Convergence of Turbo Algorithms for Systems Defined by Local Constraints

Eric Fabre

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

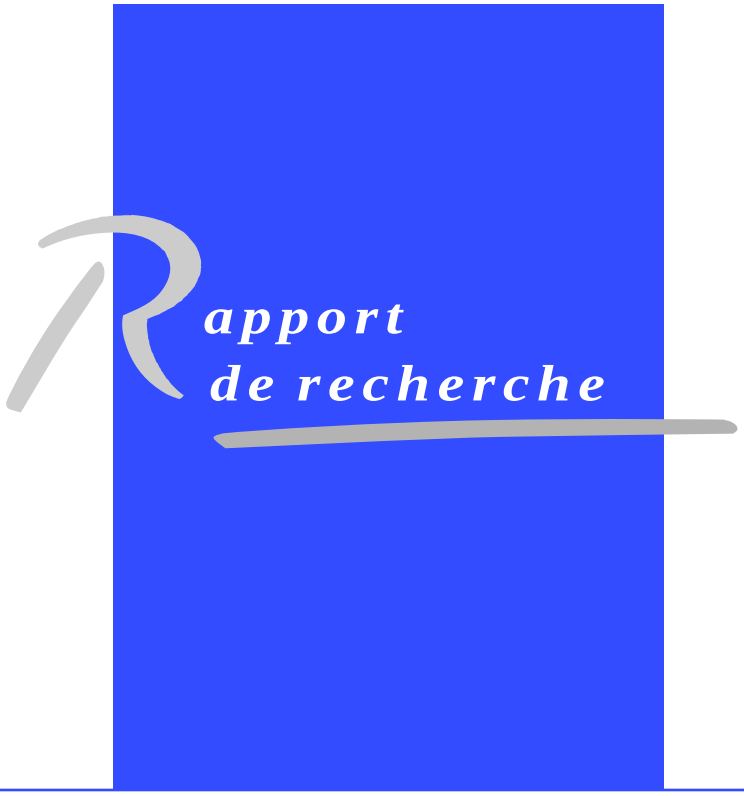
Convergence of Turbo Algorithms for Systems Defined by Local Constraints

Eric Fabre

N°4860

Juin 2003

THÈME 4



ISSN 0249-6399 ISRN INRIA/RR--4860--FR+ENG

INRIA
RENNES

Convergence of Turbo Algorithms for Systems Defined by Local Constraints

Eric Fabre

Thème 4 — Simulation et optimisation
de systèmes complexes
Projet Sigma 2

Rapport de recherche n ° 4860 — Juin 2003 — 39 pages

Abstract: We consider a general notion of system, composed of a set of variables and a set of legal (possibly random) "behaviors" on these variables. Such systems are provided with a composition law, which allows to build large models out of elementary components. This framework captures large constraint systems, Markov random fields (in particular graphical models of powerful error correcting codes), but also some aspects of distributed dynamic systems. Many problems with these models can be expressed under the form of a "reduction" operation: *i.e.* compute the influence of the whole compound system on a given component. For example compute the posterior distribution of a given bit, in a codeword, given all available measurements on that codeword at the output of a channel. The reduction of a system to one or each of its components is generally NP hard, but good approximations can be obtained, which take advantage of the so-called interaction graph of the compound system. We investigate properties of one of them, known as the turbo procedure, initially derived for error correcting codes. In particular, we show that turbo algorithms are possible as soon as some axioms on composition and reduction are satisfied. Convergence cannot be proved in general, except for constraint systems, which can be considered as the simplest family of compound systems. Finally, in our axiomatic framework, we study the structure of approximately reduced components obtained at a stationary point of the turbo procedure, which extends results previously published by Weiss *et al.*

Key-words: turbo algorithm, fix point, graphical model, bayesian network, constraint system, estimation, error correcting code

(Résumé : tsvp)

Unité de recherche INRIA Rennes
IRISA, Campus universitaire de Beaulieu, 35042 RENNES Cedex (France)
Téléphone : 02 99 84 71 00 - International : +33 2 99 84 71 00
Télécopie : 02 99 84 71 71 - International : +33 2 99 84 71 71

Convergence des algorithmes turbo pour les systèmes de contraintes

Résumé : On considère une classe générale de systèmes, définis par un ensemble de variables, et par des "comportements" permis sur ces variables (éventuellement aléatoires). Ces systèmes sont munis d'une loi de composition qui permet d'obtenir de grands modèles à partir de composants élémentaires. Ce cadre permet de modéliser les systèmes de contraintes, les champs de Markov (et en particulier des modèles graphiques de codes correcteurs d'erreurs), de même que certains aspects des systèmes dynamiques distribués. De nombreux traitements associés à ces modèles peuvent formuler sous la forme d'une opération de *réduction* ; cela consiste à calculer l'influence d'un (gros) système composite sur l'un (ou chacun) de ses composants. Par exemple, pour les codes correcteurs d'erreurs, cela revient à calculer la distribution *a posteriori* de chaque bit du mot de code connaissant toutes les mesures reçues sur les bits de ce mot en sortie de canal. Le problème de réduction est en général de complexité NP, mais on peut en donner de bonnes solutions approchées, en utilisant la structure d'interaction entre les composants. Nous nous intéressons à l'une d'entre elles, la procédure "turbo", proposée à l'origine pour les codes correcteurs (les turbo-codes). Nous montrons que ces algorithmes turbo peuvent être mis en oeuvre dès qu'un ensemble d'axiomes sur la composition et la réduction de systèmes sont vérifiés. La convergence n'est pas démontrable en général, mais elle l'est pour les systèmes de contraintes, que l'on peut voir comme la classe la plus simple de systèmes composites. Nous nous intéressons ensuite aux propriétés des points fixes des algorithmes turbo, ce qui permet d'étendre des résultats présentés par Weiss *et al.*

