Safety-Level Aware Bin-Packing Heuristic for Automatic Assignment of Power Plants Control Functions

Mohamed Benazouz, Jean-Marc Faure

Safety-Level Aware Bin-Packing Heuristic for Automatic Assignment of Power Plants Control Functions

Mohamed BENAZOUZ mohamed.benazouz@gmail.com
Jean-Marc FAURE jean-marc.faure@ens-cachan.fr
LURPA, ENS Cachan, France

***Abstract*—Finding a suitable set of controllers to which a large set of control functions with different safety levels can be assigned, while minimizing cost, is a significant task during the design of the operational control system of a critical process, like a power plant. This task is currently performed by experts and extremely time-consuming, which explains why its automation is a real concern. This paper shows first that the above assignment problem can be identified as a Multiple-Choice Vector Bin-Packing with Conflicts problem, a combination of different variants of the well-known one-dimensional bin-packing problem. Such a problem is known to be strongly NP-Hard and exact techniques to solve it on large-sized examples are too time and/or space consuming because of the combinatorial explosion. To solve this problem in polynomial time, this article proposes a fast heuristic based on a FFD (First-Fit Decreasing) approach. Two strategies to perform this heuristic and several criteria to rank the functions before assignment are defined. These strategies and criteria are then compared on the basis of numerous experiments. These experiments show that the proposed heuristic scales well and provides results that are very close to optimum; the difference in the worst case is less than 1%.**

Note to Practitioners: ***Abstract*—Designing the operational control architecture of a critical process, like a power plant, is an extremely time-consuming task that requires in particular to find a suitable and minimum-cost set of industrial controllers to which the numerous control functions can be assigned and thereafter implemented. This design task is currently performed by experts who imagine, evaluate and compare different solutions. Hence, several iterations of the assignment process are necessary during this task. The practical aim of this research is to automate this process to facilitate and speed up architecture design. This work has been made in the frame of a cooperative research project with a company which designs and implements control systems of power plants. Nevertheless, the generic results that are presented in this article can be used for other critical processes (oil, chemical processes, water management) where a large number of control functions with different impacts on safety is to be assigned to controllers, because the proposed heuristic scales well.**

***Index Terms*—Dependable control, distributed systems, critical systems, vector bin-packing, multiple-choice**

I. Introduction

FROM a functional point of view, the control system of a power plant is composed of several thousands of functions (actuators control, processes synchronization, variables monitoring, etc.) that are aiming at ensuring the correct operation of the electrical power production process as well as safety of people and environment in case of failure of a component of the plant. This set of functions is defined in the earliest phases of design and is the starting point of this work. The internal structure of every function, in the form of ECC (Execution Control Chart) [1] or SFC (Sequential Function Chart) [2] for instance, is unknown at these phases and in what follows. A control function will be seen as a black-box and only the requirements to implement the function, like its number of input/output data and its CPU consumption, will be considered. Moreover, a power plant is a critical system and some functions are more important for safety than other ones. Hence, a safety level, letter that represents the importance of the function with regard to safety, is associated to every function and the whole set of functions is divided into several subsets, termed classes. All the functions of a given class own the same safety level. Last, among every class, some functions are redundant, *i.e.* they may perform the same service in different operation modes. When an active function is no more able to perform its service because the plant components that it controls have failed, it is replaced by a redundant one that ensures the same service with other faultless plant components.

On the other side, from an operational point of view, the control system is composed of a set of some hundreds of industrial controllers, processing devices (Programmable Logic Controllers, industrial computers) connected to sensors and actuators and that execute the control functions. The internal structure of an industrial controller is not considered in this paper, only its capacities (overall number of input and output interfaces, processing capacity) are known. Moreover, it matters to underline that different categories of controllers, with different abilities to be fault-tolerant to internal failures, are available to implement the control functions. The ability of a controller to be fault-tolerant is represented by an integer which is termed its criticality factor. The smaller this factor, the more fault-tolerant the controller is. It is obvious that the cost of the controller increases when its criticality factor decreases.

Before implementing every function on a given controller, an assignment process must be performed. The first aim of the assignment process (Fig. 1) is to find a suitable set of controllers such that every function is assigned to one and only one controller while satisfying two kinds of constraints: capacity constraints and safety constraints. Capacity constraints mean merely that several functions can be assigned to a given controller if and only if the capacities of this controller are large enough to host all these functions. It is obviously not possible, for instance, to assign three functions with M/2 input

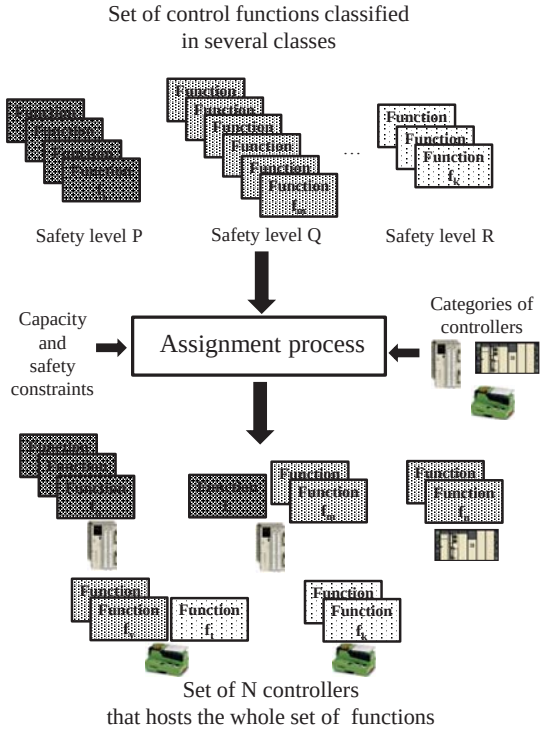

Fig. 1. Overview of the assignment process.

data each to a controller that owns only M input interfaces. Two kinds of safety constraints must be introduced in addition. The first one relates the safety level of a function to the criticality factor of the controller to which this function may be assigned. It is not allowed for instance to assign a function which strongly impacts safety to a controller which is not very fault-tolerant. The second safety constraint, termed separation constraint, means that two redundant functions must not be assigned to the same controller, to avoid that the failure of this controller provokes the loss of the service. A formal definition of the assignment problem will be given in the next section.

Assuming that there always exists a controller which can host the larger function (otherwise this function must be decomposed in smaller ones), a trivial assignment solution is to choose as many controllers as there are functions and to define the criticality factor of every controller according to the safety level of the only function it hosts. The separation constraint will be automatically satisfied because only one function will be assigned to every controller. This solution is obviously not economically viable[1]. Hence, the overall objective of the assignment process is not only to find a feasible solution that satisfies all constraints, as presented in [3], but to minimize the cost of the set of controllers of this solution.

