Constraint-based Modeling and Exploitation of a Vehicle Range at Renault's: Requirement analysis and complexity study

Jean Marc Astesana[1] and **Laurent Cosserat**[1] and **Hélène Fargier**[2]

Abstract. As outlined by [10] "constraint programming is well-suited to treat complex configuration constraints such as compatibility and requirement- constraints in addition to standard arithmetic, comparison, and boolean constraints". As a consequence several configurators have been implemented that are based on this technology. Nevertheless, configuration applications are not be restricted to the sole problem of on-line product configuration. Different business tasks may occur at different steps of the life cycle of the product range, e.g. the design of the model, the management of the Bill of material, etc. In the present paper, we analyse and formalize a series of the business needs that arise around the problem of car configuration at Renault's, the French car manufacturer. It appears that these tasks call for a series of requests to a constraint satisfaction problem (CSP) that often differ from the ones classically handeled by the contraint programming frameworks. This application actually appears as a fruitful source of research problematics for both the CSP community and the Configuration community. We aim at motivating this research, by a formalization and a theoretical study, in terms on complexity, of the series of requests risen by this application.

1 Introduction

The product range, or "product diversity", of the vehicle diversity is very broad, with a huge number of different vehicles for each single model: for instance, there are about 10^{21} possible "Trafic" vehicles (small delivery vans). It is of course not possible to list all these vehicles in a database ; this set of vehicle has therefore to be represented in intention. The possible vehicles in product range of a given model, e.g. the Trafic model, are defined by finite domain variables, for instance the variable $fuel\_type$, whose values are $\{petrol, diesel, LPG\}$. The values available for a variable generally depend on the ones assigned to other variables: the possible options, for instance, typically depend on the entry-level versions of the vehicle. Such dependencies are specified by constraints between the variables. In other terms, the diversity of the model is defined by a Constraint Satisfaction Problem (CSP). Different business tasks occur at different steps of the life cycle of the product range associated to a given model: the modeling and the documentation of the vehicle range, the management of the Bill of material, the on-line exploitation of the model by end customers, i.e. the task of configuration.

As outlined by [10] "constraint programming is well-suited to treat complex configuration constraints such as compatibility and requirement- constraints in addition to standard arithmetic, comparison, and boolean constraints". As a consequence several configurators have been implemented that are based on this technology [1, 19, 10], taking advantage of the search algorithms and local consistency techniques such as arc-consistency provided by these formalisms. The later type of algorithm suits interactive applications since it reduce the domains from which users pick their choices. (Preference) - based search algorithms allow to get a complete configuration from a specification of the user's preference. The CSP tool box has been extended with conflict search algorithms that to provide the user with explanations of the inconsistency of her choices [12].

Nevertheless, configuration applications should not be restricted to the sole problem of on-line product configuration. The analysis of the business needs that arise around the problem of car configuration and more generally the management of the vehicle range at Renault lead to the definition of other tasks. All these tasks call for a series of requests to the CSP, that often differ from the ones classically handelled by the contraint programming frameworks.

This application thus appears as a fruitful source of research problematics In the present paper, we aim at motivating this research, by formalization and a theoretical study, in terms on complexity, of the series of requests risen up by this application. Section 2 presents the specificities of the constraint based modeling of a vehicle product range. Section 3 is a general description of the application. Section 4 then deals with its formalization in terms of CSP-oriented requests and presents our complexity study.

2 Representing the vehicle range as a CSP

2.1 Notations and background

A *CSP* is classically defined as triplet $P = < X, D, C >$. For any variable x in X, $D(x)$ is the (finite) domain of x. For any constraint c in C, $vars(c)$ denotes the set of variable involved in c.

A *assignment* d is a function that maps any $x \in X$ to a value in its domain or to the symbol $* $ ($d(x) = *$ means that the variable has not yet been assigned a value). d is complete iff it does not involve this symbol. $vars(d)$ is the set of the variables assigned by d (i.e. the x such that $d(x) \neq *$) We say that d is a complete assignment of $Y \subseteq X$ iff for any $x \in Y, d(x) \neq *$. $D(Y) = \{d, \forall x, d(x) = * \leftrightarrow x \notin Y\}$ is the set of complete assignments of Y. Finally, we say that d' extends d (to $vars(d')$) iff $var(d) \subseteq vars(d')$ and $\forall x \in vars(d), d(x) = d'(x)$ - we write this $d' \models d$.

A constraint c involves a set $vars(c)$ of variables and can be viewed as a function c from the set of assignments of $vars(c)$ to $\{\top, \bot\}$: $c(d) = \top$ iff d satisfies the constraint. No assumption is

---

[1] Renault SA, 13 avenue Paul Langevin 92359 Plessis Robinson email: jean-marc.astesana@renault.com

[2] IRIT/RPDMP, France, email:fargier@irit.fr

done about the way used to represent the constraints: it is only assumed that, for any d assigning (at least) all the variables of c, it can be checked in polytime, ideally in linear time, whether d satisfies c (denoted $d \models c$) or violates it ($d \nvDash c$).

A solution of P is a complete assignment of X that satisfies all its constraints. $Sols(P)$ is the set of its solutions. P is *consistent* iff $Sols(P) \neq \emptyset$, otherwise, it is *inconsistent*. $Sols(P)^{\downarrow Y} = \{d \in D(Y)$ s.t. $\exists d' \in Sols(P), d' \models d\}$ denotes the *projection* of $Sols(P)$ on $Y \subseteq X$. The CSP is globally consistent iff its domains are globally consistent, i.e. iff $\forall x \in X, D(x) = Sols(P)^{\downarrow\{x\}}$. Finally, we say that a additional constraint c (not necessarily in P) is *valid* (or inferred by the CSP) iff $\forall d \in Sols(P), d \models c$.