Mots-clé : algorithme turbo, point fixe, modèle graphique, réseau bayésien, système de contraintes, estimation, code correcteur d'erreurs

Contents

1 Introduction

A natural way to design large complex systems is to combine elementary components, by some appropriate composition law. This modular construction principle is widely used in practice to design the systems themselves, but is not always central in the design of subsequent processings (verification, monitoring, etc.). In this paper, we take inspiration from turbo algorithms, a recent breakthrough in digital communications, and propose a general framework to handle modular systems with modular algorithms. This potentially allows to extend some classical processings to large complex systems, for example state estimation for large distributed dynamic systems in the HMM setting (Hidden Markov Models). Hopefully, this will also shed some light on algebraic aspects of turbo procedures.

All along this paper, a system represents the definition of legal behaviors over a set of variables. A central point is the presence of many variables, while the notion of "behavior" can be specified to capture many situations of interest. Let us briefly mention three examples. First of all *constraint systems*, which define the permitted tuples of values $(v_1, \ldots, v_n)$ for variables $V_1, \ldots, V_n$, by means of inequalities for example. The combination of constraint systems is simply done by sharing variables : a large constraint system is thus obtained by gathering local constraints operating on a small number of variables. Secondly, *Markov random fields*, or the related *Bayesian networks*, which can be regarded as "soft" constraint systems [13]. Every component not only defines possible values $(v_1, \ldots, v_n)$ on a subset of variables[1], but also assigns a *weight* to each tuple, which is related to its likelihood. Local weight functions of components can be combined into a probability distribution over all variables appearing in the compound system (details of this construction are given in section 2.2.2). The third example is more on the side of *dynamic systems*. A component can be considered as a (possibly stochastic) automaton specifying local dynamics on a subset of variables. Again, composition can be defined by sharing variables, which means that two components having a common variable can both change the value of this variable by firing one of their transitions. This definition models the exchange of information between components, and allows to design distributed dynamic systems [14]. More interestingly, runs of these dynamic systems can be obtained by composing local runs of their components, as shown in [18].

Large systems are generally intractable as a whole, due to the explosion of the number of "states," where a state is one configuration of the system, for example a tuple of values for all variables in the case of constraint systems. Fortunately, one is generally not interested in studying a large system $\mathcal{S}$ as a whole, but rather in

[1] In many applications, e.g. image processing, all tuples are permitted.

understanding the influence of the complete structure on one of its components $\mathcal{S}_i$, or on a subset of variables. We call this operation the *reduction* of the global system to a set of variables. In the case of constraint systems, for example, the reduction of $\mathcal{S}$ to the variables of component $\mathcal{S}_i$ compiles the constraints carried by all other components about $\mathcal{S}_i$. For Markov fields, reduction can encode the computation of the marginal distribution of a subset of variables. Moreover, if some variables are set to a particular value, reduction provides the posterior distribution of some variables conditionally to these values. Many applications in image processing, in expert systems, as well as the decoding of some error correcting codes can thus be expressed as a reduction problem [11, 12, 4, 9]. Finally, in the case of distributed dynamic systems, one could be interested in recursively estimating the state of each component, given some observable events collected while the system is running. Again, this problem can be stated as a reduction operation [15, 16], and is actually a central motivation of the present paper.

The reduction operation applied to a compound system is NP-hard and thus unaffordable for large systems. Fortunately, good approximate reduction algorithms have been evidenced for turbo-codes, a family of error correcting codes [4, 8, 9]. These "turbo algorithms" do not operate directly on the large system $\mathcal{S}$, but are rather based on local computations at the level of each component. The strength of this approach is precisely to coordinate local computations involving small subsets of variables. Excellent convergence results have been evidenced in practice, and research is still active to understand better these properties. In the field of dynamic systems, turbo algorithms naturally give rise to distributed monitoring algorithms, for example in view of estimating the state of each component. They naturally describe the so-called *orchestration* [20] part of computations, *i.e.* what should be computed locally, what information should be exchanged, and when.

The purpose of this paper is to provide an appropriate abstract framework to describe and study these algorithms, regardless of the application field. We focus in particular on a minimal set of axioms (section 2) on composition and reduction that allow the algebraic derivation of turbo algorithms (section 3). We also study in details properties of stationary points of the algorithm : how far can we trust them ? We provide answers in terms of extendibility and local optimality of behaviors remaining in the approximately reduced components (section 4), which generalize results presented in [1]. Particular attention is paid to constraint systems, which can be considered as the simplest framework, *i.e.* as a benchmark for reduction algorithms. In particular, extra reduction algorithms can be derived for them, and convergence can be proved. Let us stress that the focus is on the algebraic side of the question, which already yields interesting results. More subtle properties of turbo procedures related to analytical aspects are out of the scope of this paper.

