Optimized Selection of Reliable and Cost-Effective Cyber-Physical System Architectures

Nikunj Bajaj*, Pierluigi Nuzzo*, Michael Masin‡, Alberto Sangiovanni-Vincentelli*
* EECS Department, University of California at Berkeley, email: {nikunj.bajaj, nuzzo, alberto}@eecs.berkeley.edu
‡ IBM Haifa Research Lab, Haifa, Israel, email: michaelm@il.ibm.com

***Abstract*—We address the problem of synthesizing safety-critical cyber-physical system architectures to minimize a cost function while guaranteeing the desired reliability. We cast the problem as an integer linear program on a reconfigurable graph which models the architecture. Since generating symbolic probability constraints by exhaustive enumeration of failure cases on all possible graph configurations takes exponential time, we propose two algorithms to decrease the problem complexity, i.e. Integer-Linear Programming Modulo Reliability (ILP-MR) and Integer-Linear Programming with Approximate Reliability (ILP-AR). We compare the two approaches and demonstrate their effectiveness on the design of aircraft electric power system architectures.**

I. Introduction

In a typical cyber-physical system (CPS) architecture, software components running on a hardware computing platform are connected in feedback with physical processes to form a large, distributed, control system subject to tight cost, safety and reliability constraints. A major obstacle to the development of model-based design tools for these systems is the heterogeneity of the requirements, often expressed using different mathematical formalisms, and hard to account for at once. It is, therefore, desirable to devise abstractions that enable scalable co-design and optimization of complex CPS architectures for several, possible conflicting concerns, while guaranteeing design correctness and fault tolerance.

In this paper, we propose an optimization-based methodology for the selection of CPS architectures whose reliability is a function of the interconnection structure. Our goal is to minimize the overall system cost (e.g. number and weight of components) while guaranteeing that an upper bound on system failure probability is met. Our contributions can be summarized as follows:

(i) We provide a general graph representation of an architecture that allows an efficient casting of the design problem as an integer linear program (ILP), capable of modeling a variety of system requirements, such as connectivity, safety, reliability and energy balance;

(ii) We propose two algorithms to decrease the complexity of exhaustively enumerating all failure cases on all possible graph configurations, i.e. Integer-Linear Programming Modulo Reliability (ILP-MR) and Integer-Linear Programming with Approximate Reliability (ILP-AR). ILP-MR *lazily* combines an ILP solver with a background *exact* reliability analysis routine. The solver *iteratively* provides candidate configurations that are analyzed and accordingly modified, only when needed, to satisfy the reliability requirements. Conversely, ILP-AR *eagerly* generates *monolithic* problem instances in polynomial time using *approximate* reliability computations that can still generate estimations to the correct order of magnitude, and with an explicit theoretical bound on the approximation error;

This work was partially supported by IBM and United Technologies Corporation (UTC) via the iCyPhy consortium, and by TerraSwarm, one of six centers of STARnet, a Semiconductor Research Corporation program sponsored by MARCO and DARPA.

(iii) We compare the performance of the two approaches and demonstrate their effectiveness on the design of safety-critical industrial-scale architectures for aircraft electric power distribution.

Traditional safety and reliability assessment is based on a set of methods that are hard to incorporate into automatic design exploration and optimization frameworks. In addition to the complexity of exact network reliability analysis, which is an NP-hard problem [1], techniques such as Fault Tree Analysis (FTA) or Reliability Block Diagrams (RBD) often rely on a set of system abstractions, which are hardly interoperable with system design flows [2]. For instance, in FTA, causal chains leading to some failure are depicted as a tree, inherently describing a hierarchical breakdown. However, in FTA, decomposition into modules mostly relates to the hierarchy of failure influences rather than to the actual system architecture. Therefore, the integration of fault trees with other system design models is not directly possible.

In contrast, we propose to evaluate reliability directly from the system structure, by associating a reliability model to each system component and interconnection, as proposed by Kaiser et al. [2]. This *compositional* approach allows us to formulate two algorithms to *concurrently* optimize for reliability and cost. Instead of formulating a single, "flat" optimization problem, the ILP-MR algorithm avoids the expensive generation of symbolic reliability constraints via an iterative approach inspired by the *ILP Modulo Theory* paradigm [3], [4]. On the other hand, the ILP-AR algorithm aims to efficiently solve a *single optimization* problem, albeit of a larger size, without repetitive calls to the exact reliability analysis function, which may also be expensive. ILP-AR exploits an approximate "algebra" for reliability calculation, inspired by the work of Helle et al. [5]. However, our algebra is *richer* than the previously proposed one, since it accounts for the number of redundant paths implementing a certain function as well as the number of components of the same *type* that are actually used in these paths. As a result, we can relax the simplifying assumption that any used component is either maximally redundant (i.e. participates in just one path) or essential (i.e. participates in all paths).