Some constraints will be represented as boolean expressions on fluents - this is typically the case of the option constraints that relate the versions of the vehicle to the option constraints (c.f.. Section 2.2). A *fluent* is a couple (x, A), where $x \in X$ and generally, $A \subseteq D(x)$, but this is not compulsory; the fluent is *elementary* when A is a singleton - it then represents an assignment of x. A *boolean expression* or *boolean condition* is a formula built on fluents using the logical operators $\vee, \wedge, \implies, \neg, \leftrightarrow$. $vars(c)$ is the set of variables used by the boolean condition c. An assignement d such that $x \in vars(d)$ satisfies the fluent $f = (x, A)$ iff $d(x) \in A$ (this is denoted $f(d) = \top$). For any assignment d of $vars(c)$, the values it returns for d, denoted $c(d)$ is computed according to the usual semantics of the logical operators.

As previously said, many of the constraints used in the specification of the range will be expressed as boolean conditions. Such expressions will also be used in other tasks and requests, typically for the definition of the Bill of materials (c.f. Section 3.3). Since it is the case in our application, we assume each of the boolean expressions handled in the paper is internally consistent ($\exists d \in D(vars(c)), c(d) = \top$) and that the satisfaction (or violation) of an expression by an assignement can be checked in polytime.

2.2 Specificities of the constraint-based representation of a vehicle range

Several extensions of the CSP paradigm have been proposed in order to handle the constraint-based definition of a catalog or a range of (configurable) products. These extensions have been motivated by difficulties and characteristics that are specific to the modeling and the handling of catalogs of configurable products. Dynamic CSPs [13], for instance suit the problems where the existence of some optional variables depends on the value of another variable. Other extensions include composite CSPs [15], interactive CSPs [9], hypothesis CSP [2], generative constraint satisfaction [18, 8], etc. Some of the characteristics handeled by these formalisms may happen in our application, but they are marginal (for instance, almost all the variables must be assigned a value, so no distinction is to be made between "mandatory" and "optional" variables).

The representation of the data is thus not the point of the paper. In the following, assume that the product range is specified by a classical CSP, and we focus on the requests that are addressed to it.

As it is the case with many real-life instances, the (again, classical) CSPs considered in our application have special features, both at the semantic level (type of variables, type of constraints) and syntactical level (characteristics of the constraints graph). Renault's range indeed follows the so-called version-option modeling. According to this modeling, the CSP variables are hierarchized in two sets:

- The *major variables* are used to define versions. They form a subset $Maj \subset X$. $Sols(P)^{\downarrow Maj}$, the restriction of $Sols(P)$ to the

variables of Maj, defines the set of versions of the range. A particular variable sometimes encodes the version itself, but it may happen that the version exists only in an implicit way, as a combination of major variables such as engine or gearbox, for instance.

- All other variables define features that depend on the version. Let us call them *minor variables*. They form a distinct subset Min of X. A given value in the domain of such a minor variable may be unavailable on a given version, possible on a second other one and mandatory on a third one. This is typically the case of the value $sunroof$ in the domain of the minor variable $roof\_type$.

Several different types of constraints are involved:

- *The major constraints* restrict the values of major variables. A major constraint is generally defined by a table, i.e. by the explicit list of the valid combinations of values for its variables.
- *The "option" constraints* link minor variables to major variables. They are of the form "$(x = v) \Rightarrow (y \in A)$", where x is a major variable, v is a value of x, y is a minor variable. For instance, suppose that specific radios are proposed as options but only in top-of-the-range versions: an option constraint specifies the radios that are available for each version.
- *The Pack constraints* are used to gather several options into a single one - this composite option is called a pack and the original options are called the components of the pack. The simplest example is the constraint that models the fact that selecting a pack means selecting all its components: $(x = pack_j) \Rightarrow \wedge_{i=1}^{n}(x_i = opt_i)$.
- *Miscellaneous constraints* gather constraints that do not belong to any of the previous categories. In practice, these constraints involve 2 to 4 major variables (in addition to their minor variables).

The way in which the constraints are represented is not restricted. Constraints may be in particular expressed as boolean expressions, e.g. the constraint $(fuel\_type = diesel) \wedge (gearbox = automatic) \Rightarrow (heating = air\_conditioning)$. It is also possible to use a purely CNF oriented description of the range, where all the variables are boolean but the present paper does not impose this restriction.

The complexity of the CSP is mainly impacted by:

- The number of major and minor variables. There are generally about 10 major variables, and no more than 150 minor variables.
- The number of different valid combinations of major variables: this number is generally lower than 100 when for the commercial ranges, but it can exceed 10,000 for some technical ranges.
- The existence of packs. The definition of packs indeed leads to many constraints, that may dramatically increase the complexity of the CSP. As a matter of fact, consider that a given option may be involved as a component of several packs, these packs being mutually exclusive. Packs are at worst up to 10. Each of them involves most often between 2 and 5 components.
- The existence of miscellaneous constraints - they may be up to 100. Unlike major and option constraints, that link variables in a hierarchical way and define an (hyper) tree, these constraints can relate any variables, increasing the complexity of the graph.

3 Design and exploitation of the vehicle range

We now describe the business requests that occur at the different steps of the lifecycle of a vehicle range : design, debug and use.

