Automatic Extraction of a Contradiction Genealogic Tree from Optimization with an Object-Oriented Simulator

Céline Conrardy, Roland Guio

HAL Id: hal-01567773

https://hal.inria.fr/hal-01567773

Submitted on 24 Jul 2017

Automatic extraction of a contradiction genealogic tree from optimization with an object-oriented simulator

Céline Conrardy[1], Roland de Guio[2]

[1] Lafarge Centre de Recherche, Saint-Quentin Fallavier, France
celine.conrardy@lafarge.com
[2] LGECO, INSA de Strasbourg, 24 bld. de la Victoire, 67000 Strasbourg, France
roland.deguio@insa-strasbourg.fr

Abstract. In order to go beyond optimization strategies in Computer Aided Innovation, it has been demonstrated that model changes are required [1,2] during Inventive Problem Solving Process (IPSP). TRIZ proposes a universal way of generating model changes thanks to contradiction statement and contradiction solving but it does not provide methods or tools for obtainting them from typical CAD or other kind of standard data. The aim of the following paper is to propose an algorithm which extracts from an object-oriented simulator a "genealogy" of contradictions systems (both physical and technical contradictions) and formulates corresponding Substance-Field models at the basis of TRIZ Inventive Standard application. This algorithm is fed by optimizations performed on various assemblages of objects constituting the simulator program. It helps disclosing contradictions that cannot be seen by domain experts due to high complexity of problem and is an additional step towards formalization and integration of TRIZ models.

Keywords: Optimization, Inventive design, Hilbert space, TRIZ, ARIZ, Contradiction system, Problem formulation

1   Introduction

Various authors have proposed enhancements of problem formulation in border of TRIZ. These methods are dedicated to improve effectiveness and efficiency of Inventive Problem Solving Process (IPSP). For concision purpose, it is not discussed in detail how those approaches contribute to that goal and what elements of the puzzle are still missing. It is at least useful to mention that

- some authors explore problem formulation and contradiction statement based on networks of non formalized data [3,4,5,6,7,8,9 and 10], whereas others have proposed to use mathematically formalized knowledge in order to disclose geometrical contradictions (a specific kind of physical contradictions) by using topological optimization algorithm[11] or disclose generalized contradictions by using CSP or design of experiments [12,13];
- a theoretical framework that enables comparison of such approaches still requires to be built (this article is a step towards building this framework).

1.1   Optimization in an infinite dimensional design space

Several facets of modeling activities and search strategies when using ARIZ 85-C [14] (hereafter named ARIZ), a supposed convergent IPSP developed in border of TRIZ have been studied in [15]. "Convergent" should be understood as the property of the algorithm to provide a set of successive partial solutions that satisfy step by step more and more requirements of the design problem, until all requirements are satisfied. Since the objects (i.e elements, properties, physical relations) that constitute the various partial solutions and the requirements are not entirely known at the beginning and are disclosed on the way, this study has been performed in an infinite dimensional space. It has been proposed that parameters disclosed to describe objects handled during ARIZ are preexisting dimensions of the infinite dimension space. Define and reformulate contradictions at several system levels is a cognitive pattern in that space. This article goes a step further in direction of a mathematical formalization of search strategies in such a space.

1.2   Motivations

The algorithm of contradiction genealogic tree extraction is proposed hereafter in order to:

- Provide means of complex problems analysis by studying interaction between unusual elements, when expert's knowledge is lacking;
- Disclose rapidly multi system-level problem statement;
- Provide quantitative means of choosing which contradiction of a contradiction system has to be solved in first part of ARIZ.

The following article is more particularly focussed on formalizing the interaction of three elements of TRIZ (system view, contradiction systems, Su-Field models). The contribution proposed may also help to understand in the future how other elements of TRIZ (not considered in this paper) interact with the three elements selected, when performing ARIZ.

With a more general perspective, mathematical formalization of ARIZ search strategy in an infinite dimension space may enable a combination with evolutionary computation [16,17 and 18] strategies for improving IPSP efficiency and effectiveness. To the knowledge of authors, no model, enabling to understand the convergence of ARIZ, have been proposed, although empirical results have shown ARIZ is an algorithm that converges towards solutions for a vast range of complex design problems. It is expected that elements of models developed for ARIZ study may be easily extended in order to depict other IPSP. It will so contribute to form a relevant meta-model of all IPSP.

1.3   Paper organization

The paper begins with some reminders about invention and optimization problems.

The second part of the paper describes the generic algorithm for extracting the contradiction genealogic tree with their associated Su-Field models.

The third part is devoted to application of this algorithm on a T shaped concrete beam example.

Last part is a discussion about the limitations of the approach, the contributions brought by the genealogic tree and the expected results that may be derived from it.

2   Optimization and invention problem models

2.1   From optimization problem…

Let us consider a computer simulator X. {P}={P[1], P[2], P[3],…} are input parameters of the simulator X. The simulator may enable these parameters to vary in a predefined range. When a value is given to each input parameter, we name this set of values a configuration of X. The simulator is constituted of objective and constraint functions. We have named objective function Evaluation Parameter and noted {EP} = {EP[0]}. Constraint functions are named Constraint Requirements and noted {CR}={CR[1], CR[2], CR[3], …}.

An optimization problem consists in finding the set of input parameter values that lead to the best value of objective functions while satisfying constraint functions. In the article, we restrain ourselves to mono-objective optimizations. During optimization, we are interested in the various quantitative values taken by evaluation parameter for different configurations in order to compare these configurations and in knowing about the satisfaction of constraint requirements. This is given hereafter by Boolean values, either satisfied (true) or not satisfied (false). However, for computation purpose hereafter, we may also refer to a numerical value to measure variations of the distance to the threshold delimiting satisfied or not satisfied constraints.

An optimization result will be described with the following notations:

— P_0 are the values of {P} that optimize EP[1] while keeping all constraints {CR} satisfied. P_0 is the result of this optimization problem and is a particular configuration of X;

— $P_1, P_2, ..., P_i, ...$ are configurations obtained by optimizing parameters {P} to improve EP[1] when the ith constraint CR[i] is relaxed. The result of such an optimization problem will either lead to break the constraint, i.e. $CR[i](P_i) = false$ or to satisfy the constraint, i.e. $CR[i](P_i) = true$ if the constraint had no influence on P_0. In the article, we restrain ourselves to mono-constraint relaxation.

2.2   …To invention problem

We may then consider:

— (P_0, P_i) is the couple formed by the two configurations P_0 and P_i;

— if $P_0 \neq P_i$, {P} is known as the action parameter, P_0 and P_i as the couple of opposite values of this action parameter, EP[0] and CR[i] as respectively the evaluation parameters and constraint requirement of a contradiction system noted $CS[0; i]$. Those elements are depicted on Fig.1. NB: In the article, we distinguish between evaluation parameters and constraint requirements. The two of them may be known indifferently as evaluation parameters in TRIZ literature [2] and this distinction is introduced here for convenience purpose. The contradictions systems considered hereafter will always involve an evaluation parameter and a constraint requirement. The restriction to this particular type of contradiction systems will be discussed in section 4.
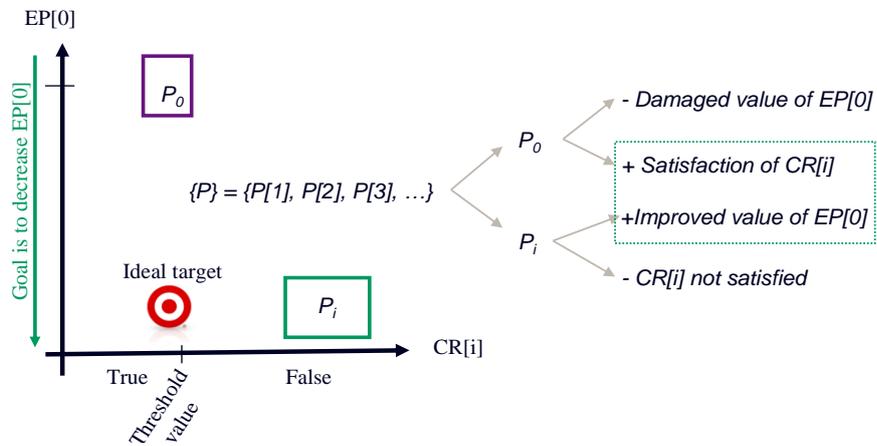

Fig. 1. Contradiction System CS[0;i]

In border of TRIZ, inventive problem solving consists in finding means of obtaining satisfaction of constraint and improved value of EP[0], without generating new problems in the system. ARIZ, for instance, is a cognitive algorithm to perform such a task in a more or less controlled manner. For our concern, the important thing is that this algorithm uses three TRIZ models in particular:

— contradiction system, which link technical to physical contradiction
— system view
— Su-Field models which depict the nature of interaction between elements involved in a contradiction system.

