Technical Report

Pattern Recognition and Image Processing Group
Institute of Computer Aided Automation
Vienna University of Technology
Favoritenstr. 9/1832
A-1040 Vienna AUSTRIA
Phone: +43 (1) 58801-18351
Fax: +43 (1) 58801-18392
E-mail: jm@prip.tuwien.ac.at
URL: http://www.prip.tuwien.ac.at/

PRIP-TR-76 August 4, 2003

Functional Graphical Models

Jocelyn Marchadier

Abstract

Functional models are frequently used in computer vision and photogrammetry, as they enable the mathematical formulation of several problems such as pose computation and more generally the parameter estimation problem. However, the structural properties of such models have only seldom been studied. This contribution is dedicated to the analysis of such properties. We propose a new formalism that enables the analysis and design of functional models.

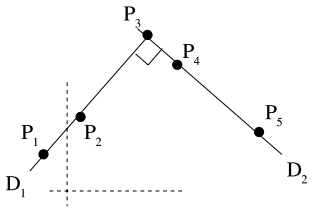


Figure 1: Orthogonal lines

1 Introduction and related work

In this contribution, we use a hypergraph representation of implicit systems for characterizing structural properties of such systems, designing and proving algorithms that decompose the systems into subsystems, and that construct compact codes representing entire systems.

This work is linked with the work of Michelena and Papalambros, for example, who have studied the so called design problem for calculating decomposition of large systems by the means of hypergraphs [9]. The design problem consists of finding a solution such that an equation system is satisfied. They use partitioning techniques for finding subsets of variables achieving a minimum cost decomposition of the system. They are not interested in finding a decomposition into subsystems that can be solved, and do not propose a general framework for analysis of systems. Other related works exist in the computer vision field such as [3][5][8], where functional models are used to code different image primitives, as well as more complicated 3D models. Function-described Graphs [1] are graphs of random variables describing a set of attributed graphs. Only graphs are considered, and arcs labelled with a random variable do not describe functional relations between vertices.

Let us first state informally the definition of functional models. Consider Fig. 1, representing two orthogonal lines L_1 and L_2 crossing at point P_3, such that L_1 is passing through two other points P_1 and P_2 and L_2 is passing through the points P_4 and P_5.

The following implicit system describes completely figure "Orthogonal lines" (Fig. 1).

relations	*equations*
$P_1 \in L_1$	$f(X_1, Y_1, \theta_1, d_1) = 0$
$P_2 \in L_1$	$f(X_2, Y_2, \theta_1, d_1) = 0$
$P_3 \in L_1$	$f(X_3, Y_3, \theta_1, d_1) = 0$
$P_3 \in L_2$	$f(X_3, Y_3, \theta_2, d_2) = 0$
$P_4 \in L_2$	$f(X_4, Y_4, \theta_2, d_2) = 0$
$P_5 \in L_2$	$f(X_5, Y_5, \theta_2, d_2) = 0$
$\angle(L_1, L_2)$	$g(\theta_1, \theta_2) = 0$

(1)

where X_i and Y_i are the coordinates of the point P_i, θ_j and d_j are the parameters of the line L_j, $f(X_i, Y_i, \theta_j, d_j) = X_i \cos(\theta_j) + Y_i \sin(\theta_j) - d_j$ and $g(\theta_i, \theta_j) = \sin(\theta_i - \theta_j)$.

We associate to each relation of the system the set of the variables involved in the relation. We obtain the following tuple : $F_1 = (f, X_1, Y_1, \theta_1, d_1)$, $F_2 = (f, X_2, Y_2, \theta_1, d_1)$, $F_3 = (f, X_3, Y_3, \theta_1, d_1)$, $F_4 = (f, X_3, Y_3, \theta_2, d_2)$, $F_5 = (f, X_4, Y_4, \theta_2, d_2)$, $F_6 = (f, X_5, Y_5, \theta_2, d_2)$,

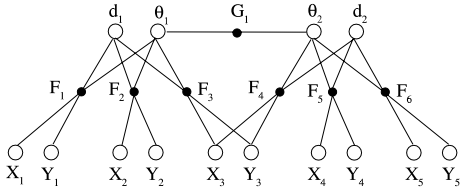


Figure 2: Hypergraph representation of "Orthogonal lines"

and $G_1 = (g, \theta_1, \theta_2)$. We can relate the variables of the implicit system into a graphical form displayed in Fig. 2.

The graph structure depicted as a bi-colored graph on Figure 1 is in fact a hypergraph with two kinds of elements, i.e. the vertex of the hypergraph (the white vertices of the bi-colored graph), which are the variables of the implicit system, and the n-ary relations of the hypergraph, each describing one implicit equation of the system (depicted as black vertices).

Such a structure can be used to derive interesting structural properties of the implicit system. For example, by giving values of some variables of the problem, we derive a simple testing procedure for knowing if the system admits from a purely structural (and not analytical) point of view a unique solution. Another proposed algorithm can quickly decompose a model into models that can be optimized in an independent manner without altering the solution of the initial problem. We also propose an algorithm that decomposes a model with unknown variables into a family of simple models that can be solved or optimized in a certain order.

We derive a simple way to find the minimal code of a given model, which is the sub-model containing the information needed to retrieve all the relations and variables of a complex model.

Section 2 introduces the definition of a new structure, the functional graphical model (FGM), based on hypergraphs that can be used to describe complex concrete problems. Some basic properties of this structure are demonstrated. In section 3, we characterize determined FGMs, i.e. FGMs for which a solution may be calculated from a structural point of view. Section 4 introduces the decomposition of functional graphical models. Section 5 gives perspectives of the presented work.

2 Functional Graphical Model

Functional dependencies of an implicit equation system may be described by a hypergraph. The vertices of the hypergraph represent the variables of the system, and the hyperedges represent the equations of the system. With functional graphical models, we intend to define a formalism which exploits the underlying graph of a functional model.

In this section, we give some definitions and basic results (section 2.1) needed for defining functional graphical models, and to establish some of its properties (section 2.2). We illustrate the definitions on examples (section 2.3).

2.1 Variables and implicit relations

The basic construction blocks of functional graphical models are outlined here. We also demonstrate some basic results.

We define a variable as follows :

Definition 1. *A* (real-valued) variable V *is a symbol representing a point in* $D \subset \mathbb{R}^d$. D *is the* domain *of* V *and represents the admissible values of* V.

For convenience, we use the vectorial notation $V = (C_1 \ ... \ C_d)^t$. $C_1, ..., C_d$ are variables of dimension 1, referred in the following text as *components* of the variable V. The *dimension* of V is the number of components of V, noted $dim(V) = d$.

Definition 2. *An* instance *of a variable* V *is a constant vector* $v \in D$.

A variable describes a set of admissible parameters associated to a primitive without specifying them. A variable may be viewed as a means to identify a specific primitive. In the following text, we do not distinguish between the terms primitive and variable.

A real-valued function f in $\mathbb{R}^m$ of variables $V_1, ..., V_n$, is the relation defined as :

$$\begin{gathered} f : D_1 \times ... \times D_n \to \mathbb{R}^m \\ (v_1, ..., v_n) \mapsto f(v_1, ..., v_n) \end{gathered}$$

where D_i is the domain of V_i and v_i is an instance of V_i. Note that the domain of f is of dimension $\sum_{i=1}^{n} dim(V_i)$.

Definition 3. *A* functional constraint F *is an* n*-ary relation characterized by the* $(n+1)$*-tuple* $F = (f, V_1, ..., V_n)$*, where* f *is a function and* $V_1, ..., V_n$ *are variables. We say that* F *is* satisfied *for* $(v_1, ..., v_n) \in D_1 \times ... \times D_n$*, with* D_i *the domain of the variable* V_i*, iff* $f(v_1, ..., v_n) = 0$. *The* domain set *of the functional constraint* $F = (f, V_1, ..., V_n)$ *is the set* $Dom(F) = \{V_1, ..., V_n\}$.

For convenience, a functional constraint $F = (f, V_1, ..., V_n)$ is denoted $F = [f(V_1, ..., V_n) = 0]$. We will note $dim(F) = d$, the dimension of the range of function f, and we will note abusively $|F| = n$, the cardinal of the domain set of F.

Example : *For example, we define a variable* $P = (x \ y)^t$ *that encodes a point in the plane. Let* x *and* y *be the cartesian coordinates of* P. *Let us define another variable* $L = (\theta \ d)^t$ *encoding a line in the plane, where its components are the angle* θ *between the* x *axis of the coordinate system and the distance* d *of the origin to the line. The relation specifying that a line encoded by* L *passes through a point encoded by* P *is written as :*

$$F_1 = [f(P, L) = 0] = [x \ \cos\theta + y \ \sin\theta - d = 0] \tag{2}$$

Its domain set is then $Dom(F_1) = \{P, L\}$*, and its dimension is* $dim(F_1) = 1$ *as the range of* f *is of dimension 1.*

Definition 4. *The* restriction *of a functional constraint* $F = [f(V_1, ..., V_n) = 0]$ *to a set of variables* $\mathcal{P} \subset Dom(F)$ *is the functional constraint* $(F|\mathcal{P})$ *such that the variables* $V \notin \mathcal{P}$ *are constants.*

$(F|\mathcal{P})$ is a relation of the form $(F|\mathcal{P}) = [g(V'_1, ..., V'_{n'}) = 0]$ such that $Dom(F|\mathcal{P}) = \mathcal{P} = \{V'_1, ..., V'_{n'}\} \subset Dom(F)$.

Example : *For illustration, by considering the relation F_1 of equation 2 above, $(F_1|\{L\})$ would be the relation $[x\ \cos\theta + y\ \sin\theta - d = 0]$ with x and y constants.*

Note that the restriction operation does not change the dimension of the function involved in the equation, i.e. $dim(F|\mathcal{P}) = dim(F)$. It is clear that the restriction operation is commutative, as we have $((F|\mathcal{P})|\mathcal{P}') = (F|\mathcal{P} \cap \mathcal{P}') = ((F|\mathcal{P}')|\mathcal{P})$.

Definition 5. *Let $F = [f(V_1, ..., V_n) = 0]$ be a functional constraint with :*

$$f(V_1, ..., V_n) = \begin{pmatrix} f_1(V_1, ..., V_n) \\ ... \\ f_m(V_1, ..., V_n) \end{pmatrix}$$

where f_i is a function which range is of dimension 1. such that its derivative is also continuous.