2 Modular systems and the reduction problem

Notations. The systems we consider operate on sets of variables. Variables are denoted by capital letters: $A, B, V \ldots$ Respectively, they take values $a, b, v \ldots$ in domains $\mathcal{D}_A, \mathcal{D}_B, \mathcal{D}_V \ldots$ Variable sets are denoted by script letters $\mathcal{V}_1, \mathcal{V}_2, \mathcal{W} \ldots$ We distinguish in particular a set $\mathcal{V}_{max}$ of system variables, and a set $\mathcal{W}$ of auxiliary variables; the latter can be empty. Their respective roles will be specified later. All variables lie in $\mathcal{V}_{max} \cup \mathcal{W}$. Let $\mathcal{V} = \{V_1, \ldots, V_n\}$ be a variable set; a *state* (or *configuration*) $\mathbf{v}$ is a function that assigns to each variable of $\mathcal{V}$ a value of its domain. By abuse of notations, we represent states as tuples $\mathbf{v} = (v_1, \ldots, v_n)$, assuming there exists some natural ordering on variables of $\mathcal{V}$, and we denote by $\mathcal{D}_\mathcal{V} = \mathcal{D}_{V_1} \times \cdots \times \mathcal{D}_{V_n}$ the domain of these states.

2.1 Definitions

2.1.1 Basic axioms

We consider an abstract notion of system, generically denoted by $\mathcal{S}$. To help intuition, systems can be considered as state sets, *i.e.* as (possibly infinite) subsets of $\mathcal{D}_{\mathcal{V}_{max}} \times \mathcal{D}_\mathcal{W}$. Systems are provided with two basic operations: a *composition* law, and a set of *reduction* operators. The composition of systems, denoted by $\mathcal{S} = \mathcal{S}_1 \wedge \mathcal{S}_2$, is commutative and associative. The family of reduction operators $\{\Pi_\mathcal{V}, \mathcal{V} \subseteq \mathcal{V}_{max}\}$ is indexed by sets of variables; reductions operate on a single system: $\mathcal{S}' = \Pi_\mathcal{V}(\mathcal{S})$. Basically, reduction captures the influence of a system on a chosen variable set. We assume composition and reduction satisfy the following axioms.

$$\forall\, \mathcal{V}_1, \mathcal{V}_2 \subseteq \mathcal{V}_{max}, \qquad \Pi_{\mathcal{V}_1} \circ \Pi_{\mathcal{V}_2} \;=\; \Pi_{\mathcal{V}_1 \cap \mathcal{V}_2} \tag{a1}$$

which expresses that reduction operators are actually projections.

$$\forall \mathcal{S},\ \exists \mathcal{V} \subseteq \mathcal{V}_{max} :\ \Pi_\mathcal{V}(\mathcal{S}) \;=\; \mathcal{S} \tag{a2}$$

System $\mathcal{S}$ is said to *operate on variables of* $\mathcal{V}$. Axiom (a1) induces the existence of a smaller set of variables on which $\mathcal{S}$ operates, denoted by $\mathcal{V}_\mathcal{S}$. The central axiom concerns the relation between composition and reduction. Let $\mathcal{S}_1, \mathcal{S}_2$ be two systems operating respectively on $\mathcal{V}_1, \mathcal{V}_2$, then

$$\forall \mathcal{V}_3 \supseteq \mathcal{V}_1 \cap \mathcal{V}_2, \qquad \Pi_{\mathcal{V}_3}(\mathcal{S}_1 \wedge \mathcal{S}_2) \;=\; \Pi_{\mathcal{V}_3}(\mathcal{S}_1) \wedge \Pi_{\mathcal{V}_3}(\mathcal{S}_2) \tag{a3}$$

This is a kind of conditional distributivity property of Π with respect to $\wedge$. It expresses that the interaction between systems $\mathcal{S}_1$ and $\mathcal{S}_2$ is completely captured by their shared variables $\mathcal{V}_1 \cap \mathcal{V}_2$, which thus operate as an interface between the two

systems[2]. Replacing $\mathcal{S}_i$ by $\Pi_{\mathcal{V}_i}(\mathcal{S}_i)$ on the right hand side of (a3), and applying (a1), one gets

$$\forall \mathcal{V}_3 \supseteq \mathcal{V}_1 \cap \mathcal{V}_2, \qquad \Pi_{\mathcal{V}_3}(\mathcal{S}_1 \wedge \mathcal{S}_2) \;=\; \Pi_{\mathcal{V}_3 \cap \mathcal{V}_1}(\mathcal{S}_1) \wedge \Pi_{\mathcal{V}_3 \cap \mathcal{V}_2}(\mathcal{S}_2) \tag{1}$$