II. Problem Formulation

We assume that a design is assembled out of a *library* (collection) $\mathcal{L}$ of *components* and *composition rules*. Each component is associated with a set of *attributes*, which are used to capture both its functional and non-functional properties, such as energy, performance, and cost. Components can be connected via *terminals* and terminal *variables*. At this level of abstraction, terminals are *logic* in nature. *Input* terminals are used to receive a signal or the value of a terminal variable; *output* terminals are used to send a signal or assign a value of a terminal variable. Composition rules define how terminals are connected and terminal variables are assigned between components. For the purpose of this paper, $\mathcal{L}$ is parameterized by a set of vectors, including terminal variables $\boldsymbol{w}$, component costs $\boldsymbol{c}$, and failure probabilities $\boldsymbol{p}$. Moreover, we assume that each component can also be labelled with a *type*, defining its functionality (role or task) in a system, as formalized below.

978-3-9815370-4-8/DATE15/©2015 EDAA

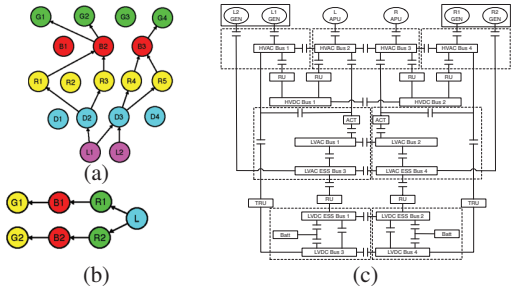


Figure 1: (a) Architecture template configuration example; (b) Architecture analyzed in Example 1; (c) Sample single-line diagram of an aircraft electric power system from [4]: contactors are represented by double bars.

Definition II.1 (Architecture)**.** *A system* architecture *consists of a finite set of* components *and their* interconnections*, and can be modeled as a directed graph $\mathcal{G} = (V, E)$, where each node $v_i \in V$ represents a component and each edge $e_{ij} \in E$ represents the interconnection from v_i to v_j ($i, j \in \{1, \ldots, |V|\}$, $|V|$ being the cardinality of V).*

As shown in Fig. 1a, a *template* $\mathcal{T}$ is an architecture, in which the set of nodes is fixed, while the interconnection structure is variable and can be reconfigured. In a template, the edges can also be represented using a set of Boolean variables $E = \{e_{ij}\}$, each denoting the presence or absence of an interconnection. An assignment over E is a *configuration*. In practice, every node in an abstract architecture can be mapped to an element of a physical architecture; edges can also be conveniently associated with *switches* to denote interconnections that are selectively activated. Based on the attributes of our library, both nodes and edges can be labeled with the terminal variables $\boldsymbol{w}$, costs $\boldsymbol{c}$ and failure probabilities $\boldsymbol{p}$. Finally, we assume that an external control unit can react to component failures, by modifying the link (switch) configuration to activate alternative paths from sources to sinks. The reliability of an architecture is then determined by its topological structure and the redundancy of the paths available to perform a critical function, associated to a *functional link*. To define a functional link, we first introduce a partition on $\mathcal{G}$, to which we link the notion of component type as follows.

Definition II.2 (Graph Partition and Component Type)**.** *A* partition $\Pi = \{\Pi_1, \Pi_2, \ldots, \Pi_n\}$ *over the set of nodes V of $\mathcal{G}$ is a set of nonempty subsets of V such that V is a disjoint union of these subsets. We say that two nodes a and b have the same* type*, written $a \sim b$, when they belong to the same set in Π. If a is in Π_i, then we say that its type is i.*

We also recall that a *walk* $\mu(v_a, v_b)$ of $\mathcal{G}$ is a sequence of nodes $\{n_0, \ldots, n_k\}$ such that $n_0 = v_a$, $n_k = v_b$ and $e_{n_i n_{i+1}} \in E$ for each i. When all nodes in μ are distinct, we say that μ is a (simple) *path*, and write $|\mu|$ to denote the length of μ. Let Π_1 and Π_n be the subsets of V including, respectively, all sources and sinks. Then, a *functional link* F_i is the set of paths from any source in Π_1 to a sink $v_i \in \Pi_n$ that are used to perform an essential system function, on which a reliability requirement is given; in practice, such a function may consist in transferring data or energy from a source to the sink through a sequence of input-output links.

Based on the definitions above, we cast the architecture selection problem as an optimization problem. Given a library $\mathcal{L}$ and a template $\mathcal{T}$, our goal is to derive a configuration that satisfies a set of interconnection and reliability requirements, while minimizing the cost and the complexity (number of components) of the overall network. The set of Boolean variables E will then include our decision variables. Based on the final assignment over E, some of the edges and nodes in $\mathcal{T}$ will be selected to generate an optimal architecture; unnecessary nodes and edges will instead be pruned away to minimize the overall cost. In the following, we provide example formulations for the objective function and the requirements in terms of Boolean arithmetic constraints.

Objective Function. Let $\mathbf{e}$ be the adjacency matrix of $\mathcal{T}$, i.e. $e_{ij} = 1$ if there is one connection from v_i to v_j, and 0 otherwise. Then, the *objective function* can be expressed as the sum of the costs of all components (associated with nodes) and switches (associated with edges) used in the template, i.e.

$$\sum_{i=1}^{|V|} \delta_i c_i + \sum_{i=1}^{|V|} \sum_{j=i+1}^{|V|} (e_{ij} \vee e_{ji}) \tilde{c}_{ij} \tag{1}$$

where c_i is the cost of component i, $\tilde{c}_{ij}$ is the cost of the switch on edge e_{ij}, and δ_i is a binary variable equal to one if the component is instantiated in a configuration and zero otherwise. We express δ_i in terms of the edge variables as $\delta_i = \bigvee_{j=1}^{|V|} (e_{ij} \vee e_{ji})$, meaning that δ_i is one if there exists at least an edge (either ingoing or outgoing) between v_i and any other node in the graph (we assume $e_{ii} = 0$ for all i). Moreover, we use $(e_{ij} \vee e_{ji})$ when computing the cost of the switches, to avoid double counting the contribution of a switch associated to a bidirectional interconnection.

