A New Structural Rigidity for Geometric Constraint Systems

Christophe Jermann*[1], Bertrand Neveu[1], and Gilles Trombettoni[1]

{Christophe.Jermann, Bertrand.Neveu, Gilles.Trombettoni}@sophia.inria.fr
COPRIN Team, INRIA-I3S-CERMICS,
2004 route des lucioles, BP 93, 06902 Sophia.Antipolis, France

Abstract. The structural rigidity property, a generalization of Laman's theorem which characterizes rigid bar frameworks in 2D, is generally considered a good heuristic to detect rigidities in a geometric constraint system. The gap between rigidity and structural rigidity essentially resides in the fact that structural rigidity is not a geometric property.
In this article, we propose a new definition of structural rigidity which takes into account geometric aspects of rigidity. We also propose a new flow-based algorithm for identifying geometric constraint systems verifying our new definition. This algorithm works on a geometric constraint network introduced by Hoffmann et al.

1 Introduction

Geometric constraint satisfaction problems arise naturally in several areas, such as architecture, design of mechanisms and molecular biology. The rigidity concept is in the heart of many of these problems: deciding whether a geometric constraint system is rigid or not, detecting rigid sub-parts, detecting over-rigid sub-parts, and so on.

Several solving methods [Kra92,BFH$^+$95,LM97,DMS97,HLS00,JTNR00] for geometric constraint systems have to handle rigidity related problems. In particular, recursive rigidification techniques decompose a geometric system into a sequence of rigid subsystems to be solved separately and then assembled.

The techniques used so far for rigidity detection can be classified in two categories: pattern-based approaches [FH93,BFH$^+$95,Kra92] depend on a repertoire of rigid bodies of known *shape* which cannot cover all practical instances. Flow-based approaches [HLS97,LM97] use flow (or maximum matching) machinery to identify subsystems verifying a structural property: the *structural rigidity*. This property is based on a degree of freedom count.

The latter approaches are more general eventhough structural rigidity is only an approximation of rigidity. Heuristics have been proposed to enhance structural rigidity capabilities, none of which succeeded to fully cover the gap between structural rigidity and rigidity.

* Supported by CNRS and Région Provence Alpes Côte d'Azur

In this paper, we present a thorough analysis of this gap (Section 2.2). This results in defining a new concept: the *degree of rigidity* of a geometric system (Section 2.3). We propose a new definition of structural rigidity based on this concept and discuss its interest.

In the second part of this paper, we present an algorithm which identifies systems verifying our new definition (Section 3). It is derived from the algorithm `Dense` proposed by Hoffmann et al [HLS97] for previous structural rigidity. The complexity and practical use of our algorithm is discussed (Section 4).

2 Structural rigidity and degree of rigidity

First, this section provides the necessary background definitions. Then, it introduces the structural rigidity concept and proposes an analysis of the gap between structural rigidity and rigidity. A general explanation for one of the differences between rigidity and structural rigidity is proposed: the degree of rigidity concept. Our new definition of structural rigidity is based on this concept. This definition is superior to the previous structural rigidity definition: we prove it and illustrate this fact on a simple example.

2.1 Basics

Let us first define a geometric constraint system (an example is provided in figure 1).

Definition 1 Geometric constraint system
A **geometric constraint system***, also referred to as* geometric system*, S in a geometric space of dimension d is defined by a pair (O, C), and noted $S = (O, C)$, where:*

- *O is a set of geometric objects in dimension d,*
- *C is a set of geometric constraints binding objects in O.*

$S' = (O', C')$ is a **subsystem** *of $S = (O, C)$ (noted $S' \subset S$) iff $O' \subset O$ and every constraint in $C' \subset C$ binds only objects in O' (i.e. S' is induced by O').*

In this paper, we limit our study to indeformable objects; for example, variable-radius circles are not considered. Also, geometric constraints cannot involve dimension parameters; for instance, variable-distance constraints are not allowed. These limitations are intrinsic to recursive rigidification methods[1] in particular, and simplify the following definitions.

A solution to a geometric system S is a set C_S containing one position and one orientation for each geometric object in O and satisfying all the geometric constraints in C. For the solving purpose, a geometric system is translated into a system of equations: each object is represented by a set of unknowns (over the reals) called *parameters* which determine its position and orientation; each constraint becomes a subsystem of equations on the parameters of the objects it constrains.

[1] Work has been done to partially overcome these restrictions [JASR99].

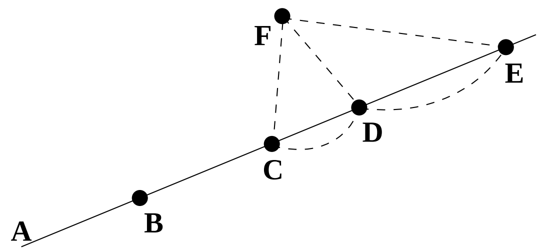

Fig. 1. A geometric system in 3D composed of one line (A) and 5 points (B, C, D, E and F). The constraints are 4 point-line incidences (B, C, D and E on A) and 5 point-point distances (CD, CF, DE, DF and EF) represented in dots.

