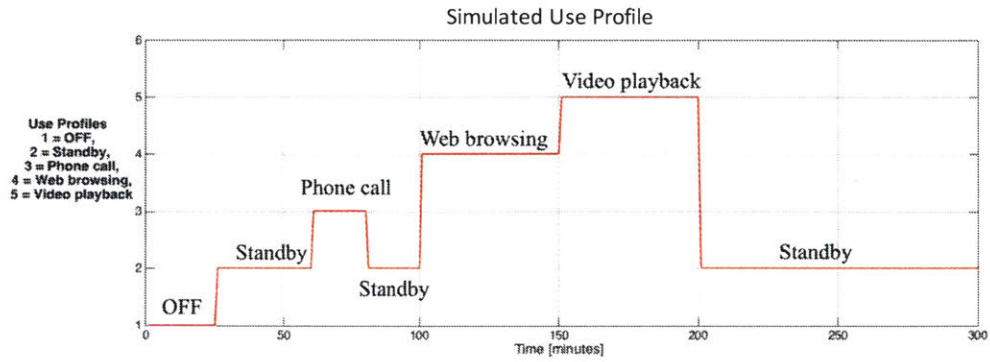


Figure 6.3: Dynamic use profile [21]

State	Screen (Basic)	Screen (Advanced)	Audio (Basic)	Audio (Advanced)	Environmental Sensor	Medical Sensor	Security Sensor	Antenna (Basic)	Antenna (Advanced)	Game	Interface (Basic)	Interface (Advanced)	Camera (Basic)	Camera (Advanced)	Memory 16Gb	Memory 64Gb	Memory 256Gb	Processor	
Off	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Standby	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
Talk	1	1	2	2	0	0	0	2	2	0	0	0	0	0	0	0	0	0	1
Web	2	2	1	1	1	1	1	2	2	1	0	0	0	0	2	2	2	2	2
Video	2	2	2	2	0	0	0	0	0	0	0	0	0	0	2	2	2	2	2
Music	2	2	2	2	0	0	0	0	0	0	0	0	0	0	2	2	2	2	2

Figure 6.4: Use profile (0 off, 1 standby, 2 on) [21]

State	Screen (Basic)	Screen (Advanced)	Audio (Basic)	Audio (Advanced)	Environmental Sensor	Medical Sensor	Security Sensor	Antenna (Basic)	Antenna (Advanced)	Game	Interface (Basic)	Interface (Advanced)	Camera (Basic)	Camera (Advanced)	Memory 16Gb	Memory 64Gb	Memory 256Gb	Processor	
Off	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Standby	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.02
Talk	0.02	0.07	0.02	0.03	0.00	0.00	0.00	0.18	0.36	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.18
Web	0.14	0.18	0.03	0.07	0.00	0.02	0.01	0.18	0.36	0.04	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.46
Video	0.13	0.18	0.02	0.03	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.46
Music	0.02	0.07	0.02	0.03	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.18

Figure 6.5: Power consumptions in each state (Watts) from [21]

$$\text{Total Power Consumption}_{\text{regime } k} = \sum \text{Power consumption}_{\text{module } i, \text{regime } k} \quad (6.1)$$

$$\text{Average Total Power Consumption}_{\text{regime } k} = \sum_{i=1}^k w_i \text{Total Power Consumption}_{\text{regime } k} \quad (6.2)$$

$$\text{Battery Life} = \frac{\text{Battery Energy}}{\text{Average Total Power Consumption}} \quad (6.3)$$

6.1.2 Benefit and Price Utility Model

To compute the utility from benefits and the utility from price of individual mobile device configurations we rely on the approach taken in [22]. Benefit utility $_j^h$ expresses the benefit architecture j provides to user h . Price utility $_j^h$ expresses the utility associated with price that architecture j provides to user h . Different potential users h will derive different benefit from the same configuration j . This is because their part-worth utilities b_i^h and b^h (preferences for modules and sensitivity to price) can be different. In the equation below b_i^h represents the part-worth utility of module i to user h and u_i indicates the presence of a module.

$$\text{Benefit Utility}_j^h = \prod \exp(b_i^h u_i) \quad (6.4)$$

We can also estimate the utility of price as shown in [22]. In the equation below b represents the utility of price. $price_i$ is the price of modules to the consumer.

$$\text{Price Utility}_j^h = \exp\left(b^h \sum price_i\right) \quad (6.5)$$

Value is defined as the ratio of benefit and price utility.

$$Value_j^h = \frac{\text{Benefit Utility}_j^h}{\text{Price Utility}_j^h} \quad (6.6)$$

Part worth utilities (representing user preferences) were estimated from a survey of 200 individuals in Puerto-Rico. Ward's clustering algorithm was used to determine 5 distinct user types from the data based on common part-worth utilities [22]. The dendrogram in figure 6.6 depicts relative size of different clusters of users. Figure 6.7 depicts part-worth utilities for modules and price. Different clusters can have example significantly different preferences for modules. Table 6.1 provides a high level interpretation of the distinguishing features of different user clusters.

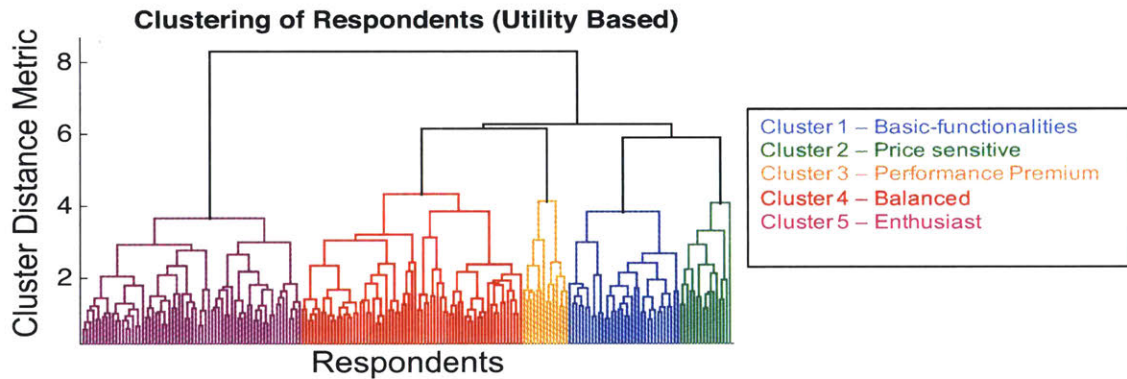


Figure 6.6: Dendrogram representing clustering of preferences (Ward's method)

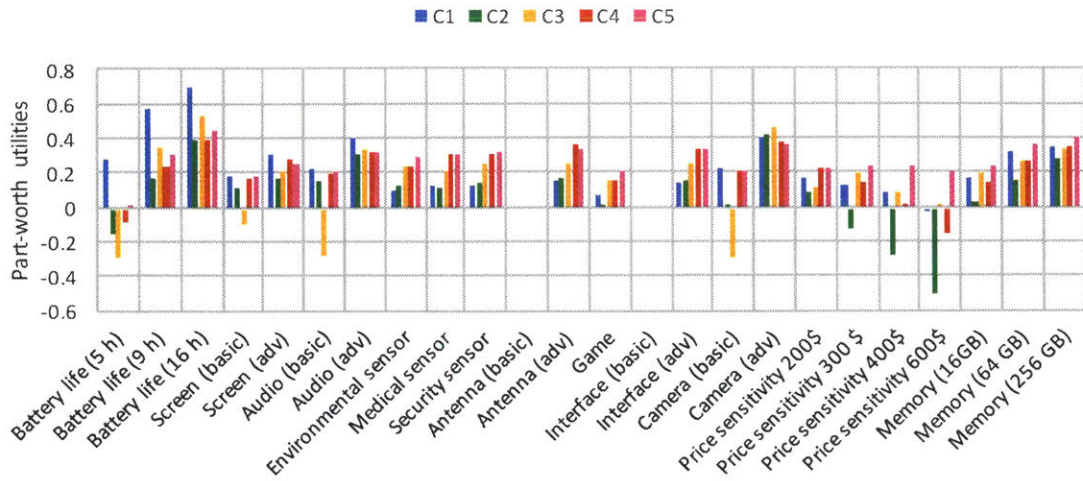


Figure 6.7: Part-worth utilities (preferences for different modules, prices and battery lives), [21]

Cluster number	Cluster name	Distinctive feature(s)	Percentage of customers
C1	Basic-functionalities	Main interest in feature phone functionalities	17%
C2	Price sensitive	Extremely sensitive to price	8%
C3	Performance premium	High preference for high-performance modules	8%
C4	Balanced	No distinctive features	34.5%
C5	Enthusiast	High utility for most of the modules, low price sensitivity	33.5%

Table 6.1: Description of different clusters of users

6.2 Configuration Generation

We now conduct a search of all possible reconfigurable mobile device configurations. The primary assumptions for this analysis were:

- The platform has a 3.5 Volt 1000 mAh battery.
- Modules can be distributed across different slots.

- We assume there are 21 Modules and use approximately 60 component and connection rules.
- The total area taken up by modules is less than or equal to total area available. Note that the reconfigurable mobile device is composed of a grid of 18 square units of area on the back of the device, 3 units of area on front of the device as well as a large slot for a screen. The smallest modules occupy a single unit of area and all modules occupy integer multiples of the basic unit of area. The areas taken up by the various modules are given graphically in the figure 6.8.

For this analysis we assume that the device must at a minimum provide basic functions of a typical low-end smartphone. Table 6.2 depicts the different modules used in this analysis along with their descriptions and a qualitative performance level from [21]. Figure 6.8 shows a diagram of the device along with the library of components considered in this analysis. The front of the device, which is shown in the top right corner of figure 6.8, consists of a large slot for screens and a smaller slot for other modules. The back of the device consists of 8 slots as shown in the bottom right corner of figure 6.8. The sizes of individual modules are depicted graphically. Each module has an integer number of units of area. Sensors are 1 unit of area whereas batteries are 4 units of area.

The baseline DS2M and convex hull filtered DS2M are given in figures 6.9 and 6.10 respectively. Architecture generation yields 21,168 configurations of the mobile device.

TYPE	PRIMARY FUNCTION	PERFORMANCE LEVEL
Endoskeleton	Supporting and connecting the modules	---
Processing unit	Processing data	---
Screen (advanced)	Displaying information	High resolution
Screen (basic)	Receiving inputs Displaying information	Mid resolution
Audio (advanced)	Receiving inputs Generating sounds	Hi-fi quality
Audio (basic)	Generating sounds	Standard quality
Antenna (advanced)	Communicating data	Optimal reception everywhere
Antenna (basic)	Communicating data	Good reception
Camera (advanced)	Taking pictures	Professional quality
Camera (basic)	Taking pictures	Amateur quality
Interface (advanced)	Connecting to other devices Connecting to electric plugs	Fast data transfer rate
Interface (basic)	Connecting to other devices Connecting to electric plugs	Slow data transfer rate
Environmental sensor	Monitoring environmental conditions	---
Medical sensor	Capturing health data	---
Security sensor	Preventing unauthorized access	---
Gaming interface	Provide inputs to videogames	---
Memory (advanced)	Storing data	256 GB
Memory (intermediate)	Storing data	64 GB
Memory (basic)	Storing data	16 GB
Battery (basic)	Storing and providing electrical energy	250 mAh
Battery (intermediate)	Storing and providing electrical energy	500 mAh
Battery (advanced)	Storing and providing electrical energy	750 mAh

Table 6.2: List of 21 modules and the platform which they can occupy. Basic modules consist of existing technologies repackaged to fit within a reconfigurable mobile device. Intermediate modules represent technology with functionality found only in the most advanced integral devices. Advanced modules may not have any analog in integral smartphones [21].

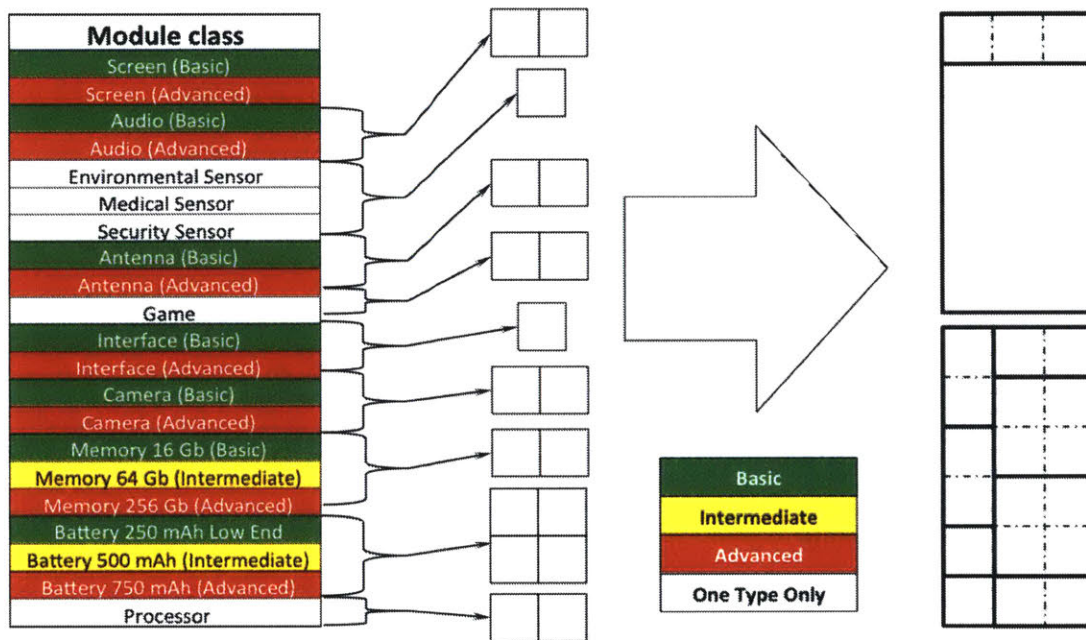


Figure 6.8: List of 21 modules and the platform which they can occupy. Basic modules consist of existing technologies repackaged to fit within a reconfigurable mobile device. Intermediate modules represent technology with functionality found only in the most advanced integral devices. Advanced modules may not have any analog in integral smartphones. Module prices are based on data collected in [21].

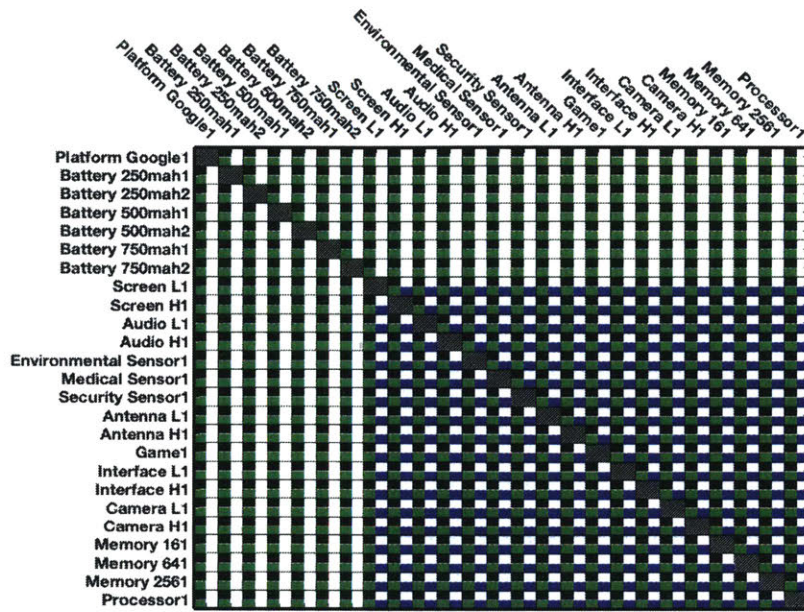


Figure 6.9: Google ARA DS2M. Energy connections: **green**, mechanical connections: **black**, information connections: **blue**

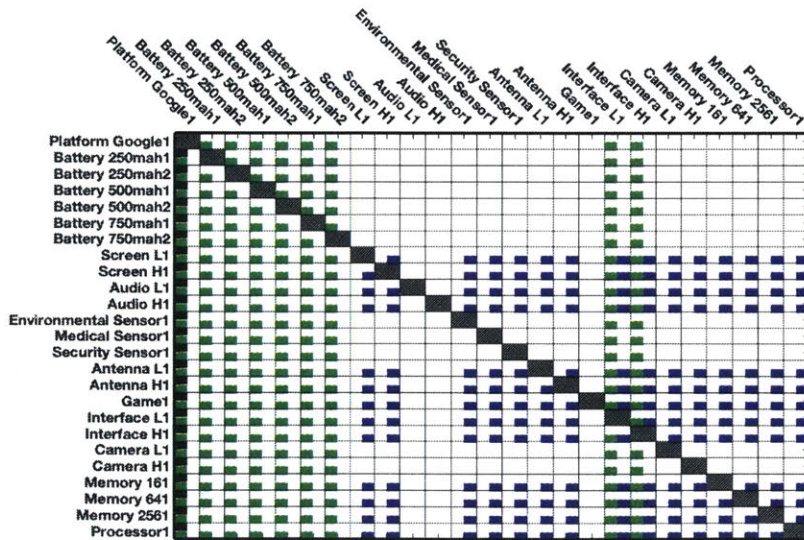


Figure 6.10: Google ARA DS2M Filtered (82.5 % Reduction in Number of Connections). Energy connections: **green**, mechanical connections: **black**, information connections: **blue**

The results of the architecture generation are given in figure 6.11 6.12. Figure 6.11 depicts the log of benefits utility and price utility for all 5 clusters of users. The largest difference between users is between clusters 2 and 5. Going from left to right on the Pareto frontiers generally entails moving to configurations with a larger number of more advanced modules. This increases the cost, increasing the log of price utility. Cluster 2 does not experience a significant increase in benefit utility in exchange for additional features. The opposite is true for cluster 5 which experiences a rapid increase in benefit utility with the addition of new modules.

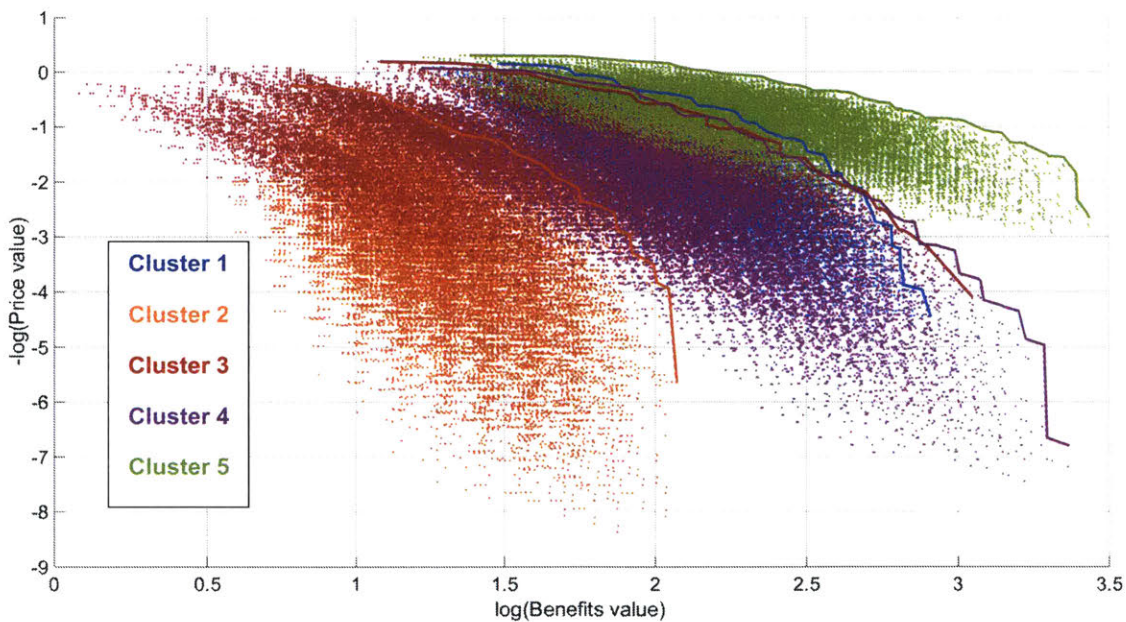


Figure 6.11: Trade-space utility analysis for ARA architectures evaluated according to all 5 clusters of users. Note this figure was generated during a team project and already appears in [22]. Note that this was generated via experience based rules. Some of these rules governing pairwise interaction between components were replaced via convex hulls after generation of this results.

We now examine architectures on the Pareto frontiers in more detail. Figure 6.12 shows log benefits utility and log price utility for cluster 4 of users. Three different architectures with

increasing benefit utility are chosen from the Pareto frontier for further investigation and depicted in figure 6.13.

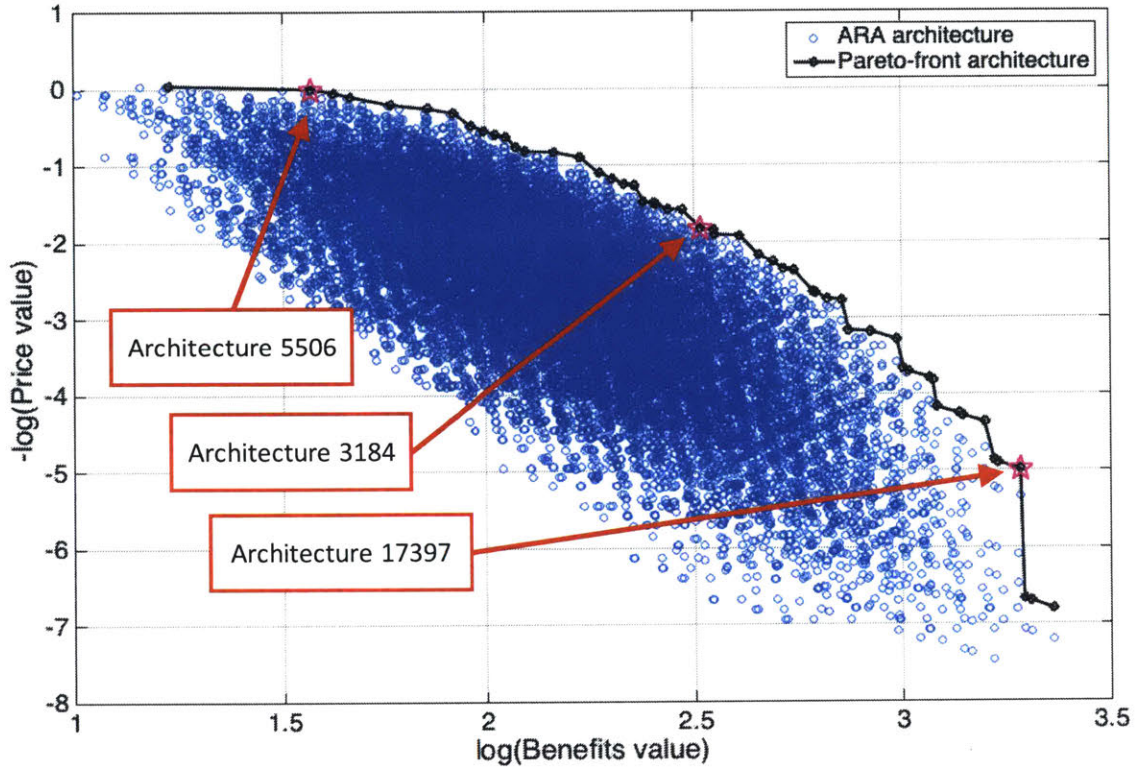


Figure 6.12: Trade-space utility analysis for ARA architectures evaluated according to Cluster 4 part-worth utilities. Note this figure was generated during a team project and already appears in [22].

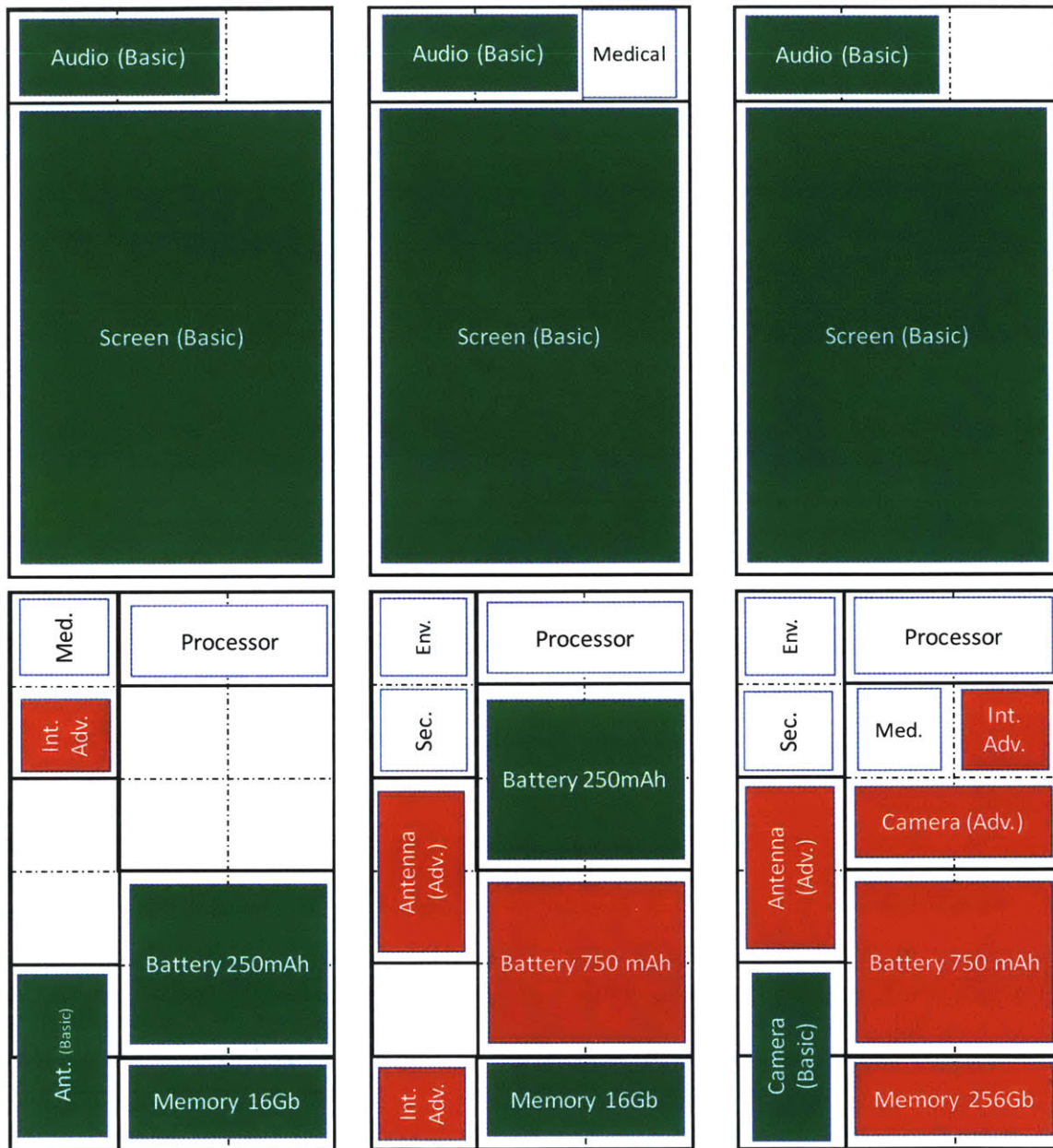


Figure 6.13: Architecture 5506 (Left) Benefit utility (1.57) Price utility (0.0034), Architecture 3184 (Middle) Benefit utility (2.52) Price utility (-1.82), Architecture 17397 (Right) Benefit utility (3.28) Price utility (-4.99). Basic modules are green, Advanced modules are red, Modules with only one type not highlighted.

Architecture 5506 has relatively few modules and small battery capacity generating the lowest benefit utility. Architecture 3184 has more battery modules, an advanced antenna and an advanced interface module along with security and environmental sensors. Architecture 17397 has the highest benefit utility and consists primarily of advanced modules. The chart shows how as we increase module performance, cost and therefore complexity, we increase the performance metric, benefit utility.

Thus far we have generated and simulated 21,168 feasible ARA configurations. It is informative to also consider the total number of possible configurations. By this we mean, given a reconfigurable module device with m slots and a set of possible n modules to select from, what are the total number of configurations that exist ignoring any feasibility constraints. The number of possible module configurations may be found by investigating all possible selections of between 1 and m modules (maximum number allowed on platform) from a set of n (total number of modules available). This can be accomplished using a sum of binomial coefficients [21]:

$$n_{possible} = \sum_{i=1}^{\min\{m,n\}} \frac{n!}{(n-i)!i!} \quad (6.7)$$

Figure 6.14 depicts the set of possible configurations for a set of hypothetical platforms with 1 to 10 slots. The number of possible configurations is calculated for module libraries ranging from 1 module to 21 modules. The Spiral 2 ARA platform given 21 modules is in the top right corner with approximately 1 million possible configurations. Only about 2% of these are feasible according to the architecture generation/evaluation that was carried out in this case study. Therefore, if a user were to randomly assemble a set of modules the likelihood that they would create a non-functional configuration is on the order of 98%. Thus, the platform manufacturer must also carefully consider a way of guiding users towards desired

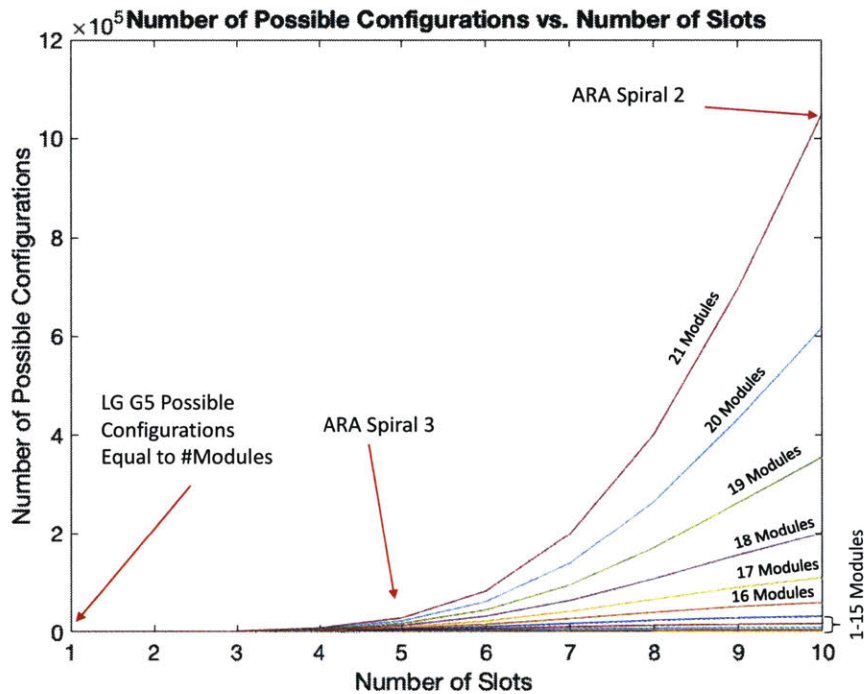


Figure 6.14: Possible configurations.

configurations while maintaining feasibility. Exactly how such guidance can be implemented is out of the scope of this thesis. The Spiral 3 configuration of Google ARA which had 5 slots rather than 10 integrated many of the functions required for a feasible phone into the platform, reducing the likelihood that a randomly assembled configuration would be infeasible. It therefore reduced the space of “possible” configurations while not reducing the space of feasible configurations.

We now examine the distributions of complexity of different mobile device configurations as depicted in figure 6.15. Even the most complex phone configuration that was generated was qualitatively not very different from the most advanced current mobile devices. The key point to take away is that while none of the individual configurations generated for this work were exceedingly complex in isolation, the complexity associated with the tens of thousands of potentially feasible configurations taken together is large.

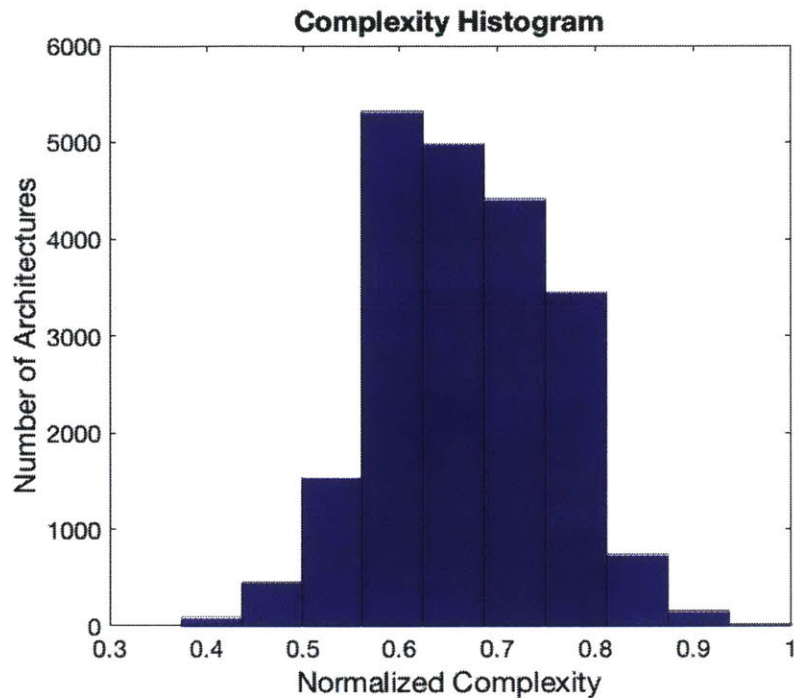


Figure 6.15: Histogram of feasible configurations normalized by the most complex, using notional component and connection complexities. Higher complexities, broadly corresponding to price of modules were set for more advanced modules.

To conclude this chapter, the portfolio of potentially feasible ARA mobile device configurations was generated and evaluated for different classes of users. This information can be used for more detailed verification and validation activity to reduce the likelihood of an infeasible configuration being fielded. While no individual ARA architecture was found to be exceedingly complex, the number of potentially feasible configurations lies in the tens of thousands. It may therefore be said that the complexity of the tradespace of possible configurations rather than any individual configuration was the primary driver of non-recurring engineering cost.

Chapter 7

Summary, Conclusions and Areas of Future Work

The research discussed in this thesis provides both domain specific contributions and methodological contributions. We summarize these here starting with methodological contributions.

Prior to delving into the summary of this work we revisit the Heilmeier research catachism that was outlined in the introduction and map the high level summary of this work to it. Since the main focus of this work was aircraft engines, we will discuss the relevant Heilmeier questions in this context.

H1: What are you trying to do? Articulate your objectives using absolutely no jargon.

H2: How is it done today and what are the limits of current practice?

H3: Who cares? If it is successful, what difference will it make?

H4: What is new in your approach and why do you think it will be successful?

H5: What are the risks and the payoffs?

H6: How much will it cost? How long will it take?

H7: What are the “midterm” and “final” exams to check for its success?

An approach that combined architecture generation, complexity quantification and performance quantification/optimization was presented (*H1*). The approach combined generation of feasible sets of architectures and configurations, performance and complexity quantification, rules that relied on resource flows between components expressed as convex hulls, control over false negatives and was demonstrated at more than one level of abstraction *H4*. This allowed tradespace exploration of families of architectures which had not been explored in detail in the literature (*H2,H4*). The generated portfolio of engine architectures can be drawn upon in future aircraft configuration studies to help meet ICAO fuel efficiency goals, while explicitly showing the complexity associated with improved uninstalled engine performance *H3*.

