INTERNATIONAL DESIGN CONFERENCE - DESIGN 2012
Dubrovnik - Croatia, May 21 - 24, 2012.

# PRODUCT ARCHITECTURE GENERATION AND EXPLORATION USING BAYESIAN NETWORKS

M -L. Moullec, M. Bouissou, M. Jankovic and J -C. Bocquet

*Keywords: architecture generation, Bayesian network, product design, uncertainty*

## 1. Introduction

Conceptual design of complex systems starts with generation of possible architectures, their exploration and eventually selection of one [Ulrich and Eppinger 2000]. This process should ideally start from a large set of potential solutions and then progressively be focused towards the solutions offering the best performances. In order to be really successful, this multi-objective optimization must start from a sufficiently exhaustive set of admissible solutions. However in real situations, designers do not have enough time and resources to envisage solutions that are really far from existing products.
The purpose of this paper is to propose an architecture generation and exploration method that is able to generate automatically a large set of potential solutions, taking into account various constraints like the compatibility of components and minimum levels for some required performances. The originality of proposed approach comes from integration of information uncertainty in the expression of the constraints and performances. Another advantage is that our method is easy to use and intuitive: all the data about the design problem can be input in a single and *graphical* model: a Bayesian network. No programming at all is required.
The sequel of this paper is organized as follows: the next section is about the motivations of our work and the state of the art. Since the reader may not be familiar with Bayesian networks, the following section is a short introduction to this mathematical model. Then we describe our method, using a bike example to make the general explanations more concrete. The last section gives the strengths and weaknesses of our method, and some perspectives.

## 2. State of the art

### 2.1 Product architecture definition

In a «systems engineering» point of view, a product can be considered as a system. According to [ESD Committee 2004], the "system architecture is an abstract description of its entities and the relationships between those entities". Ulrich [Ulrich 1995] completes this definition by adding that the product architecture is "the scheme by which the function of the product is allocated to physical components". And more precisely: "(1) the arrangement of functional elements; (2) the mapping from functional elements to physical components; (3) the specification of the interfaces among interacting physical components".
In our paper, we will focus on the last two points. We provide a method to synthesize numerous product physical architectures. But we also want to validate some product specifications. Therefore, our model allows to represent and to estimate some required performances in order to be able to make objective choices. Choices made during this phase are crucial: certain studies (e.g. [Zablit and Zimmer

2001]) showed that approximately 70 % of the product relative costs are committed by the decisions taken during the conceptual phase of the product development. Ulrich [Ulrich 1995] assessed the implications of product architecture on product life-cycle, but also on the company and notably on its product policy and its design team's management.

## 2.2 Product architecture generation

Despite its strategic role, few methods and tools are available to support this step [Yannou 2001]. Product architecture is the stage where all possible solutions are considered and evaluated to keep only some key architectural concepts which will be studied further in detail. With the introduction of new electronic technologies, products require more and more multidisciplinary approaches [Kreimeyer 2009]. Difficulties due to the diversity and to the high number of potential technical solutions appear in the conceptual design decision, when designers need to combine numerous solutions and compare them. If product is too large or complex, they cannot handle their design problem in its wholeness. They have to make preliminary choices between a few first concepts that only rely on their experience and their knowledge. This is all the more difficult given that quantitative data, at this early stage of product development, are often unknown or uncertain: it is impossible to use CAD tools or simulation to optimize a concept. One natural solution for designers is to practise *redesign* [Dieter 2000]. It reduces risks associated to the product development but also reduces the number of opportunities to design a product in rupture. Computational tools are now needed to support and steer designers in their approach of product designing and trade-offs deciding. Few methods were developed to generate automatically product architectures. We can distinguish two main approaches [Antonsson and Cagan 2001]. The first one is based either on product requirements fulfillment using black box concept and/or physics equations. A second stream consists in the expert knowledge elicitation under the form of design rules.
The following methods are specifically designed to support designers in the generation of new concepts. Bryant [Bryant 2005] presents a method that generates physical concepts from a functional architecture directly built by designers. His method uses a set of *design structure matrices (DSM)* which represents diverse dependencies between functions and components, and then allocates to functions the components coming from a design repository (i.e. a database of design knowledge). This method generates product configurations taking into account flow and structural interactions to assure the feasibility of concepts. Applied on an industrial case, the main weakness of this method is that it may not be suited to certain domains since the predefined catalogue forces the user to adapt to the model and its vocabulary. This method is however very useful in a stage of intensively creative research where designers start from scratch. To avoid this issue, Wyatt [Wyatt et al. 2012] proposes a method in which designers describe the design problem: design alternatives of components and their relations must be specified. Design rules definition is used for generation of design alternatives. This method is mainly based on expert knowledge. However, it relies on a specific graph typology that does not allow a quick understanding of the system and that may be difficult to implement during first uses. Furthermore, the generated concepts are feasible but it is not guaranteed that they meet all product requirements since there is no estimation of product performances, which can be an issue when some alternatives must be chosen. To consider product performances, the method Hierarchical Subjective Objective System (HSOS) [Rosenstein and Reich 2011] uses a genetic algorithm: design alternatives and theirs relations, called building blocks, are described under the form of a "Boolean" gene to ensure the concept feasibility. The capability of target performances fulfillment for every building block is then evaluated in a qualitative way. The global performances of each complete design alternative are calculated with weighted linear relations. Finally, a genetic algorithm generates optimal concepts with regard to performances. This method is useful when no quantitative information about alternatives is available: the generated concepts are optimal from a qualitative point of view but the fulfillment of specific quantified requirements is not ensured.  Furthermore, an additional view is needed to show how the problem has been modeled and to provide a vision of design problem that collaborators can share. This should avoid missing parameters in problem definition, or personal bias in the estimation of some performances. Another way to reduce this bias is to use quantitative data to estimate performances. In preliminary design, the available data mainly come from previous designs and expert