For details about these models, insight about convergence control during ARIZ IPSP, please refer to [19].

2.3   Reformulate various invention problems linked with the former one

We may view any configuration P as an assemblage of components (known as sub-systems in border of TRIZ). For the simulator X, which computes the behaviour of any configuration, this decomposition of each configuration may be linked to an

object-oriented way of programming the functions of X. Details concerning common points and differences between those concepts coming from optimization, software science and TRIZ background are not discussed in this article but are to be found in an upcoming paper. For clarity purpose, we remind hereafter few basic and simplified elements concerning the structure of an object oriented program that will be a key resource in the proposed contradiction tree extraction algorithm.

In such software architecture, the simulator is decomposed into various components known as class of objects. An object is built by providing particular values to the arguments of a class. On Fig.2, the simulator X is depicted as a graph of classes O[m] (with boxes around). Each class defines the characteristics of all the objects (noted hereafter O_m) it enables to build. Construction of objects of each class requires providing as argument:

- Objects of the classes depicted at next level and connected with arrow (in boxes)
- Input parameters depicted at next level and connected with a line (circled parameter).

The class also provides to any of its object additional functions that depict things the object can do. These functions are noted O[m].F[k] and depicted on Fig.2 with a triangle around. They will be used hereafter as partial evaluation parameters and constraint requirements in the decomposition of invention problem. The function O[m].F[k] of the class O[m] takes as argument either input parameters of O[m] (like P[i] on Fig.2 for instance), or results returned by functions of objects passed as arguments of O[m] (result returned by O[f].F[l] on Fig.2 for instance). We will note $O_m.F_k$ the result returned by O[m].F[k].
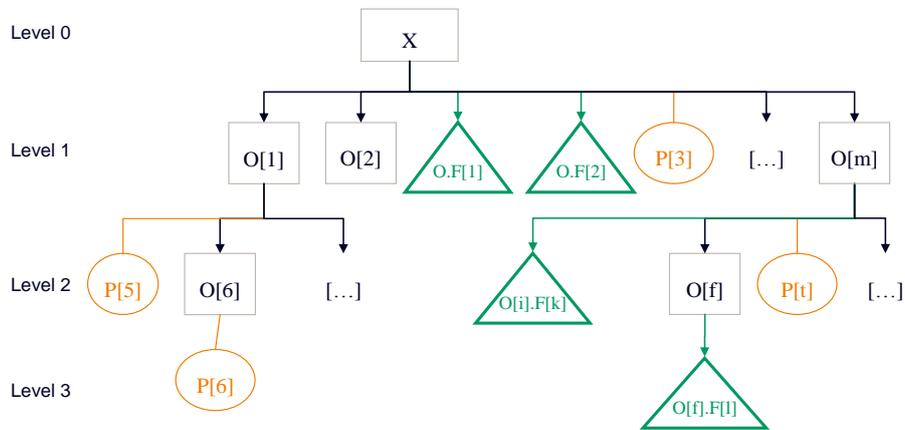

Fig. 2. Structure of an object-oriented simulation program. Boxes are class of objects; circles are parameters; triangles are functions linking parameters and functions of other classes of the considered level.

The object-oriented program on Fig.2 will then take the following generic form:

Definition of classes and their instances
[…]

```
Class Object_i
# constructor
        Def_init_(Of, … other objects,  Pt,… other parameters)
        [...] # provide the values of the arguments to parameters of object

# definition of the characteristics of the class (properties and methods)
        Fk = a formula that may involve Of, Pt, etc…
        […]   # define of other functions
```
```
# Main part of the program
        Of = Object_f()
        Oi = Object_i(Of, value of Pt, …)
        O6 = Object_6(value of P7)
        O3 = Object_3()
        O2 = Object_2(value of P5, O6, …)
        O0 = Object_0(O2, O3, P4, …, Oi)
```

This decomposition of a system into components is useful to reformulate the invention problem at those levels. At each level of the decomposition, there may be (or not) inventive problems that have impact on the inventive problem at previous level of decomposition. A way to reformulate contradictions at next or previous level has been proposed in [15]. Such a reformulation is useful when one tries to control which part of the system will change and which part will remain as it is during ARIZ IPSP. This mini-algorithm of contradiction statement formulation and reformulation will be described in a more formalized manner in the next section. The algorithm proposed also goes a step further since it enables the automation of Su-Field models construction for each contradiction system disclosed.

3   The contradiction genealogic tree extraction algorithm

3.1   Definitions and notations

The contradiction genealogic tree is obtained thanks to a sequence of objects generated by solving various optimization problems.

$P_{0.m}$ is the configuration obtained by solving the following optimization problem:

— try to improve input parameters leading to O[m] variation (i.e arguments of O[m], for instance P[t] or parameters of objects built by classes at higher level of decomposition, arguments of O[f] for instance) to improve EP[0]. We will shortly say that component O_m of $P_{0.m}$ was mobile during optimization;

— while keeping values of other input parameters of X constant during the computation, we will then say that other components of $P_{0.m}$ were fixed during optimization;

— relax CR[i].

By symmetry, $P_{i.m}$ is obtained by optimizing parameters of O[m] to improve EP[0], while satisfying CR[i] (and all other constraints) and keeping other components of P_i constant. $(P_0 , P_{0.m})$ and $(P_{i.m}, P_i)$ form then respectively the two contradiction systems CS[0;0.m] and CS[i.m;i];

— EP[0;i]=EP[0] and CR[0;i]=CR[i] are the evaluation parameter and the constraint requirement of CS[0;0.m] and CS[i.m;i]
— CS[0;i] is then the parent of CS[0;0.m], which is the child of CS[0;i] in the contradiction tree (Fig.4);
— Su-field models attached to the contradiction system CS[0;0.m] form a list noted SF[0;0.m] and depict the nature of unsatisfying relationships between mobile component O_m and other components of P_0 remained fixed during optimization. How to built Su-Field models will be detailed in the next section.
— These other components of P_0 are named adjacent components of O_m.

The notations above remain valid when the indexes "0" and "i" are replaced by combination of index obtained when following the algorithm like $0.[\ldots].m \neq 0$ or $i.[\ldots] \neq i$. If CS[i;j] is the parent of CS[i;i.m], CS[i;i.m] takes the generic form depicted on Fig.3. By extension, we will also consider input parameters as components, following the same computation process described above, i.e. O[m] may be substituted by P[4] without any changes in the previously defined notations.