Taking $\mathcal{V}_3 = \mathcal{V}_1 \cup \mathcal{V}_2$ in (1) yields

$$\begin{aligned} \Pi_{\mathcal{V}_1 \cup \mathcal{V}_2}(\mathcal{S}_1 \wedge \mathcal{S}_2) &= \Pi_{\mathcal{V}_1}(\mathcal{S}_1) \wedge \Pi_{\mathcal{V}_2}(\mathcal{S}_2) \\ &= \mathcal{S}_1 \wedge \mathcal{S}_2 \end{aligned} \tag{2}$$

which expresses that $\mathcal{S}_1 \wedge \mathcal{S}_2$ operates on variables of $\mathcal{V}_1 \cup \mathcal{V}_2$. As a last axiom in this first group, we assume the existence of an identity element $\mathbb{I}$ for composition :

$$\forall \mathcal{S}, \quad \mathcal{S} \wedge \mathbb{I} \;=\; \mathcal{S} \tag{a4}$$

It is natural to require that $\mathbb{I}$ do not operate on any variable, *i.e.* $\mathcal{V}_{\mathbb{I}} = \emptyset$, or $\Pi_{\emptyset}(\mathbb{I}) = \mathbb{I}$. (A more elegant property would be $\forall \mathcal{S}, \Pi_{\emptyset}(\mathcal{S}) = \mathbb{I}$, but we actually don't need this stronger assumption in the sequel.) By (a1), this induces $\Pi_{\mathcal{V}}(\mathbb{I}) = \mathbb{I}$ for all $\mathcal{V} \subseteq \mathcal{V}_{max}$. As a consequence,

$$\forall \mathcal{S}, \forall \mathcal{V} \subseteq \mathcal{V}_{max}, \quad \mathcal{S} \wedge \Pi_{\mathcal{V}}(\mathbb{I}) \;=\; \mathcal{S} \tag{3}$$

Remark. The form of (a3) clearly reminds a conditional independence statement. This interpretation must be clarified however. $\Pi_{\mathcal{V}_3}(\mathcal{S}_1)$ mustn't be read as the "remaining behavior" of $\mathcal{S}_1$ once valued are fixed for variables of $\mathcal{V}_3$. On the contrary, $\Pi_{\mathcal{V}_3}(\mathcal{S}_1)$ characterizes the "marginal behavior" of variables $\mathcal{V}_3$ in $\mathcal{S}_1$. Nevertheless, (a3) does express that $\mathcal{S}_1$ and $\mathcal{S}_2$ have no interaction outside variables of $\mathcal{V}_3$. As a consequence, given a value $\mathbf{v}_3$ for variables of $\mathcal{V}_3$, the remaining behaviors of $\mathcal{V}_1 \setminus \mathcal{V}_3$ and $\mathcal{V}_2 \setminus \mathcal{V}_3$ in $\mathcal{S}_1$ and $\mathcal{S}_2$ respectively are independent.

2.1.2 Involutivity

We finally introduce a last property which will be satisfied by only part of the systems we consider in the sequel. Composition is said to be *involutive* iff

$$\forall \mathcal{S}, \forall \mathcal{V}, \quad \mathcal{S} \wedge \Pi_{\mathcal{V}}(\mathcal{S}) \;=\; \mathcal{S} \tag{a5}$$

i.e. composing a system with "part of itself" doesn't change that system. We will sometimes say that $\Pi_{\mathcal{V}}(\mathcal{S})$ is *absorbed by* $\mathcal{S}$. Observe that, in an involutive setting, $\Pi_{\mathcal{V}}(\mathbb{I}) = \mathbb{I}$ comes as a consequence of involutivity. Involutivity is a strong property, and will turn out to be a powerful tool in the sequel. Let us mention some of its immediate consequences here.

[2]In the sequel, we will mostly use axiom (a3) with $\mathcal{V}_3 = \mathcal{V}_1 \cap \mathcal{V}_2$, and $\mathcal{V}_i = \mathcal{V}_{\mathcal{S}_i}$. Observe also that, given (a1,a2), it is enough to state (a3) for $\mathcal{V}_1 \cap \mathcal{V}_2 \subseteq \mathcal{V}_3 \subseteq \mathcal{V}_1 \cup \mathcal{V}_2$, it then extends to all $\mathcal{V}_3 \supseteq \mathcal{V}_1 \cap \mathcal{V}_2$.

Lemma 1 *Assuming (a5),*

$$\mathcal{S}_1 \wedge \mathcal{S}_2 = \mathcal{S}_1 \quad \Rightarrow \quad \forall \mathcal{V} \subseteq \mathcal{V}_{max},\ \Pi_{\mathcal{V}}(\mathcal{S}_1) \wedge \Pi_{\mathcal{V}}(\mathcal{S}_2) = \Pi_{\mathcal{V}}(\mathcal{S}_1) \tag{4}$$