3.1 Modeling the vehicle range

The first task is to properly achieve the modeling of the range by a set of variables and constraints. The model must be consistent in a

broad sense: many constraints have actually a double meaning. Following the standard semantics of constraints, the first one is negative: constraints forbid combinations. The second one is positive: the possibilities left by some constraints have to be effective. Let us assume, for instance, that a constraint means "The heating type of vehicles with type M3 engine may be manual or automatic". If some other constraint excludes the vehicles with manual heating, the specification of the range is considered as inconsistent ; we call this property *positive consistency*. The main requests of the modeling task are:

- *(Contextual) consistency of the range:* is there at least one vehicle that satisfies the constraint-based specification ? Is there at least one vehicle that is consistent with a given partial assignment of the variables? More generally, is there at least one vehicle consistent with a given boolean expression upon elementary expressions $(x = value)$ (e.g. the expression "heating = manual") ?
- *Conflicts:* When the range is not consistent, the inconsistency has to be explained, which means finding a set of inconsistent constraints, that is minimal w.r.t. inclusion.
- *Contextual Conflicts:* Given a given partial assignment, find a minimal set of constraints inconsistent with this assignment.
- *Positive consistency of the constraints:* For any constraint c, and any tuple in $D(vars(c))$ allowed by the constraint, is it a vehicle in the range that implements this possibility ? Similarly, for any variable and any value in its domain, is it a vehicle in the range that assigns this value to the variable ?

At the end of the step of modeling of the range, the CSP is globally consistent, and obviously, consistent. All the requests that are described in the next sections deal with a (globally) consistent CSP.

3.2 Reporting the vehicle diversity

Once the range has been consistently defined, the Renault experts look for the extraction of relevant and summarized data. The idea is to select some variables (generally, less than 6) and ask the software for a summarized description of the list of all the valid combinations of values of these variables; the summarized descriptions are typically boolean expressions on simple atoms of the form $(variable = value)$ or $(variable \in subdomain)$, i.e on fluents. The main requests involved by the task of report of the vehicle diversity are:

- *Validity of a summarized description:* Given a summarized description, does all the vehicles of the range comply with it ?
- *Simplification of a summarized description:* Is a summarized description minimal, i.e. is it possible to remove a variable from it without altering its validity ?
- *Generation of summarized descriptions:* Given a set of variables, what are the combinations of their values that correspond to at least one vehicle in the diversity ? the task is here to compute a (minimal) projection of the range of the variable.
- *Counting:* How many vehicles does the range contain ? How many vehicles consistent with a given assignment (or more generally with a boolean expression) ? How many combinations of values for a given subset of variables (e.g. how many different vehicles does the range contain, this number being taken independently of the commercial name, color and country variables)?

3.3 Management of the Bill of materials

The definition of the Bill of materials (BOM) is not included in the specification of the product range. This information, that specifies on which vehicles the different parts are set up, is crucial from a business point of view. The next task deals with the design and debug of the BOM. This task takes place after the specification of the range (Section 3.1) : at this step of the application the CSP is thus globally consistent. The management of the BOM then adds new variables, and new constraints, to this CSP.

The Bill of materials is indeed organized in "generic parts": a generic part is a function fulfilled by a part. For instance, radio, as a function, is a generic part which may be fulfilled by different radios, as parts. The relationships between the CSP-based specification of the range, the generic parts and the corresponding, real parts, is defined by boolean expressions called "use cases": the use case of a part p_i is a boolean expression (over the variables of the vehicle range) specifying on which vehicles the part is set up. Generic parts can be regarded as extra variables, whose values are associated parts $p_1, ..., p_n$, including sometimes a special one that encodes the absence of the function (if the vehicles do not necessarily have a radio, the absence of radio is considered as a part.). Other modelings are possible (e.g. resorting to description logic [**?**]), but the Renault problem is simple enough to be captured by the classical CSP model without any increase of complexity; in particular, the physical parts $p_1, ..., p_n$ are not decomposed further.

For a given generic part, the Bill of materials is consistent iff each vehicle implements exactly one part (i.e. satisfies exactly one use case), and if each part is used in at least one vehicle. The following requests occur during the design and debug of the BOM:

- *Concision of the BOM:* For any given part, is there at least one vehicle in the range that uses it ?
- *Exhaustivity of the BOM:* Is any vehicle in the range using a one the parts of a given generic part ? Some vehicles are violating this condition generate a boolean condition representing them
- *Exclusivity:* Do some vehicles of the range simultaneously use two (or more) of the parts of a given generic part ? If yes, generate boolean condition representing this set of vehicles.

The expressions produced to explain the non exhaustivity or the non exclusivity of the BOM should involve as few variables as possible.

3.4 On-line configuration

The topic of constraint-based configuration has been widely dealt with in the litterature (see [13] [18] [15] [8][17] [2] [11], among others). The key idea in on-line configuration is that a customer, or a business-oriented user, step by step defines her product by choosing interactively values for the variables. She can also backtrack by relaxing some of her choices. After each action of the user, the system updates the domains so as to rule out the values that are inconsistent with the current choices (e.g. by listing them in grey rather than black). However, the user may select a ruled-out value, thus making the CSP inconsistent. Roughly, the requests involved by the task of on-line configuration are:

- *Incremental global consistency:* Ensure, after each modification of the choices that the domains are globally consistent.
- *Restoration:* In case of inconsistency, identify a (resp. all the) maximal consistent subset(s) of the user choices.
- *Conflict analysis:* In case of inconsistency, identify a (resp. all the) the (minimal) subsets of choices that are inconsistent, or inconsistent with a given partial assignment of the variables.
- *Assessment of the price, assessment of the delay:* Compute the minimum and the maximum price (resp. delay) of the vehicles of the range that are consistent with the current choices.