Intuitively, a geometric system is rigid if it is indeformable and can be displaced anywhere in the considered geometric space. It is under-rigid if it is deformable, and over-rigid if it cannot be displaced anywhere or is over-constrained (no solution). More formally, rigidity is defined[2] as follows:

Definition 2 Rigidity
A **flex** *of a geometric system S is a function which associates one solution of S to each real in $[0, 1]$. It is* **trivial** *if it corresponds to a rigid-body motion (translation+rotation).*

A geometric system S is **rigid** *if it admits only, and all, the trivial flexes. It is* **under-rigid** *if it admits all the trivial flexes and at least a non-trivial flex; it is* **over-rigid** *if it does not admit all the trivial flexes or is over-constrained.*

In the example of figure 1, the subsystem CDF is rigid since a triangle is indeformable and can be displaced anywhere in the Euclidean 3D space. The subsystem AF is under-rigid: the point F moves independently from the line A since they are not constrained. The subsystem $ACDEF$ is over-rigid since, in the generic case, it has no solution.

2.2 Structural Rigidity

The *structural rigidity* corresponds to an analysis of *degrees of freedom* (in short dof) in a geometric system. Intuitively, one dof represents one independent movement in a geometric system. More formally:

Definition 3 Degree of freedom (dof)
The number of degrees of freedom $dof(o)$ offered by a geometric object o is the number of independent parameters that must be set to determine its position and orientation.

[2] These definitions are borrowed from the Structural Topology community which has studied various types of rigidity [Whi87].

The number of degrees of freedom $dof(c)$ removed by a geometric constraint c is the number of parameters the constraint allows to determine[3].

The number of degrees of freedom $dof(S)$ of a geometric system $S = (O, C)$ is given by the following formula: $dof(S) = \sum_{o \in O} dof(o) - \sum_{c \in C} dof(c)$.

For instance, 3D points have 3 dof, 3D lines have 4 dof. Point-line incidences remove 2 dof, and point-point distances remove 1 dof. Hence, subsystems CDF, AF and $ACDEF$ from our example have respectively 6, 7 and 5 dof.

Laman's theorem characterizes generic rigidity of 2D bar frameworks through a dof analysis:

Theorem 1 [Lam70]
A bar framework $S = (O, C)$ in 2D is generically rigid iff:

1. $dof(S) = 3$,
2. $\forall S' \subset S$, $dof(S') \geq 3$.

Structural rigidity is a generalization of Theorem 1 to geometric systems composed of objects and constraints of any type and in any dimension:

Definition 4 Structural rigidity (s_rig1)
A geometric system $S = (O, C)$ in dimension d is **structurally rigid** *(in short s_rig1) iff:*

1. $dof(S) = \frac{d(d+1)}{2}$,
2. $\forall S' \subset S$, $dof(S') \geq \frac{d(d+1)}{2}$.

S is under-s_rig1 iff $dof(S) > \frac{d(d+1)}{2}$ and contains no over-s_rig1 subsystem;
S is over-s_rig1 iff $\exists S' \subseteq S$, $dof(S') < \frac{d(d+1)}{2}$.

Obviously, structural rigidity is equivalent to rigidity for bar frameworks in 2D (Laman's theorem). In the general case, it is only considered a *good heuristic* to identify rigid (or over-rigid) systems [Hen92,LM97,HLS97]. The main cause for the gap between rigidity and structural rigidity is well known: redundant systems mislead structural rigidity. Indeed, redundant constraints remove dof that are already removed by other constraints. Detecting redundancies is a difficult problem in itself, especially in non-generic systems: it is equivalent to deciding whether or not there are dependencies in an equation system.

We highlight here another limit which applies even in the non-redundant case: there exist geometric systems in dimension d which are rigid but have less than $\frac{d(d+1)}{2}$ dof.

For instance, let us consider the subsystem CD in our example of figure 1. This subsystem is a segment in 3D, which is a rigid body: a segment is indeformable and can be displaced anywhere in the Euclidean 3D space. However, this

[3] In practice, we take the number of independent equations in the subsystem of equations representing the constraint. This approximation, which comes from the traditional assumption that one equation fixes one unknown over the reals, is not always correct: $x^2 + y^2 = 0$ is a single equation fixing two unknowns at a time.

subsystem has only 5 dof. Since $\frac{d(d+1)}{2} = 6$ in dimension $d = 3$, it is over-s_rig1 and is then identified as over-rigid according to this definition of structural rigidity.

Moreover, according to the second condition in definition 4, any system embedding an over-s_rig1 subsystem is over-s_rig1. Hence, any system including a segment will be over-s_rig1: the first definition of structural rigidity seems very inaccurate for practical geometric systems in 3D since it cannot be used as long as the system contains points linked by distance constraints.