knowledge. They are often fuzzy or incomplete at preliminary design stages, which make concepts synthesis strenuous. However, in his method, Matthews [Matthews 2011] exploits quantitative data coming from previous designs examples to automatically induce a design model. This model takes the form of a Bayesian network where parameters and characteristics are linked by a joint probability distribution. The Bayesian network structure and the related conditional probability tables are automatically created by the analysis of parameters and characteristics values of previous design examples. An experiment with a laboratory case proved that the automatic knowledge acquisition process was successful. Matthews also tried the method on an industrial case and noticed that experts did not feel comfortable with this probabilistic model, since they did not control the net construction, and had found some relations between parameters that did not make physical sense. The generated net could not represent a basis that allowed to improve the global understanding of the problem and thus to support designers' reflection. This study emphasizes that it is necessary to provide designers with a tool that is adapted to their knowledge level and to their vision of design problem.
In view of above studies, we can deduce the following requirements for product architecture generation:

1. Design problem representation should be modelled and represented integrating designer's point of view,
2. It is necessary to manage uncertainty due to the lack of information,
3. Enlarging the solution space and ensuring solution consistency is important to support the design team,
4. A support for comparison of proposed architectures must be integrated.

In this paper, we address only a part of the architecture design framework and it concerns mostly structural architecture. The part concerning the functional architecture is to be considered and under development.

### 2.3 The improvements brought by our method

In the framework of this paper, we consider that there is some missing information, and therefore using experts' knowledge seems more appropriate. We propose a product architecture generation and exploration method for the conceptual phases based upon Bayesian networks. In order to address this problem, we have used software with an adequate graphical interface. It has the advantage of being intuitive since the model is developed with the different templates that we will define in section 4.
With regard to problem design specification, our method allows the use of qualitative and quantitative data. Moreover, the uncertainty that is inherent to designer knowledge is modelled. This makes possible to determine a level of confidence linked to the feasibility of the generated architecture. In terms of results, the model allows a global estimation of architecture performances and generates only the solutions which respect required thresholds. Thus the designer can knowingly choose which architectures he will continue to develop.
Concerning the method appropriation by designers, no notion of programming is required to implement the model. The generic character of Bayesian networks is a second advantage: indeed, the model is not based on any database or specific language. The modellers can therefore customize their model as they understand it, with the vocabulary which seems the most appropriate both for qualitative and quantitative data. The fact that designers construct their own view of problem makes them more confident in the proposed solutions. The graphical representation constitutes a common basis for a better understanding and sharing of the design problem. Moreover, if an aspect of the problem at hand was forgotten, or else, if new data is available, this can be added without difficulty. Also, the modeller can delete data which would not be relevant any more without this having an impact on the rest of the problem. Thus, the model evolves in conjunction with the designer's reflection.