- *Completion of the configuration:* As soon as the major variables have been set, the user can ask for a completion of the configuration. The system must then search for a vehicle consistent with the current choices (and upon demand, optimizing price or delay).

The requests dealing with the price and the delay deserve some details. The price of a vehicle is actually a sum of elementary costs, each of them being associated with a value of a variable or a boolean expression (the elementary cost can depend on several variables). Notice that an elementary cost may be negative, because it is possible to configure a vehicle with less options than the standard one. The delay is also a sum of elementary delays: the industrial delay (which specifies when the assembly will actually begin), the assembly delay, the transportation delay, etc. The transportation delay, and actually each of the other delay typically depend on the localization of the plant that will assemble the vehicle. This parameter is not a variable of configuration. In the present presentation, we simplify the description, assuming that this localization has been chosen upstream. All the delays but the industrial one can then be considered as constants. Concerning the industrial delay, it depend on the time windows in which the different options will be available. To this extend, a distinguished variable x_t is added to the CSP. The information relative to the delay is encoded by boolean expressions of the form: $c_i \implies x_t \in UI_i$, IU_i being a union of intervals and c_i a boolean expression (eg: $color = green \implies x_t \in (-\infty, 8] \cup [10, +\infty)$).

3.5 CPU time

With respect to interactive applications, e.g. on-line configuration, the CPU time must be (at worst) about one second. Office off-line applications can take hours; but if they have to process many requests, the CPU time assigned to each request is about one second or even one millisecond. However it is possible to take a longer time for especially difficult requests (say, minutes) as long as the global time is not appreciably increased. The requests related to the design of the vehicle range, as well as the one related to the definition of the BOM are proceeded within this order of magnitude.

4 Formal modeling of the request and complexity

The present Section presents a formalization and a complexity analysis of the requests detailed in Sections 3.1 to 3.4, Some of these requests clearly define decision problems in the sense of complexity theory, e.g. problem of consistency of the product range: it obviously comes down to CSP satisfiability. Others are NP-difficult search problems. The complexity of such problems directly depends on (1) the complexity of the task of deciding the existence of the object(s) searched for (e.g. a solution of the CSP, a minimal inconsistent set of constraints) and (2) the complexity of checking the correctness of the object. In some cases, the check is tractable but the problem of existence is intractable (e.g. when searching for an assignment satisfying all the constraints); in some others, deciding of existence of the object searched for is easy, while checking its correctness is a difficult task (given an inconsistent CSP, checking that a subset of constraint is minimal inconsistent is intractable, but deciding the existence of such a subset is trivial).

4.1 Designing the vehicle range

Since the range is modeled by a CSP, deciding whether the range is consistent is a classical problem of satisfiability of a CSP, hence its

NP-completeness. In the contextual variant the CSP is consistent, as well as the boolean expression (say, c) which represents the vehicles the user is interested in. However, deciding whether the range contains vehicles satisfying this requirement is a NP-difficult problem:

Proposition 1 [3] *Given a consistent CSP $P =< X, D, C >$ and a consistent boolean condition c, deciding whether $< X, D, C \cup \{c\} >$ is consistent is a NP-complete problem. The results still hold when c is a fluent and P is consistent.*

Unsurprisingly, a *conflict* is basically a set of fluents that is inconsistent with a set of constraints, and conflicts that are minimal w.r.t. inclusion are searched for. One recognize the notion on minimal unsatisfiable subset used in AI since the late 80's, e.g. [5][6]and more recently in configuration [7][2] (for contextual conflicts, see in particular [6] and [7])

Definition 2 *Given a CSP $< X, D, C >$, a minimal conflict is a subset $C' \subseteq C$ s.t. $< X, D, C' >$ is inconsistent and $\forall C" \subsetneq C'$, $< X, D, C" >$ is consistent.*
Given a consistent CSP $< X, D, C >$ and a set of constraints $Context$ such that $< X, D, C \cup Context >$ is inconsistent, a minimal contextual conflict is a subset $C' \subseteq C$ such that $< X, D, C' \cup Context >$ is inconsistent and $\forall C" \subsetneq C'$, $< X, D, C" \cup Context >$ is consistent.

Finding a conflict is a NP-hard problem. Testing whether a subset of clauses (or constraints) is minimal inconsistent is a well-known BH_2 problem. Roughly, the complexity class BH_2 is defined by $BH_2 = \{L \in L_1 \cap L_2 | L1 \in NP, L2 \in CO - NP\}$. The hardest problems of this class are referred to as BH2-complete problems. Among them is the SAT-UNSAT problem that consists in determining, given a pair of CNF whether the first one is satisfiable and the second one unsatisfiable. The result extends to contextual conflicts.

Proposition 3 *Given CSP $P =< X, D, C >$, a set of constraints $Context$ such that $P =< X, D, C \cup Context >$ is inconsistent, and $C' \subseteq C$, deciding whether C' is a minimal contextual conflict is a BH_2 complete problem. The result still holds when P is consistent. It also holds when C' is a set of fluents.*

Finally, the notion of positive consistency is clearly a notion of global consistency of a constraint w.r.t. a consistent set of constraints.

Definition 4 *Given a consistent CSP $P =< X, D, C >$ and a constraint $c \in C$, c is globally consistent iff, for any $d \in D(var(c))$, $d \models c$ iff $\exists d' \in Sols(P), d' \models d$*

Proposition 5 *Given a CSP $P =< X, D, C >$ and a constraint $c \in C$, deciding whether c is globally consistent is a NP-hard problem. It is NP-complete whenever the $Card(vars(c))$ is bounded. The result still holds when P is consistent and c is a unary constraint.*

In summary, the requests considered in this task are NP-Hard, even when considering that the basic CSP (the range) is consistent. Most of them amount at a *sequence* of consistency checks on a sequence of CSPs. Generally, the CSPs in the sequence do not differ from each other but for one constraint.