Proof. Notice first that $\mathcal{S}_1 \wedge \Pi_{\mathcal{V}}(\mathcal{S}_2) = \mathcal{S}_1 \wedge \mathcal{S}_2 \wedge \Pi_{\mathcal{V}}(\mathcal{S}_2) = \mathcal{S}_1 \wedge \mathcal{S}_2 = \mathcal{S}_1$. Then, since $\Pi_{\mathcal{V}}(\mathcal{S}_2)$ operates on $\mathcal{V}$, (a3) applies and yields $\Pi_{\mathcal{V}}(\mathcal{S}_1) = \Pi_{\mathcal{V}}(\mathcal{S}_1 \wedge \Pi_{\mathcal{V}}(\mathcal{S}_2)) = \Pi_{\mathcal{V}}(\mathcal{S}_1) \wedge \Pi_{\mathcal{V}}(\mathcal{S}_2)$. □

Lemma 2 *Assuming (a5), let $\mathcal{S}_i$ operate on $\mathcal{V}_i$, $1 \leq i \leq N$, then*

$$\mathcal{S} = \mathcal{S}_1 \wedge \ldots \wedge \mathcal{S}_N \quad \Rightarrow \quad \mathcal{S} = \Pi_{\mathcal{V}_1}(\mathcal{S}) \wedge \ldots \wedge \Pi_{\mathcal{V}_N}(\mathcal{S}) \tag{5}$$

In other words, if $\mathcal{S}$ *factorizes*, the *reduced* components $\mathcal{S}'_i \triangleq \Pi_{\mathcal{V}_i}(\mathcal{S})$ give another factorization of $\mathcal{S}$, named the *canonical factorization.*

Proof. It relies on (a3) : $\Pi_{\mathcal{V}_i}(\mathcal{S}) = \mathcal{S}_i \wedge \Pi_{\mathcal{V}_i}(\bigwedge_{j \neq i} \mathcal{S}_j)$. Then

$$\bigwedge_i \Pi_{\mathcal{V}_i}(\mathcal{S}) \quad = \quad (\bigwedge_i \mathcal{S}_i) \wedge [\bigwedge_i \Pi_{\mathcal{V}_i}(\bigwedge_{j \neq i} \mathcal{S}_j)] \tag{6}$$

By involutivity, every term $\Pi_{\mathcal{V}_i}(\bigwedge_{j \neq i} \mathcal{S}_j)$ is absorbed by $\bigwedge_i \mathcal{S}_i$, whence the result. □

2.2 Examples

2.2.1 Constraint systems

In this case, $\mathcal{W}$ is empty. A system is defined through a subset of possible configurations $\mathcal{O} \subseteq \mathcal{D}_{\mathcal{V}_{max}}$, hence $\mathcal{S}$ specifies constraints on the possible values that variables can take. Let $\mathcal{V}' \cup \mathcal{V}''$ be a partition of $\mathcal{V}_{max}$, and $(\mathbf{v}', \mathbf{v}'')$ denote a state of $\mathcal{D}_{\mathcal{V}_{max}} = \mathcal{D}_{\mathcal{V}'} \times \mathcal{D}_{\mathcal{V}''}$. The reduced system $\mathcal{S}' = \Pi_{\mathcal{V}'}(\mathcal{S})$ is determined by $\mathcal{O}'$ defined as

$$\mathcal{O}' \quad = \quad \{(\mathbf{v}', *) : \exists (\mathbf{v}', \mathbf{v}'') \in \mathcal{O}\} \tag{7}$$

where $(\mathbf{v}', *)$ represents all elements obtained by letting the $\mathbf{v}''$ part take any value in $\mathcal{D}_{\mathcal{V}''}$. Observe that $\mathcal{S}' = \Pi_{\mathcal{V}'}(\mathcal{S}')$, which underlines that $\mathcal{S}'$ specifies constraints on variables of $\mathcal{V}'$ only. For simplicity of notations, we identify $(\mathbf{v}', *)$ with $\mathbf{v}'$, and adopt notation $\mathcal{S}' = (\mathcal{V}', \mathcal{O}')$ in the sequel, where $\mathcal{O}' \subseteq \mathcal{D}_{\mathcal{V}'}$ represents a set of (local) states $\mathbf{v}'$.

Composition is defined as the conjunction of constraints defined by two systems. Given systems $\mathcal{S}_i = (\mathcal{V}_i, \mathcal{O}_i)$, $i = 1, 2$, then $\mathcal{S} = \mathcal{S}_1 \wedge \mathcal{S}_2$ is defined by $\mathcal{O} = \mathcal{O}_1 \cap \mathcal{O}_2$ (recall that an element $\mathbf{v}_i$ or $(\mathbf{v}_i, *) \in \mathcal{O}_i$ actually stands for all possible elements $(\mathbf{v}_i, \mathbf{v}'_i)$ where $\mathcal{V}'_i = \mathcal{V}_{max} \setminus \mathcal{V}_i$; intersection is taken in that sense). One easily checks that the states of $\mathcal{S}$ are obtained by merging pairs of states $\mathbf{v}_1, \mathbf{v}_2$ which coincide on shared variables $\mathcal{V}_1 \cap \mathcal{V}_2$.