Given instances $(v_1, ..., v_n)$ of the variables from $Dom(F)$ for which F is satisfied, a functional constraint F is well-formed *iff each function f_i :*

5.a *is continuous,*

5.b *has a continuous derivative around $(v_1, ..., v_n)$,*

5.c *its partial derivatives according to each component of the variables involved in the relation are not equal to zero around $(v_1, ..., v_n)$.*

F is well-formed if it fulfills the conditions of application of the implicit function theorem (see appendix C) for each of the components of the variables involved. The instances of $V_1, ..., V_n$ for which F is satisfied, describe a C^1 manifold of dimension exactly equal to $\sum_{j=1}^{n} dim(V_j) - dim(F)$ in $\mathbb{R}^{\sum_{j=1}^{n} dim(V_j)}$. For a well formed functional constraint F, its domain set includes variables, the component of which are actually involved in the function.

Example : *The constraint F_1 defined in equation 2 is well-formed. The constraint $F_2 = (f_2, P, L) = [d(x - y) = 0]$ is not well-formed, as θ can take any value without modifying the instances of P for which F_2 holds (and $\frac{\delta(d(x-y))}{\delta\theta} = 0$, contradicting condition 5.c).*

Theorem 1. *The restriction of a well-formed functional constraint is a well-formed functional constraint.*

Proof. The proof of the preceding theorem derives simply from the definitions :

Let $F = [f(V_1, ..., V_n) = 0]$ be a well-formed functional constraint. Each monodimensional function of f is of class C^1 and its partial derivatives are not zero arround one solution. Then, its restriction will have the same properties. □

2.2 Functional Graphical Model : definitions and properties

In this section, we give the basic definitions and results associated to functional graphical models.

Definition 6. *A* functional graphical model *M is a couple $M = (\mathcal{V}, \mathcal{F})$ with :*

6.a *$\mathcal{V}$ a set of primitives $V_i \in \mathcal{V}$,*

6.b *$\mathcal{F}$ a set of functional constraints $F_j \in \mathcal{F}$ of form $F_j = [f_j(V_{j_1}, ..., V_{j_{n_j}}) = 0]$, where $j_1, ..., j_{n_j}$ is a permutation, $V_{j_i} \in \mathcal{V}$ and f_j is a function, the range of which is of dimension d_j.*

The induced hypergraph structure *of a fuctional graphical model $M = (\mathcal{V}, \mathcal{F})$ is the hypergraph $(\mathcal{V}, \{Dom(F_i)|F_i \in \mathcal{F}\})$.*

The definition of an hypergraph can be found in the appendix A. In the rest of this report we will use the acronym FGM standing for functional graphical model. An FGM may be viewed as the explicit graphical description of a system of implicit equation . The main motivations for the graphical description are :

- to analyze the dependencies of the domains of functions involved in such systems, i.e. their structural properties meaningful for estimation, optimization and consistent model construction,
- to construct a symbolic description of the objects that are involved in such models.

The notion of FGM's instance is given below :

Definition 7. *Let $M = (\mathcal{V}, \mathcal{F})$ be an FGM, with $\mathcal{V} = \{V_1, ..., V_{|\mathcal{V}|}\}$,$\mathcal{F} = \{F_1, ..., F_{|\mathcal{F}|}\}$ and $F_j = [f_j(V_{j_1}, ..., V_{j_{n_j}}) = 0]$. An* instance *$(M, \mathcal{V} = v)$ of M is the attributed hypergraph $(v, \mathcal{G}, r)$ with :*

7.a *a set of instances $v = \{v_1, ..., v_{|\mathcal{V}|}\}$ with $v_i \in V_i$,*

7.b *a set of hyperedges $\mathcal{G} = \{G_1, ..., G_{|\mathcal{F}|}\}$ with $G_j = \{v_{j_1}, ..., v_{j_{n_j}}\}$,*

7.c *The function $r(G_j) = -f_j(v_{j_1}, ..., v_{j_{n_j}})$ giving for each hyperedge G_j the* residuals *of its associated constraint.*

In [1], Function-Described graphs represent an ensemble of attributed graphs, while FGMs represent an ensemble of their instances.

A condition that will be used for demonstrating certain properties extends the "well-formmedness" property from functional constraints to FGMs :

Definition 8. *An FGM M is said to be* well-formed *iff all its functional constraints are well-formed.*

The following two definitions restrict the analysis of FMGs to sub-models, for which some variables are relaxed (considered as constant), and to partial models, for which some constraints are relaxed.

An FGM such that the values of certain of its primitives are considered known (data) is sub-model :

Definition 9. *The* sub-model $(M|\mathcal{P})$ *of an FGM* $M = (\mathcal{V}, \mathcal{F})$ *generated by a subset of primitives* $\mathcal{P} \subset \mathcal{V}$ *is an FGM* $M' = (\mathcal{P}, \mathcal{F}_\mathcal{P})$*, excluding constraints not involving primitives of* P *:* $\mathcal{F}_\mathcal{P} = \{(F_i|\mathcal{P})|F_i \in \mathcal{F} \wedge Dom(F_i) \cap \mathcal{P} \neq \emptyset\}$.

If $M' = (\mathcal{V}', \mathcal{F}')$ is a sub-model of an FGM $M = (\mathcal{V}, \mathcal{F})$, then $(\mathcal{V}', \{Dom(F_i)|F_i \in \mathcal{F}'\})$ is a sub-hypergraph of $(\mathcal{V}, \{Dom(F_i)|F_i \in \mathcal{F}\})$ (see appendix A).

Theorem 2. *Let* $M = (\mathcal{V}, \mathcal{F})$ *be an FGM, and let us consider one of its sub-models* $M' = (M|\mathcal{P}')$*. A sub-model* $M'' = (M'|\mathcal{P}'')$ *of* M' *is the sub-model* $(M|\mathcal{P}'')$ *of* M*.*

Proof. Consider the sets $\mathcal{F}_{\mathcal{P}'}$ and $\mathcal{F}_{\mathcal{P}''}$ of relations of M' and M''. We have $\mathcal{F}_{\mathcal{P}'} = \{(F_i|\mathcal{P}')|F_i \in \mathcal{F} \wedge Dom(F_i) \cap \mathcal{P}' \neq \emptyset\}$ and $\mathcal{F}_{\mathcal{P}''} = \{(F_i|\mathcal{P}'')|F_i \in \mathcal{F}_{\mathcal{P}'} \wedge Dom(F_i) \cap \mathcal{P}'' \neq \emptyset\}$. As $\mathcal{P}'' \subset \mathcal{P}'$, we have $Dom(F_i) \cap \mathcal{P}'' \neq \emptyset \Rightarrow Dom(F_i) \cap \mathcal{P}' \neq \emptyset$, and $((F_i|\mathcal{P}')|\mathcal{P}'') = (F_i|\mathcal{P}' \cap \mathcal{P}'') = (F_i|\mathcal{P}'')$. Then, $\mathcal{F}_{\mathcal{P}''} = \{(F_i|\mathcal{P}'')|F_i \in \mathcal{F} \wedge Dom(F_i) \cap \mathcal{P}'' \neq \emptyset\}$, and we can write $M'' = (M|\mathcal{P}'')$. □

We have shown how to define sub-models that are generated by a subsset of primitives. The following definition shows how to define partial models, which are models defined by a set of constraints.

Definition 10. *The* partial FGM *of an FGM* $M = (\mathcal{V}, \mathcal{F})$ *generated by a family* $\mathcal{E} \subset \mathcal{F}$ *is an FGM* $(\mathcal{V}_\mathcal{E}, \mathcal{E})$ *where* $\mathcal{V}_\mathcal{E} = \cup_{F_i \in \mathcal{E}} Dom(F_i)$*, i.e.* $\mathcal{V}_\mathcal{E}$ *is the set composed of all the variables involved in the constraints included in* $\mathcal{E}$*.*

A partial model is thus constituted by a subset of constraints of the original model.

The link with the hypergraph theory is straightforward. If $M' = (\mathcal{V}', \mathcal{F}')$ is a partial FGM of an FGM $M = (\mathcal{V}, \mathcal{F})$, then $(\mathcal{V}', \{Dom(F_i)|F_i \in \mathcal{F}')$ is a partial hypergraph of $(\mathcal{V}, \{Dom(F_i)|F_i \in \mathcal{F}\})$ (see appendix A).

Thus, a well-formed FGM is an FGM involving only well-formed functional constraints. The following two theorems come straightforwardly from the preceding definitions.

Theorem 3. *The sub-model* $(M|\mathcal{P})$ *of a well-formed FGM* M *is a well-formed FGM.*

Proof. The restriction of a well-formed implicit equation is a well-formed implicit equation (theorem 1). Given a well-formed FGM $M = (\mathcal{V}, \mathcal{F})$ and a set of variables $\mathcal{P} \subset \mathcal{V}$, it comes that the set $\mathcal{F}_\mathcal{P} = \{(F_i|\mathcal{P})|F_i \in \mathcal{F} \wedge Dom(F_i) \cap \mathcal{P} \neq \emptyset\}$ is composed of well-formed implicit equations. Thus $(M|\mathcal{P})$ is well-formed. □

Theorem 4. *The partial model of a well-formed FGM* $M = (\mathcal{V}, \mathcal{F})$ *generated by a family* $\mathcal{E} \subset \mathcal{F}$ *is a well-formed FGM.*

Proof. The proof is straightforward. $\mathcal{F}$ being composed of well-formed functional constraints, $\mathcal{E}$ is composed of well-formed functional constraints. □

We will also need in the following text to characterize the union of two FGMs.

Definition 11. *The* union *of two FGMs* $M_1 = (\mathcal{V}_1, \mathcal{F}_1)$ *and* $M_2 = (\mathcal{V}_2, \mathcal{F}_2)$ *is the FGM* $M = (\mathcal{V}_1 \cup \mathcal{V}_2, \mathcal{F}_1 \cup \mathcal{F}_2)$.