7.1 Methodological Contributions

Previous work in architecture exploration has examined explicitly classifying required inputs and outputs of components by type. Other, related previous work has used linear constraints on components inputs and outputs in configuration generation problems (problems where the constituent components are fixed). The approach here applies to systems architecting problems and explicitly relates possible flows at interfaces to detailed simulation through convex hulls with flow direction implicitly defined by the convex hulls themselves rather than specified a-priori via subject matter expert opinion. For the engine case study convex hulls were used to inform architecture generation (assembly of unsized functionally constrained components), whereas for the Google ARA case study configurations of different mobile devices were generated (assembly of sized components). For systems with heterogeneous flows (wide range of temperatures, pressures and mass flows) like gas turbine engines, the convex hull constraints result in large reductions in the number of possible connections (88 % reduction for the gas turbine engine case study). For less heterogeneous but centralized

systems like the reconfigurable mobile phone the reduction in the number of connections was 82 %. Thus to summarize the methodological contributions of this thesis are the following:

1. Explicit linking of resource flow constraints at component interfaces to more detailed simulation via convex hulls around operating points.
2. Using convex hulls to link abstraction layers in a domain independent way.
3. Using convex hulls to generate architectures.
4. Allowing the designer to parametrically control the number of false negatives to reduce the size of the design space by varying the normalized allowable normalized volume of intersection of convex hulls.
5. Combining convex-hull based architecture generation, complexity quantification and performance evaluation, allowing tradeoffs between development effort and performance to be made explicitly.

7.2 Domain Specific Contributions

Two different case studies were examined in this thesis. The first studied the complexity performance tradespace of hybrid-electric aircraft engine architectures, while the second examined a reconfigurable mobile phone concept. We now discuss domain specific contributions for each of these.

7.2.1 Air Breathing Hybrid Electric Propulsion

One of the primary motivations for this thesis was the examination of the performance-complexity tradespace of a broad range of air-breathing engine architectures. 71 engine

architectures were generated for this work and optimized by genetic algorithm followed by gradient based optimization. The primary contribution here was the visualization and description of this set of architectures. The full list of findings is as follows:

1. A complexity-performance tradespace of air-breathing propulsion systems for commercial applications was generated. The tradespace included both existing engine architectures and families of distributed mechanically driven/electrically driven architectures. Performance was optimized for minimum uninstalled fuel consumption and complexity was calculated for each architecture in isolation from the aircraft. The reason for doing this was that distributed and ultra-high bypass ratio architectures are difficult to integrate into current low wing airframes due in part to limited space underneath the wing [26]. New engine architectures will likely drive novel airframe configurations each of which will have multiple options for engine-airframe integration. The results here represent a baseline performance which will be augmented depending on integration with future airframe configurations. As such it represents a catalog of new options designers can draw upon.
2. Uninstalled performance of highly distributed architectures in which all propulsors (fans) are electric was found to be dominated for flight at Mach 0.8 at an altitude of 11000 meters given assumed transfer losses and cooling requirements in the electrical systems. The increases in uninstalled performance due to propulsive efficiency for these architectures are more than made up for by increased transfer losses. In addition such architectures were found to be approximately 2.3 times as massive as equivalent mechanically driven distributed architectures due to the limited power to weight ratio of electric motors/electric generators compared to gearboxes and cooling system mass. Finally, distributed electrically driven architectures were found to be up to approximately twice as complex as the current geared turbofan architecture..

3. In addition to architectures in which all fans were electric, families of engines were generated with all different combinations of electric and mechanical propulsors (fans). These hybrid-electric families had performance much closer to mechanically driven distributed architectures. Some of these concepts, while up to twice as complex as the geared turbofan architecture, had improved uninstalled performance compared to current engine architectures due to gains in propulsive efficiency. In addition the weight penalty for such architectures due to poor power to weight ratios of electric motors/generators was approximately 50% smaller than fully electric ultrahigh bypass ratio architectures since the fraction of power transferred electrically was lower than for fully electric systems.

Current gearbox efficiencies are on the order of 99%. Currently electrical transfer losses are on the order of 20% (given approximately 10% losses when converting mechanical power to electrical power and 10% loss when converting electrical power to mechanical power). Hybrid architectures allow for sources and sinks of power to be far away from each other (reduce mechanical complexity). Placing an electric fan powered by a gas turbine in a boundary layer ingesting position does make even non attractive [90] compared to the current state of the art. This work shows that uninstalled thrust specific fuel consumption could be improved even without boundary layer ingestion through partially electric ultrahigh bypass ratio distributed propulsion systems by on the order of 5-10% compared to the modern geared turbofan architecture.

If we are interested in total fuel burn we must also take into account the contribution of the weight of the engine itself to the drag of the aircraft. While a detailed analysis is left for future work, we can gain attain a first order understanding relatively easily. Assuming constant aircraft lift over drag [28], every 1% increment to aircraft mass will result in a 1% increase in drag and therefore thrust and fuel consumption. Assuming

that thrust to weight of modern aircraft engines is on the order of 5 [8] and that thrust to weight of civil twin engined aircraft is on the order of 0.3 at takeoff [26] the ratio of engine mass to aircraft mass is on the order of 6%. If engine mass were to approximately double, as would be the case for the partially electric architecture with 3 fans (44) discussed in the air-breathing engine case study, we would increase aircraft mass by approximately 6% resulting in an approximately 6% fuel burn penalty assuming airframe integration was carried out in a manner that did not increase drag. The improvement of uninstalled performance of architecture 44 was approximately of 5-10 %. This first order approximation means that to gain any statistically significant benefit in overall fuel consumption from turboelectric architectures BLI will be required. Note that this is also subject to the assumption that all fan nozzles located downstream of geared/electric fans are the same size.

It is important to also take a step back and evaluate long term objectives of propulsion when examining new engine architectures. If the objective is to reduce emissions of carbon dioxide along with other greenhouse gases, achieving fully electric propulsion concepts for regional aircraft is one potential path to such a goal. Hybrid electric concepts where some propulsors are electrically powered and placed in boundary layer ingesting positions could be valuable testbeds for such future concepts since they would provide performance improvements relative to existing engine architectures while simultaneously achieving technology development goals in high power density electrical motors/generators. It is however important to note that such systems are perhaps the most complex, due their heterogeneity and complexity of supporting systems.

4. Even with “optimistic” assumptions on the relationship of complexity with development time, 3 geared turbofan architectures require at least 50% more development effort (compared to the current geared turbofan architecture). Looking at the next

step in aircraft engines, it is clear that ultrahigh bypass ratio turbofans are the least complex while providing propulsive efficiency benefits. These architectures will however require some form of aircraft configuration change, since space underneath the wings of low-wing aircraft configurations is limited [26].

7.2.2 Reconfigurable Mobile Device

The second case study featured a reconfigurable mobile device based on Google's ARA reconfigurable mobile phone concept. The idea behind this concept was to give users the ability to highly customize and reconfigure their phones with "plug and play" hardware modules using an "Endoskeleton" bus which would facilitate mechanical, information and energy connections between modules. Due the interoperability of electronics, and the presence of many optional modules, the number of architectures generated for the mobile device case study was much larger than for the gas turbine engines (21,168).

The primary conclusions/contributions from this case study are the following:

1. The first conclusion is that while none of 21,168 architectures may be said to be particularly complex compared to existing mobile phones, the success of the concept hinged to a large degree on all potentially feasible configurations operating as expected. In other words, rather than designing a single phone or small family of functionally related mobile devices, the objective was to create a platform which would function as expected for all functionally disparate configurations which were predicted to be feasible. The 21,168 generated for this work relied on a relatively small library of components. The number of feasible configurations would be exponentially larger if a market of third-party modules were to emerge. Thus system integration efforts would have to somehow guarantee that this uncontrolled set of exponentially growing possible

architectures remained feasible.

2. The second conclusion of this work is that while the number of generated feasible configurations is seemingly very large, it represents only a few percent of the number of possible configurations. This means that if a user were to randomly put together modules their likelihood of building a potentially functioning configuration would be on the order of 2 %. This adds another layer design to the platform, since now, not only must potentially feasible configurations work as expected, users must be guided through a sparse set of configurations in a way that allows them to actualize a configuration that matches their preferences.
3. The third and final conclusion of the work is related to the penalty of modularity. Mobile devices are highly integrated devices which often trade performance for battery life. In marketing studies and from public data, during the Google ARA case study it was found that user satisfaction was highly sensitive to battery life of their device. Literature has suggested that modularity often comes with a performance penalty, often associated with the size and weight of interfaces and losses across interfaces. Due to the high degree of modularity of the initial ARA concept and associated losses the utility of the phone would likely have been diminished even if the challenges described in the first two points were overcome.

7.3 Future Work

A number of areas of future work are identified in this section based on the discussion in the limitations section in the methodology section as well as case study specific considerations.

7.3.1 Methodological Future Work

Perhaps the most important area of future work is addressing the state dependency limitation. As described in the methodology section, convex hulls are defined for the library of components a-priori and do not change as components are added to an architecture. This change was not taken account and is a source of false positives, and necessitates the use of rules implemented by the user based on subject matter expertise. One good example of this for the engine case study was the rule that stipulated the need for an oil pump in every oil loop to provide a finite mass flow rate. Previous work with linear formulations of component transfer functions allows these state dependences to be handled directly in the linear programming problem. To enforce state dependency in generic non-linear systems one would have to simulate the space of possible inputs and outputs of partially formed architectures (as we add components and connections when building an architecture). This is a non-trivial problem, both due to the run time associated with repeatedly analyzing thousands of partially formed architectures and due to solver stability when checking partially formed architectures. Geometric programming is an area of active research which relies on functions being expressed via polynomials and allows rapid network optimization. This is perhaps one direction this research can take in the future to further reduce additional constraints imposed by the designer which potentially biases results towards legacy designs.

Another area of future work is related to the staged process in which architecture generation occurs. Feasible component combinations are first generated via user defined constraints. Feasible sets of connections are then searched for given convex hull constraints. Ideally connection and component enumeration should occur simultaneously to remove the need for additional component combination rules during the first stage of exploration, leaving only flow compatibility rules which rely directly on underlying component physics, reducing bias

towards legacy architectures.

An additional area of future work is related to the explicit division between function and form within architecture exploration. For the engine case study, components in general were treated as functions with constraints expressed via convex hulls. For the Google ARA case study, it was assumed that components were existing pieces of hardware that could not be resized.

To provide greater clarity between function and form one potential avenue of future work is to use a formal modeling language to help designers formulate problems that they would like to solve, explicitly dividing function and form. One such language is called “Object-Process Methodology”. The ontology of OPM consists of objects (things which unconditionally exist) and have a state associated with them and processes which can transform the state of objects. The figure below graphically depicts some aspects of OPM. The instrument link relates a process to an object that is required to carry out that process. Arrows from objects (shown via rectangles) to processes (shown via ellipses) represent the situation in which a process consumes an object. Arrows from processes to object represent a result link indicating that the process generates an object. The smaller rectangles within the objects represent the state of the object (e.g. Temperature). We can think of the convex hull grammar described in terms of an OPM diagram as shown in the figure 7.2. The function of a compressor is to increase stagnation pressure. The stagnation pressure increasing process consumes low stagnation pressure flow and produces high stagnation pressure flow while requiring shaft power. The compressor object here is connected to the stagnation pressure increasing process via an instrument link, indicating that a compressor object is required to carry out the stagnation pressure increasing process. The shaft power providing process can be achieved via a turbine. In this case the shaft power providing function takes in the high stagnation pressure flow object and produces the low stagnation pressure flow object. It also

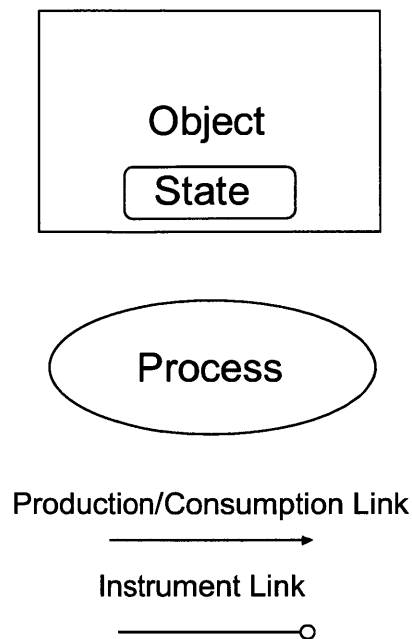


Figure 7.1:

produces the shaft power object which is consumed by the stagnation pressure increasing function. This is a useful way of describing objects and processes (function and form) since it is unambiguous and encourages the designer to consider other ways of achieving the same function. In this formulation the convex hulls described in the thesis are treated as objects.

We can extend the OPM description to the Brayton cycle in general as shown in the figure below in which each of the subfunctions in the cycle are depicted via processes (ellipses) and the objects that they consume/produce are the convex hulls that were described throughout this thesis. Notice that the shaft power providing functions can be achieved via a turbine or a electrical motor each of which will have different objects (gas flow vs. electrical power) which they produce/consume. Currently convex hulls are coded into data structures directly prior to design space exploration. In the proposed future approach OPM would be used as a graphical way of formulating the architecture exploration problem that utilizes convex hulls.

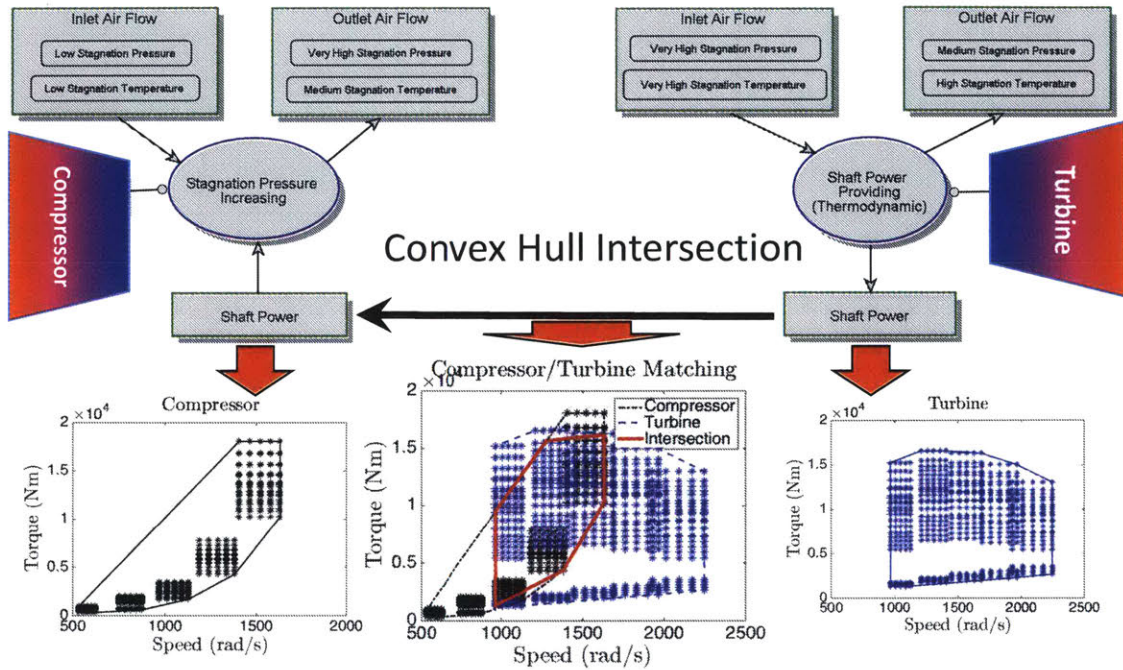


Figure 7.2: Object-Process Methodology to explicitly capture function and form

Not much has been discussed in this work about human interaction with design space exploration. One area of future work is allowing subject matter experts to interact with the design space exploration process during generation of feasible architectures. While human interaction can cause bias, subject matter experts can also provide invaluable input to reduce false positives. If humans play an active part in architecture generation they can eliminate complete or partially formed architectures which are infeasible for reasons which are not captured in the model. In addition humans can benefit from interacting with architecture generation in this way since it compels them to consider new concepts. It is the opinion of the author that such interaction could kindle rather than hinder creativity.

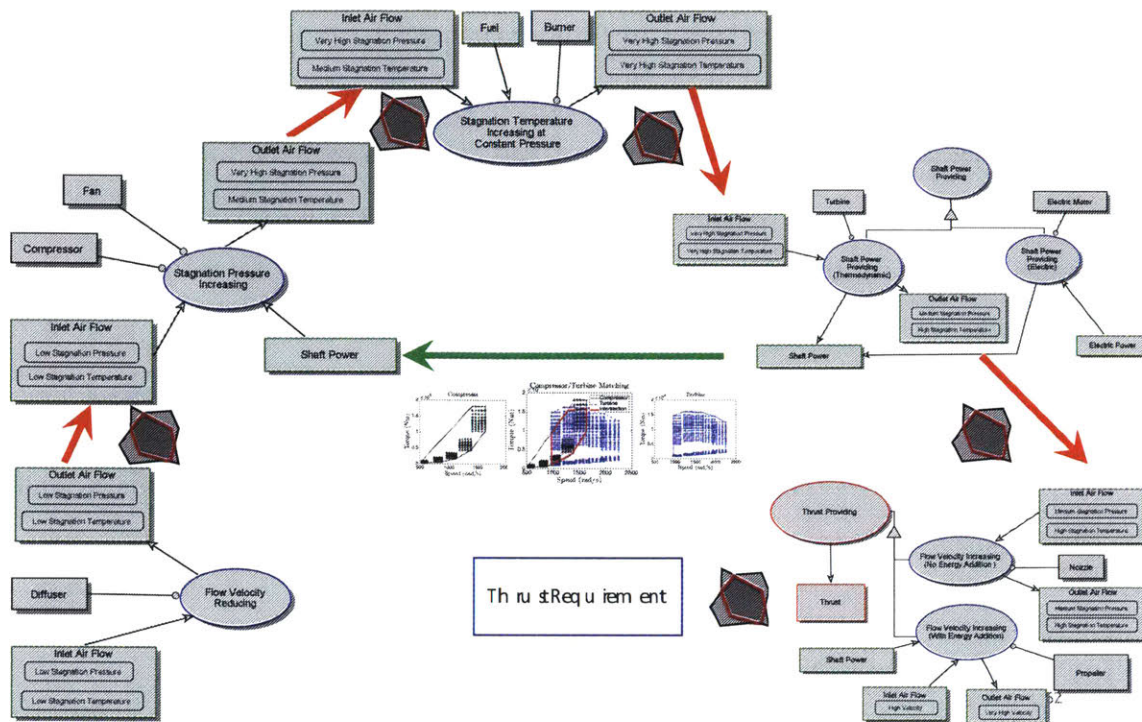


Figure 7.3: Object-Process Methodology to explicitly capture function and form

7.3.2 Engine Case Study Future Work

A primary area of future work for the engine case study is airframe integration that will allow boundary layer ingestion benefits and other system integration benefits to be taken full advantage of. This will allow system level scenarios for different transfer efficiencies and different propulsor placements to be examined explicitly and inform technology roadmapping efforts. Another area of future work for the engine case study is the addition of transient engine performance optimization using on board electrical energy storage. Finally taking into account noise levels from different engine configurations in different operational regimes will allow examination of situations in which noise can be limited during takeoff in noise-sensitive areas by relying more on low pressure ratio electrical propulsors (fans or propellers).

Another area of future work will be to define the most useful hybrid-electric propulsion system demonstrator that could lead to future commercial hybrid aircraft. This would require selection of a non-dominated hybrid-electric propulsion architecture, integrating it into an existing carrier aircraft and defining a test campaign over the full flight envelope. From a modeling perspective it will involve taking into account electrical energy storage and optimizing over the entire flight envelope and perhaps even defining a new flight envelope that could take full advantage of such a novel architecture.

One final area of future work will be to expand the library of components to include superconducting electric motors and generators. Adding these components will require adding cryogenic cooling, which was out of the scope of this thesis.

7.3.3 Reconfigurable Mobile Device Future Work

For the reconfigurable mobile device case study a number of avenues for future work exist. Further research into quantifying the complexity of families of architectures/platforms is required. One of the challenges of any such platform is the complexity associated with the vast and largely uncontrolled set of possible configurations enabled by it. Examining complexity and its relationship with development effort for such situations is in the opinion of the author of vital importance for the success of future reconfigurable hardware platforms. Ensuring that consumers can take advantage of the customizability/reconfigurability with tools which guide them through the sparse but still very large set of feasible configurations is another aspect of vital importance.

Bibliography

- [1] C. J. Petrie. Automated Configuration Problem Solving. *Springer Briefs in Computer Science*, 2012.
- [2] P. Eremenko. Formal Model-Based Design & Manufacture: A Template for Managing Complexity in Large-Scale Cyber-Physical Systems. In *Conference on Systems Engineering Research*. Defense Advanced Research Projects Agency, 2013. URL http://www.cser13.gatech.edu/sites/default/files/attachments/cser13_{_}keynote_{_}eremenko.pdf.
- [3] N. Augustine. *Augustin's Laws*. American Institute of Aeronautics and Astronautics Press, 6 edition, 1997.
- [4] M. Arena, O. Younossi, K. Brancato, I. Blickstein, and C. Grammich. Why Has The Cost of Fixed Wing Aircraft Risen? A macroscopic Examination in U.S. Military Aircraft Costs over the Past Several Decades. Technical report, RAND Corporation, 2008.
- [5] C. Gorbea. *Vehicle Architecture and Life Cycle Cost Analysis In a New Age of Architectural Competition*. PhD thesis, Der Technische Universität München, 2011.
- [6] P. Schulz, P. Clausing, E. Fricke, and H. Negele. Development and Integration of Winning Technologies as Key to Competitive Advantage. *Systems Engineering*, 3:180–211, 2000.

- [7] Lycoming XR-7755-3, Radial 36 Engine, Museum Smithsonian National Air and Space, 2016. URL <https://airandspace.si.edu/collection-objects/lycoming-xr-7755-3-radial-36-engine>.
- [8] M. Daly and B. Gunston. Jane's Aero-Engines, Issue Twenty-eight 2010, Inc., 2010 by Jane's Information Group Limited, 2010.
- [9] SAE. Pratt & Whitney's next leap in engine technology, 2016. URL <https://www.sae.org/aeromag/techinnovations/1298t10.htm>.
- [10] E. M. Greitzer, P. A. Bonnefoy, E. la Rosa Blanco, C. S. Dorbian, M. Drela, D. K. Hall, R. J. Hansman, J. I. Hileman, R. H. Liebeck, J. Lovergren, P. Mody, J. A. Pertuze, S. Sato, Z. S. Spakovszky, C. S. Tan, J. S. Hollman, J. E. Duda, N. Fitzgerald, J. Houghton, J. L. Kerrebrock, G. F. Kiwada, D. Kordonowy, C. J. Parrish, J. Tylko, A. E. Wen, and W. K. Lord. N+3 Aircraft Concept Designs and Trade Studies Final Report. 2010.
- [11] M. Mohan, J. J. Shah, S. Narsale, and M. Khorshidi. Capturing Ideation Paths for Discovery of Design Exploration Strategies in Conceptual Engineering Design. In *Design Computing and Cognition'12*, pages 589–604. Springer, 2014.
- [12] O. de Weck. 16.842 Fundamentals of Systems Engineering, Massachusetts Institute of Technology, 2015.
- [13] KC10. URL <http://media.defense.gov/2003/Apr/23/2000031242/-1/-1/0/030317-F-7203T-013.JPG>.
- [14] KC10 Refueling F/A 18. URL <http://ep.yimg.com/ay/mcmahanphoto/f-18-hornet-marauders-kc-10-refueling-photo-print-4.jpg>.
- [15] F/A 18 Superhornet. URL <http://www.oregonairshow.com/wp-content/uploads/>

2013/02/super-hornet-3.jpg.

- [16] J. Kurzke. Map Collection 2. <http://www.gasturb.de/index.html>, 2010.
- [17] J. Kurzke. Gasturb11. <http://www.gasturb.de/index.html>, 2010.
- [18] Unified Engineering, Massachusetts Institute of Technology. URL <http://web.mit.edu/16.unified/www/SPRING/propulsion/notes/node131.html>.
- [19] Google ARA Spiral 2 Diagram, 2016. URL <https://www1-lw.xda-cdn.com/files/2014/04/ara.jpg>.
- [20] O. de Weck. Modularity: Fundamental Principles and Application to the Google Ara Platform, Google ARA DevCon2, 2015.
- [21] O. de Weck, K. Sinha, N. R. Shougarian, E. F. Colombo, I. Josan-Drinceanu, and O. Willner. Google ARA Final Report, Massachusetts Institute of Technology, 2016.
- [22] E. F. Colombo. *Open innovation meets Changeability: Strategic design analyses for Cyber-physical Industry platforms*. PhD thesis, Politecnico di Milano, 2016.
- [23] (University of Michigan) J. Penner, D. Lister, (Defense Research and Evaluation Agency), (UK Meteorological Office) D. Griggs, (University Corporation for Atmospheric Research) D. Dokken, and (Dupont Fluoroproducts) M. McFarland. Ipc special report. Technical report, United Nations Intergovernmental Panel on Climate Change, 1999. URL <https://www.ipcc.ch/pdf/special-reports/spm/av-en.pdf>.
- [24] ATAG. Air Transport Action Group (ATAG) Environmental Summit, 2008. URL <http://www.atag.org/our-activities/climate-change.html>.
- [25] ICAO. ICAO Environmental Report 2010. Technical report, 2010. URL <http://www.icao.int/environmental-protection/Documents/Publications/ENV{ }Report{ }2010.pdf>.

- [26] D. Kellari. *What's next for the airliner? Historical Analysis and Future Predictions of Aircraft Architecture and Performance*. Masters thesis, Massachusetts Institute of Technology, 2016.
- [27] K. Sinha. *Structural Complexity and its Implications for Design of Cyber- Physical Systems*. PhD thesis, Massachusetts Institute of Technology, 2014.
- [28] D. Raymer. *Enhancing Aircraft Conceptual Design Using Multidisciplinary Optimization*. PhD thesis, Royal Institute of Technology Stockholm, 2002.
- [29] C. Riegler and C. Bichlmaier. The Geared Turbofan Technology – Opportunities, Challenges, and Readiness Status. *Proceedings of 1st CEAS European Air and Space Conference, Berlin, Germany, 10-13 September., 2007*.
- [30] A. H. Epstein. Aeropropulsion for Commercial Aviation in the Twenty-First Century and Research Directions Needed. *AIAA*, 52(5), 2014.
- [31] MIT Gas Turbine Laboratory. Early Gas Turbine History MIT Gas Turbine Laboratory, 2016. URL <http://web.mit.edu/aeroastro/labs/gtl/early{ }GT{ }history.html>.
- [32] N. Lang. Aurel Stodola and his influence on the ETH and on Mechanical Engineering. Technical report, ETH Zurich, 2016.
- [33] M. Guillaume. Patent Number 534.801, Office National De La Propriete Industrielle, France, 1921.
- [34] J. Golley. *Genesis: Frank Whittle and the Invention of the Jet Engine*. The Crowood Press, 1997.
- [35] Z. Spakovszky, F. Ehrich, and J. Sabnis. 16.511 Aircraft Engines and Gas Turbines Lecture Notes, Massachusetts Institute of Technology, 2016.

- [36] N Cumpsty. *Jet Propulsion: A simple guide to the aerodynamic and thermodynamic design and performance of jet engines*. Cambridge University Press, 1997.
- [37] Paul Barton. *Mixed Integer and Nonconvex Optimization*. Massachusetts Institute of Technology, 2011.
- [38] Rolls-Royce Civil Aerospace. URL <http://www.rolls-royce.com/products-and-services/civil-aerospace/products/civil-large-engines.aspx>.
- [39] 30 Years in the Making, A Simple Gearbox is Posed to Change the Jet Engine. URL <http://www.popularmechanics.com/flight/a17813/purepower-gtf-coming-to-market/>.
- [40] E. Murman, E. M. Greitzer, J. L. Craig, and J. Deyst. *16.621 Lecture Notes: Development of Hypothesis, Objective and Success Criteria*. Massachusetts Institute of Technology, 2003.
- [41] E. Crawley, B. Cameron, and D. Selva. *System Architecture: Strategy and Product Development for Complex Systems*. Prentice Hall Press, 2015.
- [42] E. Crawley. Introduction to System Architecture Architecture to Value, System Architecture Lecture Notes, 2007. URL <https://ocw.mit.edu/courses/engineering-systems-division/esd-34-system-architecture-january-iap-2007/lecture-notes/lec1.pdf>.
- [43] S. J. Kapurch. *NASA Systems Engineering Handbook*. DIANE Publishing, 2010.
- [44] V. S. Johnson. Optimizing conceptual aircraft designs for minimum life cycle cost. *In its Recent Advances in Multidisciplinary Analysis and Optimization, Part 3 p 1195-1217(SEE N 89-25201 19-05)*, 1989.

- [45] N P Suh. *The Principles of Design*, volume 990. Oxford University Press New York, 1990.
- [46] J A Barton, D M Love, and G D Taylor. Design determines 70% of cost? A Review of Implications for Design Evaluation. *Journal of Engineering Design*, 12(1):47–58, 2001.
- [47] K. D. II Sobek, C. A. Ward, and K. J. Liker. Toyota’s Principles of Set-Based Concurrent Engineering. *Sloan Management Review*, 40(2), 1999.
- [48] J. Genta. *Using the Principles of Set-Based Design to Realize Ship Design Process Improvement*. Masters thesis, Massachusetts Institute of Technology, 2016.
- [49] J. C. Walker. *Multi-Attribute Tradespace Exploration for US Navy Surface Ship Survivability: A Framework for Balancing Capability, Survivability, and Affordability*. Masters thesis, Massachusetts Institute of Technology, 2016.
- [50] J. I. Bernstein. *Design Methods in the Aerospace Industry: Looking for Evidence of Set-Based Practices*. Masters thesis, Massachusetts Institute of Technology, 1998.
- [51] L. E. Zeidner, H. M. Reeve, R. Khire, and S. Becz. Architectural Enumeration and Evaluation for Identification of Low-Complexity Systems. In *13th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, Fort Worth, TX, AIAA, AIAA-2010-9264*, 2010.
- [52] L. E. Zeidner, B. E. Rock, N. A. Desai, H. M. Reeve, and M. Strauss. Application of a Technology Screening Methodology for Rotorcraft Alternative Power Systems. Number January 2010. 48th AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition, 2015. ISBN 9781600867392.
- [53] F. Zwicky. The morphological approach to discovery, invention, research and construction. In *New methods of thought and procedure*, pages 273–297. Springer, 1967.

- [54] M. Mohan, Y. Chen, and J. J. Shah. Towards a Framework for Holistic Ideation in Conceptual Design. In *ASME 2011 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, pages 661–672. American Society of Mechanical Engineers, 2011.
- [55] J. J. Shah, S. M. Smith, and N. Vargas-Hernandez. Metrics for measuring ideation effectiveness. *Design studies*, 24(2):111–134, 2003.
- [56] S. W. Miller, T. W. Simpson, and M. A. Yukish. Design as a Sequential Decision Process: A Method for Reducing Design Set Space Using Models to Bound Objectives. In *ASME 2015 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference IDETC/CIE 2015 August 2-5, 2015*, Boston, Massachusetts, USA, 2015.
- [57] D. Selva. *Rule-Based System Architecting of Earth Observation Satellite Systems* by. Phd, Massachusetts Institute of Technology, 2012.
- [58] J. Agte, O. de Weck, J. Sobieszczanski-Sobieski, P. Arendsen, A. Morris, and M. Spieck. MDO: Assessment and Direction for Advancement—An Opinion of One International Group. *Structural and Multidisciplinary Optimization*, 40(1-6):17–33, 2010.
- [59] J. Agte. *Multistate Analysis and Design: Case Studies in Aerospace Design and Long Endurance Systems*. PhD thesis, Massachusetts Institute of Technology, 2011.
- [60] N. R. Shougarian, J. Agte, and O. de Weck. Multistate Analysis and Optimization of a Geared Turbofan Engine Lubrication System. In *12th AIAA Aviation Technology, Integration, and Operations (ATIO) Conference and 14th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, Indianapolis, Indiana*, 2012.
- [61] J. Gu, P. W. Purdom, J. Franco, and B.W. Wah. *Algorithms for the Satisfiability (SAT) Problem*. Springer, 1999.

- [62] R. J. Bayardo Jr and R. Schrag. Using CSP Look-Back Techniques to Solve Real-World SAT Instances. In *AAAI/IAAI*, pages 203–208, 1997.
- [63] H. Zhang. SATO: An Efficient Propositional Prover. In *Automated Deduction—CADE-14*, pages 272–275. Springer, 1997.
- [64] M. W. Moskewicz, C. F. Madigan, Y. Zhao, L. Zhang, and S Malik. Chaff: Engineering an Efficient SAT Solver. In *Proceedings of the 38th annual Design Automation Conference*, pages 530–535. ACM, 2001.
- [65] L. Zhang and S. Malik. The Quest for Efficient Boolean Satisfiability Solvers. In *Computer Aided Verification*, pages 17–36. Springer, 2002.
- [66] E. Goldberg and Y. Novikov. BerkMin: A Fast and Robust SAT-Solver. *Discrete Applied Mathematics*, 155(12):1549–1561, 2007.
- [67] B. C. Williams and R. J. Ragno. Conflict-Directed A* and its Role in Model-Based Embedded Systems. In *Journal of Discrete Applied Mathematics*. Citeseer, 2003.
- [68] B C Williams. 16.413 Principles of Autonomy and Decision Making Lecture Notes, Massachusetts Institute of Technology. 2012.
- [69] S. Neema, J. Sztipanovits, G. Karsai, and K. Butts. Constraint-Based Design-Space Exploration and Model Synthesis. In *Embedded Software*, pages 290–305. Springer, 2003.
- [70] S. Do and O. de Weck. HAB.NET – An Integrated Framework for Analyzing the Sustainability of Planetary Habitats. *GLEX-2012.10.2.6x12391.*, 2012.
- [71] S Do and O de Weck. A Grammar for Encoding and Synthesizing Life Support System Architectures using the Object Process Methodology. *44th International Conference on Environmental Systems*, 2014.

- [72] D. Dori. *Object-Process Methodology*. Springer, 1 edition, 2002.
- [73] M. Heinrich and E. W. Jungst. The Resource-Based Configuring Technical Systems Paradigm: from Modular Components. Technical report, AAAI Technical Report FS-96-03, 1996.
- [74] T. Ishimatsu. *Generalized Multi-Commodity Network Flows: Case Studies in Space Logistics and Complex Infrastructure Systems*. PhD thesis, Massachusetts Institute of Technology, 2013.
- [75] K. Ho. *Dynamic Network Modeling for Spaceflight Logistics with Time-Expanded Networks*. Phd, Massachusetts Institute of Technology, 2015.
- [76] R. J. Piacenza, S. Proper, M. A. Bozorgirad, I. Y. Tumer, and C. Hoyle. Robust Topology Design of Complex Infrastructure Systems. 2015.
- [77] S. Twu and K. Choi. Configuration Design Sensitivity Analysis for Design Optimization. *Computer and Systems Sciences Concurrent Engineering: Tools and Technologies for Mechanical System Design*, 108, 1992.
- [78] J. C. Peralta and J. P. Gallagher. Convex Hull Abstractions in Specialization of CLP Programs. In *Logic Based Program Synthesis and Transformation*, pages 90–108. Springer, 2003.
- [79] W. L. Simmons. *A Framework for Decision Support in Systems Architecting*. PhD thesis, Massachusetts Institute of Technology, 2008.
- [80] Design Structure Matrix. URL <http://www.dsmweb.org/>.
- [81] L. Schrenk. Development of an ISRU Analysis Module for HabNet, 2015.
- [82] P. Norvig and S. Russel. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2009.