The goal of the algorithm is to extract a contradiction genealogic tree as depicted on Fig.4. Each node of the genealogic tree is a contradiction system to which is bound Su-Field models. The algorithm acting upon the object-oriented simulator consists of a succession (with loops) of elementary steps detailed in Table 1. An example of application is proposed in part 4.
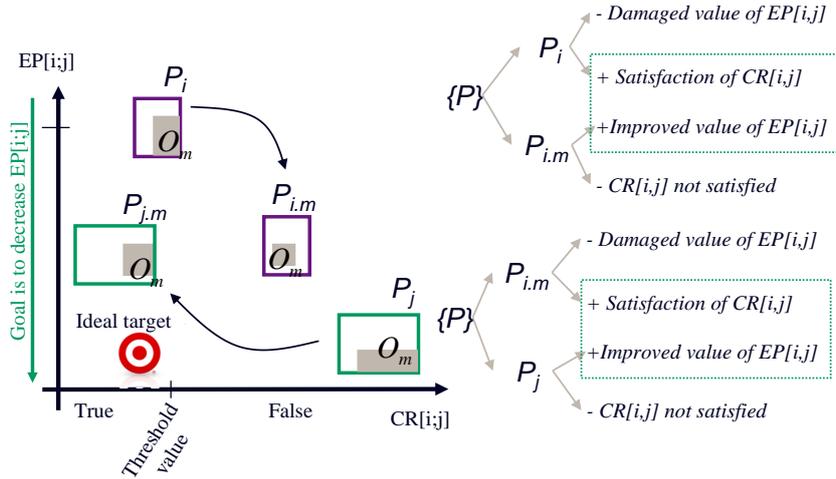

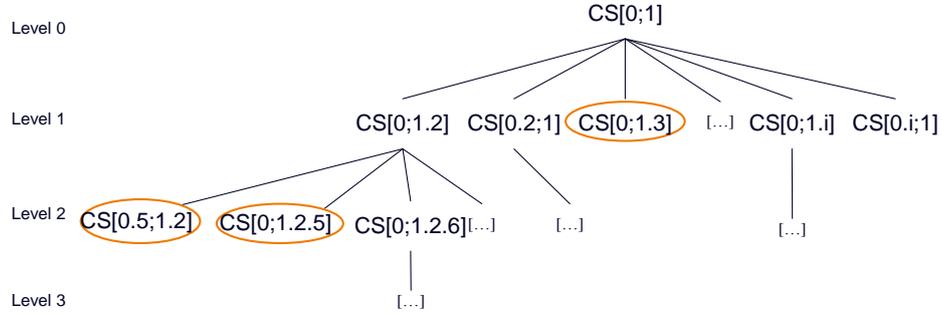**Fig. 3.** The two contradiction systems CS[i;j.m] and CS[i.m;j]]

Fig. 4. Partial contradiction genealogic tree related to decomposition depicted on Fig.2. The branch CS[0;1] is partially developed. Circled contradictions systems are leaves of the tree, which means that their physical contradiction involves a single input parameter.

3.2   Elementary steps of the algorithm

The algorithm proposed hereafter is a series of steps described in Table 1 and are repeated until reaching the last level of decomposition.

Table 1. Elementary steps of the algorithm

N°	Function	Details on how to perform the step
1	Obtain a first set of technical contradictions	Optimize {P} in order to improve EP[0] while relaxing the constraint CR[i]. Constraints are to be relaxed one by one.
2	Identify contradiction system CS[i;j.m] that is a child of CS[i;j], $\forall O_m$ a component of Pi.	Optimize parameters of O[m] to improve EP[i;j], while keeping adjacent components to the values they have in Pi and relaxing CR[i;j]. O_m is the sole varying component of Pi during optimization. If a component is shared between varying and non varying components, it should not vary during the optimization. If $P_{i.m} = P_i$, there is no contradiction.
3	Identify the contradiction system CS[i.m;j] that is a child of CS[i;j], $\forall O_m$ a component of Pj.	Optimize parameters of O[m] to satisfy CR[i;j], while keeping other component fixed and improving EP[i;j]. If a component is shared among varying and non varying components, it should not vary during the optimization. If $X_{i.m} = X_i$ there is no contradiction.
4	State Su-Field models of CS[i;j.m].	The nature of relationships between O_m and its various adjacent components O_u in the configuration P_i (resp. $P_{j.m}$) is disclosed by examination of EP[i;j.m] (resp. CR[i;j.m]) variations. Variations in EP[i;j.m] (resp. CR[i;j.m]) = f(O[m].F[1],O[m].F[k],O[m].F[l], …) take the following generic form:

$$\Delta EP[i;\ j.m] \atop (resp.\Delta CR[i;\ j.m]) = \left| \begin{array}{c} \dfrac{\partial f}{\partial O[m].F[1]}\Delta O[m].F[1] \\ ... \\ \dfrac{\partial f}{\partial O[m].F[k]}\Delta O[m].F[k] \\ ... \\ \dfrac{\partial f}{\partial O[m].F[l]}\Delta O[m].F[l] \end{array} \right.$$
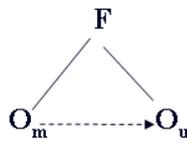
For each variation function, the analysis proposed in table2 has to be performed.

5	State Su-Field models of CS[i.m;j]	The nature of relationships between O_m and its various adjacent components O_u is disclosed by examination of EP[i.m;j] (resp. CR[i.m;j]) variations, following the same method than step 4 above.
6	Create new nodes in contradiction genealogic tree	Insert CS[i.m;j], CS[i;j.m] and their associated Su-Field models as child of CS[i;j] in contradiction genealogic tree.
7	Disclose EP[i.m;j] (resp. EP[i;j.m]) and CR[i.m;j] (resp.CR[i:j.m]) of CS[i.m;j]	These functions are obtained by replacing parameters of P_i (resp. P_j) that remained constant during optimization (i.e. input parameters involving adjacent components of O_m) by their numerical values in EP[i;j] and CR[i;j]. This operation provides new functions (EP[i.m;j] and CS[i;j.m]) that are themselves two functions of the functions O[m].F[k], $(\forall k)$ that belong to O[m].

Table 2. Interpretation of mathematical relations leading to SF[i;j.m]

| Su-Field Model	Variations to be examined in Evaluation Parameter and Constraint Requirement. EP and CR may be handled in the same way. For concision purpose, only EP analysis has been developed.
	If $\Delta EP[i;\ j.m]\big|_{P_i \to P_{j.m}} \times \dfrac{\partial f}{\partial O[m].F[k]}(P_i) > 0$, then $O_m.F_k$ and so O_m has an insufficient action on all the adjacent components O_u involved in the expression of $\dfrac{\partial f}{\partial O[m].F[k]}$. A Su-Field is drawn for each O_u. NB: if several components are involved, it may be considered in a first approach that O_m has an action on the relationship between these objects. This point still requires to be clarified.

If $\Delta EP[i; j.m]\big\|_{P_i \to P_{j.m}} \times \dfrac{\partial f}{\partial O[m].F[k]}(P_i) < 0$

and $\dfrac{\partial f}{\partial O[m].F[k]}$ is a constant function, O_m has a negative

effect on itself.

If $\Delta EP[i; j.m]\big\|_{P_i \to P_{j.m}} \times \dfrac{\partial f}{\partial O[m].F[k]}(P_i) < 0$,
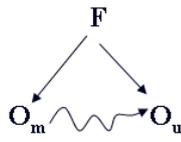
and O[m].F[k] can be null in EP[i;j.m], then O_m has an harmful action on all the adjacent components involved in the expression of

$\dfrac{\partial f}{\partial O[m].F[k]}$. A Su-Field is drawn for each O_u .

NB: if several components are involved, it may be considered that O_m has an action on the relationship between these components. This point still requires to be clarified.

If $\Delta EP[i; j.m]\big\|_{P_i \to P_{j.m}} \times \dfrac{\partial f}{\partial O[m].F[k]}(P_i) < 0$,
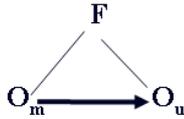
and $O_m.F_k$ cannot be null in EP[i;j.m] (division for instance), then O_m has an excessive action on all the adjacent components involved

in expression of $\dfrac{\partial f}{\partial O[m].F[k]}$. A Su-Field is drawn for each O_u .
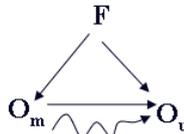
If the analysis of EP[i;j.m] (resp. CR[i;j.m]) leads to extraction of an harmful and an insufficient action between O_m and the same other components, it means there is both useful and harmful relationships between them and the standards are transformed into this last category of standard.

3.3   The loops to be performed

The steps of algorithm have to be performed in the following order:

```
For each constraint requirements i
    Perform step 1
    Level = top-level
    node=O₀
      For each level
        For each node of the level
          For each component of the node
            Perform steps 2,3,4,5,6,7
            node=next node at same level
```

```
level=next level
node=first node of the level
```

Last nodes of the genealogic tree are leaves, the physical contradictions of which involve a single input parameter.

