



Science Arts & Métiers (SAM)

is an open access repository that collects the work of Arts et Métiers Institute of Technology researchers and makes it freely available over the web where possible.

This is an author-deposited version published in: <https://sam.ensam.eu>
Handle ID: <http://hdl.handle.net/10985/22255>

To cite this version :

Yanjia HUANG, Qian ZHAO, Sunny YEUNG - Geometric Over-Constraints Detection: A Survey - Archives of Computational Methods in Engineering - Vol. 28, n°7, p.4331-4355 - 2021

Any correspondence concerning this service should be sent to the repository

Administrator : scienceouverte@ensam.eu





2 Geometric Over-Constraints Detection: A Survey

3 Hao Hu^{1,2,3} · Mathias Kleiner³ · Jean-Philippe Pernot³ · Chao Zhang¹ · Yanjia Huang¹ · Qian Zhao¹ · Sunny Yeung²

4 Received: 23 February 2020 / Accepted: 3 October 2020
5 © CIMNE, Barcelona, Spain 2020

6 Abstract

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

18 1 Introduction

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

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

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-

A1 ✉ Hao Hu
A2 huhaoseu@gmail.com

A3 ¹ Guangdong Bright Dream Robotics, Foshan, China

A4 ² Guangdong Country Garden School, Foshan, China

A5 ³ Arts et Métiers Institute of Technology, LISPEN, HESAM
A6 Université, Aix-en-Provence 13617, France

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

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

69 However, information provided by users may be inconsis-
70 tent and the overall set can be over-constrained when
71 manipulating CAD models directly [47]. In most of today's
72 modeler, a geometric configuration can be of three types:

- 73 • Under-constrained: number of unknowns is greater than
74 the number of equations. Such case happens quite often
75 since designers often insert extra DoFs to satisfy require-
76 ments.
- 77 • Well-constrained: number of unknowns is equal to the
78 number of equations.
- 79 • Over-constrained: number of unknowns is less than the
80 number of equations. The type of extra equations have
81 two possibilities:
 - 82 • Redundant: these equations are consistent with the
83 other ones. That is, they do not affect the solution of
84 the original system.
 - 85 • Conflicting: fully inconsistent with the others when
86 constraints express contradictory requirements and
87 lead to no solution.

88 A geometric system may be solvable if it is under-con-
89 strained or well-constrained. But when it is over-constrained,
90 the system is hard to solve or even non-solvable. To make a
91 system consistent with designer's requirements, it is neces-
92 sary to detect geometric over-constraints and present them
93 to designers for debugging purpose. In this paper, we collect
94 and classify a state-of-the-art methods for detecting geomet-
95 ric over-constraints, including: (1) definitions of geometric
96 over-constraints; (2) clear identification of criteria used to
97 characterize methods; (3) study the methods and compare
98 them according to the criteria; (4) proposed frameworks for
99 detecting geometric over-constraints.

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

2 Representations and Definitions 106

2.1 Representation of Geometric Constraints Systems 107

108
109 *Equations* CAD modelers provide their solvers of geo-
110 metric constraints and usually the solver has its own con-
111 straints editor. Basically, the constraints concern verti-
112 ces, straight lines, planes, circles, spheres, cylinders or
113 freeform curves and surfaces whose parameters are the
114 unknown variables. Constraints ranging from level 1 to
115 level 3 (Sect. 1) can be represented with equations. Those
116 equations can be linear or non-linear. Classical solvers
117 use these constraints to sketch and constrain the shape of
118 desired models. For example, the 2D distance constraint
119 d between two points (x, y) and (x_0, y_0) is translated to
120 the equation $(x - x_0)^2 + (y - y_0)^2 - d^2 = 0$. Continuity
121 constraints between two patches can also be represented
122 with equations. Moreover, those mathematical equations
123 can also be represented using computational graph, which
124 is based on Directed Acyclic Graphs (DAGs). In such a
125 representation, a DAG is a tree with shared vertices. The
126 leaves of the tree are either variables (i.e. parameters or
127 unknowns) or numerical coefficients. The internal nodes
128 of the tree are either elementary arithmetic operations
129 or functions such as exp ; sin ; cos ; tan . The DAG is also
130 called white box DAG, since it allows for computing the
131 derivatives and Hessians automatically. If mathematical
132 equations associated to geometric constraints are avail-
133 able, it is possible to compute the expressions of the
134 derivatives with formal calculus, which can be resorted
135 to using the Grobner basis or Wu-Ritt method if all the
136 constraints are algebraic and can be triangulated into the
137 form $f_1(U; x_1) = f_2(U; x_1; x_2) = \dots = 0$ (U is the param-
138 eters vector and x_i are the unknown variables).

139 *Black boxes* On the contrary, a DAG is called a black
140 box DAG, and a constraint is called a black box constraint
141 when it cannot be represented with equations or are not
142 computable in practice [3]. This corresponds to con-
143 straints of level 4 discussed in Sect. 1. Examples such as,
144 maximum of the Von Mises stress should be smaller than
145 100MPa, the final product should cost less than 100, are
146 requirements which cannot be transformed into a set of
147 equations. In the work of [3], they proposed to use black
148 box DAGs for variational geometric modeling of free-form
149 surfaces and subdivision surfaces. A prototype, DECO,
150 is presented to show the feasibility and promises of the
151 approach. Black box constraints happen when free-form
152 surfaces are generated tediously from modeling functions
153 (e.g. sweep, loft, blend). They cannot be manipulated in
154 the same way as if some equations were available and solv-
155 ers have to take into account these constraints expressed by

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

160 **2.2 STAR Definitions of Geometric Over-Constraints**

161 **2.2.1 Definitions at the Level of Geometries**

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

171 *Definitions based on constraint graph* Here, we use
 172 $G = (V, E)$ to represent a constraint system with $|V|$ num-
 173 ber of entities and $|E|$ number of constraints.

174 In Rigidity Theory [4], Laman’s theorem [5] character-
 175 izes the rigidity of bar frameworks, where a geometric
 176 system is composed of points constrained by distances.

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

181 The constraints and entities are limited to distances and
 182 points respectively. Podgorelec [6] extended the theorem
 183 by assuming that each geometric element has 2 DoFs and
 184 each constraint eliminates 1 DoF. Therefore, the weight
 185 of vertices and edges are of the constraint graph is 2 and
 186 1 respectively.

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

- 189 • *Structurally over-constrained* if there is a subgraph
 190 $G' = (V', E')$ with $1 \cdot |E'| > 2 \cdot |V'| - 3$,
- 191 • *Structurally under-constrained* if G is not *structurally*
 192 *over-constrained* and $1 \cdot |E| < 2 \cdot |V| - 3$, or
- 193 • *Structurally well-constrained* if G is not *structurally*
 194 *over-constrained* and $1 \cdot |E| = 2 \cdot |V| - 3$.

195 **Definition 2** A constraint e is a *structural over-constraint*
 196 if a structurally over-constrained subsystem $G' = (V', E')$ of
 197 G with $e \in E'$, can be derived such that $G'' = (V', E' - e)$
 198 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 dif-
 ferent constraints in 2D space. As it is shown in the Fig. 1a,
 it is over-constrained because it contains 3 distance con-
 straints and 3 vertical position constraints. Since each con-
 sumes 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 $3 < 2 \times 3 - 3$.

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 con-
 straints (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 con-
 straints, $e \leq 3n - 6$. This condition is necessary but not
 sufficient. A counter example is the double banana geom-
 etry 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].

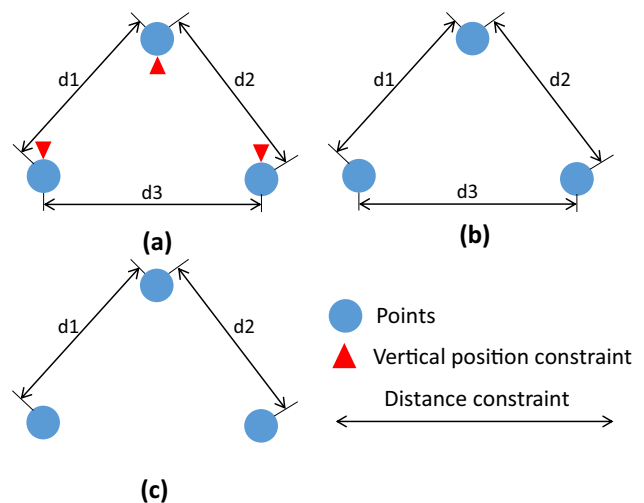


Fig. 1 a Structurally over-constrained, b structurally well-constrained, c structurally under-constrained

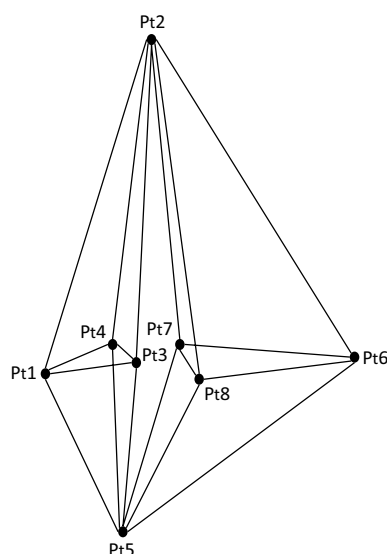


Fig. 2 Double-Banana geometry where all vertexes are variables and all edges are distance constraints

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

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

242 **Definition 4** Degree of freedom of a geometric con-
 243 straint ($DoC(e)$, e is the constraint) is the number of
 244 independent equations needed to represent it. For a sys-
 245 tem $G = (V, E)$, the DoFs all constraints can remove is
 246 $DoC(E) = \sum_{e \in E} DoC(e)$.

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

- 249 • *Structurally over-constrained* if there is a subgraph
 250 $G' = (V', E')$ satisfying $DoC(E') > DoF(V') - D$,
- 251 • *Structurally well-constrained* if $DoC(E) = DoF(V) - D$
 252 and all subgraphs $G' = (V', E')$ satisfying
 253 $DoC(E') \leq DoF(V') - D$,
- 254 • *Structurally under-constrained* if
 255 $DoC(E) < DoF(V) - D$ and contains no *structurally*
 256 *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, Jer-
 man 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 *structur-*
ally 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 paral-
 lel 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/collinear-
 ity 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 unsatu-
 rated 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

304 maximal weighted matching. An unsaturated constraint is a
 305 *structural over-constraint*. The weights of edges are com-
 306 puted by maximal weighted matching of a bipartite graph.