---

[3] Proofs have been omitted for lack of space but can be found at `ftp://ftp.irit.fr/IRIT/RPDMP/PapersFargier/wsecai10.pdf`.

4.2 Reporting the vehicle diversity

This task aims at building summarized descriptions of the vehicle range - at inferring valid information - under the form of a boolean combination of fluents. The summarized descriptions used in this task are intrinsically consistent (the consistency of a sole description is not a source of complexity). Let us first consider the question of the test of validity of such a description. Since capturing the classical problem of clausal inference as a particular case, the request is CO-NP complete. The problem is also closely related to the test of redundancy of a constraint, which is known to be CO-NP complete [3]: c is valid iff it is redundant w.r.t. the CSP.

Proposition 6 *Given a consistent CSP and a consistent boolean condition c, deciding whether c is valid is a CO-NP complete request.*

With respect to the question of generation of summarized description, we consider the request of checking whether a description does coincide with the projection of the range, i.e. of $Sols(P)$, on the variables of interest - the question is Π_2^P-complete:

Proposition 7 *Given a consistent CSP $< X, D, C >$ and a consistent boolean condition c, deciding whether c is equivalent to $Sols(p)^{\downarrow vars(c)}$ (i.e. whether*
$\forall d, c(d) = \top \Leftrightarrow d \in Sols(P)^{\downarrow vars(c)}$) *is a Π_2^P-complete request.*

This proposition studies the question of recognizing a projection. Beyond this decision problem, projecting the CSP is a function problem that can be addressed by variable elimination algorithms.

Once a descriptor has been built that is valid or better, that is a projection of the diversity on the variables of interest, we would like it to be minimal with respect to the number of variables it involves.

Definition 8 *c is minimal if there is no $x \in vars(c)$ such that the satisfaction of c does not depend on the value given to x, i.e. no $x \in vars(c)$ such that for any $d, d' \in D(vars(c))$ if $d(y) = d'(y) \forall y \neq x$, then $d \models c \iff d' \models c$.*

Assuming that the number of variables in $vars(c)$ is bounded, as it is the case in our application, deciding whether c is minimal is a polynomial task. Indeed, the tuples in $d, d' \in D(vars(c))$ can then be listed in polytime. Remark that the request is probably not polytime when $Card(vars(c))$ is not bounded. The last request, the counting of the diversity, is a classical problem of counting the number of solutions of a CSP, known to be $\#P$ complete.

4.3 Management of the Bill of Materials

As said in Section 3.3, the idea is to represent each use case of a generic part by a consistent boolean condition (the internal consistency of the use cases is not a source of complexity);

Definition 9 *Given a CSP $P = < X, D, C >$ and C' the specification of a generic part, i.e. a set of (consistent) boolean conditions:*

- C' *is concise iff $\forall c' \in C', \exists d \in Sols(P), d \models c'$:*
- C' *is exhaustive iff $\forall d \in Sols(P), \exists c' \in C'$ t.q. $d \models c'$:*
- C' *is a pairwise exclusive set of use cases iff*
  $\forall c', c'' \in C', < X, D, C \cup \{c', c''\} >$ *is inconsistent.*

I.e. the BOM is concise w.r.t. the generic part considered whenever for any $c' \in C'$, the set of use cases associated to this part, $< X, D, C \cup \{c'\} >$ is consistent, and it is exhaustive whenever $< X, D, C \cup \bigcup_{c' \in C'} \{\neg c'\} >$ is inconsistent.

Proposition 10 *Given a CSP P and a set of boolean conditions C',*

- *The problem of deciding whether C' is concise w.r.t. P is NP-complete.*
- *The problem of deciding whether C' is exhaustive (resp. pairwisely exclusive) w.r.t. P is CO-NP complete.*

The result still holds when P is consistent, as it is the case for the product range once the task of design has been achieved.

When the BOM fails to satisfy the exclusivity condition, the user needs to describe the vehicles that make the condition false. The idea is to generate a boolean condition c that capture these vehicles. The task is not that difficult, provided that the pair of non exclusive use cases has been identified : it is enough to make the conjunction of the boolean description of the two use cases. Let c denote this condition. Many of the vehicles satisfying the new condition do not belong to the range. The task is then to consider the CSP $P' = < X, D, C \cup \{c\} >$ and to project $Sols(P')$ on $vars(c)$. We have seen in the previous section that this task is intractable (e.g. deciding whether a boolean condition is a marginalization of $Sols(P')$ is a Π_P^2 complete problem). The same result holds for the characterization of the vehicles falsifying the exhaustivity condition. Defining a boolean expression c as the disjunction of the negations of the use cases is easy, but many vehicles satisfying c do not belong to the range: projecting $Sols(< X, D, C \cup \{c\} >)$ on $vars(c)$ is an intractable task. Given an additional boolean condition c, the problem of deciding whether c is a concise representation of the uncovered vehicles of C' (resp. of non exclusive use cases in C') is thus NP-hard.