## 3. Bayesian networks

In the context of this article, we are going to use Bayesian networks in a very peculiar way. Bayesian networks were originally created in order to perform Bayesian inference on probabilistic models, but we will be using them largely to represent a set of deterministic constraints that can be fulfilled or not by various architecture designs. However, we will use also some truly probabilistic features of Bayesian networks.

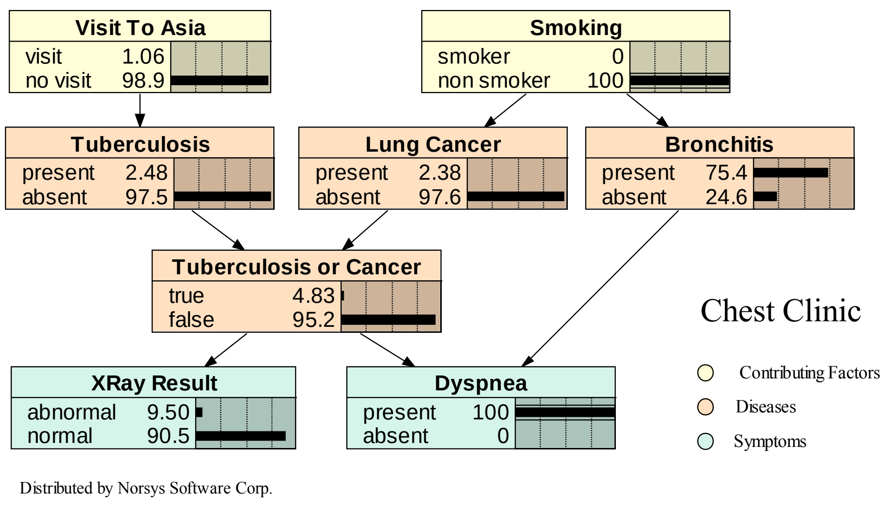


**Figure 1. "Chest Clinic", a classical example in the Bayesian networks literature**

A Bayesian network can be considered as a concise representation of the joint probability distribution of a set of random variables ($X_1$, $X_2$,....$X_n$): it is the most complete information which one can give about this set of random variables. To give a global and intuitive view of dependences between variables, it is useful to represent the Bayesian network as a directed acyclic graph, the nodes of which are associated to the random variables $X_i$. The links pointing at one variable correspond to a conditional probability distribution defining how this variable depends on its parents.
The main use of Bayesian networks is to do what is called inference: it consists in updating the probability distribution of unobserved variables of the Bayesian network, knowing the value of some observed variables. Figure 1 is a screenshot of the tool [Netica] showing the "Chest clinic" model (this is a famous pedagogical example in the literature on Bayesian networks). The tool has updated the probability distributions of all the remaining nodes, knowing that the patient has the symptom dyspnea, and that he does not smoke. Given this evidence, it is much more probable that the disease explaining the dyspnea of the patient is bronchitis, rather than tuberculosis or lung cancer.
Inference algorithms in Bayesian networks are a complex topic, which is still a field of active research. In the context of the present paper, we will rely on the tool [Netica] which uses a classical algorithm that gives exact results, but works only for discrete variables. Because of this restriction, we will have to discretize continuous variables when they are needed. Further information about the use and interest of Bayesian network is available for example in [Jensen and Nielsen 2007].

## 4. Product architecture generation and exploration using Bayesian networks

### 4.1 Introduction

Our method encompasses two main steps. The first step is manual and consists in building a Bayesian network encoding all the decision variables of the design problem, the constraints existing between them (sometimes with uncertainty), the performances as functions of the decision variables. To make this work easier, we propose to use a small number of templates that we describe hereafter. The second step is fully automatic and consists in the generation of all the admissible solutions by a program based on Bayesian inference.
To make understanding easier, we are going to illustrate our method with a bike design problem: we consider several types of components, compatible or not, and we want to create a bike weighing at most 15kg. For pedagogic reasons, we simplified the problem by considering only compatibility constraints. In reality, the architecture of a product might also depend on the components number, as

well as their relative location. Moreover, constraints could concern some components characteristics and could be coupled. We are currently testing this method also for a much larger complex system.

