Archives of Computational Methods in Engineering
https://doi.org/10.1007/s11831-020-09509-y

ORIGINAL PAPER

# Geometric Over-Constraints Detection: A Survey

Hao Hu[1,2,3] · Mathias Kleiner[3] · Jean-Philippe Pernot[3] · Chao Zhang[1] · Yanjia Huang[1] · Qian Zhao[1] · Sunny Yeung[2]

Received: 23 February 2020 / Accepted: 3 October 2020


**Abstract**

Currently, geometric over-constraints detection is of major interest in several different fields. In terms of product development process (PDP), many approaches exist to compare and detect geometric over-constraints, to decompose geometric systems, to solve geometric constraints systems. However, most approaches do not take into account the key characteristics of a geometric system, such as types of geometries, different levels at which a system can be decomposed e.g numerical or structural. For these reasons, geometric over-constraints detection still faces challenges to fully satisfy real needs of engineers. The aim of this paper is to review the state-of-the-art of works involving with geometric over-constraints detection and to identify possible research directions. Firstly, the paper highlights the user requirements for over-constraints detection when modeling geometric constraints systems in PDP and proposes a set of criteria to analyze the available methods classified into four categories: level of detecting over-constraints, system decomposition, system modeling and results generation. Secondly, it introduces and analyzes the available methods by grouping them based on the introduced criteria. Finally, it discusses possible directions and future challenges.



## 1 Introduction

Product design is a cyclic and iterative process, which manages the creation of the product itself with different requirements. The development process is composed of the idea generation stage, concept stage, product design stage and detailed engineering stage, all of which are conducted to satisfy requirements at different stages. According to [1], requirements can be specified from preliminary design to process planing, which adopts criteria to evaluate design variants and selects the one of best performance when using the product.

Usually, a product shape generates from an optimization problem where the various requirements are specified. In fact, the final shape of a product often results from a long optimization process which tries to satisfy different requirements. Those requirements can be of different types and their computation may require the need of external tools or libraries.

✉ Hao Hu
huhaoseu@gmail.com

1 Guangdong Bright Dream Robotics, Foshan, China

2 Guangdong Country Garden School, Foshan, China

3 Arts et Métiers Institute of Technology, LISPEN, HESAM Université, Aix-en-Provence 13617, France

In general, the creation of a product can be treated as a result of an optimization process where various requirements (e.g. functional, aesthetic, economical, feasibility) have to be satisfied. Requirements can be seen as constraints. For example, the shape of a turbine blade is a result of a complex optimization process which is to get the best solution satisfying aerodynamic and mechanical constraints. Since requirements are added at all stages of product design process, different users have different intention of applying constraints. According to [2], in the context of shape generation and modification, constraints can be classified into four semantic levels, depending on the type of the constrained entity:

- Level 1: constraints attached to a geometric element of a configuration: such as position constraints used to manipulate the shape of a geometry.
- Level 2: constraints between two or more geometric elements of a configuration: for instance, G0/G1/G2 continuity between trimmed patches.
- Level 3: constraints attached to the whole configuration like a volume constraint.
- Level 4: constraints related to the product itself rather than to the geometry. For example, to resist the usage of a product, there needs a requirement on the mechanical properties such as the acceptable maximum stress. There-



| Journal : Large 11831 | Article No : 9509 | Pages : 25 | MS Code : 9509 | Dispatch : 23-7-2021 |
|---|---|---|---|---|

fore, constraints should be specified to link the geometry with parameters of the material or boundary conditions of the product.

The above levels describe how to express constraints attached to a product. They are generally expressed either with equations, a function to be minimized, and/or using procedures [3]. The latter refers to the notion of black box constraints, which will be discussed in Sect. 2.

However, information provided by uers may be inconsistent and the overall set can be over-constrained when manupulating CAD models directly [47]. In most of today's modeler, a geometric configuration can be of three types:

- Under-constrained: number of unknowns is greater than the number of equations. Such case happens quite often since designers often insert extra DoFs to satisfy requirements.
- Well-constrained: number of unknowns is equal to the number of equations.
- Over-constrained: number of unknowns is less than the number of equations. The type of extra equations have two possibilities:

  - Redundant: these equations are consistent with the other ones. That is, they do not affect the solution of the original system.
  - Conflicting: fully inconsistent with the others when constraints express contradictory requirements and lead to no solution.

A geometric system may be solvable if it is under-constrained or well-constrained. But when it is over-constrained, the system is hard to solve or even non-solvable. To make a system consistent with designer's requirements, it is necessary to detect geometric over-constraints and present them to designers for debugging purpose. In this paper, we collect and classify a state-of-the-art methods for detecting geometric over-constraints, including: (1) definitions of geometric over-constraints; (2) clear identification of criteria used to characterize methods; (3) study the methods and compare them according to the criteria; (4) proposed frameworks for detecting geometric over-constraints.

The paper is organized as follows. Section 2 introduces definitions of geometric over-constraints. Section 3 defines criteria for evaluating different detection methods. The details of evaluating each method is then discussed in Sect. 4. Finally, Sect. 5 concludes the paper as well as future work.

# 2 Representations and Definitions

## 2.1 Representation of Geometric Constraints Systems

*Equations* CAD modelers provide their solvers of geometric constraints and usually the solver has its own constraints editor. Basically, the constraints concern vertices, straight lines, planes, circles, spheres, cylinders or freeform curves and surfaces whose parameters are the unknown variables. Constraints ranging from level 1 to level 3 (Sect. 1) can be represented with equations. Those equations can be linear or non-linear. Classical solvers use these constraints to sketch and constrain the shape of desired models. For example, the 2D distance constraint $d$ between two points $(x, y)$ and $(x_0, y_0)$ is translated to the equation $(x-x_0)^2+(y-y_0)^2-d^2=0$. Continuity constraints between two patches can also be represented with equations. Moreover, those mathematical equations can also be represented using computational graph, which is based on Directed Acyclic Graphs (DAGs). In such a representation, a DAG is a tree with shared vertices. The leaves of the tree are either variables (i.e. parameters or unknowns) or numerical coefficients. The internal nodes of the tree are either elementary arithmetic operations or functions such as *exp*; *sin*; *cos*; *tan*. The DAG is also called white box DAG, since it allows for computing the derivatives and hessians automatically. If mathematical equations associated to geometric constraints are available, it is possible to compute the expressions of the derivatives with formal calculus, which can be resorted to using the Grobner basis or Wu-Ritt method if all the constraints are algebraic and can be triangulated into the form $f1(U;x1)=f2(U;x1;x2)=:::=0$ ($U$ is the parameters vector and $x_i$ are the unknown variables).

*Black boxes* On the contrary, a DAG is called a black box DAG, and a constraint is called a black box constraint when it cannot be represented with equations or are not computable in practice [3]. This corresponds to constraints of level 4 discussed in Sect. 1. Examples such as, maximum of the Von Mises stress should be smaller than 100MPa, the final product should cost less than 100, are requirements which cannot be transformed into a set of equations. In the work of [3], they proposed to use black box DAGs for variational geometric modeling of free-form surfaces and subdivision surfaces. A prototype, DECO, is presented to show the feasibility and promises of the approach. Black box constraints happen when free-form surfaces are generated tediously from modeling functions (e.g. sweep, loft, blend). They cannot be manipulated in the same way as if some equations were available and solvers have to take into account these constraints expressed by





Journal : Large 11831 | Article No : 9509 | Pages : 25 | MS Code : 9509 | Dispatch : 23-7-2021

functions i.e. constraints requiring the call to a function. In this paper, we will only consider configurations involving constraints can be defined by a set of equations. Configurations involving black box constraints will not be addressed.

## 2.2 STAR Definitions of Geometric Over-Constraints

### 2.2.1 Definitions at the Level of Geometries

At this level, definitions are classified into two groups: constraint graph group and bipartite graph group. A constraint graph is transformed into a weighted constraint graph, where the weight of a vertex represents DoFs (Degree of freedoms) of an entity and the weight of an edge represents DoFs removed by a constraint. For the bipartite graph group, only the weight of vertices are added: the weight of an entity equals to its DoFs and the weight of a constraint equals to the DoFs it can remove.

*Definitions based on constraint graph*Here, we use $G = (V, E)$ to represent a constraint system with $\mid V \mid$ number of entities and $\mid E \mid$ number of constraints.

In Rigidity Theory [4], Laman's theorem [5] characterizes the rigidity of bar frameworks, where a geometric system is composed of points constrained by distances.

**Theorem 1** *A constraint system in the 2D plane composed of N points linked by M distances is rigid iff* $2 \cdot N - M = 3$ *and for any subsystem composed of n points and m distances,* $2 \cdot n - m \geq 3$.

The constraints and entities are limited to distances and points respectively. Podgorelec [6] extended the theorem by assuming that each geometric element has 2 DoFs and each constraint eliminates 1 DoF. Therefore, the weight of vertices and edges are of the constraint graph is 2 and 1 respectively.

**Definition 1** For constraint graph $G = (V, E)$, a geometric constraint system is:

- *Structurally over-constrained* if there is a subgraph $G' = (V', E')$ with $1 \cdot |E'| > 2 \cdot |V'| - 3$,
- *Structurally under-constrained* if G is not *structurally over-constrained* and $1 \cdot |E| < 2 \cdot |V| - 3$, or
- *Structurally well-constrained* if G is not *structurally over-constrained* and $1 \cdot |E| = 2 \cdot |V| - 3$.

**Definition 2** A constraint $e$ is a *structural over-constraint* if a structurally over-constrained subsystem $G' = (V', E')$ of $G$ with $e \in E'$, can be derived such that $G'' = (V', E' - e)$ is structurally well-constrained.

An example is given to illustrate the Definition 1. The system is composed of 3 points (each has 2 DoFs) with different constraints in 2D space. As it is shown in the Fig. 1a, it is over-constrained because it contains 3 distance constraints and 3 vertical position constraints. Since each consumes 1 DoF, the total system consumes 6 DoFs, satisfying $6 > 2 \times 3 - 3$. The configuration of the Fig. 1b is structurally well-constrained since the constraints are reduced into 3 distance constraints, satisfying $3 = 2 \times 3 - 3$. The configuration of the Fig. 1c is structurally under-constrained since only 2 distance constraints are left, satisfying $3 < 2 \times 3 - 2$.

The Definition 1 is correct if only all geometric entities are points and all constraints are distance constraints in 2D. It cannot be used to characterize geometric constraints systems where constraints other than distance constraints are involved. For example, in the case of angle constraints in 2D: 3 line segments with 3 incidence constraints form a triangle with $3 \cdot 4 - 3 \cdot 2 = 6$ DoFs. If added 3 angle constraints (each remove 1 DoF), the system will be *Structurally well-constrained* according to the Definition 1. However, 2 angle constraints are enough since the third one is a linear combination of the other two.

In 3D, Laman's theorem can be extended as follows: for the relative location of $N$ points to be well defined, $E = 3N - 6$ number of distance constraints are needed, and no subsystem is over-constrained, i.e., for all subsystems with $n$ number of points and $e$ number of distance constraints, $e \leq 3n - 6$. This condition is necessary but not sufficient. A counter example is the double banana geometry shown in the Fig. 2. It is common to use Laman's conditions to decompose geometric systems in 3D since these conditions are necessary [7].

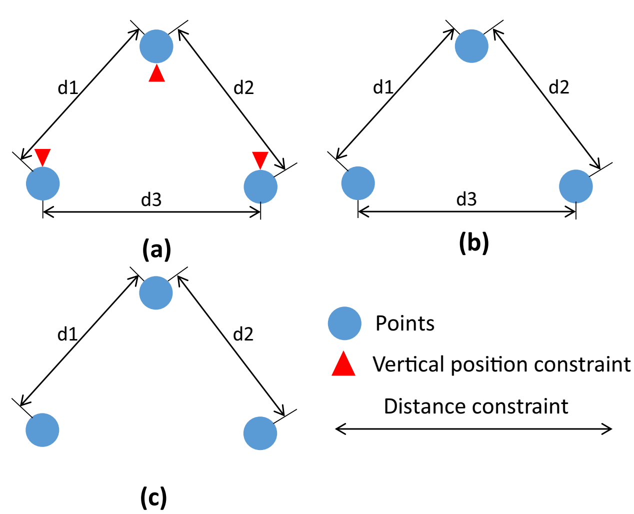



**Fig. 1** **a** Structurally over-constrained, **b** structurally well-constrained, **c** structurally under-constrained

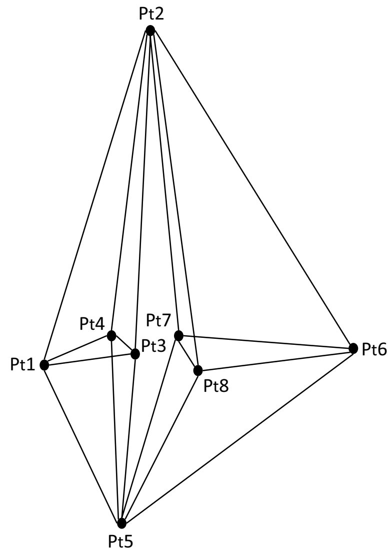



**Fig. 2** Double-Banana geometry where all vertexes are variables and all edges are distance constraints

Sitharam and Zhou [8] introduced a set of new definitions trying to adapt Laman's theorem to deal properly with the double banana geometry. First, they replaced the value 3 in the Definition 1 with $D$, which is a function of dimension $d$: $D = (d + 1) * d/2$. Then, they defined the DoF, DoC as follows.

**Definition 3** Degree of freedom (DoF) of a geometry entity ($DoF(v)$, $v$ is the geometry) is the number of independent parameters that must be set to determine its position and orientation. For a system $G = (V, E)$, its DoFs is defined as $DoF(G) = \sum_{v \in V} DoF(v)$.

**Definition 4** Degree of freedom of a geometric constraint ($DoC(e)$, $e$ is the constraint) is the number of independent equations needed to represent it. For a system $G = (V, E)$, the DoFs all constraints can remove is $DoC(E) = \sum_{e \in E} DoC(e)$.

**Definition 5** For constraint graph $G = (V, E)$, a geometric constraint system is:

- *Structurally over-constrained* if there is a subgraph $G' = (V', E')$ satisfying $DoC(E') > DoF(V') - D$,
- *Structurally well-constrained* if $DoC(E) = DoF(V) - D$ and all subgraphs $G' = (V', E')$ satisfying $DoC(E') \leq DoF(V') - D$,
- *Structurally under-constrained* if $DoC(E) < DoF(V) - D$ and contains no *structurally over-constrained* subgraphs.