To overcome this problem, a common heuristic [Sit00] recommends to consider only *non-trivial* subsystems, i.e. systems composed of at least $d + 1$ objects in dimension d. This heuristic, while interesting for systems composed only of points, does not eliminate the problem of definition 4. Indeed, there exist non-trivial rigid geometric systems having less than $\frac{d(d+1)}{2}$ degrees of freedom.

Let us consider the subsystem $ACDE$ in figure 1. This subsystem is non-trivial since it is composed of 4 objects in 3D. Moreover, this subsystem is rigid: the three points can only be translated altogether along the line thanks to distance constraints, and the line can be displaced anywhere in the Euclidean 3D space. However, this subsystem has only 5 dof, which renders it over-s_rig1.

On the other hand, the subsystem $ABCD$ is also non-trivial but clearly under-rigid: point B can move along line A independently of points C and D, which renders this subsystem deformable. However, this subsystem has 6 dof, which makes it s_rig1.

In addition, using this heuristic, it becomes impossible to characterize trivial subsystems; this implies inability to detect the triangle CDF or even the segment CD, but also inability to detect over-rigidities in trivial subsystems, which may mislead the answer for systems containing trivial over-rigid subsystems.

All the counter-examples we have presented have a common characteristic: when rigid, they have *less* degrees of freedom than the number expected by definition 4. This means that, instead of looking for $\frac{d(d+1)}{2}$ degrees of freedom in any subsystem in dimension d, one should look for a number of dof *depending on the considered subsystem*. This number, called the *degree of rigidity*, is described in the following section.

2.3 Degree of rigidity

The degree of rigidity is defined for a set of geometric objects O' in the context of a geometric system S embedding these objects. We note this degree of rigidity $dor(O', S)$. It represents the number of degrees of freedom the induced subsystem $S' = (O', C') \subseteq S$ is expected to have if it is rigid. We define the following problem:

Problem 1 DOR
Data: $S = (O, C)$ *a geometric system,* $O' \subseteq O$ *a set of geometric objects in* S *and* k *a positive integer;*
Question: *Is* $dor(O', S)$ *equal to* k*?*

This problem is difficult in the general case. We discuss its complexity and present practical solutions in section 4.

Given the concept of degree of rigidity, we propose this new definition of structural rigidity:

Definition 5 Structural Rigidity (s_rig2)
A geometric system $S = (O, C)$ in dimension d is **structurally rigid** *(in short s_rig2) iff:*

1. $dof(S) = dor(O, S)$,
2. $\forall S' = (O', C') \subset S$, $dof(S') \geq dor(O', S)$.

S is under-s_rig2 iff $dof(S) > dor(O, S)$ and contains no over-s_rig2 subsystem; S is over-s_rig2 iff $\exists S' = (O', C') \subseteq S$, $dof(S') < dor(O', S)$.

Table 1 presents the answers given by the different structural rigidities over some subsystems of our example. Each line in this table presents one subsystem. There is one column in this table for rigidity, one for each structural rigidity and two more ones for dof and dor of each subsystem. The column *s_rig1+h* corresponds to s_rig1 applied only to non-trivial subsystems.

	Rigidity	dof	s_rig1	s_rig1+h	dor	s_rig2
CD	exact	5	over	trivial	5	exact
CDF	exact	6	over	trivial	6	exact
BCD	under	7	over	trivial	5	under
ACDE	exact	5	over	over	5	exact
ABCD	under	6	over	exact	5	under
ACDEF	over	5	over	over	6	over
ABCDE	under	6	over	over	5	under
ABCDEF	over	6	over	over	6	over

Table 1. Comparison table for rigidity and structural rigidities on subsystems of the example presented in figure 1.

This table illustrates the problems with s_rig1 and the associated heuristic. In particular, one can see that every subsystem of our example that contains a segment is considered over-rigid by s_rig1. For s_rig1+h, we see that non-trivial systems having $dor < 6$ ($\frac{d(d+1)}{2} = 6$ in dimension $d = 3$) can mislead s_rig1. This is the case for the subsystem $ACDE$ which is over-s_rig1 while rigid. Subsystems $ABCD$ and $ABCDE$ are respectively s_rig1 and over-s_rig1 while under-rigid for the same reason. Finally, the system $ABCDEF$ is over-s_rig1 since it contains $ACDE$, $ACDEF$ and $ABCDE$ which are non-trivial over-s_rig1 subsystems.

The degree of rigidity of each subsystem presented in this table has been computed using the following rules:

- A set of points in a geometric system in 3D has $dor = 6$ if not all the points are aligned. Else, it has $dor = 5$ (as long as at least 2 points are non-coincident).

– A set of points and one line in a geometric system in 3D has $dor = 6$ if at least one point is not incident to the line. Otherwise, it has $dor = 5$.

One can check that s_rig2 corresponds to rigidity for all the subsystems presented in table 1. Unfortunately, the gap between structural rigidity and rigidity still remains for redundant systems: the famous counter-example in 3D by Laman [Lam70] still misleads our new structural rigidity.