Interconnection Constraints. *Interconnection* requirements originate from the composition rules in $\mathcal{L}$ and are used to enforce legal connections among components. For example, let D, L, and B be subsets of V. Then, we can prescribe that there exists at least (most) one connection from a node in L and a node in D as follows:

$$\sum_{i=1}^{|D|} e_{l_j d_i} \geq (\leq)\ 1 \quad \forall\ j \in \mathbb{N} :\ 1 \leq j \leq |L|, \tag{2}$$

where $e_{l_j d_i}$ is the edge from node l_j to node d_i, and the inequality turns into an equality when one and only one connection is admitted. Moreover, we can state that if there exists an interconnection from any node in L to a node d_j in D, then d_j must be connected to at least one node in B, using a constraint of the form:

$$\bigvee_{1 \leq i \leq |L|} e_{l_i d_j} \leq \bigvee_{1 \leq k \leq |B|} e_{d_j b_k} \quad \forall\ j \in \mathbb{N} :\ 1 \leq j \leq |D|. \tag{3}$$

Another set of interconnection constraints can be used to enforce conservation laws or balance equations in physical systems, e.g. requiring that the maximum power provided by a source in $\mathcal{T}$ is greater than or equal to the maximum power required by the connected sinks. Let d be a node in the graph, which is neither a source nor a sink. Let B be the set of direct predecessors of d, and L be the set of its direct successors. Then, the balance equation at the terminals of d can be written as

$$\sum_{i=1}^{|B|} w_{b_i} e_{b_i d} \geq \sum_{j=1}^{|L|} w_{l_j} e_{d l_j}. \tag{4}$$

Interconnection requirements as the ones above originate linear arithmetic constraints in the decision variables, or include logical operations (conjunctions and disjunctions) that can be linearized with standard techniques [6].

Reliability Constraints. A typical reliability requirement prescribes that the failure probability of a sink, i.e. the probability that a sink gets disconnected from a source because of failures, should be less than a desired threshold. Therefore, to formulate a reliability constraint, we need to compute the probability of composite failure events in the system, starting from the failure probabilities of the components. Specifically, we denote as *system failure* R_i an event in which there is no possible

Algorithm 1 *ILP Modulo Reliability (ILP-MR)*

Input: Architecture template $\mathcal{T}$, component variables $\boldsymbol{w}$, costs $\boldsymbol{c}$ and failure probabilities $\boldsymbol{p}$, reliability requirement r^*
Output: Adjacency matrix $\mathbf{e}^*$ of the final architecture $\mathcal{G}^*$

```
r ← 2r*
(Cost, Cons) ← GenILP(T, w, c)
while r > r* do                                   ▷ failure probability
    e* ← SolveILP(Cost, Cons)
    if e* = [ ] then return UNFEASIBLE
    r ← RelAnalysis(e*, p)
    if r > r* then
        Cons ← LearnCons(Cons, r, r*, e*)
    if Cons = [ ] then return UNFEASIBLE
return e*
```

connection between any of the available sources and a sink i, i.e. when the functional link F_i breaks, and as *reliability level* r_i the probability of R_i. Practically, the above notion of failure models the interruption of any information or energy transfer to an essential portion of the system. We assume that when a component fails it cannot be recovered, and the adjacent links are no longer usable. Moreover, failures in different components are independent.

Let P_i be the event that component i fails (self-induced failure). Then, the event R_i of a system failure affecting component i can be recursively computed as follows

$$R_i = P_i \cup \bigcap_{1 \le j \le |V|, e_{ji} \neq 0} R_j, \tag{5}$$

where e_{ji} is j^{th}-row, i^{th}-column element of the adjacency matrix $\mathbf{e}$ of $\mathcal{T}$. In other words, component i fails when either a failure is generated in itself, or when failures are induced through its predecessors. A symbolic constraint for r_i can then be generated using (5) to enumerate all possible failure events while traversing $\mathcal{T}$ from node i to the sources. However, such an exact computation, based on the enumeration of all possible component failure events, has exponential complexity on a fixed graph configuration [1]. The problem is further exacerbated when compiling a symbolic expression for a reconfigurable graph, since, in general, enumerating all possible configurations has also exponential complexity. To overcome this issue, we propose two approaches to the solution of the optimal architecture selection problem, namely, ILP-MR and ILP-AR, which we detail in the following sections.

III. ILP Modulo Reliability

The ILP Modulo Reliability (ILP-MR) algorithm avoids the expensive generation of symbolic reliability constraints by adapting the ILP Modulo Theory approach [3] to reliability computations, as summarized in Algorithm 1. ILP-MR receives as inputs: the library of components $\mathcal{L}$, together with their attributes $\boldsymbol{c}$, $\boldsymbol{w}$ and $\boldsymbol{p}$, the template $\mathcal{T}$, and the set of requirements, including interconnection and reliability constraints. To simplify, we assume that r is the worst case failure probability over a set of nodes of interest, for which the same reliability requirement r^* must be satisfied.