307 **2.2.2 Definitions at the Level of Equations**

308 In this section, we summarize the definitions used when
 309 a qualitative study of geometric systems is performed at
 310 the level of equations. Modeling at the level of geometries
 311 preserves geometric information of a system. Modeling at
 312 the level of equations, however, discards geometric proper-
 313 ties of a system but enables a fine detection of geometric
 314 over-constraints.

315 *Structural definitions* System of equations are trans-
 316 formed into bipartite graph, where vertices represent equa-
 317 tions and variables respectively. The characterization is
 318 based on the results of maximum matching [11]. Here, we
 319 assume that $G = (U, V, E)$ is a bipartite graph with U and
 320 $V (U \cap V = \emptyset)$ representing variables and equations respec-
 321 tively, and E representing edges.

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

- 324 • *Structurally over-constrained* if the number of elements
 325 in U' is smaller (in cardinality) than the number of V' . i.e.
 326 $|U'| < |V'|$
- 327 • *Structurally well-constrained* iff G' has perfect matching.
- 328 • *Structurally under-constrained* if the number of elements
 329 in U' is larger (in cardinality) than the number of ele-
 330 ments in V' . i.e. $|U'| > |V'|$

331 **Definition 9** Let M be a maximum matching of
 332 $G = (U, V, E)$. If M is not perfect matching and V' is the
 333 subset of V which is not saturated by M , then equations of
 334 V' are the *Structural over-constraints*.

335 *Numerical definitions* Informally, an over-constrained
 336 constraints system has no solutions, a well-constrained con-
 337 straints system has a finite number of solutions, and a under-
 338 constrained constraints system has infinite solutions. In the
 339 work of Hu et al. [12] as well as the recent work of Zou et al.
 340 [46], they gave the following definitions.

341 **Definition 10** Let $G = (E, V, P)$ be a geometric constraints
 342 system, where E is a set of equations, V is a set of variables
 343 and P is a set of parameters. The set of solutions to G is
 344 denoted $Sol(G)$. A geometric constraints system is *inconsis-*
 345 *tent* iff $Sol(G) = \emptyset$ and is *consistent* iff $Sol(G) \neq \emptyset$.

346 **Definition 11** Let $G = (E, V, P)$ be a *consistent* geometric
 347 constraints system. Let $G' = (E \cup E_c, V, P')$ be an *inconsis-*
 348 *tent* geometric constraints system, where E_c is a set of

equations forming a constraint $C = \{E_c \mid E_c \cap E = \emptyset\}$ 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 con-
 straints system. Let $G' = (E \cup E_r, V, P')$ be a consistent geo-
 metric constraints system, where E_r is a set of equations
 forming a constraint $R = \{E_r \mid E_r \cap E = \emptyset\}$ 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 follow-
 ing two subsystems: $G_b = (E_b, V, P)$ and $G_o = (E_o, V, P)$ with
 $\{E = E_b \cup E_o, E_b \cap E_o = \emptyset\}$. If any constraint E_{oi} in E_o is
 either redundant or conflicting with respect to G_b , and if
 $\text{card}(E_b) \geq \text{card}(E_o)$, then E_b is a set of *basis constraints*
 and E_o is a set of *numerical over-constraints*.

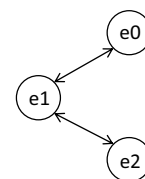
However, we have to mention that the Definition 12 is not
 consistent with matroid theory. For example, in the Fig. 3, $e1$
 is conflicting both with $e0$ and $e2$. According to our definitions
 of redundant, conflicting, and basis constraints (Definitions
 11 and 12), the result would be: $e1$ is conflicting 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 $\text{rank}(A)$ and $\text{rank}(B)$ is 1 because A and B are depend-
 ent respectively. We can also deduce that $\text{Rank}(A \cup B) = 2$ and
 $\text{Rank}(A \cap B) = 1$. As a result, we should have $2 + 1 \leq 1 + 1$,
 which is wrong. We redefine the Definition 11 and the Defini-
 tion 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 estab-
 lished 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 estab-
 lished 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 con-
 flicting with $e0$ and $e2$



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

406 2.3 Evaluation

407 A set of criteria are defined to evaluate these defini-
 408 tions (Table 1). These criteria are: D is a dimension
 409 (system)-dependent constant; geometries refer to the geo-
 410 metric type a definition used to specify; counter example
 411 lists geometries that a definition cannot deal with.

412 From the table below, we can see that definitions can be
 413 divided into two groups: Definition 1,5,6 and Definition
 414 7,8,10. Because the former group manipulate geometric
 415 elements directly at the level of geometries and geometric
 416 constraints are usually supposed to be independent of all
 417 coordinates system, they can not be used to determine the

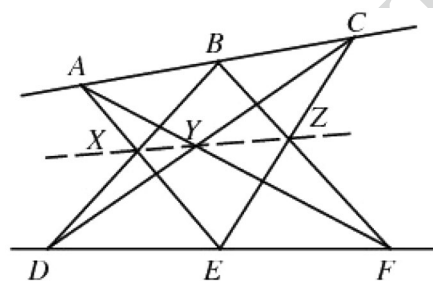


Fig. 4 Pappus's hexagon theorem: Points X, Y and Z are collinear on the Pappus line (dotted line). The hexagon is $AFBDCE$

418 location and orientation of a geometric configuration (no
 419 fixation) as well as carefully defined the value of D . Also, the
 420 defined type of geometries and constraints are limited. For
 421 example, the Definition 1 and Definition 5 are defined for
 422 points geometries and distances constraints only. But col-
 423 linear (and cocyclic, coconic, cocubic, etc.) points are for-
 424 bidden. The Definition 6 extends the type of geometries to
 425 points, lines and planes as well as it allows for incidence
 426 constraints to be defined. The Definition 7 extracts geomet-
 427 ric entities and constraints to DoFs and DoCs, and define
 428 over-constraints by simply comparing the number of DoFs of
 429 geometric entities and DoCs of geometric constraints. In this
 430 way, the definition is not limited to any specific class of geo-
 431 metric entities and constraints. The Definition 8, and the Def-
 432 inition 10, however, are numerical definitions dealing with
 433 geometries and constraints at the level of equations. These
 434 definitions can cover any geometric entities and constraints
 435 once they are represented with equations. Counter examples
 436 are the black box constraints which cannot be represented
 437 with equations. Moreover, these definitions require systems
 438 to be fixed with respect to a global coordinate system and
 439 thus $D = 0$. Finally, since geometric redundancy does not
 440 use parameters of any geometries of a constraints system, it
 441 cannot be covered by any of these definitions.

442 To cover cases like geometric redundancy as well as be
 443 consistent with the matroid theory, we redefine basis equa-
 444 tions, redundant and conflicting equation as follows.

Definition 13 Let $G = (E, V, P)$ be a geometric constraints
 445 system, where E is a set of equations, V is a set of variables
 446 and P is a set of parameters. Let E_r be a non-empty collec-
 447 tion of subsets of E , called basis equations (we call it basis
 448 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
 454 system. Let E_r be a basis. For an equation e , adding it to E_r
 455

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

456 forming a new group: $\{E_r \cup e\}$. If $\{E_r \cup e\}$ is solvable, then
 457 e is a redundant equation.

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

462 **Definition 16** Let $G = (E, V, P)$ be a geometric
 463 constraints system which is composed of two sub-
 464 systems: $G_b = (E_b, V, P)$ and $G_o = (E_o, V, P)$ with
 465 $\{E = E_b \cup E_o, E_b \cap E_o = \emptyset\}$. If E_b is a basis, then E_o is a set
 466 of numerical over-constraints.

467 *Spanning group* For an over-constraint $E_{oi} \in E_o$,
 468 the *Spanning Group* E_{sg} of E_{oi} is a group of inde-
 469 pendent constraints, with which E_{oi} is redundant or
 470 conflicting. For linear systems, the spanning group
 471 $E_{sg} = \{e_{sg1}, e_{sg2}, \dots, e_{sgn}\} \subset E_b$ of E_{oi} satisfies:

472
$$E_{oi} = \sum_{j=1}^n c_j e_{sgj} + b \tag{1}$$

473 where $c_j \neq 0$ and is the corresponding scalar coef-
 474 ficient, $\{e_{sg1}, e_{sg2}, \dots, e_{sgn}\}$ are linear independent
 475 and b is the bias. Thus, E_{oi} is a linear combination of
 476 $\{e_{sg1}, e_{sg2}, \dots, e_{sgn}, b\}$. Moreover, E_{oi} is redundant if $b = 0$
 477 otherwise it is conflicting.

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

482
$$\begin{aligned} e1 : x_1 + x_2 + x_3 + x_4 &= 1 \\ e2 : x_1 + 2x_2 + 3x_3 + x_4 &= 4 \\ e3 : x_1 - 2x_2 + x_3 + x_4 &= 5 \\ e4 : 6x_1 + x_3 + 2x_4 &= 7 \\ e5 : 8x_1 + 5x_3 + 4x_4 &= 17 \\ e6 : 11x_1 + x_2 + 10x_3 + 7x_4 &= 27 \end{aligned} \tag{2}$$

483 Clearly, the system is over-constrained since there are
 484 more equations than variables. Through linear analysis
 485 of the system, we find that $e5$ is a linear combination of
 486 $\{e2, e3, e4, 1\}$ and is spanned by $\{e2, e3, e4\}$; $e6$ is a lin-
 487 ear combination of $\{e1, e2, e3, e4, 1\}$ and is spanned by
 488 $\{e1, e2, e3, e4\}$ (Fig. 5). Since the bias of the two groups is
 489 1, both $e5$ and $e6$ are conflicting. In this case, $\{e1, e2, e3, e4\}$
 490 can be treated as a set of basis constraints since all the equa-
 491 tions are independent and the number of them equals to the
 492 number of variables.

