PROCEEDINGS OF MOSES2023 CONFERENCE
4TH INTERNATIONAL CONFERENCE ON MODELLING AND OPTIMISATION OF SHIP ENERGY SYSTEMS
26-27 OCTOBER 2023, DELFT, NETHERLANDS

Topology Generation of Naval Propulsion Architecture

Florian Dugast[a,*], Stéphane Bénac[b], Pierre Marty[a], and Pascal Chessé[a]

[a]Nantes Université, École Centrale Nantes, CNRS, LHEEA, UMR 6598, F-44000 Nantes, France
[b]Naval Group, CEMEP, Technocampus Ocean, Bouguenais, France
[*]florian.dugast@ec-nantes.fr

Abstract

Reducing shipping emissions at an affordable cost is critical and can be achieved through propulsion architecture optimization. The multiple choices of components and constraints to be fulfilled (required speed, fuel consumption, maintenance, ...) make architecture design increasingly complex. Some optimization methods have already been used, for example, to optimize diesel generators (number, type and load) for fuel consumption reduction. More complex architectures have been studied by including the absence or presence of some components in a superconfiguration but in the end the number of configurations remains limited. In this study, the ship architecture is not predetermined but is generated by a list of components associated with constraints and rules, making architecture creation more flexible. The algorithm written for this purpose follows the principles found in hybrid vehicle design but with adapted rules and components for naval applications. The main objective of this paper is to explain in detail the topology generation architecture algorithm rather than to find an optimal architecture for a specific ship. From this perspective, the test cases presented are general to demonstrate that the algorithm can be applied to various system configurations. The components are linked together based on their input and output energy type and the architecture is generated to comply with propulsion and hotel load requirements. Next, physical constraints are added to build realistic designs such as avoiding spurious redundant connections or defining the maximal occurrence for each component. All the constraints and the generation algorithm are written in Prolog. Two numerical applications are presented where the list of components covers different types of propulsion (mechanical and/or electrical with gas turbines and/or diesel engines) along with hotel load or heating requirements.

Keywords: Synthesis design; logic programming; naval propulsion architecture.

1 INTRODUCTION

In engineering design, system architecture can be increasingly complex to establish due to the multiple choice of components and the number of constraints to be fulfilled (performance, cost, environmental impact, energy savings, maintenance, etc) [1], [2]. In this context, the design of naval vessels is particularly critical because of their important role in transportation or military and the need for efficiency to meet greenhouse gas reduction requirements at an affordable cost. In general, the conception of any system relies on different levels such as topology, size and control [3], [4]. The topology refers here to the presence/absence of components and how they are connected to each other. The “size level” determines the number of each component in the architecture whereas some optimal design points are determined at the “control level”. For a naval design, the topology can refer to the type of propulsion (electric, mechanical or hybrid) [5] and how the hotel load is generated (combined with the propulsion system or not) whereas the number of diesel engines, for example, and their load are determined at the size and control level respectively. Note that the terminology ”topology, size, control” is mostly employed in hybrid vehicle design whereas ”synthesis, design, operation” are widely used in ship design literature [6], [7] referring to the same concepts. In this article, we focus on the synthesis level as it is a challenging topic that needs further development [6].

The architecture at the synthesis level is described as a list of components and connections between two components. Network representation is used for ship design in [8] to represent zone decks (e.g hangar, propulsion plant, flight deck) and their connectivity. The arrangement of the different nodes is evaluated based on logical and physical levels, representing the system connectivity and the spatial locations, respectively. The design exploration of the main users in ship architecture was also realized in [9]. The components are similar to those in [8] with some additional details on the propulsion

DOI: https://doi.org/10.59490/moses.2023.667

system : eight options available (COmbined Diesel And Gas, COmbined Gas And Gas, Hybrid Electric Drive or Integrated Propulsion System) with two main propulsion engines and four secondary engines. In these studies, the entire propulsion system is considered as one object in the ship structure. In our work, we intend to explore in more detail the architecture of the propulsion system as it plays a critical role in ship performance, so the next articles presented will focus on this topic. In [10], the connections in ship energy distribution systems were determined by a combination of an adjacency matrix and a genetic algorithm. The adjacency matrix exhibits unfeasible constraints whereas the genetic algorithm minimizes the number of connections or maximizes system reconfigurability. The number of possibilities for different connections configurations is very important but the number of each component is fixed (i.e 4 DG, 4 switchboards, etc) which can limit design space exploration. Compared to [10], the number of components is not fixed in [11] but calculated using an optimization process. However the absence/presence of components is predetermined by a "superconfiguration" and at the end the flexibility of the topology in the architecture is relatively low, relying only on the presence/absence of HRSG units and with only one HRSG per engine. Some extensions of this work have been published to simulate the use of gas turbines [12] or to add ship resistance for the optimization of ship speed [13]. A superconfiguration is also used in [14] to incorporate ORC, Stirling units or thermal storage systems. Depending on the number of components and interactions between them, the creation of such a superconfiguration can become cumbersome or intractable.