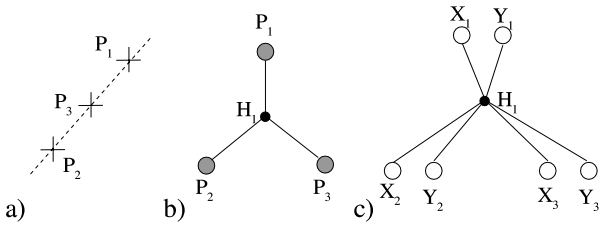


Figure 3: M_1 : three aligned points

Two FGMs $M_1 = (\mathcal{V}_1, \mathcal{F}_1)$ and $M_2 = (\mathcal{V}_2, \mathcal{F}_2)$ are *disjoint* iff they don't have constraints in common : $\mathcal{F}_1 \cap \mathcal{F}_2 = \emptyset$. Two FGMs are *independent* iff they don't have primitives in common : $\mathcal{V}_1 \cap \mathcal{V}_2 = \emptyset$. Remark that two independent FGMs are also disjoint.

For FGMs which are not well-formed, it is sometimes difficult to characterize some of their properties, as their relations are not necessarily well-formed. We introduce the notion of simple FGM, which enables one to proceed to the analysis component by component.

Definition 12. *A* simple *FGM $M = (\mathcal{V}, \mathcal{F})$ is an FGM such that all the variables of $\mathcal{V}$ are scalars ($\forall V_i \in \mathcal{V}, dim(V_i) = 1$) and all the constraints of $\mathcal{F}$ involve functions in $\mathbb{R}$ ($\forall F_j \in \mathcal{F}, dim(F_i) = 1$).*

The simple FGM $M' = (\mathcal{V}', \mathcal{F}')$ derived from an FGM $M = (\mathcal{V}, \mathcal{F})$ is the FGM such that :

- $\mathcal{V}'$ is constructed by concatenation of the components of each vector of $\mathcal{V}$
- $\mathcal{F}'$ is the set of all the equations involved in the implicit system described by M.

2.3 Examples

The preceding definitions are illustrated on the following examples.

Fig. 3 presents three aligned points (Fig. 3.a), the hypergraph structure of the FGM M_1 encoding the alignement constraint (Fig. 3.b) and its simple derived FGM (Fig. 3.c). The primitives of M_1 are the three points $P_1 = (X_1\ Y_1)^t$,$P_2 = (X_2\ Y_2)^t$, and $P_3 = (X_3\ Y_3)^t$. The function associated to the alignment constraint $H_1 = [h(P_1, P_2, P_3) = 0]$ has the form $h(P_1, P_2, P_3) = (X_2 - X_1)(Y_3 - Y_1) - (X_3 - X_1)(Y_2 - Y_1)$. The relation is well-formed.

We can also explicitly represent the parameters of the line L_1. Then, the preceding FGM is turned into the FGM (and its simple derived FGM) M_2, the induced hypergraph structures of which are depicted in Fig. 4 . The points P_1, P_2, and P_3 are vectors of the form $P_i = (X_i\ Y_i)^t$. The primitive representing the line is a vector of the form $L_j = (\theta_j\ d_j)^t$. The constraints F_1, F_2, and F_3 are well-formed functional constraints of the form $F_k = [f(P_i, L_j) = 0] = [X_i \cos(\theta_j) + Y_i \sin(\theta_j) - d_j = 0]$.

The two preceding models can be gathered into a single model M_3, which reinforces the property of alignment (Fig. 5).

We can add a line L_2 orthogonal to L_1 to the preceding model (Fig. 6.a). We obtain the FGM M_4, the induced hypergrah structure of which is drawn in Fig. 7. The orthogonality

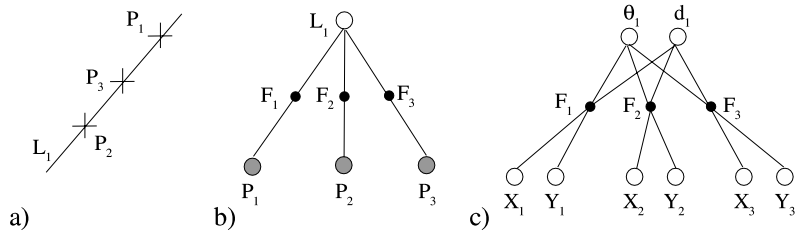


Figure 4: M_2 : three points on a line

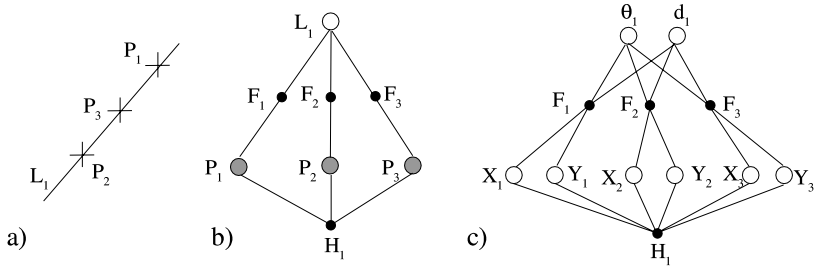


Figure 5: M_3 : three points aligned on a line

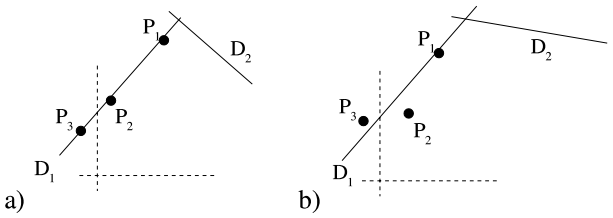


Figure 6: Two instances of the FGM M_4

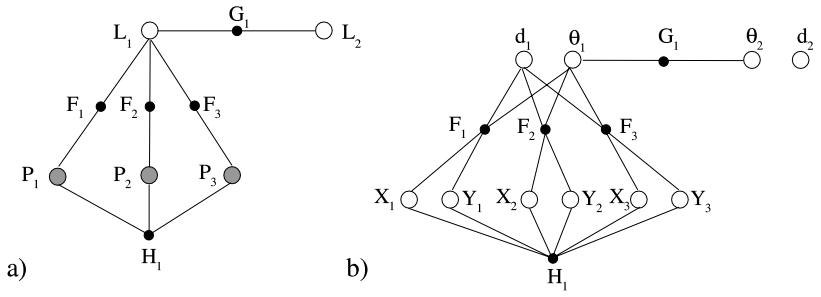


Figure 7: M_4 : three points aligned on a line orthogonal to another line

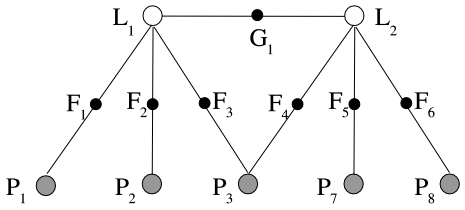


Figure 8: M_5 : 2 orthogonal lines, 5 points and 7 relations

constraint $G_1 = [g(L_i, L_j) = 0] = [cos(\theta_i - \theta_j) = 0]$. As such, the constraint is not well-formed since the parameters d_i and d_j of the lines L_i and L_j can be arbitrarily fixed, giving the same result for the equation G_1. However, the simple FGM drawn on Fig. 7.b is well-formed when omitting the variable d_2 which is unconstrainted. Fig. 6.a illustrate an instance satisfying the constraints of the model, whereas the instance of M_4 drawn on Fig. 6.b does not statisfy the constraints.

The last proposed model is the FGM M_5 drawn in Fig. 8 that represents two orthogonal lines L_1 and L_2, crossing at a point P_3, each line passing threw two other interest points. The simple derived FGM of M_5 is given in Fig. 2. An instance of M_5 is drawn in Fig. 1.

3 Redundancy and determinability

The redundancy r of a functional model is the difference $r = n - m$ between the number of equations n and the number of parameters m of the model. The quality of an estimate calculated thanks to traditional adjustment procedures depends on the redundancy of a functional model [6] :

undetermined If $r < 0$ (i.e. there are more parameters than equations in the functional model), and if there exists a set of parameters satisfying the model, then there exists in general an infinity of sets of parameters satisfying the model. The set of solutions describes generally a manifold, and the functional model is underdetermined.

determined If $r = 0$, then the only analysis that can be achieved is a sensitivity analysis on the parameters resulting from an adjustment. The observational gross errors committed on the data and on the relations of the model can not be put in evidence.

If there is a solution to the functional model, it is generally unique, and the system is said to be uniquely determined.

overdetermined If $r > 0$, there exists in general no solution to the underlying equation system, and a "close" solution can be found by approximation. The functional model is said to be overdetermined.

1. If $r = 1$, it is possible to test the validity of the model with a single gross error, but this error cannot be localized.
2. If $r = 2$, it is possible to localize a single observational gross error. Deficiencies provoked by double gross errors may be detected but the errors can not necessarily be localized [6].
3. If the redundancy $r > 2$, then $r - 1$ gross errors can be localized, and r gross errors are detectable.

If the redundancy of a functional model is not sufficient to guarantee the existence and uniqueness of a solution to the underlying system, it can be used to deduce from a structural point of view the multiplicity of solutions to a system, and to have an indicator on the resistance of a model to gross errors. It is a useful indicator for functional model design.

However, the traditional definition of the redundancy of a functional model is not sufficient for analyzing structural conception problems of the design of functional models. Conception problems may appear by considering subsets of parameters. For example, let's consider the following model (where data are not shown) :

$$\begin{aligned} f_1(V_1, V_2, V_3, V_4) &= 0 \\ f_2(V_3, V_4) &= 0 \\ f_3(V_3, V_4) &= 0 \\ f_4(V_3, V_4) &= 0 \end{aligned} \tag{3}$$

The redundancy of this model is null. However, if f_1, f_2, f_3, and f_4, are smooth functions, and if that system admits a solution (v_1, v_2, v_3, v_4) such that $\frac{\delta f_1}{\delta V_4}(v_1, v_2, v_3, v_4) \neq 0$, then according to the implicit function theorem, the system will have an infinite number of solutions. This design problem can be revealed by noticing that the set of variables $\{V_1, V_2\}$ appears in a single equation. If V_3 and V_4 are known, and if the equations where V_1 and V_2 do not appear are removed from the model, the redundancy of the new model is negative.

The hypergraph structure of an FGM enables the analysis of such discrepancies.

Definition 13. *The* redundancy *of a well-formed FGM $M = (\mathcal{V}, \mathcal{F})$ is defined by :*

$$r(M) = \sum_{F_j \in \mathcal{F}} dim(F_i) - \sum_{V_i \in \mathcal{V}} dim(V_i)$$

The following theorem relates the redundancies of a sub-model $(M|P)$ with those of the FGM M and the complamentary partial model defined with its relations included in $(V \setminus P)$.