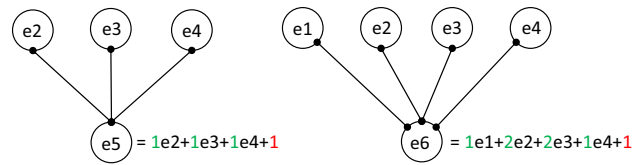


Fig. 5 Spanning group of $e5$ and $e6$: numbers marked green are coef-
 ficients while the ones marked red are the biases

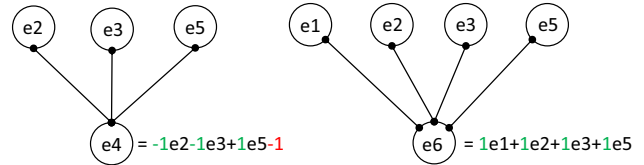


Fig. 6 Spanning group of $e4$ and $e6$: numbers marked green are coef-
 ficients while the one marked red is the bias

494
$$\begin{aligned} e1 : x_1 + x_2 + x_3 + x_4 &= 1 \\ e2 : x_1 + 2x_2 + 3x_3 + x_4 &= 4 \\ e3 : x_1 - 2x_2 + x_3 + x_4 &= 5 \end{aligned} \tag{3}$$

495
$$e4 : 6x_1 + x_3 + 2x_4 = 7$$

496 However, if we replace $e4$ with $e5$, the new set
 497 $\{e1, e2, e3, e5\}$ is also the basis constraints set. Linear
 498 analysis result shows that $e4$ is a linear combination of
 499 $\{e2, e3, e5, -1\}$ and is spanned by $\{e2, e3, e5\}$; $e6$ is a
 500 linear combination of $\{e1, e2, e3, e5\}$ and is spanned by
 501 $\{e1, e2, e3, e5\}$ (Fig. 6). Also, $e4$ is conflicting and $e5$ is
 502 redundant according to the corresponding bias values.

504
$$\begin{aligned} e1 : x_1 + x_2 + x_3 + x_4 &= 1 \\ e2 : x_1 + 2x_2 + 3x_3 + x_4 &= 4 \\ e3 : x_1 - 2x_2 + x_3 + x_4 &= 5 \\ e5 : 8x_1 + 5x_3 + 4x_4 &= 17 \end{aligned} \tag{4}$$

505 From the Fig. 5 and the Fig. 6, we can see that the span-
 506 ning group of $e6$ is not unique, which depends on the set
 507 of basis constraints. Also, the type of an over-constraint
 508 can change: $e6$ is conflicting with respect to the basis con-
 509 straints set $\{e1, e2, e3, e4\}$ while redundant with respect
 510 to the basis constraints set $\{e1, e2, e3, e5\}$.

511 To the best of our knowledge, there is no formal defini-
 512 tions that can cover cases like black box constraints, let
 513 alone the corresponding detection methods. Therefore, in
 514 this paper, both of the definitions and the detection meth-
 515 ods address only white box constraints.

518 3 Evaluation Criteria

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

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

549 3.1 Criteria Attached to the Level of Detecting 550 Over-Constraints

551 The first criterion is relative to the type of geometric
 552 over-constraints (Fig. 7a), which are either numerical
 553 ($a\oplus$) or structural ($a\ominus$). Second criterion concentrates
 554 on distinguishing redundant and conflicting constraints
 555 detected by numerical methods (Fig. 7b). Finally, in
 556 engineering design, designers could better debug and

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

557 modify a geometric over-constraint if its spanning group
 558 is informed (Fig. 7c). This criterion evaluates numerical
 559 methods only (Table 3).

560 3.2 Criteria Related to the System Decomposition

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

576 Decomposition methods should take into account the
 577 singularities. Indeed, many methods work under a generic-
 578 ity hypothesis and decompose systems into generically
 579 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

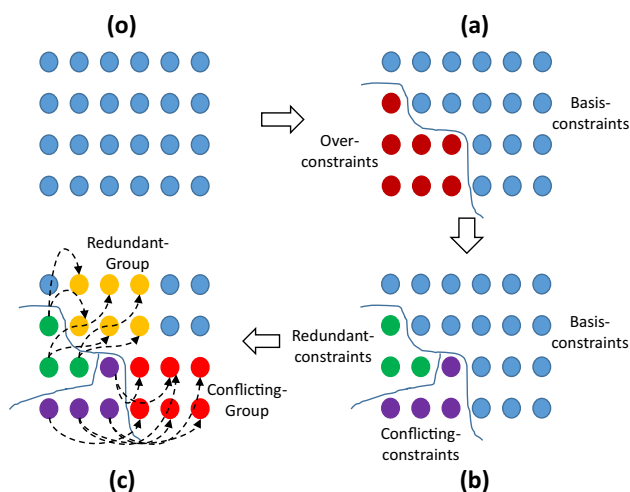


Fig. 7 (o): Level o (a): Level a (b): Level b (c): Level c

Table 4 Criteria related to system decomposition (set 2)

Decomposition		Gradation of criteria	
Level	Criteria	\oplus	\ominus
d	Over-constrained components	Yes	No
e	Rigid subsystems	Numerical	Structural
f	Singular configuration	Yes	No

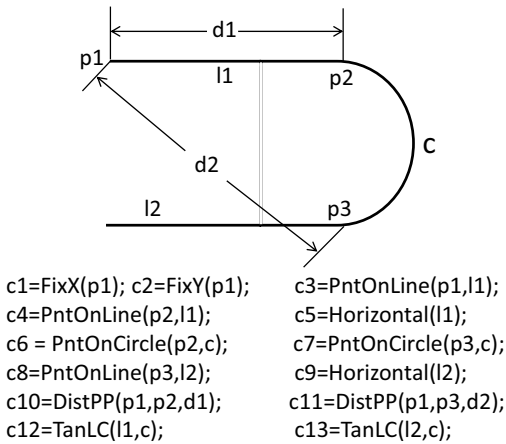


Fig. 8 Singular configuration as described in [17]

580 rigid (non-rigid) before and after an infinitesimal perturba-
 581 tion [4]. A singular configuration, however, transforms from
 582 rigid (non-rigid) to non-rigid (rigid) after an infinitesimal
 583 perturbation. It happens when geometric elements are drawn
 584 with unspecified properties (collinearity, coplanarity, etc.). It
 585 may be the case that a solution of a decomposed system
 586 lies into a singular variety, e.g., includes some unspecified
 587 collinearity or coplanarity. In this case, it happens that the
 588 generically solvable components are no more solvable. For
 589 instance, the doublebanana geometry (Fig. 2) is generically
 590 over-constrained but becomes under-constrained if the
 591 height of both bananas is the same since the two “bananas”
 592 can fold continuously along the line passing through their
 593 extremities. Moreover, the Jacobian matrix at singular con-
 594 figurations is rank deficiency, which introduces dependences
 595 between constraints. For example, the Jacobian matrix of the
 596 subsystem $\{p3, l2, c, c7, c8, c13\}$ of the Fig. 8 is of size 7×7
 597 . But its rank is 5, which is a singular configuration. Obvi-
 598 ously, there is no redundant constraints and the singularity
 599 comes from the tangent constraints between c and $l1, l2$ [17].

3.3 Criteria Related to the System Modeling

601 This set of criteria characterize detection approaches with
 602 respect to system modeling: the type of geometries (g)
 603 and constraints (h), modeling at the level of equations or

Table 5 Criteria related to system modeling (set 3)

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

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

Results generation		Gradation of criteria	
Level	Criteria	\oplus	\ominus
k	Way of detection	Single-pass	Iteratively
l	Debugging	Yes	No

604 geometries (i), 3D or 2D space (j). The first criterion char-
 605 acterizes the type of geometries. Currently, geometric enti-
 606 ties are either Euler geometries ($g\ominus$) such as line segments,
 607 cylinders, spheres etc. or free-form geometries ($g\oplus$). The
 608 second criterion deals with linear ($h\ominus$) and non-linear ($h\oplus$)
 609 constraints. The third criterion describes a system either at
 610 the level of equations ($i\oplus$) or geometries ($i\ominus$). Finally, a
 611 modeling system can either be in 2D ($j\ominus$) or 3D ($j\oplus$) space
 612 (Table 5).

3.4 Criteria Related to the Results Generation

614 In reality, a designer may require that a modeler outputs
 615 geometric over-constraints iteratively when modeling a geo-
 616 metric system interactively. Iteratively means the method
 617 enables to generate results through steps/loops ($k\ominus$) while
 618 single-pass methods generate the results all at once ($k\oplus$)
 619). Also, a user-friendly method should enable the treatment
 620 of results for debugging purpose ($l\oplus$). That is, locate the
 621 results at the level of geometries so that users can modify/
 622 remove them (Table 6).

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

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

630 4.1 Methods Working at the Level of Geometries

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

635 *Reduction* Fudos and Hoffman [18] introduced a con-
 636 structive approach to solve a constraint graph, where geo-
 637 metric entities are lines and points, geometric constraints
 638 are distances and angles. In their reduction algorithm,
 639 triangles are found and merged recursively until the ini-
 640 tial graph is rewritten into a final graph. The structur-
 641 ally over-constrained system/subsystem are detected in two
 642 ways. Firstly, before finding triangles, the approach checks
 643 if the subgraph is structurally over-constrained. Secondly,
 644 if a 4-cycle graph is met during the reduction process,
 645 then the system is structurally over-constrained. A 4-cycle
 646 graph corresponds to two clusters sharing two geometric
 647 elements, which is structurally over-constrained.

648 Results of evaluating the method are as following:

- 649 • *Criteria set 1* Although the method allows for checking
 650 the constrained status of a system, it does not specify
 651 how to find the structural over-constraints as well as
 652 finding the spanning groups (a,c?). Since the method is
 653 structural, it is meaningless to distinguish redundant and
 654 conflicting constraints (b⊙).
- 655 • *Criteria set 2* The method enables to identify a 4-cycle
 656 graph which is structurally over-constrained (d⊕). Also,
 657 the triangles found during the recursive process are the
 658 rigid subsystems (e⊖). In terms of dealing with singu-
 659 lar configurations, it is not mentioned in the original
 660 paper (f?).
- 661 • *Criteria set 3* Normally, a constraints system is com-
 662 posed of Euler geometries (g⊖) with non-linear con-

663 straints (distances, angles $h\oplus$) and modeled at the
 664 level of geometries ($i\ominus$) in 2D space ($j\ominus$).