A typical example that the Definition 5 cannot treat properly is 2 points binding with distance constraint in 3D. It allows for only 5 of 6 possible independent displacements since the system cannot rotate around axis crossing the 2 points. More counter examples in [9] suggest that the value of $D$ depends on the system itself rather than dimension. Therefore, Jerman et al introduced the *Degree of Rigidity* (DoR) to replace the DoFs of a system if it is rigid. Their definitions are as follows.

**Definition 6** For constraint graph $G = (V, E)$, a geometric constraint system is:

- *Structurally over-constrained* if there is a subgraph $G' = (V', E')$ satisfying $DoC(E') > DoF(V') - DoR(V')$,
- *Structurally well-constrained* if $DoC(E) = DoF(V) - DoR(V)$ and all subgraphs $G' = (V', E')$ satisfying $DoC(E') \leq DoF(V') - DoR(V')$,
- *Structurally under-constrained* if $DoC(E) < DoF(V) - DoR(V)$ and contains no *structurally over-constrained* subgraphs.

The rule of computing the DoR is described in [9]. Within the rule, for two secant planes in 3D, the DoR is 5 while for two parallel planes is 4. Similarly, the DoR of 3 collinear points is 2, while the DoR of 3 non collinear points is 3.

A pure graph based method cannot determine whether 3 points are collinear or not, or whether two planes are parallel or not. It either assumes the configuration is generic or it verifies if the parallelism/collinearity is an explicit constraint of a system; but it may happen that the parallelism/collinearity is a remote consequence of a set of constraints, thanks to Desargues, or Pappus, or Pascal, or Miquel theorems: the incidence in the conclusion is a nontrivial consequence of the hypothesis. This will be further discussed in the Definition 12.

*Definitions based on bipartite graph* Latham et al [10] introduced similar definitions based on a connected graph. It is a graph where vertices represent geometric entities and constraints, which can be treated as a bipartite graph. Note that, we use $G = (U, V, E)$ to denote a bipartite graph whose partition has the vertices $U$ (entities) and $V$ (constraints), with $E$ denoting the edges of the graph.

**Definition 7** For bipartite graph $G = (U, V, E)$, a geometric constraint system is:

- *Structurally over-constrained* if it contains an unsaturated constraint,
- *Structurally under-constrained* if it contains an unsaturated entity.

A vertex $u$ or $v$ is said to be unsaturated if $DoF(u)$ or $DoC(v)$ is not equal to the number of weights of incident edges in a





| Journal : Large 11831 | Article No : 9509 | Pages : 25 | MS Code : 9509 | Dispatch : 23-7-2021 |
|---|---|---|---|---|

maximal weighted matching. An unsaturated constraint is a *structural over-constraint*. The weights of edges are computed by maximal weighted matching of a bipartite graph.

### 2.2.2 Definitions at the Level of Equations

In this section, we summarize the definitions used when a qualitative study of geometric systems is performed at the level of equations. Modeling at the level of geometries preserves geometric information of a system. Modeling at the level of equations, however, discards geometric properties of a system but enables a fine detection of geometric over-constraints.

*Structural definitions* System of equations are transformed into bipartite graph, where vertices represent equations and variables respectively. The characterization is based on the results of maximum matching [11]. Here, we assume that $G = (U, V, E)$ is a bipartite graph with $U$ and $V\,(U \cap V = \varnothing)$ representing variables and equations respectively, and $E$ representing edges.

**Definition 8** For bipartite graph $G = (U, V, E)$ and its subgraph $G' = (U', V', E')$. $G'$ is:

- *Structurally over-constrained* if the number of elements in $U'$ is smaller (in cardinality) than the number of $V'$. i.e. $\mid U' \mid < \mid V' \mid$
- *Structurally well-constrained* iff $G'$ has perfect matching.
- *Structurally under-constrained* if the number of elements in $U'$ is larger (in cardinality) than the number of elements in $V'$. i.e. $\mid U' \mid > \mid V' \mid$

**Definition 9** Let $M$ be a maximum matching of $G = (U, V, E)$. If $M$ is not perfect matching and $V'$ is the subset of $V$ which is not saturated by $M$, then equations of $V'$ are the *Structural over-constraints*.

*Numerical definitions* Informally, an over-constrained constraints system has no solutions, a well-constrained constraints system has a finite number of solutions, and a under-constrained constraints system has infinite solutions. In the work of Hu et al. [12] as well as the recent work of Zou et al. [46], they gave the following definitions.

**Definition 10** Let $G = (E, V, P)$ be a geometric constraints system, where $E$ is a set of equations, $V$ is a set of variables and $P$ is a set of parameters. The set of solutions to $G$ is denoted *Sol(G)*. A geometric constraints system is *inconsistent* iff $Sol(G) = \varnothing$ and is *consistent* iff $Sol(G) \neq \varnothing$.

**Definition 11** Let $G = (E, V, P)$ be a *consistent* geometric constraints system. Let $G' = (E \cup E_c, V, P')$ be an *inconsistent* geometric constraints system, where $E_c$ is a set of equations forming a constraint $C = \{E_c \mid E_c \cap E = \varnothing\}$ and $P \subset P'$. As a result, $C$ is a *conflicting constraint* with respect to $G$.

**Lemma 1** *Let $G = (E, V, P)$ be a consistent geometric constraints system. Let $G' = (E \cup E_r, V, P')$ be a consistent geometric constraints system, where $E_r$ is a set of equations forming a constraint $R = \{E_r \mid E_r \cap E = \varnothing\}$ and $P \subset P'$, and Sol(G) is the same as Sol(G'). As a result, $R$ is a redundant constraint with respect to $G$.*

**Definition 12** Let $G = (E, V, P)$ be a weakly connected geometric constraints system (its components are weakly connected [13]) which can be decomposed into the following two subsystems: $G_b = (E_b, V, P)$ and $G_o = (E_o, V, P)$ with $\{E = E_b \cup E_o, E_b \cap E_o = \varnothing\}$. If any constraint $E_{oi}$ in $E_o$ is either redundant or conflicting with respect to $G_b$, and if card $(E_b) \geq$ card $(E_o)$, then $E_b$ is a set of *basis constraints* and $E_o$ is a set of *numerical over-constraints*.

However, we have to mention that the Definition 12 is not consistent with matroid theory. For example, in the Fig. 3, $e1$ is conflicting both with $e0$ and $e2$. According to our definitions of redundant, conflicting, and basis constraints (Definitions 11 and 12), the result would be: $e1$ is conflicting with basis constraints $\{e0, e2\}$.

According to the matroid theory [14], for any two subsets $A$ and $B$ of $E$, $r(A \cup B) + r(A \cap B) \leq r(A) + r(B)$. That is, the rank is a submodular function. Suppose $A = e0, e1, B = e1, e2$. Both rank(A) and rank(B) is 1 because A and B are dependent respectively. We can also deduce that Rank$(A \cup B)$ =2 and Rank$(A \cap B)$ =1. As a result, we should have $2 + 1 \leq 1 + 1$, which is wrong. We redefine the Definition 11 and the Definition 12 so as to be consistent with matroid theory in the next section.

*Geometric redundancy Geometric redundancy* refers to those additional constraints trying to constrain internally established relations. The relations are consequences of domain-dependent mathematical theorems hidden in a geometric configuration. Users are typically not aware of these implicit constraints and will always try to constrain the internal established relations by additional constraints.

Geometric redundancy does not use parameters. Therefore, this type of geometric over-constraints cannot be detected by methods based on DoF-counting. In 2D, a typical example is the 3-angles constraints specified on a triangle. Obviously,

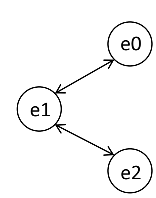



**Fig. 3** An example: $e1$ is conflicting with $e0$ and $e2$





Journal : Large 11831 | Article No : 9509 | Pages : 25 | MS Code : 9509 | Dispatch : 23-7-2021

total value of three angles equals to 180°. It is not necessary to specify all three angles as constraints because the value of third one can be easily derived once the values of other two angles are defined. Therefore, specifying the 3-angles constraints will generate a geometric redundancy that is either redundant or conflicting. In 3D, every incidence theorem (Desargues, Pappus, Pascal etc) provides implicit dependent constraints [15]. For example, Pappus's hexagon theorem [16] states that given one set of collinear points $A$, $B$, $C$, and another set of collinear points $D$, $E$, $F$, then the intersection points $X$, $Y$, $Z$ of line pairs $AE$ and $DB$, $AF$ and $DC$, $BF$ and $EC$ are collinear, lying on the Pappus line. In this case, if specifying line pairs $XY$ and $YZ$ to be collinear, then this constraint is the geometric redundancy (Fig. 4).

## 2.3 Evaluation

A set of criteria are defined to evaluate these definitions (Table 1). These criteria are: $D$ is a dimension (system)-dependent constant; geometries refer to the geometric type a definition used to specify; counter example lists geometries that a definition cannot deal with.

From the table below, we can see that definitions can de divided into two groups: Definition 1,5,6 and Definition 7,8,10. Because the former group manipulate geometric elements directly at the level of geometries and geometric constraints are usually supposed to be independent of all coordinates system, they can not be used to determine the location and orientation of a geometric configuration (no fixation) as well as carefully defined the value of $D$. Also, the defined type of geometries and constraints are limited. For example, the Definition 1 and Definition 5 are defined for points geometries and distances constraints only. But collinear (and cocyclic, coconic, cocubic, etc.) points are forbidden. The Definition 6 extends the type of geometries to points, lines and planes as well as it allows for incidence constraints to be defined. The Definition 7 extracts geometric entities and constraints to DoFs and DoCs, and define over-constraints by simply comparing the number of DoFs of geometric entities and DoCs of geometric constraints. In this way, the definition is not limited to any specific class of geometric entities and constraints. The Definition 8, and the Definition 10, however, are numerical definitions dealing with geometries and constraints at the level of equations. These definitions can cover any geometric entities and constraints once they are represented with equations. Counter examples are the black box constraints which cannot be represented with equations. Moreover, these definitions require systems to be fixed with respect to a global coordinate system and thus $D = 0$. Finally, since geometric redundancy does not use parameters of any geometries of a constraints system, it cannot be covered by any of these definitions.

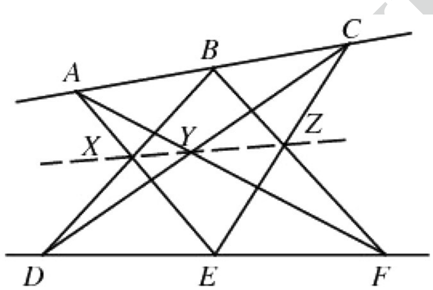



**Fig. 4** Pappus's hexagon theorem: Points $X$, $Y$ and $Z$ are collinear on the Pappus line (dotted line). The hexagon is $AFBDCE$

To cover cases like geometric redundancy as well as be consistent with the matroid theory, we redefine basis equations, redundant and conflicting equation as follows.

**Definition 13** Let $G = (E, V, P)$ be a geometric constraints system, where $E$ is a set of equations, $V$ is a set of variables and $P$ is a set of parameters. Let $E_r$ be a non-empty collection of subsets of $E$, called basis equations (we call it basis in short), satisfying:

- no basis properly contains another basis;
- if $E_{r1}$ and $E_{r2}$ are basis respectively and if $e$ is any equation of $E_{r1}$, then there is an equation $f$ of $E_{r2}$ such that $\{(E_{r1} - e) \cup f\}$ is also a basis.

**Definition 14** Let $G = (E, V, P)$ be a geometric constraints system. Let $E_r$ be a basis. For an equation $e$, adding it to $E_r$

**Table 1** Evaluations of definitions

| | D | Geometries | Constraints | Counter example |
|---|---|---|---|---|
| Definition 1 | 3 | Points | Distances | Double banana |
| Definition 5 | 0,3,6 | Points | Distances | ex1 |
| Definition 6 | DoR | Points,lines,planes | Distances, incidencies | ex2 |
| Definition 7 | 0 | Any | Any | ? |
| Definition 8 | 0 | Any | Any | Black box constraints |
| Definition 10 | 0 | Any | Any | Black box constraints |

ex1: 2 points binding with distance constraint in 3D

ex2: configurations with geometric redundancy forming a new group: $\{E_r \cup e\}$. If $\{E_r \cup e\}$ is solvable, then $e$ is a redundant equation.

**Definition 15** Let $G = (E, V, P)$ be a geometric constraints system. Let $E_r$ be a basis. For an equation $e$, adding it to $E_r$ forming a new group: $\{E_r \cup e\}$. If $\{E_r \cup e\}$ is non-solvable, then $e$ is a conflicting equation.

**Definition 16** Let $G=(E,V,P)$ be a geometric constraints system which is composed of two subsystems: $G_b=(E_b,V,P)$ and $G_o=(E_o,V,P)$ with $\{E=E_b \cup E_o, E_b \cap E_o = \varnothing\}$. If $E_b$ is a basis, then $E_o$ is a set of numerical over-constraints.

*Spanning group* For an over-constraint $E_{oi} \in E_o$, the *Spanning Group* $E_{sg}$ of $E_{oi}$ is a group of independent constraints, with which $E_{oi}$ is redundant or conflicting. For linear systems, the spanning group $E_{sg} = \{e_{sg1}, e_{sg2}, \dots, e_{sgn}\} \subset E_b$ of $E_{oi}$ satisfies:

$$E_{oi} = \sum_{j=1}^{n} c_j e_{sgj} + b \tag{1}$$

where $c_j \neq 0$ and is the corresponding scalar coefficient, $\{e_{sg1}, e_{sg2}, \dots, e_{sgn}\}$ are linear independent and $b$ is the bias. Thus, $E_{oi}$ is a linear combination of $\{e_{sg1}, e_{sg2}, \dots, e_{sgn}, b\}$. Moreover, $E_{oi}$ is redundant if $b = 0$ otherwise it is conflicting.