In this article, we propose an algorithm for the creation of architecture topology by searching for both the number of possible components and the connections between them to allow the generation to go over a large design space. The objective is to obtain a ship architecture without the use of a predefined superconfiguration, but with only a list of components and generic constraints included in a constraint-satisfactory problem. Automation of the creation of architecture connections can be useful to handle an important number of components and obtaining ship architecture with all possible configurations in an efficient way. Most of the articles published on this topic have focused on hybrid vehicle design [15]–[19] due to its important market. For example, Wijknet et al. [19] used the Prolog (logic programming) language [20] to implement constraints for powertrain architecture generation in order to reduce the design space. More details regarding the architecture generation algorithms for hybrid vehicles can be found in the review of Hu et al. [18].

In this work, the architecture is generated with similar methods developed for hybrid vehicle design (constraint-satisfaction problem solved in Prolog) but with an application on ship architecture. The connections between components are realized based on identical input/output energy types and the algorithm is written to achieve both propulsion and hotel load requirements. Other constraints are added to obtain feasible and realistic ship architectures. The algorithm for the architecture topology generation is presented in Section 2 and two numerical examples are given in Section 3 as an illustration of the method.

2 ARCHITECTURE TOPOLOGY GENERATION ALGORITHM

The architecture is described by a list of connections between blocks, where a block can be a resource, a component or a user. In this case, a component can be used either for energy conversion or to gather energy from different sources.The block database is user-defined and the list of connections is obtained by logic programming implemented in Prolog. In this section, the constraints and process developed to generate the architecture will be detailed. The reference example used throughout the article is based on the list of blocks in Table 1:

full name	**input**	**output**
Propulsion (prop)	sea	/
Hotel load (hl)	el.network	/
Propeller (ppel)	mech.en	sea
Elec. motor (elm)	el.network	mech.en
Elec. network (eln)	el.en	el.network
Gas turbine (gt)	fuel	mech.en
Diesel engine (de)	fuel	mech.en
Alternator (al)	mech.en	el.en
Fuel tank (fuel)	/	fuel

Table 1: Reference block list

The input and output of a block can be an energy type or a specific interface name. In Table 1, a block with no input is a resource (e.g Fuel tank), a block with no output is a user (e.g Propulsion or Hotel load) and the remaining blocks are components.

This list has been composed to be representative of the components for the energy propulsion system of a large cruise or a naval vessel with possible high requirements on ship velocity or hotel load. With the inputs/outputs listed in Table 1, the electricity supply can be used for either propulsion or hotel load needs.

2.1 Constraints implementation

The different physical or logical constraints used to generate the architecture are :

1. Constraint 1 : Two components are linked together if they share a common input/output, see Fig. 1

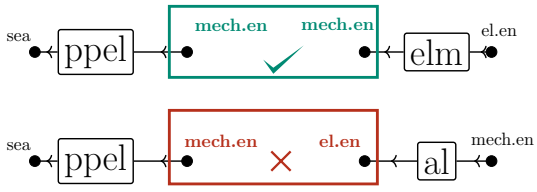

Figure 1: Constraint on connection input/output

2. Constraint 2 : From empirical knowledge, it is assumed that the same connection cannot occur twice in the elementary chain to avoid unrealistic architectures. For example in Fig. 2 on top, if an electric motor is connected to an alternator, this alternator is not allowed to be connected to an electric motor again.

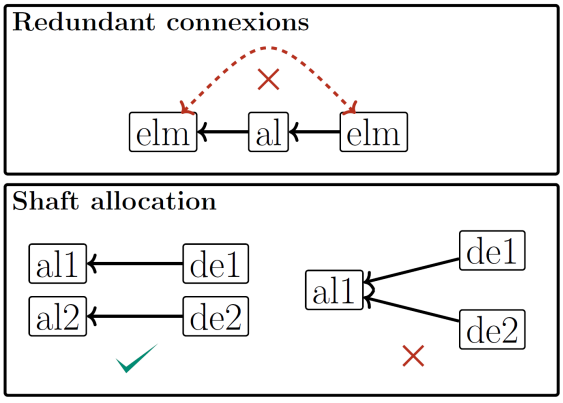

Figure 2: Constraints on redundant connection and shaft allocation

3. Constraint 3 : Only one mechanical component is allowed per shaft; for example two diesel engines cannot be connected to the same alternator (see Fig. 2 at the bottom).