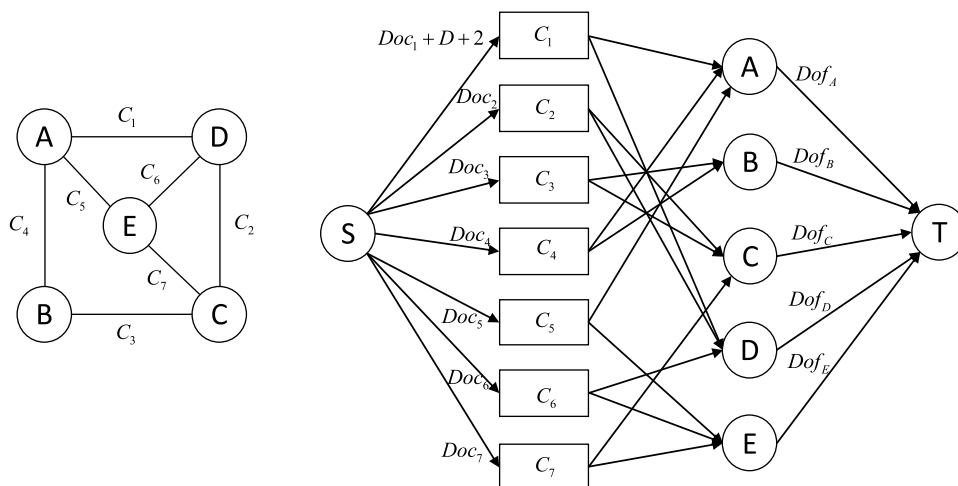
- 665 • *Criteria set 4* Since detecting geometric over-constraints
 666 are not addressed, there is no meaning discussing how
 667 the over-constraints are generated (k⊙) as well as debug-
 668 ging them (l⊙).

669 *Dense* Hoffman et al adapted their *Dense* algo-
 670 rithm [19] to locate 1-overconstrained subgraph (sat-
 671 isfying $DOCs > DOFs - D + 1$) of 1-overconstrained
 672 graph [20]. The algorithm is composed of four main steps.

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

680 As it is shown in the Fig. 9, generally locating a mini-
 681 mal subgraph of density $-D + 1$ is done as follows: first,
 682 by distributing an flow of weight $Doc_i + D + 2$ from each
 683 constraint to its end points(entities) to find a subgraph of
 684 density $-D + 1$. Such dense graph is found when there exists
 685 an edge whose edge cannot be distributed with redistribu-
 686 tion [21]. The algorithm continues to locate minimal 1-over-
 687 constrained subgraph. But in our opinion, to check whether
 688 a system is over-constrained or not, it is sufficient that the
 689 algorithm terminates at step 3. The authors suggested to fur-
 690 ther extend the algorithm to incrementally detect k-Over-
 691 constrained graphs. The algorithm allows for updating
 692 constraints efficiently. Once the constraints are identified,
 693 they are removed. However, the algorithm excludes large
 694 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)



695 Results of evaluating the method are:

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

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

- 717 1. The overload is applied on a virtual node *R* (Fig. 10
718 and Fig. 11) whereas in the *Dense* algorithm, it is
719 applied on a constraint node.
- 720 2. The overload is $Dor + 1$ and the computation of Dor
721 depends on the subsets of constraint entities to which *R*
722 is attached. For example, $Dor(A, B)$ ($\{A, B\}$ is the sub-
723 set of the configuration in the Fig. 10) is different from
724 $Dor(C, D, E)$ ($\{C, D, E\}$ is the subset of the configuration

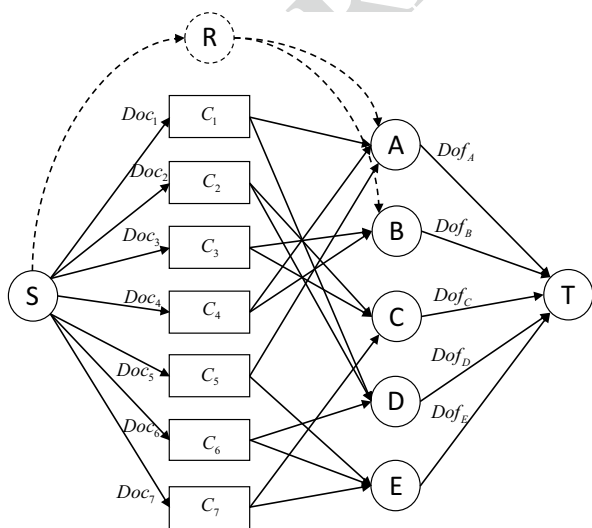


Fig. 10 Overloading to subset {A,B}

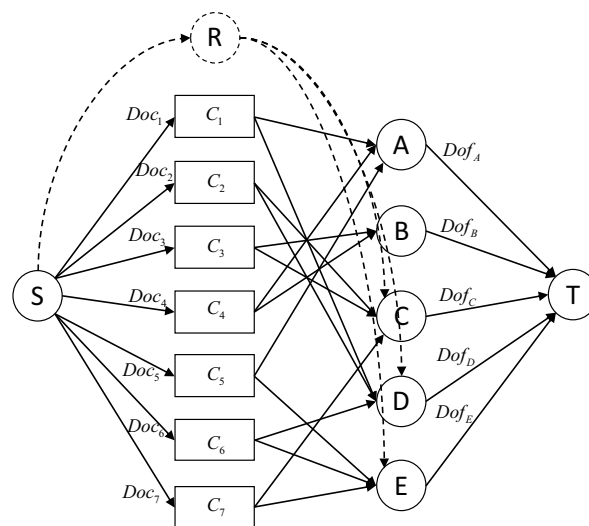


Fig. 11 Overloading to subset {C,D,E}

in the Fig. 11). But in the *Dense* algorithm, the overload
is invariant with different subsystems.

- 725
- 726
- 727 3. The *R* node is attached to *Dor-minimal* subsets of objects
728 in order to find over-rigid subsystems.

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 specifically 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 sub-
graphs 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 con-
straints and enables to identify under- and over-constrained
configurations.

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

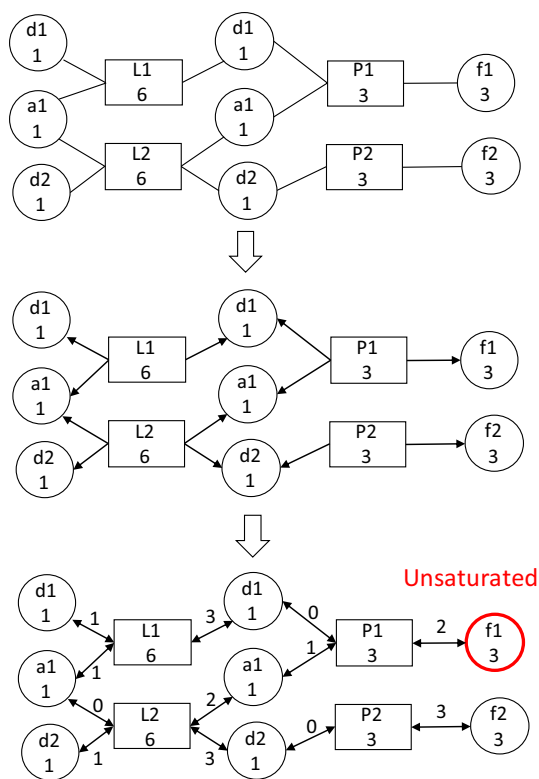


Fig. 12 Over-constraints detection process of an example taken from [10]. The unsaturated node is the geometric over-constraint

757 constraints and entities is applied and those unsaturated
 758 constraints nodes are geometric over-constraints. Moreover,
 759 over, they addressed the over-constrained problems by pri-
 760 oritizing the given constraints, where over-constraints can
 761 automatically be modified based on constraints priorities.
 762 Results of evaluating the method are:

- 763 • *Criteria set 1* The unsaturated constraints are geomet-
 764 ric over- constraints ($a\ominus$). Since the detected over-con-
 765 straints are structural, there is no meaning to discuss
 766 redundant and conflicting constraints as well as the span-
 767 ning groups ($b,c\odot$).
- 768 • *Criteria set 2* The subgraph containing an unsaturated
 769 constraint node is the over-constrained component ($d\oplus$

770). It can be found by tracing the descendant nodes of
 771 the unsaturated node. Moreover, the algorithm enables a
 772 decomposition of the system into balanced subsets which
 773 are rigid subsystems ($e\ominus$). Also, analyzing the singular
 774 configurations is not discussed ($f?$).

- 775 • *Criteria set 3* The results of evaluation with respect to
 776 this set of criteria are the same with those of the *Over-*
 777 *rigid* algorithm except the whole system is modeled in
 778 3D space ($j\oplus$).
- 779 • *Criteria set 4* The over-constraints are detected in the sin-
 780 gle-pass way ($k\oplus$). And they proposed to modify the
 781 constraints according to constraints priorities ($l\oplus$).

4.2 Methods Working at the Level of Equations

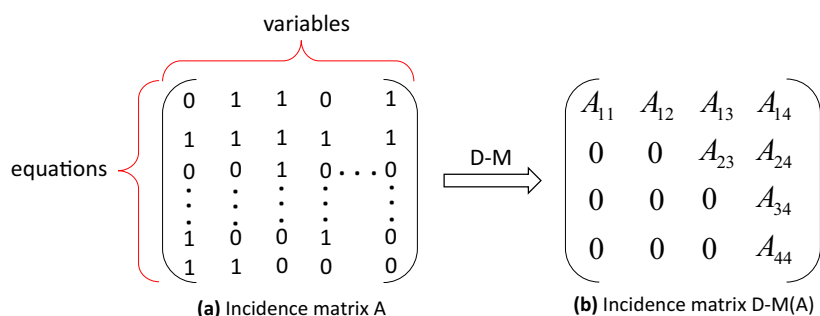
4.2.1 Linear Methods

784 In general, almost all the geometric constraints can be trans-
 785 lated mechanically into a set of algebraic equations [19]. There-
 786 fore, detecting geometric over-constraints is equivalent with
 787 identifying a set of conflicting/redundant equations. However,
 788 even if detection works at the level of equations, the treatment
 789 needs to be done at the level of geometries.

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

803 The process of the D-M decomposition: $D-M(A) =$
 804 $A(p, q)$ does not require A need to be square or full struc-
 805 tural rank (Fig. 13). $A(p, q)$ is split into a 4-by-4 set of
 806 coarse blocks: where A_{12}, A_{23} , and A_{34} are square with
 807 zero-free diagonals. The columns of A_{11} are the unmatched
 808 columns, and the rows of A_{44} are the unmatched rows. Any

Fig. 13 D-M decomposition of incidence matrix A



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

811 *Coarse decomposition*

- 812 • [A11 A12] is the under-constrained part of a system and it
 813 is always rectangular and with more columns than rows,
 814 or does not exist.
- 815 • A23 is the well-constrained part of a system and it is
 816 always square.
- 817 • [A34; A44] is the over-constrained part of a system and
 818 it is always rectangular with more rows than columns, or
 819 does not exist.