4.4 On-line configuration

We have seen that the global consistency of the domains is ensured upstream, before the configuration session, in order to remove from the domains any value that cannot lead to a solution. It must also be ensured during the configuration session, after each (de)assignment of some variable by the user. Hence the notion of global consistency w.r.t. an assignment:

Definition 11 *$P = < X, D, C >$ is said globally consistent w.r.t. a partial assignement d iff $\forall x \in X, \forall v \in D(x) \exists d' \in sols(P)$ such that $d' \models d$ and $d'(x) = v$.*

Hence P is globally consistent iff it is globally consistent w.r.t. the empty assignement (d, s.t. $d(x) = *, \forall x$). Recovering the global consistency after the assignment of some variables is intractable:

Proposition 12 *Given a CSP $P = < X, D, C >$, and a (partial) assignement d , deciding whether P is globally consistent w.r.t. d is a NP-complete problem. The result still holds if P is globally consistent and d assigns only one variable.*

Proposition 13 *Let $< X, D, C >$ be a globally consistent CSP, $x \in X$ and $v \in D(x)$. Unless $P = NP$, there is no polynomial algorithm that provides, for any assignment d such that $vars(d) = x$ and $d(x) = v$, a restriction D' of D (i.e. for any $x \in X, D'(x) \subseteq D(x)$) such that $< X, D', C >$ is globally consistent w.r.t. the assignment d.*

The formalization and complexity analysis of the question of the productions of conflicts and restorations have been studied e.g. in [2], assuming that contexts contains only elementary expressions (fluents). Their results extends straightforwardly to any kind of expression.

Definition 14 *[2] Given a CSP $P = < X, D, C >$ and a set $Context$ of consistent boolean expressions on the variables of X, a conflict (resp. a restoration) is a subset $R \subseteq Context$ such that $< X, D, C \cup R >$ is an inconsistent (resp. consistent) CSP. It is minimal (resp. maximal) iff there is no subset conflict (resp. restoration) such that $R' \subsetneq R$ (resp. $R' \supsetneq R$).*

Proposition 15 *[2] Given a consistent CSP $< X, D, C >$, a set $Choices$ of boolean expressions on the variables of X, and a set $R \subseteq Choices$, deciding R is a minimal conflict (resp. a maximal restoration) is a BH-2 complete problem. The result still hold when $Choices$ is a set of fluents.*

The question of maximal and minimal price assessment can be easily formalized thanks to the notion of soft CSP (see e.g. [4]), extended to the case of negative costs. Recall that given a totally ordered set L, a soft constraint (or "cost function") is a function s mapping each element of $D(vars(s))$ to a value in L, generally $L = \mathbb{Z}^+ \cup \{+\infty\}$. Classically, the elements in L are combined by a binary operation $\oplus$ that is associative, commutative and monotonic, and equipped with an identity element. Here, we relax the assumption of monotony and let the costs being *positive* integer values (negative costs are possible). In the present application, we are looking for the maximal and minimal prices of the vehicles compatible with the current assignment.

Proposition 16 *Given a consistent CSP $< X, D, C >$, a set S of cost functions on X taking its values in $\mathbb{Z}$, a partial assignement d, and an integer $\alpha \in \mathbb{Z}$, deciding whether there exist a $d' \in Sols(P)$ such that $d' \models d$ and $\Sigma_{s \in S} s(d) < \alpha$ (resp. $\Sigma_{s \in S} s(d) > \alpha$) is a NP-complete problem.*

The question of the delay assessment can be modeled in a simpler way, using a distinguished variable x_t and representing the data relative to the delay by boolean expressions of the form: $c_i \implies x_t \in UI_i$, IU_i being a union of intervals and c_i a boolean expression (eg: $color = green \implies x_t \in (-\infty, 8] \cup [10, +\infty))$.

Proposition 17 *Given a consistent CSP $P = < X, D, C >$, a positive integer variable $x_t \in X$ such that each constraint bearing on x_t is of the form $c_i \implies x_t \in IU_i$ (c_i being a boolean expression and UI_i a union of intervals), d a (partial) assignment and a positive integer α, deciding whether there exists an assignment $d' \in Sols(P)$ such that $d' \models d$ and $d'(x_t) < \alpha$ (resp. $d'(x_t) > \alpha$) is a NP-complete problem. The result still holds when each constraint of C bears on x_t and each IU_i contains only one interval*

The completion of the configuration under a criteria of minimization or maximization of the price or the delay is obviously a variant of the previous one: finding an complete assignement d' such that $d' \in Sols(P), d' \models d$ and d' maximizes/minimizes the price/the delay is thus a NP-hard problem. The completion of the configuration without any criterion is probably not easier, even under the assumption that the CSP is globally consistent: we know that there exists $d' \in Sols(P)$ such that $d' \models d$, but we do not know how to get it. The conjecture is that, unless P=NP, there is no polynomial algorithm providing a solution to any globally consistent problem.

5 Conclusion

Through the study of a series of business tasks exploiting a constraint-based modeling of a vehicle range, it appears that configuration problems call for much more requests than those usually studied. The search for an (optimal) solution satisfying all the constraints is far from being the only request : the notion of global consistency, in different variants, play for instance a central role ; other meaningful requests include not only counting problems, but also inference and marginalization problems that are sometimes beyond NP but stay in the first levels of the polynomial hierarchy.

For most of the tasks, the CSP containing the basic instance is highly consistent (it contains millions of solutions), known in advance, and is exploited by several different tasks. That is why the Applied Artificial Intelligence Department which is in charge of these applications in the Renault company, have oriented its developments toward an approach of the problem by knowledge compilation [14], that proves to be efficient (e.g. about 10^{-2} seconds for the problem of interactive on-line configuration). The question is now to determine (1) whether the algorithms promoted by the CSP community can be used in this context with a comparable efficiency, and (2) whether this community can provide efficient algorithms for problems that strongly differ from the search for an (optimal) solution, e.g. the marginalization of the CSP.