However, $E_{sg}$ is not unique for a given $E_{oi}$. For example, assuming a linear system of constraints represented at the level of equations:

$$\begin{aligned} e1 &: x_1 + x_2 + x_3 + x_4 = 1 \\ e2 &: x_1 + 2x_2 + 3x_3 + x_4 = 4 \\ e3 &: x_1 - 2x_2 + x_3 + x_4 = 5 \\ e4 &: 6x_1 + x_3 + 2x_4 = 7 \\ e5 &: 8x_1 + 5x_3 + 4x_4 = 17 \\ e6 &: 11x_1 + x_2 + 10x_3 + 7x_4 = 27 \end{aligned} \tag{2}$$

Clearly, the system is over-constrained since there are more equations than variables. Through linear analysis of the system, we find that $e5$ is a linear combination of $\{e2, e3, e4, 1\}$ and is spanned by $\{e2, e3, e4\}$; $e6$ is a linear combination of $\{e1, e2, e3, e4, 1\}$ and is spanned by $\{e1, e2, e3, e4\}$ (Fig. 5). Since the bias of the two groups is 1, both $e5$ and $e6$ are conflicting. In this case, $\{e1, e2, e3, e4\}$ can be treated as a set of basis constraints since all the equations are independent and the number of them equals to the number of variables.

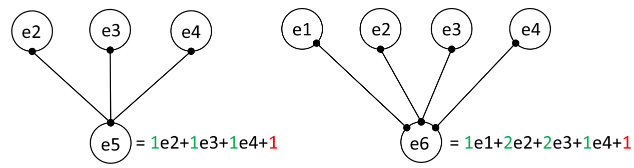



**Fig. 5** Spanning group of $e5$ and $e6$: numbers marked green are coefficients while the ones marked red are the biases

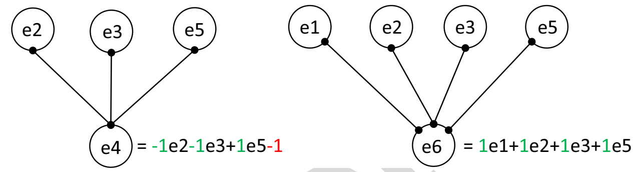



**Fig. 6** Spanning group of $e4$ and $e6$: numbers marked green are coefficients while the one marked red is the bias

$$\begin{aligned} e1 &: x_1 + x_2 + x_3 + x_4 = 1 \\ e2 &: x_1 + 2x_2 + 3x_3 + x_4 = 4 \\ e3 &: x_1 - 2x_2 + x_3 + x_4 = 5 \\ e4 &: 6x_1 + x_3 + 2x_4 = 7 \end{aligned} \tag{3}$$

However, if we replace $e4$ with $e5$, the new set $\{e1, e2, e3, e5\}$ is also the basis constraints set. Linear analysis result shows that $e4$ is a linear combination of $\{e2, e3, e5, -1\}$ and is spanned by $\{e2, e3, e5\}$; $e6$ is a linear combination of $\{e1, e2, e3, e5\}$ and is spanned by $\{e1, e2, e3, e5\}$ (Fig. 6). Also, $e4$ is conflicting and $e5$ is redundant according to the corresponding bias values.

$$\begin{aligned} e1 &: x_1 + x_2 + x_3 + x_4 = 1 \\ e2 &: x_1 + 2x_2 + 3x_3 + x_4 = 4 \\ e3 &: x_1 - 2x_2 + x_3 + x_4 = 5 \\ e5 &: 8x_1 + 5x_3 + 4x_4 = 17 \end{aligned} \tag{4}$$

From the Fig. 5 and the Fig. 6, we can see that the spanning group of $e6$ is not unique, which depends on the set of basis constraints. Also, the type of an over-constraint can change: $e6$ is conflicting with respect to the basis constraints set {e1, e2, e3, e4} while redundant with respect to the basis constraints set {e1, e2, e3, e5}.

To the best of our knowledge, there is no formal definitions that can cover cases like black box constraints, let alone the corresponding detection methods. Therefore, in this paper, both of the definitions and the detection methods address only white box constraints.

## 3 Evaluation Criteria

To carry out appropriate analyses and comparisons between the over-constraints detection approaches, various evaluation criteria and a ranking system are proposed in this section. Considering the detection process as well as users' needs for debugging, these approaches are classified into four main categories: criteria related to the level of detecting over-constraints; criteria related to the system decomposition; criteria related to the system modeling; criteria related to the way of generating results. Such ranking system permits a qualitative classification of the various approaches according to the specified criteria. In this section, following the tagging system used in [48], a boolean scale is adopted to characterize the capabilities of approaches. Firstly, the symbol $\ominus$/$\oplus$ is used to tag the methods not adapted/well adapted, incomplete/complete with respect to the considered criterion (Table 2). They state a negative/incomplete ($\ominus$) or positive/complete ($\oplus$) tendency of the approaches with respect to the given criteria. They are defined in such a way that the optimal method would never be assigned the symbol($\ominus$). Secondly, in case the information contained in the articles do not enable the assessment of a criterion, symbol (?) is used. Finally, the symbol ($\odot$) means criteria that have no meaning for the method and are simply not applicable.

For example, distinguishing redundant and conflicting constraints is a criteria for evaluating numerical detection methods. However, there is no meaning to apply it to evaluate structural detection methods since the latter only generate structural over-constraints. Of course, synthesis results are from our understanding of the publications.

### 3.1 Criteria Attached to the Level of Detecting Over-Constraints

The first criterion is relative to the type of geometric over-constraints (Fig. 7a), which are either numerical (a$\oplus$) or structural (a$\ominus$). Second criterion concentrates on distinguishing redundant and conflicting constraints detected by numerical methods (Fig. 7b). Finally, in engineering design, designers could better debug and modify a geometric over-constraint if its spanning group is informed (Fig. 7c). This criterion evaluates numerical methods only (Table 3).

**Table 2** Symbols used to characterize the approaches

| Symbols | Criteria |
|---|---|
| $\ominus$ | Not adapted/incomplete |
| $\oplus$ | Well adapted/complete |
| ? | Not appreciable |
| $\odot$ | No meaning/not applicable |

### 3.2 Criteria Related to the System Decomposition

Decomposition is an important phase in geometric constraints solving domain. A large system is decomposed into small solvable subsystems which speeds up the solving process. A desirable method should return the decomposition result to a user for debugging purpose by generating over-constrained components, which helps him/her locating the geometric over-constraints (d$\oplus$). Also, the ability to generate rigid subsystems should be considered. Here, the *rigid* is of two meanings. Numerical methods detect the rigid subsystem which is solvable (finite solutions, e$\oplus$) while structural methods detect the rigid subsystem which is structurally well-constrained (Definition 5, e$\ominus$). Usually, the rigid subsystems are arranged with solving order and over-constraints within each subsystem can be detected by analyzing the subsystem individually (Table 4).

Decomposition methods should take into account the singularities. Indeed, many methods work under a genericity hypothesis and decompose systems into generically solvable components. A generic configuration remains

**Table 3** Criteria attached to the detection level (set 1)

| Detection level | | Gradation of criteria | |
|---|---|---|---|
| Level | Criteria | $\oplus$ | $\ominus$ |
| a | Type | Numerical | Structural |
| b | Redundant/conflicting | Yes | No |
| c | Spanning group | Yes | No |

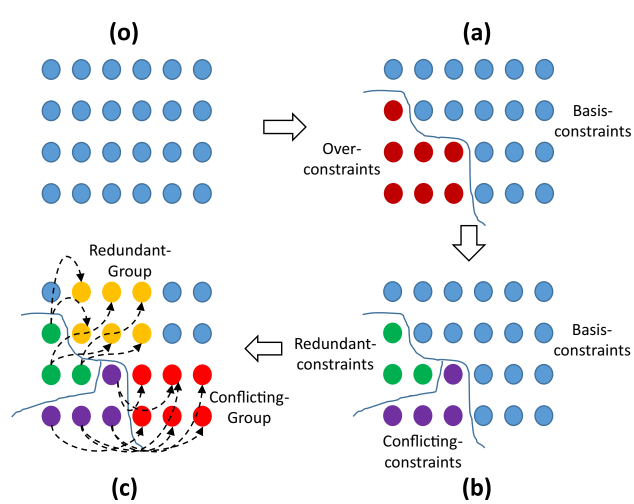



**Fig. 7** (o): Level o (**a**): Level a (**b**): Level b (**c**): Level c





Journal : Large 11831 | Article No : 9509 | Pages : 25 | MS Code : 9509 | Dispatch : 23-7-2021

**Table 4** Criteria related to system decomposition (set 2)

| Decomposition | | Gradation of criteria | |
|---|---|---|---|
| Level | Criteria | ⊕ | ⊖ |
| d | Over-constrained components | Yes | No |
| e | Rigid subsystems | Numerical | Structural |
| f | Singular configuration | Yes | No |

**Table 5** Criteria related to system modeling (set 3)

| System modeling | | Gradation of criteria | |
|---|---|---|---|
| Level | Criteria | ⊕ | ⊖ |
| g | Geometries | Free-form | Euler |
| h | Constraints | Non-linear | Linear |
| i | Modeling | Equation | Geometry |
| j | Dimension | 3D | 2D |

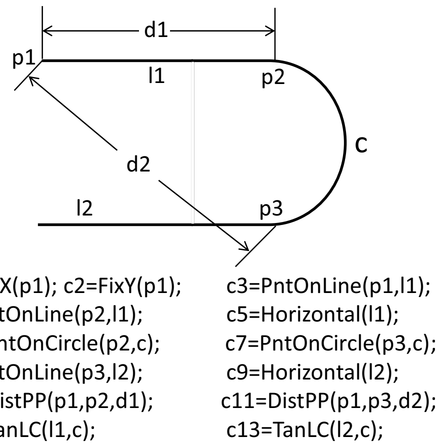



**Fig. 8** Singular configuration as described in [17]

**Table 6** Criteria related to the way of generating results (set 4)

| Results generation | | Gradation of criteria | |
|---|---|---|---|
| Level | Criteria | ⊕ | ⊖ |
| k | Way of detection | Single-pass | Iteratively |
| l | Debugging | Yes | No |



rigid (non-rigid) before and after an infinitesimal perturbation [4]. A singular configuration, however, transforms from rigid (non-rigid) to non-rigid (rigid) after an infinitesimal perturbation. It happens when geometric elements are drawn with unspecified properties (collinearity, coplanarity, etc.). It may be the case that a solution of a decomposed system lies into a singular variety, e.g., includes some unspecified collinearity or coplanarity. In this case, it happens that the generically solvable components are no more solvable. For instance, the doublebanana geometry (Fig. 2) is generically over-constrained but becomes under-constrained if the height of both bananas is the same since the two "bananas" can fold continuously along the line passing through their extremities. Moreover, the Jacobian matrix at singular configurations is rank deficiency, which introduces dependences between constraints. For example, the Jacobian matrix of the subsystem $\{p3, l2, c, c7, c8, c13\}$ of the Fig. 8 is of size $7 \times 7$ . But its rank is 5, which is a singular configuration. Obviously, there is no redundant constraints and the singularity comes from the tangent constraints between $c$ and $l1$, $l2$ [17].

### 3.3 Criteria Related to the System Modeling

This set of criteria characterize detection approaches with respect to system modeling: the type of geometries ($g$) and constraints ($h$), modeling at the level of equations or geometries ($i$), 3D or 2D space ($j$). The first criterion characterizes the type of geometries. Currently, geometric entities are either Euler geometries (g⊖) such as line segments, cylinders, spheres etc. or free-form geometries (g⊕). The second criterion deals with linear (h⊖) and non-linear (h⊕) constraints. The third criterion describes a system either at the level of equations (i⊕) or geometries (i⊖). Finally, a modeling system can either be in 2D (j⊖) or 3D (j⊕) space (Table 5).

### 3.4 Criteria Related to the Results Generation

In reality, a designer may require that a modeler outputs geometric over-constraints iteratively when modeling a geometric system interactively. Iteratively means the method enables to generate results through steps/loops (k⊖) while single-pass methods generate the results all at once (k⊕ ). Also, a user-friendly method should enable the treatment of results for debugging purpose (l⊕). That is, locate the results at the level of geometries so that users can modify/remove them (Table 6).

## 4 State-of-the-Art on the Detection of Over-Constrained Geometric Configurations

This section gathers together existing approaches that are capable of detecting geometric over-constraints. Approaches are classified with respect to the Definitions in Sect. 2. The Table 9 summarizes the final analysis results.



Journal : Large 11831 | Article No : 9509 | Pages : 25 | MS Code : 9509 | Dispatch : 23-7-2021

### 4.1 Methods Working at the Level of Geometries

This group of methods detect geometric over-constraints based on DoF analysis. Since these methods operate geometric entities directly, geometric information of the over-constraints are retained and thus easy to interpret.

*Reduction* Fudos and Hoffman [18] introduced a constructive approach to solve a constraint graph, where geometric entities are lines and points, geometric constraints are distances and angles. In their reduction algorithm, triangles are found and merged recursively until the initial graph is rewritten into a final graph. The structurally over-constrained system/subsystem are detected in two ways. Firstly, before finding triangles, the approach checks if the subgraph is structurally over-constrained. Secondly, if a 4-cycle graph is met during the reduction process, then the system is structurally over-constrained. A 4-cycle graph corresponds to two clusters sharing two geometric elements, which is structurally over-constrained.

Results of evaluating the method are as following:

- *Criteria set 1* Although the method allows for checking the constrained status of a system, it does not specify how to find the structural over-constraints as well as finding the spanning groups (a,c?). Since the method is structural, it is meaningless to distinguish redundant and conflicting constraints (b⊙).
- *Criteria set 2* The method enables to identify a 4-cycle graph which is structurally over-constrained (d⊕). Also, the triangles found during the recursive process are the rigid subsystems (e⊖). In terms of dealing with singular configurations, it is not mentioned in the original paper (f?).
- *Criteria set 3* Normally, a constraints system is composed of Euler geometries (g⊖) with non-linear constraints (distances, angles h⊕) and modeled at the level of geometries (i⊖) in 2D space (j⊖).
- *Criteria set 4* Since detecting geometric over-constraints are not addressed, there is no meaning discussing how the over-constraints are generated (k⊙) as well as debugging them (l⊙).

*Dense* Hoffman et al adapted their *Dense* algorithm [19] to locate 1-overconstrained subgraph (satisfying $DOCs > DOFs - D + 1$) of 1-overconstrained graph [20]. The algorithm is composed of four main steps.

1. overloads the capacity from one arc from the source to a constraint by $D + 2$.
2. distributes a maximum flow in the overloaded network.
3. finds subgraph of density $\geq -D + 1$, where the density of a subgraph $A : d(A) = DOCs(A) - DOFs(A)$.
4. locates a minimal 1-overconstrained subgraph by deleting vertices one by one.