However, we can prove that s_rig2 is superior to s_rig1: indeed, when $dor \neq \frac{d(d+1)}{2}$, s_rig1 is misled by the dof analysis while s_rig2 is not. When $dor = \frac{d(d+1)}{2}$ the two are equivalent[4].

Thus, every method that is based on previous definitions of structural rigidity should now consider using our new definition of structural rigidity: indeed, in the following section we will show that a polynomial[5] algorithm for s_rig2 can be derived from a polynomial algorithm for s_rig1.

3 Algorithms

In this section, we present algorithms for the main problems associated to the notion of rigidity: deciding if a geometric system is rigid, detecting rigid or over-rigid subsystems, and so on. All are based on our new definition of structural rigidity (definition 5). All use flow machinery on a network representing the geometric system. This network was introduced in [HLS97] for the algorithm `Dense`, an algorithm designed for definition 4 of structural rigidity.

3.1 Object-Constraint Network and function `Distribute`

A geometric system $S = (O, C)$ can be represented by an *object-constraint network* $G = (S, V, T, E, w)$ (an example is provided in figure 2(a)) where:

- S is the source, T is the sink.
- Each object $o \in O$ becomes an object-node $v_o \in V$.
- Each constraint $c \in C$ becomes a constraint-node $v_c \in V$.
- For each object $o \in O$ there is an arc $v_o \to T$ of capacity $w(v_o \to T) = dof(o)$ in E.
- For each constraint $c \in C$, there is an arc $S \to v_c$ of capacity $w(S \to v_c) = dof(c)$ in E.
- For each object $o \in O$ constrained by $c \in C$ there is an arc $v_c \to v_o$ of capacity $w(v_c \to v_o) = \infty$ in E.

A maximum flow in this network represents an optimal distribution of the dof of the geometric constraints among the dof of the geometric objects. If the maximum flow in this network does not saturate all the arcs outgoing the source,

[4] if the heuristic described in section 2.2 is not used; indeed we have already explained that this heuristic can mislead the answer of s_rig1 for systems including trivial over-rigid subsystems (see end of section 2.2).

[5] Under the assumption that determining dor for a given set of objects in a geometric system is feasible in polynomial time. This issue is discussed in section 4.

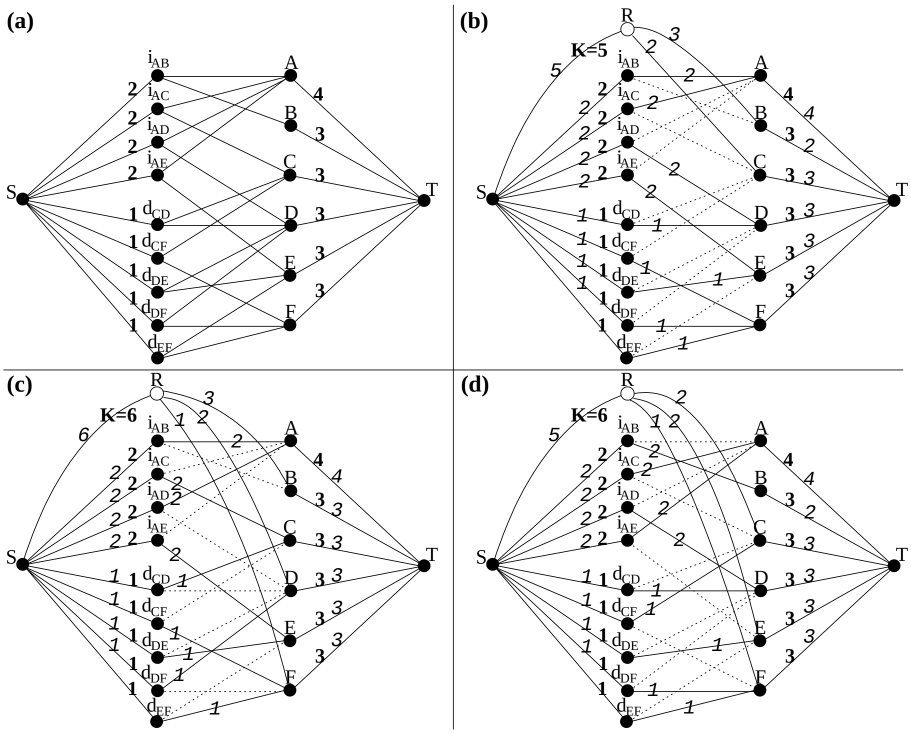

Fig. 2. (a) The network corresponding to the geometric system presented in figure 1 Capacities are represented in bold on arcs. (b) The *modified* network which is built for a call to `Distribute`$(G, 5, \{B, C\})$, (c) `Distribute`$(G, 6, \{B, D, F\})$, (d) `Distribute`$(G, 6, \{C, E, F\})$; maximum flow distributions for these networks are proposed in italic on their arcs.

this means that some constraints dof cannot be removed from the objects dof of the geometric system, i.e. $dof(S) < 0$.