- [83] L. K. Zhang. Product configuration: a review of the state of art and future research. *International Journal of Production Research*, 2014.
- [84] J. R. Kent, W. E. Carlson, and R. E. Parent. Shape transformation for polyhedral objects. In *SIGGRAPH '92 Proceedings of the 19th annual conference on Computer graphics and interactive techniques*, 1992.
- [85] P. Fritzson. Introduction to Object-Oriented Modeling and Simulation with Modelica Using OpenModelica, 2012.
- [86] K. Gwang, K. Junjung, S. S. Eun, and A. Jaemyung. Correlation between Architectural Complexity of Engineering Systems and Actual System Design Effort. *ASME Journal of Mechanical Design*, 2016.
- [87] D. Vogt. Turbomachinery Lecture Notes KTH, Course MJ2429, 2007. URL <http://www.energy.kth.se>.
- [88] F. Schwarz, W. Sheridan, and G. Levasseur. Geared turbofan high gearbox power density, 2014. URL <https://www.google.com/patents/WO2014055122A1?cl=en>.
- [89] Y. Cengel. *Heat Transfer: A Practical Approach*. Mcgraw-Hill, 2 edition, 2002.
- [90] J. Welstead, J. Felder, and M. Patterson. Conceptual Design of a Single-Aisle Turboelectric Commercial Transport with Fuselage Boundary Layer Ingestion, 2013. URL http://www.nianet.org/ODM/ODMTuesdaypresentationsFinal/19Pattersonstarcabl_{_}scitech2016_{_}shortened_{_}distribution_{_}ODM.pdf.
- [91] Heat Transfer from a Fin, Unified Engineering, Massachusetts Institute of Technology. URL <http://web.mit.edu/16.unified/www/FALL/thermodynamics/notes/node128.html>.

- [92] Genetic Algorithm. URL <https://www.mathworks.com/discovery/genetic-algorithm.html>.
- [93] A. P. Plas, M. A. Sargeant, V. Madani, D. Crichton, and E. M. Greitzer. Performance of a Boundary Layer Ingesting (BLI) Propulsion System. In *45th AIAA Aerospace Sciences Meeting and Exhibit*, 2007.
- [94] Google. Google Advanced Technology and Projects, 2016. URL <https://atap.google.com/ara/>.

Appendix

Engine Case Study

We provide all architectures generated at level two of abstraction for completeness. In addition, we provide a table of assumptions for parameters which were not explicitly listed in the main body of this work below.

Parameter	Value	Reference
Electric Motor and Inverter Power To Mass Ratio	$7.4KWkg^{-1}$	[90]
Electric Generator Power To Mass Ratio	$13.2KWkg^{-1}$	[90]
Gearbox Power to Mass Ratio	$100hplb^{-1}$	[88]

Table 7.1: Geometric and efficiency parameters based primarily on [8].

Component	k	w	Isentropic Efficiency	Efficiency	Aggregate Density
Inlet	NA	0.98 (Does not include nacelle)	NA	NA	$345(Cruise Thrust)^{-0.53}2200 (kg/m^3)$
Bypass Nozzle	NA	0.98 (Does not include nacelle)	NA	NA	$345(Cruise Thrust)^{-0.53}2200 (kg/m^3)$
Fan	0.25	NA	0.9	NA	$345(Cruise Thrust)^{-0.53}440 (kg/m^3)$
Gearbox	NA	NA		0.995	NA
Electric Motor	NA	NA		0.9 (0.95)	NA
Electric Generator	NA	NA		0.9 (0.95)	NA
Compressor	0.85 (inlet) 0.95 (outlet)	NA	0.9	NA	$345(Cruise Thrust)^{-0.53}3300 (kg/m^3)$
Burner	NA	NA		NA	$345(Cruise Thrust)^{-0.53}3300(kg/m^3)$
Turbine	0.95 (inlet) 0.8 (outlet)	0.95 (inlet) 0.8 (outlet)	0.9	NA	$345(Cruise Thrust)^{-0.53}3300(kg/m^3)$

Table 7.2: Geometric and efficiency parameters based primarily on [8].

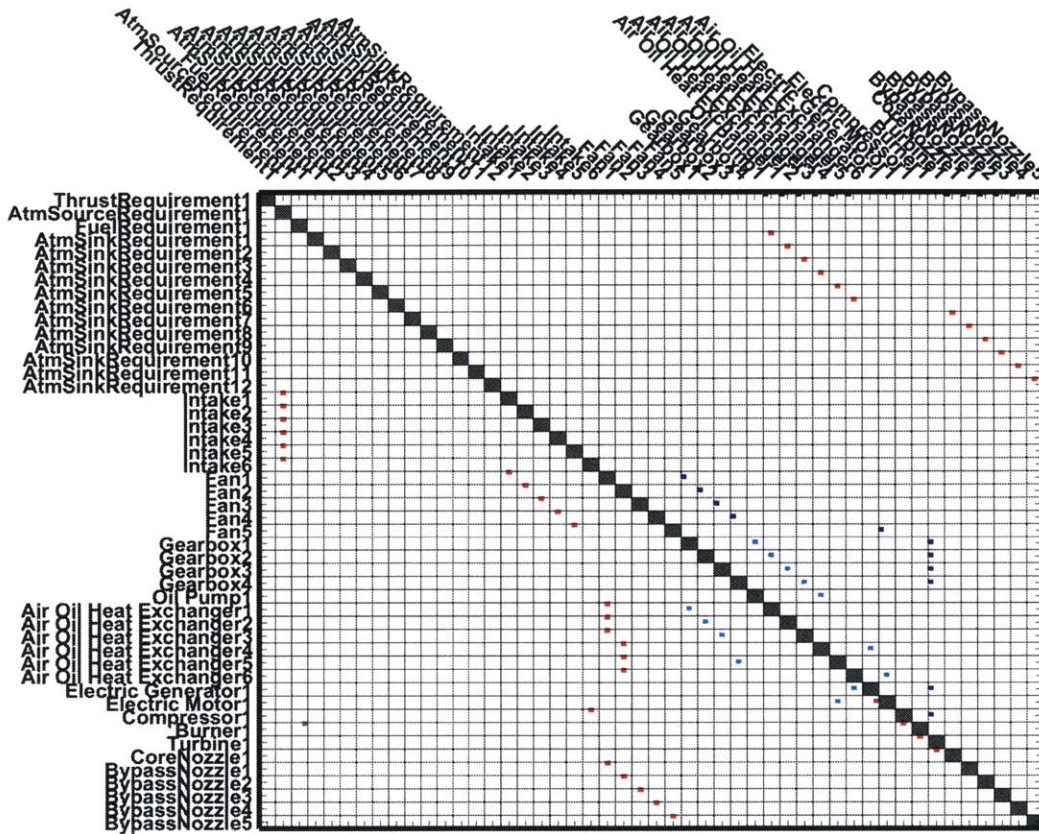


Figure 7.4: Architecture 1 (Turboelectric). Gas connections: **red**, shaft power connections: **blue**, mechanical connections: **black**, fuel connections: **green**, oil connections: **cyan**, electrical connections: **magenta**.

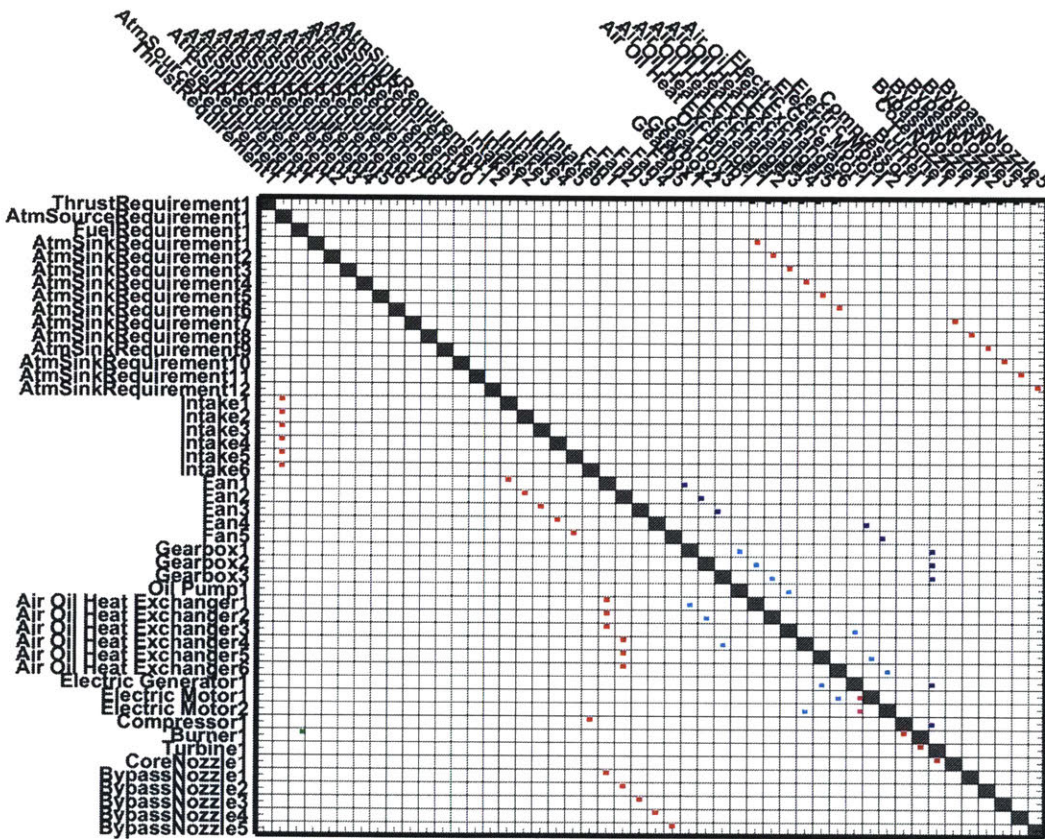


Figure 7.5: Architecture 2 (Turboelectric). Gas connections: red, shaft power connections: blue, mechanical connections: black, fuel connections: green, oil connections: cyan, electrical connections: magenta.

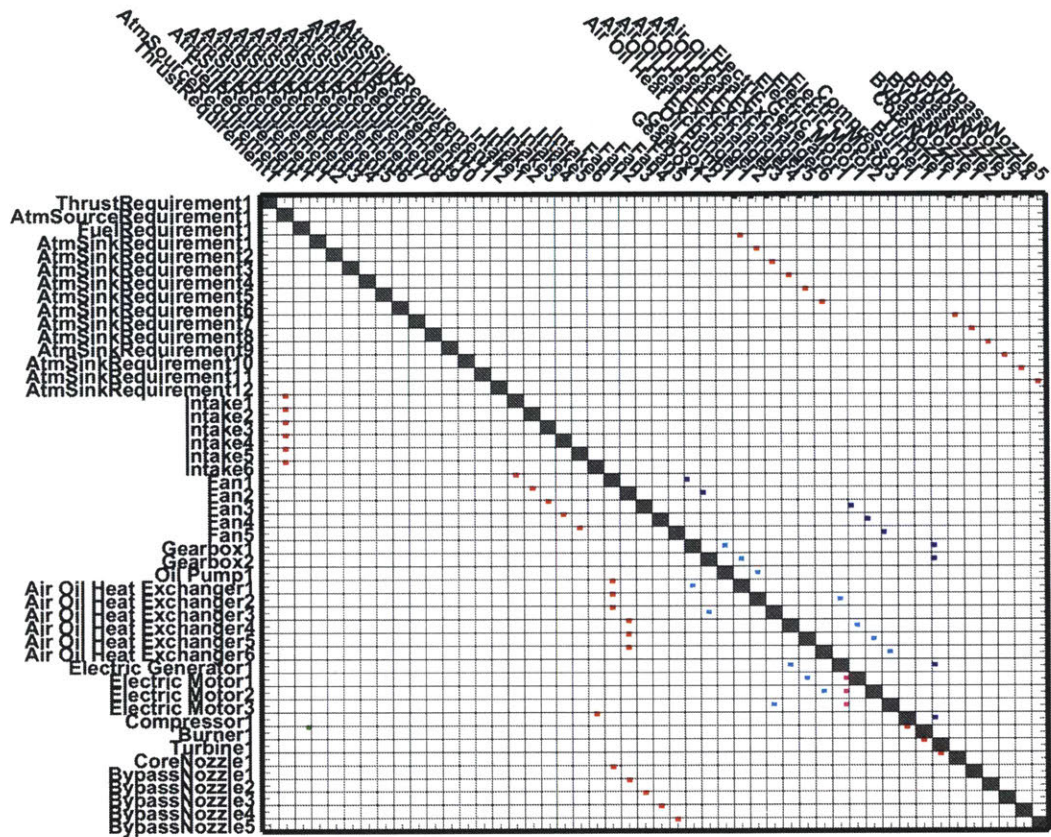


Figure 7.6: Architecture 3 (Turboelectric). Gas connections: red, shaft power connections: blue, mechanical connections: black, fuel connections: green, oil connections: cyan, electrical connections: magenta.

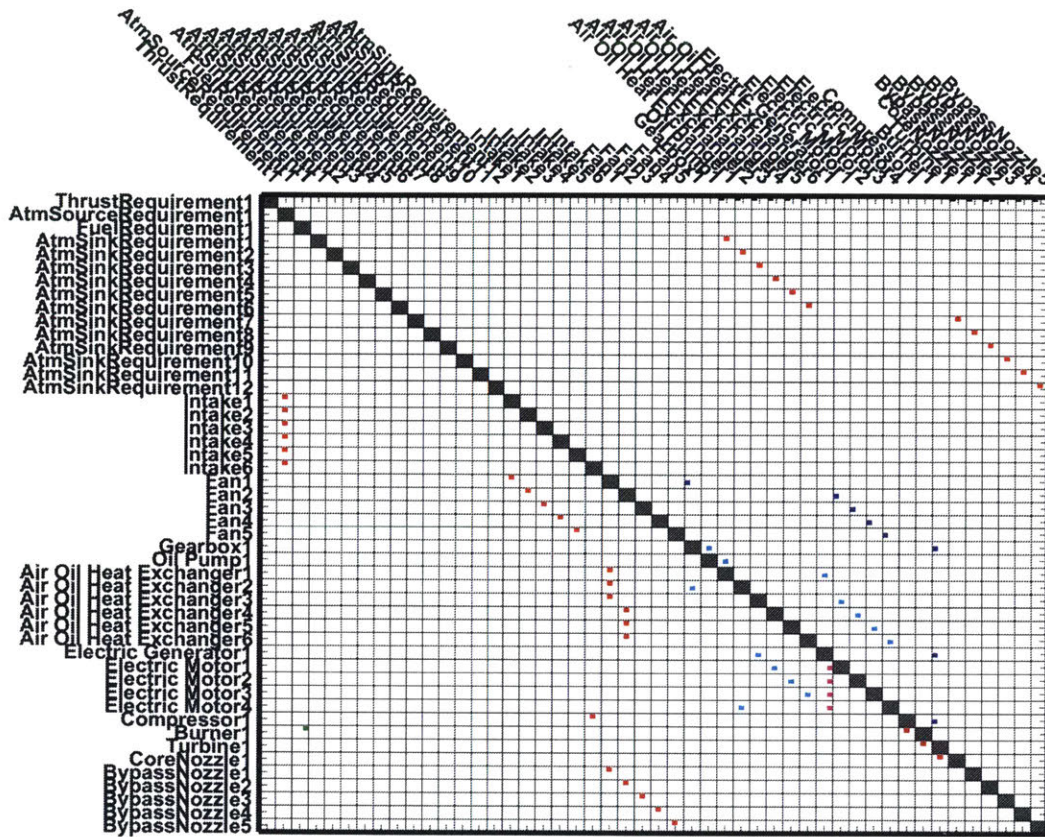


Figure 7.7: Architecture 4 (Turboelectric). Gas connections: **red**, shaft power connections: **blue**, mechanical connections: **black**, fuel connections: **green**, oil connections: **cyan**, electrical connections: **magenta**.

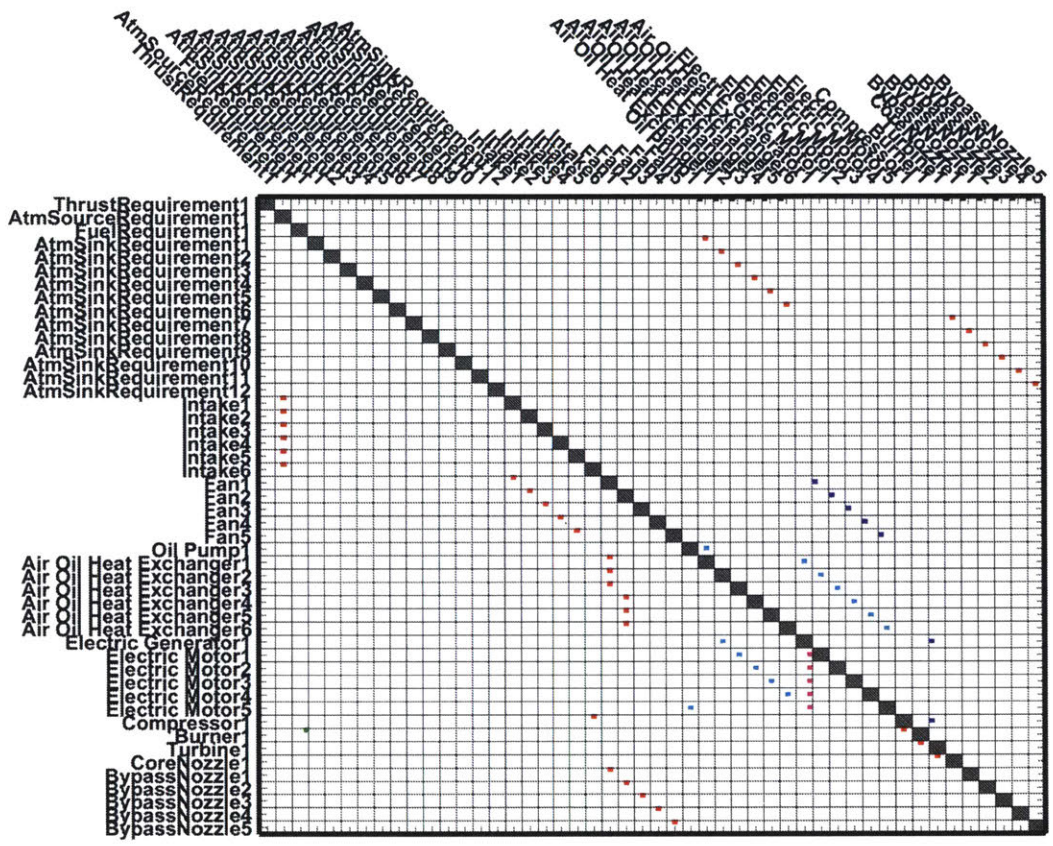


Figure 7.8: Architecture 5 (Turboelectric). Gas connections: **red**, shaft power connections: **blue**, mechanical connections: **black**, fuel connections: **green**, oil connections: **cyan**, electrical connections: **magenta**.

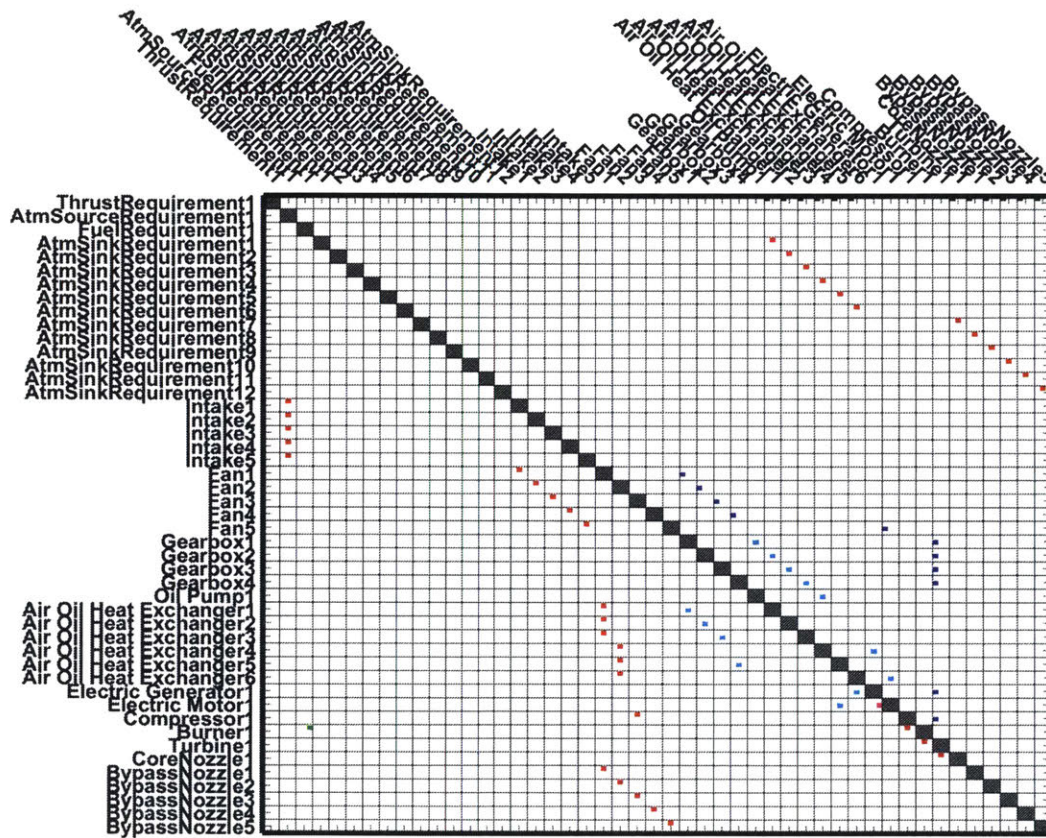


Figure 7.9: Architecture 6 (Turboelectric). Gas connections: **red**, shaft power connections: **blue**, mechanical connections: **black**, fuel connections: **green**, oil connections: **cyan**, electrical connections: **magenta**.

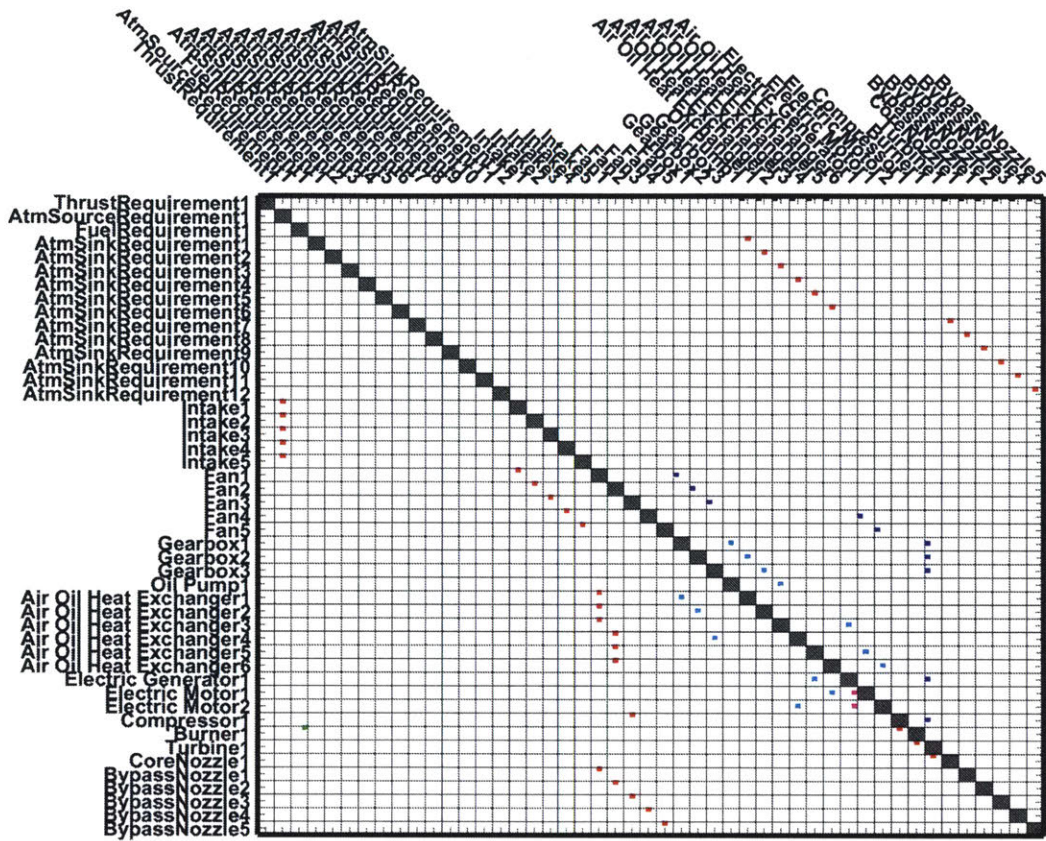


Figure 7.10: Architecture 7 (Turboelectric). Gas connections: **red**, shaft power connections: **blue**, mechanical connections: **black**, fuel connections: **green**, oil connections: **cyan**, electrical connections: **magenta**.

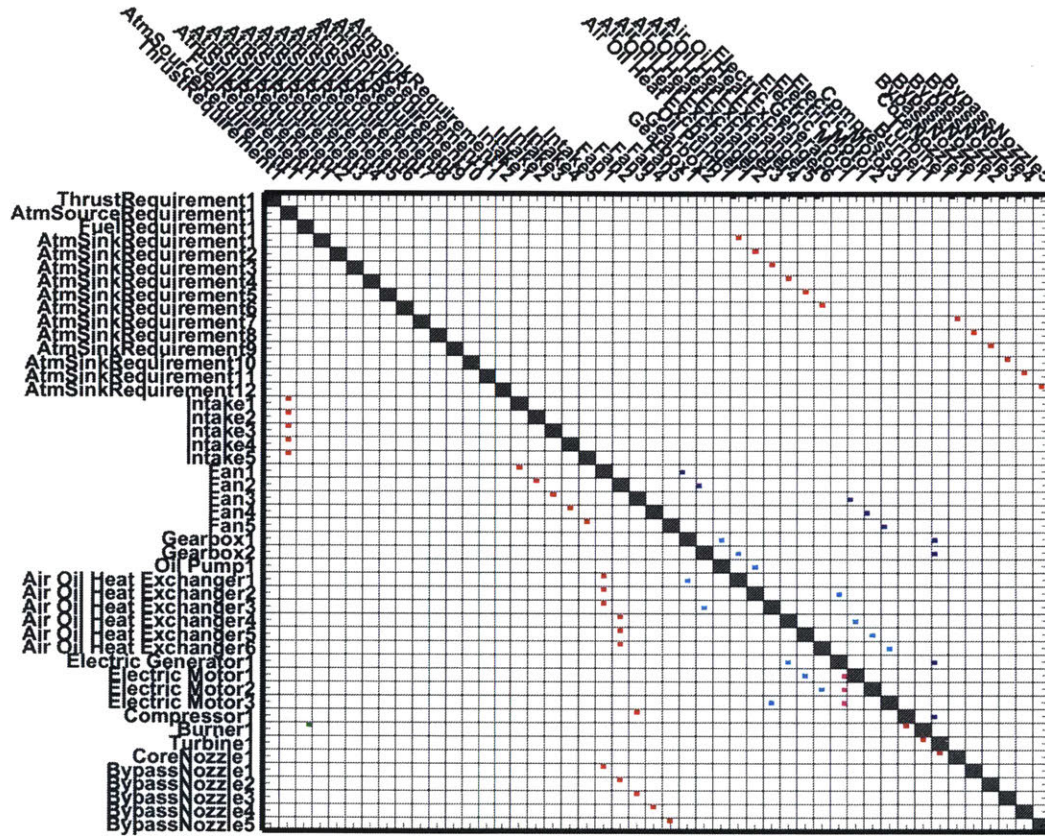


Figure 7.11: Architecture 8 (Turboelectric). Gas connections: **red**, shaft power connections: **blue**, mechanical connections: **black**, fuel connections: **green**, oil connections: **cyan**, electrical connections: **magenta**.

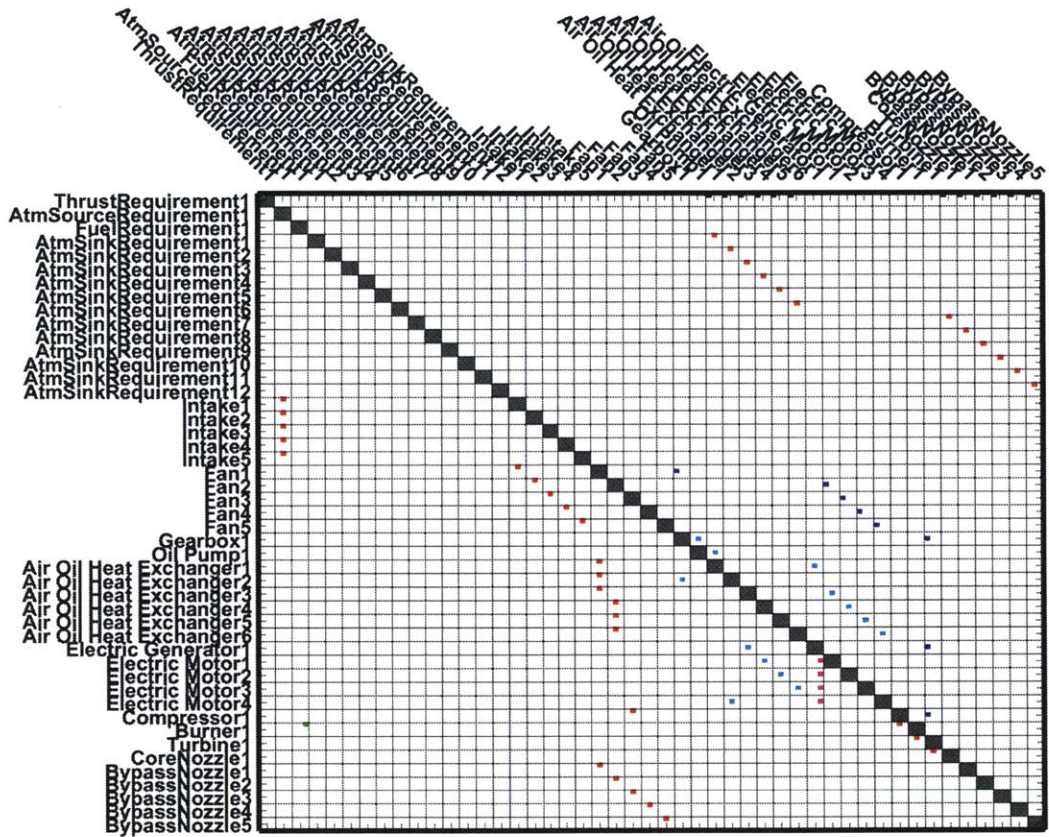


Figure 7.12: Architecture 9 (Turboelectric). Gas connections: red, shaft power connections: blue, mechanical connections: black, fuel connections: green, oil connections: cyan, electrical connections: magenta.

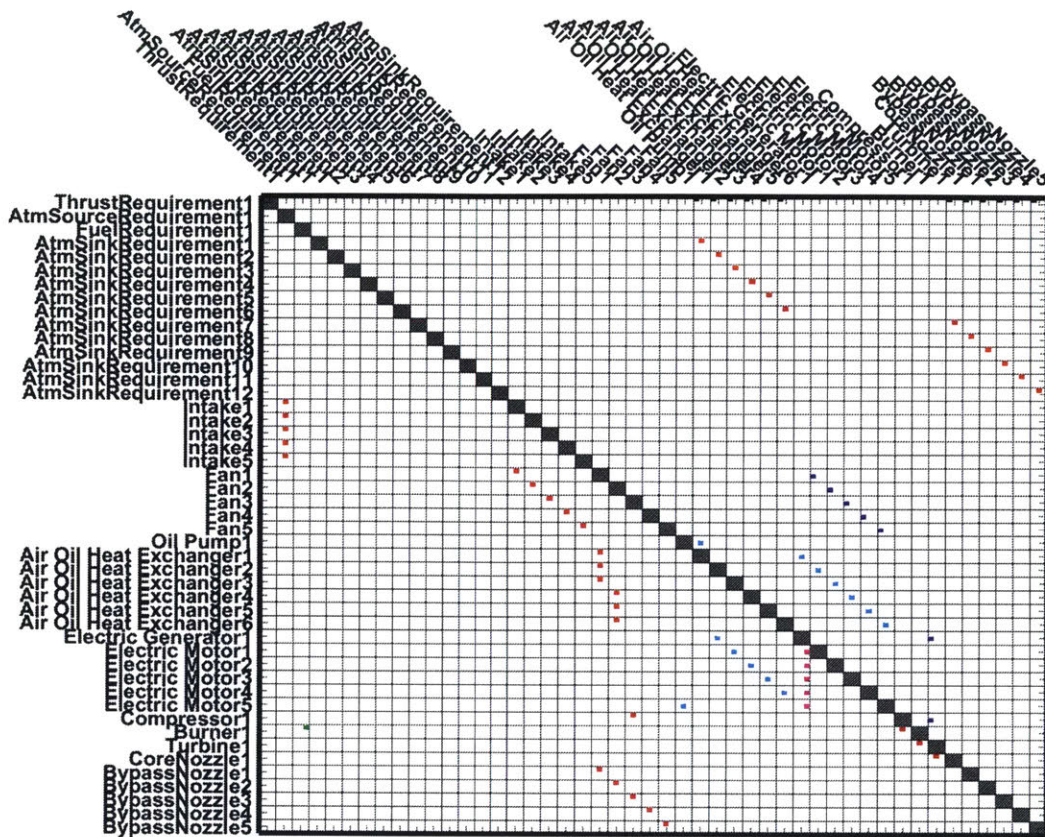


Figure 7.13: Architecture 10 (Turboelectric). Gas connections: **red**, shaft power connections: **blue**, mechanical connections: **black**, fuel connections: **green**, oil connections: **cyan**, electrical connections: **magenta**.

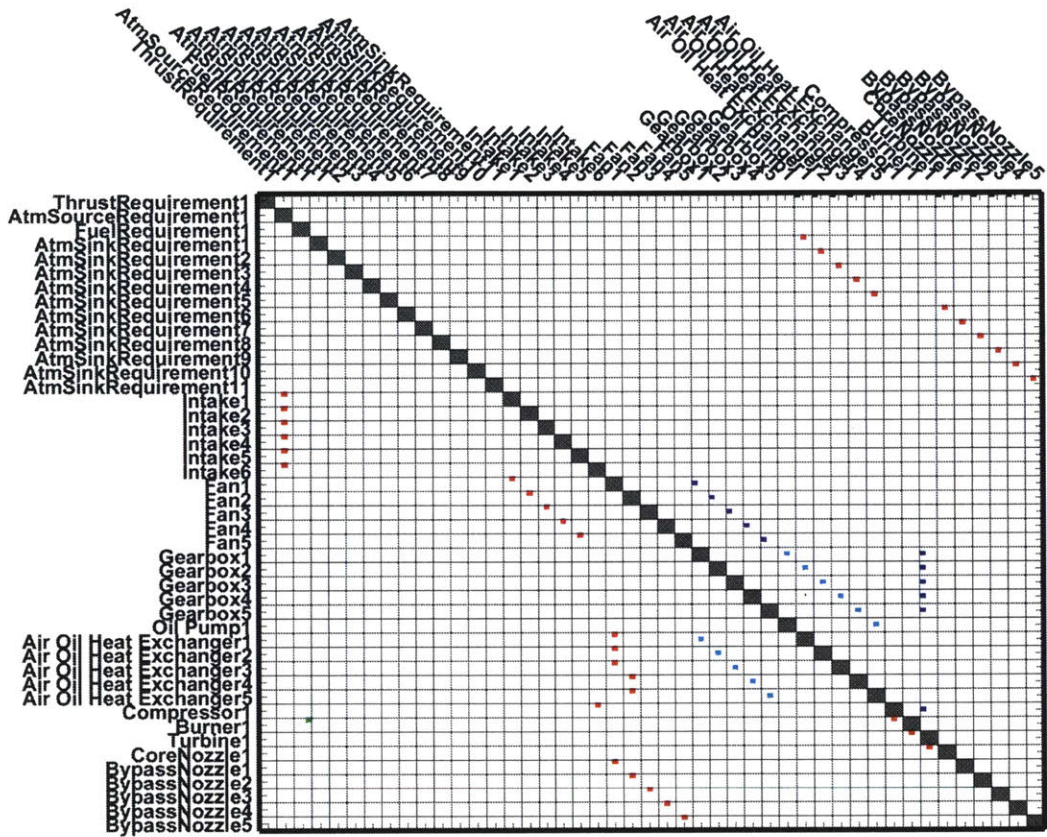


Figure 7.14: Architecture 11. Gas connections: **red**, shaft power connections: **blue**, mechanical connections: **black**, fuel connections: **green**, oil connections: **cyan**, electrical connections: **magenta**.