Currently, the assignment process is performed by experts in companies that design and implement large control systems. This process is extremely tedious and time consuming. Moreover, minimization of the cost is generally achieved only by comparing several feasible solutions. This work aims to tackle out these issues by proposing a bin-packing heuristic to assign automatically the control functions, while minimizing the cost, in a short time. Preliminary results have been presented in [4]. This article extends these results by proposing another novel assignment strategy and comparing the two strategies on the basis of a larger set of experiments.

The outline of the paper is the following. An ILP formulation of the assignment problem is given in the next section. It is shown in the third section that this problem is a Multiple-choice Vector Bin-Packing (MVBP) with Conflicts problem. Two strategies are then proposed in the fourth section to assign the different classes of functions. For both strategies, a heuristic based on a FFD (First Fit Decreasing) approach is detailed in section V. Several criteria to sort the functions before applying the FFD are also defined in this section. The relative performances of the two strategies and the sorting criteria are compared and discussed on the basis of experimental results in the sixth section, while concluding remarks and outlooks are drawn up in the seventh one.

II. Problem Formulation

In this section, the problem adressed in this paper is modelled by an ILP. For the sake of comprehension and without impacting generality, the formulation is limited to the 2-dimensional case (only 2 capacities will be considered). The ILP formulation can be unlimitedly extended to multiple capacities given that they follow a linear additive consumption scheme.

Capacities considered are cpu processing capacity and the number of inputs (*resp.* outputs) from (*resp.* to) the physical processes of the plant that are required for a control function to perform its task. Inputs and outputs are considered interchangeable as controllers dispose of cards that can be adapted following requirements. Therefore, the number of inputs and outputs required by a function are summed and this sum is noted inout.

Let us consider a set of functions $\mathbb{F}$ and a set of controllers $\mathbb{C}$. Each function $f_i \in \mathbb{F}$ is characterized by its

- $inout_i$: the number of its inputs and outputs,
- cpu_i : its cpu consumption, and
- sl_i : its safety level.

Four safety levels are defined in IEC standards [5] for power plants control functions: A, B, C and NC. A function f_i with $sl_i = A$ (*resp.* B,C,NC) is said to be of class A (*resp.* B,C,NC). A function whose safety level is A impacts safety more than a function of class B which itself is more significant for safety than a function of class C. NC functions (not classified) do not impact safety.

On the other hand, each controller $C_j \in \mathbb{C}$ is characterized by its

- $INOUT_j$: its capacity in terms of inputs/outputs,
- CPU_j : its processing capacity,
- CF_j : its criticality factor with $CF_j \in \{0, 1, 2\}$,

[1]The observation of existing solutions in power plants shows that a controller hosts on average several tens of functions.

- $COST_j$: its cost.

A controller C_j with $CF_j = 0$ (*resp.* 1,2) is said to be of category 0 (*resp.* 1,2). Controllers with smaller criticality factor are more reliable. The cost of a controller is proportional to its characteristics. Since in the addressed problem all controllers are considered to have the same processing capacity CPU, *i.e.*

$$\forall C_j \in \mathbb{C}, CPU_j = CPU,$$

and the same capacity of inputs/outputs $INOUT$, *i.e.*

$$\forall C_j \in \mathbb{C}, INOUT_j = INOUT,$$

the cost of a controller depends only on its criticality factor. The more reliable it is, the more expensive it is.

The most critical functions (class A) must be assigned to the most reliable controllers (category 0) and the less critical functions (class NC) must be assigned to the cheapest, then less reliable controllers (category 2). Moreover, two functions whose safety levels are too different (for instance A and C, A and NC, B and NC) must not be assigned to the same controller because development, testing and monitoring of the codes which will implement these functions must be different. Hence, a controller of

- category 0 can only host A-class and B-class functions,
- category 1 can only host B-class and C-class functions,
- category 2 can only host C-class and NC-class functions.

A graphical representation of these compatibility relations is given in Table I. The X mark indicates that a category of controllers is compatible with the class of functions.

TABLE I
SAFETY COMPATIBILITIES BETWEEN THE CLASSES OF FUNCTIONS AND THE CATEGORIES OF CONTROLLERS

	A	B	C	NC
0	X	X		
1		X	X	
2			X	X

A loose upper bound on the number of controllers of each category can then be derived. Let us define by $\mathbb{F}_A$ (*resp.* $\mathbb{F}_B$, $\mathbb{F}_C$,$\mathbb{F}_{NC}$) the set of A-class (*resp.* B,C,NC) functions such that $\mathbb{F} = \mathbb{F}_A \cup \mathbb{F}_B \cup \mathbb{F}_C \cup \mathbb{F}_{NC}$. Then, as we seek to minimize the cost of controllers, we need at most $|\mathbb{F}_A|$ (*resp.* $|\mathbb{F}_B|$, $|\mathbb{F}_C| + |\mathbb{F}_{NC}|$) controllers of category 0 (*resp.* 1, 2)[2]. In fact, if we consider the solution that consists in each controller hosting only one function, functions of $\mathbb{F}_A$ would be packed in controllers of category 0, those of $\mathbb{F}_B$ would be packed into controllers of category 1, and functions of $\mathbb{F}_C$ and $\mathbb{F}_{NC}$ would be packed into controllers of category 2.

Using this loose upper bound and following safety constraints, we can construct $\mathbb{S}$, the set of couples $(f_i, C_j) \in \mathbb{F} \times \mathbb{C}$ such that the class of f_i is incompatible with the category of C_j.

Besides, as an input of our problem we are given a list $\mathbb{L}$ of couples of redundant functions $(f_i, f_{i'})$. The redundancy relation is commutative (*i.e.* if $(f_i, f_{i'}) \in \mathbb{L}$ then $(f_{i'}, f_i) \in \mathbb{L}$) and transitive (*i.e.* if $(f_i, f_{i'}) \in \mathbb{L}$ and $(f_{i'}, f_{i''}) \in \mathbb{L}$ then $(f_i, f_{i''}) \in \mathbb{L}$).

Let us consider also the following Boolean variables

- x_{ij} that should take the value 1 iff the function f_i is assigned to controller C_j, and
- $Used_j$ that should take the value 1 iff the controller C_j is used, *i.e.* hosting at least one function.