Theorem 5. *Let $M = (\mathcal{V}, \mathcal{F})$ be a well-formed FGM, $\mathcal{P}$ be a subset of $\mathcal{V}$, and $M' = (\mathcal{V} \setminus \mathcal{P}, \mathcal{F}')$ be the complementary partial model of M generated by the set of relations $\mathcal{F}' = \mathcal{F} \setminus \mathcal{F}'_{\mathcal{P}}$, with $\mathcal{F}'_{\mathcal{P}}$ the subset of $\mathcal{F}$ involving at least one variable from $\mathcal{P}$ ($\mathcal{F}'_{\mathcal{P}} = \{F_j | F_j \in \mathcal{F} \wedge Dom(F_j) \cap \mathcal{P} \neq \emptyset\}$). Then $r(M) = r(M|\mathcal{P}) + r(M')$.*

Proof. Let $M = (\mathcal{V}, \mathcal{F})$ be a well-formed FGM, and $\mathcal{P}$ and $\mathcal{D}$ be a partition of $\mathcal{V}$, i.e. $\mathcal{V} = \mathcal{P} \cup \mathcal{D}$ and $\mathcal{P} \cap \mathcal{D} = \emptyset$. Let $\mathcal{F}'$ and $\mathcal{F}'_{\mathcal{P}}$ be the two subsets partitionnning $\mathcal{F}$: $\mathcal{F}' = \{F_j, F_j \in \mathcal{F} \wedge Dom(F_j) \subset \mathcal{D}\}$ and $\mathcal{F}'_{\mathcal{P}} = \{F_j, F_j \in \mathcal{F} \wedge Dom(F_j) \cap \mathcal{P} \neq \emptyset\}$. Let us consider the set $\mathcal{F}_{\mathcal{P}} = \{(F_j|\mathcal{P}) | F_j \in \mathcal{F} \wedge Dom(F_j) \cap \mathcal{P} \neq \emptyset\}$. As the restriction operation does not change the dimension of equations, we have $\sum_{F_j \in \mathcal{F}'_{\mathcal{P}}} dim(F_j) = \sum_{F_j \in \mathcal{F}_{\mathcal{P}}} dim(F_j)$. Remember also that a sub-model of a well-formed FGM is a well-formed FGM (c.f. theorem 3), and that a partial model of a well-formed FGM is also a well-formed FGM. Hence :

$$
\begin{aligned}
r(M) - r(M') &= \sum_{F_j \in \mathcal{F}} dim(F_j) - \sum_{V_i \in \mathcal{V}} dim(V_j) - \sum_{F_j \in \mathcal{F}'} dim(F_j) + \sum_{V_i \in \mathcal{D}} dim(V_i) \\
&= \left(\sum_{F_j \in \mathcal{F}} dim(F_j) - \sum_{F_j \in \mathcal{F}'} dim(F_j) \right) - \left(\sum_{V_i \in \mathcal{V}} dim(V_i) - \sum_{V_i \in \mathcal{D}} dim(V_i) \right) \\
&= \sum_{F_j \in \mathcal{F}'_{\mathcal{P}}} dim(F_j) - \sum_{V_i \in \mathcal{P}} dim(V_i) \\
&= \sum_{F_j \in \mathcal{F}_{\mathcal{P}}} dim(F_j) - \sum_{V_i \in \mathcal{P}} dim(V_i) \\
&= r(M|\mathcal{P})
\end{aligned}
$$

□

For simple well-formed FGMs, we have the following obvious result :

Theorem 6. *The redundancy of a simple well-formed FGM $M = (\mathcal{V}, \mathcal{F})$ is $r(M) = |\mathcal{F}| - |\mathcal{V}|$.*

Proof. The proof comes directly from the definitions. □

In order to detect the model defects previously outlined, we want to test if the redundancies of all possible non-empty sub-models are positive. We introduce the following two definitions which generalize this property.

Definition 14. *Let $M = (\mathcal{V}, \mathcal{F})$ be an FGM. The set $\mathcal{P} \subset \mathcal{V}$ of primitives is said to be* determined *by M iff $r(M|\mathcal{P}) \geq 0$ (the sub-model $(M|\mathcal{P})$ has a positive redundancy).*

Definition 15. *An FGM $M = (\mathcal{V}, \mathcal{F})$ is* determinable *iff all the non-empty subsets of $\mathcal{V}$ are determined by M.*

If M is not determinable, then there exist in general an infinity of instances of the variables exactly satisfying the model. If M is composed of linear equations, the preceding statement is turned into an implication.

Example : *To illustrate the preceding definitions, let's consider the model M_5 depicted in Fig. 8. Considering its derived simple FGM (Fig. 2), we can calculate the redundancies of the following sub-models :*

- $(M_5|\{L_1\})$, *determinable* $(r(M_5|\{L_1\}) = 4 - 2 = 2)$,
- $(M_5|\{L_2\})$, *determinable* $(r(M_5|\{L_2\}) = 4 - 2 = 2)$,
- $(M_5|\{L_1, L_2\})$, *determinable* $(r(M_5|\{L_1, L_2\}) = 7 - 4 = 3)$,
- $(M_5|\{P_2\})$, *non determinable* $(r(M_5|\{P_2\}) = 1 - 2 = -1)$,
- $(M_5|\{L_1, L_2.P_2\})$, *non determinable* $((M_5|\{L_1, L_2.P_2\}) = 7 - 6 = 1$ *but* $\{P_2\}$ *is not determined by* M_5 *as* $r(M_5|\{P_2\}) = -1$ *)*,
- $(M_5|\{L_1, L_2.P_3\})$, *determinable,*
- M *is not determinable.*

We now derive some properties characterizing determinable FGM.

Theorem 7. *Let* $M = (\mathcal{V}, \mathcal{F})$ *be a determinable FGM. Then,* $\forall \mathcal{P} \subset \mathcal{V}$*, the sub-model* $(M|\mathcal{P})$ *is determinable.*

Proof. As all subsets of $\mathcal{V}$ are determined by M, all the subsets of $\mathcal{P}$ are determined by M, and thus by $(M|\mathcal{P})$. □

Theorem 8. *Let* $M = (\mathcal{V}, \mathcal{F})$ *be a well-formed model such that* $r(M) \geq 0$*. If for all subset* $\mathcal{V}'$ *of* $\mathcal{V}$*, the partial model generated by* $\{F_i|F_i \in \mathcal{F} \wedge Dom(F_i) \subset \mathcal{V}'\}$ *is not determined, then* M *is determinable.*

Proof. Let $M = (\mathcal{V}, \mathcal{F})$ be a well-formed model such that $r(M) \geq 0$. Suppose that M is not determinable, i.e. there exists a subset $\mathcal{P}$ of $\mathcal{V}$ such that $r(M|\mathcal{P}) < 0$. We have $r(M) = r(M|\mathcal{P}) + r(M')$ (theorem 5), $M' = (\mathcal{V}', \mathcal{F}')$ being the partial model generated by $F' = F \setminus F'_P$, with $\mathcal{V}' = \mathcal{V} \setminus \mathcal{P}$ and $\mathcal{F}'_{\mathcal{P}} = \{F_j|F_j \in \mathcal{F} \wedge Dom(F_j) \cap \mathcal{P} \neq \emptyset\}$. Thus $\mathcal{F}' = \{F_i|F_i \in \mathcal{F} \wedge Dom(F_i) \subset \mathcal{V}'\}$, and $r(M') \geq -r(M|\mathcal{P}) > 0$. Thus M' is determined, and the modus ponens rule leads to the theorem statement. □

Theorem 9. *Let* $M_1 = (\mathcal{V}_1, \mathcal{F}_1)$ *and* $M_2 = (\mathcal{V}_2, \mathcal{F}_2)$ *be two independant well-formed FGMs (* $\mathcal{V}_1 \cap \mathcal{V}_2 = \emptyset$*). The FGM* $M = (\mathcal{V}_1 \cup \mathcal{V}_2, \mathcal{F}_1 \cup \mathcal{F}_2)$ *is determinable iff the models* M_1 *and* M_2 *are determinable.*

Proof. The proof is straightforward. □

4 Decomposition

The adjustment procedures of functional models with numerous parameters and constraints are in practice a heavy burden. When the optimization criterion is the least squares criterion, then each iteration of the adjustment necessitates to compute the solution of a linear system. So each iteration is then done in $O(n^3)$ where n is the number of unknown components. Moreover, when one tries to compute predicted variances of the parameters and the relations, then the entire matrix of size $O(n^2)$ has to be inverted. A solution for accelerating the overall procedures involved in the calculation of estimates

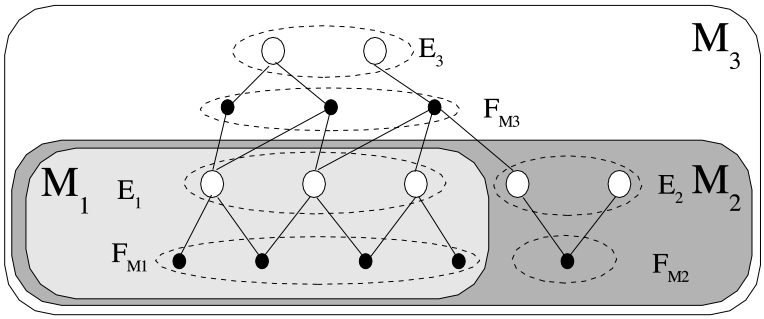


Figure 9: Example of decomposition of a simple FGM

would be to decompose a functional into smaller models which can be solved in a given order. Such a decomposition can be calculated when using an FGM.

The first part of this section is dedicated to the definition of the decomposition problem (section 4.1). In the second part of this section, we introduce a class of algorithms constructing a decomposition in an incremental way (section 4.2). We also prove that one of the algorithms computes, if it exists, a determinable decomposition in polynomial time. This algorithm can be used to test if a given FGM is determinable.

4.1 The decomposition problem

Let's begin by defining what a decomposition of a FGM is.