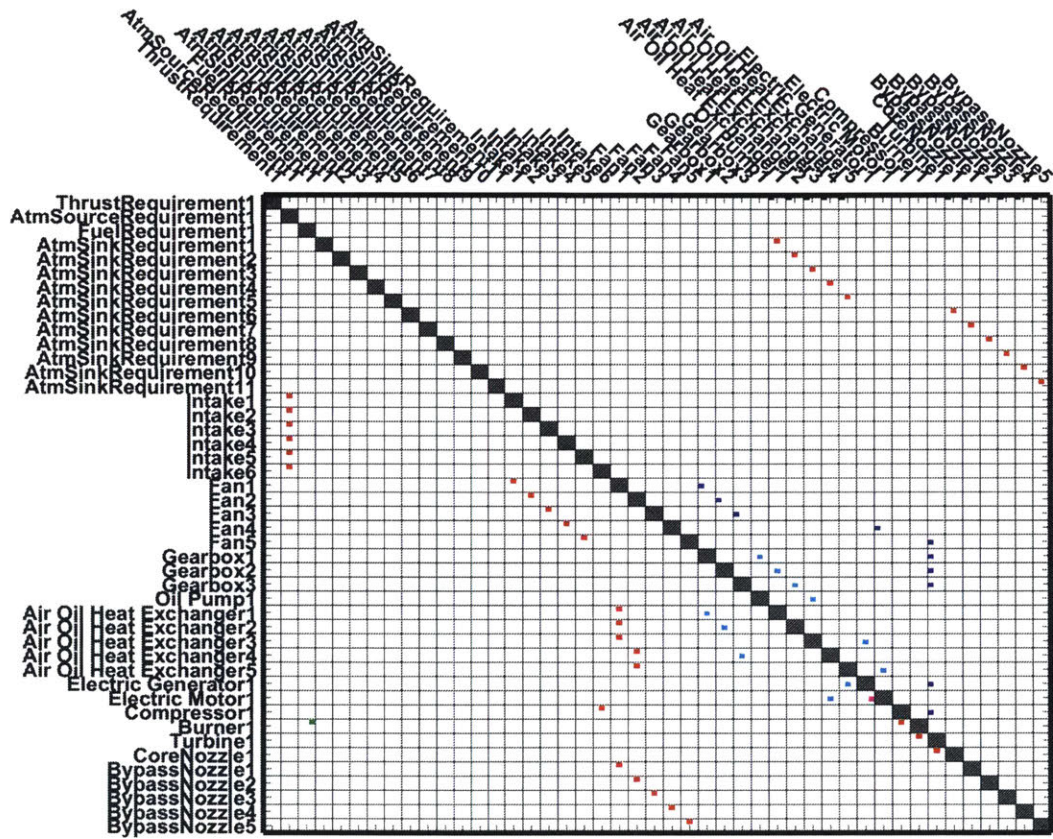


Figure 7.15: Architecture 12 (Turboelectric). Gas connections: **red**, shaft power connections: **blue**, mechanical connections: **black**, fuel connections: **green**, oil connections: **cyan**, electrical connections: **magenta**.

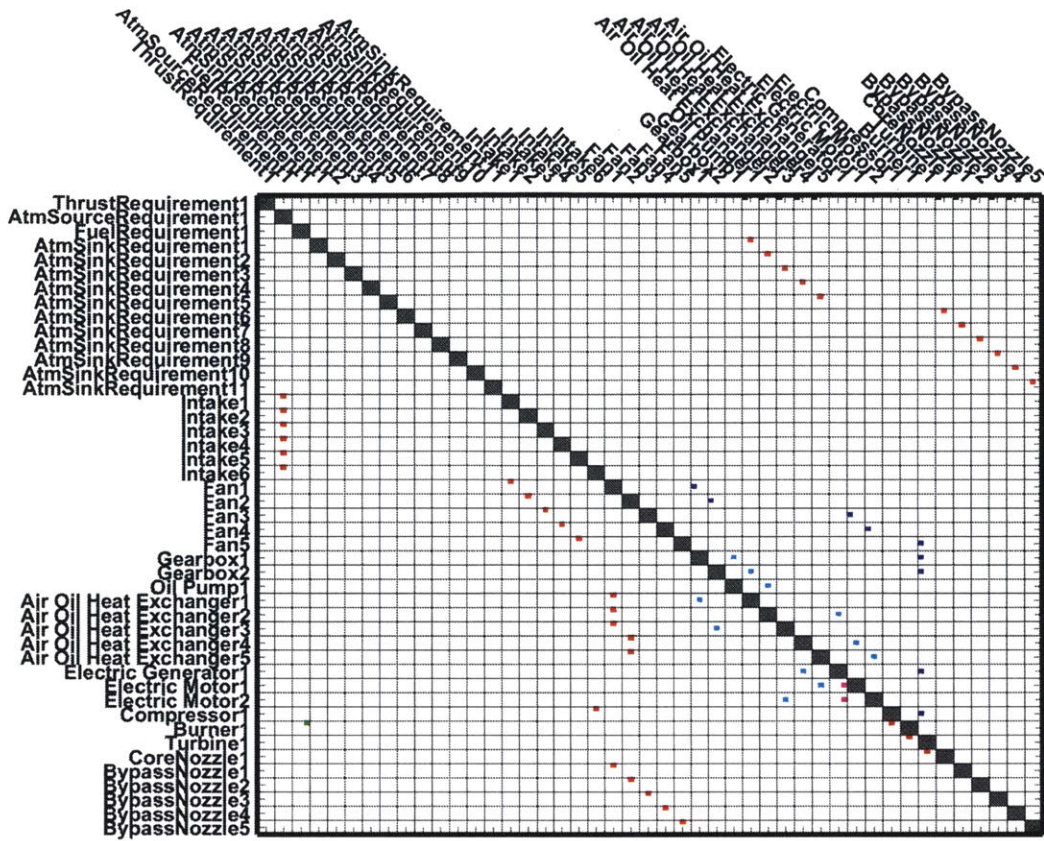


Figure 7.16: Architecture 13 (Turboelectric). Gas connections: **red**, shaft power connections: **blue**, mechanical connections: **black**, fuel connections: **green**, oil connections: **cyan**, electrical connections: **magenta**.

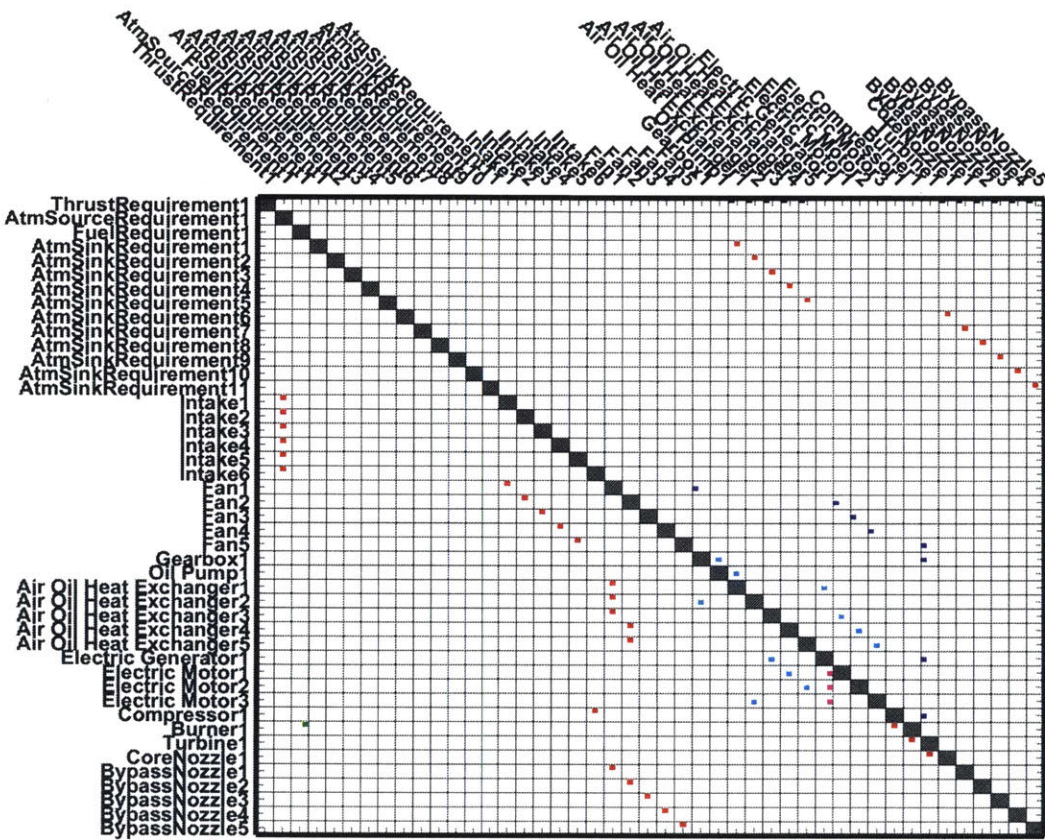


Figure 7.17: Architecture 14 (Turboelectric). Gas connections: **red**, shaft power connections: **blue**, mechanical connections: **black**, fuel connections: **green**, oil connections: **cyan**, electrical connections: **magenta**.

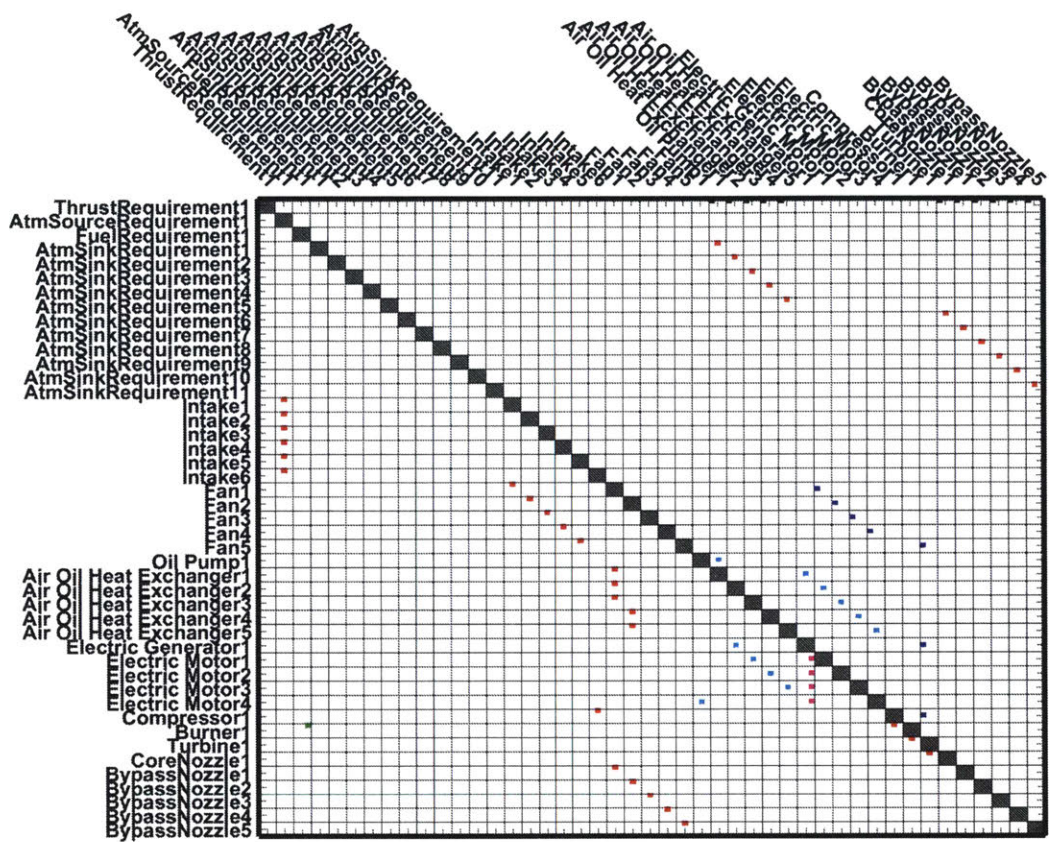


Figure 7.18: Architecture 15 (Turboelectric). Gas connections: **red**, shaft power connections: **blue**, mechanical connections: **black**, fuel connections: **green**, oil connections: **cyan**, electrical connections: **magenta**.

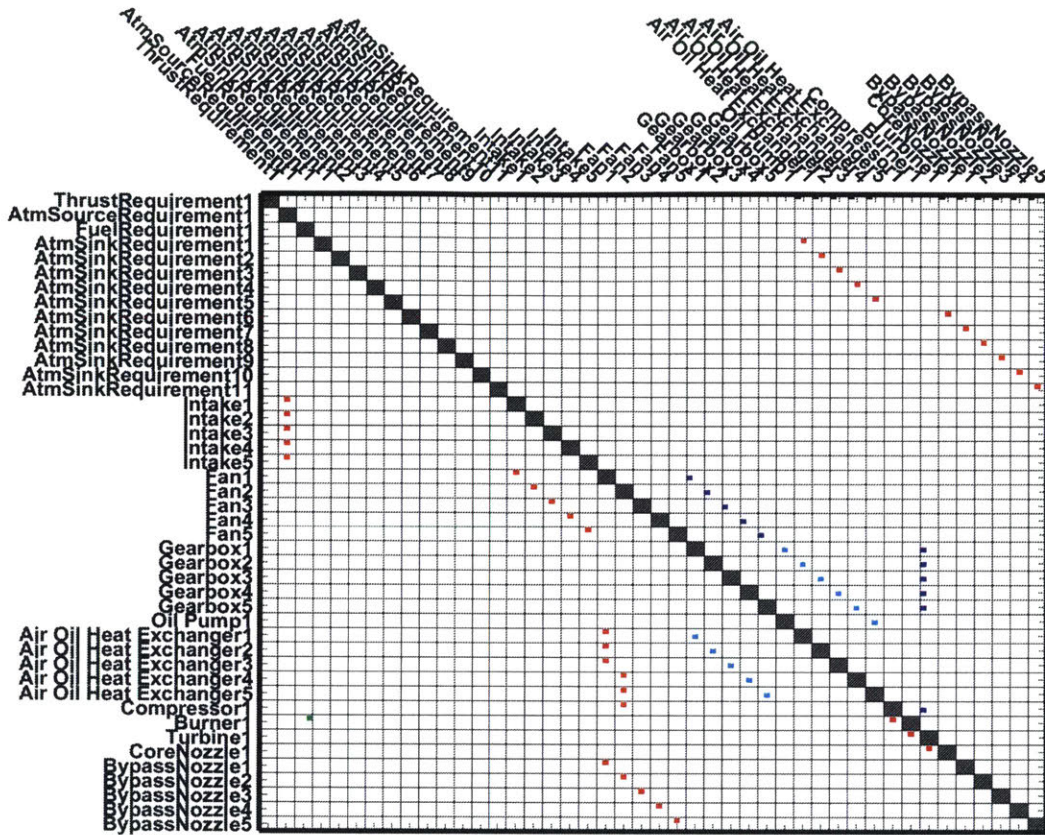


Figure 7.19: Architecture 16. Gas connections: red, shaft power connections: blue, mechanical connections: black, fuel connections: green, oil connections: cyan, electrical connections: magenta.

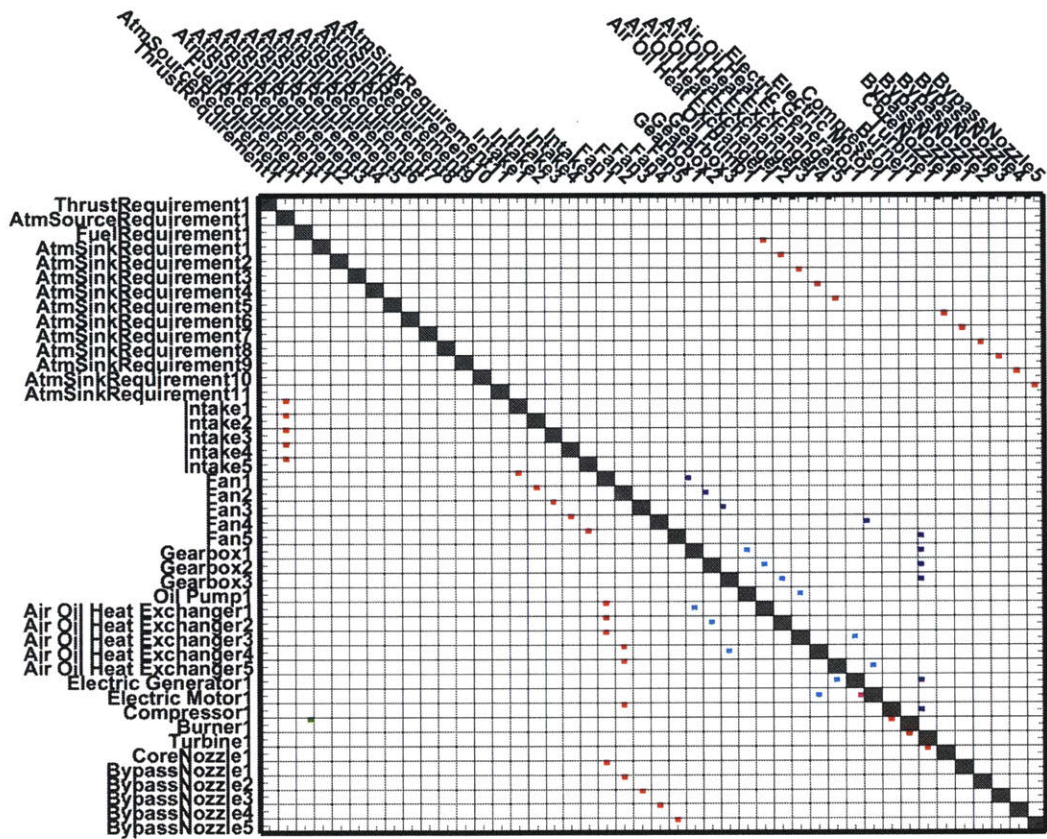


Figure 7.20: Architecture 17 (Turboelectric). Gas connections: **red**, shaft power connections: **blue**, mechanical connections: **black**, fuel connections: **green**, oil connections: **cyan**, electrical connections: **magenta**.

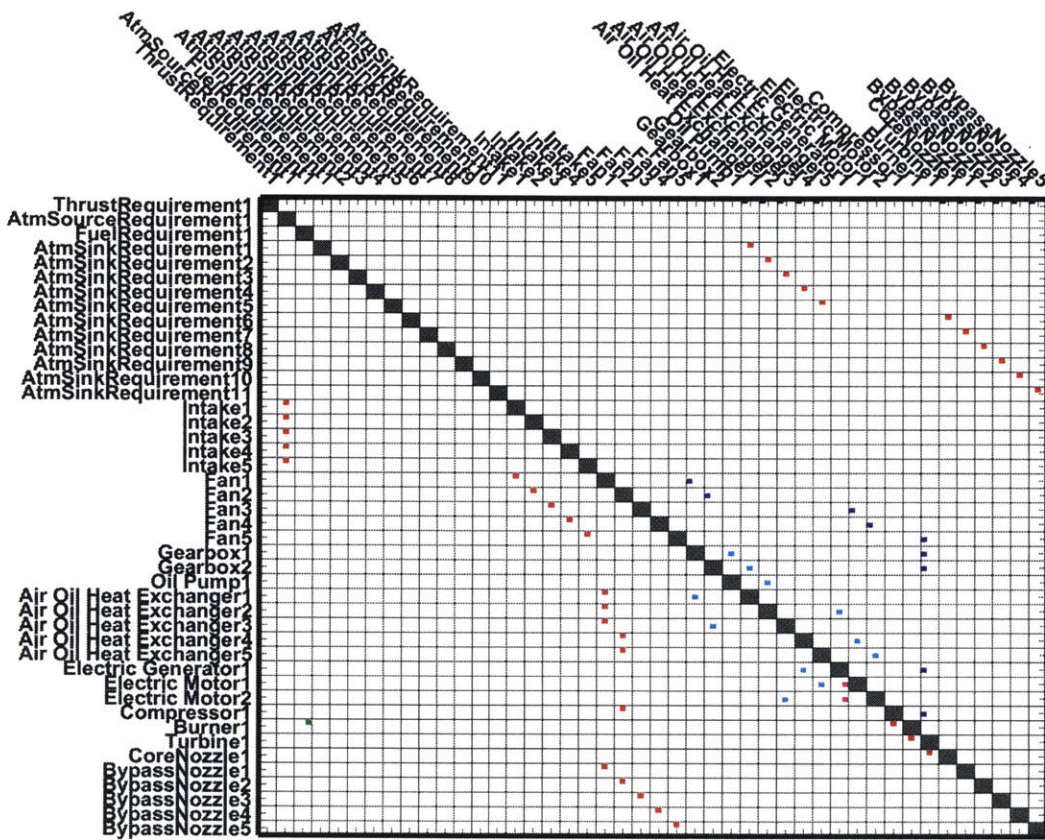


Figure 7.21: Architecture 18 (Turboelectric). Gas connections: **red**, shaft power connections: **blue**, mechanical connections: **black**, fuel connections: **green**, oil connections: **cyan**, electrical connections: **magenta**.

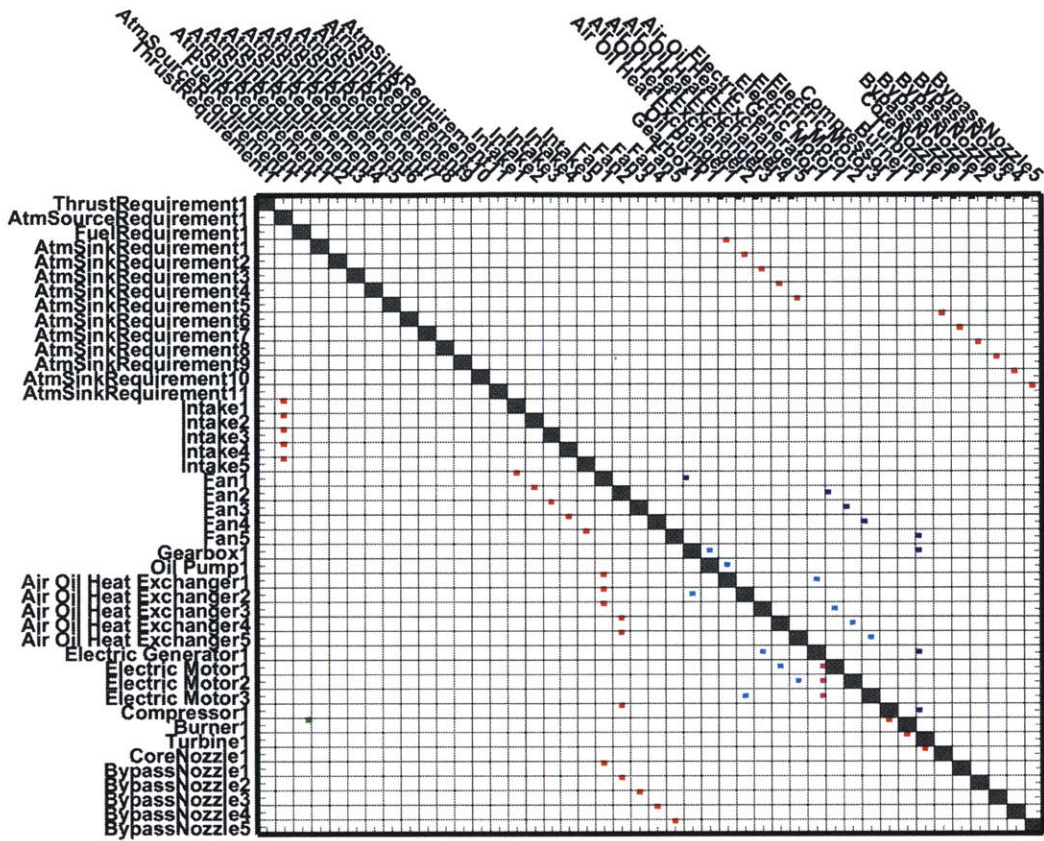


Figure 7.22: Architecture 19 (Turboelectric). Gas connections: **red**, shaft power connections: **blue**, mechanical connections: **black**, fuel connections: **green**, oil connections: **cyan**, electrical connections: **magenta**.

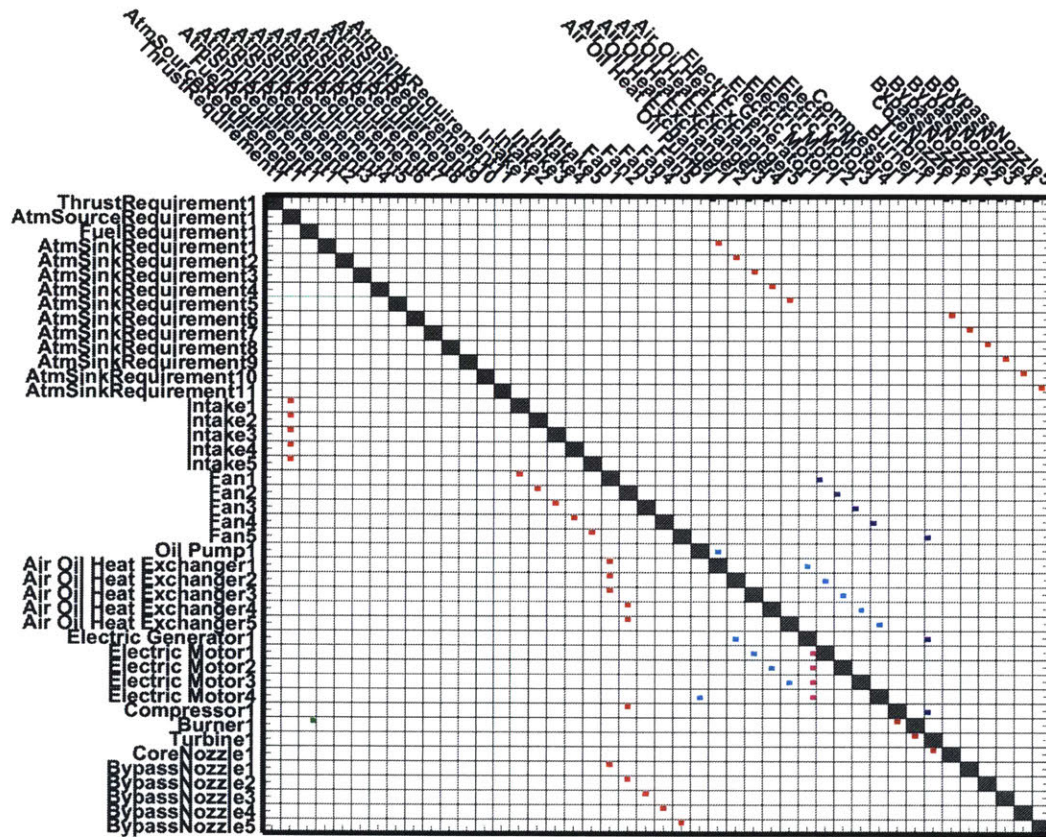


Figure 7.23: Architecture 20 (Turboelectric). Gas connections: **red**, shaft power connections: **blue**, mechanical connections: **black**, fuel connections: **green**, oil connections: **cyan**, electrical connections: **magenta**.

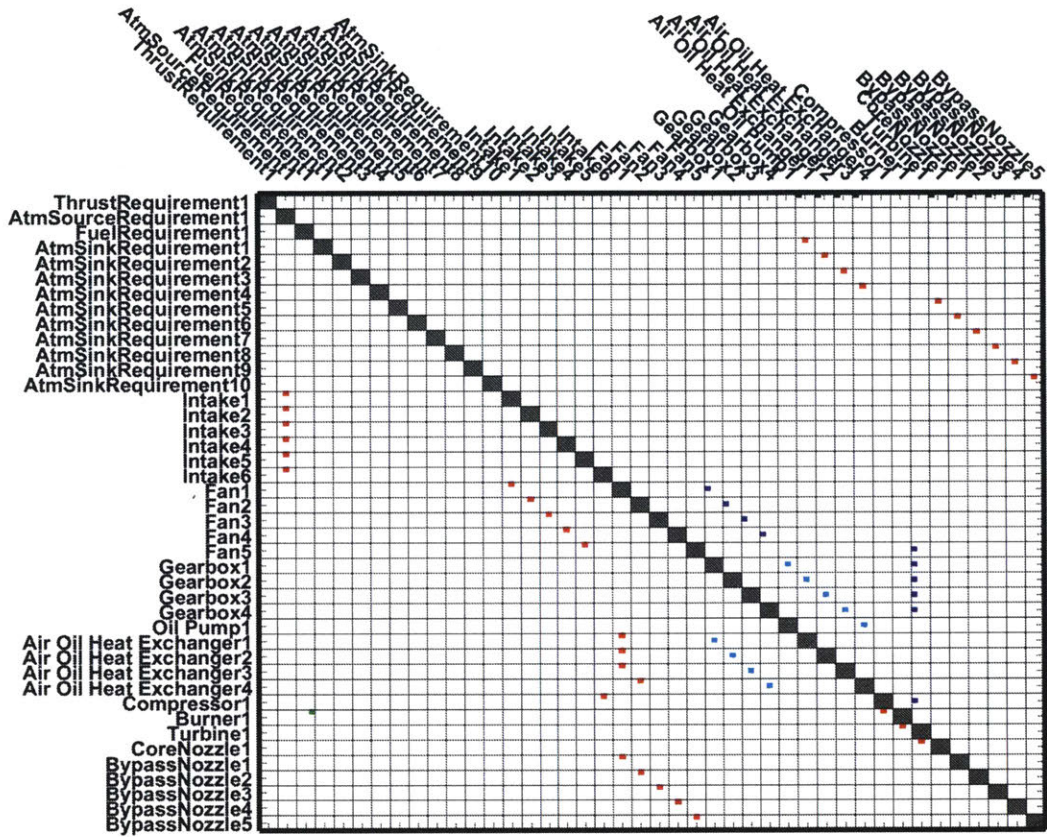


Figure 7.24: Architecture 21. Gas connections: **red**, shaft power connections: **blue**, mechanical connections: **black**, fuel connections: **green**, oil connections: **cyan**, electrical connections: **magenta**.

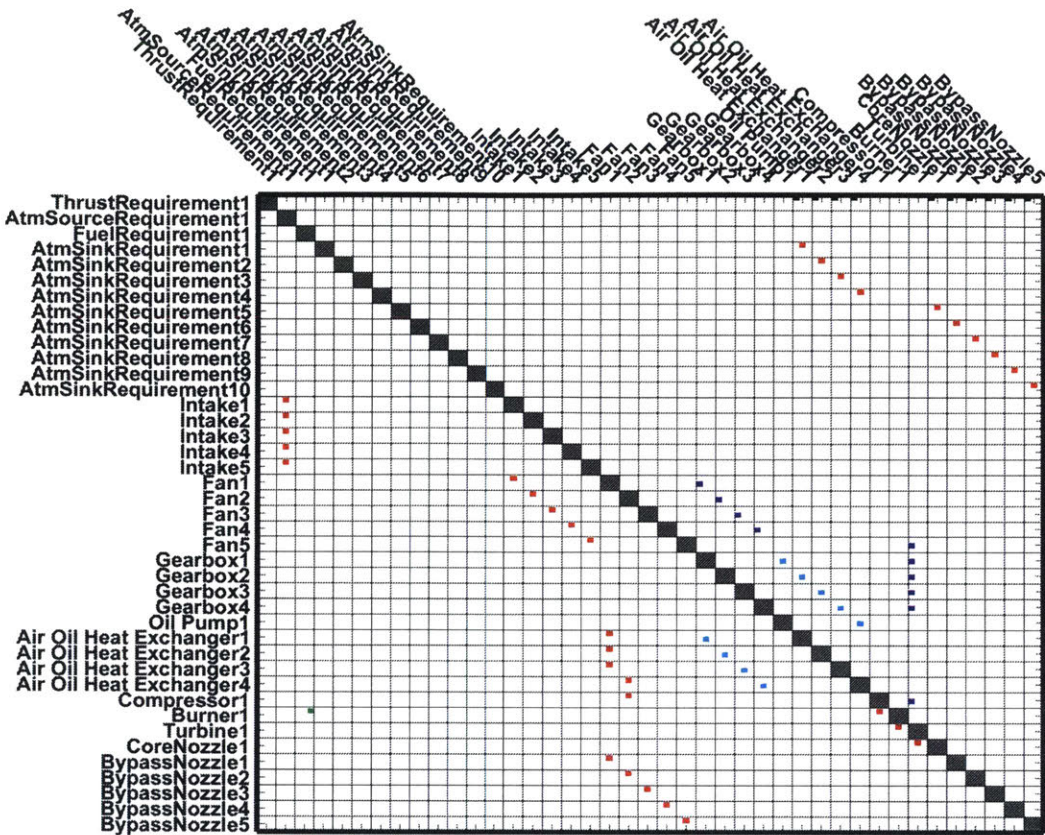


Figure 7.25: Architecture 22. Gas connections: red, shaft power connections: blue, mechanical connections: black, fuel connections: green, oil connections: cyan, electrical connections: magenta.

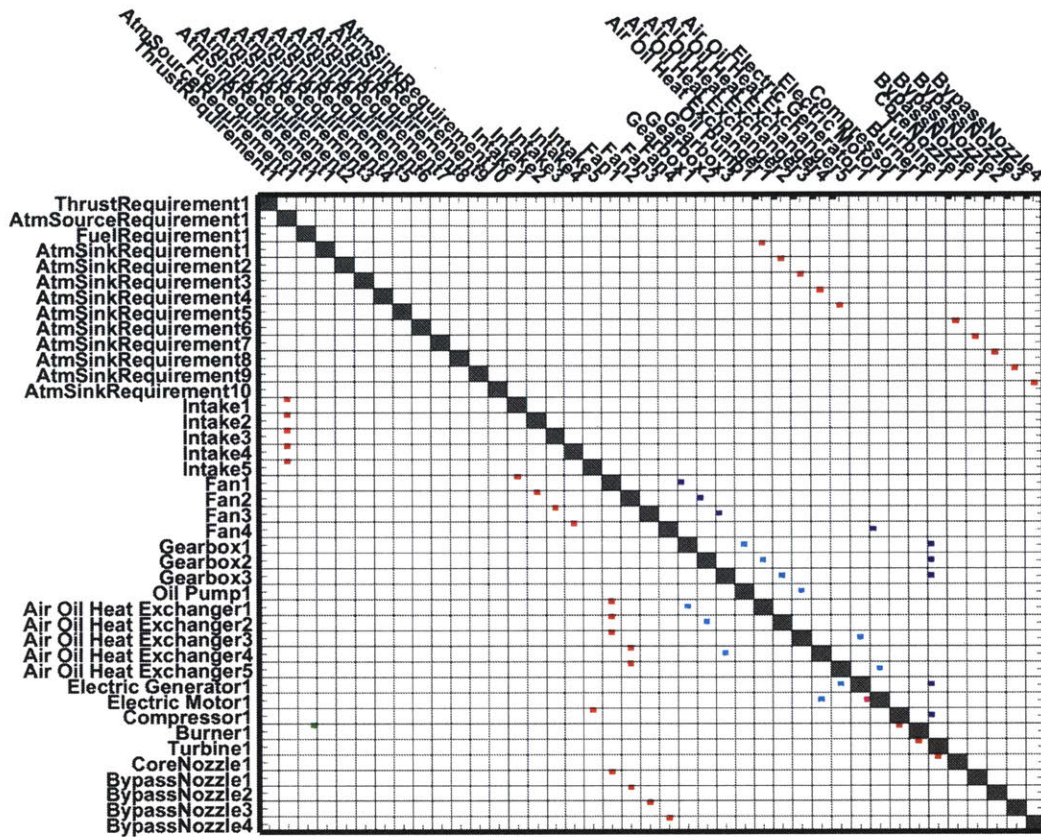


Figure 7.26: Architecture 23 (Turboelectric). Gas connections: **red**, shaft power connections: **blue**, mechanical connections: **black**, fuel connections: **green**, oil connections: **cyan**, electrical connections: **magenta**.