Then, the assignment problem can be modelled by the following Integer Linear Program (ILP):

$$minimize \sum_{C_j \in \mathbb{C}} COST_j \cdot Used_j \quad \textit{subject to}$$

$$\begin{cases} \forall f_i \in \mathbb{F}, & \sum_{C_j \in \mathbb{C}} x_{ij} & = 1 & (1) \\ \forall C_j \in \mathbb{C}, & \sum_{f_i \in \mathbb{F}} inout_i \cdot x_{ij} & \leq INOUT & (2) \\ \forall C_j \in \mathbb{C}, & \sum_{f_i \in \mathbb{F}} cpu_i \cdot x_{ij} & \leq CPU & (3) \\ \forall (f_i, C_j) \in \mathbb{F} \times \mathbb{C}, & x_{ij} & \leq Used_j & (4) \\ \forall C_j \in \mathbb{C}, \forall (f_i, f_{i'}) \in \mathbb{L}, & x_{ij} + x_{i'j} & \leq 1 & (5) \\ \forall (f_i, C_j) \in \mathbb{S}, & x_{ij} & = 0 & (6) \\ \forall (f_i, C_j) \in \mathbb{F} \times \mathbb{C}, & x_{ij} & \in \{0,1\} & (7) \\ \forall C_j \in \mathbb{C}, & Used_j & \in \{0,1\} & (8) \end{cases}$$

This ILP minimizes the global cost of used controllers. Constraint (1) models the fact that a function must be assigned to one and only one controller. Constraints (2) and (3) satisfy the limited capacities of controllers in terms of CPU and INOUT. Constraint (4) indicates whether a controller is used or not. Constraint (5) ensures the separation of redundant functions, *i.e.* their assignment to different controllers. Constraint (6) ensures the compatibility of the classes of functions with the categories of controllers to which they are assigned.

For solvers that may be more efficient on ILPs without Boolean constraints, the domain of variables x_{ij} in (7) and $Used_j$ in (8) can be relaxed to $\mathbb{N}$. In fact, the minimization objective and the set of contraints (1) and (4) bind their values to 0 or 1.

III. PROBLEM IDENTIFICATION

The assignment problem addressed in this paper can be seen as a generalization of more than one variant of the Bin-Packing problem (BP) [6]. This combinatorial problem consists of packing items of different sizes into the minimum possible number of identical bins with a given capacity. In its multi-dimensional variant also known as Vector Bin-Packing (VBP), items and bins are multi-dimensional [7]. The VBP problem is an important problem that arises in a variety of industrial applications such as virtual machine placement [8]. In control functions assignement, dimensions may correspond to cpu and memory consumption and the number of inputs/outputs required.

When considering separation constraints, the problem becomes a VBP with conflicts [9]. This problem combines the VBP with the Vertex Coloring problem and has many applications [10]. A conflict graph in which each vertex corresponds to a conflicting item is provided. A couple of conflicting items that cannot be assigned to the same bin are represented by an edge in the conflict graph between the two corresponding vertices. The number of conflicts of an item, which indicates the number of items it is in conflict with, is

[2] $|\mathbb{F}|$ represents the cardinality of the set $\mathbb{F}$.

then given by the degree of its corresponding vertex in the conflict graph.

The conflict-free version of our problem, *i.e.* without considering separation constraints, can be seen as an instance of a Multiple-Choice VBP (MVBP) [11]. In this variant of the bin-packing problem, different types of bins with differents costs and characteristics are provided, and items have different incarnations depending on these types. Due to safety constraints, the addressed problem is a special case of the MVBP in which a function would have no incarnation for incompatible types of controllers.

Both the bin-packing and the vertex coloring problems are strongly NP-hard [12]. Thus, as a generalization of these two problems, the control functions assignment problem addressed in this paper is also strongly NP-hard. Real instances we are aiming at may have around ten thousand functions and need hundreds of controllers with an average number of functions per controller that may exceed 40. Such instances are beyond the solving capacity of exact algorithms for which there exists instances of two hundred items that are still open, *i.e.* for which no optimal solution is known [13]. On the other hand, for such instances with high number of items per bin, heuristics provide good solutions because the waste in bins tends to be smaller.

A. Contribution

The main contribution of this paper is a safety-level aware vector bin-packing approach that mimimizes the cost of the set of controllers of the operational architecture by assigning functions to safety compatible controllers while satisfying capacity and separation constraints. This is accomplished as follows.

First, two strategies are introduced in order to handle the multiple-choice facet of the problem that results from safety compatibilities between functions classes and controllers categories. These strategies, a class oriented and a category oriented one, specify orders into which the different classes of functions are assigned.

Then, at each step of these strategies, a bin-packing algorithm is called to assign functions. A heuristic that is an improved version of the First-Fit Decreasing algorithm (FFD) is proposed. The FFD, that is a greedy algorithm, has been shown to be a powerful fast heuristic to solve the one-dimentional bin-packing problem. The version proposed was developed specifically to manage the multi-dimensional and the conflicts aspects in accordance with the proposed strategies. The FFD operates by first sorting functions which may greatly impact the results. Three different sorting criteria are proposed that alternate between giving priority to capacity constraints satisfaction and conflicts resolving.

Finally, several experiments are performed in order to check the accuracy of the ordering strategies proposed and the adapted version of the FFD.

B. Related works

In [3], the authors propose a technique based on the verification of a reachability property on a network of communicating automata to model the assignment problem. However, this technique is meant more to find a feasible solution that satisfies constraints than to obtain an optimized one. Besides, controllers are considered of the same cost no matter their criticality factor.

As mentioned before, the problem addressed in this paper is one of several variants of bin-packing problem. Despite the extremely rich literature in this field, we were not able to find works that consider jointly both the multiple-choice of bins and the conflicting items when solving a vector bin-packing problem. In [13] and [14], the authors propose an exact technique based on an Arc-Flow formulation that represents all the feasible packing solutions of a problem in a compact graph. Different arc-flow models are then built using this formulation for the VBP, the one-dimensional BP with conflicts and the MVBP. Though it seems possible to combine these models in order to build an arc-flow model for our specific problem and for general MVBP with conflicts, we will be quickly limited by the instances that can be treated using such formulation. In fact, in addition to the size of our real instances that may exclude the use of any exact methods, this arc-flow technique is very sensitive to the number of items per bin (10 items per object is a maximum limit).

In [11], a polynomial-time approximation algorithm for the MVBP is proposed; the approximation ratio depends on the number of dimensions. On the other hand, VBP with conflicts are hard to approximate for general graphs of conflicts. Then, in most of the existing works this variant is considered on restricted graphs of conflicts. The authors in [15] propose approximations for the 2-dimensional bin-packing with perfect and bi-partite conflicts graphs. In our instances specification, separation constraints do not follow a specified pattern; their conflict graph should then be considered as general. In [16], a heuristic named *ModifiedFFD*, which adds a test for the absence of conflicts, is proposed to solve the BP with Conflicts. The approach we propose in the present work extends, in some way, this heuristic with multidimensional and multiple-choice aspects.