4. Constraint 4 : The configuration of the propellers must exhibit a symmetry pattern, as shown in Fig. 3.

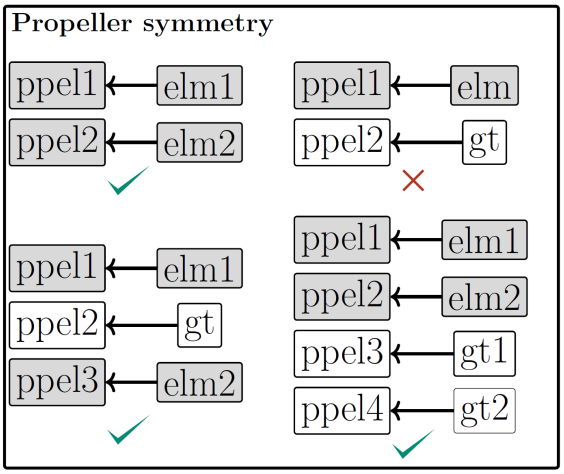

Figure 3: Constraint on propeller arrangement

5. Constraint 5 : A limitation on the maximal number of each component is provided (see Table 2 for the blocks listed in Table 1).

Block name	Max1	Max2	Max3
Propulsion	1	1	1
Hotel load	1	1	1
Propeller	4	2	4
Electrical motor	4	2	4
Electrical network	1	1	2
Gas turbine	4	2	4
Diesel engine	4	6	4
Alternator	4	6	4
Fuel tank	1	1	1

Table 2: Maximal occurrence of the reference block list

The maximal occurrence of users (Propulsion, Hotel load) and resources (Fuel tank) is set to 1. For the components, different cases will be studied :

- Max1 configuration : all the occurrences are arbitrarily set to four apart from the electrical network, as a good compromise between the diversity and compactness of the architecture
- Max2 configuration : the maximal occurrences are based on values found in literature ([13], [21])
- Max3 configuration : identical to Max1 but with two electrical networks

2.2 Architecture construction

The generation of the architecture is divided in five main steps :

1. Research of the elementary chains required for each user, the definition of an elementary chain will be given later;
2. Construction of the architecture structure based on the maximal mutualization (i.e with the smallest possible number of components) of all the elementary chains;
3. Determination of the occurrence of each group found in the architecture structure. This choice considers constraints based on the maximal number of each component;
4. Creation of the architecture connections;
5. Creation of the diagram of nodes and edges for visualization.

In step 3, the system of equations is generated manually from step 2 at this point. Au automation of this step is currently under development. For step 5, the graph generation algorithm has not been deeply investigated at the moment so it is sometimes necessary to manually adjust the node locations to avoid overlapping. Otherwise all the other steps are done automatically by the algorithm. After this process, the architecture is given as a list of connections between components. Each component name is composed of the component key and a unique number to distinguish between all the components.

2.2.1 Creation of elementary chains

An elementary chain (EC) consists of a minimal number of connections that can bring power to a user in the ship architecture, that is, propulsion or hotel load, in the example given above. Starting from a user, the connections are established based on the input and output of each component. Components A and B are connected if the output of A is equal to the input of B based on Constraint 1. If several connections are possible for the block A, each connection creates a different EC. In Table 1, there is only one input and one output for each component but it is possible to specify several inputs or outputs if necessary. The research on the corresponding inputs and outputs is also restrained by Constraint 2 and the EC ends when a block without output is encountered. The list of ECs based on the example in Table 1 is shown in Fig. 4.

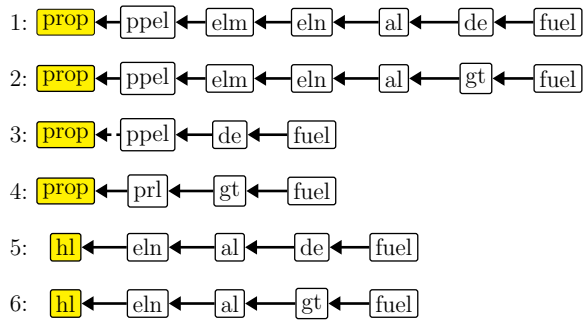

Figure 4: List of ECs for the reference block list

The first four elementary chains represent different ship propulsion alternatives : electric (ECs 1 and 2) or mechanical (ECs 3 and 4). For both options, diesel engines or gas turbines can be used to deliver mechanical power. The last two elementary chains provide an hotel load for the ship, with either diesel engines or gas turbines.