4   Application of the algorithm on an example

4.1   Optimization problem on T shaped beam example

The starting beam from which will be extracted the contradiction tree is the result of the following optimization problem (see Fig.5):

  — {P}=(b,h,As,R) are the input parameters of the simulator;
  — 5 constraints on geometry of the beam and mechanical resistance should be satisfied in order to obtain feasible solutions. Those constraints are either satisfied (true) or not satisfied (false);
  — the cost of the beam is the evaluation parameter to be optimized;
  — due to constraints of the environment (insertion of the wings in adjacent concrete slabs for example), c, hw and bw are fixed properties and will not appear hereafter.

The optimized beam is found for a particular set of b, h, As, R values, so that all constraints are satisfied while the cost is minimal. Decrease the cost even more constitutes the starting administrative contradiction of the problem.

$$EP[0] = beam.\cos t()$$
$$c = 0.09m$$
$$hw = 0.2m$$
$$bw = 0.4m$$
$$CR[1] = (\sigma < \sigma \max)$$
$$CR[2] = (h\min(= 0.2m) < h < h\max(= 1m))$$
$$CR[3] = (b\min(= 0.2m) < b < b\max(= 1m))$$
$$CR[4] = (R\min(= 10MPa) < R < R\max(= 90MPa))$$
$$CR[5] = (2.h.b < As(cm2) < 40.h.b)$$

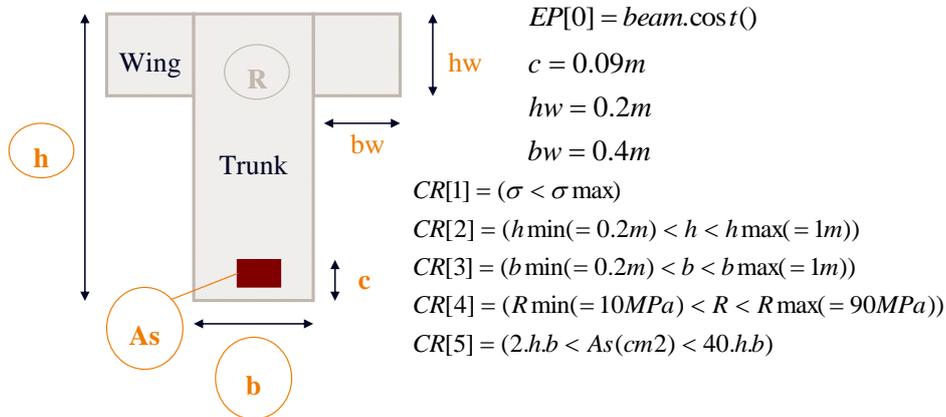**Fig. 5.** Parametrization of T-shaped beam, Evaluation Parameters, Constraint Requirements and values of parameters that will remain fixed during the whole process.

The various optimizations are performed thanks to a free library EASEA [20] and a proprietary interface library. This library has been developed for linking automatically any kind of simulators and building any kind of optimization problem on EASEA

standard interface, which enables to construct and solve automatically the various optimization problems handled through the presented generic algorithm.

4.2   Description of the simulator of T shaped beams used as example

Warning. Some details concerning the real mathematical equations and computation results are omitted or simplified for concision purpose. Rough numerical values are given in order to clarify examples.

The simulator, provided by courtesy of Lafarge, computes the behaviour of the beam depicted on Fig.5.
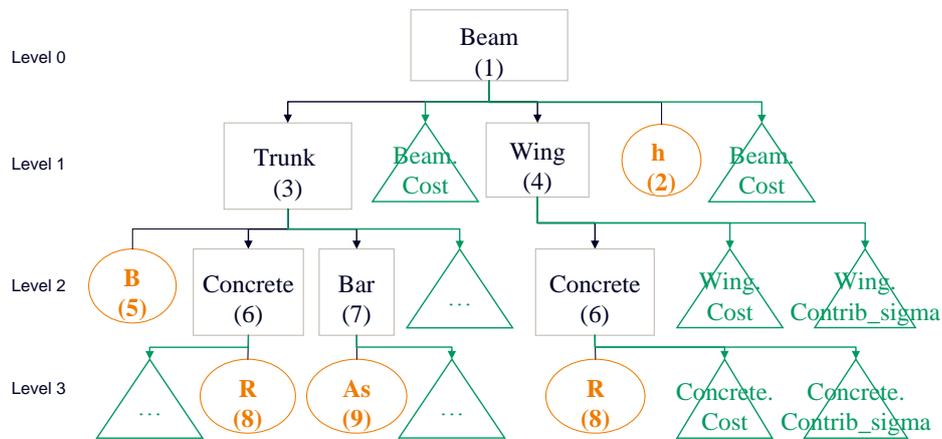
Fig. 6. Structure of the object oriented simulator of T-shaped beam, with functions (triangle), components (box) and input parameters (circle). Numbers provided below each object name are used for easy reference purpose. . Trunk and Wing share the same component Concrete.

The structure of the simulator is an object oriented program (see Fig.6). The simplified code is reproduced below.

```
# Definition of classes and their instances
    Class Concrete
            Def_init_(R)
                    cost = R*100.
                    contrib_sigma = 1/R
    Class Bar
            Def_init_(As)
                    cost = As*900.
                    contrib_sigma = 1/ (As * 600)
    Class Trunk
            Def_init_(b, Concrete, Bar)
                    cost_concrete = 20*b*Concrete.cost
                    cost_steel = Bar.cost
```

```
                              contrib_sigma = Bar.contrib_sigma − Bar.contrib_sigma^2
                              …/ (b*Concrete.contrib _sigma)
           Class Wing
                  Def_init_(Concrete)
                              bw = 0.3
                              hw = 0.2
                              contrib _sigma = Concrete.contrib _sigma *…
                                      hw * sqrt(bw)
                              cost = bw*hw*Concrete.cost
           Class Beam
                  Def_init_(h, Trunk, Wing)
                              bt = 0.7
                              c = 0.9
                              l = 8.1
                              cost = h*Trunk.cost_concrete + Trunk.cost_steel +…
                                      2*Wing.cost
                              sigma = (1/h^2) * Trunk.contrib _sigma +…
                                      2* Wing.contrib _sigma


   # Main part of the program
        R=10
        As=2.1
        b=1
        h=1
        Concrete = Concrete(R)
        Bar = Bar (As)
        Trunk = Trunk (b, Concrete, Bar)
        Wing = Wing (Concrete)
        Beam = (h, Trunk, Wing)
```

4.3   The algorithm for contradiction genealogic tree extraction

Warning. Drawings of optimization results are given with an explanatory purpose only and not as outcome of a real computation.

Let us relax one by one each constraint. Single constraint relaxation may lead (or not) to technical contradiction. In the example, relaxing constraint on hmin for instance has no impact on optimization and so does not lead to a contradiction. However, Fig.7 shows some optimization results P1 and P2 when relaxing constraints $\sigma \max$ and $h \max$ respectively. (P0, P1) and (P0, P2) form respectively the contradiction systems CS[0;1] and CS[0;2] at the level 0 of the genealogic tree (see Fig.16).