REFERENCES

[1] M. Aldanondo, H. Fargier, and M. Veron, 'From csp to configuration problems.', in *AAAI-1999 Workshop on Configuration*, pp. 101–106,, (1999).

[2] J. Amilhastre, H. Fargier, and P. Marquis, 'Consistency restoration and explanations in dynamic CSP - application to configuration', *Artificial Intelligence*, **135**(1-2), 199–234, (2002).

[3] A. Chmeiss, V. Krawczyk, and L. Sais, 'Redundancy in csps', in *ECAI*, pp. 907–908, (2008).

[4] M. C. Cooper and T. Schiex, 'Arc consistency for soft constraints', *Artificial Intelligence*, **154**(1-2), 199–227, (2004).

[5] Johan de Kleer, 'Problem solving with the atms', *Artificial Intelligence*, **28**(2), 197–224, (1986).

[6] J. L. de Siqueira N. and J-F Puget, 'Explanation-based generalisation of failures', in *ECAI*, pp. 339–344, (1988).

[7] A. Felfernig, G. Friedrich, D. Jannach, and M. Stumptner, 'Consistency-based diagnosis of configuration knowledge bases', in *ECAI*, pp. 146–150, (2000).

[8] G. Fleischanderl, G. Friedrich, A. Haselböck, H. Schreiner, and M. Stumptner, 'Configuring large-scale systems with generative constraint satisfaction', *IEEE Intelligent Systems, Special Issue on Configuration*, **13**(4), (July 1998).

[9] E. Gelle and R. Weigel, 'Interactive configuration using constraint satisfaction techniques', in *PACT-96*, pp. 37–44, (1996).

[10] U. Junker and D. Mailharro, 'The logic of ilog (j)configurator: Combining constraint programming with a description logic', in *Proc. IJCAI-03 Workshop on Configuration*, pp. 13–20, (2003).

[11] U. Junker and D. Mailharro, 'Preference programming: Advanced problem solving for configuration', *AI EDAM*, **17**(1), 13–29, (2003).

[12] Ulrich Junker, 'Quickxplain: Conflict detection for arbitrary constraint propagation algorithms', in *IJCAI'01 Workshop on Modelling and Solving problems with constraints (CONS-1)*, pp. 81–88, (2001).

[13] S. Mittal and B. Falkenhainer, 'Dynamic constraint satisfaction problems', in *AAAI*, pp. 25–32, (1990).

[14] B. Pargamin, 'Vehicle sales configuration: the cluster tree approach', in *ECAI'02 Configuration Workshop*, (2002).

[15] D. Sabin and E. C. Freuder, 'Configuration as composite constraint satisfaction', in *AI and Manufacturing Research Planning Workshop*, pp. 153–161, (1996).

[16] T. Schiex, 'Possibilistic constraint satisfaction problems or "how to handle soft constraints?"', in *UAI*, pp. 268–275, (1992).

[17] T. Soininen, E. Gelle, and I. Niemela, 'A fixpoint definition of dynamic constraint satisfaction', in *Proceedings of CP'99*, pp. 419–433, (1999).

[18] M. Stumptner and A. Haselböck, 'A generative constraint formalism for configuration problems', in *Proceedings of the Third Congress of the Italian Association for Artif. Int. (AI*IA)*, 302–313, (1993).

[19] M. Veron and M. Aldanondo, 'Yet another approach for ccsp for configuration problem', in *Proc. ECAI-2000 Workshop on Configuration*, pp. 59–62, (2003).

Annex: Sketches of the proofs

Sketch of proof [Proposition 1] The membership to NP is obvious[4]. The NP-hardness is get by reduction of SAT-CNF: X contains all the boolean variables of the CNF plus an additional boolean variable x, and C contains as many constraint $x = \top \implies cl_i$ as the number of clauses cl_i in the CNF. Obviously, the $< X, D, C >$ is consistent and the CNF is satisfiable iff $< X, D, C \cup \{(x = \top)\} >$ is consistent. $\square$

Sketch of proof [Proposition 3] The proof of BH_2 hardness is inherited from [2] : this paper indeed considers contextual conflicts built on unary constraints, i.e. on fluents. The membership to BH_2 is easy to show : showing the inconsistency of $< X, D, C \cup C' >$ belongs to CO-NP. To show that C' is minimal, for any $c \in C'$, consider the set $C'' = C' \setminus \{c\}$ and show that $< X, D, C \cup C'' >$ is consistent (using a NP- oracle). $\square$

Sketch of proof [Proposition5] Reduction from SAT-CNF : X contains all the Propositional variables of the CNF and an additional variable x; C contains as many constraints $x = 1 \implies cl_i$, as the number of clauses cl_i in the CNF; C moreover contains the constraint $c : x \in \{1, 2\}$. P is consistent (just set $x = 2$). c is globally consistent iff the CNF is satisfiable.
The membership to NP can be easily proved assuming that the number of variables in $vars(c)$ is bounded: list the assignments $d \in D(vars(c))$ such that $d \models c$, use a NP-oracle to guess a d', and check $d' \in Sols(P)$ and that $d' \models d$. $\square$

Sketch of proof [Proposition 6 (validity of a description)] Simply reduce the problem of clausal inference, P being the CNF and c the clause the inference of which is to be shown. The membership to CO-NP obvious, considering the co-problem: let the NP-oracle guess a d, check that it satisfies P then that it does not satisfy c. $\square$