The unit system $\mathbb{I}$ is defined by $\mathcal{O} = \mathcal{D}_{\mathcal{V}_{max}}$, *i.e.* $\mathbb{I}$ allows all possible states. Obviously, $\forall \mathcal{V} \subseteq \mathcal{V}_{max}$, $\Pi_{\mathcal{V}}(\mathbb{I})$ is defined by $\{(\mathbf{v}, *) : \mathbf{v} \in \mathcal{D}_{\mathcal{V}}\}$, so $\Pi_{\mathcal{V}}(\mathbb{I}) = \mathbb{I}$. $\mathbb{I}$ thus operates on no variable, and can be represented by the single universal state "$*$": $\mathbb{I} = (\emptyset, \{*\})$.

It is straightforward to check that composition and reduction on constraint systems satisfy all the above axioms. In particular, constraint systems are involutive.

2.2.2 Systems with cost functions

The latter are refinements of constraint systems. We now have $\mathcal{W} = \{C\}$ with $\mathcal{D}_C = \mathbb{R}$. A system $\mathcal{S}$ is defined through a set $\mathcal{O} \subseteq \mathcal{D}_{\mathcal{V}_{max}} \times \mathcal{D}_{\mathcal{W}}$, so its elements are pairs $(\mathbf{v}, c)$: $\mathbf{v}$ represents a legal state for the system, and c is its associated weight or cost. As before, given the partition $\mathcal{V}_{max} = \mathcal{V}' \cup \mathcal{V}''$, a (local) state $(\mathbf{v}', *, c)$, or $(\mathbf{v}', c)$ for short, stands for any element $(\mathbf{v}', \mathbf{v}'', c)$, where $\mathbf{v}'' \in \mathcal{D}_{\mathcal{V}''}$.

The reduced system $\Pi_{\mathcal{V}'}(\mathcal{S})$ is defined by the set $\mathcal{O}'$

$$\mathcal{O}' \;=\; \{(\mathbf{v}', c') \,:\, \exists (\mathbf{v}', \mathbf{v}'', c) \in \mathcal{O},\; c' = \min_{\mathbf{v}'', c\,:\,(\mathbf{v}', \mathbf{v}'', c) \in \mathcal{O}} c\} \tag{8}$$

This definition satisfies (a1), but in order to ensure (a2), we limit ourselves to systems satisfying $\mathcal{S} = \Pi_{\mathcal{V}_{max}}(\mathcal{S})$, which means that $\mathcal{S}$ is defined by a set of legal tuples $\mathbf{v} \in \mathcal{D}_{\mathcal{V}_{max}}$ and a cost *function* on these tuples.

Composition $\mathcal{S} = \mathcal{S}_1 \wedge \mathcal{S}_2$, with $\mathcal{S}_i = (\mathcal{V}_i, \mathcal{O}_i)$, is given by $\mathcal{O}$:

$$\mathcal{O} \;=\; \{(\mathbf{v}, c) \,:\, \exists (\mathbf{v}, c_1) \in \mathcal{O}_1,\; \exists (\mathbf{v}, c_2) \in \mathcal{O}_2,\; c = c1 + c2\} \tag{9}$$

where $\mathbf{v}$ denotes an element of $\mathcal{D}_{\mathcal{V}_{max}}$, and again $(\mathbf{v}, c_i) \in \mathcal{O}_i$ means that this element is encompassed by some $(\mathbf{v}_i, *, c_i)$ of $\mathcal{O}_i$. Composition is associative and commutative, and property $\mathcal{S}_i = \Pi_{\mathcal{V}_{max}}(\mathcal{S}_i)$ is transmitted to $\mathcal{S}$. We leave as an exercise the verification of (a3). $\mathbb{I}$ is defined by the set of all possible states $\mathcal{D}_{\mathcal{V}_{max}}$, with a null cost for each of them : $\mathbb{I} = (\emptyset, \{(*, 0)\})$.

Observe that with definitions above, systems are not involutive : combining a system with itself doubles the cost function.

Systems with cost are closely related to Markov random fields. Let us denote by $\mathcal{C}$ the cost function on states $\mathbf{v} \in \mathcal{D}_{\mathcal{V}_{\mathcal{S}}}$ of a system $\mathcal{S}$. Taking $\exp(-\mathcal{C})$, and renormalizing it by its sum over all states $\mathbf{v}$ allowed by $\mathcal{S}$, yields a probability distribution

on variables $\mathcal{V}_\mathcal{S}$. For $\mathcal{S} = \mathcal{S}_1 \wedge \ldots \wedge \mathcal{S}_N$, the cost function $\mathcal{C}_i$ in $\mathcal{S}_i$ represents the so-called *potential* function of clique[3] $\mathcal{V}_i$, while the cost function of the global system $\mathcal{S}$ is referred to as the *energy* function.

We have chosen the pair $(\min, +)$ to define reduction and composition. In the probabilistic interpretation above, reduction can thus be read as a maximum likelihood operation : the cost of a state corresponds to $-\log$ of its probability, so in (8), likelihood is maximized over discarded variables. But other pairs than $(\min, +)$ would work as well, for example $(\max, +)$, or, for positive cost functions, $(\max, *)$ and $(+, *)$.