Function `Distribute` generalizes[6] this principle: it identifies subsystems S' verifying $dof(S') < K$ for any positive integer K. To do so, it introduces a fictive constraint R, having $dof(R) = K$, on a subset of objects O' in the geometric system S. If there exists an over-constrained subsystem S' in S after introducing this new constraint R, then this subsystem without the fictive constraint R verifies $dof(S') < K$, i.e. S' has at most $K - 1$ remaining dof.

Function `FordFulkerson`, a standard maximum flow algorithm [FF62], computes a maximum flow distribution in the *overloaded network*, i.e. the object-constraint network corresponding to S with the fictive constraint R. Figure 2(b) depicts this network for a call to `Distribute`$(S, 5, \{B, C\})$. The value returned by function `FordFulkerson` is a subset V of object-nodes in the overloaded

[6] Rigid subsystems have remaining dof, which explains why we want function `Distribute` to be able to identify subsystems in which remains some dof.

Algorithm 1 Distribute (in S: geometric system; in K: integer; in O': set of objects) returns S': geometric system

$G \leftarrow$ `Overloaded-Network(`S, K, O'`)`
$V \leftarrow$ `FordFulkerson(`G`)`
$S' \leftarrow$ `Object-Induced-subsystem(`V, S`)` *{If $V = \emptyset$, $S' = EmptySystem$, a constant representing an empty geometric system.}*
Return S'

network G. V is empty if all the arcs outgoing from the source are saturated, otherwise it is the set of object-nodes that were traversed during the last search for an augmenting path in G. The subsystem $S' \subset S$ induced by the objects in V verifies $dof(S') < K$ (this property was proven in [HLS97] for algorithm `Dense`).

Figure 2(d) presents a maximum flow distribution in the overloaded network for `Distribute(`$S, 6, \{C, E, F\}$`)`. No augmenting path exists in this network. Moreover, arc $S \rightarrow R$ is unsaturated. Since the object-nodes traversed during the last search for an augmenting path are A, C, D, E, F, the identified subsystem is $ACDEF$. Indeed, this subsystem has 5 dof, which is strictly less than the overload parameter $K = 6$.

Function `Distribute` is closely related to a function with the same name in [HLS97]. The main difference is the introduction of the special node R for the purpose of the overload distribution in our function. The similar function by Hoffmann et al. directly applies the overload K by increasing the dof of a constraint c ($dof(C) \leftarrow dof(C)+K$). We modified this function for the following reason: next section will show that, to identify over-s_rig2 subsystems, we will need to call distribute for each *dor-minimal* subset O' of objects in the geometric system S. Since nothing ensures that such a subset is linked to a single constraint in S, we could not use the same principle as Hoffmann et al.

Complexity of `Distribute`: The complexity of function `Distribute` is dominated by that of function `FordFulkerson` and is $O(n^2(n+m))$ where n is the number of nodes and m the number of arcs [FF62].

Remark that if several calls to function `Distribute` are performed for the same system S, this function could be modified to compute maximum flow in an incremental way, yielding a better complexity.

3.2 Algorithm `Over-S_Rigid`

Based on our `Distribute` function, we now propose the algorithm `Over-S_Rigid` for the following problem: *Identify an over-s_ rig2 subsystem in a given geometric system?*

The principle behind this algorithm is the following: one call to function `Distribute(`$S, dor(O', S), O'$`))` for each $O' \subset O$ in $S = (O, C)$ will identify over-s_rig2 subsystems. Indeed, if the call for a given O' returns a non-empty subsystem $S'' = (O'', C'')$, then it verifies $dof(S'') < dor(O', S)$. The following proposition ensures that $dof(S'') < dor(O'', S)$, which is a sufficient condition to be over-s_rig2 (see definition 5).

Algorithm 2 Over-S_Rigid (S: geometric system) returns S':geometric system

$S' \leftarrow EmptySystem$
$M \leftarrow$ `DOR-Minimals(`S`)` *{builds the set M of all dor-minimal subsets of objects in S}*
while $S' = EmptySystem$ and $M \neq \emptyset$ **do**
  $O' \leftarrow$ `FirstElement(`M`)`
  $M \leftarrow M \setminus \{O'\}$
  $S' \leftarrow$ `Distribute(`$S, dor(O', S), O'$`)` *{the overload is the degree of rigidity of the examined subset of objects}*
end while
Return S'

Proposition 1 *Let $S = (O, C)$ be a geometric system in dimension d and $O' \subset O'' \subset O$ two subsets of its objects. Then $dor(O', S) \leq dor(O'', S)$.*

Proposition 1 comes from the following intuitive statement: each dof in a rigid system represents an independent displacement (translation or rotation) in the considered geometric space. Adding a new object o to a rigid geometric system $S = (O, C)$ in a rigid manner (i.e. so that $S' = (O \cup \{o\}, C \cup C')$ remains rigid) cannot remove the independent displacements already granted to S, i.e. $dor(O, S) \leq dor(O \cup \{o\}, S')$.