2.2.2 Construction of the architecture structure

The structure is a combination of all the elementary chains found for the set of components. The elementary chains have some components in common so one component can belong to several elementary chains if it does not violate any constraint, it is called mutualization in the following of the article. For example, the components diesel engine and alternator are both in the elementary chains 1 and 5, see Fig. 5.

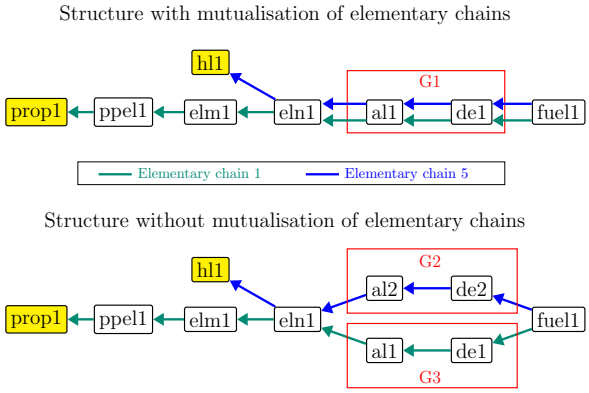

Figure 5: Mutualization of elementary chains in the architecture structure

At this stage, two options are possible to represent it : a mutualization of the diesel-alternator to combine both EC 1 and EC 5 into one group (see G1 on top of Fig. 5) or a separation into two groups for each elementary chain (see G2 and G3 at the bottom of Fig. 5). When this case is encoun-

tered, the mutualization is always chosen to avoid duplicate architectures in the end. Indeed after the creation of the architecture structure, the occurrence of each components is decided and for example, the architecture with one group G2 and no group G3 would be the same than with no group G2 and one group G3. However the same configuration is only represented by one group G1 for the structure with mutualization which is more advantageous. For the set of components indicated in Table 1, the architecture structure is presented in Fig. 6. One can see that for the electricity generation with the diesel engine and the gas turbine, one alternator is dedicated to each component due to Constraint 3. The presence (=1) or absence (=0) of each elementary chain in the architecture structure is summarized in Table 3.

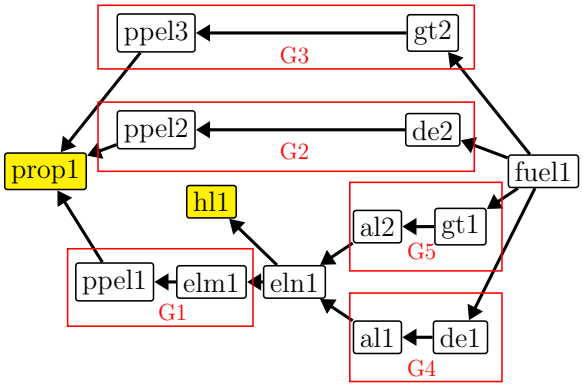

Figure 6: Architecture structure 1

EC	G1	G2	G3	G4	G5
1	1	0	0	1	0
2	1	0	0	0	1
3	0	1	0	0	0
4	0	0	1	0	0
5	0	0	0	1	0
6	0	0	0	0	1

Table 3: Relationships between groups and elementary chains

2.2.3 Occurrence of the architecture structure groups

After the creation of the architecture structure, the next step to obtain a complete definition of the architecture topology is to define the occurrence of each component in the structure based on the maximal number of each component defined by the user and on some constraints. When some components are linked together due Constraint 3, they are linked together in a group because their occurrence must be the same. The different groups are represented by red boxes in Fig. 6 and the number of occurrences of each group Gi is indicated as N_{Gi}. This vector is determined by the following system of equations :

$$\begin{aligned} & N_{Gi} \in [0, \infty],\ i \in [1, 5] \\ & 0 < N_{G4} + N_{G5} \leq N_{\text{al}}^{\max} \\ & 0 < N_{G1} + N_{G2} + N_{G3} \leq N_{\text{ppel}}^{\max} \\ & N_{G2} + N_{G4} \leq N_{\text{de}}^{\max} \\ & N_{G3} + N_{G5} \leq N_{\text{gt}}^{\max} \\ & \text{checkSymmetry1}(N_{G1}, N_{G2}, N_{G3}) \end{aligned} \tag{1}$$