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

- 827 • *Criteria set 1* Equations of [A34; A44] are structural
 828 over-constraints ($a\ominus$). Evaluation of criteria b and c is
 829 meaningless since the method is structural ($b,c\ominus$).
- 830 • *Criteria set 2* [A34; A44] after coarse decomposition is
 831 the structural over-constrained subpart ($d\oplus$). The strong
 832 connected components after fine decomposition are
 833 structural rigid subsystems ($e\ominus$). The method does not
 834 discuss on the analysis of singular configurations ($f\ominus$).
- 835 • *Criteria set 3* Since the modeling is based on system of
 836 equations, any geometric constraints that are able to be
 837 transformed into system of equations can be analyzed
 838 by the method. Therefore, the results for evaluating the
 839 method according to this set of criteria are ($g\oplus\ominus, h\oplus\ominus,$
 840 $i\oplus, j\oplus\ominus$).

- 841 • *Criteria set 4* Structural over-constraints are contained
 842 in the over-constrained part and output in a single-pass
 843 way ($k\oplus$). The method does not discuss on debugging the
 844 over-constraints ($l?$).

845 In this section, we gather together the approaches from lin-
 846 ear algebra that are capable of analyzing linear system of
 847 constraints. We consider linear system of constraints in the
 848 matrix form $Ax = b$, where A has dimension $m \times n$, and
 849 $n \geq m \geq r$ with r being the rank. The notation $A[i : j, l : k]$
 850 defines the matrix obtained by slicing the i th to j th rows,
 851 and the l th to k th columns of A. According to [25], methods
 852 such as Gauss-Jordan Elimination, LU and QR Factoriza-
 853 tion present a good characteristic of locating inconsistent/
 854 redundant equations.

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

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

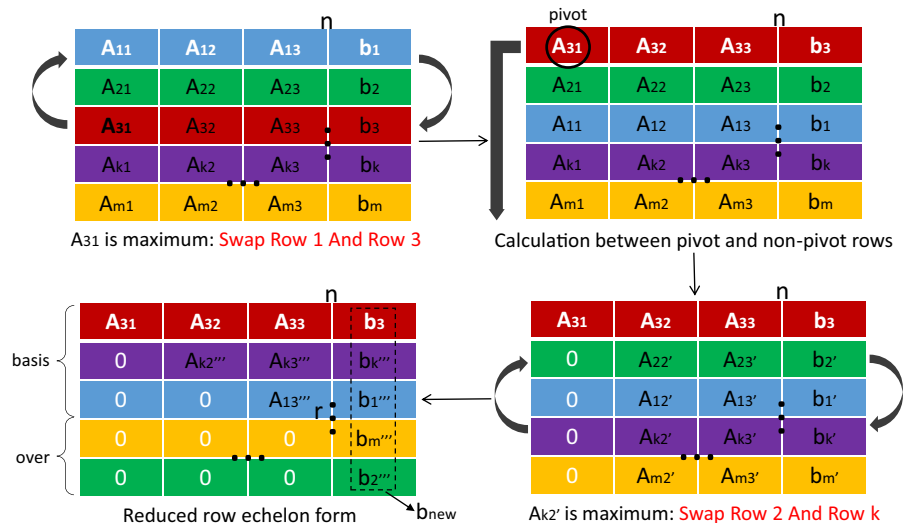
860 where $A_{ik}^{(k)} = A[i, k]$, an element of the i th row and the k th
 861 column in A.

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

867 Results of evaluating the method are as follows:

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

Fig. 14 Gauss elimination with partial pivoting



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

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

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

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

$$897 \left. \begin{matrix} Ax = b \\ PA = LU \end{matrix} \right\} Ux = L^{-1}Pb \quad (6)$$

898 Now the distinguish step is similar to the one of
899 the G-J. That is, by comparing the last *m - r* elements of
900 $L^{-1}Pb$ with 0, redundant and conflicting constraints can be
901 distinguished. However, one has to notice that the deduction
902 process is under the condition that *L* should be invertible.

903 To the best of our knowledge, using this method to
904 detect geometric over-constraints is not convincingly
905 demonstrated in literature. Evaluations of the method
906

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:

$$915 \left\| A_k^{(k)}(k : m) \right\|_2 = \max_{j \geq k} \left\| A_j^{(k)}(k : m) \right\|_2 \quad (7)$$

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

917 As it is shown in the Fig. 16, *P* is the permutation
918 matrix where the information of exchanging columns is
919 stored. *R* is a triangular matrix where rank *r* is the number
920 of non-zeros diagonal elements. Equations corresponding
921 to $A^t.p[:, r + 1 : m]$ are the over-constraints [28].
922

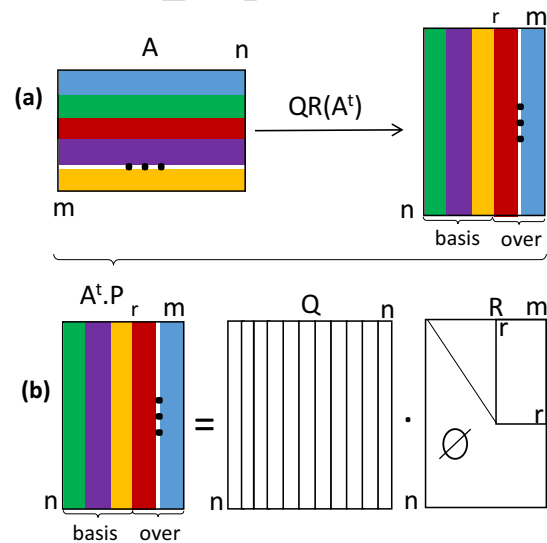
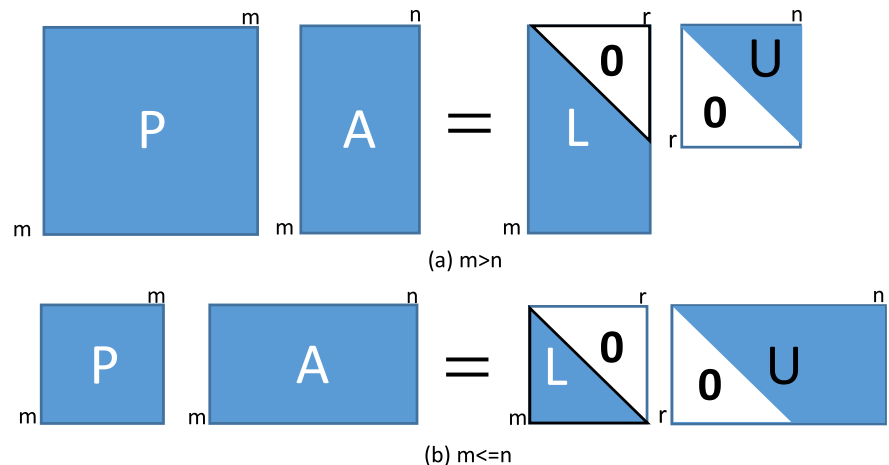


Fig. 16 QR Factorization with column pivoting

Fig. 15 LU Factorization with partial pivoting



923 Similar to the LU, further deduction is needed to distinguish
 924 redundant and conflicting constraints. First, the matrix
 925 $Q(:, 1 : r)$ is inverted using the following equation:

926 $A^t(:, 1 : r) = Q(:, 1 : r).R(1 : r, 1 : r)$ (8)

927 and is then used in the following equation:

929 $A^t(:, r + 1 : n) = Q(:, 1 : r).R(1 : r, r + 1 : n)$ (9)

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

933 $A^t(:, r + 1 : n) = A^t(:, 1 : r).R(1 : r, 1 : r)^{-1}R(1 : r, r + 1 : n)$
 934 (10)

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

942 $b_{new} = b(r + 1 : n) - b(1 : r).R(1 : r, 1 : r)^{-1}R(1 : r, r + 1 : n)$
 943 (11)

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

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

952 **4.2.2 Non-Linear Methods**

953 Detecting non-linear geometric constraints systems is more
 954 complicated than linear ones. Since non-linear detection
 955 methods such as symbolic methods using abstract algebra,
 956 we introduce the mathematical fundamentals to make following
 957 discussions easy to understand. The following two theorems
 958 are induced from [29]. Readers can find more details about
 959 concepts like ideals, affine varieties etc. in the book.
 960

961 **Theorem 1** For a system of polynomial equations
 962 $f_0 = f_1 = \dots = f_s = 0$, where $f_0, f_1, \dots, f_s \in \mathbb{C}[x_1, \dots, x_n]$; If
 963 affine variety $W(f_1, \dots, f_s) \neq \emptyset$ while $W(f_0, f_1, \dots, f_s) = \emptyset$
 964 , then $f_0 = 0$ is a conflicting equation; If
 965 $W(f_0, f_1, \dots, f_s) = W(f_1, \dots, f_s) \neq \emptyset$, then $f_0 = 0$ is a redundant
 966 equation; If $W(f_0, f_1, \dots, f_s) \neq \emptyset, W(f_1, \dots, f_s) \neq \emptyset$ and

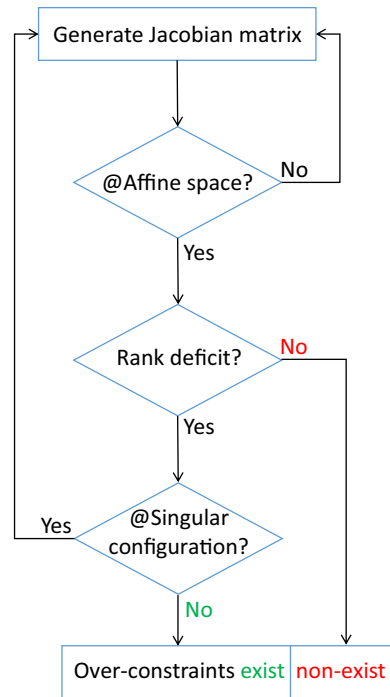


Fig. 17 Over-constraints detection based on the Jacobian matrix analysis

$W(f_0, f_1, \dots, f_s) \neq W(f_1, \dots, f_s)$, then $f_0 = 0$ is an independent
 967 equation. 968

Theorem 2 (Hilbert’s weak Nullstellensatz) Let k be an
 969 algebraically closed field. If $f, f_1, \dots, f_s \in k[x_1, \dots, x_n]$ are such
 970 that $f \in I(W(f_1, \dots, f_s))$, then there exists an integer $m \geq 1$
 971 such that $f^m \in \langle f_1, \dots, f_s \rangle$ (and conversely). 972

Based on the Theorem 2, Michelucci et al [15] deduced
 973 the Corollary 1. 974

Corollary 1 Let k be an algebraically closed field and
 975 $W(f_1, \dots, f_s) \neq \emptyset$. If f, f_1, \dots, f_s have the common root w ,
 976 then $\text{rank}([f'(w), f_1'(w), \dots, f_s'(w)]^T) < s + 1$. 977

Informally, the corollary 1 tells that if a system of
 978 polynomial equations containing redundant equations,
 979 then the Jacobian matrix of the equations at the affine
 980 space (solution space) must be row rank deficiency. How-
 981 ever, the reverse is not correct. In other words, if there
 982 exists the Jacobian matrix whose rank is deficiency at the
 983 solution space, then system of polynomial equations does
 984 not necessarily contain redundant equations. A typical
 985 example is the singular configuration in the Fig. 8: the
 986

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 ($b\oplus$); 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 ($a, b\oplus$). However, the method does not enable to find the spanning group of the constraint ($c\ominus$).
- **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 ($d, e\ominus$). The method does not analyze the singularity of a configuration ($f\ominus$).