## 4.2 Model building

The Bayesian network we will build has to represent the entire design problem which is composed of:

- Decision variables: all the choices that designers must make;
- Characteristics: all data that are linked to decision variables and that are useful for the performances evaluation (e.g. brake weight is a characteristic linked to the choice of brakes and will be used in the estimation of the total bike weight);
- Performances: the requirements that the future product must meet;
- Constraints: compatibilities between the choices made for two or more decision variables. There are two cases:
    - Crisp constraints: it is a deterministic knowledge;
    - Uncertain constraints: designers do not have the entire knowledge to represent and to resolve their design problem. Hence, they can be led to make assumptions on some data or relations. Modelling uncertainty is going to allow us to measure the confidence that an architecture concept represents in feasibility terms.

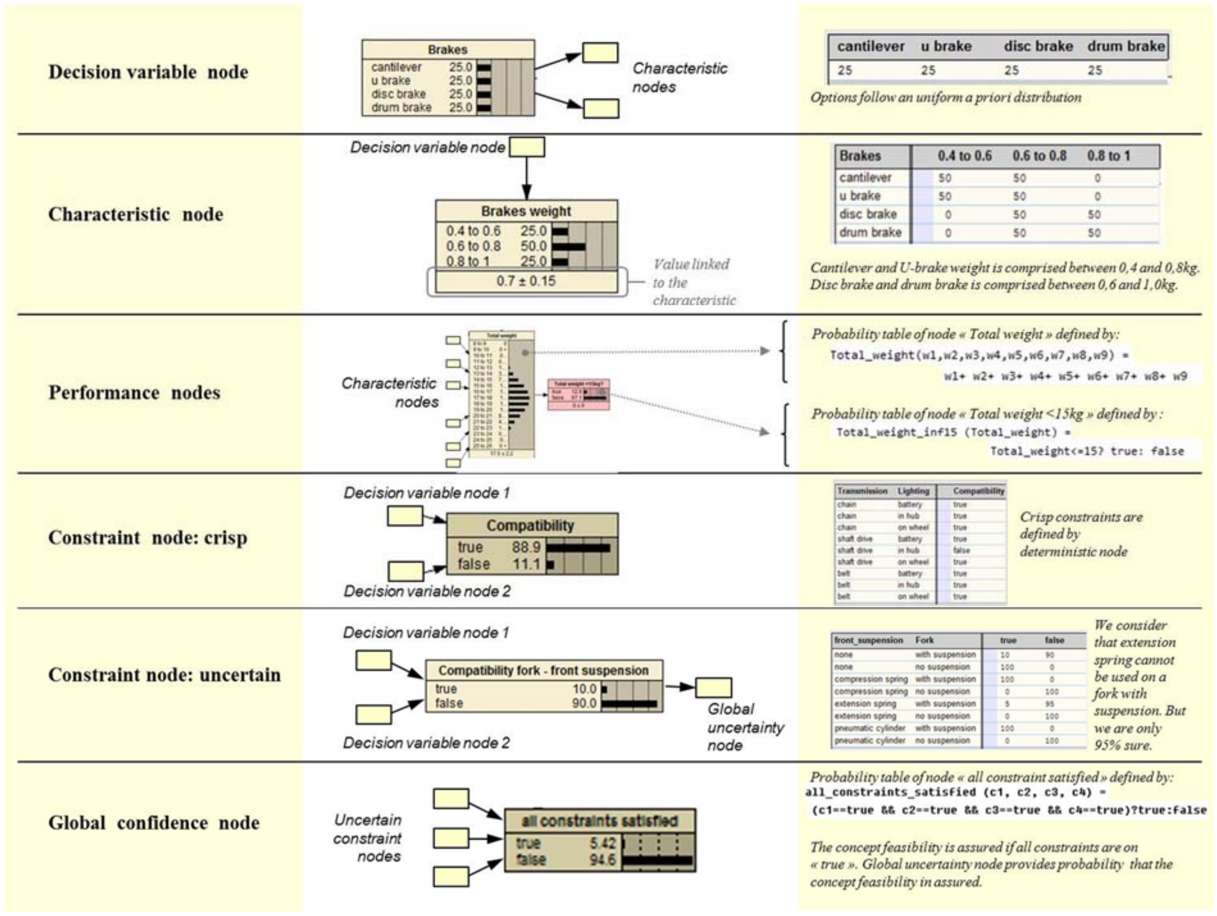
Each of the above points follows a particular template that we are going to describe in the following section.