Generally the number of occurrence of one group can be zero in order to indicate the absence of a specific part of the architecture structure. However the users needs (propulsion and hotel load) must always be met, that is why the constraints $N_{G1} + N_{G2} + N_{G3} > 0$ and $N_{G4} + N_{G5} > 0$ exists, due to the propulsion and hotel load requirement respectively. The function *checkSymmetry1* verifies Constraint 4 by counting the number of pairs of each different components connected to the propellers.

From this point, the exhaustive search for the N_{Gi} is performed in Prolog based on the constraints in (1). Note that the N_{Gi} could also be found by an optimization process if a physical model and data are available but it is beyond the scope of this article.

2.2.4 Creation of architecture connections

Once the number of each group is determined, there is a potential last degree of freedom in the architecture generation regarding how two components from different groups are connected together. For example, if there is more than one electrical network in the architecture and several alternators, one could find different ways of connecting these components, as shown in Fig. 7.

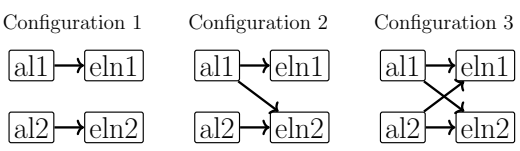

Figure 7: Architecture connections configurations

Note that some configurations are not represented in this figure as there would be similar to one of the three configurations already drawn by assuming that components 1 and 2 are identical. One drawback of these different configurations is the requirement of additional variables, that is the energy

partition between the different components if the architecture is to be calculated with a physical model. In addition, it can lead to identical performance at the end depending on the energy partition. For example, the energy gathered in eln1 and eln2 will be the same for configurations 1 and 3 if the energy is equally balanced in configuration 3. Therefore, the maximal number of electrical networks has been usually set to 1 in Table 2 and Table 6. A study has been added about the use of two electrical networks if redundancy is a critical criterion in the ship design.

3 ARCHITECTURE GENERATION RESULTS

In the following numerical cases, calculations and timings have been performed on a standard laptop computer.

3.1 Reference case

Based on the architecture structure presented in Fig. 6, different architectures are shown in Fig. 8.

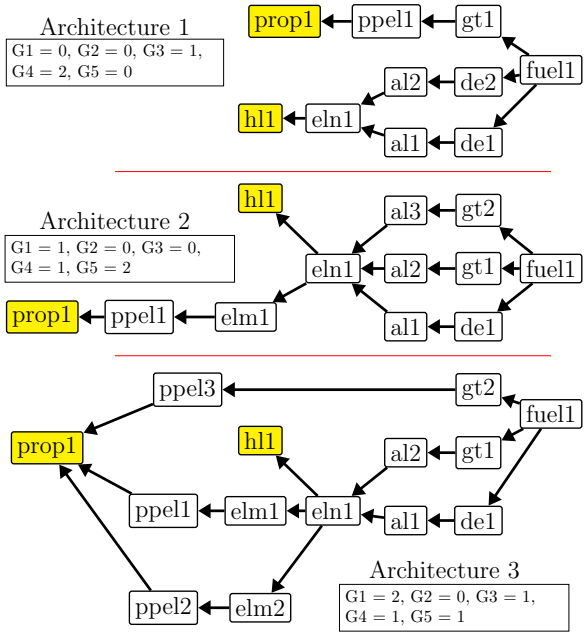

Figure 8: Architecture generation results 1

Architectures 1, 2 and 3 exhibit a pure mechanical, a pure electrical and a mechanical-electrical propulsion, respectively. The number of architectures obtained and the corresponding computational time depend on the maximal occurrence of each components, and the results have been gathered in Table 4.