- **Criteria set 3** The method analyzes non-linear equations. Therefore, any geometries ($g\oplus\ominus$) with non-linear constraints ($h\oplus$) in 3D or 2D space ($j\oplus\ominus$) modeling at the level of equations ($i\oplus$) 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 ($k\ominus$). 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 $P = \{f_0 = f_1 = \dots = f_s = 0\}$, where $f_0, f_1, \dots, f_s \in \mathbb{Q}[x_1, \dots, x_n]$ represent system of constraints and $\text{Zero}(P)$ 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:

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

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*:

$$\text{Zero}(P) = \bigcup_{1 \leq i \leq k} \text{Zero}(TS\{i\}/I\{i\}) \quad (13)$$

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:

- 1072 • *Criteria set 1* As discussed above, their method enable
1073 to detect conflicting and redundant constraints (a,b⊕)
1074 but cannot find the spanning groups (c⊖).
- 1075 • *Criteria set 2* The method decomposes a set of polyno-
1076 mials into a union of zero sets in triangular form. No
1077 over-constrained subparts, rigid subsystems are gen-
1078 erated as well as singular configurations are ana-
1079 lyzed (d,e,f⊖).
- 1080 • *Criteria set 3* The method analyzes non-linear equa-
1081 tions. Any geometries (g⊕⊖) with non-linear con-
1082 straints (h⊕) in 3D or 2D space (j⊕⊖) modeling at
1083 the equation level (i⊕) can be applied with the method.
- 1084 • *Criteria set 4* The results are the same as those of eval-
1085 uating the Grobner basis method.

1086 Symbolic detection methods are sound in theory but
1087 the computation cost is high. As discussed previously,
1088 the worstcase can be doubly exponential. Moreover, the
1089 reduced Grobner basis has to be computed every time of
1090 analyzing an equation. Therefore, this method is limited
1091 to deal with large systems of equations.

1092 The second group of methods analyze the Jacobian
1093 matrix of a system of equations. Different from symbolic
1094 methods, these numerical methods are more practical
1095 in computation but are theoretical deficiency in some
1096 cases. On one hand, if the affine space of a system does not
1097 exist, an equivalent one that sharing similar the Jacobian
1098 structure should be found. On the other hand, even if the
1099 Jacobian matrix of a configuration is row rank deficiency
1100 in affine space, the corresponding configuration should
1101 not be singular.

1102 *Perturbation* Haug proposed a perturbation method to
1103 deal with singular configurations and detect redundant
1104 constraints in mechanical systems [38]. More precisely,
1105 assuming a system of equations $\Phi(q) = 0$ and the corre-
1106 sponding Jacobian matrix Φ_q is rank deficiency at q . As we
1107 discussed before, it is not sufficient to determine the exist-
1108 ence of the over-constraints since the singular configura-
1109 tion can also make a Jacobian matrix rank deficiency. He
1110 suggested to analyze the Jacobian matrix at more configu-
1111 rations with the following:

- 1112 • Add a small perturbation δq to q and obtain $\Phi_q \delta q = 0$
1113 . The process is based on the Implicit Function Theo-
1114 rem [39].
- 1115 • Applying the G-J to Φ_q , $\Phi_q \delta q = 0$ is transformed into
1116 $\begin{bmatrix} \Phi_u^I & \Phi_v^I \\ 0 & \Phi_v^R \end{bmatrix} \begin{bmatrix} \delta u \\ \delta v \end{bmatrix} = 0$. Φ_u^I is the upper triangular matrix
1117 with 1s as diagonal elements. Φ_v^R can be treated as the

1118 matrix with all 0s under given tolerance. Equations in
1119 $\Phi(q) = 0$ corresponding to Φ_u^I part: $\Phi^I(q) = 0$ are inde-
1120 pendent.

- 1121 • Now, $\Phi_q \delta q = 0$ can be simplified into $\Phi_u^I \delta u + \Phi_v^I \delta v = 0$
1122 a n d t h u s $\delta v = -(\Phi_v^I)^{-1} \Phi_u^I \delta u$,
- 1123 $\delta q = \begin{bmatrix} \delta u \\ \delta v \end{bmatrix} = \begin{bmatrix} \delta u \\ -(\Phi_v^I)^{-1} \Phi_u^I \delta u \end{bmatrix}$
- 1124 • Assume q is perturbed to new point q^* satisfying
1125 $q^* = q + \delta q$. To ascertain it lies in the affine space,
1126 is should satisfy $\Phi(q^*) = 0$. This is equivalent to
1127 $\Phi^I(q^*) = 0$ since the latter is composed of all inde-
1128 pendent equations of the former.
- 1129 • S o l v i n g $\Phi^I(q^*) = 0$, $q^* = q + \delta q$,
1130 $\delta q = \begin{bmatrix} \delta u \\ \delta v \end{bmatrix} = \begin{bmatrix} \delta u \\ -(\Phi_v^I)^{-1} \Phi_u^I \delta u \end{bmatrix}$, the value of q^* is
1131 obtained.
- 1132 • Computing the rank of the Jacobian matrix at q^* : Φ_{q^*}
1133 and checking if it is rank deficiency.

1134 As we can see from the above, obtaining an appropriate
1135 value of the perturbation δq so that q^* lies in the affine
1136 space is the main part of the method.

1137 Results of evaluating the method are as follows:

- 1138 • *Criteria set 1* The method enables to detect geometric
1139 over-constraints (a⊕) but does not distinguish redun-
1140 dant and conflicting constraints (b⊖). Finding the span-
1141 ning groups is also not supported (c⊖).
- 1142 • *Criteria set 2* The method mainly detects the over-
1143 constraints based on analyzing the Jacobian matrix of
1144 a whole system. There is no meaning to evaluate the
1145 method with respect to system decomposition crite-
1146 ria (d,e,f⊖).
- 1147 • *Criteria set 3* The method analyzes both linear and
1148 non-linear equation systems. Therefore, any geome-
1149 tries (g⊕⊖) with non-linear and linear constraints (h⊕⊖)
1150 in 3D or 2D space (j⊕⊖) modeling at the equation
1151 level (i⊕) can be applied with the method.
- 1152 • *Criteria set 4* The over-constraints are generated in a
1153 single-pass way since the Jacobian matrix analysis is on
1154 the whole system at once (k⊕). However, debugging the
1155 over-constraints is not discussed (l?).

1156 His method selects two points in the affine space to deter-
1157 mine the existence of geometric over-constraints. If the Jaco-
1158 bian matrix at any configuration is full rank, then there is no
1159 over-constraint. However, if the rank of the Jacobian matrix
1160 at both configurations is deficiency, then there exists geo-
1161 metric over-constraints.

1162 *NPM* Roots of system of equations can be sometimes
 1163 hard to find or even do not exist. In these cases, the aff-
 1164 ine space does not exist. Sebti Foufou et al. [7] suggested a
 1165 Numerical Probabilistic Method (NPM), which is to test the
 1166 Jacobian matrix at random configurations instead of the aff-
 1167 ine space. However, there is a risk that the Jacobian matrix
 1168 is row rank deficiency at the chosen configuration and the
 1169 corresponding configuration happens to be singular. They
 1170 suggested to test more configurations to reduce the possibil-
 1171 ity of happening such case. Moreover, in order to get more
 1172 confidence, authors suggested that testing at 10 different
 1173 configurations should be sufficient. The NPM is practical
 1174 in computation but is not sound in theory since the testing
 1175 configurations are not necessarily all in affine space.

1176 Results of evaluating the method are as follows:

- 1177 • *Criteria set 1* The method enables to identify numeri-
 1178 cal over-constraints ($a\oplus$). However, it can neither distin-
 1179 guish redundant and conflicting constraints nor finding
 1180 the spanning group of an over-constraint ($b,c\ominus$).
- 1181 • *Criteria set 2* The method can be used to decompose a
 1182 system into rigid subsystems ($e\ominus$). However, decomposi-
 1183 tion into over-constrained components as well as analyz-
 1184 ing singular configurations are not supported ($d,f\ominus$).
- 1185 • *Criteria set 3* The method analyzes both linear and
 1186 non-linear system of equations. Therefore, any geome-
 1187 tries ($g\oplus\ominus$) with non-linear or linear constraints ($h\oplus\ominus$)
 1188 in 3D or 2D space ($j\oplus\ominus$) modeling at the level of equa-
 1189 tions ($i\oplus$) can be applied with the method.
- 1190 • *Criteria set 4* Numerical over-constraints are detected all
 1191 at once ($k\oplus$) but debugging them is not discussed ($l?$).