Algorithm `Over-S_Rigid` applies an overload only for each element in the set M of all *dor-minimal* subsets of objects (computed by `DOR-Minimals(`S`)`):

Definition 6 Dor-minimal subsets of objects
A subset O' of objects in a geometric system $S = (O, C)$ is **dor-minimal** *if it contains no proper subset with the same dor, i.e. $\forall O'' \subsetneq O'$, $dor(O'', S) \neq dor(O', S)$.*

The following proposition ensures that it is sufficient to distribute an overload for each dor-minimal subset of objects:

Proposition 2 *Let O' and $O'' \subset O'$ be two subsets of objects in a geometric system S. If the call to* `Distribute(`S, K, O'`)` *returns a subsystem S', then a call to* `Distribute(`S, K, O''`)` *returns a subsystem $S'' \subseteq S'$.*

This proposition rises from a maximum flow property in overloaded networks: `Distribute(`S, K, O'`)` computes a maximum flow in an overloaded network $G_{O'}$ while `Distribute(`S, K, O''`)` computes it for the overloaded network $G_{O''}$. Since $O'' \subseteq O'$, the only difference between these two networks is that $G_{O'}$ contains more arcs of the type $R \rightarrow v_o$. Hence, it is easier in $G_{O'}$ to distribute the overload than in $G_{O''}$. Thus, if a maximum flow in $G_{O'}$ cannot saturate all the arcs outgoing from the source, a maximum flow in $G_{O''}$ cannot either.

It results that algorithm `Over-S_Rigid` identifies an over-s_rig2 subsystem in a geometric system if there exists one. This algorithm is to over-s_rig2 what algorithm `Dense` is to over-s_rig1.

Example: Let us take back our sample geometric system S presented in figure 1. Let $M = \{\{B,C\},\{B,D,F\},\{C,E,F\},...\}$ be the set of dor-minimal subsets of objects generated by `DOR-Minimals(`S`)`. Algorithm `Over-S_Rigid` then proceeds as follows:

1. First turn, $O' = \{B,C\}$ and $K = dor(\{B,C\},S) = 5$. Figure 2(b) represents the call to `Distribute(`$S, 5, \{B,C\}$`)`. All the arcs outgoing from the source being saturated, no over-s_rig2 subsystem is identified.
2. Second turn, $O' = \{B,D,F\}$ and $K = dor(\{B,D,F\},S) = 6$. The call to `Distribute(`$S, 6, \{B,D,F\}$`)` is represented in figure 2(c). Again, all the arcs outgoing from the source are saturated, and no over-s_rig2 subsystem is identified yet.
3. Third and last turn, $O' = \{C,E,F\}$ and $K = dor(\{C,E,F\},S) = 6$. The call to `Distribute(`$S, 6, \{C,E,F\}$`)` is represented in figure 2(d). This turn, the arc $S \rightarrow R$ is unsaturated. Since the set of object-nodes traversed during the last search for an augmenting path is $\{A,C,D,E,F\}$, the identified subsystem is $ACDEF$ which is over-s_rig2.

Complexity: The complexity of algorithm `Over-S_Rigid` depends on the number of dor-minimal subsets in the geometric system S. We have proven that the size of every dor-minimal subset of objects for geometric systems in 3D in the class ({points, lines, planes},{distances, angles, incidences, parallelisms}) is at most 3. Hence, the number of dor-minimal subsets in a geometric system $S = (O,C)$ from this class is at most $O(N^3)$ if $|O| = N$.

Let us call C_{Dor} the complexity of *dor* computation which is used for all dor-minimal subsets of objects in M, and $C_{Distribute}$ that of function `Distribute`, discussed in the previous section. Then, the worst-case complexity of algorithm `Over-S_Rigid` is $O(N^3(C_{Dor} + C_{Distribute}))$.

We will discuss in section 4 the complexity C_{Dor}. Assuming that it is polynomial and can be neglected in comparison to $C_{Distribute}$, we end up with the same worst-case complexity as algorithm `Dense`[7].

3.3 Other algorithms

In this section, we recall a list of interesting problems related to rigidity in geometric systems and propose algorithms, derived from algorithm `Over-S_Rigid` and function `Distribute`, for these problems.

Decide whether a geometric system is rigid or not. To check the rigidity of a geometric system $S = (O,C)$ via definition 5 of structural rigidity, one has to verify the two conditions of this definition:

1. check that $dof(S) = dor(O,S)$,
2. check that $\forall S' = (O',C') \subset S$, $dof(S') \geq dor(O',S)$, i.e. S contains no over-s_rig2 subsystem.

[7] In fact, algorithm `Dense` has a slightly different complexity for two reasons: the flow is computed incrementally and it is optimized for object-constraint networks including binary constraints only.

Algorithm `S_Rigid?` checks the two previous conditions as follows: it computes $dof(S)$ by simple arithmetic in linear time, and $dor(O, S)$ in $O(C_{dor})$ time to check the first condition. The second condition can be verified by a single call to `Over-S_Rigid(`S`)`. Indeed, if this call returns $EmptySystem$, then S contains no over-s_rig2 subsystem.