	Max1	Max2	Max3
Number of architectures	226	82	1888
Computational time (ms)	29	9	98

Table 4: Architecture results for the reference case

3.2 Case 2

The same methodology is applied to a more complex set of components to evaluate the flexibility of the algorithm. This example is referred to as *Case 2* in the following section. The list of components still refers to a standard ship architecture propulsion but it also considers the production and circulation of heat. The heat can be used from the engines to produce steam, which can then be used either for heating or as an input to a steam turbine. The list of components for this second case along with their inputs and outputs are given in Table 5.

full name	input	output
Propulsion (prop)	sea	/
Hotel load (hl)	el.network	/
Heating (heat)	steam	/
Propeller (ppel)	shaft	sea
Shaft (shaft)	mech.en	shaft
Elec. motor (elm)	el.network	mech.en
Elec. network (eln)	el.en	el.network
Gas turbine (gt)	fuel	mech.en,heat
Steam turbine (st)	steam	mech.en,heat
HRSG (hrsg)	water,heat	steam
AB (ab)	fuel,water	steam
Diesel engine (de)	fuel	mech.en,heat
Alternator (al)	mech.en	el.en
Fuel tank (fuel)	/	fuel
Water tank (water)	/	water

Table 5: Case 2 : block list

Figure 9: Elementary chain example for Case 2

HRSG and AB represent the heat recovery steam generator and auxiliary boiler, respectively. One change to the reference case is the use of several keywords for some components in input or output. For example, both water and heat are required to produce steam in the HRSG component. It means that on the elementary chains, one component can have

several connections, as seen in Fig. 9. The number of elementary chains for this case is 18 compared to 6 for the reference case : 10 for propulsion, 5 for hotel load and 3 for heating. Also in this example, multiple components can be connected to the same shaft but the diesel engines and turbines cannot be connected simultaneously on a drive shaft and on a alternator. Based on the list of components and on these rules, the architecture structure is shown in Fig. 10.

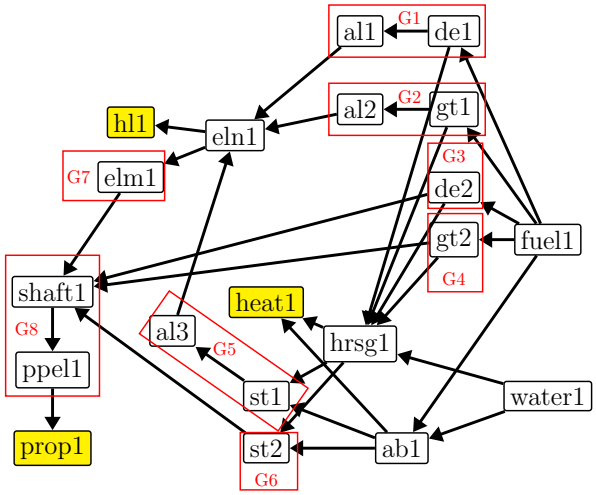

Figure 10: Architecture structure 2

The study about the influence on the maximal occurrence of each component is also performed for this case, see Table 6.

Block name	Max1	Max2	Max3
Propulsion	1	1	1
Hotel load	1	1	1
Heating	1	1	1
Propeller	4	2	4
Shaft	4	2	4
Elec. motor	4	2	4
Elec. network	1	1	2
Gas turbine	4	2	4
Steam turbine	4	2	4
HRSG	1	1	1
AB	1	1	1
Diesel engine	4	2	4
Alternator	4	2	4
Fuel tank	1	1	1
Water tank	1	1	1

Table 6: Maximal occurrence of the block list for Case 2

Apart from the users (heat,p,hl) and the resources (water,fuel), the components ab,hrsg and eln have been set to 1 for simplification. At the end of topology generation, the objective is to evaluate the architecture performance based on its fuel consumption to achieve a user requirement (specified speed, hotel load and heating) and the speed seems the most critical goal so the focus is on the diesel engines and turbines configuration rather than on the steam generation process here.

Based on the structure in Fig. 10, the following system of equations is solved to determine the architecture set :

$$
\begin{aligned}
& N_{Gi} \in [0, \infty],\ i \in [1, 8] \\
& 0 < N_{G8} < \min(N_{\text{ppel}}^{\max}, N_{\text{shaft}}^{\max}) \\
& N_{G6} + N_{G4} + N_{G3} + N_{G7} > 0 \\
& 0 < N_{G1} + N_{G2} + N_{G5} \leq N_{\text{al}}^{\max} \\
& N_{G5} + N_{G6} \leq N_{\text{st}}^{\max} \\
& N_{G7} \leq N_{\text{elm}}^{\max} \\
& N_{G2} + N_{G4} < N_{\text{gt}}^{\max} \\
& N_{G1} + N_{G3} < N_{\text{de}}^{\max} \\
& \text{checkSymmetry2}(N_{G3}, N_{G4}, N_{G7}, N_{G8})
\end{aligned}
\tag{2}
$$