Systems with positive cost functions correspond to random systems where probability would be handled directly, not in log form. In that case, the reduction defined for the framework $(+, *)$ computes a marginal distribution (the probability function is summed over variables to discard), while $(\max, *)$ computes the maximal likelihood state, or the MAP. In practice, systems with positive cost functions are handled under a renormalized form. This renormalization can be incorporated to the composition and reduction operators without altering their properties.

Note that the variety of choices for defining $\wedge$ and Π in systems with cost functions was already enlighten in [5].

2.3 Graphs associated to a compound system

We now consider compound systems $\mathcal{S} = \mathcal{S}_1 \wedge \ldots \wedge \mathcal{S}_M$, satisfying $\mathcal{V}_i \not\subseteq \mathcal{V}_j, i \neq j$, for $\mathcal{V}_i \triangleq \mathcal{V}_{\mathcal{S}_i}$, $1 \leq i, j \leq M$. The structure of such systems can be displayed by means of a hypergraph $\mathcal{H}$: variables are vertices, and sets $\mathcal{V}_i$ define the edges. Figure 1 gives examples of this representation (left hand side). A more useful notion for us is that of connectivity graph $\mathcal{G}^{cnx}$ for components of $\mathcal{S}$: it has $\{1, \ldots, M\}$ as vertices, or equivalently systems $\mathcal{S}_i$, and (i, j) is an edge iff $\mathcal{V}_i \cap \mathcal{V}_j \neq \emptyset$. A communication graph $\mathcal{G}^c$ for $\mathcal{S}$ is obtained by recursively removing superfluous edges of the connectivity graph, until minimality is reached. An edge (i, j) is said to be superfluous iff there exists a path $(i, k_1, k_2, \cdots, k_L, j)$ such that $\mathcal{V}_i \cap \mathcal{V}_j \subseteq \mathcal{V}_{k_l}$ and $k_l \notin \{i, j\}$ for $1 \leq l \leq L$. In other words, the direct interaction between $\mathcal{S}_i$ and $\mathcal{S}_j$ can be captured by the alternate path $(\mathcal{S}_i, \mathcal{S}_{k_1}, \cdots, \mathcal{S}_{k_L}, \mathcal{S}_j)$. This will take a more precise meaning in section 3 devoted to reduction algorithms. Observe that, in general, a system has several communication graphs, as illustrated by figure 1.

2.4 The reduction problem

Composition is a natural tool to build large complex systems from small simple components. Generally, the large system $\mathcal{S} = \mathcal{S}_1 \wedge \ldots \wedge \mathcal{S}_N$ is intractable. Fortunately,

[3] The term clique refers to a notion of graph for compound systems that we introduce below.

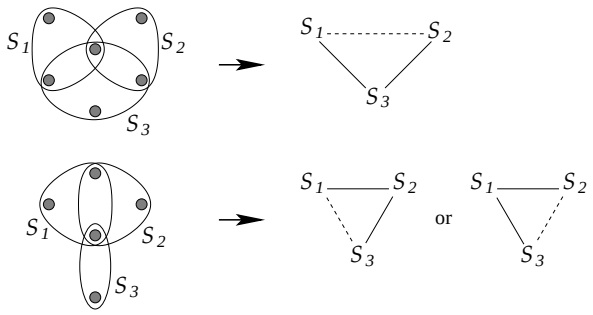


Figure 1: *Left: systems as hypergraphs, where vertices stand for variables, and variable sets $\mathcal{V}_i$ of the different components define (hyper)edges. Right: corresponding communication graphs between components. Dashed edges represent superfluous edges removed from the connectivity graph.*

in many applications, one is rather interested in computing the influence of the large system $\mathcal{S}$ on a given component $\mathcal{S}_i$. Combined to other components, $\mathcal{S}_i$ becomes $\mathcal{S}_i' \triangleq \Pi_{\mathcal{V}_i}(\mathcal{S})$, and computing these $\mathcal{S}_i'$ defines what we call the *reduction problem*. Naturally, one would like to determine or approximate these reduced components without computing $\mathcal{S}$ itself. This is the purpose of the next sections.

Before, we give two application examples of the reduction problem in coding theory. Both concern the decoding of so-called low-density parity check (LDPC) codes. These error correcting codes are constructed in the following way : codewords have a length of N bits, represented as variables $B_1, \ldots, B_N$ taking values 0 or 1. The code is obtained by forbidding some configurations among the 2^N possible ones. Specifically, M (independent) linear constraints $\mathcal{S}_1, \ldots, \mathcal{S}_M$ are applied to these variables. Each $\mathcal{S}_i$ involves a *small* subset of bits $\mathcal{V}_i \subseteq \{B_1, \ldots, B_N\}$ and allows configurations (states) satisfying $\sum_{B_n \in \mathcal{V}_i} B_n = 0$, where addition is modulo 2. This reduces the number of possible configurations to 2^K instead of 2^N, $K = N - M$, which corresponds to a rate $\frac{K}{N}$ code.