Definition 16. *A* decomposition *of a well-formed FGM $M = (\mathcal{V}, \mathcal{F})$ is a family of well-formed sub-models $(M_i|\mathcal{E}_i)_{i\geq 1}$ with :*

- *$\mathcal{E}_1, ..., \mathcal{E}_n$ are disjoint sets of primitives of M partitioning $\mathcal{V} = \cup_{k=1}^{n}\mathcal{E}_k$*
- *$M_1, ..., M_n$ are disjoint partial models of M defined as $M_i = (\cup_{j=1}^{i}\mathcal{E}_j, \cup_{j=1}^{i}\mathcal{F}_{M_j})$, where $\mathcal{F}_{M_j}$ is a subset of $\mathcal{F}$ such as the domain set of each constraint $F_k \in \mathcal{F}_{M_j}$ is included in $\cup_{j=1}^{i}\mathcal{E}_j$ and contains variables of $\mathcal{E}_i$: $\mathcal{F}_{M_i} = \{F_j | F_j \in \mathcal{F} \wedge Dom(F_j) \cap \mathcal{E}_i \neq \emptyset \wedge Dom(F_j) \subset \cup_{j=1}^{i}\mathcal{E}_j\}$.*

A decomposition of a FGM can be seen as a nested set of partial models. As the sets of primitives $\mathcal{E}_1, ..., \mathcal{E}_n$ are disjoint, a decomposition forms a hierarchy of models.
Example : *Fig. 9 illustrates the previous definition. On that figure, a simple FGM is depicted. The sets E_1, F_{M_1}, E_2, F_{M_2}, E_3 and F_{M_3} form the decomposition. The FGMs forming the decomposition are $M_1 = (E_1, F_{M_1})$, $M_2 = (E_1 \cup E_2, F_{M_1} \cup F_{M_2})$ and $M_3 = (E_1 \cup E_2 \cup E_3, F_{M_1} \cup F_{M_2} \cup F_{M_3})$.*

Definition 17. *A* determinable decomposition *$(M_i|\mathcal{E}_i)_{i\geq 1}$ of a well-formed FGM M is a decomposition such that each (well-formed) submodel $(M_i|\mathcal{E}_i)$ is determinable.*

The previous definition leads to the following theorem, which relates determinable decompositions to determinable FGMs :

Theorem 10. *If there exists a determinable decomposition $(M_i|\mathcal{E}_i)_{i\geq 1}$ of a FGM M then M is determinable.*

Proof. We consider a decomposition $(M_i|\mathcal{E}_i)_{i\geq 1}$ of an FGM $M = (\mathcal{V}, \mathcal{F})$. We will show that if M is not determinable, then there exists i such that $(M_i|\mathcal{E}_i)$ is not determinable.

If M is not determinable, then there exists, according to the definition, a non empty subset $\mathcal{P}$ of $\mathcal{V}$ such that $\mathcal{P}$ is not determined by M. According to the definition of a decomposition of an FGM, there exists i such that $\mathcal{P}$ is included in the variables of M_i, i.e. $\mathcal{P} \subset \cup_{k=1}^{i} \mathcal{E}_k$. As M_i contains only a subset of the relations of M, $\mathcal{P}$ is not determined by M_i. Let's decompose $\mathcal{P}$ into two disjoint sets $\mathcal{P}' = \mathcal{P} \cap \mathcal{E}_i$ and $\mathcal{P}'' = \mathcal{P} \setminus \mathcal{P}'$. Let us consider that $\mathcal{P}''$ is determined by a model M_j with $j < i$, i.e. $r(M_j|\mathcal{P}'') \geq 0$ (if it is not the case, we can recursively take $\mathcal{P}''$ and M_j in place of $\mathcal{P}$ and M_i).

According to the theorem 5, we calculate the redundancies : $r(M_i|\mathcal{P}) = r((M_i|\mathcal{P})|\mathcal{P}') + r((M_i|\mathcal{P})') < 0$ (1), with $(M_i|\mathcal{P})' = (\mathcal{P}'', \mathcal{F}'')$ being the partial model generated by $\mathcal{F}''$ the set of all the relations of $(M_i|\mathcal{P})$ included in $\mathcal{P}'' = \mathcal{P} \setminus \mathcal{P}'$.

Note that by construction, $\cup_{F_j \cap \mathcal{P} \neq \emptyset} Dom(F_j|\mathcal{P}) \subset \mathcal{P}$. Then, $\cup_{F_j \subset \mathcal{F}_{M_j}} Dom(F_j|\mathcal{P}) \subset \mathcal{P}''$, and consequently $r((M_i|\mathcal{P})') \geq r(M_j|\mathcal{P}'') \geq 0$ (2).

The relations (1) and (2) above lead to $r((M_i|\mathcal{P})|\mathcal{P}') = r(M_i|\mathcal{P}') < 0$, and $\mathcal{P}'$ is not determined by M_i. $(M_i|\mathcal{P}_i)$ is not determinable (as a subset $\mathcal{P}'$ of $\mathcal{P}_i$ is not determined by $(M_i|\mathcal{P})$) and thus $(M_i|\mathcal{E}_i)$ is not determinable. The theorem holds by applying the modus ponens rule. □

4.2 A decomposition algorithm

In this section, we present an algorithm computing a determinable decomposition of an FGM. Section 4.2.1 describes the algorithm. Section 4.2.2 gives a concrete example. Section 4.2.3 details the complexity of the decomposition algorithm.

4.2.1 The algorithm

Decompositions of an FGM can be computed by the algorithm *decomposition*(M, M_1) (procedure 1). It constructs a decomposition of an FGM $M = (\mathcal{V}, \mathcal{F})$ iteratively from a given partial model $M_1 = (\mathcal{E}_1, \mathcal{F}_1)$ generated by $\mathcal{F}_1 \subset \mathcal{F}$.

$C_M(\mathcal{D}, \mathcal{P})$ is the set containing all the relations of M involving variables from the set $\mathcal{D}$ and $\mathcal{P}$.

H is an oriented inclusion graph (c.f. appendix B) that is used in that case for storing functional constraints. To each vertex s of H, we associate a couple $(E[s], F[s])$ where $E[s] \subset \mathcal{V}$ is a subset of variables of M, and $F[s] \subset \mathcal{F}$ is a subset of constraints of M satisfying $\forall F_i \in F[s], Dom(F_i) = E[s]$ (the domain set of each constraint stored in $F[s]$ should be equal to the set $E[s]$ associated with the same vertex). The successors s' of a vertex s in H are associated with the biggest sets $E[s']$ included in $E[s]$. H is used to retrieve in a convenient way sub-models that have required properties, thanks to one of the function *choose_model* described below.

H may be used to retrieved all the different partial models of an FGM. It can be done by considering the coverings of a vertex s of H, which are composed of a subset of the covering of maximum size of s (c.f. appendix B, section B.3).

The function *choose_model*(H) (function 2) is a possible candidate for choosing different models that form a determinable decomposition. It selects thanks to H the sets $\mathcal{E}_i$

Procedure 1 *decomposition*$(M = (\mathcal{V}, \mathcal{F}), M_1 = (\mathcal{E}_1, \mathcal{F}_1))$

1: $\mathcal{P} \leftarrow \mathcal{V} \setminus \mathcal{E}_1$
2: $\mathcal{D} \leftarrow \mathcal{E}_1$
3: $i \leftarrow 2$
4: **while** $\mathcal{E}_{i-1} \neq \emptyset$ **and** $\mathcal{P} \neq \emptyset$ **do**
5: $\quad \mathcal{E} \leftarrow \emptyset$
6: $\quad$ **for all** $F_j \in C_M(\mathcal{D}, \mathcal{P})$ **do**
7: $\quad\quad \mathcal{E} \leftarrow \mathcal{E} \cup (Dom(F_j) \cap \mathcal{P})$
8: $\quad\quad insert(H, (F_j|\mathcal{P}))$
9: $\quad$ **end for**
10: $\quad$ **for all** $F_j, Dom(F_j) \subset \mathcal{E}$ **do**
11: $\quad\quad insert(H, F_j)$
12: $\quad$ **end for**
13: $\quad (\mathcal{E}_i, \mathcal{F}_{M_i}) \leftarrow choose\_model(H)$
14: $\quad \mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{E}_i$
15: $\quad \mathcal{P} \leftarrow \mathcal{V} \setminus \mathcal{D}$
16: $\quad empty(H)$
17: **end while**

and $\mathcal{F}_{M_i}$ needed to define the new model $(M_i|\mathcal{E}_i)$ of a decomposition.

Function 2 *choose_model*(H)

1: $compute\_partial\_models(root[H])$
2: $\mathcal{E} \leftarrow \emptyset$
3: $\mathcal{F} \leftarrow \emptyset$
4: $simplify\_inclusions(root[H], \mathcal{E}, \mathcal{F})$
5: **return** $(\mathcal{E}, \mathcal{F})$

The procedure *compute_partial_models*(s) (procedure 3) replaces the relations $F[s]$ associated to a set $E[s]$ of variables stored in H by the set consisting of all the relations involving the variables $E[s]$ inserted in H. Thus it calculates the partial models $M[s] = (E[s], F[s])$ generated by the set $F[s]$ of relations included in each $E[s]$. This is done by a modified depth first traversal algorithm. The entries of the array $Visited$ must be initialized to $False$ before the first call of the procedure. The function $r[s] = compute\_redundancy(E[s], F[s])$ computes the redundancy of the model $M[s] = (E[s], F[s])$.

The procedure *simplify_inclusions*$(s, \mathcal{E}, \mathcal{F})$ (procedure 4) function erases from H vertices associated to non determined partial models with no determined partial models. The determined models $M[s]$ without determined partial model (which correspond to the vertices s without successor) are added to the constructed model $(\mathcal{E}, \mathcal{F})$. According to the theorem 8, such models are determinable FGMs. According to theorem 9, the constructed model $(\mathcal{E}, \mathcal{F})$ is determinable.