As it is shown in the Fig. 9, generally locating a minimal subgraph of density $-D + 1$ is done as follows: first, by distributing an flow of weight $Doc_i + D + 2$ from each constraint to its end points(entities) to find a subgraph of density $-D + 1$. Such dense graph is found when there exists an edge whose edge cannot be distributed with redistribution [21]. The algorithm continues to locate minimal 1-over-constrained subgraph. But in our opinion, to check whether a system is over-constrained or not, it is sufficient that the algorithm terminates at step 3. The authors suggested to further extend the algorithm to incrementally detect k-Over-constrained graphs. The algorithm allows for updating constraints efficiently. Once the constraints are identified, they are removed. However, the algorithm excludes large geometric structures that have rotational symmetry.



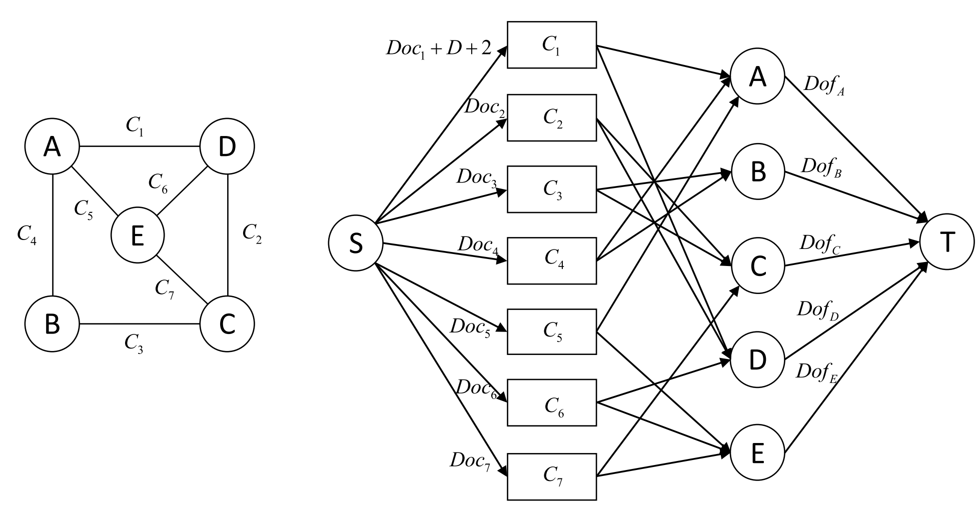



**Fig. 9** Left: The constraint graph with 5 entities and 7 constraints. Right: The flow network derived from the bipartite graph, where source S is linked to each constraint (capacity correspond to $Doc_i$ of a constraint $i$) and each entity is linked to the sink T (capacity correspond to DoF of an entity)



| Journal : **Large 11831** | Article No : **9509** | Pages : **25** | MS Code : **9509** | Dispatch : **23-7-2021** |
|---|---|---|---|---|

Results of evaluating the method are:

- *Criteria set 1* The method does not specify neither detecting geometric over-constraints nor the spanning groups (a,c?). Since the method is structural, talking about distinguishing redundant and conflicting constraints is meaningless (b⊙).
- *Criteria set 2* The algorithm locates the 1-overconstrained subgraph (d⊕) rather than rigid subsystems (e⊙ ). Regarding the singular analysis of a system, it is not addressed by the method (f?).
- *Criteria set 3* The evaluation of this set of criteria on the method is the same with the previous's one except that the modeling dimension can be both 2D and 3D (j⊕⊖).
- *Criteria set 4* Since detecting geometric over-constraints is not addressed, there is no meaning to discuss how the over-constraints are generated (k⊙) as well as debugging them (l⊙).

*Over-rigid* Hoffmann's algorithm cannot deal with constraints such as alignments, incidences and parallelisms either generic or non-generic. Based on their work, Jermann et al [9] proposed the *Over-rigid* algorithm with the following modifications:

1. The overload is applied on a virtual node *R* (Fig. 10 and Fig. 11) whereas in the *Dense* algorithm, it is applied on a constraint node.
2. The overload is $Dor + 1$ and the computation of *Dor* depends on the subsets of constraint entities to which *R* is attached. For example, $Dor(A, B)$ ($\{A, B\}$ is the subset of the configuration in the Fig. 10) is different from $Dor(C, D, E)$ ($\{C, D, E\}$ is the subset of the configuration in the Fig. 11). But in the *Dense* algorithm, the overload is invariant with different subsystems.
3. The *R* node is attached to *Dor-minimal* subsets of objects in order to find over-rigid subsystems.

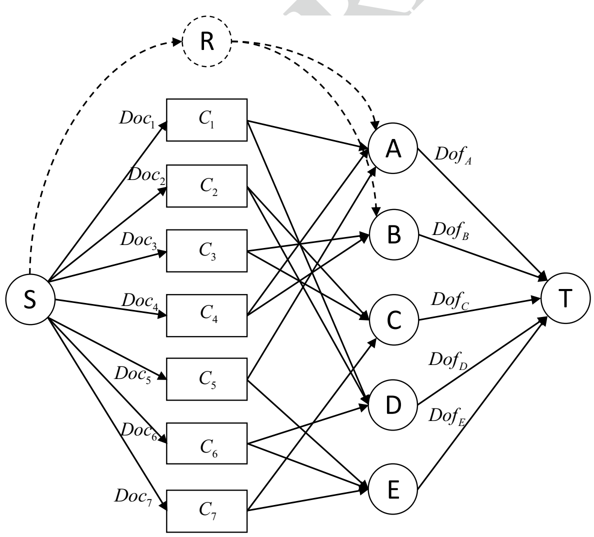



**Fig. 10** Overloading to subset {A,B}

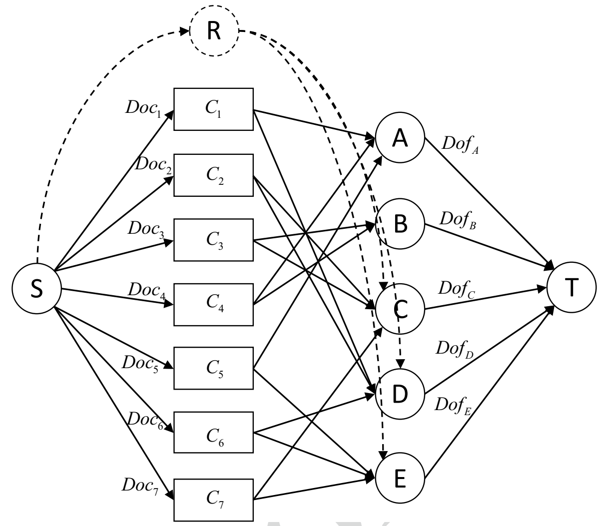



**Fig. 11** Overloading to subset {C,D,E}

The *Dor* varies with different subsystems. The *Over-rigid* algorithm is initially designed to check whether a system is structurally well-constrained or not. However, the authors do not show specificly how to detect structurally over-constrained systems as Hoffmann et al did in the *Dense* algorithm. Since it modified the *Dense* algorithm, it can be adapted to detect over-constrained systems if setting the overload to $Dor + 2$, which follows the steps 1-3 of the *Dense* algorithm. The evaluation of adapted version of the *Over-rigid* algorithm is the same as the modified version of the *Dense* algorithm.

*MWM* Latham et al [10] detected over-constrained subgraphs with DoF-based analysis by finding a maximum weighted matching (MWM) of a bipartite graph. The method decomposes the graph into minimal connected components which they called balanced sets. If a balanced set is in a predefined set of patterns, the subproblem is solved by a geometric construction, otherwise a numeric solution is used. The method addresses symbolic constraints and enables to identify under- and over-constrained configurations.

As it is shown in the Fig. 12, a constraints system is initially represented with a constraint graph with two classes of nodes representing DoFs of geometric entities and constraints respectively. Then, it is transformed into a directed graph by specifying directions from constraints nodes to entities nodes. After that, maximum matching between

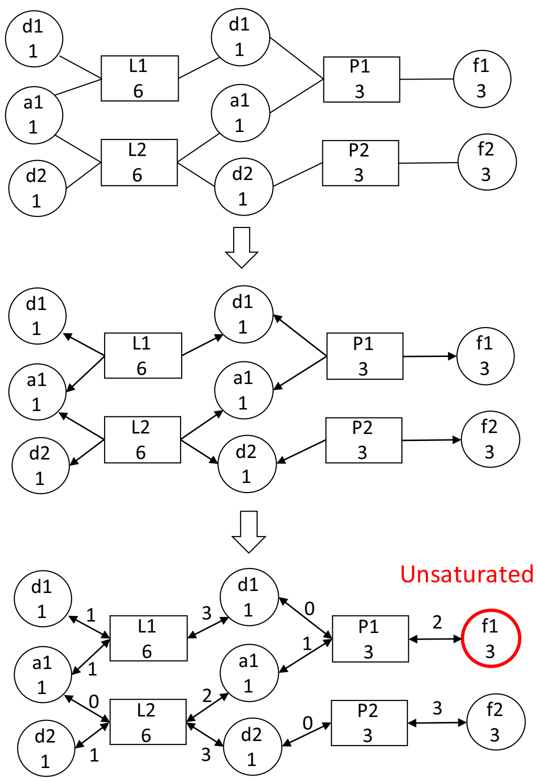



**Fig. 12** Over-constraints detection process of an example taken from [10]. The unsaturated node is the geometric over-constraint

constraints and entities is applied and those unsaturated constraints nodes are geometric over-constraints. Moreover, they addressed the over-constrained problems by prioritizing the given constraints, where over-constraints can automatically be modified based on constraints priorities.

Results of evaluating the method are:

- *Criteria set 1* The unsaturated constraints are geometric over- constraints (a⊖). Since the detected over-constraints are structural, there is no meaning to discuss redundant and conflicting constraints as well as the spanning groups (b,c⊙).
- *Criteria set 2* The subgraph containing an unsaturated constraint node is the over-constrained component (d⊕). It can be found by tracing the descendant nodes of the unsaturated node. Moreover, the algorithm enables a decomposition of the system into balanced subsets which are rigid subsystems (e⊖). Also, analyzing the singular configurations is not discussed (f?).
- *Criteria set 3* The results of evaluation with respect to this set of criteria are the same with those of the *Over-rigid* algorithm except the whole system is modeled in 3D space (j⊕).
- *Criteria set 4* The over-constraints are detected in the single-pass way (k⊕). And they proposed to modify the constraints according to constraints priorities (l⊕).

## 4.2 Methods Working at the Level of Equations

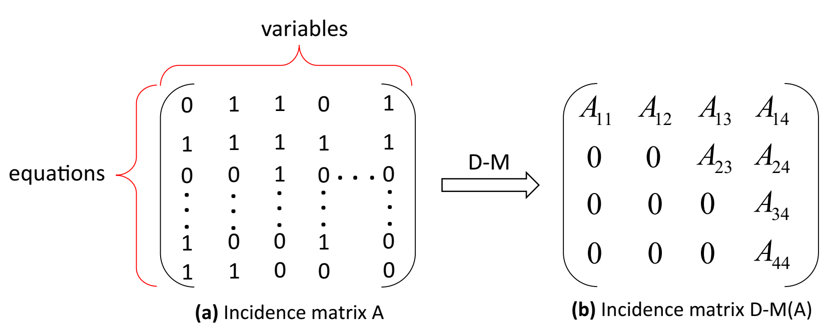
### 4.2.1 Linear Methods

In general, almost all the geometric constraints can be translated mechanically into a set of algebraic equations [19]. Therefore, detecting geometric over-constraints is equivalent with identifying a set of conflicting/redundant equations. However, even if detection works at the level of equations, the treatment needs to be done at the level of geometries.

*D-M* A variation of the Latham's method directly deals with algebraic constraints, where a maximum cardinality of bipartite matching is used. The D-M algorithm decomposes system of equations into smaller subsystems by transforming equations system into a bipartite graph and canonically decomposes the bipartite graph through maximum matching and minimum vertex covers. It decomposes a system into over-constrained, well-constrained and under-constrained subsystems [22]. It has been used for debugging in equation-based modeling systems such as the Modelica [23]. Serrano used graph-theoretic algorithm to detect over-constrained systems where all constraints and geometric entities are of DoF one [24].

The process of the D-M decomposition: D-M(A) = $A(p, q)$ does not require $A$ need to be square or full structural rank (Fig. 13). $A(p, q)$ is split into a 4-by-4 set of coarse blocks: where $A12$, $A23$, and $A34$ are square with zero-free diagonals. The columns of $A11$ are the unmatched columns, and the rows of $A44$ are the unmatched rows. Any

**Fig. 13** D-M decomposition of incidence matrix *A*



of these blocks can be empty. The whole decomposition is composed of coarse and fine decomposition.

*Coarse decomposition*

- [$A11$ $A12$] is the under-constrained part of a system and it is always rectangular and with more columns than rows, or does not exist.
- $A23$ is the well-constrained part of a system and it is always square.
- [$A34$; $A44$] is the over-constrained part of a system and it is always rectangular with more rows than columns, or does not exist.

*Fine decomposition* The above sub-matrices are further divided into block upper triangular form via the fine decomposition. Consequently, strong connected components are generated and linked with solving order [11]. By analyzing each component following the solving order, the system is updated dynamically and over-constraints are generated iteratively. Results of evaluating the method are:

- *Criteria set 1* Equations of [$A34$; $A44$] are structural over-constraints (a⊖). Evaluation of criteria b and c is meaningless since the method is structural (b,c⊙).
- *Criteria set 2* [$A34$; $A44$] after coarse decomposition is the structural over-constrained subpart (d⊕). The strong connected components after fine decomposition are structural rigid subsystems (e⊖). The method does not discuss on the analysis of singular configurations (f⊖).
- *Criteria set 3* Since the modeling is based on system of equations, any geometric constraints that are able to be transformed into system of equations can be analyzed by the method. Therefore, the results for evaluating the method according to this set of criteria are (g⊕⊖, h⊕⊖, i⊕, j⊕⊖).
- *Criteria set 4* Structural over-constraints are contained in the over-constrained part and output in a single-pass way (k⊕). The method does not discuss on debugging the over-constraints (l?).

In this section, we gather together the approaches from linear algebra that are capable of analyzing linear system of constraints. We consider linear system of constraints in the matrix form $Ax = b$, where A has dimension $m \times n$ , and $n \geq m \geq r$ with $r$ being the rank. The notation $A[i : j, l : k]$ defines the matrix obtained by slicing the $i$th to $j$th rows, and the $l$th to $k$th columns of $A$. According to [25], methods such as Gauss-Jordan Elimination, LU and QR Factorization present a good characteristic of locating inconsistent/redundant equations.