### *4.2.1 Templates description*

The following templates are composed of nodes and arcs, the classical representation used in Bayesian networks. A *chance node* is a node whose relations with its parents are probabilistic. If its parent's values are all known, and there is no further information, then its value can only be inferred as a probability distribution over possible values. On the contrary, a *deterministic node* is a node whose relationship with its parents is given as a function of the parent values. If the parent values are all known, its value can be determined with certainty. For both node types, the node can be Boolean, discrete or continuous. An arc represents a dependency relation between two nodes. Each template is illustrated by an example and explained below.

- A decision variable is represented in a chance node. A new state is added for each new "option" of decision. A priori, all options can be chosen. The probabilistic distribution of the node is therefore taken as uniform.
- A variable characteristic must be represented in a chance node that is linked with the decision variable node that it describes. Two characteristic nodes can be linked together if the two characteristics are coupled. A numerical value can be associated to this node. This value will be used for the performance calculation. In our bike example, we need to estimate the total weight of the bike. For instance, to represent the brake weight, we define a continuous node that has been discretized and we link it to the node "brake". The corresponding probability table is described on Figure 2.
- A performance is represented by the association of two nodes. The first node is defined in the same way as characteristic nodes. It is linked to characteristic and/or decision variable nodes. Its probability table can be easily filled using an equation expression involving its parent nodes. The second node is Boolean and defines the threshold that the product must reach. If performance is over the threshold, its state is "true". If not, the state is "false". This node will be used in the generation step for the selection of "good" architectures. Several performances can be represented in the model. To stay simple, we define only one performance in our bike example: the total weight. All architectures must have a weight lower than 15kg.
- A constraint is defined with a deterministic Boolean node that is linked with the variables on which the constraint is applied. If the constraint is satisfied, the node state is "true". In our example, we define compatibility constraints between components. For example, a dynamo hub is not compatible with a speed-hub: in the line of the probability table corresponding to this combination, the constraint state is "false".

- To define an uncertain constraint node, we replace the deterministic node by a chance node. Thus, we can enter an error ratio for every combination of parent nodes in its probability table. For instance: we suppose that a shaft drive is compatible with a speed hub. But, supposing we have never designed a bike equipped with a shaft drive, we are not totally sure of that fact. So we decide to say that the probability that it is compatible is 0.9.
  All nodes with uncertainty are parents of a single node of global confidence. We consider that the concept feasibility is true if all the constraints are satisfied. Thus, we know the probability of the whole concept feasibility.



| cantilever | u brake | disc brake | drum brake |
|---|---|---|---|
| 25 | 25 | 25 | 25 |

*Options follow an uniform a priori distribution*

| Brakes | 0.4 to 0.6 | 0.6 to 0.8 | 0.8 to 1 |
|---|---|---|---|
| cantilever | 50 | 50 | 0 |
| u brake | 50 | 50 | 0 |
| disc brake | 0 | 50 | 50 |
| drum brake | 0 | 50 | 50 |

*Cantilever and U-brake weight is comprised between 0,4 and 0,8kg. Disc brake and drum brake is comprised between 0,6 and 1,0kg.*

*Probability table of node « Total weight » defined by:*

```
Total_weight(w1,w2,w3,w4,w5,w6,w7,w8,w9) =
                    w1+ w2+ w3+ w4+ w5+ w6+ w7+ w8+ w9
```

*Probability table of node « Total weight <15kg » defined by :*

```
Total_weight_inf15 (Total_weight) =
                    Total_weight<=15? true: false
```

| Transmission | Lighting | Compatibility |
|---|---|---|
| chain | battery | true |
| chain | in hub | true |
| chain | on wheel | true |
| shaft drive | battery | true |
| shaft drive | in hub | false |
| shaft drive | on wheel | true |
| belt | battery | true |
| belt | in hub | true |
| belt | on wheel | true |

*Crisp constraints are defined by deterministic node*

| front_suspension | Fork | true | false |
|---|---|---|---|
| none | with suspension | 10 | 90 |
| none | no suspension | 100 | 0 |
| compression spring | with suspension | 100 | 0 |
| compression spring | no suspension | 0 | 100 |
| extension spring | with suspension | 5 | 95 |
| extension spring | no suspension | 0 | 100 |
| pneumatic cylinder | with suspension | 100 | 0 |
| pneumatic cylinder | no suspension | 0 | 100 |

*We consider that extension spring cannot be used on a fork with suspension. But we are only 95% sure.*

*Probability table of node « all constraint satisfied » defined by:*

```
all_constraints_satisfied (c1, c2, c3, c4) =
 (c1==true && c2==true && c3==true && c4==true)?true:false
```

*The concept feasibility is assured if all constraints are on « true ». Global uncertainty node provides probability that the concept feasibility in assured.*

**Figure 2. Model templates (probabilities are given in %)**

Another type of node can be introduced. This node is an input data that is needed in the evaluation of performances. It can be a constant node, or a chance node, according to the type of information it provides. For example, in our model, we do not represent all the bike components. But the missing components have a weight and thus a role in the weight estimation of our bike. We need to represent them by a single node that represents the value of the weight of all the components we choose to not represent.