With such a strategy, the procedure *decomposition* constructs a determinable decomposition if it exists (i.e. if the model is determinable) and if all the variables of the model

Procedure 3 *compute_partial_models*(s)

```
1: if Visited[s] = False then
2:    Visited[s] ← True
3:    for all s' ∈ Succ[s] do
4:       compute_partial_model(s')
5:       F[s] ← F[s] ∪ F[s']
6:    end for
7:    r[s] = compute_redundancy(E[s], F[s])
8: end if
```

Procedure 4 *simplify_inclusions*$(s, \mathcal{E}, \mathcal{F})$

```
1: if Visited[s] = False then
2:    Visited[s] ← True
3:    for all s' ∈ Succ[s] do
4:       simplify_inclusions(s', E, F)
5:    end for
6:    if Succ[s] = ∅ then
7:       if r[s] < 0 then
8:          delete(s)
9:       else
10:         E ← E ∪ E[s]
11:         F ← F ∪ F[s]
12:      end if
13:   end if
14: end if
```

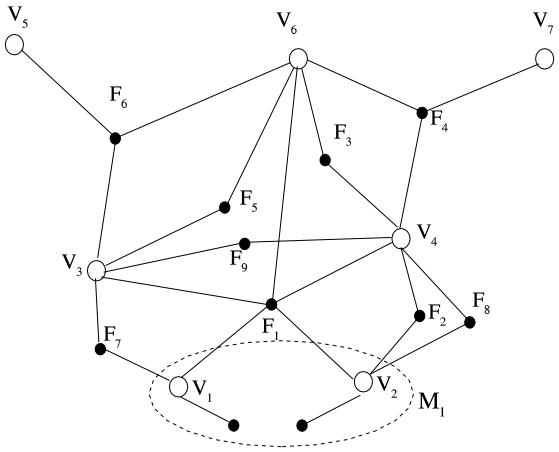


Figure 10: Hypergraph structure of a simple FGM

can be accessed from the variables of M_1. That statement can be proved by remarking that the function *choose_model* constructs the set of determinable partial models containing the smallest set of variables.

If M is not determinable, then the algorithm ends with $\mathcal{P} \neq \emptyset$, i.e. there exists subsets of undetermined variables.

4.2.2 Example

We illustrate the behavior of the algorithm on the simple FGM whose hypergraph structure is depicted in Fig. 10. In this figure, we also present the first model of the decomposition, noted M_1. In this example, the variables V_1 and V_2 of M_1 can be viewed as data of a concrete problem. Thus, the example illustrates how to compute a determinable decomposition either from a determinable partial model of a FGM, or from a subset of its primitives.

The successive iterations of the decomposition algorithm are depicted in Fig. 11. This figure shows the different inclusion graphs constructed during the first two *For* loops of the algorithm.

During the first iteration, the cut between the variables V_1 and V_2 and the other variables of the model consists of the relations F_1, F_2, F_7, and F_8, which are included in the inclusion graph by the first *For* loop (line 6). The second *For* (line 10) loop treats the relation F_9. The non-determined variables that are treated in that iteration are V_3, V_4, and V_6. The models corresponding to the vertices C and D represent determined models, and are selected for constituting the second model of the decomposition (the model (E_2, F_{M_2})).

Two more iterations are computed before the algorithm ends. The final decomposition is depicted in Fig. 12.

Note that the relation F_9 does not appear in the decomposition. Such a relation can be added by using a modified version of the function *choose_model*.

The advantages of computing a decomposition of an FGM can be illustrated on this example. The computation of instances of the variables of the FGM can be done by com-

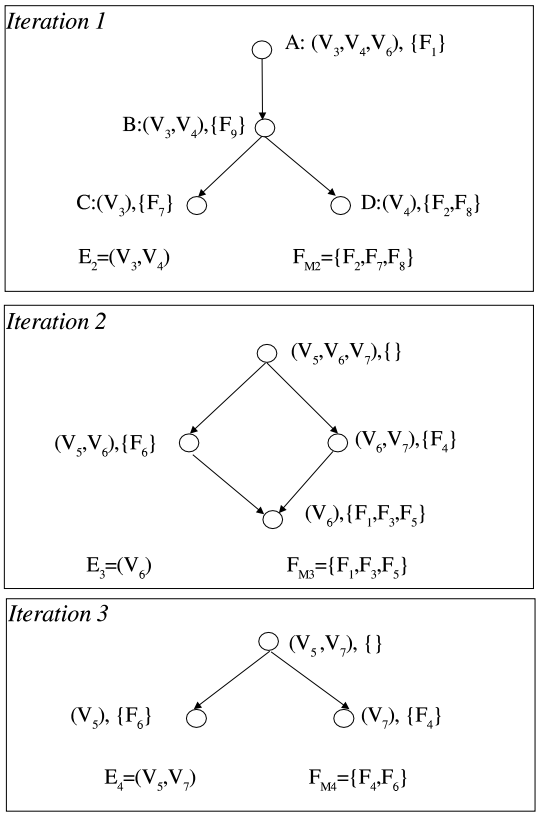


Figure 11: Iterations of the decomposition algorithm

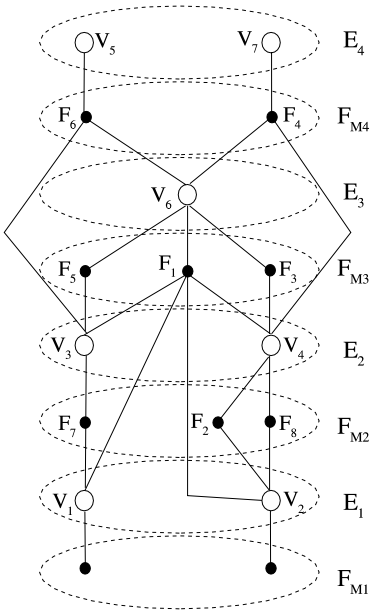


Figure 12: Calculated decomposition of the FGM in Fig. 10

puting successively the instances of the FGMs $(M_1|E_1)$, $(M_2|E_2)$, $(M_3|E_3)$ and $(M_4|E_4)$. Considering that the computation time for computing instances of the variables of a determinable FGM is of order n^3, with n being the number of variables of the FGM, on this concrete example, we compute the approximation in a time order $2^3 + 2^3 + 1 + 2^3 = 25$ per iteration with the decomposition. The resolution of the whole problem by a brute force method would be of order $7^3 = 343$ per iteration without taking into account the complexity of the building of the matrix of the linear system solved by each iteration (performed in $O(nm^2)$, with m the number of equations).

4.2.3 Upper bound of the time complexity for computing a decomposition

In this section, we discuss the time complexity of the algorithm.

The insertion in the inclusion graph is realized in $O(n_s)$, where n_s is the number of vertices of the graph. The cut can contain in the worse case all the relations of the graph. The complexity of the function *choose_model* does not exceed $O(m^2)$ where $m = |\mathcal{F}|$ (n_s vertices explored, for which the redundancy of the associated model has to be calculated).

In each iteration, at least one variable is removed from $\mathcal{P}$ (if no variable is removed, then the function stops). Thus the algorithm is in $O(nm^2)$ in the worse case, with $n = |\mathcal{V}|$. This upper bound is coarse and the true complexity is in fact smaller, using results of [10], in which the inclusion graph has been proven to have a subquadratic complexity. Moreover, the worse case of the algorithm is very unlikely to happen.

Note also that the decomposition can be performed off-line as a pre-treatment when one wants to fit several data sets.

5 Perspectives

We give two perspectives of the present work. In section 5.1, we are interested in coding FGMs with the smallest possible code. In section 5.2, we introduce a stochastic model on an FGM, extending the use of such model to real world applications.

5.1 Codes of FGMs

In this section, we are interested in coding FGMs, and more specifically in the sizes of such codes. Some applications require the derivation of a code of minimum size permitting to compute a given set of primitives of the model. This code can be used for example in applications involving the calculation of the so called MDL criterion [11][7]. This code can thus be used to calculate compressed models, which can then be used to compress data.

Moreover, the reduction operations that are presented here can also be used to reduce the number of parameters of FGMs, which can then be optimized with less operations.

A code of an instance of a simple and well-formed FGM can be decomposed into a list of instances of variables of the model, and a list of constraints involved in the described model. The instance of some variables can be deduced from the instance of other variables and constraints (with associated smooth functions) of the model. This statement is based on an argument that derives from the implicit function theorem (see appendix C).

Let us consider a simple well-formed FGM $M = (V, F)$, and an instance $(M|V = v)$ satisfying each constraint of the model. Each constraint $F_i \in F$ can be put, considering that the hypotheses of the implicit equation theorem hold, under the following form :

$$V_i = g_j(V_{i-j_1}, ..., V_{i-j_{n_j}-1})$$

Note that, according to the implicit function theorem, the expression of g_j depends on the instance of the model.

With these explicit forms, an ordered sequence of variables of M can be constructed. Thus, by giving the instances $v_1, ..., v_k$ of the first k variables $V_1, ..., V_k$ of the sequence, and by using the given explicit forms of the relations of the model, we can compute the instances of the remaining variables $V_{k+1}, ..., V_{|\mathcal{V}|}$ of the sequence.

A valid ordered sequence can be constructed from a determinable decomposition $(M_i|\mathcal{E}_i)_{i\geq 1}$ of M (c.f. section 4), such that :

- $\mathcal{E}_1 = V_1, ..., V_k$,
- $\forall i > 1, \forall F_j \in \mathcal{F}_{M_i}, |Dom(F_i|\mathcal{E}_i)| = 1$. Thus, each relation of a given model $M_i, i > 1$ of the decomposition involves at most one variable of $\mathcal{E}_i$.

A sequence can be constructed by concatenation of the sets $\mathcal{E}_i$. The set $\mathcal{E}_1$ is called in the following text a *generating* set of the simple FGM.

Definition 18. *Let $M = (\mathcal{V}, \mathcal{F})$ be a simple well-formed FGM. The set $\mathcal{G} \subset \mathcal{V}$ is a* generating set *of M iff there exists an order $V_1, ..., V_{|\mathcal{G}|}, V_{|\mathcal{G}|+1}..., V_{|\mathcal{V}|}$ of all the variables of $\mathcal{V}$ such that :*

- $\mathcal{G} = \{V_1, ..., V_{|\mathcal{G}|}\}$,
- $\forall k > |\mathcal{G}|$, $\exists F_j \in \mathcal{F}$ *such that* $V_k \in Dom(F_j)$ *and* $Dom(F_j) \cap \{V_{k+1}, ..., V|\mathcal{V}|\} = \emptyset$.

$\mathcal{G}$ is said to be minimal *if there exists no generating set smaller than $\mathcal{G}$.*

Every subset of variables of M is a generating set, leading to a decomposition that verifies the above properties. The set composed of all the variables of a simple FGM M is a generating set of M, and thus that at least this generating set exists.

Given the instances of all the variables of a generating set of a simple well-formed FGM M, the instances of the other variables can be computed in principle. If both a coder and a decoder of a model know a given simple and well-formed FGM M, then the code of minimal size coding an instance of the model satisfying exactly its constraints is one of its minimal generating set, i.e. the smallest possible generating set of M. This one compresses at most an instance of a model satisfying its constraints.