1192 *WCM* Instead of selecting configurations randomly,
 1193 Michelucci et al. suggested to study the Jacobian structure at
 1194 witness configurations where incidence constraints are satis-
 1195 fied [40]. A witness configuration and the target configura-
 1196 tion shares the same Jacobian structure, where the Jacobian

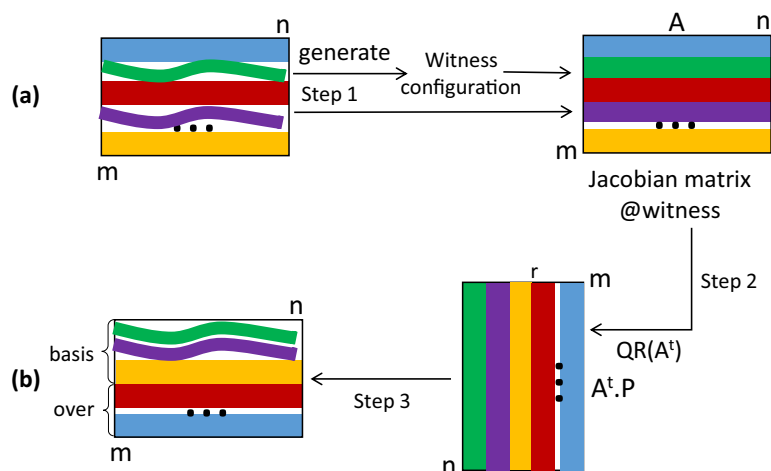
1197 matrix is non-singular in affine space. As a consequence, all
 1198 the numerical over-constraints can be identified [15]. More
 1199 recently, Moinet et al. developed tools to identify conflicting
 1200 constraints through analyzing the witness of a linearized sys-
 1201 tem of equations [41]. Their approaches have been success-
 1202 fully applied to the well-known double banana geometry to
 1203 find the numerical over-constraints.

1204 For a geometric constraints system represented with a set
 1205 of equations $F(U, X) = 0$, where U denotes a set of param-
 1206 eters with prescribed values U_T (T for target), and X is the
 1207 vector of unknowns. The solution is denoted as X_T . A wit-
 1208 ness is a couple (U_W, X_W) such that $F(U_W, X_W) = 0$. Most
 1209 of the time, U_W and X_W are different from U_T and X_T respec-
 1210 tively. The witness (U_W, X_W) is not the solution but shares
 1211 the same combinatorial features with the target (U_T, X_T) ,
 1212 even if the witness configuration and the target configuration
 1213 lie on two distinct connected components of the solution
 1214 set. Therefore, analyzing a witness configuration enables to
 1215 detect numerical over-constraints of a system [42, 43]. These
 1216 numerical over-constraints can be not only structural over-
 1217 constraints but also geometric redundancies.

1218 The Fig. 18 shows the witness method combining the
 1219 QR for detection. Step one aims at generating the witness
 1220 configuration while at step two, the QR is applied on the
 1221 Jacobian matrix A . As a result, the rows of equations are re-
 1222 ordered by P and the number of basis constraints is revealed
 1223 by r . Finally, coming back to the re-ordered original equa-
 1224 tions, the first r equations are the basis constraints while the
 1225 remaining ones are the numerical over-constraints. Note that,
 1226 the QR can be replaced with the G-J in the process, which
 1227 would generate results different from the ones of QR since
 1228 the two methods adopt different sorting rows strategies.

1229 The results of evaluating the method are the same as those
 1230 of evaluating the NPM method. Michelucci et al [15] proved
 1231 that the WCM can identify all the dependencies among con-
 1232 straints. In other words, if removing these dependent con-
 1233 straints, the remaining constraints are independent. However,

Fig. 18 Witness configuration method



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 ($e\ominus$) but finding over-constrained components is not discussed ($d?$). In terms of finding the spanning groups, it is not supported ($f\ominus$).
- *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 ($k\ominus$). 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

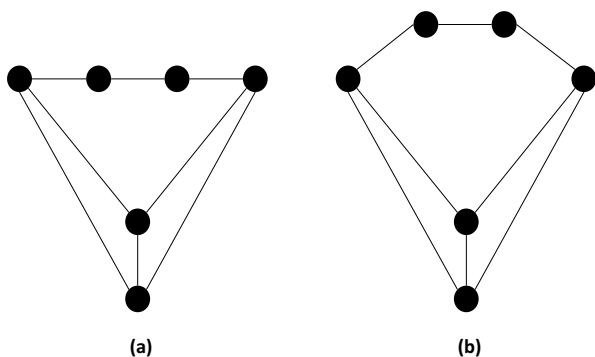


Fig. 19 a Rigid sketch b Non-rigid sketch [44]

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.

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

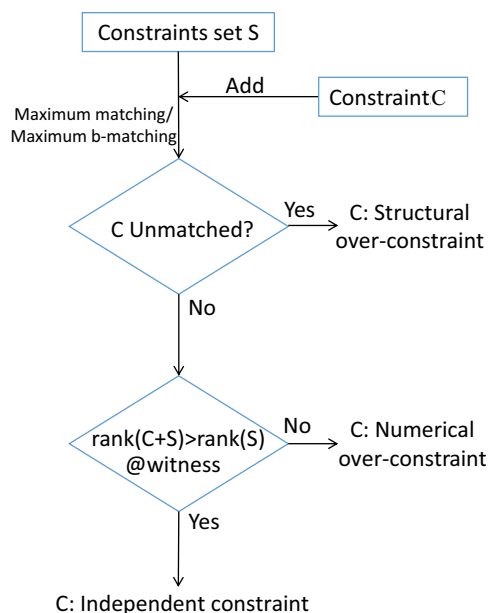


Fig. 20 Incremental detection framework where constraints are added one by one

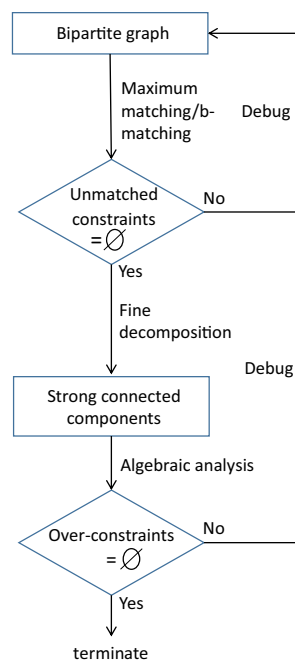


Fig. 21 Decremental detection framework

1302 the constraints and associated equations have been pre-
 1303 defined. Through analyzing methods in previous sections, we
 1304 propose two detection frameworks: incremental detection
 1305 framework and decremental detection framework. Both of
 1306 them are based on a combination of structural and algebraic
 1307 methods. These methods are listed in the Table 7 and Table 8
 1308 respectively. Details of the two frameworks will be discussed
 1309 in the next subsections.

1310 4.3.1 Incremental Detection Framework

1311 Here, we assume that the constraint C is to be added to
 1312 a set of constraints S . This framework is to test if C is an
 1313 over-constraint with respect to S . The first method is either
 1314 the D-M method or the MWM method, which detects
 1315 structural over-constraints using either maximum match-
 1316 ing method or maximum b-matching method. The method
 1317 will be applied to the new group $S + C$ after adding C . If C
 1318 is unmatched, then C is a structural over-constraint. Other-
 1319 wise, we apply the WCM method to detect numerical over-
 1320 constraints of $S + C$. If the rank of the new system $S + C$
 1321 is bigger than that of S at witness configurations, C is an

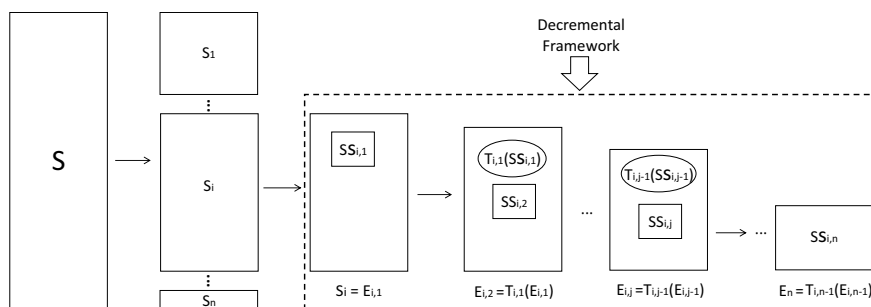
independent constraint otherwise it is a numerical over-con-
 1322 straint. Whether it is redundant or conflicting can be checked
 1323 using the Grobner basis method or the Incremental solving
 1324 method. In this framework, since constraints are added incre-
 1325 mentally, users can be informed directly if a newly inserted
 1326 constraint has been detected as an over-constraint (Fig. 20).
 1327

The advantage of this framework is that it enables design-
 1328 ers to detect and treat the over-constraints as soon as they are
 1329 detected in the modeling process.
 1330

1331 4.3.2 Decremental Detection Framework

1332 Decremental detection analyzes a set of existing con-
 1333 straints. The constraints set and its associated equations
 1334 set are initially represented with a bipartite graph. Struc-
 1335 tural over-constraints will be identified using either
 1336 the D-M method or the MWM method if there exists
 1337 unmatched constraints after maximum matching (or
 1338 b-matching). They will be removed and the system
 1339 will be updated. If there is no unmatched constraints,
 1340 strong connected components (that is, irreducible sub-
 1341 systems using the D-M method or balanced sets using
 1342 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}



1343 of the system. Then, algebraic methods are used to detect
 1344 numerical over-constraints inside each component. Since
 1345 strong connected components linked with solving order
 1346 is actually a DAG structure, components corresponding
 1347 to the source vertices are usually analyzed first. Once an
 1348 over-constraint is found, it is presented to the user for
 1349 debugging. After that, the system is updated and the
 1350 corresponding bipartite graph is rebuilt. The detection
 1351 process finishes when no numerical over-constraints are
 1352 found (Fig. 21).

1353 The advantage of this framework is that the treat-
 1354 ment of the detected over-constraints can be performed
 1355 on the entire system, which better considers the design
 1356 intent. However, if a final system after modelling is too
 1357 large, it would be preferable to detect over-constraints
 1358 incrementally during the modeling process rather than
 1359 analyze them decrementally after modeling.