An ILP is solved in a loop with a reliability analysis routine. SolveILP generates minimum cost architectures for the given set of interconnection constraints. The RelAnalysis routine computes the probability of composite failure events at critical nodes, starting from the failure probabilities of the components, a problem known as K-terminal reliability problem in the literature [1]. To do so, we implement a modified depth-first search algorithm to traverse the graph $\mathcal{G}$ from node i (root) to the source nodes (leaves), by applying a path enumeration method, and by turning event relations as

Algorithm 2 LearnCons

Input: Current constraints $Cons$, reliability r, reliability requirement r^*, adjacency matrix $\mathbf{e}^*$
Output: Final constraints $Cons$

```
k ← EstPath(r, r*, e*)
NewCons ← [ ]
S ← GetSinks(e*)
(T_1, ..., T_n) ← GetTypes(e*)
for all v ∈ S do
    if k ≥ 1 then
        for all i ∈ (T_{n-1}, T_{n-2}, ..., T_1) do
            NewCons ← AddPath(v, i, k, NewCons, e*)
    else
        i ← FindMinRedType(v, e*)
        NewCons ← AddPath(v, i, 1, NewCons, e*)
    if NewCons = [ ] then return UNFEASIBLE
Cons ← (Cons, NewCons)
return Cons
```

in (5) into probability expressions. However, any other exact reliability analysis method for directed graphs can also be used [1]. Although the K-terminal reliability problem is NP-hard, the key idea is to solve it only when needed, i.e. a small number of times, and possibly on smaller graph instances.

At each iteration of ILP-MR, if the optimal architecture satisfies the reliability constraints, it is returned as the final solution. Otherwise, LearnCons estimates the number of redundant paths needed to achieve the desired reliability and suggests a set of strategies to implement the required paths by augmenting the original optimization problem with a set of interconnection constraints. This constraint learning function is, therefore, instrumental to efficiently converge towards a reliable architecture, while minimizing the number of calls to RelAnalysis. We provide details about this function in the following section.

A. Learning Constraints to Improve Reliability

When no reliability constraints are enforced in the ILP, the solver attempts to use the minimum number of components and interconnections to perform a specific function at minimum cost. Typically, such a "minimal" architecture has also minimal redundancy, hence minimal reliability. Based on this intuition, we develop strategies that increase the reliability of the solution, albeit at a higher cost, by enforcing a larger number of redundant components and interconnections. We have recently reported an iterative approach similar to ILP-MR, based on a set of *ad hoc* strategies, customized for the specific problem at hand [4], [7]. In Algorithm 2, we introduce instead a generic template that can directly apply to the general problem formulation in Section II, hence to a broader set of applications.

Based on the current reliability level r, LearnCons estimates the number of additional redundant paths k required to satisfy the desired reliability r^* (function EstPath). As an example, under the assumption that all the paths in a functional link F are independent, then k can be computed as $k = \lfloor \log(r^*/r) / \log \rho \rfloor$, where ρ is the failure probability of a single path in F, and $\lfloor x \rfloor$ denotes the integer part of x. Since, in reality, the paths in F are not independent, this is a conservative estimation which avoids over-design. Then, if at least one additional path is required, for all the sinks and component types implementing F and used in the current architecture, AddPath generates new constraints to enforce that at least k additional components of each type have a path to the sink. These constraints do not necessarily translate into instantiating more components, as far as additional paths

to the sink can be obtained by just increasing the number of interconnections. If k additional paths cannot be obtained with the current template, ADDPATH attempts to enforce the maximum available number of paths. Conversely, if the estimated number of paths is zero, LEARNCONS attempts to still improve the overall reliability by enforcing one additional path between the sink and a component whose type has minimum redundancy in the current architecture, i.e. for which the total number of paths to the sink is minimum (as obtained from FINDMINREDTYPE). If no additional paths can be added between a sink and a component of any type, LEARNCONS terminates with `UNFEASIBLE`.

To enforce additional paths, ADDPATH uses the walk indicator matrix of $\mathcal{T}$, defined below.

Lemma 1 (Walk Indicator Matrix)**.** *Let* $\mathbf{e}$ *be the adjacency matrix of a graph* $\mathcal{T}$*; let* $\mathbf{a} \odot \mathbf{b}$ *the* logical product *of two logical matrices* $\mathbf{a}$ *and* $\mathbf{b}$ *in* $\mathbb{B}^{m\times m}$*, defined as* $(\mathbf{a} \odot \mathbf{b})_{ij} = \bigvee_{k=1}^{m} a_{ik} \wedge b_{kj}$*; let* $\mathbf{e}^k = \underbrace{\mathbf{e} \odot \ldots \odot \mathbf{e}}_{k \text{ times}}$ *be the* k*-th* logical power *of* $\mathbf{e}$*. Then, the entry in row* i *and column* j*,* η_{nij}*, of the* walk indicator *matrix* $\boldsymbol{\eta}_n = \bigvee_{k=1}^{n} \mathbf{e}^k$ *is* 1 *if and only if there exists a directed walk of length less or equal* n *from vertex* v_i *to vertex* v_j*.*