The problem treated in the following paragraphs is to determine a generating set of a simple and well-composed FGM M. It can be obtained thanks to the following two algorithms that are briefly described.

The first one iteratively removes a variable and relations linked to that variable from M. Ultimately, when no relation remains, we obtain a set of isolated vertices that is a generating set of M. Relations removed in an iteration of the algorithm can be stored, as

they can be used in order to effectively calculate the decomposition. By choosing at each iteration the variable with the minimum number of relations involving different variables, then in simple cases such as tree structures, it is easy to show that the remaining isolated variables constitute a minimal generating set.

The second algorithm works differently. It substitutes variables in the relations by choosing at each iteration a relation giving an explicit form of the variable V_i, and then by replacing the variable by the explicit form in all the relations where V_i is involved. The algorithm is re-iterated until no more substitution can been computed on the FGM, i.e. until the FGM is composed of independent constraints (each variable participating in a unique relation). The ultimate FGM constructed that way can be used to retrieve a minimal generating set of the previous FGM, by choosing a variable that is determined by each of the constructed relations (the remaining variables form the generating set). The main advantage of this algorithm is to compute a model with less variables that can be effectively used instead of the complex model it derives from. Such a model is called a *substituted* model.

Some problems are not treated here. For example, for most applications, we can be interested in encoding a subset of primitives of an FGM that are directly measured (the data) with a partial FGM such that the size of its generating set is minimum. To put it in another way, we want to delete relations and variables of an FGM that are not relevant or are redundant for coding the data.

5.2 Stochastic FGMs

Practical applications need to model the errors of variables and relations, especially when a model is confronted with real world measurements. The uncertainty on variables can be explicitly taken into account in the model by associating probability laws to vertices and hyperedges of the functional model, which leads to the following definition.

Definition 19. *A* stochastic FGM *is a FGM $M = (\mathcal{V}, \mathcal{F})$ where random variables of known distribution are attributed to the elements of $\mathcal{V}$ and $\mathcal{F}$.*

When not otherwise specified, the distribution of the variables and the residuals of a stochastic FGM are considered to be normal distributions $N(\underline{v_i}, \sum_{v_i})$ (for the variables) and $N(0, \sum_{R_j})$ (for the functional constraints' residual), $\underline{v_i}$ being the nominal value of v_i.

We are interested in calculating the propagation of variances on a FGM given an instance $(M|\mathcal{V} = v)$ and either variances of certain primitives of M, or variances of the relations of M.

We can associate a cost function denoted $c(M|V = v)$ with values in $\mathbb{R}^+$ to instances $(M|\mathcal{V} = v)$ of an FGM. According to its definition, this function can be used to compare different instances of one or many FGMs. We consider, in order to simplify the subsequent problems, that this function depends only on the residuals of the relations of the model. Classical examples of cost functions defined on stochastic FGMs are :

- the *goodness-of-fit* of a stochastic FGM $(M|\mathcal{P})$, which measures the a posteriori probability that the model relations are satisfied given instances $(M|\mathcal{V} = v)$ of the model.

- a general *model selection function* realizing a compromise between a model complexity and the adaptation of a parametric model to its data.

We can also study in that context the error propagation problem. This problem can be formulated in two ways.

1. The first formulation is to calculate the error in variables given the error in relation.
2. The second formulation is to calculate all the errors in an instance of an FGM when certain variables have been measured and the other variable instances have been estimated.

6 Conclusion

In this technical report, we introduced the notion of a functional graphical model (FGM). FGMs may be viewed as system of constraints that can be used to model and encode various problems, from geometry, stereovision, to radiometry, etc.

We have characterized some new properties, such as the determinability (definition 14 and 15) of a model, that justify the usefulness of this representation.

We also introduced a class of algorithms that can be used to drastically reduce the computing cost of optimization of variables of functional models. On a simple example in section 4.2.2, a decomposition has been shown to reduce by a factor of 10 the time to optimize parameters' model.

The perspectives of this work are various from the theoretical point of view as well as from the practical one. The coding issues and the extension to stochastic FGMs have been outlined in section 5.2. Further extensions of this work should cover:

1. compression issues,
2. cost for model selection definition,
3. model construction by (combinatorial) optimization,
4. (robust) estimation of parameters and variances,
5. ...

A Hypergraphs

We recall here some definitions of [2].

A *hypergraph* is a couple $M = (\mathcal{V}, \mathcal{F})$, where $\mathcal{V}$ is a set of objects called the vertices of the hypergraph, and $\mathcal{F}$ is a set of subsets of $\mathcal{V}$, called hyperedges of the hypergraph.

A *covering* of the vertices of a hypergraph $M = (\mathcal{V}, \mathcal{F})$ is a family $\mathcal{E} \subset \mathcal{F}$ such that $\cup_{F_i \in \mathcal{E}} F_i = \mathcal{V}$. A *covering* of a subset $\mathcal{V}'$ of vertices of a hypergraph $M = (\mathcal{V}, \mathcal{F})$ is a family $\mathcal{E} \subset \mathcal{F}$ such that $\cup_{F_i \in \mathcal{E}} F_i = \mathcal{V}$.

The *partial hypergraph* of $M = (\mathcal{V}, \mathcal{E})$ generated by a family $\mathcal{E} \subset \mathcal{F}$ is an hypergraph $(\mathcal{V}_\mathcal{E}, \mathcal{F})$ where $\mathcal{V}_\mathcal{E} = \cup_{F_i \in \mathcal{E}} F_i$.

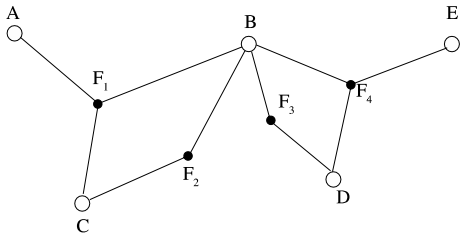


Figure 13: An example of hypergraph

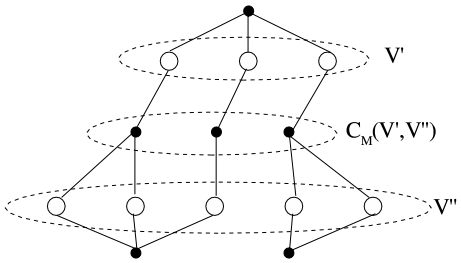


Figure 14: An example of a cut $C_M(V', V'')$ of an hypergraph

The *sub-hypergraph* of $M = (\mathcal{V}, \mathcal{F})$ generated by the set $\mathcal{A} \subset \mathcal{V}$ is the hypergraph $(\mathcal{A}, \mathcal{F}_\mathcal{A})$ where $\mathcal{F}_\mathcal{A} = \{F_i \cap \mathcal{A} | F_i \in \mathcal{F} \wedge F_i \cap \mathcal{A} \neq \emptyset\}$.

Fig. 13 represents the hypergraph $M = (\{A, B, C, D, E\}, \{F_1, F_2, F_3, F_4\})$ with $F_1 = \{A, B, C\}$, $F_2 = \{B, C\}$, $F_3 = \{B, D\}$, and $F_4 = \{B, D, E\}$. The family $\{F_1, F_4\}$ is a covering of the hypergraph. The sub-hypergraph generated by $\{A, B, C\}$ is the hypergraph $(\{A, B, C\}, \{F_1, F_2\})$. The partial hypergraph generated by $\{F_2, F_3\}$ is the hypergraph $(\{B, C, D\}, \{F_2, F_3\})$.

Let $M = (\mathcal{V}, \mathcal{F})$, and let $\mathcal{V}'$ and $\mathcal{V}''$ be two subsets of $\mathcal{V}$ such that $\mathcal{V}' \cup \mathcal{V}'' = \mathcal{V}$. The *cut* $C_M(\mathcal{V}', \mathcal{V}'')$ of M is the set of hyperedges $C_M(\mathcal{V}', \mathcal{V}'') = \{F_j | F_j \in \mathcal{F}, F_j \cap \mathcal{V}' \neq \emptyset \wedge F_j \cap \mathcal{V}'' \neq \emptyset\}$. We can notice that $C_M(\mathcal{V}', \mathcal{V}'') = \mathcal{F}_{\mathcal{V}'} \cap \mathcal{F}_{\mathcal{V}''} = \mathcal{F}_{\mathcal{V}' \cap \mathcal{V}''}$ with $\mathcal{F}_\mathcal{A} = \{F_i | F_i \in \mathcal{F} \wedge F_i \cap \mathcal{A} \neq \emptyset\}$ and $F_\emptyset = \emptyset$. Fig. 14 demonstrates the definition of a cut of a hypergraph. In that figure, $C_M(V', V'')$ is the cut of the subsets V' and V'' of vertices.

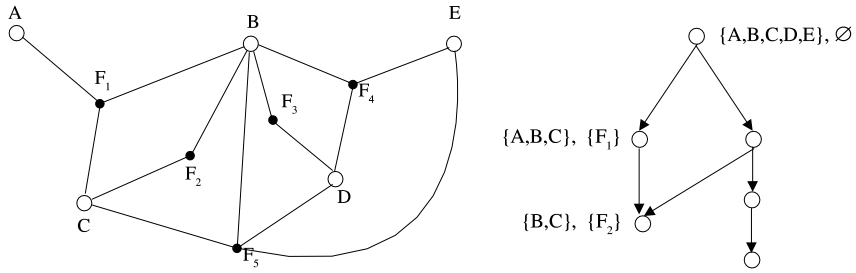
B Oriented Inclusion Graph

The data structure described in this section can be used to store an hypergraph and to efficiently compute numerous operations.

In the first section, we describe the structure (section B.1). In the second section, we propose a simple algorithm for incremental construction of the structure (section B.2). Finally, we show how the structure can be used to enumerate all the coverings of a hypergraph (section B.3).

B.1 Properties of an oriented inclusion graphs

Let's consider a set $\mathcal{V}$ of vertices and a set $\mathcal{F}$ of subsets of $\mathcal{V}$. An oriented inclusion graph $H = (S, A)$ stores the oriented Hasse diagram of the inclusion relation of the sets $F_i \in \mathcal{F}$.



a) An hypergraph b) Its oriented inclusion graph

Figure 15: An hypergraph and its inclusion graph

In [10] it is called the subset graph. Each vertex $s \in S$ of H is a couple $(E[s], F[s])$, where $E[s]$ is a part of $\mathcal{V}$, and $F[s]$ is a subset of $\mathcal{F}$, such that $\forall F_i \in F[s], F_i = E[s]$. Fig. 15.a represents an hypergraphand and Fig. 15.b its associated inclusion graph.