Identify a rigid subsystem in a geometric system. To identify a rigid subsystem using s_rig2, one can use a slightly different version of algorithm `Over-S_Rigid` which we call algorithm `OverOrExact-S_Rigid`. The difference between these two algorithms resides in the value of the overload distributed to each dor-minimal subset O': instead of applying $K = dor(O', S)$, an overload $K = dor(O', S) + \mathbf{1}$ is applied for algorithm `OverOrExact-S_Rigid`.

It results from function `Distribute` properties that every subsystem S' returned by algorithm `OverOrExact-S_Rigid` verifies $dof(S') \leq dor(O', S)$, i.e. S' is s_rig2 or over-s_rig2. One call to algorithm `S_Rigid?(`S'`)` decides between both cases. If S' is identified over-s_rig2, then the process is reitered to identify a new candidate.

Minimize an s_rig2 or over-s_rig2 geometric system. A minimal s_rig2 or over-s_rig2 subsystem does not contain any s_rig2 or over-s_rig2 proper subsystem. Minimization is in the heart of the recursive rigidification process for two reasons:

- A minimal over-s_rig2 subsystem is a minimal explanation for the over-rigidity of a geometric system.
- An s_rig2 subsystem will be translated into a subsystem of equations to be solved separately. Hence, identifying minimal s_rig2 subsystems yield smaller subsystems of equations, i.e. a finer decomposition of the geometric system.

Hoffmann et al. have already proposed in [HLS97] a minimization process, called `Minimal-Dense`, for their algorithm `Dense`. We can use a similar minimization process: remove one object o from the subsystem S' identified by algorithm OverOrExact-S_Rigid. After the removal, call algorithm `OverOrExact-S_Rigid` on the remaining subsystem $S'' = S' - o$ to identify a smaller s_rig2 or over-s_rig2 subsystem S'''. If no such subsystem exists, then the removed object was needed: retrieve subsystem S'. Else the newly identified subsystem S''' replaces the old one: $S' \leftarrow S'''$. Trying to remove each object once from S' ensures that the remaining subsystem is minimal.

Remark that algorithm `Dense` by Hoffmann et al does not always require a minimization step: due to the incremental building of the object-constraint network, this algorithm has particular properties which ensure that the identified subsystem is already minimal in specific cases. It has to be investigated whether building the network in the same incremental manner in algorithm `OverOrExact-S_Rigid` allows to obtain the same result.

4 The DOR problem

In this section, we come back to the *DOR* problem, i.e. the problem of computing the degree of rigidity of a set of objects in a geometric system.

The problem of computing $dor(O', S)$ for a given subset O' of objects in S can be reduced to that of computing the *dor* for every dor-minimal subset $O'' \subset O'$ thanks to the following propositions:

Proposition 3 *Let $S = (O, C)$ be a geometric system in dimension d and $O' \subset O$ a subset of its objects. Then $dor(O', S) \leq \frac{d(d+1)}{2}$.*

Proposition 4 *Let $S = (O, C)$ be a geometric system and M be the set of all dor-minimal subsets of objects in S. Then $dor(O, S) = max_{O' \in M} dor(O', S)$.*

Proposition 3 comes from the following statement: each dof in a rigid system represents an independent displacement (translation or rotation) in the considered geometric space. In dimension d, an Euclidean geometric space allows exactly $\frac{d(d+1)}{2}$ independent displacements. The intuition for Proposition 4 is the following: either a geometric system $S = (O, C)$ is induced by a dor-minimal set O of objects, or it contains a dor-minimal subset $O' \subset O$ of objects with $dor(O', S) = dor(O, S)$.

Given a geometric system S and a dor-minimal subset O' of its objects, $dor(O', S)$ can be computed as follows:

1. Generate the set $R_{O'}$ of all the possible *rigidifications* of O'. A **rigidification** of a set O' of geometric objects is a set $C_{O'}$ of geometric constraints such that the geometric system $S' = (O', C_{O'})$ is rigid.
2. Choose the correct rigidification $C_{O'} \in R_{O'}$ according to the complete geometric system S. Then $dor(O', S) = dof(O', C_{O'})$.

First sub-problem: Given a particular set of objects, there may be several possible rigidifications. Moreover, these rigidifications may produce rigid systems with different number of dof.

For instance, a point and a line in 3D can be rigidified by an incidence constraint, or a distance constraint. The resulting rigid systems have respectively 5 and 6 dof. Hence, this subset of objects has potentially 2 different values for its *dor*.

The problem of determining all the possible rigidifications for a dor-minimal set of geometric objects can be addressed by an exhaustive approach. Indeed, for a particular class of geometric systems, one can compute the set of all existing dor-minimal subsets of objects and produce all the possible rigidifications for each of them. We have done so for the class ({points, lines, planes}, {distances, angles, incidences, parallelisms}). In this class, there are 11 dor-minimal subsets of objects, none of which exceeds 3 objects.