### *4.2.2 Architecture generation*

The architecture generation is done automatically by an algorithm that uses the information input in the model. Only the architectures that satisfy all constraints, fulfil the defined performance threshold and respect the chosen confidence level will be generated. Here is a description of the algorithm:

```
forall n in constraint nodes do
  input evidence n.state = “true”
end forall
N = the set of variable nodes
C = confidence level // in [0;1]
call generation (N,C)

     // here is the definition of the function “generation” called in the main program
     function: generation(variables, c)
     forall v in variables do
       forall s in v.states do
          if v.s.probability > 0 then   //select only the node states whose probability > 0
             input evidence v.state = s
             // bayesian inference
             update probabilities of all nodes
             if cardinality(variables) > 1 then
          generation(variables-{v},c);
             else  // termination of the recursion
          if global_confidence_node.true.probability > c then   // the current state of the BN
             forall v in N do                  // depicts an admissible solution
               write v.finding //finding is the state of v that has probability 1
             end forall
          end if
             end if
          end if
        end forall
      end forall
      end function
```

The algorithm encompasses three main steps: the first one consists in blocking all crisp constraint nodes on “true” in order to ensure that all combinations respect crisp constraints and performance thresholds. The second step consists in blocking, node after node, state after state, a decision variable node on a given state. Only the states whose probability is not null are examined. The exploration of the Bayesian network is carried out by a recursive function. Thirdly, when all decision variables are set, the global confidence node is examined in order to check if the confidence threshold is respected: the probability that all uncertain constraints are satisfied is defined in the global confidence node. It corresponds to the probability that the global confidence node is true. If this probability is higher than the required confidence level, the configuration is considered as a solution.

## 4.3 Bike model

### *4.3.1 Model definition*

The entire bike model is given in Figure 3. It is composed of nine decision variables comprising between two and four options. They are described by eight characteristic nodes on which depends a single performance node. Seven compatibility constraints, among which four are uncertain, will be applied on decision variables. The Cartesian product of the decision variables sets of states encodes 27648 possibilities of bike architectures.

### *4.3.2 Results*

The Bayesian network building took about 2 hours. In the generation step, 861 architectures were generated in a few seconds on a classical desktop computer with a threshold for the confidence level set at 80%.

The confidence level influences the number of generated architectures: as shown in Figure 4, the number of acceptable architectures drops as the requested confidence level grows. This phenomenon is logical: in the construction of the Bayesian model, designers have introduced several confidence levels concerning some compatibility of components. The generation algorithm will link several components into a single architecture. This fact leads to aggregation of confidence levels in a single confidence level representing the global architecture feasibility. When uncertainty in defining the compatibility of components in an architecture increases, the global confidence level of this architecture decreases.

In the table below, the compatibilities of the architecture 1 are more uncertain than ones of the architecture 2. As a result, the global confidence level of architecture 2 is much better than for architecture 1. With a required level of confidence at 80%, the architecture 2 will be in the set of admissible architectures while the architecture 1 will not. If we modeled this design problem in other architectures generators, both architectures would be admissible solutions without any other distinction. Modeling uncertainty is useful to estimate the global feasibility of the architectures so that designers can knowingly choose an architecture concept.