Let us focus on CS[0;1] in order to identify physical contradictions at level n°1 of the tree. We may first vary parameter h which is the component n°2 in the object-oriented decomposition (Fig.6). Fig.8 shows contradiction system CS[0;0.2] obtained by using h as a varying component. CS[1.2;1] does not exist since the best value of h does not enable P1.2 to satisfy constraint requirement n°1 on $\sigma \max$.
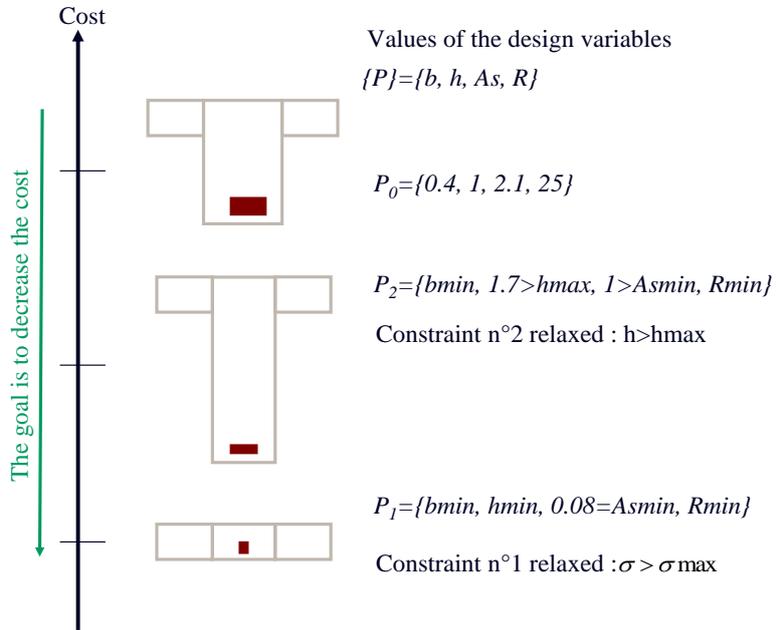
Fig. 7. Results of optimization obtained when relaxing constraints one constraint. P1 is obtained by relaxing $\sigma \max$ and P2 is obtained by $h \max$ relaxation.
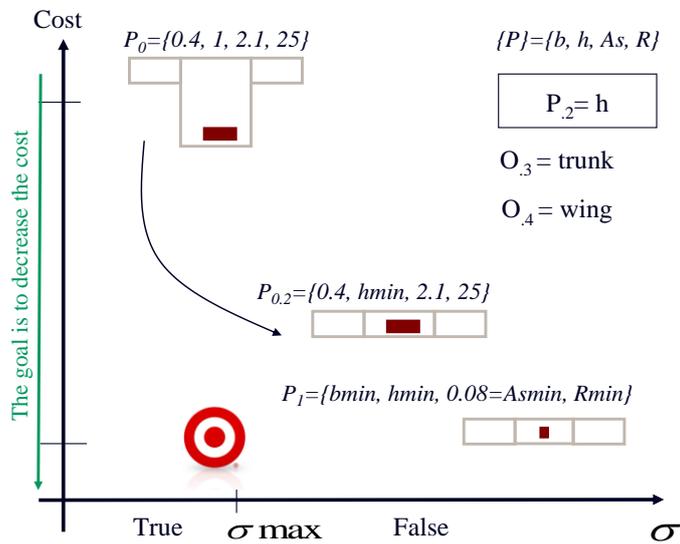

Fig. 8. Configurations P0 and P0.2 involved in CS[0;0.2]. Configuration P0.2 is obtained by optimizing h in order to improve P0 on EP[0;1] while keeping other components of P0 fixed.

Since EP[0]=Beam.cost = h*Trunk.cost_concrete + Trunk.cost_steel + 2*Wing.cost, Δ cost = Trunk.cost_concrete*Δ h and Δ cost(P0->P0.2) * Trunk.cost_concrete(P0) <0, h has an harmful impact on the trunk. The height increases indeed the cost of concrete in trunk, hence the harmful impact on the trunk in Fig.15. Moreover, CR[1]=sigma = (1/h^2)* Trunk.contrib_sigma + 2 * Wing.contrib_sigma, Δ sigma = - (1/h^3) * Trunk.contrib_sigma *Δ h and Δ sigma(P0.2->P0) * (- 1/h^3) * Trunk.contrib_sigma >0. The lever length effect on the trunk has indeed a too weak effect on trunk to decrease σ, hence the Su-Field proposed in Fig.9. A new contradiction node can be added in the contradiction genealogic tree (Fig.10). Since CS[0;0.2] is a leave of the tree, the evaluation parameters and constraint requirements functions at sub-level are not evaluated.

P0={0.4, 1, 2.1, 25}	P0.2={0.4, hmin, 2.1, 25}
Sigma : true but Cost : high	Cost : low but Sigma : false
F : h*Trunk.cost_concrete	**F** : (1/h^2) * Trunk.contrib_sigma
P.2 ~~~> **S**.3	**P**.2 - - - - -> **S**.3
h (high)	h (low)

Fig. 9. CS[0;0.2] and associated Su-Field models.
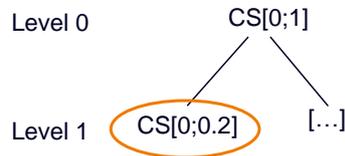
Level 0   CS[0;1]

Level 1   CS[0;0.2]   [...]

Fig. 10. CS[0;0.2] is a child of CS[0;1] in contradiction genealogic tree.

Let us now vary the parameters of the trunk, the component n°3 in the object-oriented decomposition (Fig.8). Fig.11 shows contradiction systems CS[0;0.3] and CS[1.3;1] obtained by using trunk as a varying component.

Since EP[0] = h*Trunk.cost_concrete + Trunk.cost_steel + 2*Wing.cost,

$$\Delta \cos t = \begin{cases} h * \Delta Trunk.\cos t\_concrete \\ \Delta Trunk.\cos t\_steel \end{cases}.$$

Δ cost(P0->P0.3)*h(P0)<0, so the trunk has an harmful relationship with h which leads to increase the cost of the beam, hence the harmful impact on height in Fig.12. This relationship is in fact a geometrical relationship. Δ cost(P0->P0.3)*1<0, so the trunk has also an harmful impact on itself (Fig.12). It is due to its own cost (due to steel bar inside). Since CR[1]=sigma = (1/h^2) * Trunk.contrib_sigma + 2 * Wing.contrib_sigma, Δ sigma = (1/h^2) * Δ Trunk.contrib_sigma and Δ sigma(P0.3->P0)* (1/h^2) <0. The trunk has a harmful relationship with the height, which leads to

increase the stress that the material constituting the beam has to endure, hence the Su-Field proposed in Fig.12. The operations for CS[1.3;1] are the same, only the numerical values change. The two new contradiction nodes can be added in the contradiction genealogic tree.
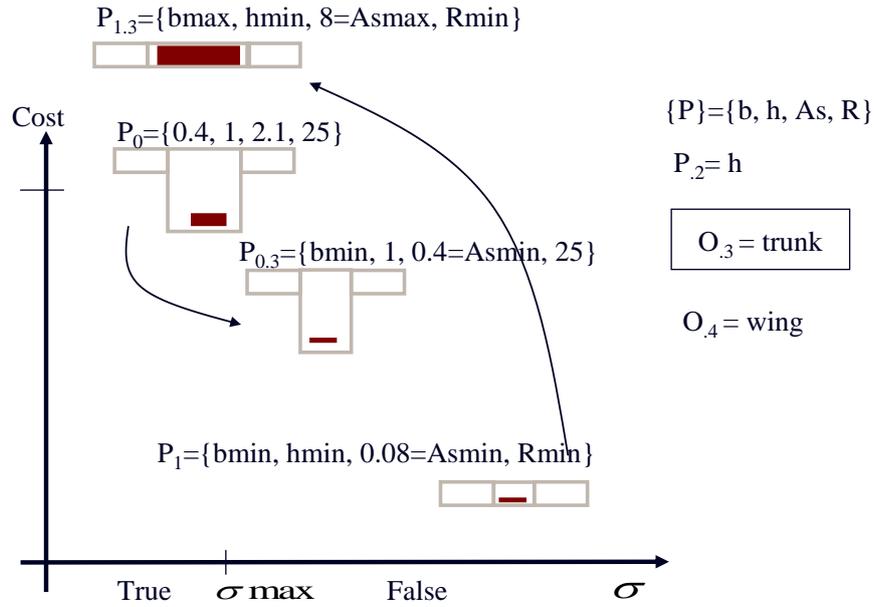
$P_{1.3}$={bmax, hmin, 8=Asmax, Rmin}

P_0={0.4, 1, 2.1, 25}

Cost

$P_{0.3}$={bmin, 1, 0.4=Asmin, 25}

{P}={b, h, As, R}

$P_{.2}$= h

$O_{.3}$ = trunk

$O_{.4}$ = wing

P_1={bmin, hmin, 0.08=Asmin, Rmin}

True σ max   False σ

Fig. 11. Configurations P0 and P0.3 involved in CS[0;0.3] and configurations P1.3 and P1 involved in CS[1;3;1]. Configuration P0.3 is obtained by optimizing the trunk while keeping other components of P0 fixed in order to improve P0 on EP[0]. Configuration P1.3 is obtained by optimizing the trunk while keeping other components of P1 fixed in order to satisfy CR[1].