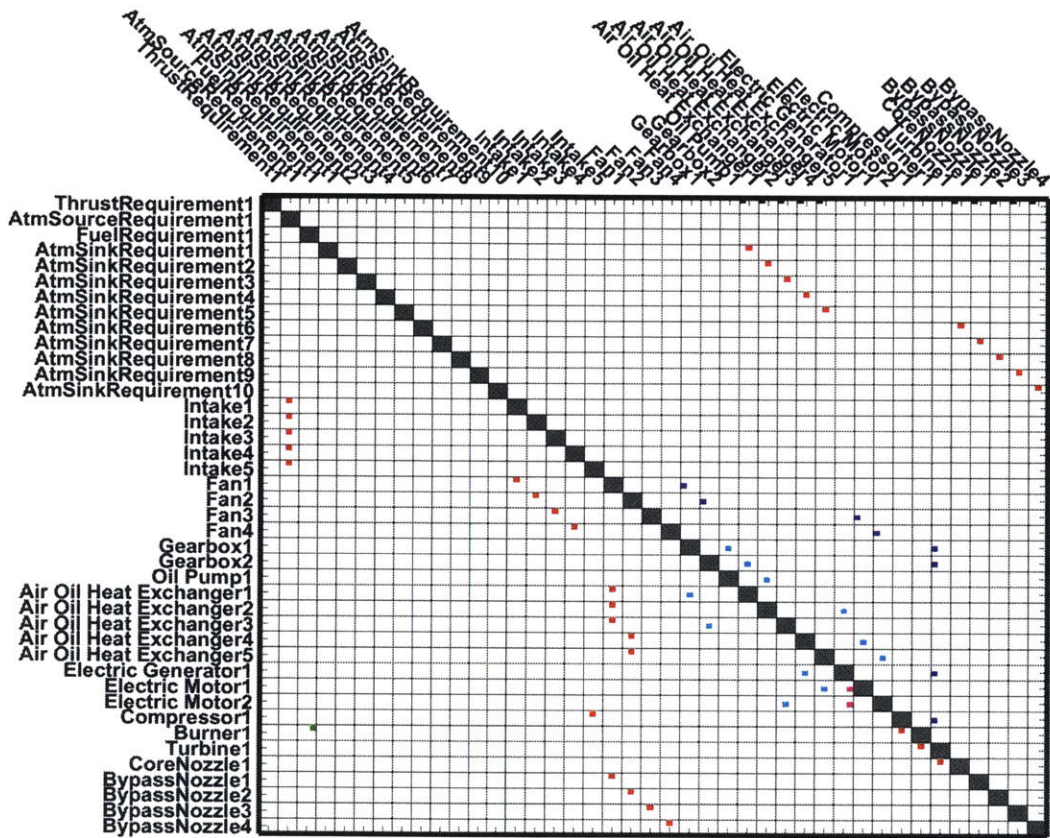


Figure 7.27: Architecture 24 (Turboelectric). Gas connections: **red**, shaft power connections: **blue**, mechanical connections: **black**, fuel connections: **green**, oil connections: **cyan**, electrical connections: **magenta**.

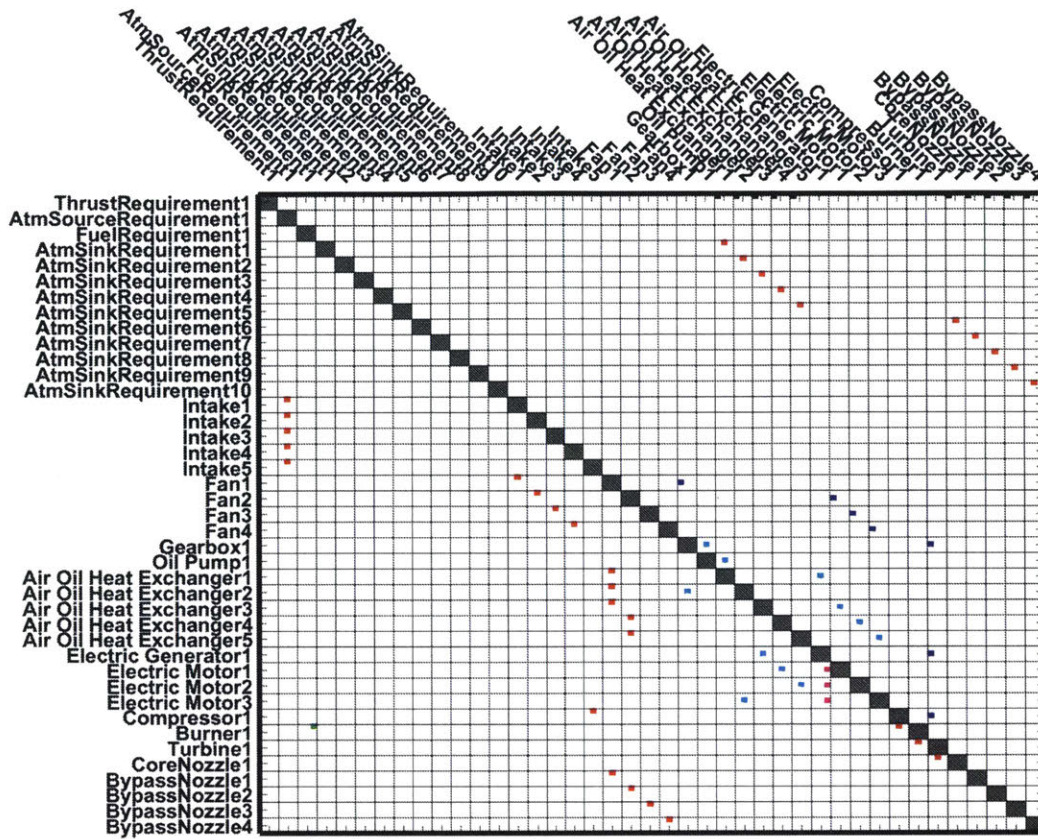


Figure 7.28: Architecture 25 (Turboelectric). Gas connections: **red**, shaft power connections: **blue**, mechanical connections: **black**, fuel connections: **green**, oil connections: **cyan**, electrical connections: **magenta**.

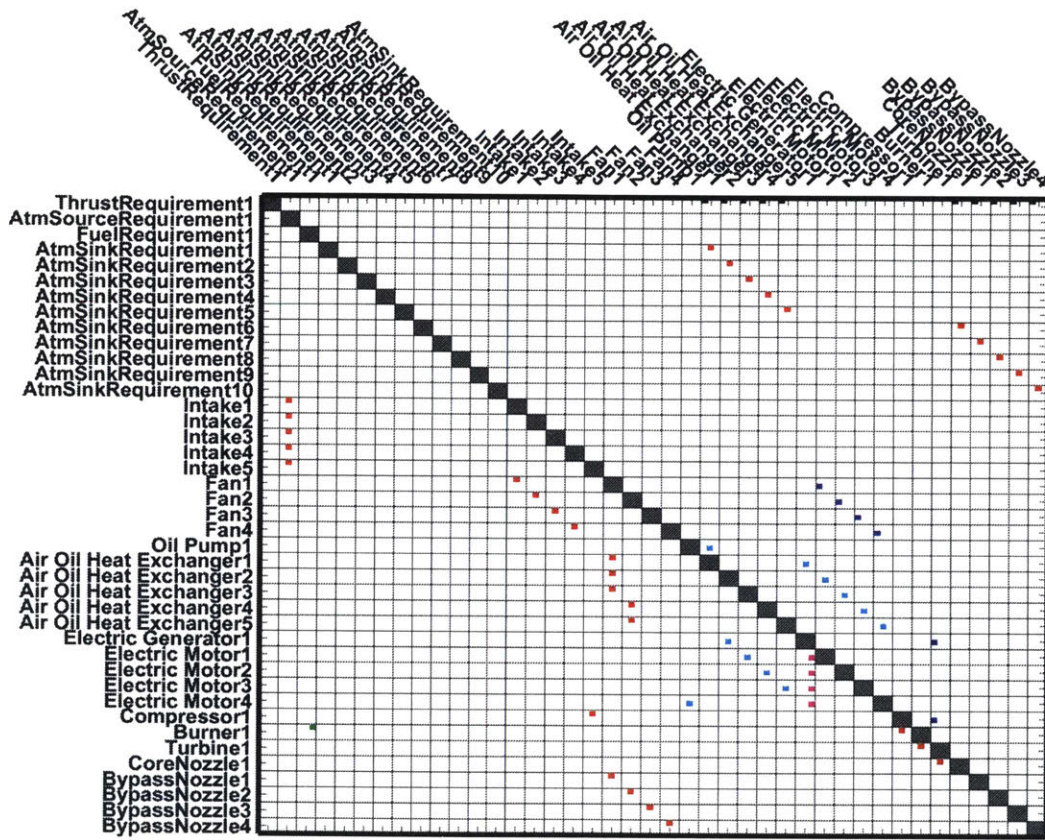


Figure 7.29: Architecture 26 (Turboelectric). Gas connections: **red**, shaft power connections: **blue**, mechanical connections: **black**, fuel connections: **green**, oil connections: **cyan**, electrical connections: **magenta**.

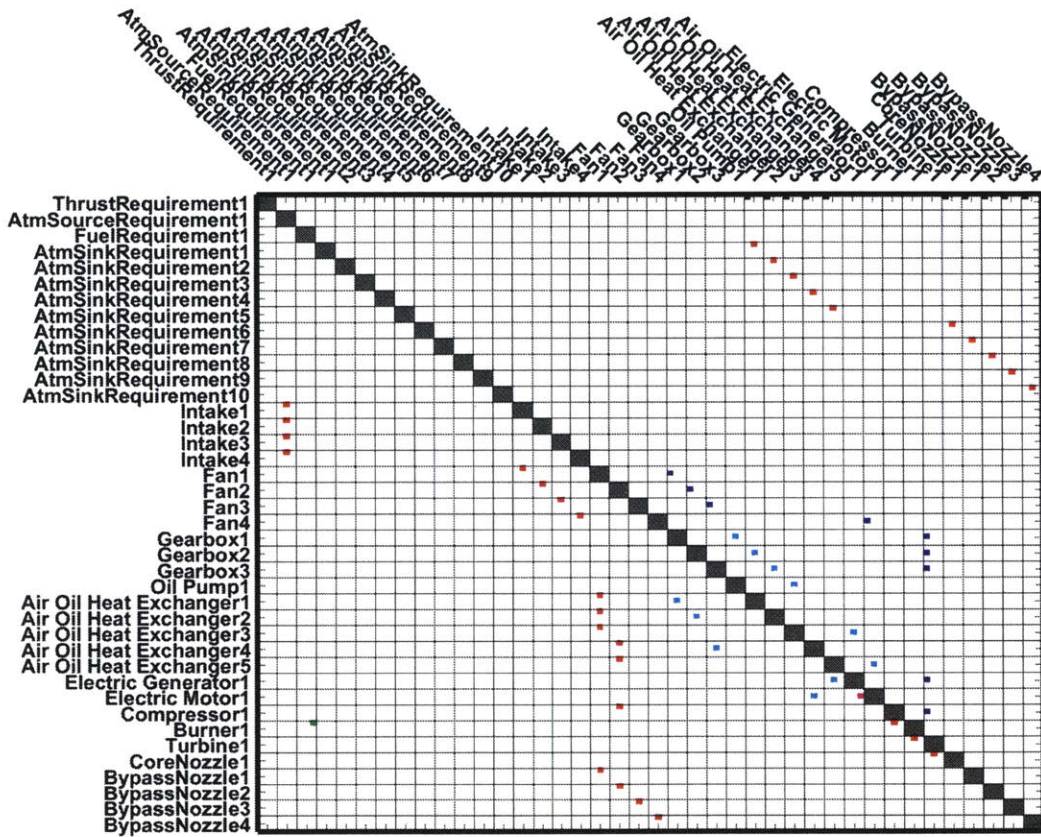


Figure 7.30: Architecture 27 (Turboelectric). Gas connections: **red**, shaft power connections: **blue**, mechanical connections: **black**, fuel connections: **green**, oil connections: **cyan**, electrical connections: **magenta**.

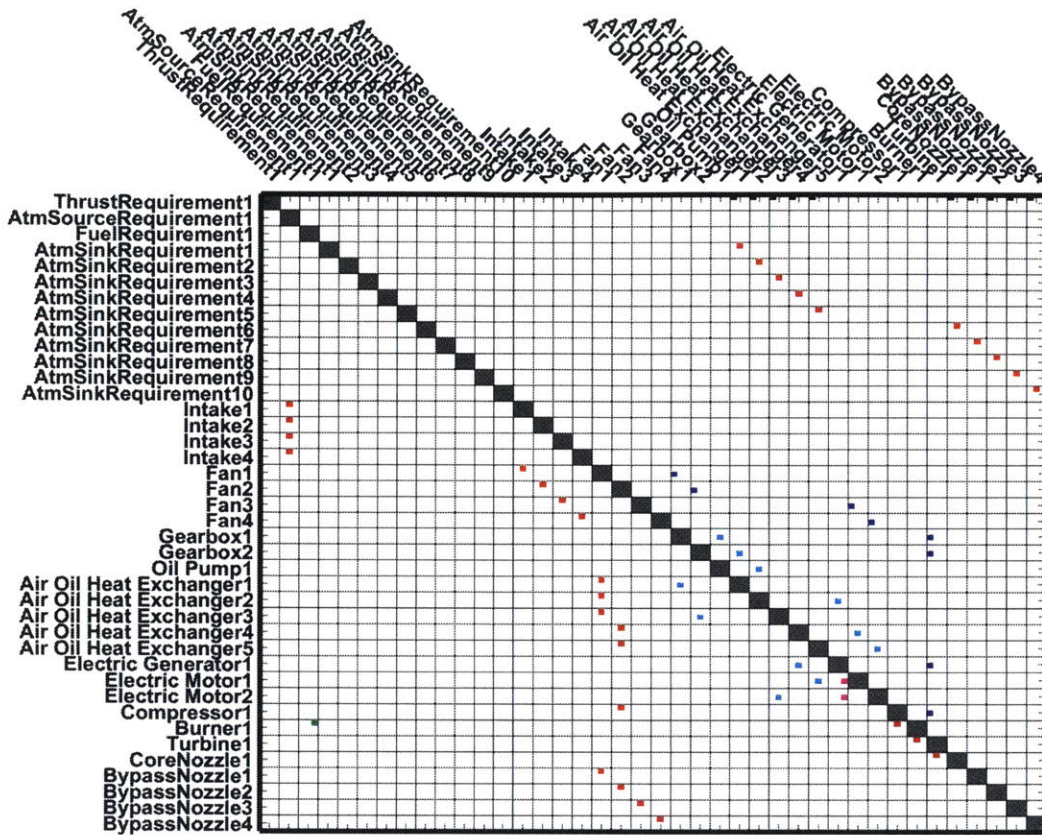


Figure 7.31: Architecture 28 (Turboelectric). Gas connections: **red**, shaft power connections: **blue**, mechanical connections: **black**, fuel connections: **green**, oil connections: **cyan**, electrical connections: **magenta**.

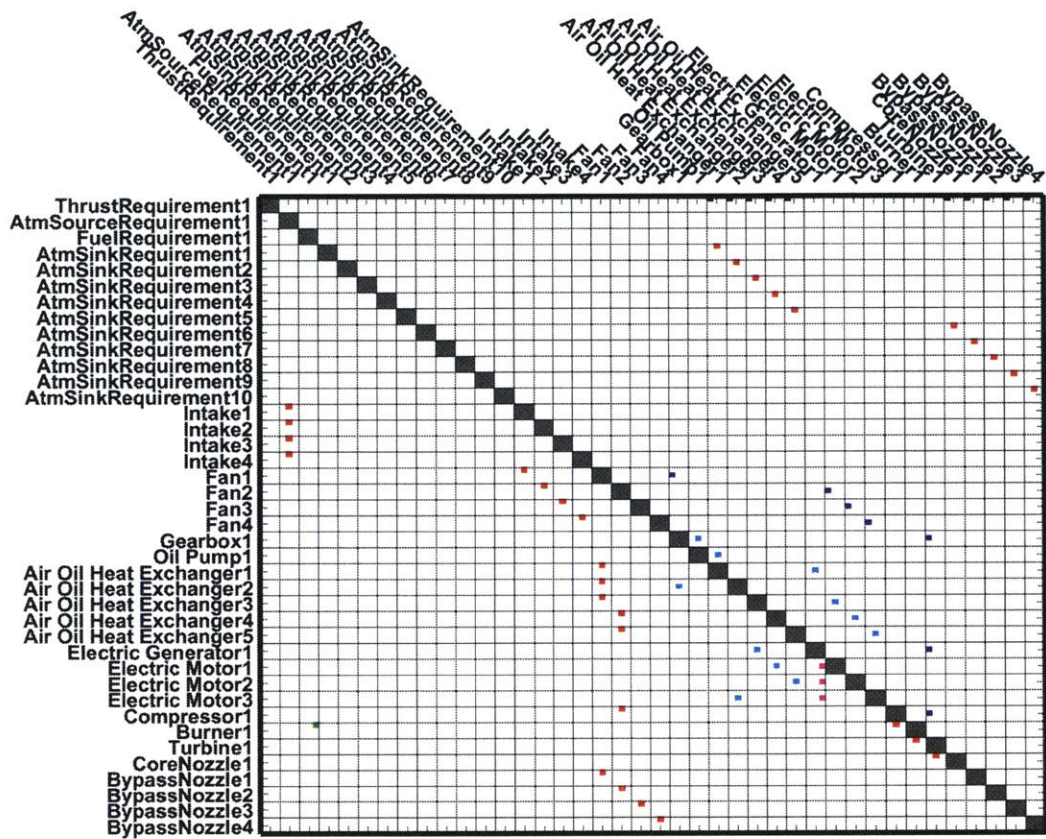


Figure 7.32: Architecture 29 (Turboelectric). Gas connections: **red**, shaft power connections: **blue**, mechanical connections: **black**, fuel connections: **green**, oil connections: **cyan**, electrical connections: **magenta**.

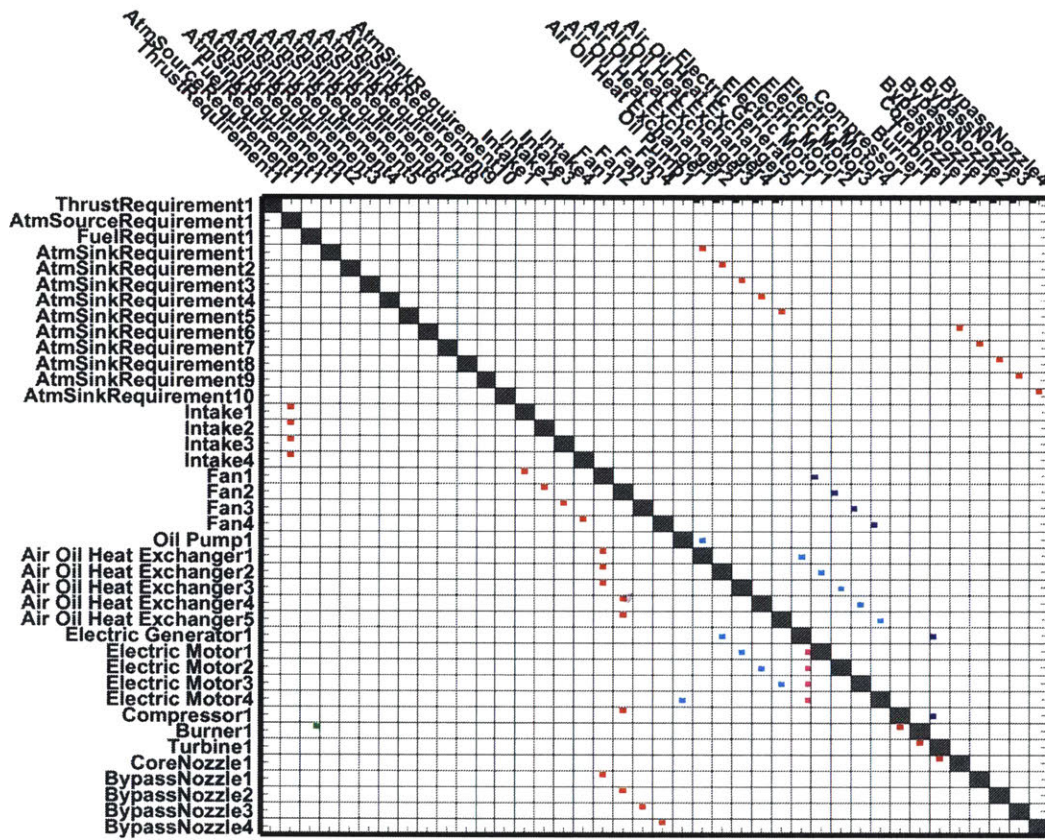


Figure 7.33: Architecture 30 (Turboelectric). Gas connections: **red**, shaft power connections: **blue**, mechanical connections: **black**, fuel connections: **green**, oil connections: **cyan**, electrical connections: **magenta**.

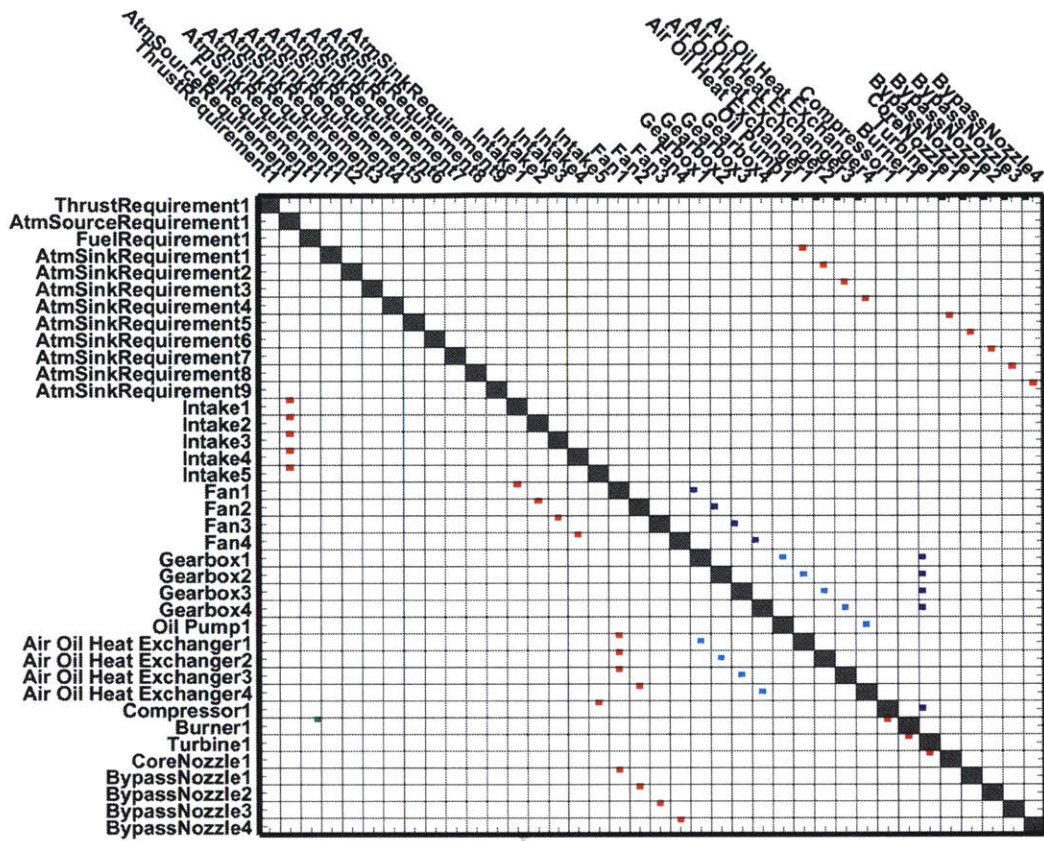


Figure 7.34: Architecture 31. Gas connections: red, shaft power connections: blue, mechanical connections: black, fuel connections: green, oil connections: cyan, electrical connections: magenta.

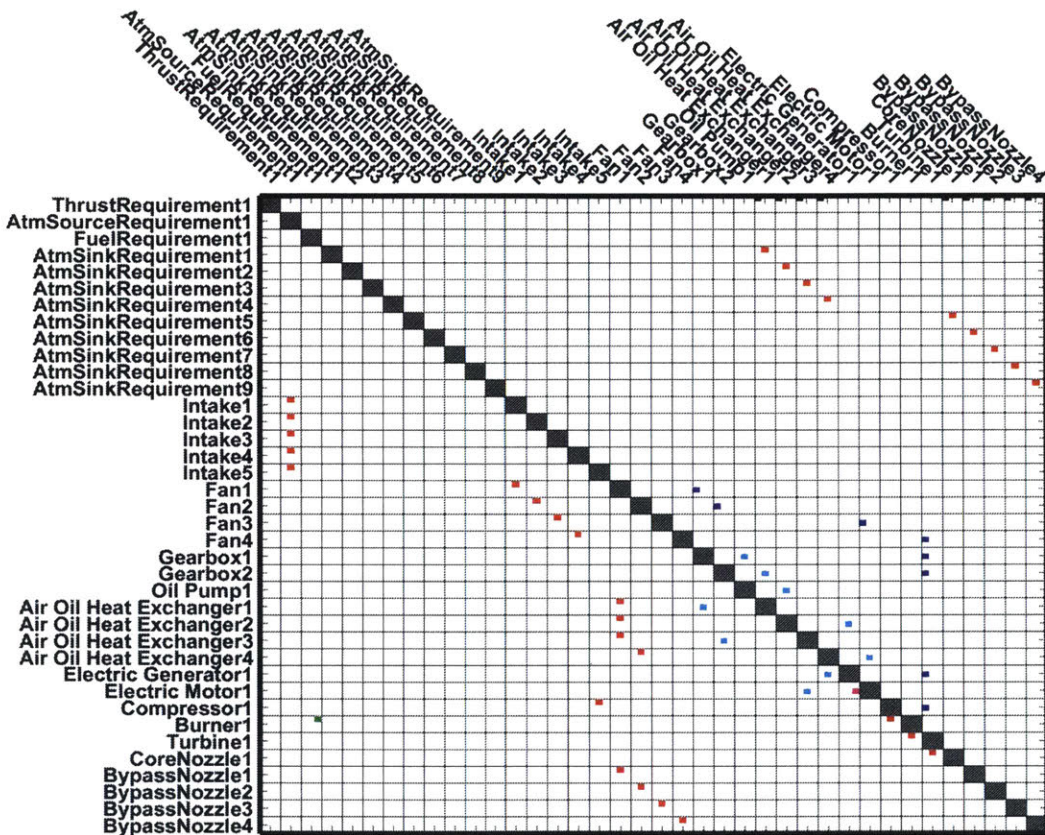


Figure 7.35: Architecture 32 (Turboelectric). Gas connections: red, shaft power connections: blue, mechanical connections: black, fuel connections: green, oil connections: cyan, electrical connections: magenta.

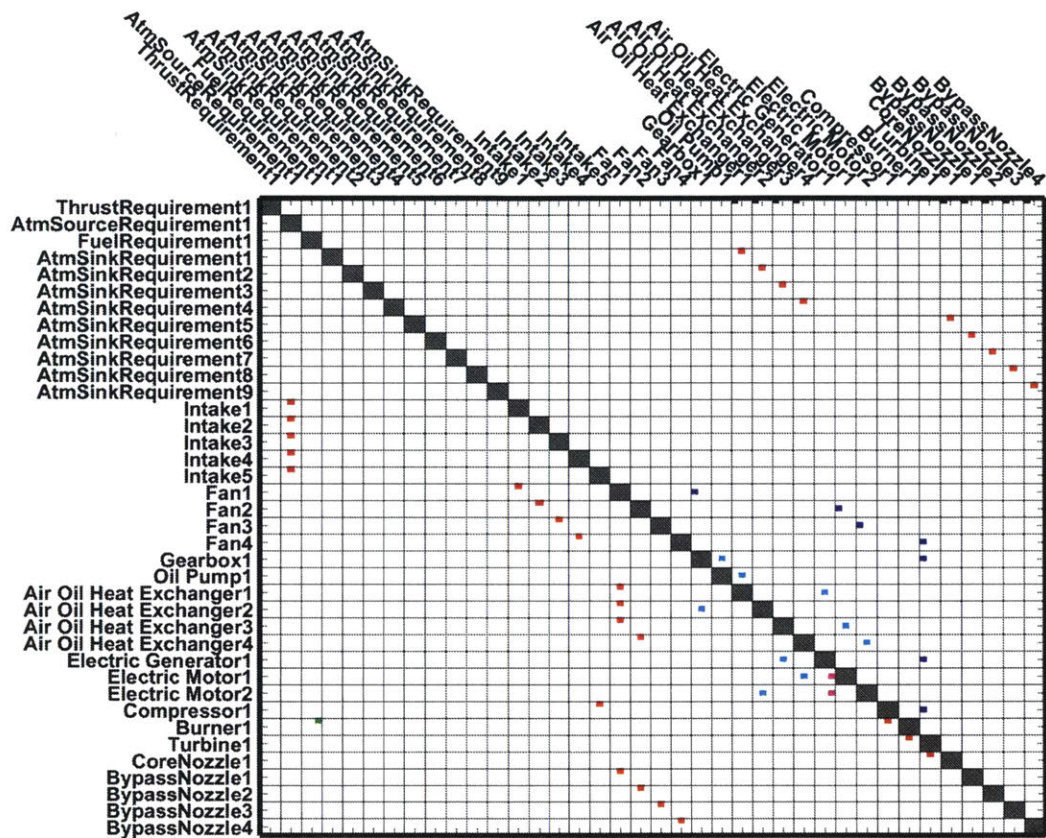


Figure 7.36: Architecture 33 (Turboelectric). Gas connections: **red**, shaft power connections: **blue**, mechanical connections: **black**, fuel connections: **green**, oil connections: **cyan**, electrical connections: **magenta**.

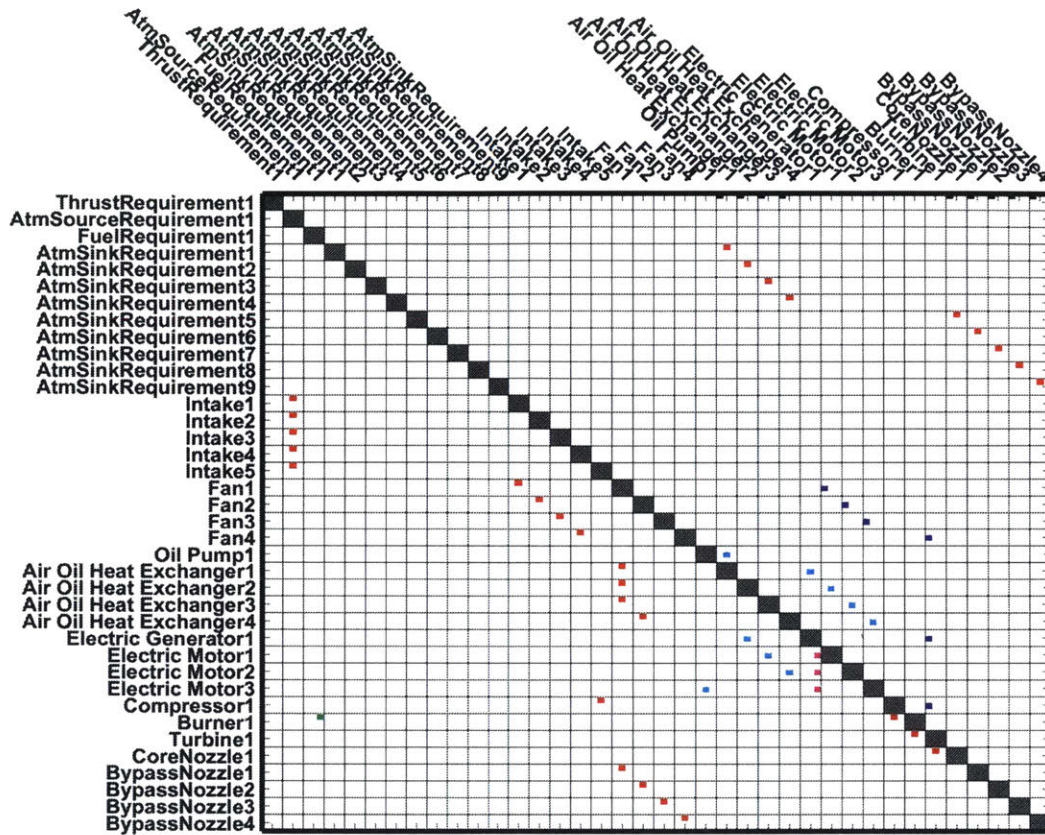


Figure 7.37: Architecture 34 (Turboelectric). Gas connections: **red**, shaft power connections: **blue**, mechanical connections: **black**, fuel connections: **green**, oil connections: **cyan**, electrical connections: **magenta**.

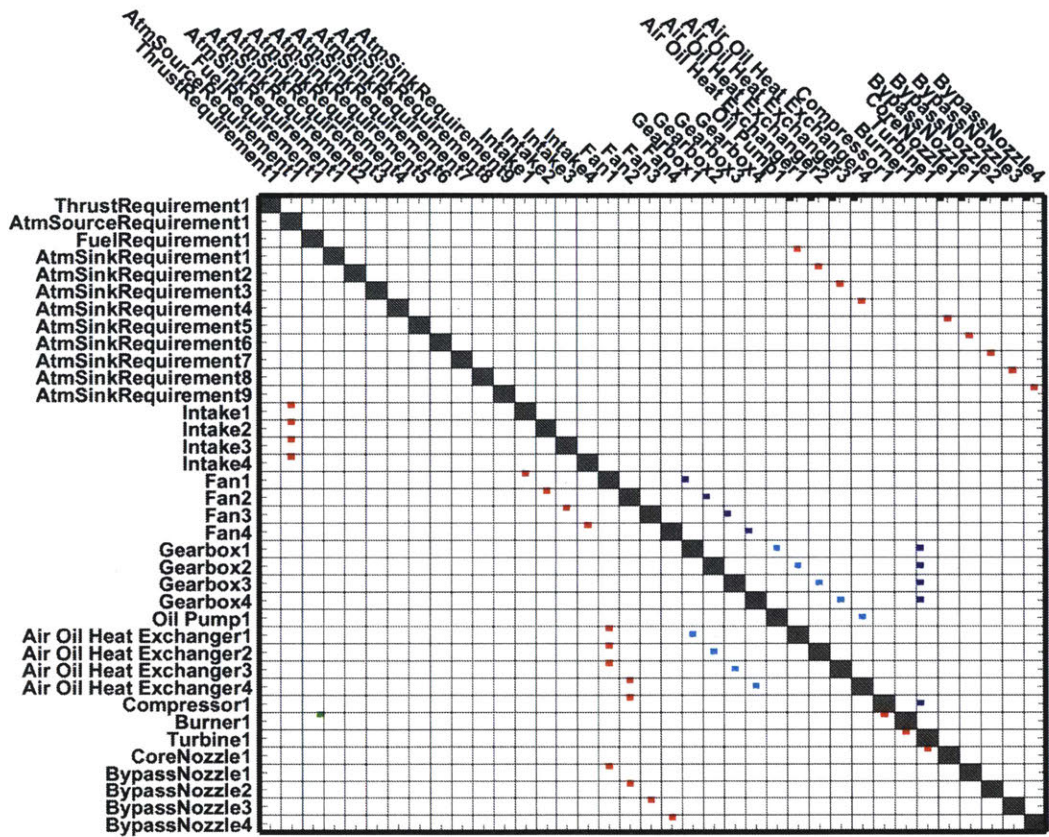


Figure 7.38: Architecture 35. Gas connections: red, shaft power connections: blue, mechanical connections: black, fuel connections: green, oil connections: cyan, electrical connections: magenta.