Second sub-problem: Choosing the right rigidification for a set of objects according to the complete geometric system which embeds these objects is a much more difficult problems.

For example, for the point and the line in 3D, if the geometric system implies that the point is incident to the line, the correct rigidification is the first one: incidence constraint; thus, the corresponding *dor* is 5. On the other hand, if the geometric system implies that the point is not incident to the line, then the second rigidification is the correct one; in this case, the corresponding *dor* is 6.

More generally, for a subset of objects in a geometric system in the class ({points, lines, planes}, {distances, angles, incidences, parallelisms}), one may have to decide alignments, incidences and parallelisms, which corresponds to geometric theorem proving in the general case. Moreover, these conditions can depend on particular values of distances or angles (non-generic case). It is not affordable for recursive rigidification methods which aim at efficiently solving the geometric system.

Fortunately, there exist classes of geometric systems for which *dor* computation is affordable. For instance, if only the generic case associated to a geometric system is considered (no alignment due to distance values for instance), the problem becomes easier : geometric systems in the class ({points, lines}, {distances, angles}) verify all conditions generically, i.e. points are not generically aligned, lines are not generically parallel and points are not generically incident to lines. This class of geometric systems is interesting for problems in molecular biology for instance.

However, as soon as incidence or parallelism constraints are involved, the problem becomes difficult. For instance, in ({points, lines},{incidences}) systems, Pappus theorem can generate point alignments which are not explicitly stated in the system. Hence, new methods must be investigated to provide good decision procedures for practical problems in these difficult classes.

5 Conclusion

We have provided a thorough analysis of the gap between structural rigidity and rigidity. This analysis has highlighted a difference which yields the introduction of the degree of rigidity concept. We have proposed a new definition for the structural rigidity, which appears to be much more accurate than the previous structural rigidity.

In the second part of this paper, we have presented a flow-based function which is derived from prior work by Hoffmann et al [HLS97]. From this flow based function, we have proposed a set of algorithms for interesting problems concerning rigidity. These algorithms have all a polynomial complexity assuming that dor computation is polynomial. We have highlighted the significance of the *dor* computation problem which remains the hard part in our algorithms. This problem, while easy for some classes of geometric constraints we have presented, can be as difficult as geometric theorem proving in the general case. The challenge will be to discover affordable decision procedures for practical geometric system classes.

References

[BFH+95] William Bouma, Ioannis Fudos, Christoph Hoffmann, Jiazhen Cai, and Robert Paige. Geometric constraint solver. *Computer Aided Design*, 27(6):487–501, 1995.

[DMS97] Jean-François Dufourd, Pascal Mathis, and Pascal Schreck. Formal resolution of geometrical constraint systems by assembling. In C. Hoffmann and W. Bronswort, editors, *Proc. Fourth Symposium on Solid Modeling and Applications*, pages 271–284, 1997.

[FF62] L.R. Ford and D.R. Fulkerson. *Flows in Networks.* Princeton University Press, 1962.

[FH93] Iaonnis Fudos and Christoph Hoffmann. Correctness proof of a geometric constraint solver. Technical Report TR-CSD-93-076, Purdue University, West Lafayette, Indiana, 1993.

[Hen92] Bruce Hendrickson. Conditions for unique realizations. *SIAM j Computing*, 21(1):65–84, 1992.

[HLS97] Christoph Hoffmann, Andrew Lomonossov, and Meera Sitharam. Finding solvable subsets of constraint graphs. In *Proc. Constraint Programming CP'97*, pages 463–477, 1997.

[HLS00] Christoph Hoffmann, Andrew Lomonosov, and Meera Sitharam. Decomposition plans for geometric constraint systems. In *Proc. J. Symbolic Computation 2000*, 2000.

[JASR99] R. Joan-Arinyo and A. Soto-Riera. Combining constructive and equational constraint solving techniques. *ACM Transactions on Graphics*, 18(3):35–55, 1999.

[JTNR00] Christophe Jermann, Gilles Trombettoni, Bertrand Neveu, and Michel Rueher. A constraint programming approach for solving rigid geometric systems. In *Principles and Practice of Constraint Programming, CP 2000*, volume 1894 of *LNCS*, pages 233–248, 2000.

[Kra92] Glenn Kramer. *Solving Geometric Constraint Systems.* MIT Press, 1992.

[Lam70] G. Laman. On graphs and rigidity of plane skeletal structures. *J. Eng. Math.*, 4:331–340, 1970.

[LM97] Hervé Lamure and Dominique Michelucci. Qualitative study of geometric constraints. In Beat Brüderlin and Dieter Roller, editors, *Workshop on Geometric Constraint Solving and Applications*, pages 134–145, Technical University of Ilmenau, Germany, 1997.

[Sit00] Meera Sitharam. Personal communication on the minimal dense algorithm. 2000.

[Whi87] Walter Whiteley. Applications of the geometry of rigid structures. In Henry Crapo, editor, *Computer Aided Geometric Reasoning*, pages 219–254. INRIA, 1987.