*G-J* The elimination process is terminated once a reduced row echelon form is obtained (An example is shown in the Fig. 14). Exchanging rows at the start of the $k$th stage to ensure that:

$$\left|A_{kk}^{(k)}\right| = \max_{i \geqslant k} \left|A_{ik}^{(k)}\right| \tag{5}$$

where $A_{ik}^{(k)} = A[i, k]$, an element of the $i$th row and the $k$th column in $A$.

Numerical over-constraints are identified by searching lines containing only 0s. The vector $b$ is updated to $b_{new}$ when transforming $[A, b]$. The last $m - r$ values of the $b_{new}$ allow to further distinguish redundant (equal to 0) and conflicting (not equal to 0) constraints.

Results of evaluating the method are as follows:

- *Criteria set 1* The method allows for detecting redundant and conflicting constraints (a,b⊕). However, the method does not tell how to find the spanning groups of an over-constraint (c?).



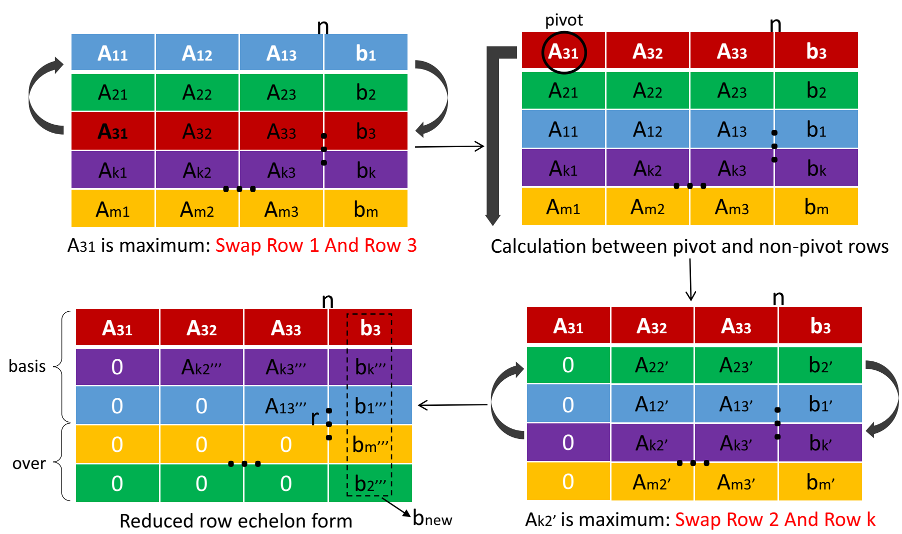



**Fig. 14** Gauss elimination with partial pivoting



Journal : Large 11831 | Article No : 9509 | Pages : 25 | MS Code : 9509 | Dispatch : 23-7-2021

- *Criteria set 2* The method does not enable to decompose a system. There is no meaning to evaluate the method with respect to system decomposition criteria (d,e,f⊙).
- *Criteria set 3* The method analyzes linear equations. Therefore, any geometry (g⊕⊖) with linear constraints (h⊖) in 3D or 2D space (j⊕⊖) modeling at the equation level (i⊕) can be handled by the method.
- *Criteria set 4* The over-constraints are output all at once (k⊕) after detection. The method does not discuss on debugging the overconstraints (l?).

In the work of [26], they used this method to detect invalid dimensioning schemes. Note that, in the following sections, G-J is short for the Gauss-Jordan elimination with partial pivoting method.

*LU* The method is a *high-level* algebraic description of the G-J [27]. The process is shown in the Fig. 15, where $P$ is the permutation matrix reordering the rows. The number of non-zero diagonal elements of $U$ is the rank $r$. The last $m-r$ rows of the reordered matrix $P*A$ corresponds to the numerical over-constraints.

However, the factorization itself does not manipulate directly on $b$, which means that distinguishing redundant and conflicting constraints is unavailable. To know them, we need further extension:

$$\left.\begin{aligned} Ax &= b \\ PA &= LU \end{aligned}\right\} Ux = L^{-1}Pb \tag{6}$$

Now the distinguish step is similar to the one of the G-J. That is, by comparing the last $m-r$ elements of $L^{-1}Pb$ with 0, redundant and conflicting constraints can be distinguished. However, one has to notice that the deduction process is under the condition that $L$ should be invertible.

To the best of our knowledge, using this method to detect geometric over-constraints is not convincingly demonstrated in literature. Evaluations of the method with respect to the criteria is the same as the ones of the G-J. Note that in the following sections, we use LU in short for the LU factorization with partial pivoting method.

*QR* Before applying the QR factorization method, coefficients matrix $A$ should be transposed first($A = A^t$) since it operates on columns of a matrix. The QR factorization exchanges columns at the start of the $k$th stage to ensure that:

$$\left\|A_k^{(k)}(k:m)\right\|_2 = \max_{j\geqslant k}\left\|A_j^{(k)}(k:m)\right\|_2 \tag{7}$$

where $A_j^{(k)}(k:m) = A[k:m,j]$

As it is shown in the Fig. 16, P is the permutation matrix where the information of exchanging columns is stored. $R$ is a triangular matrix where rank $r$ is the number of non-zeros diagonal elements. Equations corresponding to $A^t.p[:,r+1:m]$ are the over-constraints [28].

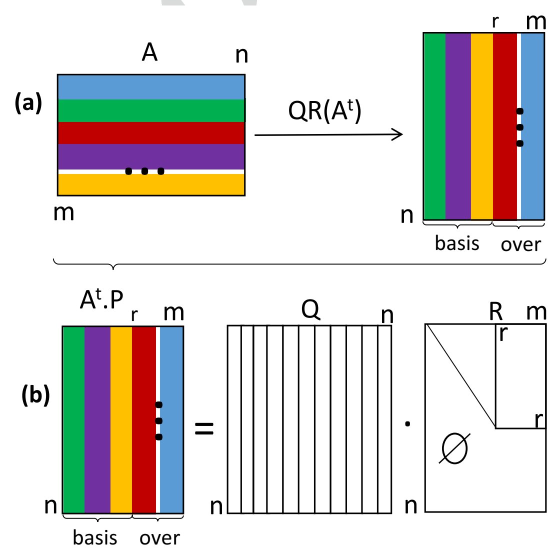
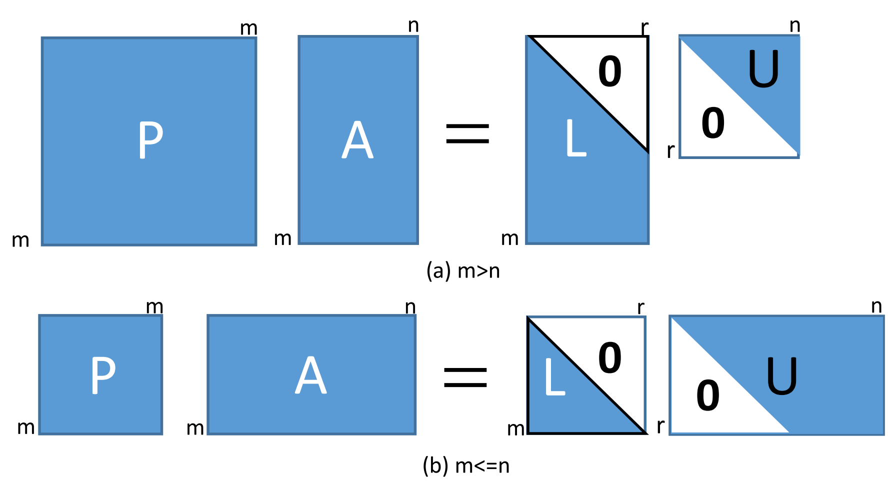



**Fig. 16** QR Factorization with column pivoting

**Fig. 15** LU Factorization with partial pivoting








| | | | | |
|---|---|---|---|---|
| Journal : **Large 11831** | Article No : **9509** | Pages : **25** | MS Code : **9509** | Dispatch : **23-7-2021** |

Similar to the LU, further deduction is needed to distinguish redundant and conflicting constraints. First, the matrix $Q(:,1:r)$ is inverted using the following equation:

$$A^t(:,1:r) = Q(:,1:r).R(1:r,1:r) \tag{8}$$

and is then used in the following equation:

$$A^t(:,r+1:n) = Q(:,1:r).R(1:r,r+1:n) \tag{9}$$

thus providing the following relationship between the two sliced matrices $A^t(:,r+1:n)$ and $A^t(:,1:r)$:

$$A^t(:,r+1:n) = A^t(:,1:r).R(1:r,1:r)^{-1}R(1:r,r+1:n) \tag{10}$$

The relationship between over-constraints and independent constraints are revealed in the matrix $R(1:r,1:r)^{-1}R(1:r,r+1:n)$ in the equation 10. From the matrix, the spanning group of an over-constraint could also be known. To identify the redundant and conflicting equations, the new $b$ vector after factorization is redefined as follows:

$$b_{new} = b(r+1:n) - b(1:r).R(1:r,1:r)^{-1}R(1:r,r+1:n) \tag{11}$$

Redundant and conflicting equations can be further distinguished by checking whether the value of the last $m-r$ elements of $b_{new}$ is 0 or not.

The method is adopted by Hu et al [12] to detect over-constraints of free-form constraints systems. Evaluation of the method with respect to the criteria is the same as the one of the G-J. We use QR in short for the QR factorization method in the following sections.

### 4.2.2 Non-Linear Methods

Detecting non-linear geometric constraints systems is more complicated than linear ones. Since non-linear detection methods such as symbolic methods using abstract algebra, we introduce the mathematical fundamentals to make following discussions easy to understand. The following two theorems are induced from [29]. Readers can find more details about concepts like ideals, affine varieties etc. in the book.

**Theorem 1** *For a system of polynomial equations $f_0 = f_1 = \cdots = f_s = 0$, where $f_0, f_1, \ldots, f_s \in \mathbb{C}[x_1, \ldots, x_n]$; If affine variety $W(f_1, \ldots, f_s) \neq \emptyset$ while $W(f_0, f_1, \ldots, f_s) = \emptyset$, then $f_0 = 0$ is a conflicting equation; If $W(f_0, f_1, \ldots, f_s) = W(f_1, \ldots, f_s) \neq \emptyset$, then $f_0 = 0$ is a redundant equation; If $W(f_0, f_1, \ldots, f_s) \neq \emptyset$, $W(f_1, \ldots, f_s) \neq \emptyset$ and $W(f_0, f_1, \ldots, f_s) \neq W(f_1, \ldots, f_s)$, then $f_0 = 0$ is an independent equation.*

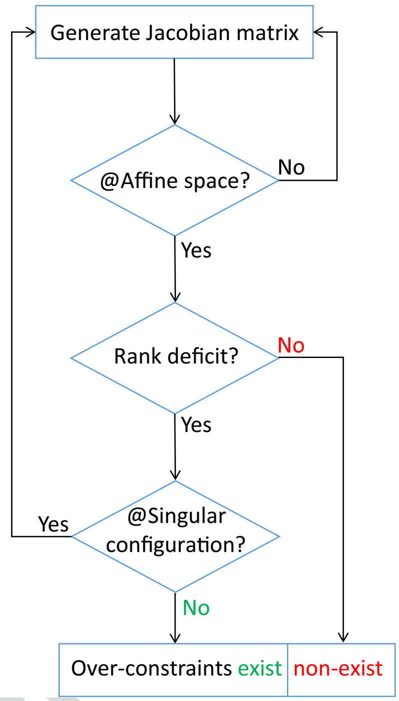



**Fig. 17** Over-constraints detection based on the Jacobian matrix analysis

**Theorem 2** (Hilbert's weak Nullstellensatz) *Let $k$ be an algebraically closed field. If $f, f_1, \ldots, f_s \in k[x_1, \ldots, x_n]$ are such that $f \in I(W(f_1, \ldots, f_s))$, then there exists an integer $m \geq 1$ such that $f^m \in \langle f_1, \ldots, f_s \rangle$ (and conversely).*

Based on the Theorem 2, Michelucci et al [15] deduced the Corollary 1.

**Corollary 1** *Let $k$ be an algebraically closed field and $W(f_1, \ldots, f_s) \neq \emptyset$. If $f, f_1, \ldots, f_s$ have the common root $w$, then $rank([f'(w), f_1'(w), \ldots, f_s'(w)]^T) < s+1$.*

Informally, the corollary 1 tells that if a system of polynomial equations containing redundant equations, then the Jacobian matrix of the equations at the affine space (solution space) must be row rank deficiency. However, the reverse is not correct. In other words, if there exists the Jacobian matrix whose rank is deficiency at the solution space, then system of polynomial equations does not necessarily contain redundant equations. A typical example is the singular configuration in the Fig. 8: the





Journal : Large 11831 | Article No : 9509 | Pages : 25 | MS Code : 9509 | Dispatch : 23-7-2021

system is row rank deficiency at the solution space but the system does not contain geometric over-constraints. It is the singular configuration that causes the system rank deficiency. Therefore, to detect over-constraints by analyzing the Jacobian matrix, one has to note that the Jacobian matrix should be computed on configurations in the solution space rather than singular configurations.

We propose a schema on determining the existence of over-constraints through analyzing the Jacobian matrix in the Fig. 17. That is, analyze the Jacobian matrix at a configuration from affine space. If the rank is full, then there is no over-constraints. Otherwise, we check if the configuration is singular. If not, then over-constraints exist otherwise we move to test other configurations in the affine space. Loops mean that one has to go back to generate the Jacobian matrix at different configurations until the existence/non-existence of over-constraints is determined. It is a recursive process of finding configurations that can be used to determine the existence/non-existence of over-constraints. In reality, however, affine space is sometimes hard to find or does not exist. Moreover, the singularity of a configuration is difficult to test in some cases. Several methods have been proposed to address the two issues.

The first group of methods are symbolic algebraic methods, which compute the Grobner basis for a system of equations. Algorithms proposed include works of the Buchberger [30], and the Wu-Ritt [31, 32].

*Grobner basis* Assume a set of polynomials $f_0, f_1, \dots, f_s \in \mathbb{C}[x_1, \dots, x_n]$. The reduced Grobner basis ($rgb_0$) of the ideal $\langle f_1, \dots, f_s \rangle$ satisfies $rgb_0 \neq \{1\}$ and $rgb_0 \neq \{0\}$ with respect to any ordering. The new reduced Grobner basis of the ideal $\langle f_0, f_1, \dots, f_s \rangle$ is $rgb_{new}$. If $rgb_{new} = \{1\}$, $f_0 = 0$ is a conflicting equation; if $rgb_{new} \equiv rgb_{old}$, $f_0 = 0$ is a redundant equation(b⊕); if $rgb_{old} \subset rgb_{new}$, $f_0 = 0$ is an independent equation [33].