We can then require at least k additional connections of components of type T_i, belonging to the set Π_i of the partition Π of $\mathcal{T}$, to a sink v via at least one path of length $n-i+1$ by enforcing

$$\sum_{w\in\Pi_i} \eta_{n-i+1\,w,v} \geq k + \sum_{w\in\Pi_i} \eta^*_{n-i+1\,w,v}, \tag{6}$$

where η_{n-i+1} and η^*_{n-i+1} are the walk indicator matrices, respectively, for $\mathcal{T}$ (decision variables) and the current architecture $\mathcal{G}^*$. The constraint (6) can be converted into an equivalent set of linear constraints in the elements of $\mathbf{e}$ (edge variables) by using standard linearization techniques. The following result summarizes the properties of the ILP-MR approach.

Theorem 1 (Correctness of ILP-MR)**.** *For a given template* $\mathcal{T}$ *and within the approximation error* ϵ *of the* SOLVEILP *and* RELANALYSIS *routines, ILP-MR (Algorithm 1) is sound and complete.*

For the sake of conciseness, the formal proof of the theorems in this paper will be omitted. Informally, we observe that, since the number of components in $\mathcal{T}$ is finite, the ILP-MR routine, based on Algorithms 1 and 2, will terminate. Moreover, because RELANALYSIS implements an exact reliability analysis method, if a final architecture is found, then it will satisfy all the requirements, being only subject to the rounding error ϵ due to the ILP solver and to RELANALYSIS. On the other hand, because all available components will be eventually used to increase the reliability, if ILP-MR terminates with `UNFEASIBLE`, then there is no architecture, obtained from the given template, which is able to satisfy all the constraints.

IV. ILP WITH APPROXIMATE RELIABILITY

The ILP-AR algorithm replaces exact reliability computations with an approximate algebra that allows encoding a reliability requirement into a number of linear constraints on Boolean variables, which is polynomial in the size of the template. In Section IV-A we introduce the approximate reliability algebra, which estimates the correct order of magnitude for the failure probabilities of all the components in an architecture by leveraging the fact that components with the highest failure probability tend to dominate the overall failure probability. Then, in Section IV-B we report the main results on correctness and complexity of ILP-AR.

A. Approximate Reliability Algebra

In the approximate algebra, components contribute to the system failure probability based on their *degree of redundancy*, which is defined based on the notions of functional link and component type defined in Section II. We recall that any path in a functional link F_i consists of an interconnection of components, each having a role or performing a sub-task defined by its type. Components of the same type can be interchanged and introduce redundancy in the system architecture. We say that a component type j, associated to a partition Π of $\mathcal{G}$, *jointly implements* a functional link F_i, written $\Pi_j \vdash F_i$, if all paths in the functional link F_i include at least one node in Π_j, i.e. $\Pi_j \vdash F_i$ iff $\forall \mu \in F_i : \mu \cap \Pi_j \neq \emptyset$. Trivially, we have $\Pi_1 \vdash F_i$ and $\Pi_n \vdash F_i$ for all i. Moreover, multiple nodes of the same type are allowed in a path as far as they are adjacent to each other. Given a path μ, possibly including multiple instances of the same type, we denote as $\hat{\mu}$ the *reduced path* obtained from μ after replacing all the instances of the same type with a single node, still of the same type.

We can then define the degree of redundancy h_{ij} associated with type j and link F_i as the number of components of type j used in at least one reduced path of F_i, i.e. $h_{ij} = |(\cup_{\mu\in F_i} \hat{\mu}) \cap \Pi_j|$. Finally, we approximate the failure probability r_i of a functional link F_i by

$$\tilde{r}_i = \sum_{j\in I_i} h_{ij} p_j^{h_{ij}} \tag{7}$$

where $I_i = \{j | \Pi_j \vdash F_i\}$ is the set of all component types that jointly implement F_i, p_j is the probability of failure of any of the components of type j, and h_{ij} is the degree of redundancy associated with type j and link F_i.

Example 1. *We illustrate the application of* (7) *to the architecture* $\mathcal{E}$ *represented in Fig. 1b. We consider a partition where* $\Pi_1 = \{G_1, G_2\}$*,* $\Pi_2 = \{B_1, B_2\}$*,* $\Pi_3 = \{D_1, D_2\}$ *and* $\Pi_4 = \{L\}$*. Sink* L *is connected to sources* G_1 *and* G_2 *via two (reduced) paths, each using components of all the four types listed above. Then, the approximate expression for the failure probability of* L *would be* $\tilde{r}_L = p_L + 2p_D^2 + 2p_B^2 + 2p_G^2$*, while exact calculations lead to*

$$r_L = p_L + (1-p_L)\{p_D + (1-p_D)[p_B + (1-p_B)p_G]\}^2.$$

When all components are assumed to fail with the same probability $p \ll 1$*, we obtain* $\tilde{r}_L = p + 6p^2$ *and* $r_L = p + 9p^2 + O(p^3)$*.*

The estimation in Example 1 has the same order of the exact calculation, and the error becomes negligible for small p. In general, it is possible to state the following theorem, providing a bound on the error of the approximate reliability algebra.