1360 **4.3.3 A Decomposition-Detection Plan**

1361 The first requirement of a Decomposition-Detection (D-D)
 1362 plan is that it should be able to find local segments of
 1363 a system of constraints if there exists any. For example,
 1364 decomposes a free-form configuration into local parts due
 1365 to the local support property of the geometry. Secondly, a
 1366 D-D plan should decompose a constraint system into small
 1367 subsystems and analyze these subsystems using algebraic
 1368 methods. Since the time cost of over-constraints detec-
 1369 tion is proportional (at least polynomial) to the size of
 1370 a system, these small subsystems should be as small as
 1371 possible so that algebraic methods can analyze them with
 1372 low computational cost. If there is no over-constraints in a
 1373 subsystem, the subsystem would be solved and the solution
 1374 would be propagated to the entire system resulting in a
 1375 simplified system. As it is shown in the Fig. 22, a D-D plan
 1376 initially decomposes the system S into $\{S_1, \dots, S_i, \dots, S_n\}$
 1377 local segments. Then, for each local segment S_i , the D-D

plan proceeds by applying the following steps at each iteration j : 1378 1379

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 ($SS_{i,1}$). 1380 1381 1382
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. 1383 1384 1385 1386 1387
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 propagating 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). 1388 1389 1390 1391 1392 1393 1394 1395 1396

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 $E_{i,j+1}$. 1397 1398 1399 1400 1401 1402 1403

4.4 Hybrid Approaches 1404

Serrano Serrano analyzed a system of equations ($h\oplus$) to select a well constrained, solvable subsets from candidate constraints[24]. His method first detects structural over-constraints ($a\ominus$) if there are equations uncovered after maximum matching. To further detect numerical over-constraints ($a\oplus$) within strong connected components ($e\ominus$), symbolic and 1405 1406 1407 1408 1409 1410

Table 9 Evaluation of the methods

Criteria set	Detection level			Decomposition			System modeling			Results generation			
	Def	Type	Redundant conflicting	Spanning group	Over-constrained components	Rigid sub-systems	Singular configuration	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	⊕⊖	⊕	⊕	⊖	⊖	⊖	⊕⊕	⊕⊕	⊕	⊕⊕	⊖	⊕

1460 detection needs to consider the multi-dimensional infor-
 1461 mation describing a geometric constraints system, which
 1462 needs to be extracted through a geometric system modeling,
 1463 decomposition and solving process. Various works in litera-
 1464 ture are addressing these issues to some extent; however,
 1465 they take into consideration only some configurations and
 1466 are applicable in some conditions. Therefore, efforts are
 1467 still needed to address the challenging applications such as
 1468 PDP. We foresee that the design of hybrid approaches will
 1469 enable advances toward practical requirements. In particu-
 1470 lar, a method that makes as much use as possible of prede-
 1471 fined patterns, and resorts to a general DoF-based analysis
 1472 strengthened by a WCM-based validation in a recursive
 1473 assembly way (allowing to interleave decomposition, solv-
 1474 ing, propagation, and recombination phases) would be more
 1475 applicable towards generality and reliability.

1476 **Acknowledgements** The authors are grateful to the China Scholar-
 1477 ship Council (No. 201406090176) for supporting this research. We are
 1478 also immensely grateful to (Jean-Philippe Pernot and Mathias Kleiner)
 1479 for their guidances on an earlier version of the manuscript, although
 1480 any errors are our own and should not tarnish the reputations of these
 1481 esteemed persons.

1482 Compliance with ethical standards

1483 **Conflict of interest** The authors declare that they have no known compet-
 1484 ing financial interests or personal relationships that could have ap-
 1485 peared to influence the work reported in this paper.

1486 References

- 1487 1. Falcidieno B, Giannini F, Léon JC, Pernot JP (2014) Processing
 1488 free form objects within a Product Development Process frame-
 1489 work, vol 1. ASME-Press, New York, pp 317–344
- 1490 2. Cheutet V, Daniel M, Hahmann S, La Greca R, Léon JC, Maculet
 1491 R, Ménegaux D, Sauvage B (2007) Constraint modeling for curves
 1492 and surfaces in CAGD: a survey. *Int J Shape Model* 13(02):159
- 1493 3. Gouaty G, Fang L, Michelucci D, Daniel M, Pernot JP, Raffin R,
 1494 Lanquetin S, Neveu M (2016) Variational geometric modeling
 1495 with black box constraints and DAGs. *Comput Aided Des* 75:1–12
- 1496 4. Graver J, Servatius B, Servatius H (1993) Combinatorial rigidity.
 1497 Graduate studies in mathematics. American Mathematical Soci-
 1498 ety, New York
- 1499 5. Laman G (1970) On graphs and rigidity of plane skeletal struc-
 1500 tures. *J Eng Math* 4(4):331–340
- 1501 6. Podgorelec D, Žalik B, Domiter V (2008) Dealing with redun-
 1502 dancy and inconsistency in constructive geometric constraint solv-
 1503 ing. *Adv Eng Softw* 39(9):770–786
- 1504 7. Foufou S, Michelucci D, Jurzak JP (2005) Numerical decompo-
 1505 sition of geometric constraints. Proceedings of the 2005 ACM
 1506 symposium on solid and physical modeling. ACM, pp 143–151
- 1507 8. Sitharam M, Zhou Y (2004) A tractable, approximate, combina-
 1508 torial 3d rigidity characterization. Fifth Automated Deduction in
 1509 Geometry (ADG)
- 1510 9. Jermann C, Neveu B, Trombettoni G (2003) Algorithms for iden-
 1511 tifying rigid subsystems in geometric constraint systems. In: 18th
 1512 International Joint Conference in Artificial Intelligence, IJCAI-
 1513 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 1514
11. Ait-Aoudia S, Jegou R, Michelucci D (2014) Reduction of con- 1515
 straint systems. [arXiv:1405.6131](https://arxiv.org/abs/1405.6131) 1516
12. Hu H, Kleiner M, Pernot JP (2017) Over-constraints detection 1517
 and resolution in geometric equation systems. *Comput Aided Des* 90:84–94 1518
13. Diestel R (2017) Graph theory. GTM 173, 5th edition, Vol 173. 1519
 Springer, Heidelberg Graduate Texts in Mathematics 1520
14. Welsh D (2010) Matroid theory. Dover books on mathematics. 1521
 Dover Publications, New York 1522
15. Michelucci D, Foufou S (2006) Detecting all dependences in sys- 1523
 tems of geometric constraints using the witness method. In: Inter- 1524
 national Workshop on Automated Deduction in Geometry, pp 1525
 98–112. Springer, Berlin 1526
16. Coxeter HSM, Greitzer SL (1967) Geometry revisited, vol 19. 1527
 Maa, New York 1528
17. Peng X, Chen L, Zhou F, Zhou J (2002) Singularity analy- 1529
 sis of geometric constraint systems. *J Comput Sci Technol* 17(3):314–323 1530
18. Fudos I, Hoffmann CM (1997) A graph-constructive approach 1531
 to solving systems of geometric constraints. *ACM Trans Gr (TOG)* 16(2):179–216 1532
19. Brüderlin B, Roller D (eds) (2012) Geometric constraint solving 1533
 and applications. Springer Science & Business Media 1534
20. Hoffmann CM, Sitharam M, Yuan B (2004) Making constraint 1535
 solvers more usable: overconstraint problem. *Comput Aided Des* 36(4):377–399 1536
21. Hoffmann CM, Lomonosov A, Sitharam M (1997, October). Find- 1537
 ing solvable subsets of constraint graphs. In: International Con- 1538
 ference on Principles and Practice of Constraint Programming. 1539
 Springer, Berlin, pp 463–477 1540
22. Dulmage AL, Mendelsohn NS (1958) Coverings of bipartite 1541
 graphs. *Can J Math* 10:517–534 1542
23. Fritzson P, Bunus P (2002, April) Modelica—a general object- 1543
 oriented language for continuous and discrete-event system mod- 1544
 eling and simulation. In: Proceedings 35th Annual Simulation 1545
 Symposium. SS 2002, IEEE, pp 365–380 1546
24. Serrano D (1987) Constraint management in conceptual design, 1547
 Doctoral dissertation, Massachusetts Institute of Technology 1548
25. Strang G (2014) Linear algebra and its applications. Elsevier 1549
 Science 1550
26. Light RA, Gossard DC (1983) Variational geometry: a new 1551
 method for modifying part geometry for finite element analysis. 1552
Comput Struct 17(5–6):903–909 1553
27. Okunev P, Johnson CR (2005) Necessary and sufficient conditions 1554
 for existence of the LU factorization of an arbitrary matrix. [arxiv: 1555
 math/0506382](https://arxiv.org/abs/math/0506382) 1556
28. Dongarra J, Kennedy K, Messina P, Sorensen DC, Voigt RG 1557
 (1990) Proceedings of the Fourth SIAM Conference on Par- 1558
 allel Processing for Scientific Computing, Chicago, Illinois, 1559
 USA. SIAM 1990, ISBN 0-89871-262-9 1560
29. Cox D, Little J, OShea D (2013) Ideals, varieties, and algorithms: 1561
 an introduction to computational algebraic geometry and com- 1562
 mutative algebra. Springer 1563
30. Bose NK (2013) Multidimensional systems theory and applica- 1564
 tions. Springer 1565
31. Chou SC (1988) An introduction to Wu’s method for mechanical 1566
 theorem proving in geometry. *J Automat Reason* 4(3):237–267 1567
32. Wu WT (2012) Mechanical theorem proving in geometries: basic 1568
 principles. Springer 1569
33. Cox DA, Little J, O’Shea D (2015) Ideals, varieties, and algo- 1570
 rithms. Springer, New York, pp 49–119 1571
34. Hoffmann CM (1989) Geometric and solid modeling: an introduc- 1572
 tion. Morgan Kaufmann Publishers Inc., San Francisco 1573
 1574
 1575
 1576
 1577
 1578
 1579

- 1580 35. Kondo K (1992) Algebraic method for manipulation of dimensional relationships in geometric models. *Comput Aided Des* 24(3):141–147 1604
- 1581 36. Chou S (1988) Mechanical geometry theorem proving. *Mathematics and its applications*. Springer, New York 1605
- 1582 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 1606
- 1583 38. Haug EJ (1989) Computer aided kinematics and dynamics of mechanical systems, vol 1. Allyn and Bacon, Boston, pp 48–104 1607
- 1584 39. Krantz S, Parks H (2012) The implicit function theorem: history, theory, and applications. *Modern Birkhäuser classics*. Springer, New York 1608
- 1585 40. Michelucci D, Foufou S (2006) Geometric constraint solving: the witness configuration method. *Comput Aided Des* 38(4):284–299 1609
- 1586 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 1610
- 1587 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 1611
- 1588 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 1612
- 1589 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 1613
- 1590 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 1614
- 1591 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](https://arxiv.org/abs/1904.00526) 1615
- 1592 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 1616
- 1593 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 1617
- 1594 1618
- 1595 1619
- 1596 1620
- 1597 1621
- 1598 1622
- 1599 1623
- 1600 1624
- 1601
- 1602
- 1603
- Publisher's Note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

REVISED PROOF