Results of evaluating the method are as follows:

- *Criteria set 1* Obviously, the above method can tell if a constraint is redundant or conflicting (a,b⊕). However, the method does not enable to find the spanning group of the constraint (c⊖).
- *Criteria set 2* The method is used to solve polynomial equations. Therefore, there is no meaning to evaluate it with the set of criteria on system decomposition (d,e⊙). The method does not analyze the singularity of a configuration (f⊖).
- *Criteria set 3* The method analyzes non-linear equations. Therefore, any geometries (g⊕⊖) with non-linear constraints (h⊕) in 3D or 2D space (j⊕⊖) modeling at the level of equations (i⊕) can be applied with the method.
- *Criteria set 4* To detect a set of over-constraints, equations are input one by one in the process of computing the reduced Grobner basis. Therefore, the over-constraints set are generated iteratively (k⊖). However, debugging these over-constraints is not discussed (l?).

Construction of a Grobner basis is a time-consuming process. Hoffman et al used the method to do geometric reasoning between geometric configurations [34]. In terms of detecting geometric over-constraints, Kondo et al [35] initially used it to test dependencies among constraints in 2D dimension.

*Wu-Ritt* Let a system of polynomial equations $P = \{f_0 = f_1 = \cdots = f_s = 0\}$, where $f_0, f_1, \dots, f_s \in \mathbb{Q}[x_1, \dots, x_n]$ represent system of constraints and Zero(P) denote the set of all common zeros of $\{f_0, f_1, \dots, f_s\}$. The system contains redundant equations only if there exists a polynomial $p$ such that:

$$Zero(P - \{p\}) \equiv Zero(P) \tag{12}$$

For the polynomial set $P$, its zero set can be decomposed into a union of zero sets of polynomial sets in triangular form using the *Wu-Ritt's zero decomposition algorithm*:

$$Zero(P) = \bigcup_{1 \leqslant i \leqslant k} Zero(TS\{i\}/I\{i\}) \tag{13}$$

where each $TS\{i\}$ is a polynomial set in triangular form, $I\{i\}$ is the production of initials of the polynomials in $TS\{i\}$ and $k$ is the number of zero sets. The system contains inconsistent equations iff $k \equiv 0$ [36]. In the work of Gao and Chou [37], they presented a complete method to identify conflicting and redundant constraints based on the Wu-Ritt's decomposition algorithm. Also, the algorithm can be used to solve the Pappus problems to decide if a configuration can be drawn with ruler and compass.

Results of evaluating the Gao's method are as follows:





Journal : Large 11831 | Article No : 9509 | Pages : 25 | MS Code : 9509 | Dispatch : 23-7-2021

- *Criteria set 1* As discussed above, their method enable to detect conflicting and redundant constraints (a,b⊕) but cannot find the spanning groups (c⊖).
- *Criteria set 2* The method decomposes a set of polynomials into a union of zero sets in triangular form. No over-constrained subparts, rigid subsystems are generated as well as singular configurations are analyzed (d,e,f⊖).
- *Criteria set 3* The method analyzes non-linear equations. Any geometries (g⊕⊖) with non-linear constraints (h⊕) in 3D or 2D space (j⊕⊖) modeling at the equation level (i⊕) can be applied with the method.
- *Criteria set 4* The results are the same as those of evaluating the Grobner basis method.

Symbolic detection methods are sound in theory but the computation cost is high. As discussed previously, the worstcase can be doubly exponential. Moreover, the reduced Grobner basis has to be computed every time of analyzing an equation. Therefore, this method is limited to deal with large systems of equations.

The second group of methods analyze the Jacobian matrix of a system of equations. Different from symbolic methods, these numerical methods are more practical in computation but are theoretical deficiency in some cases. On one hand, if the affine space of a system does not exist, an equivalent one that sharing similar the Jacobian structure should be found. On the other hand, even if the Jacobian matrix of a configuration is row rank deficiency in affine space, the corresponding configuration should not be singular.

*Perturbation* Haug proposed a perturbation method to deal with singular configurations and detect redundant constraints in mechanical systems [38]. More precisely, assuming a system of equations $\boldsymbol{\Phi}(q) = 0$ and the corresponding Jacobian matrix $\boldsymbol{\Phi}_q$ is rank deficiency at $q$. As we discussed before, it is not sufficient to determine the existence of the over-constraints since the singular configuration can also make a Jacobian matrix rank deficiency. He suggested to analyze the Jacobian matrix at more configurations with the following:

- Add a small perturbation $\delta q$ to $q$ and obtain $\boldsymbol{\Phi}_q \delta q = 0$ . The process is based on the Implicit Function Theorem [39].
- Applying the G-J to $\boldsymbol{\Phi}_q$ , $\boldsymbol{\Phi}_q \delta q = 0$ is transformed into $\begin{bmatrix} \boldsymbol{\Phi}_u^I & \boldsymbol{\Phi}_v^I \\ 0 & \boldsymbol{\Phi}_v^R \end{bmatrix} \begin{bmatrix} \delta u \\ \delta v \end{bmatrix} = 0$. $\boldsymbol{\Phi}_u^I$ is the upper triangular matrix with 1s as diagonal elements. $\boldsymbol{\Phi}_v^R$ can be treated as the matrix with all 0s under given tolerance. Equations in $\boldsymbol{\Phi}(q) = 0$ corresponding to $\boldsymbol{\Phi}_u^I$ part: $\boldsymbol{\Phi}^I(q) = 0$ are independent.
- Now, $\boldsymbol{\Phi}_q \delta q = 0$ can be simplified into $\boldsymbol{\Phi}_u^I \delta u + \boldsymbol{\Phi}_v^I \delta v = 0$ and thus $\delta v = -(\boldsymbol{\Phi}_v^I)^{-1} \boldsymbol{\Phi}_u^I \delta u$ , $\delta q = \begin{bmatrix} \delta u \\ \delta v \end{bmatrix} = \begin{bmatrix} \delta u \\ -(\boldsymbol{\Phi}_v^I)^{-1} \boldsymbol{\Phi}_u^I \delta u \end{bmatrix}$
- Assume $q$ is perturbed to new point $q^*$ satisfying $q^* = q + \delta q$. To ascertain it lies in the affine space, is should satisfy $\boldsymbol{\Phi}(q*) = 0$. This is equivalent to $\boldsymbol{\Phi}^I(q*) = 0$ since the latter is composed of all independent equations of the former.
- Solving $\boldsymbol{\Phi}^I(q*) = 0$ , $q^* = q + \delta q$ , $\delta q = \begin{bmatrix} \delta u \\ \delta v \end{bmatrix} = \begin{bmatrix} \delta u \\ -(\boldsymbol{\Phi}_v^I)^{-1} \boldsymbol{\Phi}_u^I \delta u \end{bmatrix}$, the value of $q^*$ is obtained.
- Computing the rank of the Jacobian matrix at $q^*$: $\boldsymbol{\Phi}_{q^*}$ and checking if it is rank deficiency.

As we can see from the above, obtaining an appropriate value of the perturbation $\delta q$ so that $q^*$ lies in the affine space is the main part of the method.

Results of evaluating the method are as follows:

- *Criteria set 1* The method enables to detect geometric over-constraints (a⊕) but does not distinguish redundant and conflicting constraints (b⊖). Finding the spanning groups is also not supported (c⊖).
- *Criteria set 2* The method mainly detects the over-constraints based on analyzing the Jacobian matrix of a whole system. There is no meaning to evaluate the method with respect to system decomposition criteria (d,e,f⊙).
- *Criteria set 3* The method analyzes both linear and non-linear equation systems. Therefore, any geometries (g⊕⊖) with non-linear and linear constraints (h⊕⊖) in 3D or 2D space (j⊕⊖) modeling at the equation level (i⊕) can be applied with the method.
- *Criteria set 4* The over-constraints are generated in a single-pass way since the Jacobian matrix analysis is on the whole system at once (k⊕). However, debugging the over-constraints is not discussed (l?).

His method selects two points in the affine space to determine the existence of geometric over-constraints. If the Jacobian matrix at any configuration is full rank, then there is no over-constraint. However, if the rank of the Jacobian matrix at both configurations is deficiency, then there exists geometric over-constraints.





Journal : Large 11831 | Article No : 9509 | Pages : 25 | MS Code : 9509 | Dispatch : 23-7-2021

*NPM* Roots of system of equations can be sometimes hard to find or even do not exist. In these cases, the affine space does not exist. Sebti Foufou et al. [7] suggested a Numerical Probabilistic Method (NPM), which is to test the Jacobian matrix at random configurations instead of the affine space. However, there is a risk that the Jacobian matrix is row rank deficiency at the chosen configuration and the corresponding configuration happens to be singular. They suggested to test more configurations to reduce the possibility of happening such case. Moreover, in order to get more confidence, authors suggested that testing at 10 different configurations should be sufficient. The NPM is practical in computation but is not sound in theory since the testing configurations are not necessarily all in affine space.

Results of evaluating the method are as follows:

- *Criteria set 1* The method enables to identify numerical over-constraints (a⊕). However, it can neither distinguish redundant and conflicting constraints nor finding the spanning group of an over-constraint (b,c⊖).
- *Criteria set 2* The method can be used to decompose a system into rigid subsystems (e⊖). However, decomposition into over-constrained components as well as analyzing singular configurations are not supported (d,f⊖).
- *Criteria set 3* The method analyzes both linear and non-linear system of equations. Therefore, any geometries (g⊕⊖) with non-linear or linear constraints (h⊕⊖) in 3D or 2D space (j⊕⊖) modeling at the level of equations (i⊕) can be applied with the method.
- *Criteria set 4* Numerical over-constraints are detected all at once (k⊕) but debugging them is not discussed (l?).

*WCM* Instead of selecting configurations randomly, Michelucci et al. suggested to study the Jacobian structure at witness configurations where incidence constraints are satisfied [40]. A witness configuration and the target configuration shares the same Jacobian structure, where the Jacobian matrix is non-singular in affine space. As a consequence, all the numerical over-constraints can be identified [15]. More recently, Moinet et al. developed tools to identify conflicting constraints through analyzing the witness of a linearized system of equations [41]. Their approaches have been successfully applied to the well-known double banana geometry to find the numerical over-constraints.

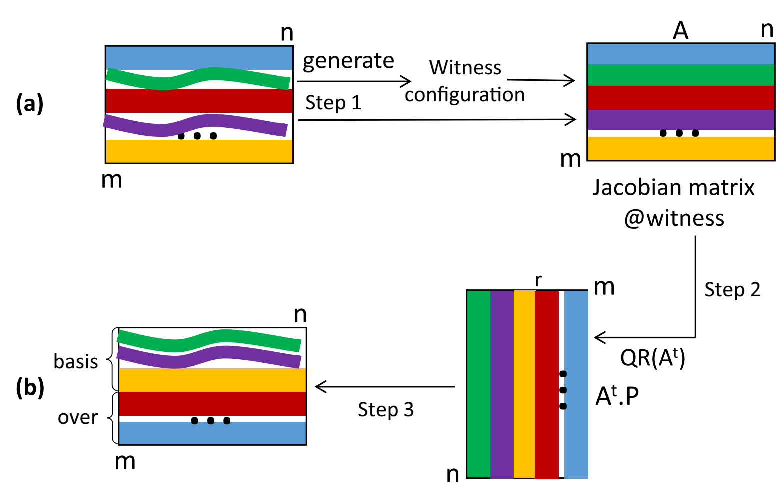
For a geometric constraints system represented with a set of equations $F(U, X) = 0$, where $U$ denotes a set of parameters with prescribed values $U_T$($T$ for target), and $X$ is the vector of unknowns. The solution is denoted as $X_T$. A witness is a couple $(U_W, X_W)$ such that $F(U_W, X_W) = 0$. Most of the time, $U_W$ and $X_W$ are different from $U_T$ and $X_T$ respectively. The witness $(U_W, X_W)$ is not the solution but shares the same combinatorial features with the target $(U_T, X_T)$, even if the witness configuration and the target configuration lie on two distinct connected components of the solution set. Therefore, analyzing a witness configuration enables to detect numerical over-constraints of a system [42, 43]. These numerical over-constraints can be not only structural over-constraints but also geometric redundancies.

The Fig. 18 shows the witness method combining the QR for detection. Step one aims at generating the witness configuration while at step two, the QR is applied on the Jacobian matrix $A$. As a result, the rows of equations are re-ordered by $P$ and the number of basis constraints is revealed by $r$. Finally, coming back to the re-ordered original equations, the first $r$ equations are the basis constraints while the remaining ones are the numerical over-constraints. Note that, the QR can be replaced with the G-J in the process, which would generate results different from the ones of QR since the two methods adopt different sorting rows strategies.

The results of evaluating the method are the same as those of evaluating the NPM method. Michelucci et al [15] proved that the WCM can identify all the dependencies among constraints. In other words, if removing these dependent constraints, the remaining constraints are independent. However,



**Fig. 18** Witness configuration method






Journal : Large 11831 | Article No : 9509 | Pages : 25 | MS Code : 9509 | Dispatch : 23-7-2021

a report on the limitations of the WCM method was summarized in [46] recently. And the authors have presented a new decision support method to over-come these limitations [47].

*WCM extension* Thierry et al [44] extend the WCM method to incrementally detect over-constraints and thus to get a well-constrained system. Also, they designed the so called *W-decomposition* to identify all well-constrained subsystems, which manages to decompose systems that are non-decomposable by classic combinatorial methods.

Results of evaluating the method are as follows:

- *Criteria set 1* The results of evaluation within this set of criteria are the same with those of the NPM method.
- *Criteria set 2* The *W-decomposition* enables to efficiently identify the maximal well-constrained subsystems of an articulated system as well as further decompose a rigid system into well-constrained subsystems (e⊖) but finding over-constrained components is not discussed (d?). In terms of finding the spanning groups, it is not supported (f⊖).
- *Criteria set 3* The results are the same with those of the NPM method.
- *Criteria set 4* Working on the witness, the naive idea would be to try and remove constraints one by one and, at each step, compute the rank again to determine if a constraint is redundant with respect to the remaining set. However, the authors pointed out that the method is computational expensive. They considered an incremental construction of the geometric constraint system to identify a set of redundant constraints with no additional cost (k⊖). The method does not discuss on debugging over-constraints (l?).