We may now evaluate the evaluation parameter and the constraints requirements of CS[0;0.3] and CS[1.3;1] children. Since EP[0] = Beam.cost= h*Trunk.cost_concrete + Trunk.cost_steel + 2*Wing.cost = h* b * Concrete.cost + Bar.cost + 2* bw* hw* R* 100, EP[0;0.3] = b * Concrete.cost + Bar.cost + 400. and EP[1.3;1] = b * Concrete.cost + Bar.cost + 160.

Symmetrically, CR[1] = $(1/h^2)$* Trunk.contrib _sigma + 2* Wing.contrib _sigma = $(1/h^2)$* (Bar.contrib_sigma – Bar.contrib_sigma^2/(b*Concrete.contrib_sigma))+ 2* 1/R * hw * sqrt(bw). So CR[0;0.3]= Bar.contrib_sigma + Bar.contrib_sigma^2/ (b* Concrete.contrib_sigma) + 0.010 and CR[1.3;1]= Bar.contrib_sigma + Bar.contrib_sigma^2/(b*Concrete.contrib_sigma) + 0.025.

P0={**0.4**, 1, **2.1**, 25}	P0.3={**bmin**, 1, **0.4=Asmin**, 25}
Sigma : true but Cost : high	Cost : low but Sigma : false
F : h*Trunk.cost_concrete **S**.3 ⁓⁓ **P**.2 **S**.3 ⁓ **F :** Trunk.cost_steel	**F** : (1/h^2) * Trunk.contrib_sigma **S**.3 ⁓⁓ **P**.2
P1.3={**bmax**, hmin, **8=Asmax**, Rmin}	P1={**bmin**, hmin, **0.08=Asmin**, Rmin}
Sigma : true but Cost : high	Cost : low but Sigma : false
F : h*Trunk.cost_concrete **S**.3 ⁓⁓ **P**.2 **S**.3 ⁓ **F:** Trunk.cost_steel	**F** : (1/h^2) * Trunk.contrib_sigma **S**.3 ⁓⁓ **P**.2

Fig. 12. CS[0;0.3] and CS[1.3;1] and associated Su-Field models. The bold parameters are the parameters varying during optimization. R is assumed to vary but cannot because it is also a parameter of the wing which is fixed.
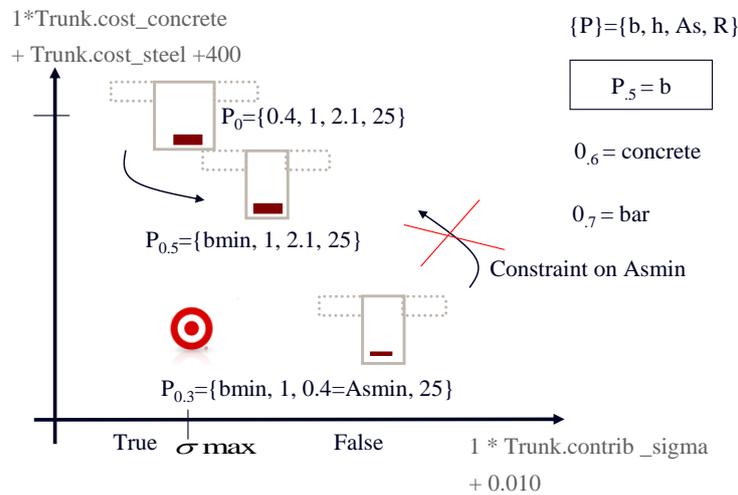

Fig. 13. Configurations P0 and P0.5 involved in CS[0;0.5]. Configuration P0.5 is obtained by optimizing b in order to improve P0 on EP[0;0.3] while keeping other components components of P0 fixed.

Let us now focus on CS[0;0.3] in order to identify physical contradictions at level n°2 of the tree. We may first vary parameter b which is the component n°5 in the

object oriented decomposition (Fig.6). Fig.13 shows contradiction system CS[0;0.5] obtained by using b as a varying component. CS[0.3.5;0.3] does not exist since there is no b value that enables P0.3.5 to satisfy constraint requirement n°5 on Asmin.

   Since EP[0;0.3] = b * Concrete.cost + Bar.cost + 400. we have Δ EP[0;0.3] = Δ b * Concrete.cost and Δ EP[0;0.3] (P0->P0.5)* Concrete.cost(P0) <0. b has an harmful impact on the concrete due to a geometrical effect that increases the volume of concrete, hence the harmful impact on the concrete modeled in Fig.21. Since CR[0;0.3] = Bar.contrib_sigma + Bar.contrib_sigma^2/ (b*Concrete.contrib_sigma) + 0.010,   we   have Δ CR[0;0.3](P0.5->P0)*   (-Bar.contrib_sigma^2/   (b^2* Concrete.contrib_sigma)>0. b has an insufficient action on the relation between bar and concrete. It is indeed not high enough for elevating the position of neutral axis separating the part of the beam that endures a tensile stress (bottom) and the part that endures a compressive stress (top of the beam), hence the Su-Field proposed in Fig.21. A new contradiction node can be added in the contradiction genealogic tree (Fig.22). Since CS[0;0.5] is a leave of the tree, the evaluation parameters and constraint requirements functions at sub-level are not evaluated.
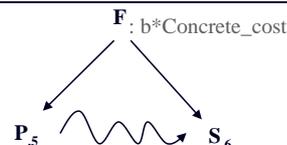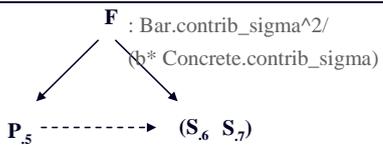
P0={**0.4**, 1, 2.1, 25}	P0.5={**bmin**, 1, 2.1, 25}
CR[0;0.3](X0) : true but EP[0;0.3](X0) : high	EP[0;0.3](X0.5) : low but CR[0;0.3](X0.5) : false
F: b*Concrete_cost $\quad$ P$_{.5}$ $\leadsto$ S$_{.6}$	**F** : Bar.contrib_sigma^2/ (b* Concrete.contrib_sigma) $\quad$ P$_{.5}$ - - - -> (S$_{.6}$ S$_{.7}$)

Fig. 14. CS[0;0.5] and associated Su-Field models.
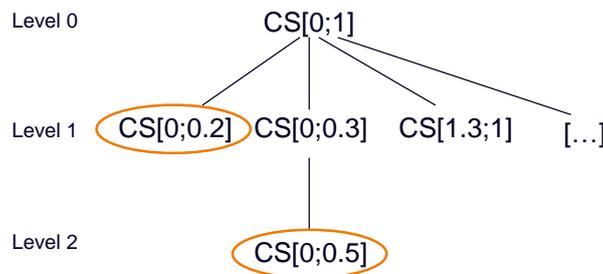

Fig. 15. CS[0;0.5] is a child of CS[0;0.3] in contradiction genealogic tree.