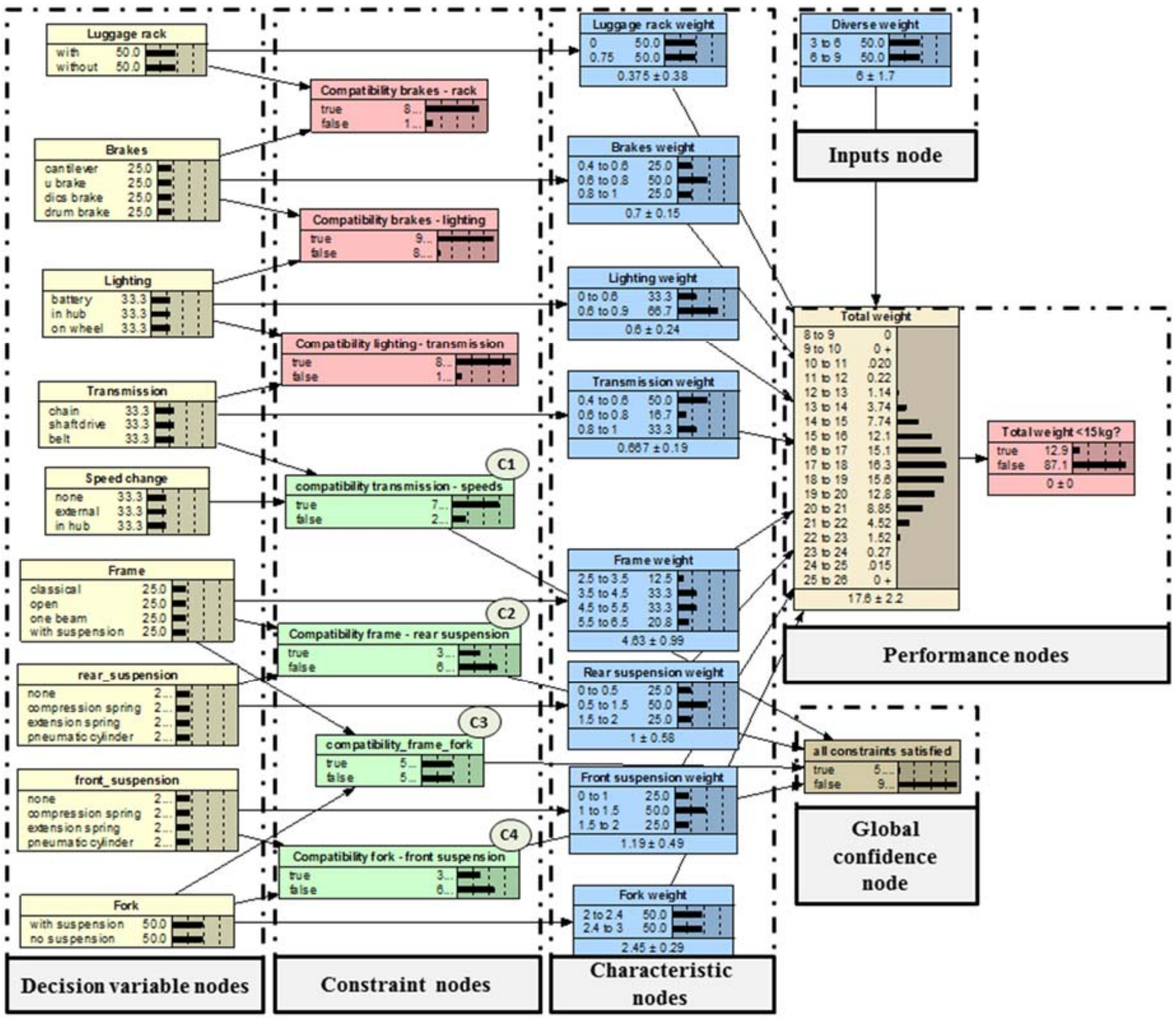


**Figure 3. Bike model**

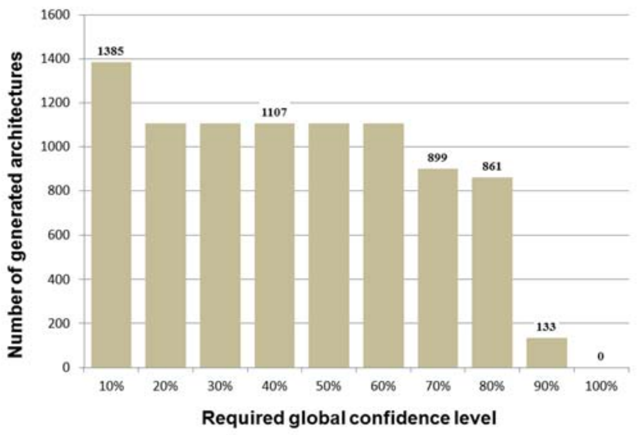


**Figure 4. Number of generated architectures in function of the required confidence level**

**Table 1. Comparison of confidence levels on two architectures**

| | | Architecture 1 | | Architecture 2 | |
|---|---|---|---|---|---|
| **C1** | *multi-speeds system* | external | 100% | in hub | 90% |
| | *transmission* | chain | | belt | |
| **C2** | *rear suspension* | extension spring | 95% | without | 100% |
| | *frame* | classical_suspended | | beam_not_suspended | |
| **C3** | *fork* | suspended | 100% | not_suspended | 100% |
| | *frame* | suspended | | beam_not_suspended | |
| **C4** | *front suspension* | pneumatic_cylinder | 80% | without | 100% |
| | *fork* | suspended | | not_suspended | |
| | **Global confidence** | **76%** | | **90%** | |