Sketch of proof [Proposition 7 (projection)] For any set of variables $Y \subseteq X$, c can be the condition $\bigwedge_{x \in Y}(x, D(x))$: this loose condition "selects" some variables without constraining their values. Hence, c is equivalent to $Sols(< X, D, C >)^{\downarrow vars(c)}$ iff, for any assignment of Y, there exist an assignment of $X \setminus Y$ that satisfies the CSP. The canonical problem of Π_2^P ($\forall Y_1, \exists Y_2 \Sigma$, Σ being CNF) can thus be polynomially reduced to the test of equivalence between the projection on Y_2 of a CSP representing the CNF and the condition $c = \bigwedge_{x \in Y_2}(x, \{true, false\})$. Proving the membership to Π_2^P is easy. The condition $d \in Sols(< X, D, C >)^{\downarrow vars(c)} \implies c(d) = \top$ can be falsified using one NP-oracle (guess d, check that $d \models P$ and that $d \nvDash c$). In order to falsify $c(d) = \top \implies d \in Sols(< X, D, C >)^{\downarrow vars(c)}$", let the oracle guess an assignment of $vars(c)$, check that $d \models c$, and that d cannot be extended to a solution of P (this sub problem belongs to CO-NP). $\square$

Sketch of proof [Proposition 10 (Concision, exhaustivity, exclusivity)] In order to prove the hardness of these requests, we use the following reduction of the SAT-CNF problem: X contains all the Propositional variables of the CNF and as many variable x_i as the number of clauses of the CNF ; the constraints in C are of the form $x_i \leftrightarrow cl_i$, where cl_i is a clause of the CNF; this CSP is obviously consistent, and testing the satisfiability of the original CSP is equivalent to testing the consistency of the CSP with the assignement $c' : x_1 = \top \wedge \cdots \wedge x_n = \top$. Let $C' = \{c'\}$: the CNF is satisfiable iff C' is concise. $C' = \{\neg c'\}$ the CNF is unsatisfiable iff C' is exhaustive. Let $c' : x_1 = \top$ and $c'' : x_2 = \top \wedge \cdots \wedge x_n = \top$.

The CNF is unsatisfiable iff $C' = \{c', c''\}$ is mutually exclusive. The membership to NP (for the problems of concision) or to CO-NP (for the exhaustivity and exclusivity requests) is obvious. $\square$

Sketch of proof [Proposition 12 (global consistency)] Consider a CNF. Build the CSP $P = < X, D, C >$ such that X contains all the variables of the CNF plus two additional boolean variables, say $x_\top$ and x_2 C contains as many constraints $x_\top = \top \wedge x_2 = \top \implies cl_i$ as the number of clauses cl_i in the CNF. Consider that $d(x_\top) = \top$ and $\forall x \in X \setminus \{x_\top\}, d(x) = *$. The problem is globally consistent and for any boolean variable $y \neq x_\top, x_2$, there is a solution d' such as $d'(y) = \top$ and $d' \models d$ and another such that $d'(y) = \bot$ and $d' \models d$ (just set $d'(x_2) = \bot$). So, the CSP is globally consistent w.r.t. d iff there is a solution d' such as $d'(x_2) = \top$, i.e. iff the CNF is satisfiable.
Membership to NP:for any x and any v, guess d' , check that $d' \in Sols(P)$ and $d' \models d$. $\square$

Sketch of proof [Proposition 13 (Incremental global consistency)] Consider the CSP of the previous proof. If there were a polynomial algorithm providing computing $< X, D', C >$ globally consistent with the assignement $d(x_1) = \top$, $d(x) = *, \forall x \neq x_1$, we would be able to decide in polytime whether the problem admits a solution d' such that $d'(x_2) = \top = d'(x_1)$, i.e. whether the CNF is satisfiable. Which does not hold unless $P = NP$. $\square$

Sketch of proof [Proposition 15(Conflits, restaurations) See [2] for the case $Choices$ being a set of fluents. The difficulty is not increased by $Choices$ being a set of consistent boolean expressions $\square$

Sketch of proof [Proposition 16 (Assessement of the price] The problem of minimization indeed encompasses the question of consistency of a CNF formula (letting d being the empty assignment and using the same transformation that for classical soft CSPs (satisfying a clause has a cost of 0, violating it a cost of 1 and $\alpha = 0$)). For the problem of maximization, the proof is similar: satisfying a clause has a cost of 0, violating it has a cost of -1 and $\alpha = 0$). The membership to NP is straightforward, even with negative costs. $\square$

Sketch of proof [Proposition 17 (Assessement of the Delay)]
Minimal Delay assessment: The problem is NP-hard when d does not assign any variables and each of the delay constraints is of the form $c_i \implies x_t \in [a_i, +\infty)$: we shall then encode a problem of optimization in a possibilitic CSP [16] (or a CNF of possibilistic logic), where the aim its to minimize the priority of the most important of the violated constraint: each constraint cl_i of priority α_i in the possibilistic base is translated into a delay constraint $\neg cl_i \implies x_t \in [\alpha_i, +\infty)$. The membership to NP is obvious.
Maximal Delay assessment: Reduction form the SAT-CNF problem: for any clause cl_i, define a constraint $\neg cl_i \implies x_t \leq 0$ (no other constraint in C, and d is the empty assignement. Then the maximal delay is greater than 0 iff there is a d' that does not satisfy any of the $\neg cl_i$ conditions, i.e. that satisfies all the clauses. The membership to NP is obvious. $\square$

---

[4] The proofs of membership as well as the proofs of polynomiality of the reductions are generally obvious, and thus skipped for the sake of brevity