*Generating a witness configuration* Sometimes, when certain geometries happen to be drawn with specific properties (collinearity, coplanarities, etc) without representing a real constraint, the sketch is not typical of the expected solution. A witness configuration should be generic when it remains rigid before and after infinitesimal perturbation. If a sketch is not rigid before perturbation, it will not be rigid after the perturbation [4]. For example, the sketch of the Fig. 19a) is not generic: a small perturbation on the dimensions of the bars will result in a non-rigid sketch shown in Fig. 19b). However, the sketch of the Fig. 19b) is generic: if a small perturbation is introduced, it will remain non-rigid. Usually, non-generic sketches are constituted with aligned line segments presented in the Fig. 19a). The collinearity will induce artificial redundancy between the constraints associated with the collinear vectors. As a result, before using the WCM, one has to make sure a witness configuration is typical of the expected solution.

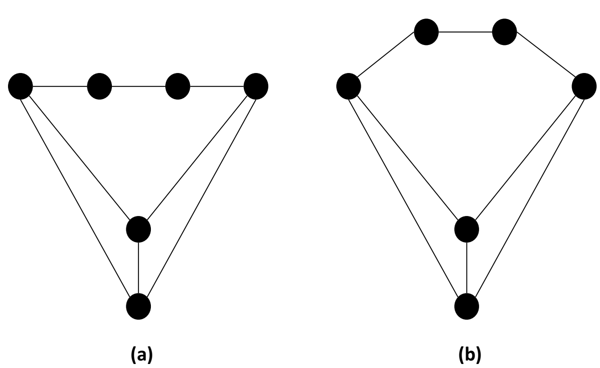



**Fig. 19** **a** Rigid sketch **b** Non-rigid sketch [44]

Here, we adopt the algorithm of Moinet [41] for generating generic witness configurations. Other methods for generating witness configurations can be found in [44, 45]. Moinet's algorithm contains the following steps:

1. Compute the Jacobian matrix for a system of equations.
2. Calculate the rank $r_{old}$ of the Jacobian matrix at the initial sketch.
3. Randomly perturb the initial sketch (usually generated by users), regenerate the Jacobian matrix, and recompute the rank $r_{new}$ at the new position (new sketch).
4. If $r_{new} > r_{old}$, replace the initial sketch with the new one and reiterate the third step.
5. Otherwise the old sketch is generic.

## 4.3 Incremental and Decremental Detection Frameworks

In real-life applications, debugging geometric constraints systems can be done in two different ways. On one hand, With CAD modelers, designers are able to detect over-constraints interactively during the modeling process in 2D sketches. Usually, constraints are added incrementally. On the other hand, the debugging process can be realized by analyzing system of constraints already exist. Here, all

**Table 7** Structural methods

| Level | Modeling | Method | Strong connected components |
|---|---|---|---|
| Equation | Bipartite graph | D-M | Irreducible subsystems |
| Geometry | Bipartite graph | MWM(Maximum Weighted Matching) | Balanced sets |

**Table 8** Algebraic methods

| | Linear method | Non-linear method |
|---|---|---|
| Over-constraints | WCM | WCM |
| Redundancies/conflicts | G-J/QR | Grobner basis/ incremental solving |

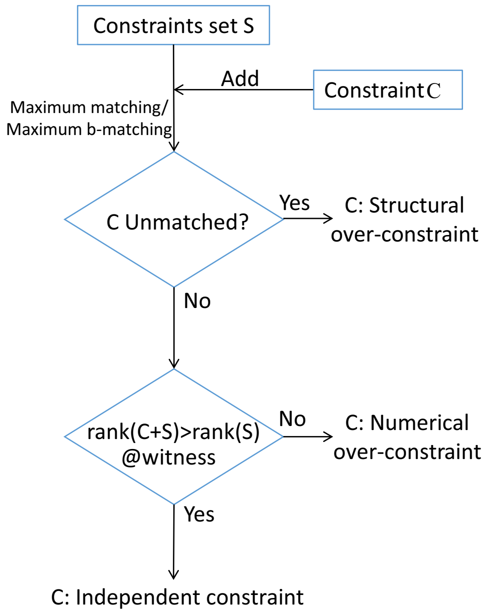



**Fig. 20** Incremental detection framework where constraints are added one by one

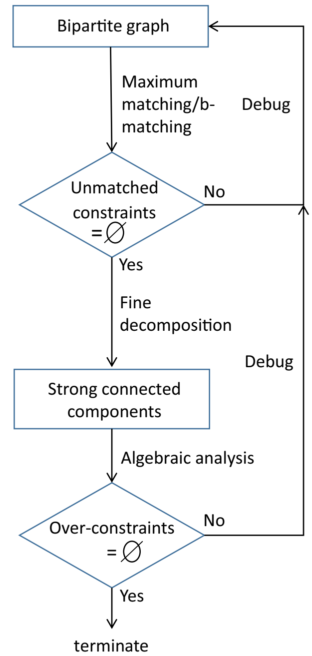



**Fig. 21** Decremental detection framework



the constraints and associated equations have been predefined. Through analyzing methods in previous sections, we propose two detection frameworks: incremental detection framework and decremental detection framework. Both of them are based on a combination of structural and algebraic methods. These methods are listed in the Table 7 and Table 8 respectively. Details of the two frameworks will be discussed in the next subsections.

#### 4.3.1 Incremental Detection Framework

Here, we assume that the constraint $C$ is to be added to a set of constraints $S$. This framework is to test if $C$ is an over-constraint with respect to $S$. The first method is either the D-M method or the MWM method, which detects structural over-constraints using either maximum matching method or maximum b-matching method. The method will be applied to the new group $S+C$ after adding $C$. If $C$ is unmatched, then $C$ is a structural over-constraint. Otherwise, we apply the WCM method to detect numerical over-constraints of $S+C$. If the rank of the new system $S+C$ is bigger than that of $S$ at witness configurations, $C$ is an independent constraint otherwise it is a numerical over-constraint. Whether it is redundant or conflicting can be checked using the Grobner basis method or the Incremental solving method. In this framework, since constraints are added incrementally, users can be informed directly if a newly inserted constraint has been detected as an over-constraint (Fig. 20).

The advantage of this framework is that it enables designers to detect and treat the over-constraints as soon as they are detected in the modeling process.

#### 4.3.2 Decremental Detection Framework

Decremental detection analyzes a set of existing constraints. The constraints set and its associated equations set are initially represented with a bipartite graph. Structural over-constraints will be identified using either the D-M method or the MWM method if there exists unmatched constraints after maximum matching (or b-matching). They will be removed and the system will be updated. If there is no unmatched constraints, strong connected components (that is, irreducible subsystems using the D-M method or balanced sets using the MWM method) are generated with fine decomposition

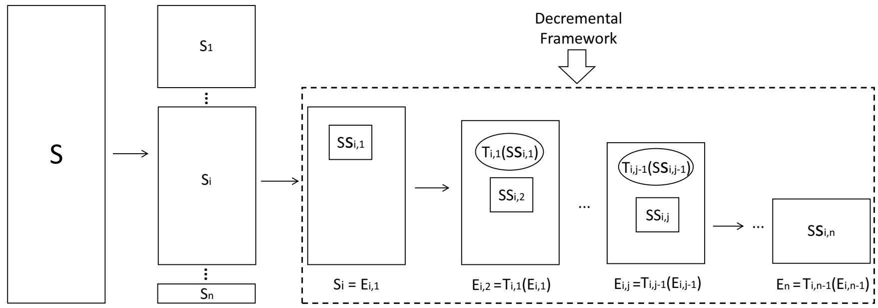



**Fig. 22** A Decomposition-Detection plan. $S_i$ is the $i$th component after decomposition; $S_{ij}$ is the component $S_i$ at $j$th step when analyzing; $E_{ij}$ refers to the entire system when analyzing $S_{ij}$

of the system. Then, algebraic methods are used to detect numerical over-constraints inside each component. Since strong connected components linked with solving order is actually a DAG structure, components corresponding to the source vertices are usually analyzed first. Once an over-constraint is found, it is presented to the user for debugging. After that, the system is updated and the corresponding bipartite graph is rebuilt. The detection process finishes when no numerical over-constraints are found (Fig. 21).

The advantage of this framework is that the treatment of the detected over-constraints can be performed on the entire system, which better considers the design intent. However, if a final system after modelling is too large, it would be preferable to detect over-constraints incrementally during the modeling process rather than analyze them decrementally after modeling.

#### 4.3.3 A Decomposition-Detection Plan

The first requirement of a Decomposition-Detection (D-D) plan is that it should be able to find local segments of a system of constraints if there exists any. For example, decomposes a free-form configuration into local parts due to the local support property of the geometry. Secondly, a D-D plan should decompose a constraint system into small subsystems and analyze these subsystems using algebraic methods. Since the time cost of over-constraints detection is proportional (at least polynomial) to the size of a system, these small subsystems should be as small as possible so that algebraic methods can analyze them with low computational cost. If there is no over-constraints in a subsystem, the subsystem would be solved and the solution would be propogated to the entire system resulting in a simplified system. As it is shown in the Fig. 22, a D-D plan initially decomposes the system $S$ into $\{S_1, \ldots, S_i, \ldots, S_n\}$ local segments. Then, for each local segment $S_i$, the D-D plan proceeds by applying the following steps at each iteration $j$:

1. Find the small subsystem $SS_{i,j}$ of the current local part $S_i$. Since small subsystems are linked with solving sequence, the ones that are the source of the sequence should be chosen first ($SS_{i,1}$).
2. Detect numerical over-constraints in $SS_{i,j}$ using algebraic methods. Users can either remove or modify them once they are detected. Otherwise, solve $SS_{i,j}$ directly using algebraic methods.
3. Replace $SS_{i,j-1}$ by an abstraction or simplification $T_{i,j-1}(SS_{i,j-1})$ as well as replacing the entire system $E_{i,j-1}$ by a simplification $E_{i,j} = T_{i,j-1}(E_{i,j-1})$. The simplification can be either the removal/modification of the over-constraints or solving $SS_{i,j-1}$ and propogating the solution to $E_{i,j-1}$. The latter operation can potentially generate over-constraints since the solution of $SS_{i,j-1}$ may cause some equations of $E_{i,j-1}$ satisfied or unsatisfied (Fig. 22).

The decremental framework can be adapted and incorporated into a D-D plan to analyze $S_i$. As such, $S_i$ is initially represented with a bipartite graph. $SS_{i,j}$ corresponds to the strong connected component of the $j$th iteration, which is to be analyzed by an algebraic method ($T_{i,j}(SS_{i,j})$). These analysis results could then be used to simplify $E_{i,j}$ through $T_{i,j}(E_{i,j})$ . As a result, $E_{i,j}$ is updated as $E_{i,j+1}$.

### 4.4 Hybrid Approaches

*Serrano* Serrano analyzed a system of equations (h⊕) to select a well constrained, solvable subsets from candidate constraints[24]. His method first detects structural over-constraints (a⊖) if there are equations uncovered after maximum matching. To further detect numerical over- constraints (a⊕) within strong connected components (e⊖), symbolic and





Journal : Large 11831 | Article No : 9509 | Pages : 25 | MS Code : 9509 | Dispatch : 23-7-2021

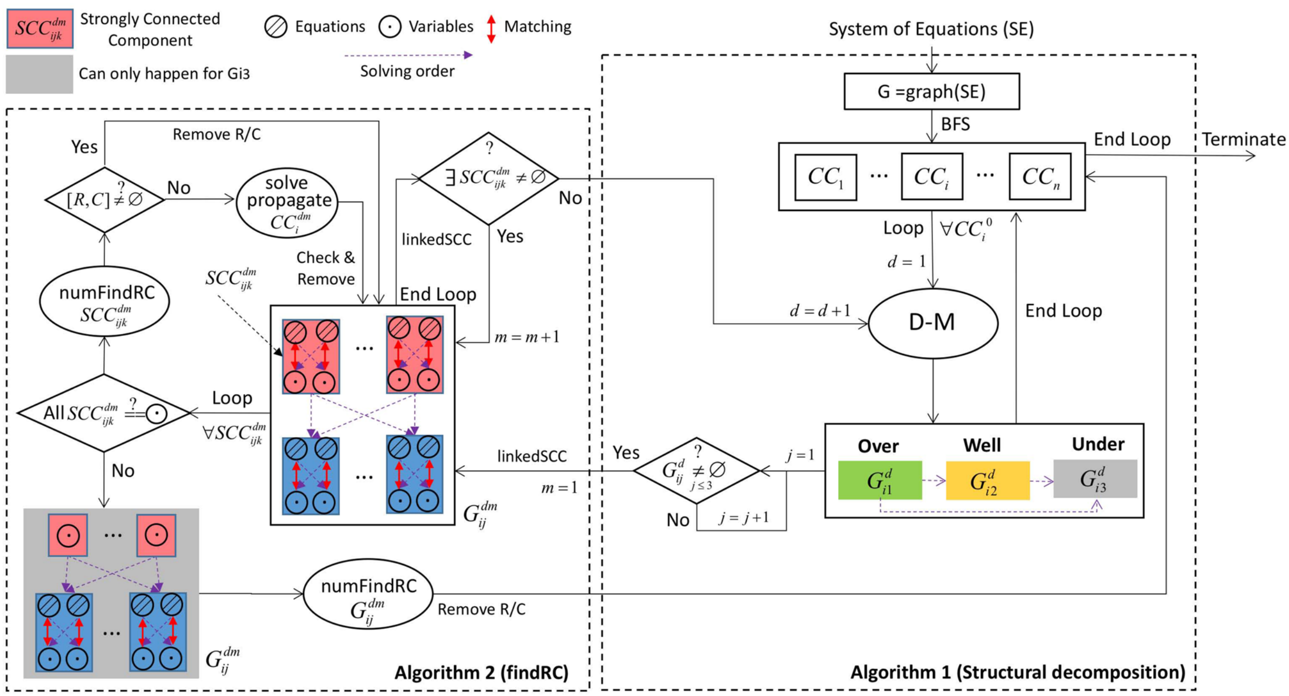



**Fig. 23** Detection framework of Hu's method

numerical methods are used. The symbolic method is pure symbolic operations, where constraints are eliminated one by one by substituting one variable into other equations until a final expression is obtained. Also, non-linear equations are linearized and the G-J method is applied to analyze them. The method repeats the above process until all redundant and conflicting constraints are distinguished (b⊕). Moreover, the method suggests the spanning group of an over-constraint is a set of constraints within the same strong connected component, and it is possible that the solution of a strong connected component results in a singular configuration since no method was proposed to prevent such cases from happenning (c, f⊕). However, it will generate wrong results if a non-linear system is linearized.