IV. Safety-Level Management

In this section, two assignment strategies are proposed to manage safety compatibilities between functions and controllers with the objective of reducing the cost of the global set of controllers of the architecture. Both strategies specify an order in which the classes of functions are assigned to compatible controllers.

A. Class based strategy

The idea of this first strategy is to define an order such that it is easy to remove the uncertainty (*i.e.* disambiguate) about the category of the controller into which a function will be assigned. Therefore, functions of classes that are compatible with only one category are assigned first since there is no ambiguity about the category of controllers to which they can be assigned. The other classes with multiple choices are then treated in a second time.

Since functions of class A and class NC can only be packed respectively into controllers of category 0 and 2, the strategy

proceeds by packing them first. Packing first the functions of class A permits to set definitely the number of controllers of category 0 which are the most expensive ones but the only ones able to host A functions; the B functions which are compatible with A functions (see Table I) may be assigned later to cheaper controllers of category 1 if there is no more room available in the controllers of category 0. On the other hand, the number of controllers of category 2 obtained by assigning functions of class NC is only a lower bound on the number of controllers of this category. These assignments are represented by steps (1) and (2) of Algorithm 1. Since they involve different classes of functions and different categories of controllers, these steps can be interchanged and even parallelized.

Algorithm 1 Class based strategy

1: Pack functions of class A into controllers of category 0
2: Pack functions of class NC into controllers of category 2
3: Pack functions of class B into the remaining space of already used controllers of category 0 or 1, otherwise use new controllers of category 1
4: Pack functions of class C into the remaining space of already used controllers of category 1 or 2, otherwise use new controllers of category 2

Afterwards, the algorithm resumes at step 3 by packing functions of class B. It uses at first the remaining capacities in already created controllers of category 0 at step 1, and if this is not sufficient, it creates new controllers of category 1 as they are compatible and cheaper. Finally, at step 4, functions of class C are packed into the available space provided by already created controllers at step 2 and 3 and by creating new controllers of category 2 if necessary.

Notice, for these last two steps, that it is important to proceed as detailed in the specified order so as to avoid the ambiguity that can arise if these steps are inverted. Indeed, if functions of class C are packed first, it would be difficult to determine whether new created controllers should be of category 2 or category 1 in anticipation of the possible assignment of functions of class B into these controllers.

B. Category based strategy

In this second strategy, the order is specified such that the number of controllers created for each category is minimized according to their cost. The categories of controllers are first ordered from the most to the least expensive. Then, following this order, new controllers of each category are allowed to be created only by the functions that have no choice for a cheaper compatible category. Unlike the class oriented strategy, this approach prevent revisiting the number of controllers of a category once it has been treated.

Since controllers of category 0 are the most expensive, they are treated first. This category is compatible with functions of classes A and B. However, for this last class, there is another choice of category of controllers that is cheaper. Thus, at this first step, only functions of class A are assigned and allowed to create controllers of category 0. Afterwards, category 1 is processed. For similar reasons, the only functions allowed to be assigned at this step are those of class B. However, since they are compatible with category 0 and in order to preserve the objective of minimizing the cost of the global architecture, the use of remaining space in the controllers of category 0 takes priority over the creation of new controllers of category 1. Finally, functions of class C are packed together with functions of class NC since they are both compatible with category 2 that is the cheapest one. Nonetheless, functions of class C that fit in the available space provided by already created controllers of category 1 must be assigned in these controllers before using controllers of category 2 for obvious reasons.

Algorithm 2 Category based strategy

1: Pack functions of class A into controllers of category 0
2: Pack functions of class B into the remaining space of already used controllers of category 0 or 1, otherwise use new controllers of category 1
3: Pack functions of class C and NC into the remaining space of already used controllers of category 1 (only class C is compatible) or 2, otherwise use new controllers of category 2

The predetermined orders specified by these strategies allow handling the multiple choice facet of the MVBP with conflicts. In fact, at each step of Algorithms 1 and 2 the category of controllers to be created is defined. Thus, at each one of these steps, the assignment problem to be solved is a VBP with conflicts for which a heuristic is proposed in the next section.

V. Heuristic for the VBP with Conflicts

The algorithm to solve the VBP with conflicts problem is first presented in this section. Since this algorithm relies on a FFD approach, several criteria to sort the functions to be assigned are then defined. Complexity of the heuristic is finally addressed.

A. First-Fit Decreasing Assignment Approach

The first-fit decreasing packing algorithm (*see.* Algorithm 3) consists in going through the list of functions (lines 4-16) and for each function to pack it in the first encountered controller in which it can fit in (lines 5-10). If there is no already created controller in which the function can fit in, then a new controller is created and the function in assigned to this controller (lines 11-15). In this decreasing variant, the functions are first sorted in a non-increasing order following their characteristics (line 2) before they are packed, which corresponds to the simple policy that consists of packing big items first in the original algorithm.

Note that each time a function is assigned to a controller, the remaining capacities of this controller must be updated (lines 7 and 13).

Initially the FFD was designed for the one-dimensional bin-packing problem and without considering conflicts between the items to be assigned. Thus, the FFD algorithm has to be

Algorithm 3 First-Fit Decreasing

input: A set of functions $\mathbb{F}$
output: A set $\mathbb{C}$ of $nbrControllers$ new controllers to which the functions of $\mathbb{F}$ are assigned

function FFD($\mathbb{F}$)
  $sorted\mathbb{F} \leftarrow$ sorted functions of $\mathbb{F}$ in a non-increasing defined order
  $\mathbb{C} \leftarrow \emptyset$ and $nbrControllers \leftarrow 0$
  for All functions $f_i \in sorted\mathbb{F}/i \leftarrow 1$ to $|sorted\mathbb{F}|$ **do**
    for All controllers $C_j \in \mathbb{C}/j \leftarrow 1$ to $nbrControllers$ **do**
      if f_i fits in C_j **then**
        Assign f_i to C_j
        Break the inner loop to assign the next function
      end if
    end for
    if f_i is not assigned **then**
      $nbrControllers \leftarrow nbrControllers + 1$
      $C_{nbrControllers} \leftarrow$ CREATENEWCONTROLLER(f_i) and $\mathbb{C} \leftarrow \mathbb{C} \cup C_{nbrControllers}$
    end if
  end for
  return $\mathbb{C}$
end function

extended and adapted to consider these new constraints. First, the fitting in conditions must be defined which is detailed in the next subsection. Besides, the policy of controllers creation has to be specified since for some classes of functions different categories may be compatible (Subsection V-C). Finally, since the addressed problem is multidimensional, it is necessary to define how functions are sorted (Subsection V-D).