The function *checkSymmetry2* ensure that every driving shaft is connected to the same components by checking $\text{mod}(N_{Gj}, N_{G8}) = 0$ with $j = [3, 4, 6, 7]$. Solving the system (2) leads to the following results in Table 7:

	Max1	Max2	Max3
Number of architectures	11143	3011	420703
Computational time (ms)	440	163	16710

Table 7: Architecture results for Case 2

Compared to the reference case with the Max1 configuration, there are 50 times more architectures in Case 2 but the simulation time only increased by a factor 15 so it demonstrates a good scalability of the algorithm. Three examples are shown in Fig. 11. Architecture 1 represents a mechanical propulsion with two gas turbines on one driving shaft. Architecture 2 is also a mechanical propulsion but with two driving shafts. In this configuration each gas turbine is connected to one shaft only. Finally, Architecture 3 is an electrical propulsion with two gas turbines for the electricity generation and one electric motor for propulsion.

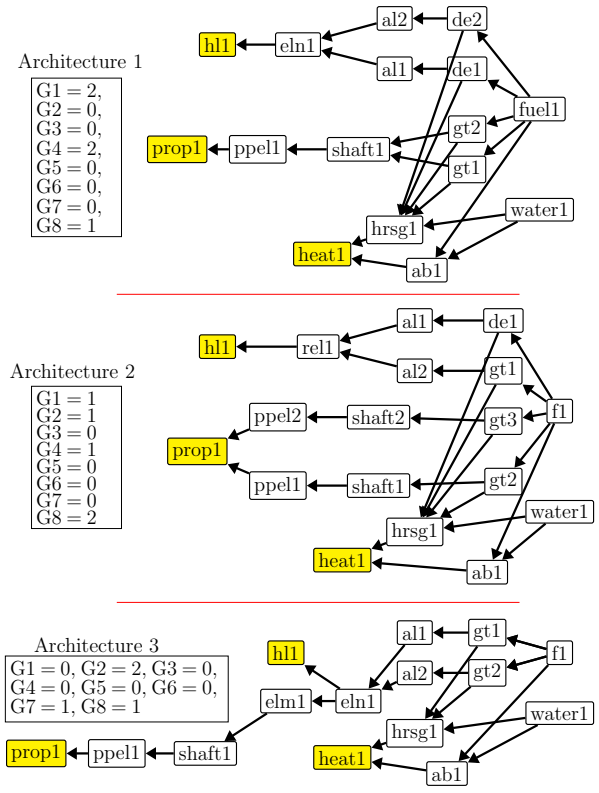

Figure 11: Architecture generation results 2

4 CONCLUSION

An architecture generation algorithm for naval applications has been presented in this article. The presence/absence of components and their connections with each other are determined with some constraints implemented in the logic programming language Prolog. The definition of input/output keywords for each component along with general constraints (prevention of redundant connections, limitation of the maximal number of components) made the algorithm easily adaptable for different configurations. Next, a large number of architectures have been found in a short computational time using constraint logic programming combined with architecture structure construction. Two numerical examples has been provided to illustrate this method, leading to diverse feasible architectures with different properties (mechanical or electrical propulsion, use of diesel engines or gas turbines among others). One perspective of this work is to add components for energy storage (batteries) and for a more detailed description of the mechanical part of the architecture (gearboxes and clutches). Also it is desirable to have more generic keywords for input and output of the components. For example, the keyword "sea" was chosen for the output of the propeller in the reference list in order to get a connection for the block Propulsion but adding custom keywords for specific connections could be difficult to handle for a larger list of components so it will be addressed in future research.

ACKNOWLEDGMENTS

We gratefully thank the Join Laboratory of Marine Technology (J.L.M.T) from Ecole Centrale Nantes and Naval Group for its financial support in this research work.

REFERENCES

[1] G. Doulgeris, T. Korakianitis, P. Pilidis, and E. Tsoudis, "Techno-economic and environmental risk analysis for advanced marine propulsion systems," *Applied Energy*, vol. 99, pp. 1–12, 2012, ISSN: 0306-2619.

[2] A. Batra, S. Sampath, T. Nikolaidis, and P. Pilidis, "Techno-economic model-based design space exploration of combined ship propulsion systems," *Journal of Marine Science and Technology*, Feb. 2023, ISSN: 0948-4280.

[3] E. Silvas, T. Hofman, N. Murgovski, L. F. P. Etman, and M. Steinbuch, "Review of optimization strategies for system-level design in hybrid electric vehicles," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 1, pp. 57–70, 2017.