The constraint manager of his method enables designers to generate geometric over-constraints iteratively (k⊖ ). When a geometric over-constraint is detected (l⊕), the constraint manager provides three alternatives, where users can select an appropriate one satisfying his/her needs. Finally, as the modeling of the system is in equations (i⊕), the method is applicable to geometries of both free-form and Euler (g⊕⊖), linear and non-linear constraints (h⊕⊖), and 3D and 2D (j⊕⊖).

*Hu's method* Hu's method is also based on the detection-decomposition plan [12]. His method is similar with Serrano's except several differences. They are:

- A system is decomposed twice before applying algebraic methods. As it is shown in the Fig. 23, the system is initially decomposed into $CC_i$s. For each $CC_i$, the D-M decomposition is used. Comparing with the Serrano's method, initial decomposition step is added to decompose a free-form configuration into subparts.
- Non-linear equations are not linearized. Hu's method adopts the Grobner basis, WCM to detect over-constraints among non-linear equations. Since the WCM method is used, solution of a strong connected component would not result in singular configurations of a system (l⊖).

Results of evaluation Hu's method with respect to the adopted criteria are the same with the one of Serrano's except for the differences discussed above.

### 4.5 Results of Evaluation

We summarize the results of evaluating detection methods in the Table 9.

## 5 Conclusion

This paper analyzes the state-of-the-art of approaches for geometric over-constraints detection grouped based on proposed criteria that allow to highlight the main characteristics of methods and to discuss open issues. These criteria reflect the features we believe important to the geometric over-constraints detection. Effective geometric over-constraints

**Table 9** Evaluation of the methods

| Criteria set | | Detection level | | | Decomposition | | | System modeling | | | | Results generation | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Criteria | Def | Type | Redundant conflicting | Spanning group | Over-constrained components | Rigid sub-systems | Singular configura-tion | Geometries | Constraints | Modeling | Dimension | Way of detection | Debugging |
| Methods | | a | b | c | d | e | f | g | h | i | j | k | l |
| Reduction | 1 | ? | ⊙ | ? | ⊕ | ⊖ | ? | ⊖ | ⊕ | ⊖ | ⊖ | ⊙ | ⊙ |
| Dense | 5 | ? | ⊙ | ? | ⊕ | ⊙ | ? | ⊖ | ⊕ | ⊖ | ⊕⊖ | ⊙ | ⊙ |
| Over-rigid | 6 | ? | ⊙ | ? | ⊕ | ⊙ | ? | ⊖ | ⊕ | ⊖ | ⊕⊖ | ⊙ | ⊙ |
| MWM | 7 | ⊖ | ⊙ | ⊙ | ⊕ | ⊖ | ? | ⊖ | ⊕ | ⊖ | ⊕ | ⊕ | ⊕ |
| D-M | 8 | ⊖ | ⊙ | ⊙ | ⊕ | ⊖ | ⊖ | ⊕⊖ | ⊕⊖ | ⊕ | ⊕⊖ | ⊕ | ? |
| G-J | 10 | ⊕ | ⊕ | ? | ⊙ | ⊙ | ⊙ | ⊕⊖ | ⊖ | ⊕ | ⊕⊖ | ⊕ | ? |
| LU | 10 | ⊕ | ⊕ | ? | ⊙ | ⊙ | ⊙ | ⊕⊖ | ⊖ | ⊕ | ⊕⊖ | ⊕ | ? |
| QR | 10 | ⊕ | ⊕ | ? | ⊙ | ⊙ | ⊙ | ⊕⊖ | ⊖ | ⊕ | ⊕⊖ | ⊕ | ? |
| Grobner basis | 10 | ⊕ | ⊕ | ⊖ | ⊙ | ⊙ | ⊖ | ⊕⊖ | ⊕ | ⊕ | ⊕⊖ | ⊖ | ? |
| Wu-Ritt | 10 | ⊕ | ⊕ | ⊖ | ⊖ | ⊖ | ⊖ | ⊕⊖ | ⊕ | ⊕ | ⊕⊖ | ⊖ | ? |
| Perturbation | 10 | ⊕ | ⊖ | ⊖ | ⊙ | ⊙ | ⊙ | ⊕⊖ | ⊕⊖ | ⊕ | ⊕⊖ | ⊕ | ? |
| NPM | 10 | ⊕ | ⊖ | ⊖ | ⊖ | ⊖ | ⊖ | ⊕⊖ | ⊕⊖ | ⊕ | ⊕⊖ | ⊕ | ? |
| WCM | 10 | ⊕ | ⊖ | ⊖ | ⊖ | ⊖ | ⊖ | ⊕⊖ | ⊕⊖ | ⊕ | ⊕⊖ | ⊕ | ? |
| WCM Extention | 10 | ⊕ | ⊖ | ⊖ | ? | ⊖ | ⊖ | ⊕⊖ | ⊕⊖ | ⊕ | ⊕⊖ | ⊖ | ? |
| hybrid-Serrano | 10 | ⊕⊖ | ⊕ | ⊕ | ⊖ | ⊖ | ⊕ | ⊕⊖ | ⊕⊖ | ⊕ | ⊕⊖ | ⊖ | ⊕ |
| hybrid-Hu | 10 | ⊕⊖ | ⊕ | ⊕ | ⊖ | ⊖ | ⊖ | ⊕⊖ | ⊕⊖ | ⊕ | ⊕⊖ | ⊖ | ⊕ |





Journal : Large 11831 | Article No : 9509 | Pages : 25 | MS Code : 9509 | Dispatch : 23-7-2021

detection needs to consider the multi-dimensional information describing a geometric constraints system, which needs to be extracted through a geometric system modeling, decomposition and solving process. Various works in literature are addressing these issues to some extent; however, they take into consideration only some configurations and are applicable in some conditions. Therefore, efforts are still needed to address the challenging applications such as PDP. We foresee that the design of hybrid approaches will enable advances toward practical requirements. In particular, a method that makes as much use as possible of predefined patterns, and resorts to a general DoF-based analysis strengthened by a WCM-based validation in a recursive assembly way (allowing to interleave decomposition, solving, propogation, and recombination phases) would be more applicable towards generality and reliability.

**Acknowledgements** The authors are grateful to the China Scholarship Council (No. 201406090176) for supporting this research. We are also immensely grateful to (Jean-Philippe Pernot and Mathias Kleiner) for their guidances on an earlier version of the manuscript, although any errors are our own and should not tarnish the reputations of these esteemed persons.

## Compliance with ethical standards

**Conflict of interest** The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## References

1. Falcidieno B, Giannini F, Léon JC, Pernot JP (2014) Processing free form objects within a Product Development Process framework, vol 1. ASME-Press, New York, pp 317–344
2. Cheutet V, Daniel M, Hahmann S, La Greca R, Léon JC, Maculet R, Ménegaux D, Sauvage B (2007) Constraint modeling for curves and surfaces in CAGD: a survey. Int J Shape Model 13(02):159
3. Gouaty G, Fang L, Michelucci D, Daniel M, Pernot JP, Raffin R, Lanquetin S, Neveu M (2016) Variational geometric modeling with black box constraints and DAGs. Comput Aided Des 75:1–12
4. Graver J, Servatius B, Servatius H (1993) Combinatorial rigidity. Graduate studies in mathematics. American Mathematical Society, New York
5. Laman G (1970) On graphs and rigidity of plane skeletal structures. J Eng Math 4(4):331–340
6. Podgorelec D, Žalik B, Domiter V (2008) Dealing with redundancy and inconsistency in constructive geometric constraint solving. Adv Eng Softw 39(9):770–786
7. Foufou S, Michelucci D, Jurzak JP (2005) Numerical decomposition of geometric constraints. Proceedings of the 2005 ACM symposium on solid and physical modeling. ACM, pp 143–151
8. Sitharam M, Zhou Y (2004) A tractable, approximate, combinatorial 3d rigidity characterization. Fifth Automated Deduction in Geometry (ADG)
9. Jermann C, Neveu B, Trombettoni G (2003) Algorithms for identifying rigid subsystems in geometric constraint systems. In: 18th International Joint Conference in Artificial Intelligence, IJCAI-03 (No CONF, pp 233-238)
10. Latham RS, Middleditch AE (1996) Connectivity analysis: a tool for processing geometric constraints. Comput Aided Des 28(11):917–928
11. Ait-Aoudia S, Jegou R, Michelucci D (2014) Reduction of constraint systems. arXiv:1405.6131
12. Hu H, Kleiner M, Pernot JP (2017) Over-constraints detection and resolution in geometric equation systems. Comput Aided Des 90:84–94
13. Diestel R (2017) Graph theory. GTM 173, 5th edition, Vol 173. Springer, Heidelberg Graduate Texts in Mathematics
14. Welsh D (2010) Matroid theory. Dover books on mathematics. Dover Publications, New York
15. Michelucci D, Foufou S (2006) Detecting all dependences in systems of geometric constraints using the witness method. In: International Workshop on Automated Deduction in Geometry, pp 98–112. Springer, Berlin
16. Coxeter HSM, Greitzer SL (1967) Geometry revisited, vol 19. Maa, New York
17. Peng X, Chen L, Zhou F, Zhou J (2002) Singularity analysis of geometric constraint systems. J Comput Sci Technol 17(3):314–323
18. Fudos I, Hoffmann CM (1997) A graph-constructive approach to solving systems of geometric constraints. ACM Trans Gr (TOG) 16(2):179–216
19. Brüderlin B, Roller D (eds) (2012) Geometric constraint solving and applications. Springer Science & Business Media
20. Hoffmann CM, Sitharam M, Yuan B (2004) Making constraint solvers more usable: overconstraint problem. Comput Aided Des 36(4):377–399
21. Hoffmann CM, Lomonosov A, Sitharam M (1997, October). Finding solvable subsets of constraint graphs. In: International Conference on Principles and Practice of Constraint Programming. Springer, Berlin, pp 463–477
22. Dulmage AL, Mendelsohn NS (1958) Coverings of bipartite graphs. Can J Math 10:517–534
23. Fritzson P, Bunus P (2002, April) Modelica—a general object-oriented language for continuous and discrete-event system modeling and simulation. In: Proceedings 35th Annual Simulation Symposium. SS 2002, IEEE, pp 365–380
24. Serrano D (1987) Constraint management in conceptual design, Doctoral dissertation, Massachusetts Institute of Technology
25. Strang G (2014) Linear algebra and its applications. Elsevier Science
26. Light RA, Gossard DC (1983) Variational geometry: a new method for modifying part geometry for finite element analysis. Comput Struct 17(5–6):903–909
27. Okunev P, Johnson CR (2005) Necessary and sufficient conditions for existence of the LU factorization of an arbitrary matrix. arxiv:math/0506382
28. Dongarra J, Kennedy K, Messina P, Sorensen DC, Voigt RG (1990) Proceedings of the Fourth SIAM Conference on Parallel Processing for Scientific Computing, Chicago, Illinois, USA. SIAM 1990, ISBN 0-89871-262-9
29. Cox D, Little J, OShea D (2013) Ideals, varieties, and algorithms: an introduction to computational algebraic geometry and commutative algebra. Springer
30. Bose NK (2013) Multidimensional systems theory and applications. Springer
31. Chou SC (1988) An introduction to Wu's method for mechanical theorem proving in geometry. J Automat Reason 4(3):237–267
32. Wu WT (2012) Mechanical theorem proving in geometries: basic principles. Springer
33. Cox DA, Little J, O'Shea D (2015) Ideals, varieties, and algorithms. Springer, New York, pp 49–119
34. Hoffmann CM (1989) Geometric and solid modeling: an introduction. Morgan Kaufmann Publishers Inc., San Francisco

35. Kondo K (1992) Algebraic method for manipulation of dimensional relationships in geometric models. Comput Aided Des 24(3):141–147
36. Chou S (1988) Mechanical geometry theorem proving. Mathematics and its applications. Springer, New York
37. Gao XS, Chou, SC (1998) Solving geometric constraint systems. II. A symbolic approach and decision of Rc-constructibility. Comput Aided Des 30(2):115–122
38. Haug EJ (1989) Computer aided kinematics and dynamics of mechanical systems, vol 1. Allyn and Bacon, Boston, pp 48–104
39. Krantz S, Parks H (2012) The implicit function theorem: history, theory, and applications. Modern Birkhäuser classics. Springer, New York
40. Michelucci D, Foufou S (2006) Geometric constraint solving: the witness configuration method. Comput Aided Des 38(4):284–299
41. Moinet M, Mandil G, Serre P (2014) Defining tools to address over-constrained geometric problems in computer aided design. Comput Aided Des 48:42–52
42. Michelucci D, Schreck P, Thierry SE, Fünfzig C, Génevaux JD (2010). Using the witness method to detect rigid subsystems of geometric constraints in CAD. In: Proceedings of the 14th ACM symposium on solid and physical modeling. ACM, pp 91–100
43. Michelucci D, Foufou S, Lamarque L, Schreck P (2006) Geometric constraints solving: some tracks. In: Proceedings of the 2006 ACM symposium on solid and physical modeling. ACM, pp 185–196
44. Thierry SE, Schreck P, Michelucci D, Fünfzig C, Génevaux JD (2011) Extensions of the witness method to characterize under-, over-and well-constrained geometric constraint systems. Comput Aided Des 43(10):1234–1249
45. Kubicki A, Michelucci D, Foufou S (2014) Witness computation for solving geometric constraint systems. In: Science and information conference (SAI), 2014. IEEE, pp 759–770
46. Zou, Qiang, and Hsi-Yung Feng (2019) On Limitations of the Witness Configuration Method for Geometric Constraint Solving in CAD Modeling. arXiv:1904.00526
47. Zou Q, Feng HY (2020) A decision-support method for information inconsistency resolution in direct modeling of CAD models. Adv Eng Inform 44:101087
48. Pernot JP, Falcidieno B, Giannini F, Léon JC (2008) Incorporating free-form features in aesthetic and engineering product design: State-of-the-art report. Comput Industry 59(6):626–637

**Publisher's Note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.





| Journal : **Large 11831** | Article No : **9509** | Pages : **25** | MS Code : **9509** | Dispatch : **23-7-2021** |
|---|---|---|---|---|