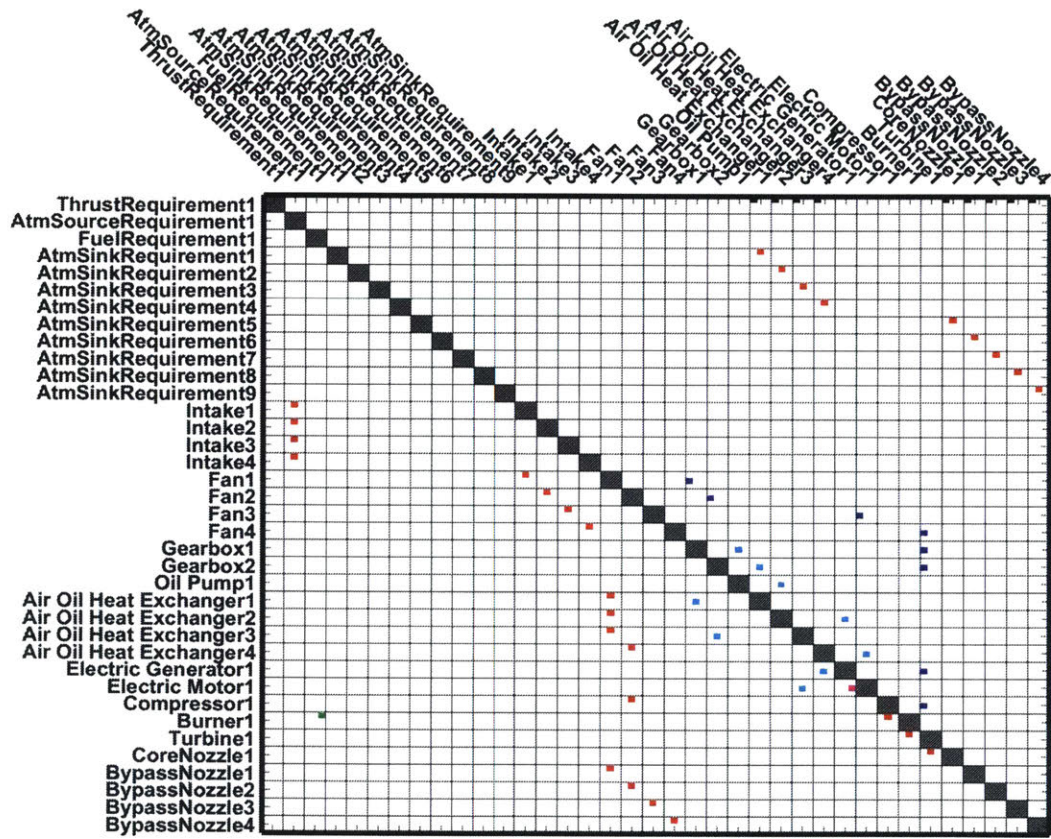


Figure 7.39: Architecture 36 (Turboelectric). Gas connections: **red**, shaft power connections: **blue**, mechanical connections: **black**, fuel connections: **green**, oil connections: **cyan**, electrical connections: **magenta**.

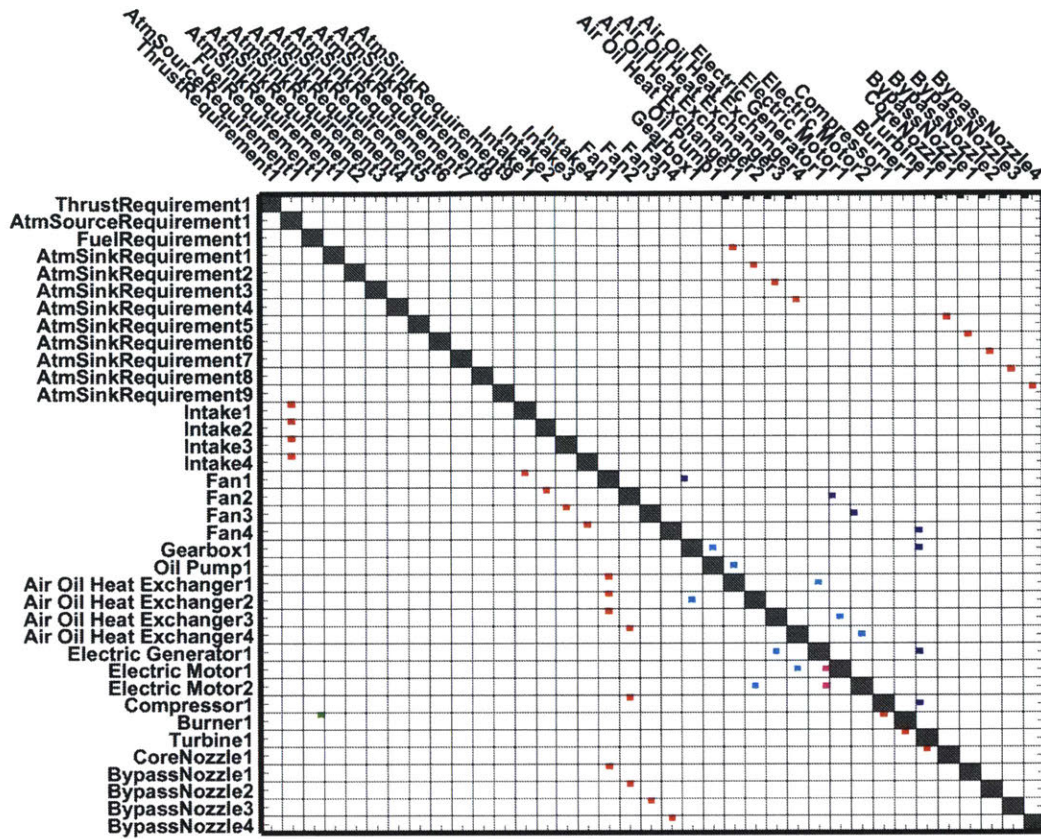


Figure 7.40: Architecture 37 (Turboelectric). Gas connections: **red**, shaft power connections: **blue**, mechanical connections: **black**, fuel connections: **green**, oil connections: **cyan**, electrical connections: **magenta**.

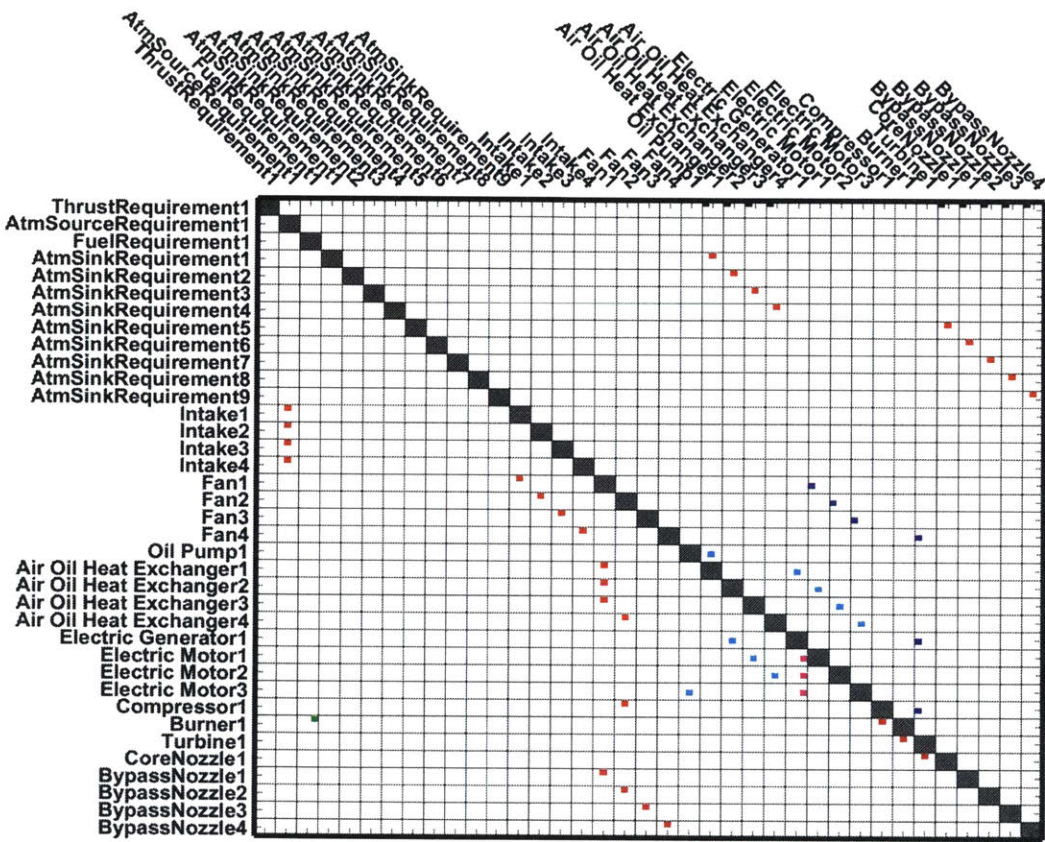


Figure 7.41: Architecture 38 (Turboelectric). Gas connections: **red**, shaft power connections: **blue**, mechanical connections: **black**, fuel connections: **green**, oil connections: **cyan**, electrical connections: **magenta**.

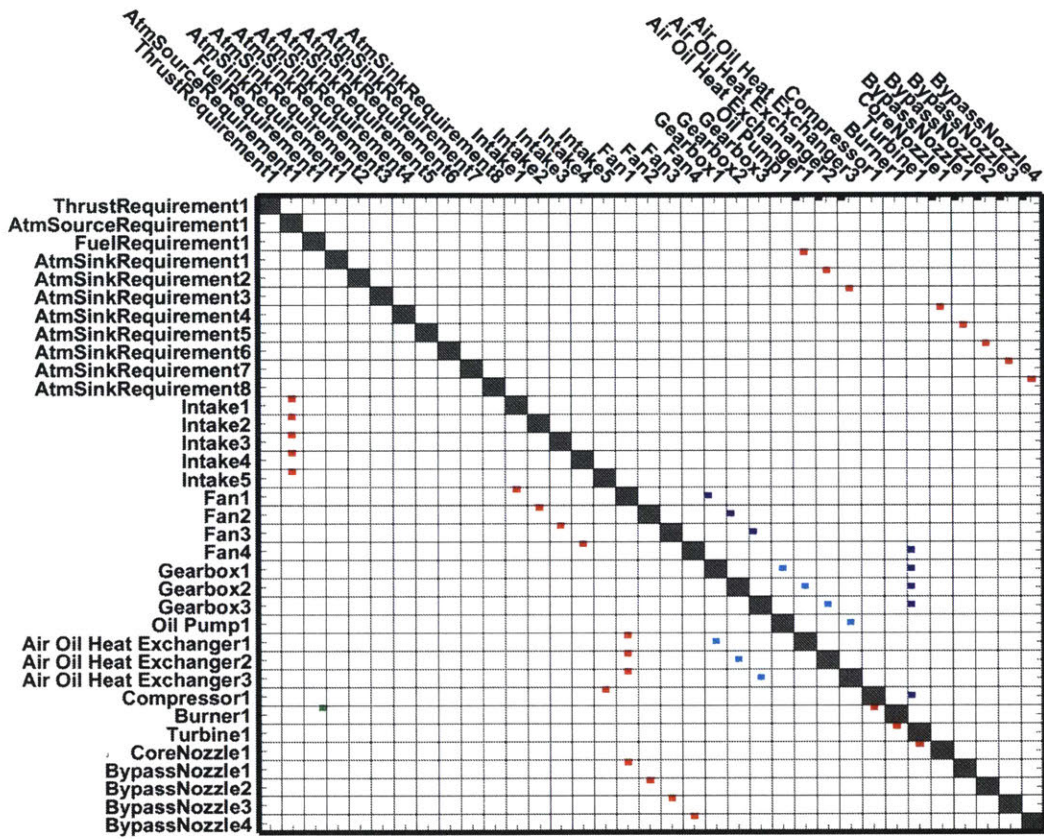


Figure 7.42: Architecture 39. Gas connections: red, shaft power connections: blue, mechanical connections: black, fuel connections: green, oil connections: cyan, electrical connections: magenta.

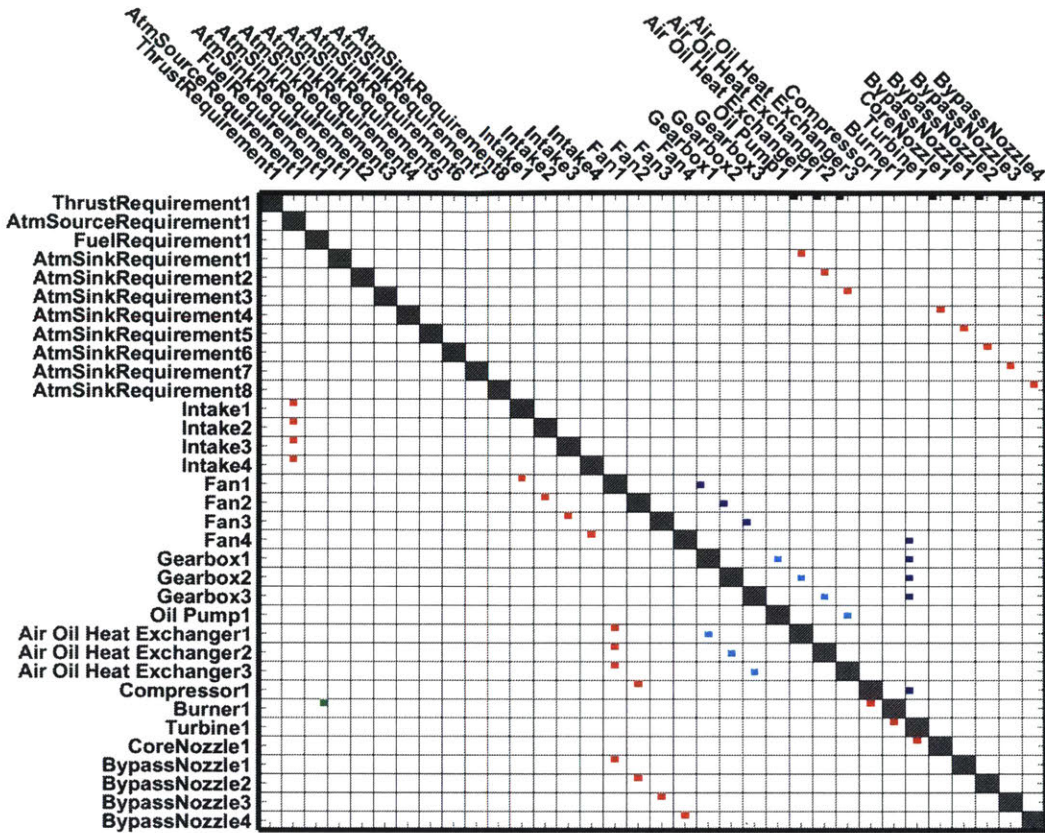


Figure 7.43: Architecture 40. Gas connections: red, shaft power connections: blue, mechanical connections: black, fuel connections: green, oil connections: cyan, electrical connections: magenta.

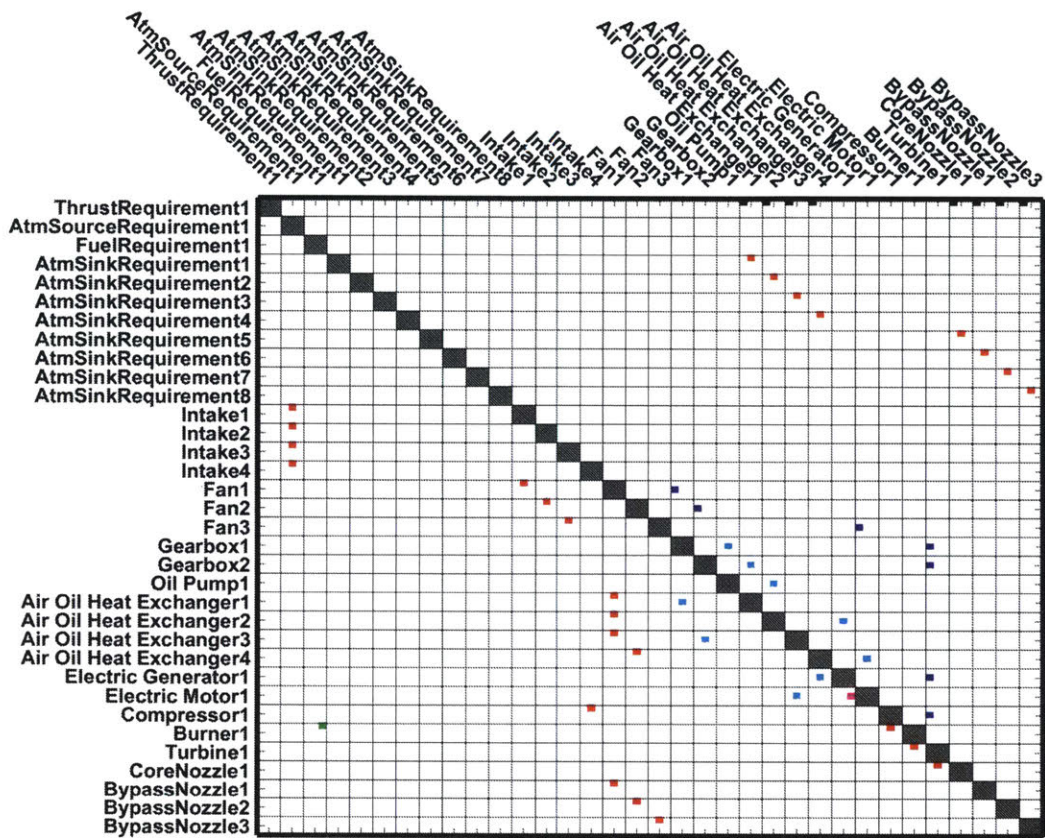


Figure 7.44: Architecture 41 (Turboelectric). Gas connections: **red**, shaft power connections: **blue**, mechanical connections: **black**, fuel connections: **green**, oil connections: **cyan**, electrical connections: **magenta**.

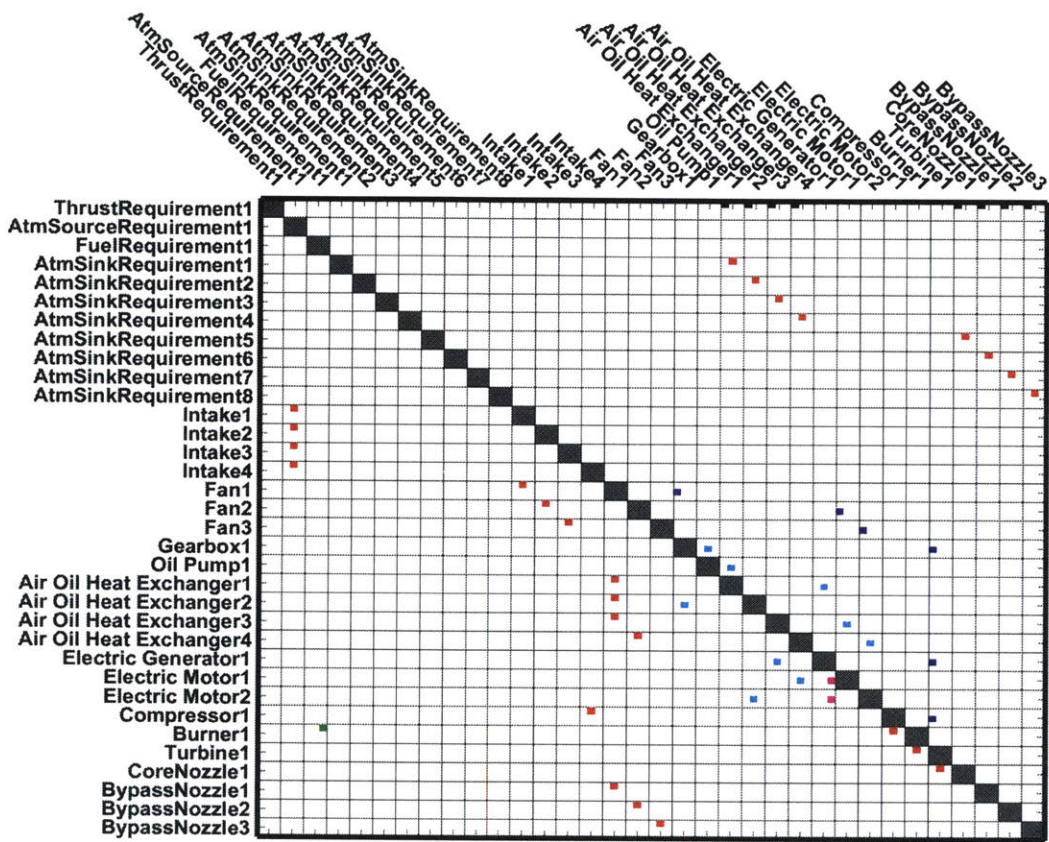


Figure 7.45: Architecture 42 (Turboelectric). Gas connections: **red**, shaft power connections: **blue**, mechanical connections: **black**, fuel connections: **green**, oil connections: **cyan**, electrical connections: **magenta**.

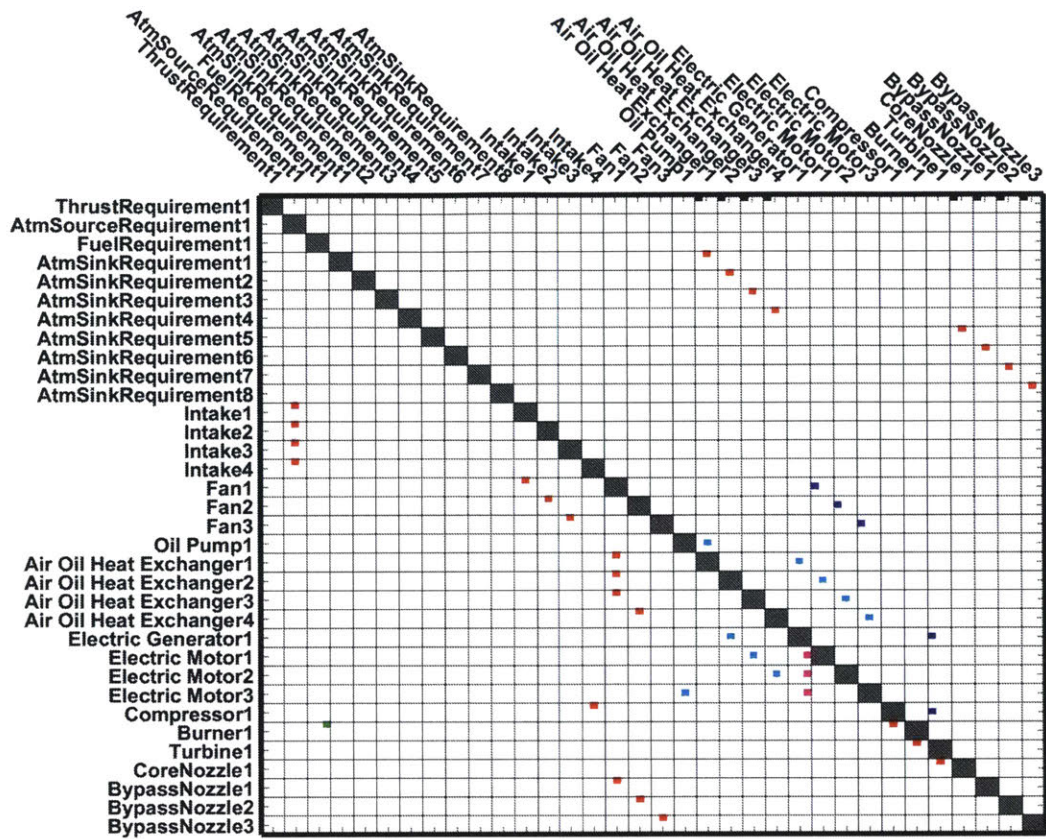


Figure 7.46: Architecture 43 (Turboelectric). Gas connections: **red**, shaft power connections: **blue**, mechanical connections: **black**, fuel connections: **green**, oil connections: **cyan**, electrical connections: **magenta**.

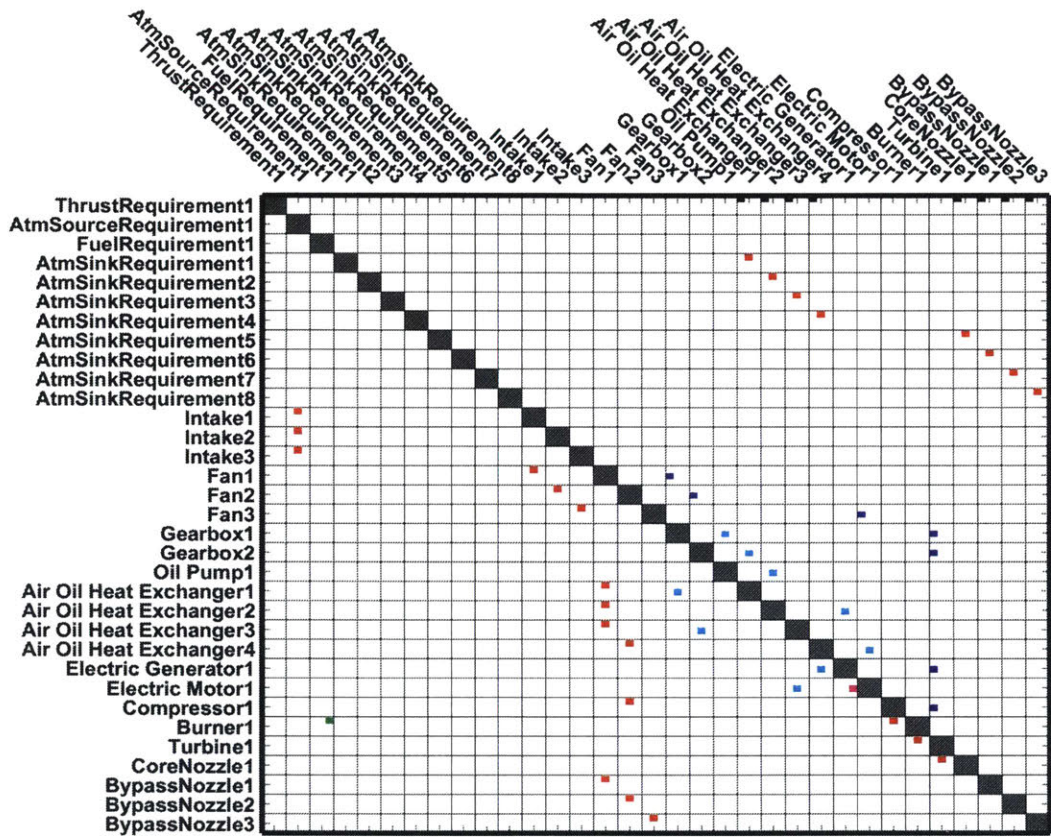


Figure 7.47: Architecture 44 (Turboelectric). Gas connections: **red**, shaft power connections: **blue**, mechanical connections: **black**, fuel connections: **green**, oil connections: **cyan**, electrical connections: **magenta**.

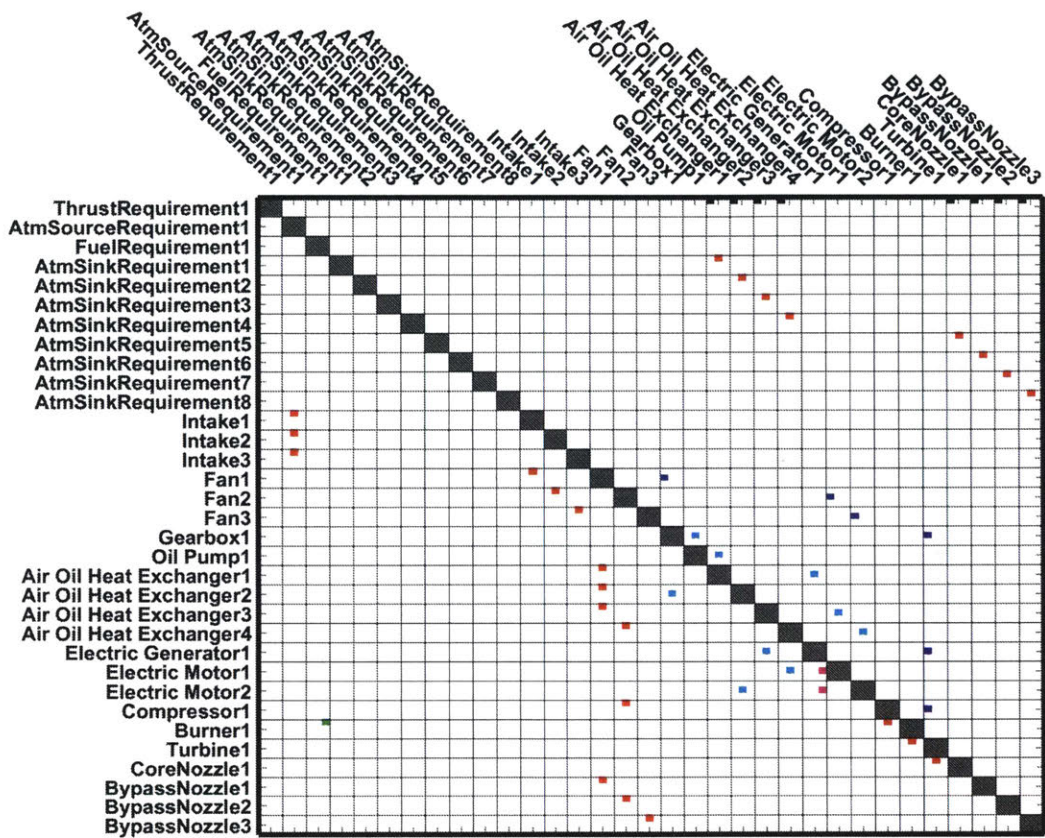


Figure 7.48: Architecture 45 (Turboelectric). Gas connections: **red**, shaft power connections: **blue**, mechanical connections: **black**, fuel connections: **green**, oil connections: **cyan**, electrical connections: **magenta**.

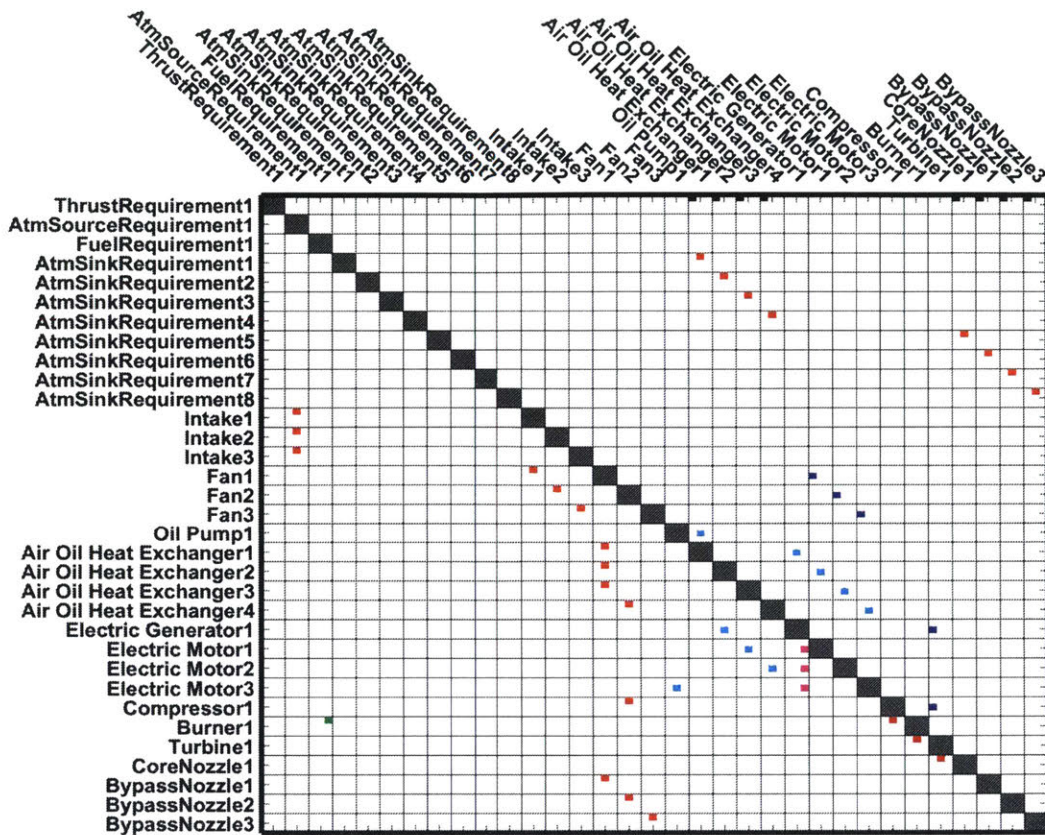


Figure 7.49: Architecture 46 (Turboelectric). Gas connections: **red**, shaft power connections: **blue**, mechanical connections: **black**, fuel connections: **green**, oil connections: **cyan**, electrical connections: **magenta**.

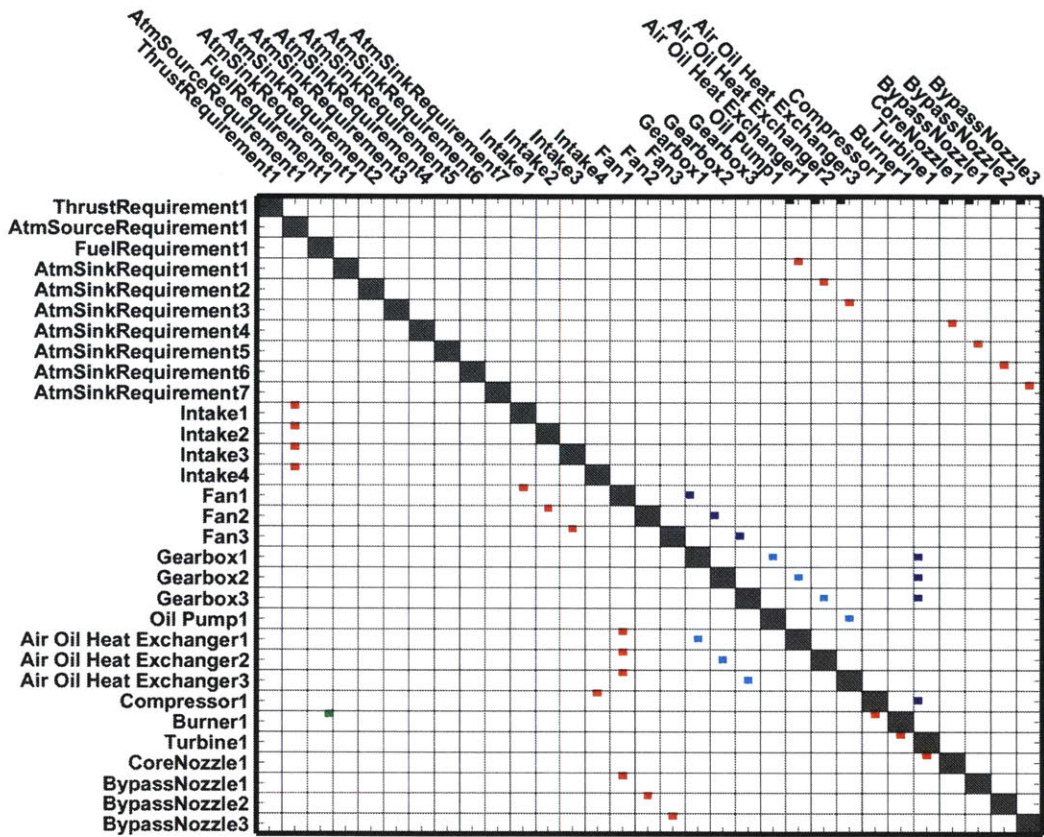


Figure 7.50: Architecture 47. Gas connections: red, shaft power connections: blue, mechanical connections: black, fuel connections: green, oil connections: cyan, electrical connections: magenta.