[4] Z. Qin, Y. Luo, W. Zhuang, Z. Pan, K. Li, and H. Peng, "Simultaneous optimization of topology, control and size for multi-mode hybrid tracked vehicles," *Applied Energy*, vol. 212, pp. 1627–1641, 2018, ISSN: 0306-2619.

[5] O. B. Inal, J.-F. Charpentier, and C. Deniz, "Hybrid power and propulsion systems for ships: Current status and future challenges," *Renewable and Sustainable Energy Reviews*, vol. 156, p. 111 965, 2022, ISSN: 1364-0321.

[6] C. A. Frangopoulos, "Developments, trends, and challenges in optimization of ship energy systems," *Applied Sciences*, vol. 10, no. 13, 2020, ISSN: 2076-3417.

[7] N. L. Trivyza, A. Rentizelas, G. Theotokatos, and E. Boulougouris, "Decision support methods for sustainable ship energy systems: A state-of-the-art review," *Energy*, vol. 239, p. 122 288, 2022, ISSN: 0360-5442.

[8] C. P. Shields, M. J. Sypniewski, and D. J. Singer, "Characterizing general arrangements and distributed system configurations in early-stage ship design," *Ocean Engineering*, vol. 163, pp. 107–114, 2018, ISSN: 0029-8018.

[9] M. A. Parsons, "Network-based naval ship distributed system design and mission effectiveness using dynamic architecture flow optimization," Ph.D. dissertation, Virginia Polytechnic Institute and State University, 2021.

[10] P. de Vos and D. Stapersma, "Automatic topology generation for early design of on-board energy distribution systems," *Ocean Engineering*, vol. 170, pp. 55–73, 2018, ISSN: 0029-8018.

[11] G. N. Sakalis and C. A. Frangopoulos, "Intertemporal optimization of synthesis, design and operation of integrated energy systems of ships: General method and application on a system with diesel main engines," *Applied Energy*, vol. 226, pp. 991–1008, 2018, ISSN: 0306-2619.

[12] G. N. Sakalis, G. J. Tzortzis, and C. A. Frangopoulos, "Intertemporal static and dynamic optimization of synthesis, design, and operation of integrated energy systems of ships," *Energies*, vol. 12, no. 5, 2019, ISSN: 1996-1073.

[13] G. N. Sakalis, G. J. Tzortzis, and C. A. Frangopoulos, "Synthesis, design and operation optimization of a combined cycle integrated energy system including optimization of the seasonal speed of a vlcc," *Proceedings of the Institution of Mechanical Engineers, Part M: Journal of Engineering for the Maritime Environment*, vol. 235, no. 1, pp. 41–67, 2021.

[14] P. Gnes, P. Pinamonti, and M. Reini, "Bi-level optimization of the energy recovery system from internal combustion engines of a cruise ship," *Applied Sciences*, vol. 10, no. 19, 2020, ISSN: 2076-3417.

[15] E. Silvas, T. Hofman, A. Serebrenik, and M. Steinbuch, "Functional and cost-based automatic generator for hybrid vehicles topologies," *IEEE/ASME Transactions on Mechatronics*, vol. 20, no. 4, pp. 1561–1572, 2015.

[16] X. Zhang, H. Peng, J. Sun, and S. Li, "Automated modeling and mode screening for exhaustive search of double-planetary-gear power split hybrid powertrains," Dynamic Systems and Control Conference, vol. 1, 2014.

[17] C. Song, J. Hwang, and D. Kum, "Efficient design space exploration of multi-mode, two-planetary-gear, power-split hybrid electric powertrains via virtual levers," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 4, pp. 3498–3509, 2022.

[18] X. Hu, J. Han, X. Tang, and X. Lin, "Powertrain design and control in electrified vehicles: A critical review," *IEEE Transactions on Transportation Electrification*, vol. 7, no. 3, pp. 1990–2009, 2021.

[19] J. Wijkniet and T. Hofman, "Modified computational design synthesis using simulation-based evaluation and constraint consistency for vehicle powertrain systems," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 9, pp. 8065–8076, 2018.

[20] I. Bratko, *Prolog Programming for Artificial Intelligence* (International Computer Science Series). Addison-Wesley, 2011, ISBN: 9780321417466.

[21] P. Marty, "Étude de l'efficacité énergétique des navires : développement et application d'une méthode d'analyse," Ph.D. dissertation, Ecole Centrale de Nantes (ECN), 2014.