## 5. Conclusion

In this paper we propose a product architecture generation and exploration using Bayesian networks. This method allows designers to define explicitly a space of possible solutions in terms of product architectures and to generate all the solutions which fulfill all constraints and meet the required performances. A first advantage of the use of Bayesian networks is that it is possible to manipulate deterministic data as well as probabilistic data. This method is therefore useful to model all performance types. The proposed method is illustrated using a bike design problem. Although this is a relatively small design problem, it illustrates well the advantages of the proposed approach. Furthermore, a larger scale design is currently experimented in order to validate the approach.

A second strength is that implementing the method is relatively generic and quick. It can be applied for both physical and functional architectures. Moreover, the introduction of a confidence level on data allows to take into account designers knowledge, which is often imperfect, and so, to soften the model up to avoid over-constraining. On the other hand, solutions depend a lot on the way designers modeled their problem, and particularly the uncertainty estimation which is subjective. However, this weakness is compensated by progressive and interactive aspects of the method which allow designers to better consider the problem in its wholeness, and to bring corrections if needed. Another weakness appears when designers' knowledge is not sufficient to compensate for the lack of information. A way to overcome this issue is to use data from previous design examples using an approach similar to Matthews' method to construct the affected parts of the Bayesian network. The last weakness appears when constraints are not extremely strong. The number of generated architectures may be large and difficult to exploit. Therefore a coupling with multi-objective optimization methods, such as Pareto optimum, could help determine the most promising architectures. This is envisaged for future works, as well as modeling more complex problems which call a bigger diversity of data in order to validate the method and its templates.

### References

*Antonsson, E., Cagan, J., "Formal engineering design synthesis", Cambridge University Press, UK, 2001.*

*Bryant, C., Mcadams, D. A., & Stone, R. B. "A computational technique for concept generation", Proceedings of ASME-IDETC/CIE 2005, Long Beach, USA, 2005.*

*Crawley, E., Weck, O. D., Eppinger, S., Magee, C., Moses, J., Seering, W., Schindall, J., Wallace, D., Whitney, D., "The influence of architecture in engineering systems", Massachusetts Institute of Technology Engineering Systems Monograph, 2004.*

*Eppinger, S., Ulrich, K.., "Product Design and Development", Irwin McGraw-Hill New York,USA, 2000.*

*Jensen, F., Nielsen, T., "Bayesian networks and decision graphs", Springer-Verlag New-York, USA, 2007.*

*Kreimeyer, M., "A Structural Measurement System for Engineering Design Processes", Technische Universität München, 2009.*

*Matthews, P.C., "Challenges to Bayesian decision support using morphological matrices for design: empirical evidence", Research in Engineering Design, Volume 22, 2011, pp.29-42.*

*NETICA, http://www.norsys.com/*

*Rosenstein, D., Reich, Y., "Hierarchical Concept Generation by SOS", Proceedings of ICED'11, Copenhagen, Denmark, 2011.*

*Ulrich, K., "The role of product architecture in the manufacturing firm", Research Policy, 2, 1995.*

*Wyatt, D. F., Wynn, D. C., Jarrett, J. P. and Clarkson, P. J., "Supporting product architecture design using computational design synthesis with network structure constraints", Research in Engineering Design, Volume 23, 2012, pp.17-52.*

*Yannou, B. , "Préconception de Produits", Ecole Centrale Paris, 2001.*

*Zablit, P., Zimmer, L.,"Global aircraft predesign based on constraint propagation and interval analysis", Proceedings of CEAS 2001, Köln, Germany, 2001.*

Marie-Lise Moullec
PhD student
Ecole Centrale Paris, Laboratoire de Génie Industriel (LGI)
Grande voie des Vignes, Chatenay-Malabry, France
Email: Marie-Lise.Moullec@ecp.fr