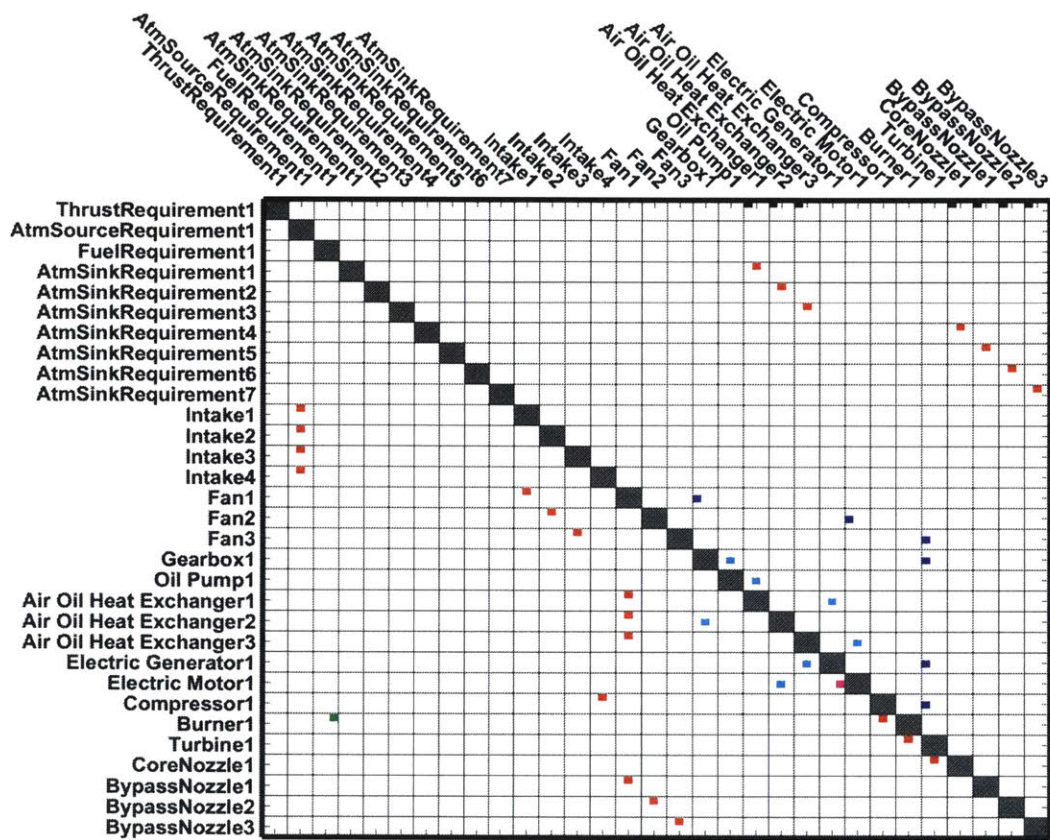


Figure 7.51: Architecture 48 (Turboelectric). Gas connections: **red**, shaft power connections: **blue**, mechanical connections: **black**, fuel connections: **green**, oil connections: **cyan**, electrical connections: **magenta**.

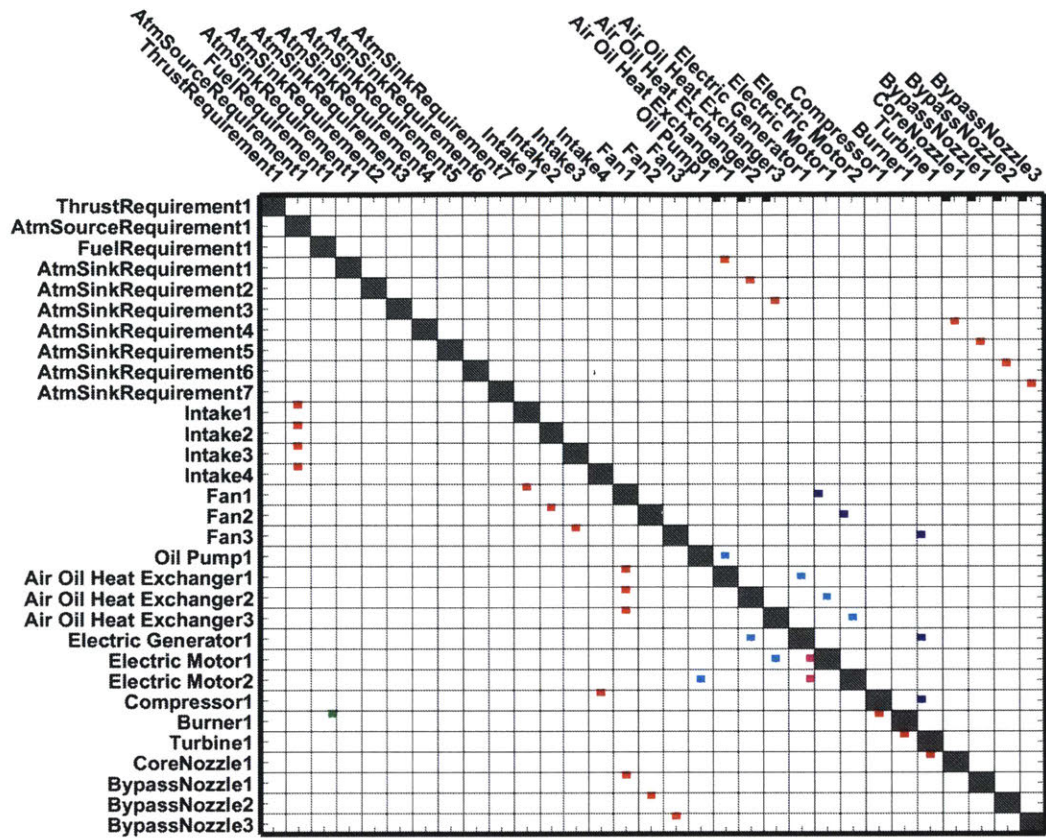


Figure 7.52: Architecture 49 (Turboelectric). Gas connections: **red**, shaft power connections: **blue**, mechanical connections: **black**, fuel connections: **green**, oil connections: **cyan**, electrical connections: **magenta**.

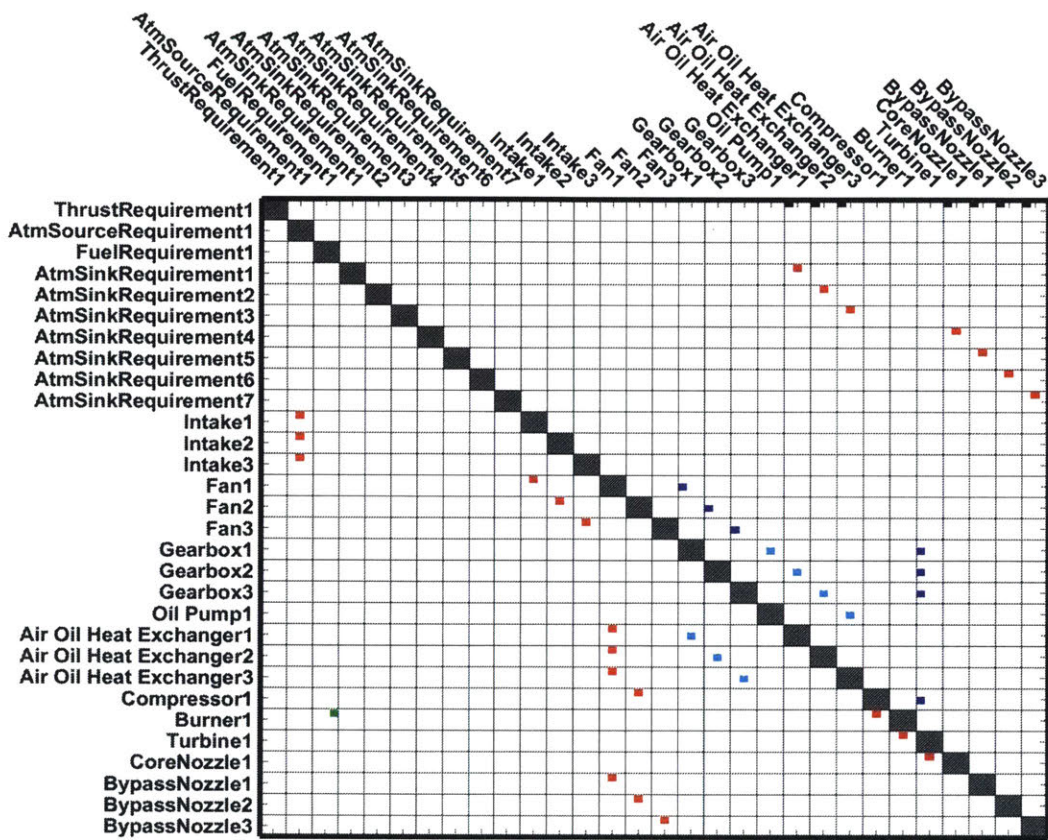


Figure 7.53: Architecture 50. Gas connections: red, shaft power connections: blue, mechanical connections: black, fuel connections: green, oil connections: cyan, electrical connections: magenta.

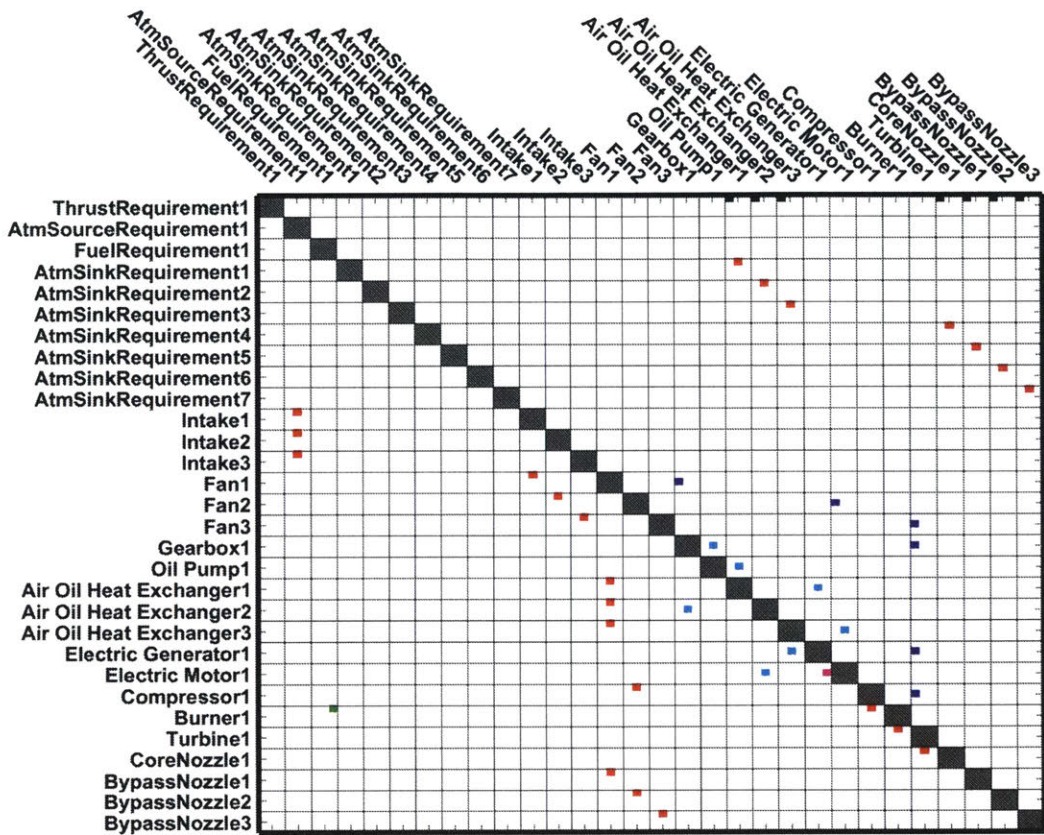


Figure 7.54: Architecture 51 (Turboelectric). Gas connections: red, shaft power connections: blue, mechanical connections: black, fuel connections: green, oil connections: cyan, electrical connections: magenta.

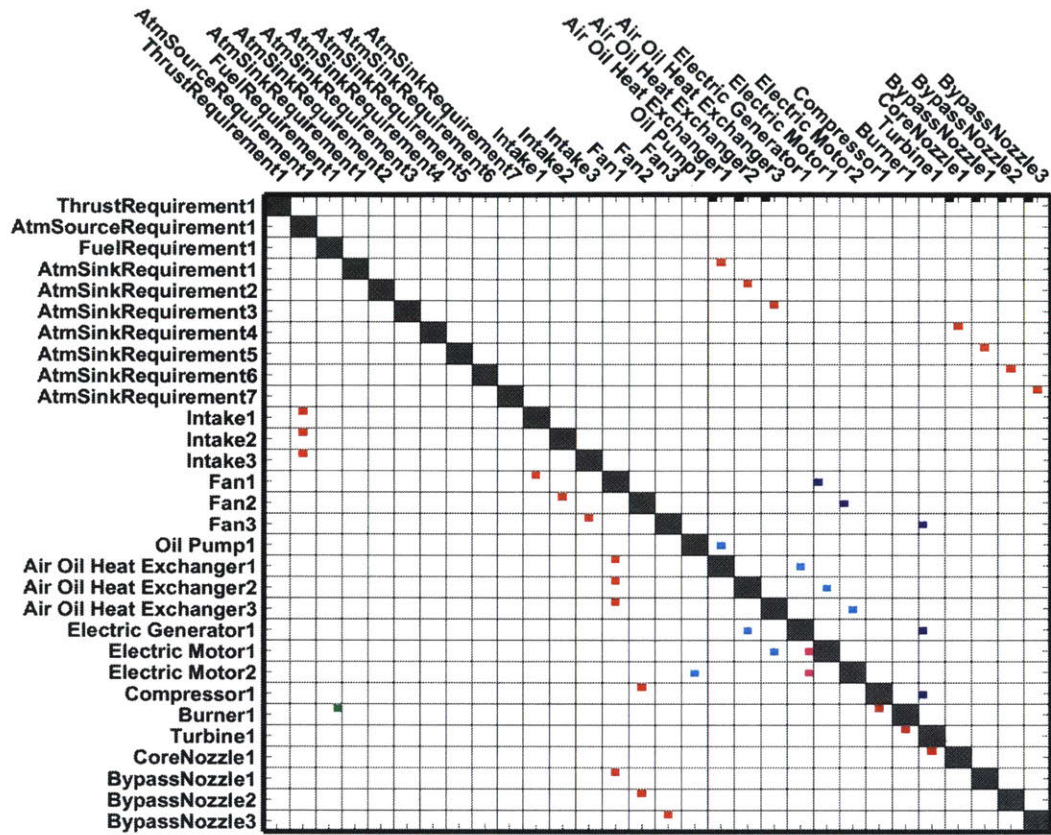


Figure 7.55: Architecture 52 (Turboelectric). Gas connections: **red**, shaft power connections: **blue**, mechanical connections: **black**, fuel connections: **green**, oil connections: **cyan**, electrical connections: **magenta**.

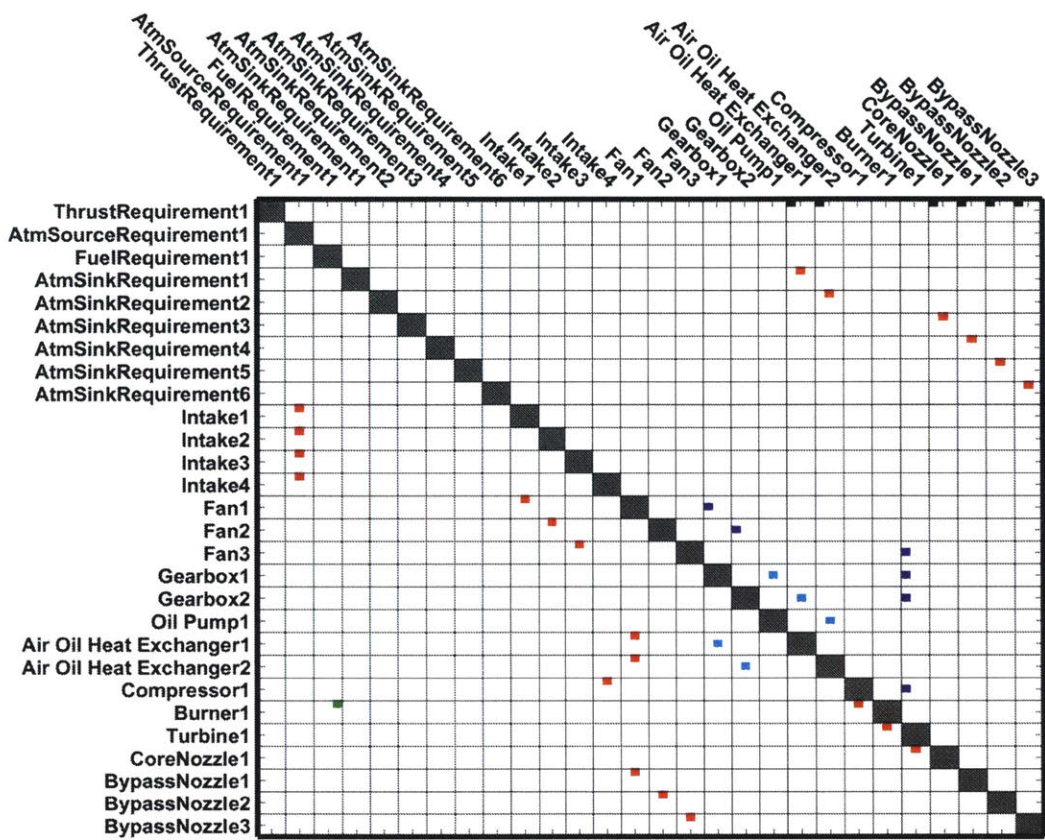


Figure 7.56: Architecture 53. Gas connections: **red**, shaft power connections: **blue**, mechanical connections: **black**, fuel connections: **green**, oil connections: **cyan**, electrical connections: **magenta**.

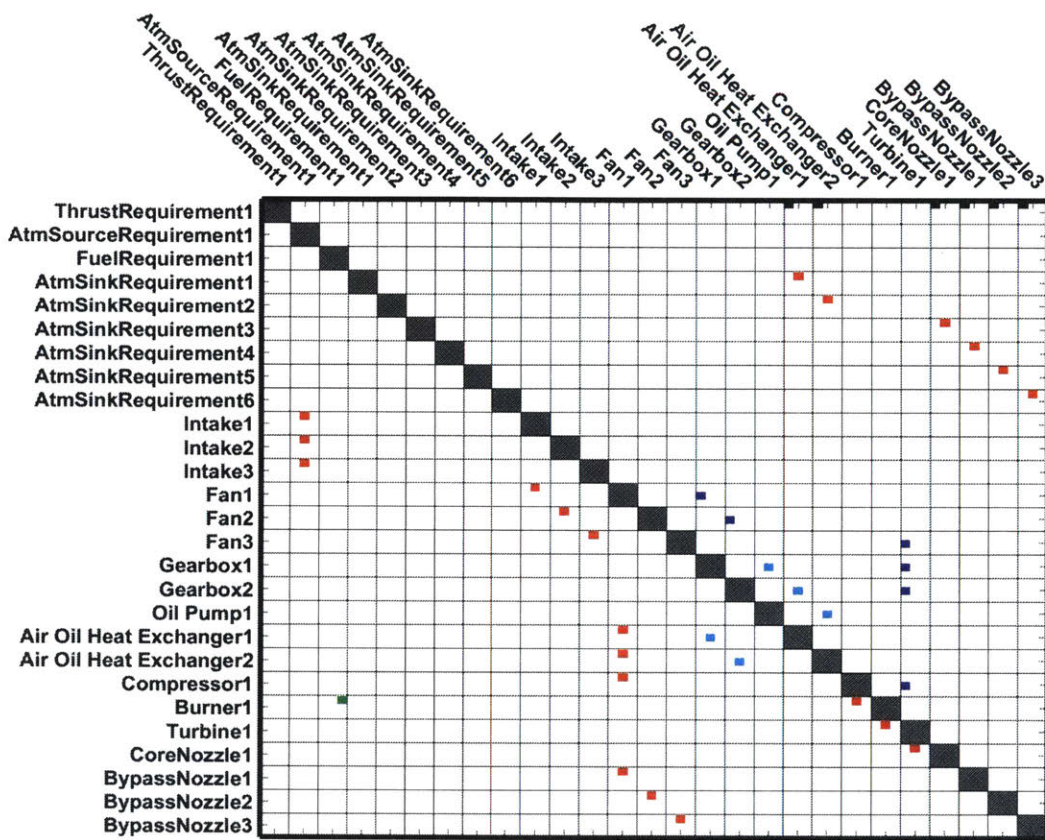


Figure 7.57: Architecture 54. Gas connections: red, shaft power connections: blue, mechanical connections: black, fuel connections: green, oil connections: cyan, electrical connections: magenta.

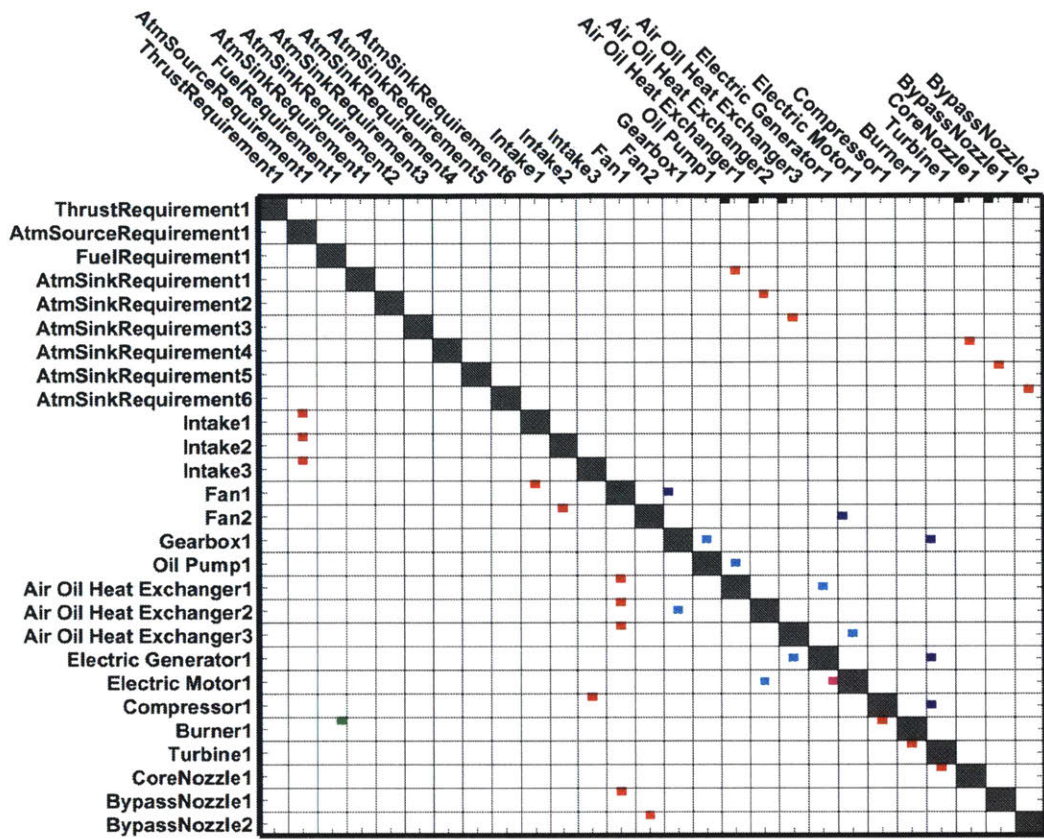


Figure 7.58: Architecture 55 (Turboelectric). Gas connections: **red**, shaft power connections: **blue**, mechanical connections: **black**, fuel connections: **green**, oil connections: **cyan**, electrical connections: **magenta**.

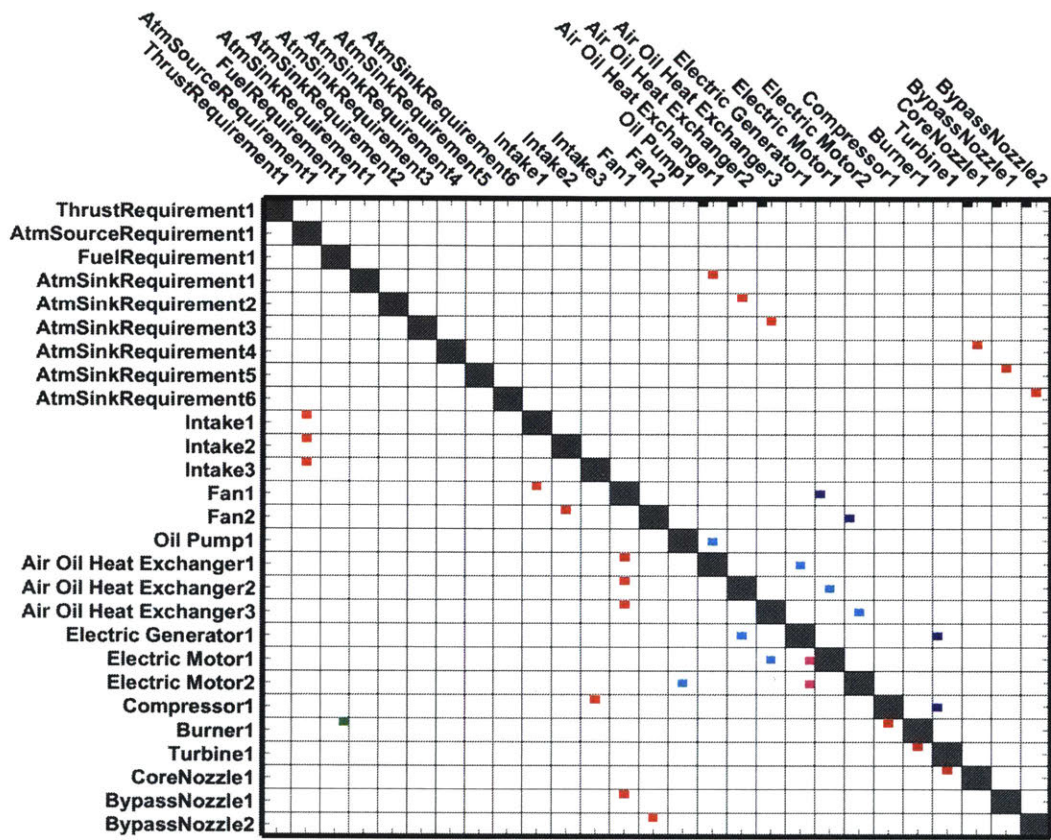


Figure 7.59: Architecture 56 (Turboelectric). Gas connections: **red**, shaft power connections: **blue**, mechanical connections: **black**, fuel connections: **green**, oil connections: **cyan**, electrical connections: **magenta**.

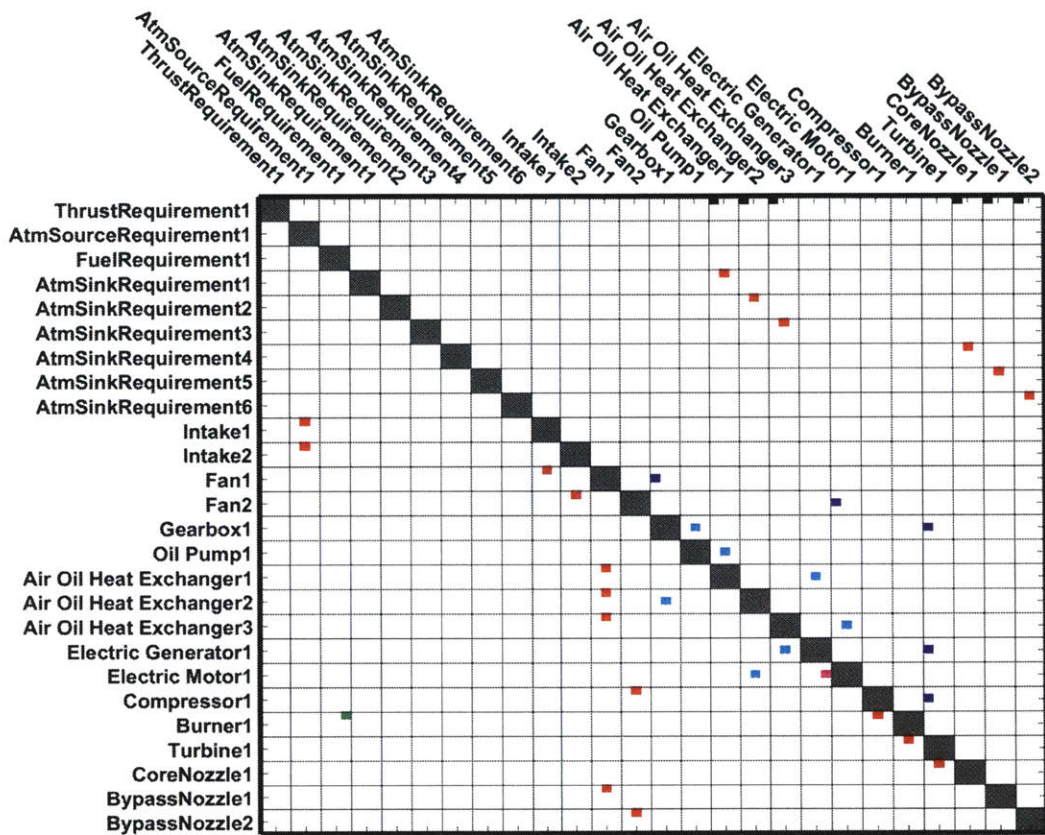


Figure 7.60: Architecture 57 (Turboelectric). Gas connections: **red**, shaft power connections: **blue**, mechanical connections: **black**, fuel connections: **green**, oil connections: **cyan**, electrical connections: **magenta**.

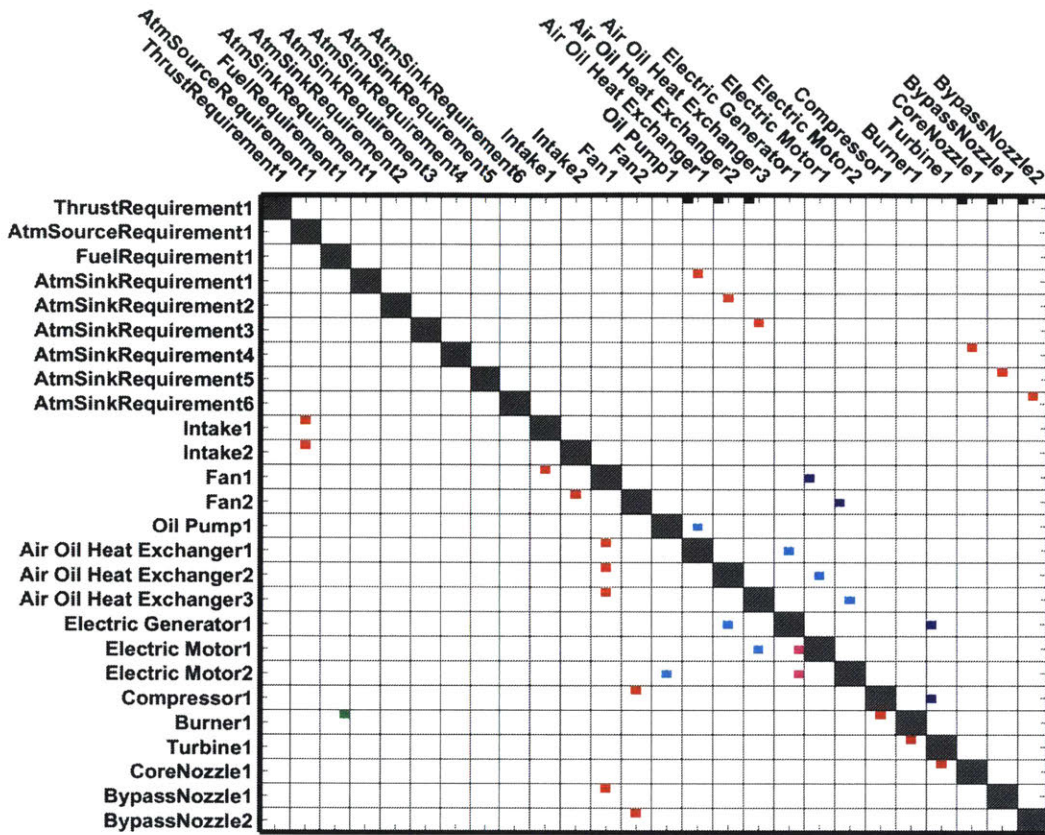


Figure 7.61: Architecture 58 (Turboelectric). Gas connections: **red**, shaft power connections: **blue**, mechanical connections: **black**, fuel connections: **green**, oil connections: **cyan**, electrical connections: **magenta**.

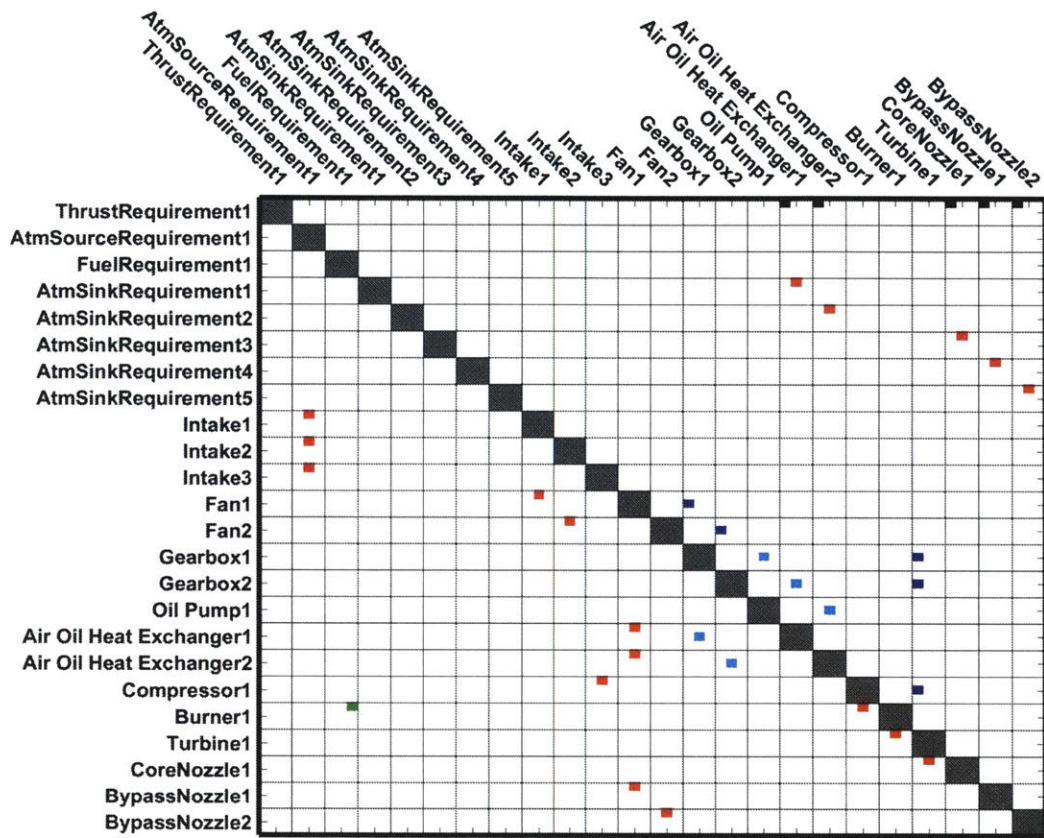


Figure 7.62: Architecture 59. Gas connections: red, shaft power connections: blue, mechanical connections: black, fuel connections: green, oil connections: cyan, electrical connections: magenta.