B. Fitting in conditions

A function f_i fits in a controller C_j if and only if the following conditions are satisfied

- the safety level sl_i is compatible with the criticality factor CF_j.
- the controller C_j has enough remaining cpu and inout capacities to perform f_i.
- f_i is not in conflict with any already assigned function $f_{i'}$ to the controller C_j.

Several ways exist to check quickly the last condition. One way is to keep for each function a list of controllers to which a conflicting function has already been assigned. These lists are updated as the assignment process advances. Then, if a controller satisfies the first two fitting in conditions but belong to this list, it is excluded.

C. Controllers creation policy

The policy used to set the category of a new created controller is defined in Algorithm 4

The consistency of a creation policy with the cost minimization objective is ensured by the order in which functions of different classes are packed. Notice that the policy specified in Algorithm 4 is compatible with the orders defined by both the class and the category based strategies presented in Section IV.

Algorithm 4 Create a new controller

input: A function f_i
output: A new controller C_j of a compatible category with f_i

function CREATENEWCONTROLLER(f_i)
  if f_i is of class A **then**
    Create a new controller C_j of category 0
  else if f_i is of class B **then**
    Create a new controller C_j of category 1
  else if f_i is of class C or NC **then**
    Create a new controller C_j of category 2
  end if
  Assign f_i to C_j
  return C_j
end function

D. Sorting criteria

In the original version of the bin-packing problem, items are sorted based on their unique dimension: their size. Given the additional set of constraints to be treated, several criteria can be adopted to sort functions. The generic idea in that case is to aggregate subsets of characteristics of an object to get only one characteristic that represents this object. In one sense, this will have for effect to reduce the multidimensional problem into a mono-dimensional one for which the original version of the algorithm can be applied [7], [8].

Two aggregators were adopted whereby functions are sorted according to:

- *maxConsumption*: the relative maximum of the consumption of inout and cpu capacities

$$\max(\frac{cpu_i}{CPU}, \frac{inout_i}{INOUT}).$$

- *meanConsumption*: the relative mean of the consumption

of inout and cpu capacities

$$\frac{cpu_i}{CPU} + \frac{inout_i}{INOUT}.$$

It is necessary to use relative values instead of absolute ones as in the general case dimensions are not of the same nature. Note also that these aggregators present the advantage of being easily extensible to more than 2 dimensions. Nonetheless, they present the disadvantage of giving priority to resolving capacity constraints and ignore the conflicting aspect of the problem addressed, which may lead to an overestimation of the necessary number of controllers. This can be highlighted by the following example.

Let us suppose that at the end of an assignment process two last functions f_j, f_k remain such that $f_j > f_k$ according to both previously defined sorting criteria. It is also supposed that there exists only two controllers C_1 and C_2 that offer enough capacities to host f_j and f_k but cannot host both of them at the same time (Fig. 2).

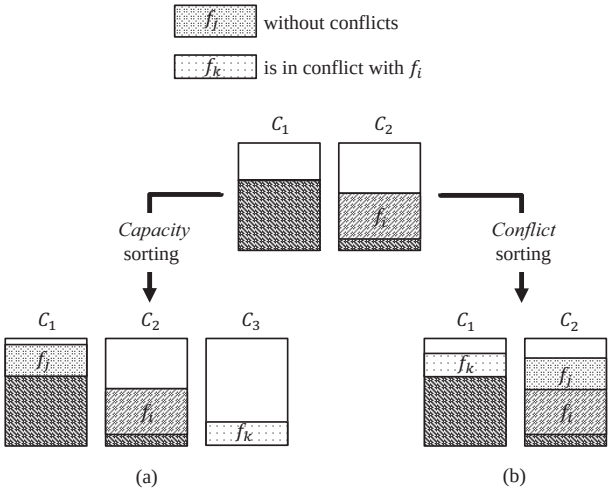

Fig. 2. Impact of sorting functions according to their number of conflicts.

Then, according to Algorithm 3 and a criterion based on relative consumption, f_j is assigned first to C_1. Now, if it is supposed that a function f_i that is in conflict with f_k is already assigned to C_2, then a new controller C_3 must be created to assign f_k (Fig. 2a) while a better solution would have been to assign f_k to C_1 and f_j to C_2 (Fig. 2b). In order to overcome such a case another aggregator is considered. This agregator sorts functions according to

- *nbrConflicts*: their number of conflicts $+(\frac{cpu_i}{CPU} + \frac{inout_i}{INOUT})/2$.

According to the first term of this sum, the more a function has conflicts with other functions the more it is preferable to assign it first. Functions with no conflicts will be assigned at last. The second term is a relative mean of capacities consumption that permits to differentiate functions with the same number of conflicts. The value of this second term is intentionally smaller than one in order to avoid disrupting the influence of the first term.

The impact of these different criteria will be discussed through experiments in the next section.

E. Complexity of the heuristic

In its original version, the First-Fit Decreasing algorithm has a complexity $O(n \log n + kn)$; n being the number of monodimensional items to be assigned and k the number of bins in the solution. The first part $O(n \log n)$ is the result of the sorting of items performed at the beginning of the algorithm [17] and the second part $O(kn)$ is due to the fitting-in test. This last part supposes that the FFD will have to visit all the bins before being able to assign any item. Using a 2-3 tree data structure [18], the complexiy of finding the first bin with enough capacity can be improved to $O(n \log k)$ instead of $O(kn)$. However, this structure assumes that bins can be ordered (*i.e.* sorted) which cannot be done when considering conflict constraints. In fact, conflicts are functions dependent unlike capacities that are controllers inherent.

In the d-dimensional case, the first part becomes $O(n \log n + dn)$ and the second part $O(d \times kn)$. If we assume that the number of conflicts of a function is of the same order of magnitude as the final number of controllers, the global complexity of the heuristic is $O(|\mathbb{F}| \log |\mathbb{F}| + d|\mathbb{F}| + d|\mathbb{C}||\mathbb{F}|)$ which is equal to $O(|\mathbb{F}| \log |\mathbb{F}| + d|\mathbb{C}||\mathbb{F}|)$.

VI. Experimental Results and Discussion

The objective of this section is twofold:

- First, the accuracy of the results which are provided by the proposed heuristic is assessed by comparing these results to optimal solutions. This comparison must clearly be done on the basis of samples of reduced size (200 functions) because optimal solutions cannot always be obtained for full scale problems (10,000 functions), otherwise heuristics would not be needed. The impact of the three sorting criteria defined at Subsection V-D is studied too in this set of experiments with reduced size samples.
- Second, the two safety-level management strategies developed in Section IV are compared on the basis of full scale samples.