Theorem 2. *Given a graph* $\mathcal{G}$ *and a partition* Π*, let* $\tilde{r}$ *and* r *be, respectively, the approximate and exact failure probability for a functional link* $F = \{\mu_1, \ldots, \mu_f\}$*, denoted by its set of* f *paths* ($f = |F|$)*. Let* $I = \{j | \Pi_j \vdash F\}$ *the set of component types jointly implementing* F*. Then, the following inequality holds:*

$$\frac{\tilde{r}}{r} \geq \frac{mf}{M_f}, \tag{8}$$

where $M_f = \prod_{j=1}^{j=f} |\mu_j|$ *and* $m = |I|$*.*

Based on Theorem 2, (7) can provide "optimistic" estimations, but the bound to such optimism can be explicitly estimated for a given graph template $\mathcal{T}$.

B. Correctness and Complexity of ILP-AR

The overall ILP-AR approach is illustrated in Algorithm 3. To implement GENILP-AR, we use the approximate algebra to

Algorithm 3 *ILP With Approximate Reliability (ILP-AR)*

Input: Architecture template $\mathcal{T}$, component variables $\boldsymbol{w}$, costs $\boldsymbol{c}$ and failure probabilities $\boldsymbol{p}$, reliability requirement r^*
Output: Adjacency matrix $\mathbf{e}^*$ of the final architecture $\mathcal{G}^*$

$(Cost, Cons) \leftarrow$ GENILP-AR($\mathcal{T}$, $\boldsymbol{w}$, $\boldsymbol{c}$, $\boldsymbol{p}$, r^*)
$\mathbf{e}^* \leftarrow$ SOLVEILP($Cost$, $Cons$)
if $\mathbf{e}^* = [\,]$ **then return** UNFEASIBLE
return $\mathbf{e}^*$

Table I: Components and attributes used in the EPS example.

Generators	g (kW)	Loads	l (kW)	Components	c
LG1	70	LL1	30	Generator	g/10
LG2	50	LL2	10	Bus	2000
RG1	80	RL1	10	Rectifier	2000
RG2	30	RL2	20	Contactor	1000
APU	100				

capture all the reliability requirements. While (7) is a nonlinear expression, a linear encoding of the same constraint can be obtained as follows

$$\sum_{k\in\{1,\ldots,k_{max}\},j\in\{1,\ldots,n\}} k \cdot x_{ijk} \cdot p_j^k \leq r_i^*, \tag{9}$$

where r_i^* is the required failure probability, x_{ijk} is an auxiliary binary variable equal to 1 if $j \in I_i$ and $h_{ij} = k$, and 0 otherwise, and k_{max} is the maximum possible value for h_{ij} in the given template, i.e. $k_{max} = \max_{1\leq j\leq n} |\Pi_j|$.

Additional constraints are needed to express the auxiliary variables x_{ijk} in terms of our decision variables. We assume that the reference template $\mathcal{T}$ only includes reduced paths. This is not a restrictive assumption, since multiple instances of adjacent nodes of the same type can be added by refining $\mathcal{T}$ in a second step of the selection process. Then, by lemma 1, we link the indicator variables x_{ijk} to the decision variables by adding the following constraints for each type j in $\{1, \ldots, n\}$:

$$\sum_{k=0}^{k_{max}} x_{ijk} \leq 1, \tag{10}$$

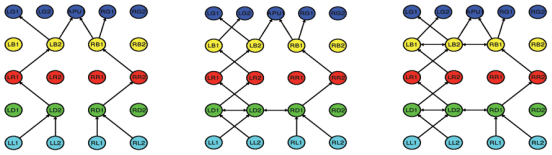
and, $\forall k \in \mathbb{N} : 0 \leq k \leq k_{max}$,

$$\sum_{w\in\Pi_j} \left(\eta_{n_w,v_i} \wedge \left(\bigvee_{s\in\Pi_1} \eta_{n_s,w} \right) \right) = k \rightarrow (x_{ijk} = 1). \tag{11}$$

The constraint in (11) counts the number of components of type j which are connected by at least one path to v_i and to any source in Π_1. The indicator variable x_{ijk} is then set to 1 if the number of such components is k. Constraint (10) enforces that only one of the x_{ijk} variables be set to one. The implication in (11) can be easily converted into a linear constraint using standard techniques [6]. Overall, it can be shown that the number of constraints (and auxiliary variables) generated by the computations in (9)-(11) is $O(|V|^3 n)$, where $n = |\Pi|$. This amounts to a polynomial complexity in the number of nodes and partitions in $\mathcal{T}$, which contrasts with the exponential complexity of the exact computations in Section II and III. Finally, the following result holds for the ILP-AR approach.

Theorem 3 (Correctness of ILP-AR)**.** *For a given template $\mathcal{T}$ and within the error bound provided in Theorem 2, ILP-AR (Algorithm 3) is sound and complete.*

Informally, the result follows from the fact that, for each type of components, ILP-AR attempts to determine the degree of redundancy needed to meet the reliability requirement.