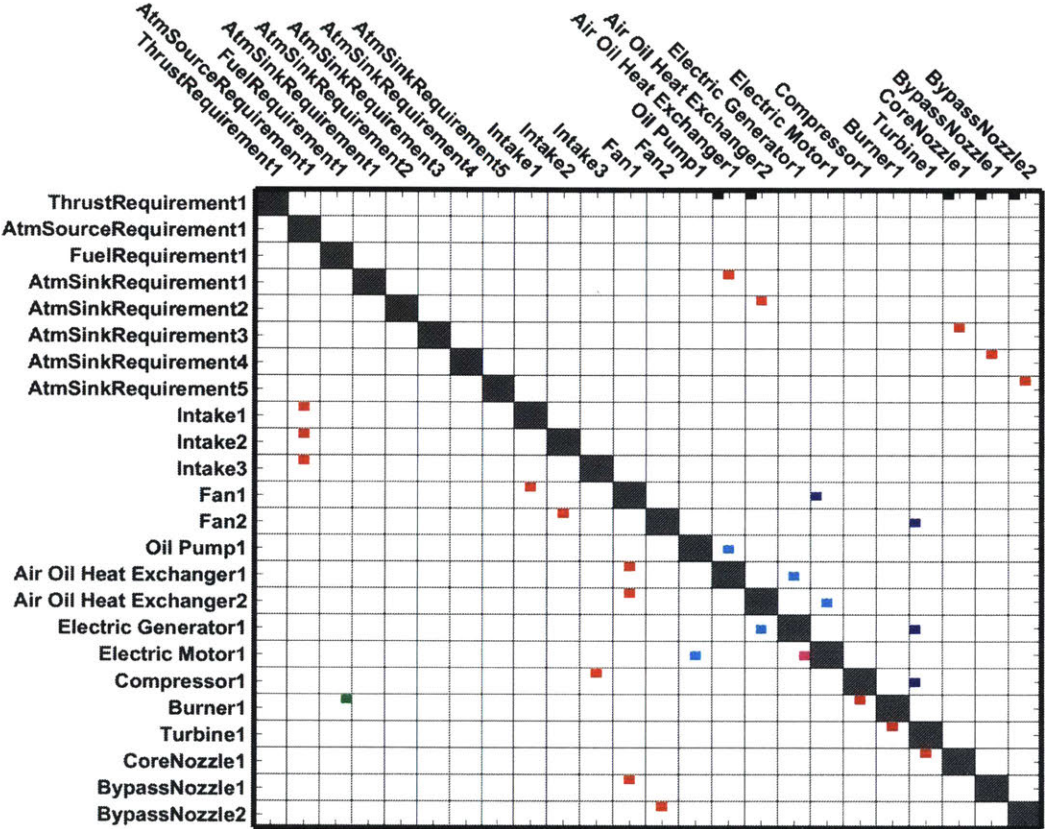


Figure 7.63: Architecture 60 (Turboelectric). Gas connections: red, shaft power connections: blue, mechanical connections: black, fuel connections: green, oil connections: cyan, electrical connections: magenta.

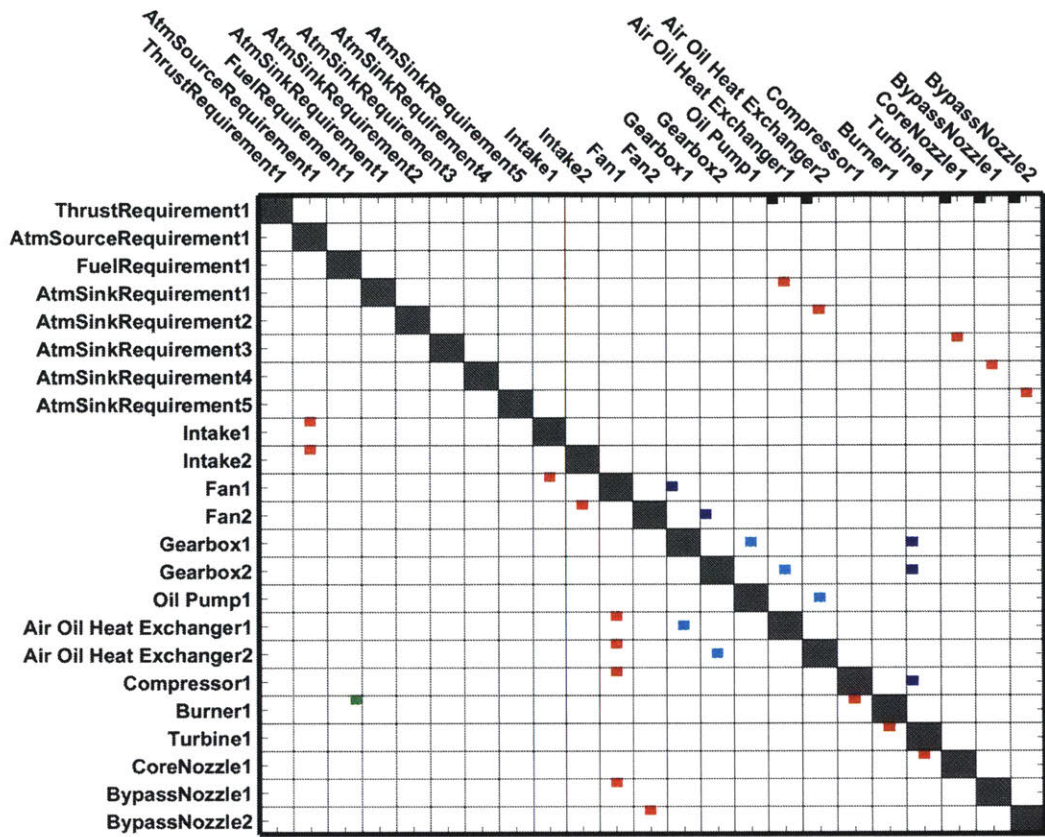


Figure 7.64: Architecture 61. Gas connections: **red**, shaft power connections: **blue**, mechanical connections: **black**, fuel connections: **green**, oil connections: **cyan**, electrical connections: **magenta**.

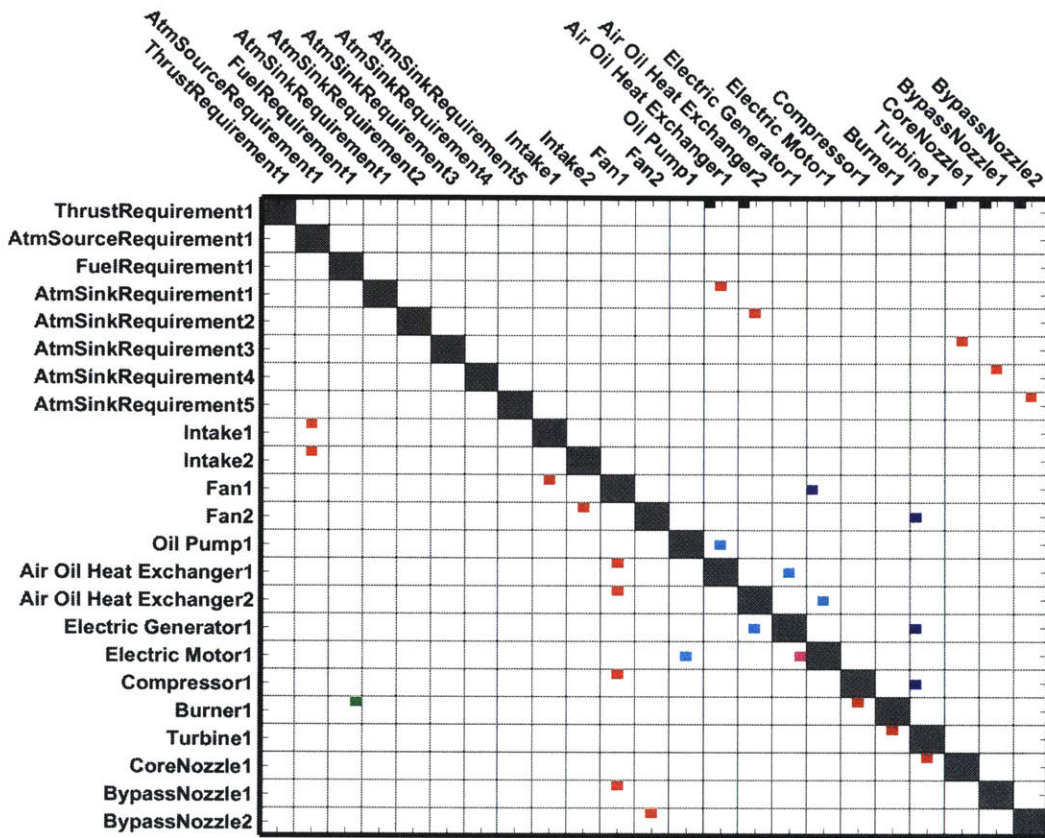


Figure 7.65: Architecture 62 (Turboelectric). Gas connections: **red**, shaft power connections: **blue**, mechanical connections: **black**, fuel connections: **green**, oil connections: **cyan**, electrical connections: **magenta**.

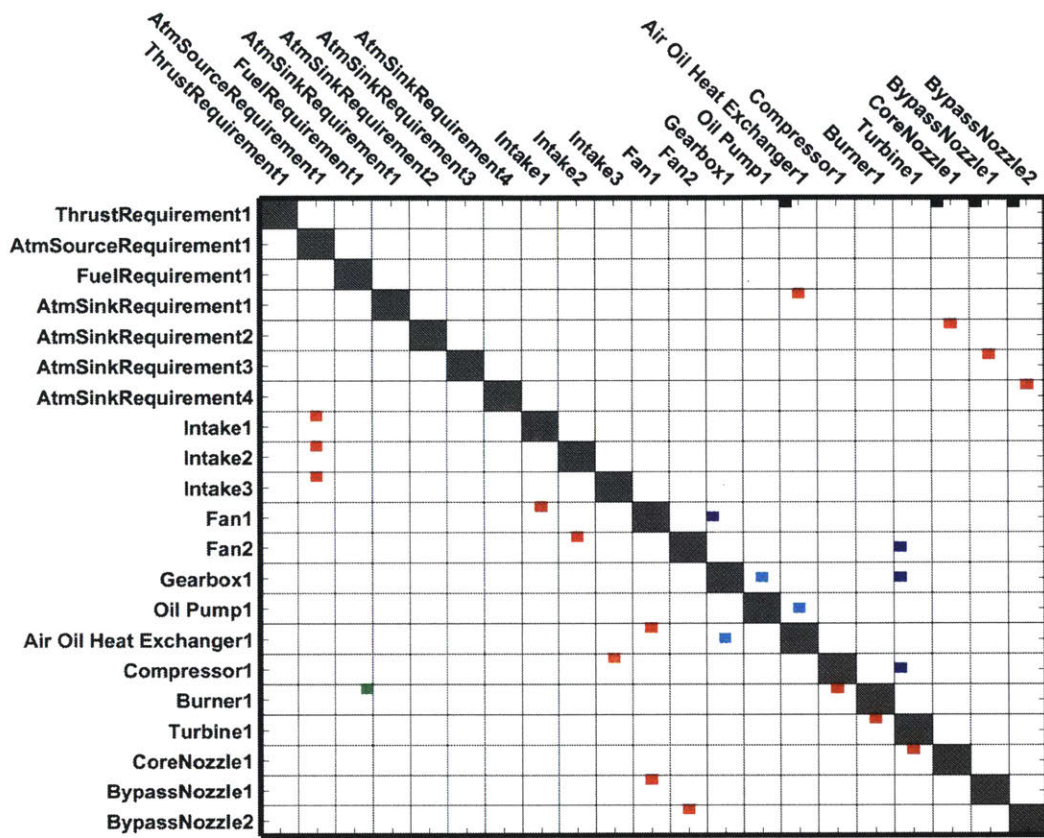


Figure 7.66: Architecture 63. Gas connections: **red**, shaft power connections: **blue**, mechanical connections: **black**, fuel connections: **green**, oil connections: **cyan**, electrical connections: **magenta**.

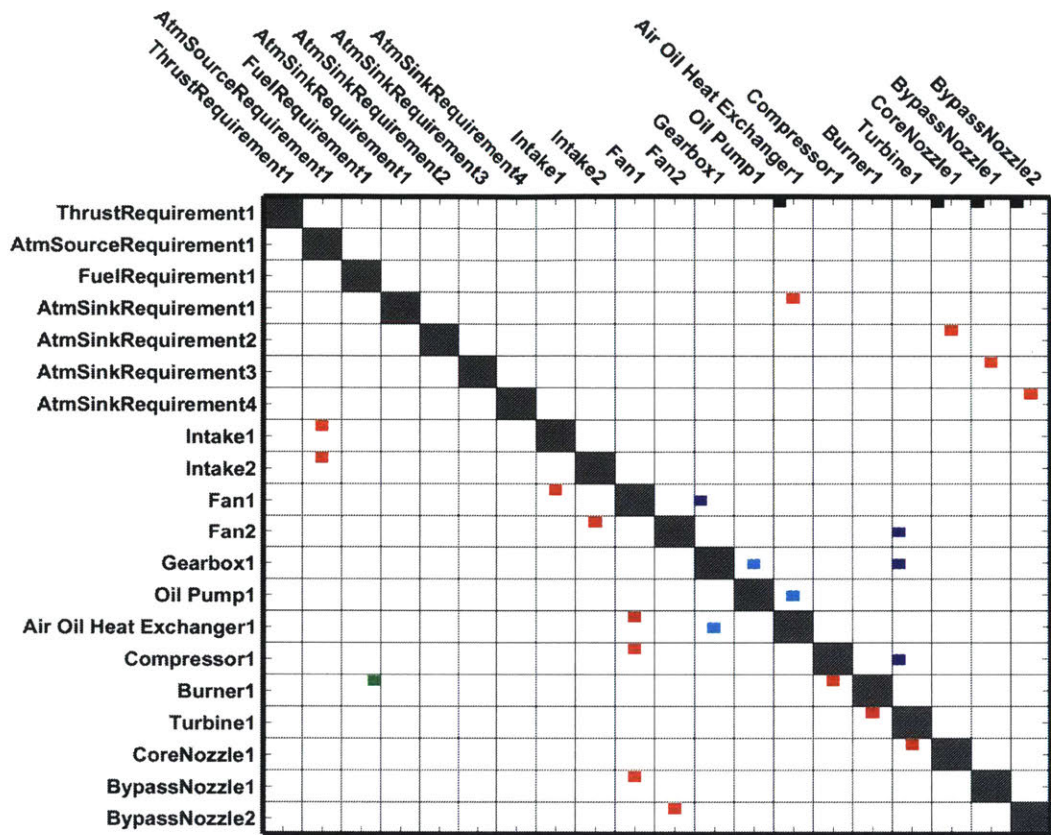


Figure 7.67: Architecture 64. Gas connections: red, shaft power connections: blue, mechanical connections: black, fuel connections: green, oil connections: cyan, electrical connections: magenta.

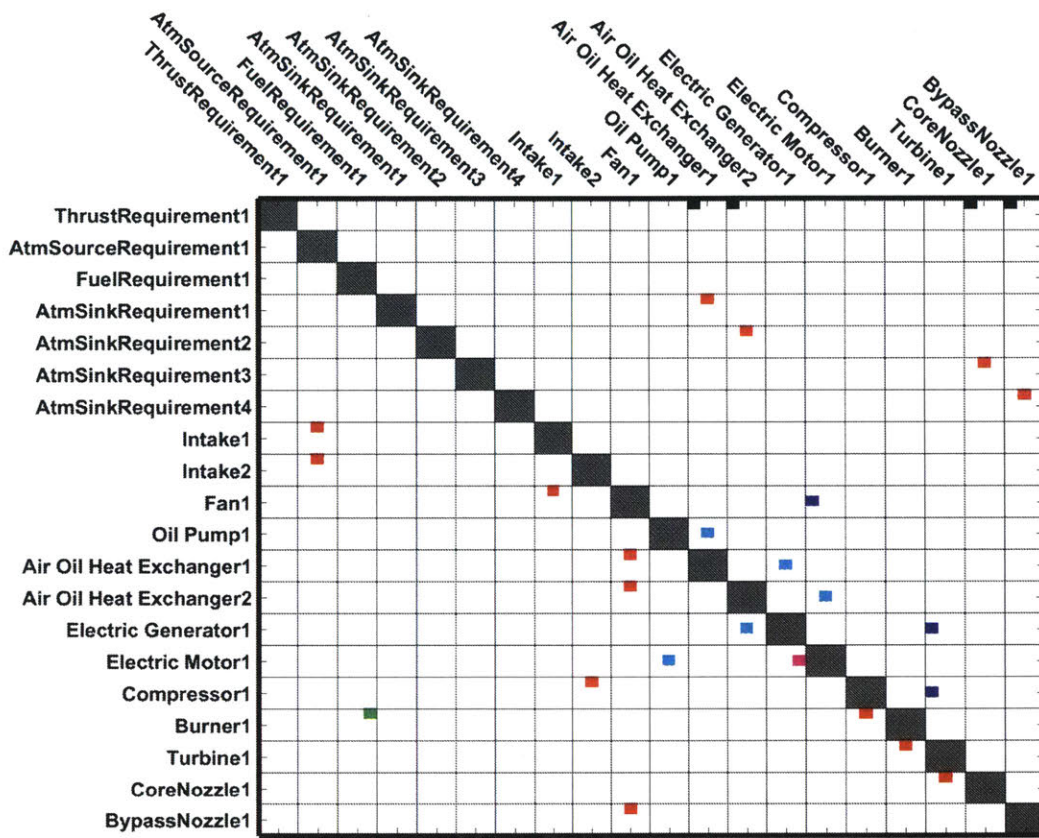


Figure 7.68: Architecture 65 (Turboelectric). Gas connections: red, shaft power connections: blue, mechanical connections: black, fuel connections: green, oil connections: cyan, electrical connections: magenta.

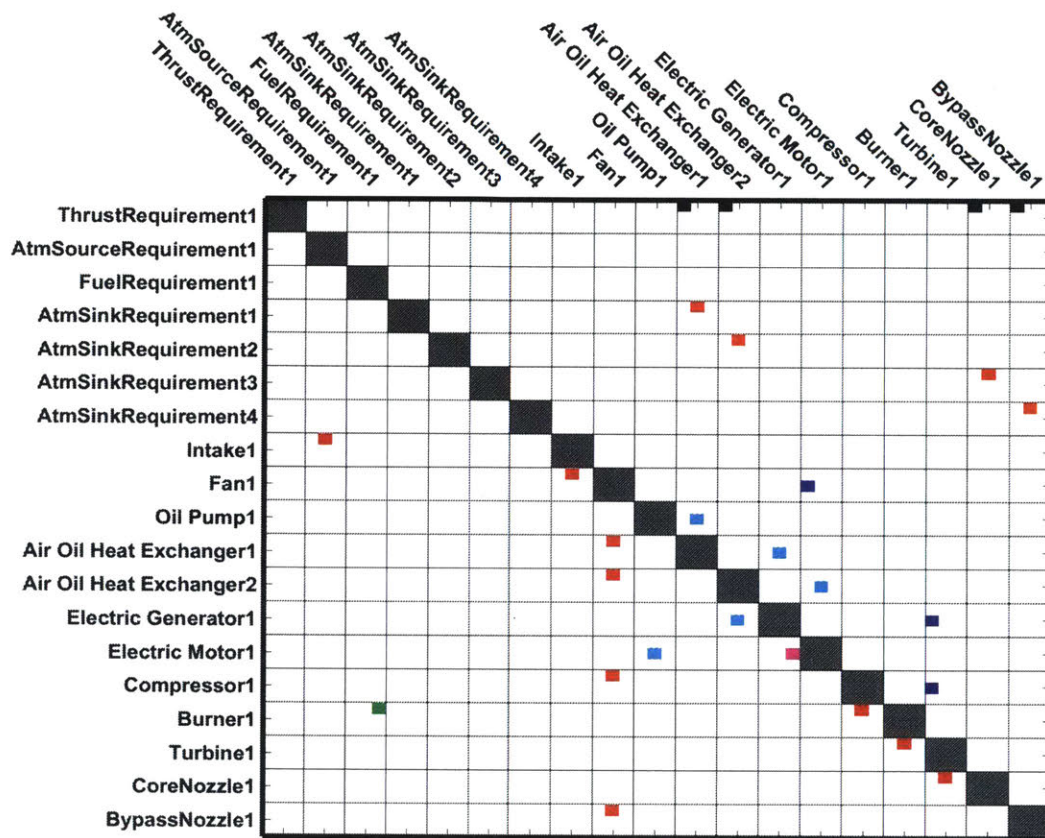


Figure 7.69: Architecture 66 (Turboelectric). Gas connections: red, shaft power connections: blue, mechanical connections: black, fuel connections: green, oil connections: cyan, electrical connections: magenta.

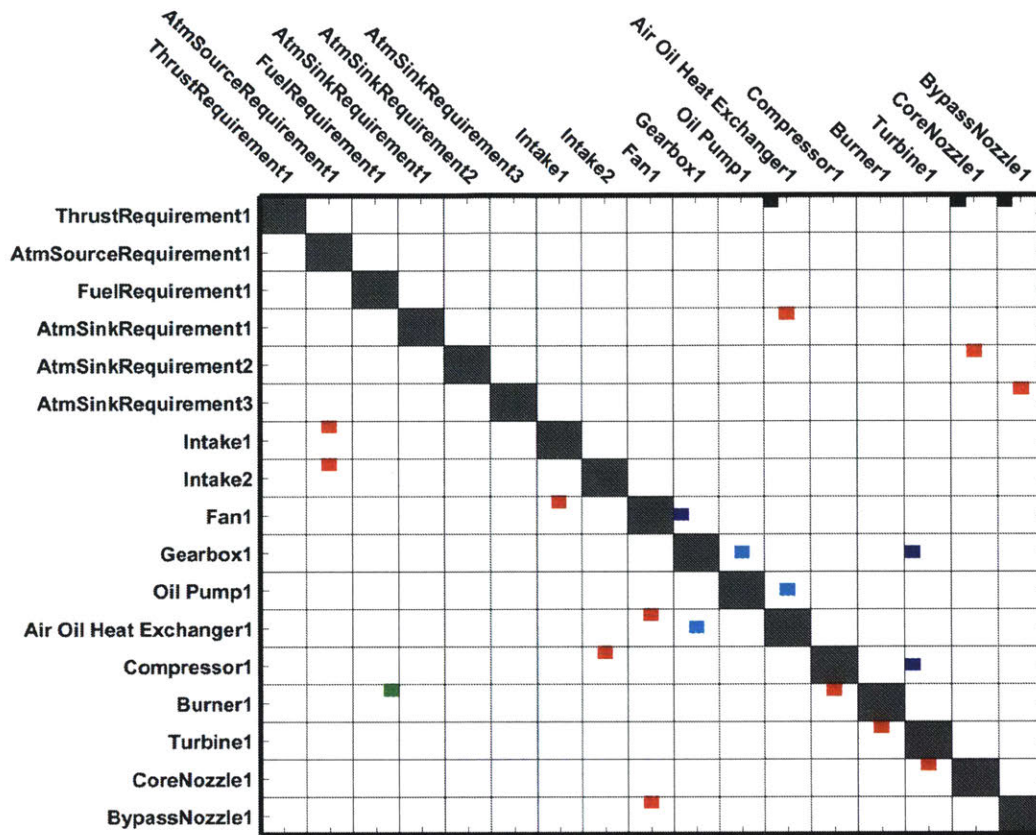


Figure 7.70: Architecture 67. Gas connections: red, shaft power connections: blue, mechanical connections: black, fuel connections: green, oil connections: cyan, electrical connections: magenta.

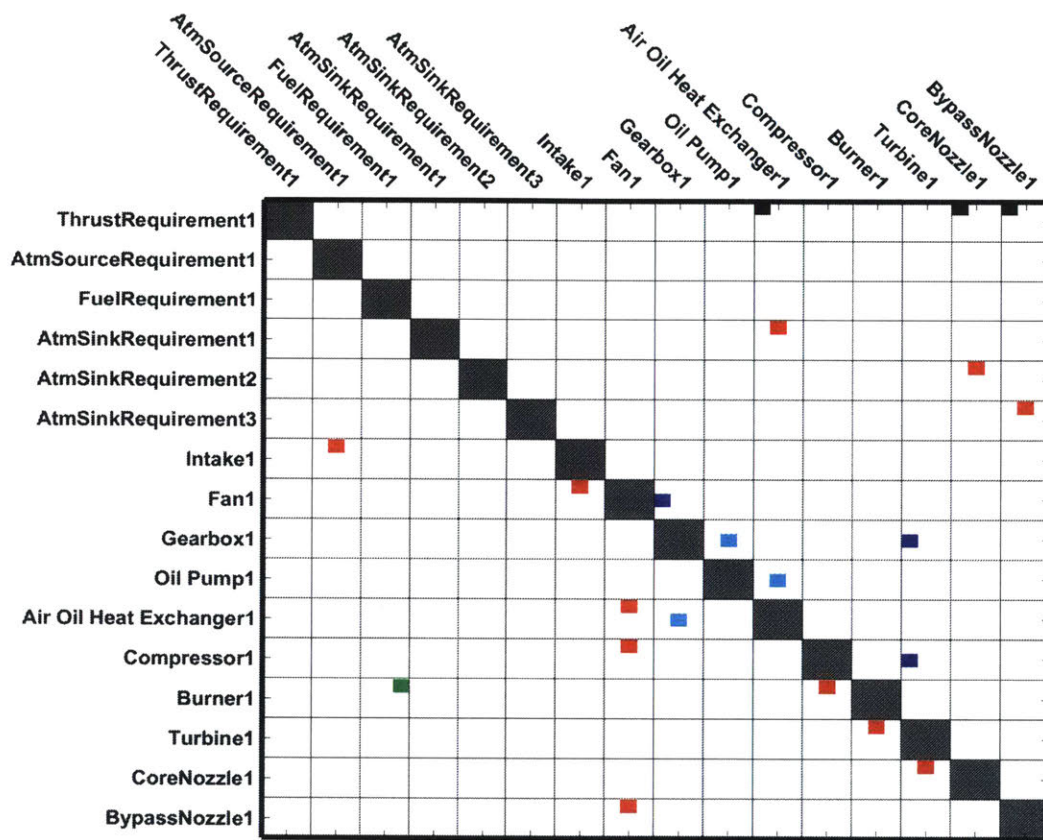


Figure 7.71: Architecture 68. Gas connections: red, shaft power connections: blue, mechanical connections: black, fuel connections: green, oil connections: cyan, electrical connections: magenta.

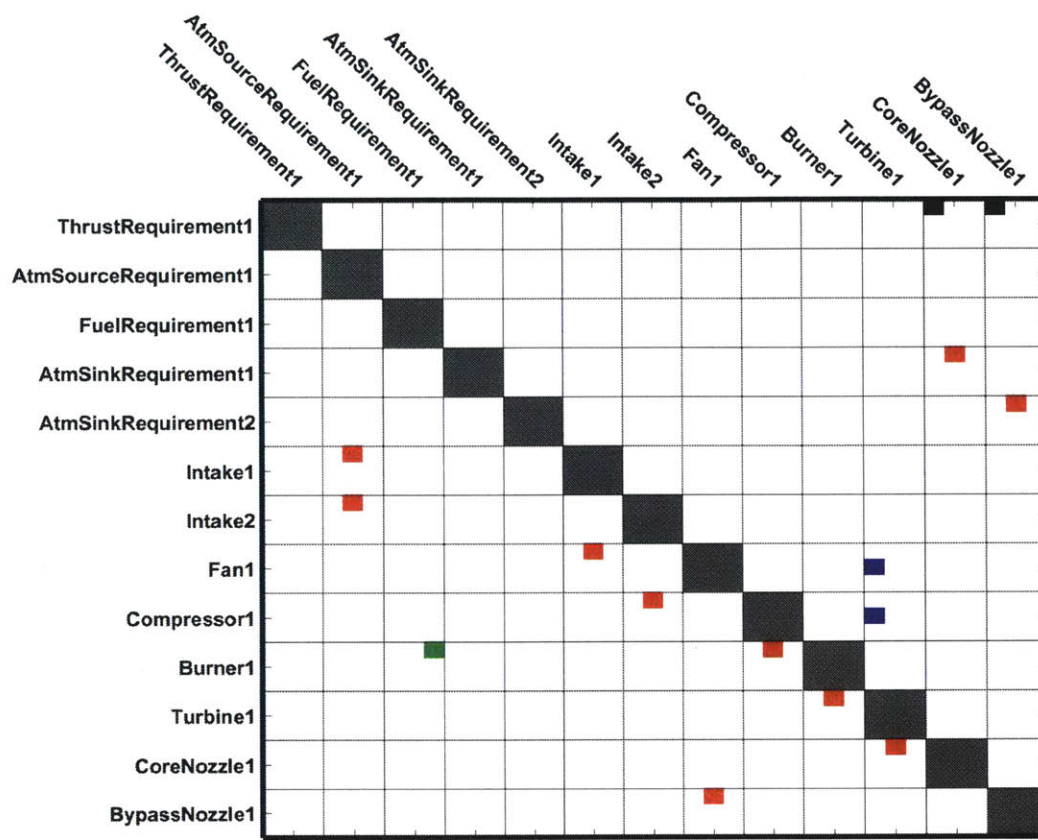


Figure 7.72: Architecture 69. Gas connections: **red**, shaft power connections: **blue**, mechanical connections: **black**, fuel connections: **green**, oil connections: **cyan**, electrical connections: **magenta**.

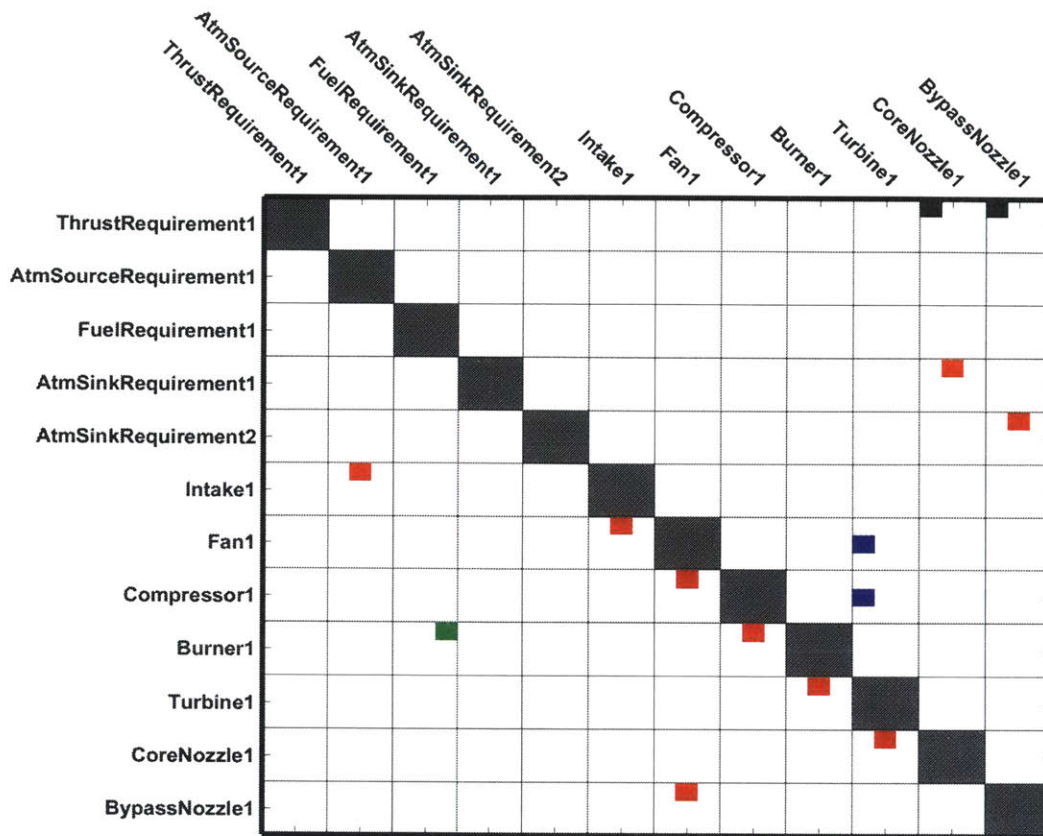


Figure 7.73: Architecture 70. Gas connections: red, shaft power connections: blue, mechanical connections: black, fuel connections: green, oil connections: cyan, electrical connections: magenta.

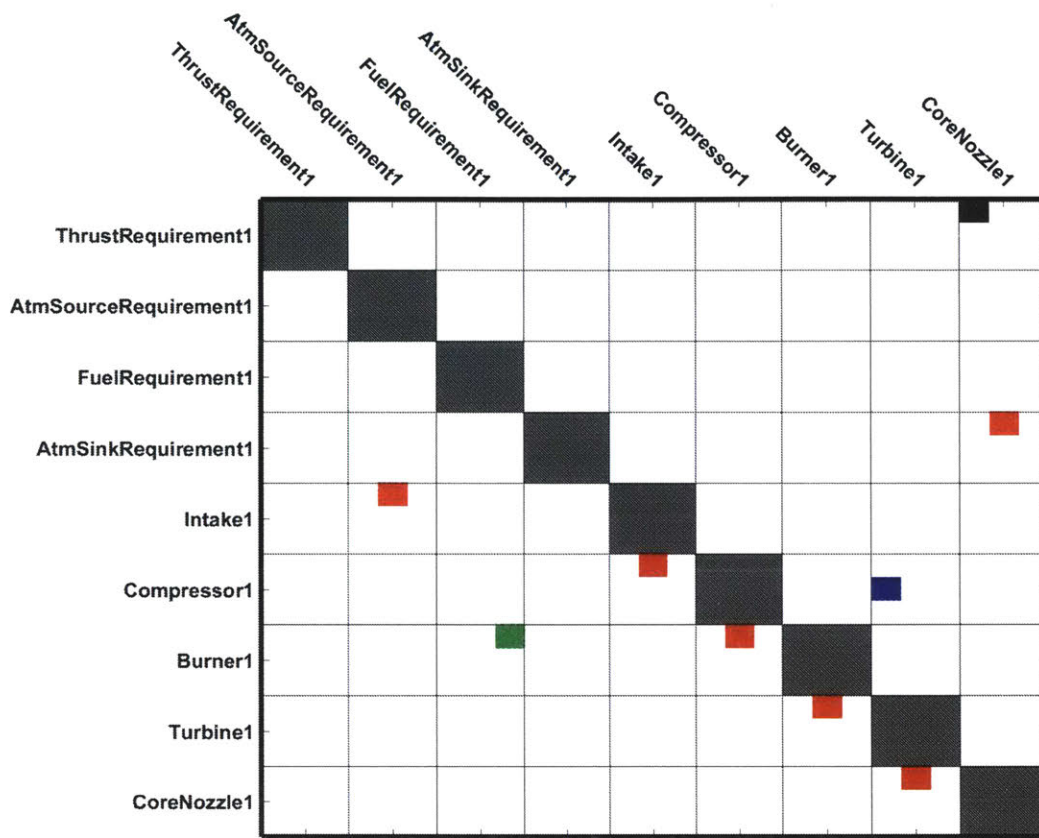


Figure 7.74: Architecture 71. Gas connections: **red**, shaft power connections: **blue**, mechanical connections: **black**, fuel connections: **green**, oil connections: **cyan**, electrical connections: **magenta**.