A. Accuracy evaluation

The input data of the experiments whose results are given and discussed in this part are one thousand samples of 200 functions each. Those samples were generated automatically according to characteristics provided in a specification of anonymized data that specifies by mean of intervals and their proportions: cpu loads, number of inouts, number of functions by class, number of conflicting functions and for each conflicting function its number of conflicts (*see.* provided datasets).

Cost computation has been based on the assumption that a controller of a certain category is twice more expensive than a controller of the immediately below category and the optimal solutions were obtained by solving the ILP proposed in Section II by using the *CPLEX* solver [19].

Before a thorough analysis of the results, two general conclusions can be drawn up:

TABLE II
COMPARISON OF SORTING CRITERIA (VALUES IN %)
Running time per sample per criterion $<$ 1ms

Criterion	CLASS BASED STRATEGY			CATEGORY BASED STRATEGY		
	NbrOptimal	*AvgRelInc*	*MaxRelInc*	*NbrOptimal*	*AvgRelInc*	*MaxRelInc*
maxConsumption	18.8	5.87	26.67	11.6	6.55	30.00
meanConsumption	20.7	5.53	24.14	10.3	6.33	24.14
nbrConflicts	56.2	1.92	14.29	67.9	1.39	12.90
Best solution	59.8	1.63	13.79	70.1	1.21	12.90

- In most cases, the optimal solutions were delivered by *CPLEX* in less than one minute per sample; however, some samples needed more than one hour to be solved. The proposed heuristic provided a solution in less than one millisecond per sample thanks to the adopted greedy approach. If this solution is accurate, *i.e.* near the optimal, enough, as it will be shown in what follows, this very short runtime is a clear benefit of our contribution, even on small-sized examples.
- For the optimal solutions, the controllers are on average filled up to 76% in INOUT and 96% in CPU. These optimal occupancy rates indicate how much of the capacities offered by the controllers are consumed in an optimal solution and, for bin-packing problems, may help to evaluate how hard is a sample to solve to the optimum using a heuristic. The high rates which were experimentally obtained, in particular for the cpu consumption, point out that the addressed problem was not trivial.

However, the heuristic may provide for some samples an overestimation of the cost of the set of controllers that is required to host every function of this sample. This overestimation may come from the strategies to manage the different safety levels which are described in Section IV, the underlying packing FFD algorithm or the adopted sorting criterion used to solve the VBP with conflicts problem. The impact of the last two causes is studied in the remainder of this subsection; that of the first cause is the topic of the next part.

1) Impact of the FFD algorithm:
In order to isolate the impact of this algorithm from that of the safety-level management strategy, focus must be put only on the number of controllers of category 0 that is obtained at the end of the first step of Algorithm 1 (or 2); the numbers of the controllers of the other categories are not considered because they are computed at following steps and therefore depend on the strategy. Hence, experiments were performed to compute this number by using separately the three criteria described in Subsection V-D for each sample. The best solution, among the three obtained, was kept to evaluate the impact of the FFD algorithm.

Those experiments showed that the FFD algorithm computed the optimal number of controllers of category 0 for 97.3% of samples. This very high performance of the heuristic is closely related to the large number (several tens) of functions per controller that characterizes the solutions and that allows reducing the unused capacities. In fact, the FFD is known for suffering from a high fragmentation rate of unused capacities due to its policy that consists in choosing the first encountered controller in which a function can fit in. As a result, while the total of unused capacities in already created controllers might be several times greater than the needed capacities to host a function, the heuristic may not be able to assign this function otherwise than by creating a new controller because of the fragmentation. Fortunately, for cases with a high number of functions per controller, these unused parts of capacities are quickly and progressively filled in by the numerous small functions. The excellent result obtained using the adapted FFD proposed avoids adapting more advanced and consuming techniques to solve the VBP with conflicts [20].

A detailed analysis of the results pinpointed that the criterion *nbrConflicts* that prioritizes conflicts resolution allowed by itself obtaining 96.8% of optimal solutions while each one of the other two criteria provides only 55% of such solutions. However, at this stage, it was too early to decide definitely whether to prune or not those criteria that consider only consumptions and not conflicts. Therefore, it was decided to deepen the analysis on the impact of the sorting criteria and their interaction with the safety-level management strategies.

2) Impact of Sorting Criterion:
Each step of Algorithm 1 or 2 requires to sort the functions that are considered at this step before assigning them. As three sorting criteria have been defined at Subsection V-D, two approaches are worth considering: using only one criterion for every step of a strategy or selecting several criteria for the different steps. The results obtained for these two approaches and the two strategies are given respectively at Table I and Table II. Three indicators were adopted to compare the criteria:

- *NbrOptimal*: The number of optimal solutions obtained.
- *AvgRelInc*: The average relative increase in cost.
- *MaxRelInc*: The maximum relative increase in cost.

The second indicator is an average value over the thousand samples whereas the third indicator highlights the worst case performance.

a) Sorting according to only one criterion: Comparison of the first and second rows of Table II shows that sorting functions according to the relative mean of their capacities consumption performs slightly better than sorting by using the *maxConsumption* criterion for the class-based strategy. This is not true for every indicator for the other strategy; only 103 optimal solutions are obtained with the *meanConsumption* criterion whereas 116 such solutions are given when the functions have been sorted according to *maxConsumption* criterion. Hence, no definite conclusion about the relative interest of these two criteria can be stated from these results.

TABLE III
COMBINING CRITERIA (VALUES IN %)
Running time per sample per combination of criteria $<$ 1ms

Combined criteria	CLASS BASED STRATEGY			CATEGORY BASED STRATEGY		
	NbrOptimal	*AvgRelInc*	*MaxRelInc*	*NbrOptimal*	*AvgRelInc*	*MaxRelInc*
maxCons. & meanCons.	31.2	4.28	21.43	23.2	4.77	20.00
maxCons. & nbrConfl.	65.8	1.37	13.33	75.0	1.01	11.76
meanCons. & nbrConfl.	65.9	1.34	13.33	76.0	0.95	10.35
All three criteria	67.9	1.24	13.33	77.6	0.88	9.68

This is not the case for the third criterion *nbrConflicts*. The third row of Table II confirms that sorting functions according to this criterion far outperforms sorting based on *maxConsumption* or *meanConsumption*.