(a) Architecture 1 (b) Architecture 2 (c) Architecture 3

Figure 2: EPS architectures and reliability as obtained at each iteration of an ILP-MR run with $r^* = 2 \times 10^{-10}$: (a) $r = 6 \times 10^{-4}$; (b) $r = 2.8 \times 10^{-10}$; (c) $r = 0.79 \times 10^{-10}$.

Therefore, if ILP-AR returns UNFEASIBLE, assuming that the interconnection constraints are feasible, then we can conclude that $\mathcal{T}$ does not provide enough redundancy to satisfy the reliability constraints. On the other hand, when ILP-AR provides an optimal topology, the solution will satisfy the reliability requirement with an approximation error which is worst case bounded by (8).

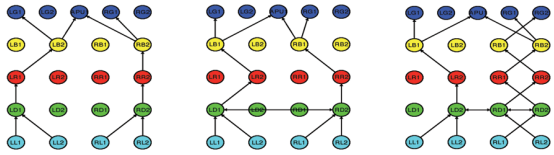
V. AIRCRAFT POWER SYSTEM ARCHITECTURE DESIGN

We apply our algorithms to the selection of optimal architectures for power generation and distribution in a passenger aircraft. Fig. 1c illustrates a sample architecture in the form of a single-line diagram, a simplified notation for three-phase power systems [4]. Typically, aircraft Electric Power System (EPS) components include power sources, such as the left and right generators (L/R-GEN) and the auxiliary power units (APU) in Fig. 1c. The generators power the buses and their loads (not shown in Fig. 1c). AC power is converted to DC power by rectifier units (TRU). A bus power control unit monitors the availability of power sources and configures a set of switches, denoted as contactors, such that essential buses remain powered even in the presence of failures.

We aim to generate an EPS architecture that satisfies a set of connectivity, power flow and reliability requirements, while minimizing the total cost. We then model the architecture as a directed graph, where each node represents a component (with the exception of contactors, which are associated with edges) and each edge represents an interconnection. We assume a template $\mathcal{T}$ consisting of the following component types: generators (LG/RG), AC buses (LB/RB), rectifiers (LR/RR), DC buses (LD/RD), loads (LL/RL), two on each side, and one APU. The platform library attributes include generator power ratings $\boldsymbol{g}$, load power requirements $\boldsymbol{l}$, component costs $\boldsymbol{c}$ and failure probabilities $\boldsymbol{p}$, as summarized in Table I. In our examples, we assume that only generators, buses and rectifiers fail with a probability of 2×10^{-4}.

Connectivity properties can be expressed by using constraints as the ones in (2) and (3). For instance, we can prescribe that any rectifier must be directly connected to only one AC bus, and that all DC buses that are connected to a load or another DC bus must be connected to at least one rectifier to receive power from an AC bus. Power-flow constraints are used to enforce that the total power provided by the generators in each operating condition is greater than or equal to the total power required by the connected loads, by using expressions as in (4). Finally, a reliability constraint prescribes that the probability that a load gets unpowered because of failures should be less than a desired threshold. A functional link will then consist of the set of paths from any generator to the load. Moreover, since our template supports only reduced paths, we use an edge between two nodes of the same type as a shorthand notation to indicate two redundant components: if v_i and v_j, with $v_i \sim v_j$, are connected by an edge, then any direct predecessor of v_i is also a direct predecessor of v_j and vice versa.

We have developed ARCHEX, a prototype framework for system architecture exploration and synthesis, implementing both the ILP-MR and ILP-AR algorithms. ARCHEX leverages



(a) Architecture 1 (b) Architecture 2 (c) Architecture 3

Figure 3: EPS architectures synthesized using ILP-AR for different reliability requirements: (a) $r^* = 2 \times 10^{-3}, \tilde{r} = 6.0 \times 10^{-4}, r = 6 \times 10^{-4}$; (b) $r^* = 2 \times 10^{-6}, \tilde{r} = 2.4 \times 10^{-7}, r = 3.5 \times 10^{-7}$; (c) $r^* = 2 \times 10^{-10}, \tilde{r} = 7.2 \times 10^{-11}, r = 2.8 \times 10^{-10}$.

YALMIP [8] and CPLEX [9] to, respectively, formulate and solve ILP problems. All the numerical experiments were performed on an Intel Core i7 2.8-GHz processor with 8-GB memory. Figure 2 shows the architectures obtained at each iteration of the ILP-MR algorithm for a load failure probability requirement $r^* = 2 \times 10^{-10}$. By solving for just the connectivity and power flow constraints, we obtain the simplest possible architecture (Fig. 2a), which only provides a single path from a load to a generator (or APU), thus showing the highest failure probability. Based on the parameters in Table I we obtain $\rho = 8 \times 10^{-4}$, which leads to $k = 2$, as discussed in Section III-A. Therefore, at the second iteration, two additional paths are enforced between each load and a generator, as shown in Fig. 2b. Since the requirement is not yet satisfied, a third iteration is used to fine tune the reliability, by adding one more path between each load and an AC bus. The total computation time to generate the architectures in Fig. 2 was about 38 s. Three architectures obtained using the ILP-AR algorithm for different load failure probability requirements are instead shown in Fig. 3. The lower the required failure probability, the higher the number of redundant paths and components instantiated from the original template, and the higher the associated cost. For each architecture, the approximate algebra provides an estimation $\tilde{r}$ of the failure probability which is extremely close to the actual value r obtained by exact computations. While the failure probability of the architecture in Fig. 3c exceeds the requirement, the error is well within the bound predicted by Theorem 2. The execution time of each optimization run in Fig. 3 was also approximately 38 s; however, about 70% of the computation time was used to generate the optimization constraints, which can also be performed off-line for a given template.