   Let us focus on CS[1.3;1] in order to identify next physical contradictions at level n°2 of the tree. We may first vary the Bar, which is the component n°7 in the object oriented decomposition (Fig.6). Fig.16 shows contradiction system CS[1.3;1.3.7] obtained by using the bar as a varying component.
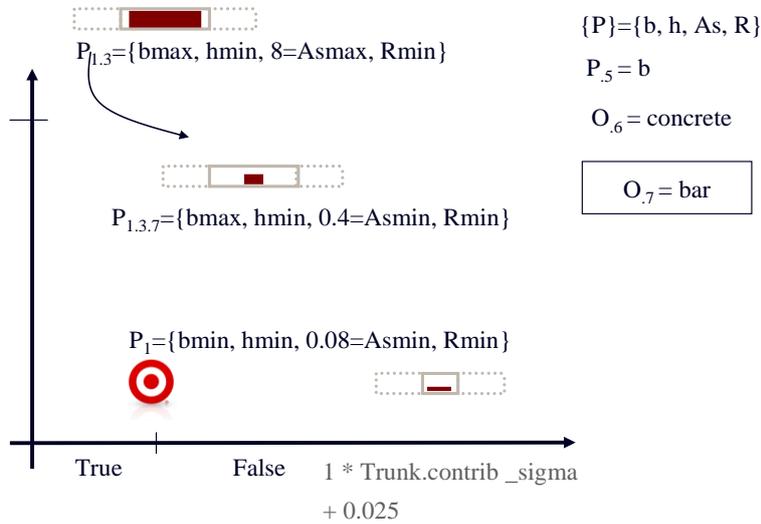
$P_{l.3}$={bmax, hmin, 8=Asmax, Rmin}

$P_{1.3.7}$={bmax, hmin, 0.4=Asmin, Rmin}

P_1={bmin, hmin, 0.08=Asmin, Rmin}

{P}={b, h, As, R}

$P_{.5}$ = b

$O_{.6}$ = concrete

$O_{.7}$ = bar

True   False   1 * Trunk.contrib _sigma

+ 0.025

Fig. 16. Configurations P1.3 and P1.3.7 involved in CS[1.3;1.3.7]. Configuration P1.3.7 is obtained by optimizing the bar in order to improve P1.3 on EP[1.3;1] while keeping other components of P1.3 fixed.

P1.3={bmax, hmin, **8=Asmax**, Rmin}	P1.3.7={bmax, hmin, **0.4=Asmin**, Rmin}
CR[1.3;1](X1.3) : true, but EP[1.3;1](X1.3) : high	EP[1.3;1](X1.3.7) : low but CR[1.3;1](X1.3.7) : false
$\mathbf{F}$: Bar.cost $\mathbf{S_{.7}}$	$\mathbf{F}$: Bar.contrib_sigma $\mathbf{F}$: Bar.contrib_sigma^2/ (b* Concrete.contrib_sigma) $\mathbf{S_{.7}}$ $(\mathbf{S_{.6}}\,\mathbf{P_{.5}})$

Fig. 17. CS[1.3;1.3.7] and associated Su-Field models.

Level 0   CS[0;1]

Level 1   CS[0;0.2] CS[0;0.3] CS[1.3;1]   […]

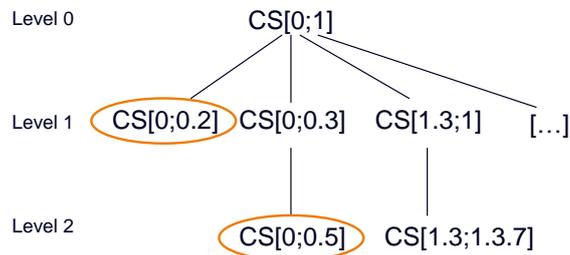Level 2   CS[0;0.5]   CS[1.3;1.3.7]

Fig. 18. CS[1.3;1.3.7] is a child of CS[1.3;1] in contradiction genealogic tree.

Since EP[1.3;1] = b * Concrete.cost + Bar.cost + 160., we have Δ EP[1;1.3] = Δ b *Concrete.cost and Δ EP[0;0.3] (P1.3->P1.3.7)* 1 < 0. The bar has a harmful impact on itself, hence the Su-Field in Fig.17. The steel that constitutes the bar has indeed a high cost. Since CR[1.3;1] = Bar.contrib_sigma + Bar.contrib_sigma^2 / (b* Concrete.contrib_sigma) + 0.025., we have

$$\Delta CR[1.3;1] = \begin{cases} \Delta Trunk.\cos t \_ concrete \\ 2 * Bar.contrib \_ sigma / (b * Concrete.contrib \_ sigma) \end{cases}.$$

CR[1.3;1](P1.3.7->P1.3)*1 < 0 means that the bar has a harmful action on itself in configuration P1.3.7 since it increases the contribution it has to provide to the beam in order to help resist to the stress. Δ CR[1.3;1](P1.3.7->P1.3)* 2 Bar.contrib_sigma / (b* Concrete.contrib_sigma)< 0. The bar has a harmful action on the relation between concrete and b, hence the Su-Field proposed in Fig.17. It means the configuration of the bar in P1.3.7 tends to oblige b and concrete to increase their contribution to resistance. A new contradiction node can be added in the contradiction genealogic tree (Fig.18).

The algorithm may be continued until all leaves are reached.

5   Discussion and conclusion

The approach of formalization proposed in this article provides insight about the crucial role of components organization when indentifying contradictions consistent with Su-Field modelling and reformulating them. A particular extraction algorithm of those contradictions has been detailed. However, many other TRIZ elements still require to be better formalized in order to obtain a complete framework to define and study convergence of ARIZ and other IPSP.

5.1   What are the possible extensions of the genealogic contradiction tree extraction algorithm presented here-above?

Table 3. Summary of limitations and opportunities of improvement of the algorithm

Limitations of current algorithm	Work to be performed in the future
Contradiction systems studied are only the ones that involve an evaluation parameter and a constraint requirement of the simulator.	We plan to develop a similar algorithm that starts from technical contradictions generated by decomposing the starting evaluation parameter into two evaluation parameters.
The proposed algorithm is restricted to object-oriented simulator in which the program text of each function of simulator can be formally derived.	The formal derivation has been used for Su-Field analysis. How to extract the Su-Field without formal derivation? The nature of knowledge that is mandatory to extract appropriate information requires to be further studied and we plan to develop alternative solutions for "black-box" simulators.
The contradiction systems disclosed have all the same	This may be viewed as an additional contribution of the article, since reformulating a contradiction system into

structure which enable a straightforward extraction of Su-Field models.	Su-Field models has not been yet formalized in part 1 of ARIZ. This lack of consistency in TRIZ models often leads to difficulties of practice for TRIZ beginners. However, it should be studied if the particular structure we propose (in which action parameter concerns a component of the system on which evaluation parameters are defined), enables to disclose all the Su-Field models that may have resulted from a less structured approach. This topic is difficult because such model reformulation may lead to add knowledge previously hidden in mind of ARIZ user.
The algorithm deals and relaxes constraints that were purposely defined as such. It does not enable to define new evaluation parameters.	An extension of this algorithm may also consider the fixed parameters p1…pj as constraints to be relaxed. Other source of embedded information may be also explored. Understanding why a source of information is more relevant than another is also a potential issue?
The algorithm considers only contradiction systems involving two EP (indeed one EP and one CR)	Another extension of the algorithm could consist in relaxing combination of constraints. The management of such "poly-contradictions" cannot be handled with TRIZ-classic tools and may eventually be studied with the purpose of reformulating them into TRIZ-classic contradictions.
In the Su-Field extraction algorithm proposed, it is assumed that, as soon as a mathematical relation exists between evaluation functions of sub-systems, a direct interaction involving a physical field also exists.	The proposed algorithm is based on the paradigm that an object-oriented simulator has been developed in a way that fits to designer representation of a real object (current paradigm in computer science). However, we plan to examine more in detail the similarities and differences in analysis when analyzing systems in TRIZ way and when defining objects to compute quantitative functions in computer science.
If the object-oriented decomposition of the simulator changes, the contradiction systems and Su-Field change.	Are there better system decompositions than others? The proposed algorithm may help us to understand what types of decompositions are leading to the most interesting contradiction systems. It should also be examined in which manner the best decomposition found fits with the 4 elements decomposition model proposed in TRIZ to analyze key elements of any system.
The object-oriented simulator considered from up to now has a simple structure: evaluation parameters and constraint requirements are computed thanks to the same object decomposition, functions involve expression that can be explicitly computed, etc…	NB: the problem generated by complexification of simulators only concerns Su-Fields extraction. Each complexification way could be studied step by step in order to check weather they can be transformed into a canonical way. This problem may also disappear if a Su-Field extraction algorithm is provided, which does not require access to the program text.
If several components are involved in the Su-Field description, it has been considered that O[m] has an action on the relationship between these objects.	There is no rationale so far for such an interpretation. This point should be examined more in particular in order to understand why this situation occurs (since, based on our knowledge, it seems not to occur when humans perform ARIZ).