This result must not be misinterpreted as a dominance of the vertex coloring facet of the addressed problem over the bin-packing one, in which case the heuristic proposed at Section V would be considered as a greedy coloring technique, *i.e.* a technique that computes a coloring of the vertices by means of a greedy algorithm [21]. Actually, additional experiments have shown that, if the capacity constraints are removed, *i.e.* the capacities of controllers are considered infinite, the global number of controllers is reduced more than three times for every sample. This result highlights that, even if conflicts resolution may greatly impact the size of the solution, the bin-packing facet is what constrains the most the addressed problem.

This is confirmed by the last row of Table II, Best solution, that keeps for each sample the best solution amongst the three solutions obtained with the sorting criteria. Comparison of the results of this line with those of the third line shows clearly that the optimal solution can be obtained with another criterion than *nbrConflicts* for some samples, whatever the strategy. For instance, the percentage of optimal solutions obtained by the class-based strategy is equal to 56.2% with this only criterion and 59.8% for the best solution.

To sum up, if only one criterion must be selected for every step of a strategy, *nbrConflicts* is the most relevant choice. Nonetheless, better solutions may be provided exclusively by consumption-based criteria, for some samples. Consequently, the combined use of several criteria is a promising idea that is investigated in the following part.

b) Sorting according to several criteria: Table III reports the results of experiments where two (first three rows) or three (last row) sorting criteria have been combined for the different steps of the strategies. When the three criteria are used, $3^4 = 81$ and $3^3 = 27$ combinations are respectively possible for the class-based and the category-based strategies. This approach increases of course the execution time; nevertheless, this increase is not prohibitive, due to the short running time of the heuristic proposed to solve the VBP with conflicts (less than one millisecond per sample per combination) and to the possibility of running simultaneously several occurrences of Algorithms 1 and 2 with different combinations of criteria.

All indicators are improved, whatever the strategy, which shows that, within the same sample, different classes of functions may need different sortings. Combining *maxConsumption* with *meanConsumption* enhances the results which nevertheless stay far behind those obtained when one of these criteria is combined with nbrConflicts. More than 65% (resp. 75%) of exact solutions with the class (resp. category) based strategy are yielded in these cases (rows 2 and 3 of Table III). This is an improvement of more than 10% in comparison with the third row of Table II. The combination of the three criteria allows getting 2% more of exact solutions. Hence, this study clearly highlights that the combination of at least one consumption-based sorting criterion (preferably *meanConsumption*) with the conflicts-based sorting criterion is mandatory for more accurate results. Moreover, it matters to underline that the solutions obtained with these combinations of two or three criteria are very close to the optimal ones. The average relative increase in cost (*AvgRelInc*) is smaller than 1.5% and the maximum relative increase in cost (*MaxRelInc*) is less than 14%. This shows experimentally the quality of our contribution.

Last, no detailed comparison of the results that are provided by the two strategies from Table III will be carried out, even if it is clear that the category-based strategy seems better than the class-based strategy from this table. The results presented have been obtained with reduced size samples (200 functions each) indeed, while our objective is to deal with full scale samples (10,000 functions each). Comparison of the two strategies on this basis is the aim of the following subsection.

B. Safety-level management strategies comparison

At full scale, all indicators, and in particular the maximum relative increase *MaxRelInc*, are expected to be improved. This expectation relies on well-known results for the one-dimensional bin-packing problem. In that case indeed, the FFD heuristic has been shown to use at most $(\frac{11}{9} \cdot Optimal + 1)$ bins[3] [22], where $Optimal$ is the number of bins for the optimal solution, which in the worst case corresponds to a maximum relative increase of $\frac{(\frac{11}{9} \cdot Optimal + 1) - Optimal}{Optimal} = \frac{2}{9} + \frac{1}{Optimal}$. Hence, higher is the number of bins in the optimal solution, better is the performance of the heuristic. Even if these formulas are using the number of bins and not the cost of this set of bins, a similar behavior is expected for the adapted FFD proposed in Section V to address the VBP with conflicts.

Several experiments have been carried out to compare the two strategies and assess their scalability. The input data of these experiments are one thousand samples of ten thousand

[3]The worst-case ratio $\frac{11}{9}$ of the FFD can be even smaller when the items are much smaller than a fraction of the bin capacity; the interested reader is referred to [18] for further details on FFD performance ratios.

TABLE IV
COMPARISON OF STRATEGIES AT FULL SCALE (VALUES IN %)
Running time per sample per combination of criteria < 60 ms

Combined criteria	CLASS BASED STRATEGY			CATEGORY BASED STRATEGY		
	NbrBest	*AvgRelInc*	*MaxRelInc*	*NbrBest*	*AvgRelInc*	*MaxRelInc*
maxConsumption	0.0	3.16	4.35	0.0	4.09	5.27
meanConsumption	0.0	2.76	3.93	0.0	3.47	4.57
nbrConflicts	92.7	0.005	0.13	42.3	0.08	0.51
maxCons. & meanCons.	0.0	2.69	3.93	0.0	3.40	4.57
maxCons. & nbrConfl.	93.1	0.005	0.13	42.3	0.08	0.51
meanCons. & nbrConfl.	97.2	0.002	0.13	42.3	0.08	0.51
All three criteria	97.4	0.002	0.13	42.3	0.08	0.51

functions each. The distribution of the functions into the different classes and the features of every function in a sample were generated automatically, as explained at the beginning of the previous subsection (*see.* provided datasets). These experiments reproduced at full scale the ones performed in the previous subsection, using several criteria, with the exception that no optimal solution was searched because this solution could not be obtained with *CPLEX* for this size of problem. The results are reported in Table IV, where the indicator *NbrOptimal* has been replaced by *NbrBest* that features, for a given strategy and a criterion or a combination of criteria, the percentage of best solutions among those that have be obtained for every strategy and combination of criteria. The other indicators are also computed in a relative way.

The execution time is a hundred times longer than in the previous experiment, where each sample was composed of 200 functions, which confirms the linearithmic (*i.e.* $O(|\mathbb{F}| \log |\mathbb{F}|)$ running time of the heuristic. Consequently, the proposed heuristic scales well and can be selected for the industrial application that motivated its development and for which only one sample is considered generally.

At full scale, the domination of the *nbrConflicts* criterion over the consumption based criteria is clearly confirmed. No best solution is obtained with one of these criteria or their combination. Moreover, it may be noticed that, for the class-based strategy, using the only *nbrConflicts* criterion provides 92.7% of the best solutions.

Unlike what was observed on reduced size samples, the class-based strategy takes the lead at full scale by delivering more than 97% of the best solutions. With only 42.3% of best solutions, among which only a little more than 6% are not already included in the set obtained by the class-based strategy, the category-based strategy obtains a very low score. However, despite this result, the overestimation of this strategy amounts to the cost of one controller of category 1 on average and two controllers of category 0 in the worst case. Brought back to the hundreds of controllers that compose the solution, this overestimation is around 0.1% on average and therefore not very significant. As the runtimes are short, the two strategies may be performed in parallel to be sure to always obtain the best solution when dealing with a given sample.