The edges are defined thanks to the predecessors $Pred[s]$ and the successors $Succ[s]$ of a vertex s. The set of predecessors $Pred[s]$ of s is composed by the vertices s' whose associated sets $E[s']$ are the smallest non equal sets containing $E[s]$. The set of successors $Succ[s]$ is composed by the vertices s' whose associated sets $E[s']$ are the biggest non equal sets included in $E[s]$.

Considering an inclusion oriented graph $H = (S, A)$ of a hypergraph $M = (\mathcal{V}, \mathcal{F})$, the following properties must be satisfied :

1. the only vertex $s = root[H]$ without predecessor is defined such that $E[s] = V$,
2. $\forall s \in S, \forall F_i \in \mathcal{F}, F_i = E[s] \Rightarrow F_i \in F[s]$,
3. $\forall s, s' \in S, E[s] = E[s'] \Rightarrow s = s'$ (the vertices represent different sets of V),
4. $\forall s \in S, \forall s' \in Pred[s], E[s] \subset E[s']$ (the predecessors s' of a vertex s are associated to sets $E[s']$ containing $E[s]$),
5. $\forall s, s' \in S, \forall s'' \in Pred[s], E[s''] \subset E[s'] \Rightarrow s' \notin Pred[s]$ ($E[s'']$ is one of the smallest sets containing $E[s]$),

The following two properties show explicitly how to construct edges of H :

- $\forall s, s' \in S, (s, s') \in A \Rightarrow s \in Pred[s']$.
- $\forall s, s' \in S, s \in Pred[s'] \Rightarrow s' \in Succ[s]$.

The following properties hold when all the previous ones hold:

- $\forall s \in S, \forall s' \in Succ[s], E[s'] \subset E[s]$ (the successors s' of a vertex s are associated to sets $E[s']$ contained by $E[s]$), which comes from property 4.
- $\forall s, s' \in S, \forall s'' \in Succ[s], E[s'] \subset E[s''] \Rightarrow s' \notin Succ[s]$ ($E[s'']$ is one of the biggest sets contained by $E[s]$), which comes from property 5.

B.2 Insertion of a set in an inclusion oriented graph

An oriented inclusion graph H of a hypergraph $M = (\mathcal{V}, \mathcal{F})$ can be constructed by an incremental algorithm. The first step consists of initializing the root vertex $root(H)$ of H with the associated sets $E[root(H)] = \mathcal{V}$ and $F[root(H)] = \emptyset$. Then, each hyperedge $F_i \in \mathcal{F}$ can be inserted incrementally in H by the algorithm 5.

Procedure 5 $insert(H, F_i)$

```
 1: EnsPred ← predecessors(root(H), F_i)
 2: if |Pred| = 1 and F[EnsPred[0]] = F_i then
 3:    E[EnsPred[0]] ← E[EnsPred[0]] ∪ {F_i}
 4: else
 5:    s ← NewVertex(F_i)
 6:    Pred[s] ← EnsPred
 7:    Succ[s] ← ∅
 8:    for all s' ∈ Pred[s] do
 9:       for all s'' ∈ Succ[s'] do
10:          if F[s''] ⊂ F_i then
11:             replace − predecessor(s', s, s'')
12:             Succ[s] ← Succ[s] ∪ {s''}
13:          else
14:             Succ[s] ← Succ[s] ∪ successors(s'', F_i)
15:          end if
16:       end for
17:    end for
18: end if
```

The function $predecessors(s, F_i)$ (algorithm 6) returns the smallest non equal sets $E[s']$ containing F_i associated to vertices s' accessible from s. If there exists a vertex s' such that $E[s'] = F_i$ then $predecessors(s, F_i)$ returns the set $\{s\}$.

The boolean array $Visited$ avoids several explorations of a vertex. Every vertex s with associated set containing F_i is visited once, and every vertex of the graph is visited at most once. By considering that the sizes of the sets F_i are bounded by a constant value (which is true for practical applications), the function is linear in the number of vertices and arcs of the graph.

The function $successors(s, F_i)$ (algorithm 7) returns the vertices s' accessible from s such that $E[s']$ are the biggest sets included in F_i. If there exists a vertex s' such that $E[s'] = F_i$ then $successors(s, F_i)$ returns the set $\{s\}$.

B.3 Coverings of an hypergraph

All the coverings of an hypergraph can be easily retrieved thanks to the oriented inclusion graph. Let $M = (\mathcal{V}, \mathcal{F})$ be an hypergraph. The covering $\mathcal{E}$ of a set $\mathcal{V}' \subset \mathcal{V}$ of the vertices of M is of minimum (resp. maximum) size iff, for every covering $\mathcal{E}'$ of $\mathcal{V}'$, $|\mathcal{E}'| \geq |\mathcal{E}|$ (resp. $|\mathcal{E}'| \leq |\mathcal{E}|$). The coverings of minimum size of the sets $F_i \in \mathcal{F}$ are the sets F_i.

Function 6 $predecessors(s, F_i)$

```
1: P ← ∅
2: if Visited[s] = False then
3:    Visited[s] ← True
4:    if F_i ⊂ E[s] then
5:       for all s' ∈ Succ[s] do
6:          if F_i ⊂ E[s'] then
7:             P ← P ∪ predecessors(s', F_i)
8:          end if
9:       end for
10:      if P = ∅ then
11:         P ← {s}
12:      end if
13:   end if
14: end if
15: return P
```

Function 7 $successors(s, F_i)$

```
1: P ← ∅
2: if Visited[s] = False then
3:    Visited[s] ← True
4:    if E[s] ⊂ F_i then
5:       return {s}
6:    else
7:       for all s' ∈ Succ[s] do
8:          if F_i ⊂ E[s'] then
9:             P ← P ∪ successors(s', F_i)
10:         end if
11:      end for
12:   end if
13: end if
14: return P
```

Let H be the inclusion oriented graph of the hyperedges of M. The covering of maximum size of any vertex s is given by a simple traversal of the graph from s (algorithm 8).

Function 8 $maximum - covering(s)$

```
C ← ∅
if Visited[s] = False then
    Visited[s] ← True
    for all s' ∈ Succ[s] do
        C ← C ∪ maximum − covering(s')
    end for
end if
return C
```

The covering $\mathcal{F}'$ of maximum size of a subset $\mathcal{V}'$ of vertices of a hypergraph can be determined very easily. The vertices s of the inclusion graph H of the hypergraph such that $E[s] \subset \mathcal{V}'$ are first determined by the function $successors(root[H], \mathcal{V}')$. Then the union of the maximum covering of all the vertices found by the previous algorithm can be found.

If no covering of V' exists, then $\cup_{F_i \in \mathcal{F}'} F_i \neq \mathcal{V}'$.

C Implicit Function Theorem

We recall here one form of the implicit function theorem that links an implicit function with its explicit form.

Theorem 11. (Implicit Function Theorem)*[4] Let $U \subset \mathbb{R}^{n+m}$ be an open set and f a C^i function defined as :*

$$f : U \to \mathbb{R}^m$$

$$(x, p) \mapsto f(x, p)$$

Suppose that (x_0, p_0) is a point of U such that $f(x_0, p_0) = 0$ and $det(\frac{\delta f}{\delta p}(x_0, p_0)) \neq 0$. Then there exists a neighborhood $U' \subset \mathbb{R}^n$ containing x_0 and a unique C^∞ function $g : \mathbb{R}^n \leftarrow \mathbb{R}^m$ such that the two relations $f(x_0, p_0) = 0$ and $g(p_0) = x_0$ are equivalent.

The definition of a (sub)manifold is

Definition 20. *Let V be a subset of $\mathbb{R}^n$. V is a d-dimensional C_p* submanifold *if, for every $x \in V$, there exists an open neighborhood $U \subset \mathbb{R}^n$ of x and a function $f : U \to \mathbb{R}^n$ such that $f(U) \subset \mathbb{R}^n$ is open, f is a C^p diffeomorphism (f is a bijection, and f and f^{-1} are of class C^p) onto its image and $f(U \cap V) = f(U) \cap \mathbb{R}^n$.*

In particular, if $U \subset \mathbb{R}^{n+m}$ be an open set and f a C^i function defined as $f : U \to \mathbb{R}^m$, then $V = \{u \in U, f(u) = 0\}$ is a n-dimensional C^i manifold.

References

[1] Alquézar R., Serratosa F., Sanfeliu A., "Distance between Attributed Graphs and Function-Described Graphs Relaxing 2nd Order Restrictions", SSPR2000&SPR 2000, FJ. Ferri & al. eds, Lecture Notes in Computer Science vol. 1876, 2000, 277–286

[2] Berge C., "Hypergraphs, Combinatorics of Finitite Sets", North-Holland Mathematical Library, vol. 45, 1989, 255 pages

[3] Baker S., Nayar S. K.,Murase H., "Parametric Feature Detection", International Journal of Computer Vision, vol. 27(1), 1998, 27–50

[4] Faugeras O., "Three-Dimensional Computer Vision, A Geometric Viewpoint", The MIT Press, Cambridge, Massachusetts, 1999, 663 pages

[5] Förstner W., "Reliability Analysis of Parameter Estimation in Linear Models with Application to the Mensuration Problems in Computer Vision", Computer Vision, Graphics and Image Processing, vol. 40, 1987, 273–310

[6] Förstner W., "Generic Estimation Procedures for Orientation with Minimum Redundant Information", 2nd Course of Digital Photogrammetry, 1999

[7] Grünwald P., "Model Selection Based on Minimum Description Length", Journal of Mathematical Psychology vol. 44, 2000, 133–152.

[8] Lowe D. G., "Fitting Parametrized Three-Dimensional Models to Images", IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. 13(5). 1991, 441–450

[9] Michelena N. and Papalambros P., "A Hypergraph Framework for Optimal Model-Based Decomposition of Design Problems", Journal of Computational Optimization and Applications, vol. 8(2), 1997, 173–196.

[10] Pritchard P., "On Computing the Subset Graph of a Collection of Sets", Journal of Algorithms, vol. 33, 1999, 187–203.

[11] Rissanen J., "A Universal Prior For Integers and Estimation by Minimum Description Length", The Annals of Statistics, vol. 11(2), 1983, 416–431.