To test the scalability of both the approaches, we designed EPS architectures with an increasing number of components. In Table II, we report on the execution time of the ILP-MR approach using Algorithm 2 (at the top) in comparison with the one obtained by a lazier approach, which only adds one additional path at each iteration between the load and a component with a minimal degree of redundancy. The dramatic reduction in time spent for reliability analysis (e.g., more than one day versus 3 min for a 50-node architecture) shows the advantage of using the analysis results to infer the number of required redundant paths, as proposed in Algorithm 2. When this inference is feasible, ILP-MR outperforms ILP-AR (see solver times in Table III) for architectures with more than 40 nodes. On the other hand, as evident from Table III, once the optimization problem is generated for a given template, ILP-AR is more competitive for smaller architectures. Yet, problems with several thousands of constraints, and including a realistic number of generators (normally less than 10), can still be formulated and solved in a few hours. We also observe that, because of the sparsity of the EPS adjacency matrix, in this case study, it was possible to reduce the number of generated constraints, which is always smaller than the asymptotic estimation in Section IV.

Overall, we infer that ILP-AR turns out to be preferable when we aim to a coarser estimation of the capability (and limitations) of an architecture template and a platform library in terms of reliability. On the other hand, ILP-MR makes it easier to incorporate domain-specific knowledge, since a designer can customize the techniques adopted to improve reliability at each iteration. Moreover, ILP-MR becomes the preferred choice, especially for larger problem instances, when we can estimate the number of redundant paths needed to satisfy the requirement as early as possible, or when we are willing to pay for a longer execution time to incrementally fine tune the reliability of the design.

Table II: Number of iterations, reliability analysis and solver time for different EPS architecture sizes ($r^* = 10^{-11}$, $n = 5$) using ILP-MR with LEARNCONS (top) and with a lazier strategy, adding only one path at each iteration (bottom).

$\|V\|$ (# Generators)	#Iterations	Analysis time (s)	Solver time (s)
20 (4)	3	34	4.3
30 (6)	3	78	9
40 (8)	3	106	14
50 (10)	3	181	18
20 (4)	4	72	13
30 (6)	7	852	28
40 (8)	10	9118	58
50 (10)	14	39563	114

Table III: Number of constraints, problem generation (setup) and solver time for different EPS architecture sizes ($r^* = 10^{-11}$, $n = 5$) using ILP-AR.

$\|V\|$ (# Generators)	# Constraints	Setup time (s)	Solver time (s)
20 (4)	5290	27	11
30 (6)	24514	402	77
40 (8)	74258	3341	494
50 (10)	176794	18902	5059

VI. CONCLUSIONS

We have introduced, characterized and demonstrated two efficient ILP-based algorithms for the optimal selection of cyber-physical system architectures subject to safety and reliability constraints. As a future work, we plan to further investigate the generalization of the presented approaches to support a broader category of systems (e.g. power grids, communication networks) and design concerns (e.g. impact of system dynamics and transients).

Acknowledgments. The authors wish to acknowledge Safa Messaoud and Mohammad Mozumdar for their contributions in the development of the design example.

REFERENCES

[1] C. Lucet and J.-F. Manouvrier, "Exact methods to compute network reliability," in *Proc. Int. Conf. on Mathematical Methods in Reliability*, 1997.

[2] B. Kaiser, P. Liggesmeyer, and O. Mäckel, "A new component concept for fault trees," in *Proc. Australian Workshop on Safety Critical Systems and Software*, 2003.

[3] C. Hang, P. Manolios, and V. Papavasileiou, "Synthesizing cyber-physical architectural models with real-time constraints," in *Proc. Int. Conf. Comput.-Aided Verification*, Dec. 2011.

[4] P. Nuzzo, H. Xu, N. Ozay, J. Finn, A. Sangiovanni-Vincentelli, R. Murray, A. Donze, and S. Seshia, "A contract-based methodology for aircraft electric power system design," *IEEE Access*, vol. 2, pp. 1–25, 2014.

[5] P. Helle, M. Masin, and L. Greenberg, "Approximate reliability algebra for architecture optimization," in *Proc. Int. Conf. on Computer Safety, Reliability, and Security*, 2012, pp. 279–290.

[6] W. L. Winston, *Operations Research: Applications and Algorithms, 4th Edition*. Independence, KY: Cengage Learning, 2004.

[7] S. Messaoud, "Optimal Architecture Synthesis for Aircraft Electrical Power Systems," Master's thesis, T.U. Munich – U.C. Berkeley, 2013.

[8] J. Löfberg, "YALMIP: A toolbox for modeling and optimization in MATLAB," in *Int. Symp. Computer Aided Control Systems Design*, 2004, pp. 284–289.

[9] (2012, Feb.) IBM ILOG CPLEX Optimizer. [Online]. Available: www.ibm.com/software/integration/optimization/cplex-optimizer/