The accuracy of obtained best solutions was assessed by comparison to a lower bound that was computed on the basis of capacity constraints as follows. The smallest number of controllers of category 0 is computed such that the sum of their capacities exceeds the sum of consumptions of functions of class A. The sum of the remaining capacities of these controllers is then substracted from the sum of consumptions of functions of class B that is in turn used to compute the smallest number of controllers of category 1. This process is repeated again for functions of class C and NC to determine the smallest number of controllers of category 2.

The difference between the best solution and this bound is less than 0.9% in the worst case and 0.15% on average. These results highlight the accuracy of the heuristic to solve the problem addressed in this paper.

VII. CONCLUSION

This paper has shown that the industrial issue that consists in assigning a large set of control functions with different safety levels to safety compatible controllers, while satisfying capacity and separation constraints and minimizing cost, can be modeled as a Multiple-choice Vector Bin Packing with Conflicts problem. Two strategies have been proposed to solve this NP-hard problem; both are based on a FFD heuristic for which several sorting criteria have been defined.

Experiments on small-size sets of functions have clearly highlighted the accuracy of the results of this heuristic. Experiments with full scale sets have shown that both strategies scale well. The proposed method has been tested by a major company on a real instance of nuclear power plant including around 30,000 functions. The engineers performing the work confirmed that the technique scales well (a solution was obtained for every variant of the initial data set) and that the number of controllers was always smaller than that of usual solutions.

Nevertheless, it must be underlined that sets and not architectures have been considered in this work. Data exchanges between functions and communications on networks between controllers [23] have not been taken into account. Considering these features for control functions assignment is a challenging issue for further work.

ACKNOWLEDGMENT

This work has been funded by the French Ministry of Industry as part of the CONNEXION project (Control systems for power plants) of the ”Investments for the Future” program.

REFERENCES

[1] W. Dai and V. Vyatkin, “Redesign distributed plc control systems using iec 61499 function blocks,” *Automation Science and Engineering, IEEE Transactions on*, vol. 9, no. 2, pp. 390–401, April 2012.

[2] F. Basile, P. Chiacchio, and D. Gerbasio, "On the implementation of industrial automation systems based on plc," *Automation Science and Engineering, IEEE Transactions on*, vol. 10, no. 4, pp. 990–1003, Oct 2013.

[3] T. Lemattre, B. Denis, J.-M. Faure, J.-F. Pétin, and P. Salaün, "Designing operational control architectures of critical systems by reachability analysis." in *CASE*. IEEE, 2011, pp. 12–18.

[4] M. Benazouz and J.-M. Faure, "Safety-Level Aware Bin-Packing Approach for Control Functions Assignment," in *The 15th IFAC/IEEE/IFIP/IFORS Symposium on Information Control Problems (INCOM 2015)*, Ottawa, Canada, May 2015.

[5] J. Lahtinen, M. Johansson, J. Ranta, H. Harju, and R. Nevalainen, "Comparison between IEC 60880 and IEC 61508 for Certification Purposes in the Nuclear Domain," in *Computer Safety, Reliability, and Security*, ser. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2010, vol. 6351, ch. 5, pp. 55–67.

[6] B. Korte and J. Vygen, *Combinatorial Optimization: Theory and Algorithms*, 4th ed. Springer Publishing Company, Incorporated, 2007.

[7] L. Kou and G. Markowsky, "Multidimensional bin packing algorithms," *IBM Journal of Research and Development*, vol. 21, no. 5, pp. 443–448, Sept 1977.

[8] R. Panigrahy, K. Talwar, L. Uyeda, and U. Wieder, "Heuristics for vector bin packing," *Technical report, Microsoft Research*, 2011.

[9] R. Sadykov and F. Vanderbeck, "Bin packing with conflicts: A generic branch-and-price algorithm," *INFORMS Journal on Computing*, vol. 25, no. 2, pp. 244–255, 2013.

[10] C. Basnet and J. Wilson, "Heuristics for determining the number of warehouses for storing non-compatible products," *International Transactions in Operational Research*, vol. 12, no. 5, pp. 527–538, 2005.

[11] B. Patt-Shamir and D. Rawitz, "Vector bin packing with multiple-choice," *Discrete Appl. Math.*, vol. 160, no. 10-11, Jul. 2012.

[12] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co, 1979.

[13] F. Brandão and J. P. Pedroso, "Bin packing and related problems: General arc-flow formulation with graph compression," *Computers & Operations Research*, vol. 69, pp. 56 – 67, 2016.

[14] ——, "Multiple-choice vector bin packing: Arc-flow formulation with graph compression," Faculdade de Ciências da Universidade do Porto, Portugal, Technical Report DCC-2013-13, 2013.

[15] L. Epstein, A. Levin, and R. van Stee, "Two-dimensional packing with conflicts," *Acta Inf.*, vol. 45, no. 3, pp. 155–175, Apr. 2008.

[16] M. Gendreau, G. Laporte, and F. Semet, "Heuristics and lower bounds for the bin packing problem with conflicts," *Computers & Operations Research*, vol. 31, no. 3, pp. 347 – 358, 2004.

[17] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms (3. ed.)*. MIT Press, 2009.

[18] S. Martello and P. Toth, *Knapsack Problems: Algorithms and Computer Implementations*. New York, NY, USA: John Wiley & Sons, Inc., 1990.

[19] IBM-ILOG, "Cplex optimizer 12.6.0.0," 2014.

[20] A. E. F. Muritiba, M. Iori, E. Malaguti, and P. Toth, "Algorithms for the bin packing problem with conflicts," *INFORMS Journal on Computing*, vol. 22, no. 3, pp. 401–415, 2010.

[21] L. Kučera, "The greedy coloring is a bad probabilistic algorithm," *Journal of Algorithms*, vol. 12, no. 4, pp. 674 – 684, 1991.

[22] E. G. Coffman, Jr., M. R. Garey, and D. S. Johnson, "Approximation algorithms for np-hard problems," D. S. Hochbaum, Ed. Boston, MA, USA: PWS Publishing Co., 1997, ch. Approximation Algorithms for Bin Packing: A Survey, pp. 46–93.

[23] H. S. Kim, J. M. Lee, T. Park, and W. H. Kwon, "Design of networks for distributed digital control systems in nuclear power plants," in *Intl. Topical Meeting on Nuclear Plant Instrumentation, Controls, and Human-Machine Interface Technologies (NPIC&HMIT 2000)*, Washington, DC, November 2000.