| The algorithm does not solve the Su-Field model problem! | Merge this contradiction extraction selection approach with an algorithm that automatically provides model change proposals as proposed in [21] may enable to build an algorithm that invents |
| | A new evolutionary computation paradigm could then consist in starting the design process with very simplistic models and then enhance the modeling approach step by step in a controlled and efficient way. When optimization reaches its limit (either because of the computational complexity of reaching global optimum or because of the unsatisfying value of global optimum), model changes suggested by Inventive Standards may enable to bypass the limits. However, the entire automation of model changes proposed in [21] to support quantitative computation remains an open issue. |

5.2   What does the proposed model contribute to? What does it fail to provide?

Table 4. Summary of contributions and partial results brought in the article

Targets of the research work	Partial answer proposed in the article
Provide means of complex systems analysis, in order to study interaction between elements when expert's knowledge is lacking.	By use of an object oriented simulator, a vast range of contradictions can be stated and organized. Such a systematic extraction may lead to consider configurations not taken into account by experts.
Disclose rapidly multi-system level problem statement	The step by step formalization of contradiction statement process and reformulation at sub-system levels proposed herein may enable a straightforward implementation of the algorithm in computer and so increase the rapidity of problem formulation, providing a simulation program is available. Computer validation will be proposed in a further paper. This result may be improved by developing an algorithm that deals with more complex simulators.
Provide quantitative means of contradiction choice.	Thanks to automation and the capacity to evaluate performances of the sequence of configurations obtained, we expect to build quantitative indicators in order to ease the selection of contradiction. Other criteria of contradiction choice may also be implemented in the algorithm.
	However, we also expect various aspects linked with the reformulation process in ARIZ to be responsible of difficulty in defining such an impact measure a priori.
Help formulating Substance-Field models at the basis of TRIZ Inventive Standard application	Su-Field modeling is a direct consequence of mathematical relations at each decomposition level, given an object oriented simulator. Hence Su-Field models of problem are fully determined by the process presented above.
Reduce modelling complexity	Automation enables to obtain problem models in a

and modelling work leading to low marginal payoff during inventive problem solving.	straightforward manner. However, this relative rapidity has to be put in balance with the invisible work of developing mathematical models and programming the simulator used for extraction.
Control of knowledge handled during IPSP and its source.	Since knowledge is embeded since the beginning in the object oriented code or consists of some elementary mathematical transformations (derivation), no additional knowledge is required for the restricted part or IPSP depicted in this article. This point may be useful for research purpose in inventive problem solving, because it enables to study separately phenomena that are currently always linked (analysis and reformulation process for instance).
	Moreover, the algorithm may provide unexpected result that will raise questions about "human implicit control" while performing ARIZ. Those control functions may then be eventually implemented in the algorithm, depending on knowledge they are based on.

Acknowledgments

References

1. Dubois S., Rasovska I. and De Guio R.: Comparison of non solvable problem solving principles issued from CSP and TRIZ, pp: 83--94, IFIP International Federation for Information Processing, Ed., Computer-Aided Innovation: Springer Boston (2008)
2. Dubois S., Eltzer T., De Guio R.: A dialectical based model coherent with inventive and optimization problems, vol. 60, issue 8, pp: 575--583, Computers in Industry, ISSN: 0166-3615 (October 2009)
3. Cavallucci D., Eltzer T.: Parameter network as a means for driving problem solving process, vol. 30, issue1-2, pp: 125--136, International Journal of Computer Applications in Technology, ISSN: 0952-8091 (November 2007)
4. Cavallucci D., Khomenko N.: From TRIZ to OTSM-TRIZ: addressing complexity challenges in inventive design, vol 4, issue 1-2, pp: 4--21, International Journal of Product Development (2007)
5. Cavallucci D., Rousselot F., Zanni C.: Initial situation analysis through problem graph, vol 2, issue 4 pp: 310--317, CIRP Journal of Manufacturing Science and Technology (2010)
6. Eltzer T., Lutz P., Khomenkho N., Cavallucci D. Contribution to to early stages of analysis : a framework for contradiction's complexity representation. In: TRIZ Future, Firenze, Italy (2004)

7. Kucharavy D., De Guio R., Gautier L., Marrony M.: Problem Mapping for the Assessment of Technological Barriers in the Framework of Innovative Design. In: 16th International Conference on Engineering Design, ICED'07, Ecole Centrale Paris, Paris, France, 2007.

8. Khomenko N., De Guio R., Lelait L. & Kaikov I.: A framework for OTSM-TRIZ based computer support to be used in complex problem management, Vol. 30, No. 1/2, pp 88--104, International Journal of Computer Applications in Technology (2007)

9. Khomenko N., De Guio R. & Cavallucci D. Enhancing ECN's abilities to address inventive strategies using OTSM-TRIZ, vol. 1 pp: 98-113, International Journal of Collaborative Engineering (2009)

10. Khomenko N., De Guio R.: OTSM Network of Problems for representing and analysing problem situations with computer support, 2nd IFIP Working Conference on Computer Aided Innovation, October 8th and 9th, 2007, Delphi Corporation, Technical Center Brighton, 12501 E. Grand River, Brighton, MI 48114, USA., Springer Publishers (2007)

11. Cugini U, Cascini G, Ugolotti M.: Enhancing interoperability in the design process – The PROSIT approach. Proceedings of the 2nd IFIP Working Conference on Computer Aided Innovation, Brighton (MI), USA, 8-9 October, 2007; pp. 189—200, Trends in Computer-Aided Innovation, Springer, ISBN 9780387754550 (2007)

12. Dubois S., Rasovska I. and De Guio R. (2009) , Towards an automatic extraction of Generalized System of Contradictions out of solutionless Design of Experiments, pp 70-79. In, Growth and Development of Computer -Aided Innovation: Springer Boston, 2009

13. Rasovska I., Dubois S. and De Guio R. (2010), Study of different principles for automatic identification of generalized system of contradictions out of design of experiments, 8th International Conference of Modeling and Simulation - MOSIM'10 - "Evaluation and optimization of innovative production systems of goods and services", May 10-12, 2010, Hammamet – Tunisia

14. Altshuller G.: Inventive Problem Solving Algorithm: ARIZ-85C© G.S., 1956-1985. English version to be found at http://www.seecore.org/d/ariz85c_en.pdf

15. Conrardy C., De Guio R., Zuber B.: Facetwise study of modelling activities in the algorithm for inventive problem solving ARIZ and evolutionary algorithms, Proceedings of the Fourth International Conference on Design Computing and Cognition, University of Stuttgart, Stuttgart, Germany (July 2010)

16. Koza J-R.: Genetic Programming: On the Programming of Computers by Means of Natural Selection. The MIT Press ed. ISBN 0262111705 (1992)

17. Koza, J. R.: Genetic Programming IV: Routine Human-Competitive Machine Intelligence. New York, NY: Springer, ISBN: 1402074468 (2003)

18. Goldberg D.E.: The Design of Innovation: Lessons from and for Competent Genetic Algorithms. Kluwer, Dordrecht ISBN 1402070985 (2002)

19. Cascini, G., et al.: TETRIS, TEaching TRIz at School, available from: http://www.tetris-project.org/ (2009)

20. Collet P. et al.: EASEA platform, SONIC (Stochastic Optimisation and Nature Inspired Computing) theme of the FDBT team at Université de Strasbourg, available from https://lsiit.u-strasbg.fr/easea/index.php/EASEA_platform

21. Bultey A., De Bertrant De Beuvron F. Rousselot F.: A substance-field ontology to support the TRIZ thinking approach, vol. 30, issue 1/2, pp: 113—124, International Journal of Computer Applications in Technology, ISSN: 0952-8091 (November 2007)