Example 1. Let us consider first the decoding problem when an LDPC code is used over an erasure channel. This random channel erases a transmitted bit with probability p, and transmits it without alteration with probability $1 - p$. The decoding problem is to recover the transmitted codeword $b_1 \ldots b_N$ from received values $r_1 \ldots r_N$, where r_n is 0, 1 or x, standing for "erased." This takes the form of a big linear system, one equation per constraint, where B_n is set to r_n if 0 or 1 was received, and left as an unknown otherwise. In our setting, for each observation r_n let us build a system $\mathcal{R}_n$ operating on variable B_n and pinning its value to r_n if 0 or 1 was received, or allowing both values otherwise. The decoding of each bit B_n is given

by $\Pi_{B_n}(\mathcal{S} \wedge \mathcal{R}_1 \wedge \ldots \wedge \mathcal{R}_N)$, which is a sub-product of the $\Pi_{\mathcal{V}_i}(\mathcal{S} \wedge \mathcal{R}_1 \wedge \ldots \wedge \mathcal{R}_N)$. If $r_1 \ldots r_N$ is decodable, this projection assigns a single value to each B_n. Otherwise, some undecodable bits remain, that can still take both values.

Example 2. As a second example, let us consider transmission of an LDPC code over, say, a Gaussian channel, where B_n is modulated as $+1$ or -1 and corrupted by the additive noise Z_n, which yields observation $r_n \in \mathbb{R}$. We now model systems $\mathcal{S}_i$ as systems with a weight function: they allow the same local configurations as above, and assign a null weight to each of them. This stands for the equiprobability of all codewords, weights being homogeneous to a log likelihood. We build observation systems as follows: $\mathcal{R}_n$ operates on B_n and assigns weights $\log \mathbb{P}(r_n|B_n = 0), \log \mathbb{P}(r_n|B_n = 1)$ to values 0 and 1 of B_n. Then in the $(\max, +)$ setting, the reduced system $\Pi_{B_n}(\mathcal{S} \wedge \mathcal{R}_1 \wedge \ldots \wedge \mathcal{R}_N)$ allows values 0 and 1 to B_n with weights

$$\log \max_{b_i,\, 1 \leq i \leq N,\, i \neq n} \mathbb{P}(b_1, \ldots, b_{n-1}, 0/1, b_{n+1}, \ldots, b_N | r_1, \ldots, r_N) + C \tag{10}$$

where C is a constant. Therefore, these values allow a maximum likelihood decoding of bit B_n. This statement may be more convincing without the log, *i.e.* in a $(\max, *)$ setting. Let us assign weight 1 to configurations of $\mathcal{S}_i$, still for equiprobability (any constant value would work as well). Observation systems now assign $\mathbb{P}(r_n|B_n = 0/1)$ to values 0 and 1 of B_n. Then $\Pi_{B_n}(\mathcal{S} \wedge \mathcal{R}_1 \wedge \ldots \wedge \mathcal{R}_N)$ yields weights proportional to

$$\max_{b_i,\, 1 \leq i \leq N,\, i \neq n} \mathbb{P}(b_1, \ldots, b_{n-1}, 0/1, b_{n+1}, \ldots, b_N, r_1, \ldots, r_N) \tag{11}$$

Observe that while the first example is expressed in an involutive setting (a constraint system), this is not the case of the second example.

3 Reduction algorithm for tree shaped systems

It is well known that the reduction problem has a nice and efficient solution for systems having a tree shaped communication graph. Its derivation relies essentially on a separation criterion between components, on the graph of $\mathcal{S}$, which allows the use of (a3). In this section, we recall this algorithm and its properties, using the algebraic setting of section 2. It forms the basis of so-called "turbo algorithms" applied to general systems in section 4.

3.1 Separation property

We first recall definitions and simple results of graph theory concerning a separation criterion between variables and systems. For $\mathcal{S} = \mathcal{S}_1 \wedge \ldots \wedge \mathcal{S}_N$ and $I \subset \{1, \ldots, N\}$, let us denote by $\mathcal{S}_I$ the system $\wedge_{i \in I} \mathcal{S}_i$, and by $\mathcal{V}_I$ the variable set $\cup_{i \in I} \mathcal{V}_i$. Let $\mathcal{H} = (\mathcal{V}, \{\mathcal{V}_1, \ldots, \mathcal{V}_N\})$ be the hypergraph associated to $\mathcal{S} = \mathcal{S}_1 \wedge \ldots \wedge \mathcal{S}_N$.

Definition 1 *Let $\mathcal{X}, \mathcal{Y}, \mathcal{Z} \subseteq \mathcal{V}$ be vertex sets of $\mathcal{H}$. $\mathcal{Y}$ is said to separate $\mathcal{X}$ from $\mathcal{Z}$ on $\mathcal{H}$, denoted by $\mathcal{X}|\mathcal{Y}|\mathcal{Z}$, iff the subgraph $\mathcal{H}_{\mathcal{V} \setminus \mathcal{Y}}$, obtained by removing vertices of $\mathcal{Y}$, is split into several connected components, and none of them contains vertices of both $\mathcal{X}$ and $\mathcal{Z}$.*

$\mathcal{Y}$ is said to be an articulation set of $\mathcal{H}$, separating $\mathcal{X}$ from $\mathcal{Z}$. Definition 1 extends to systems.

Definition 2 *Let $I, J, K \subset \{1, \ldots, N\}$ be index sets, $\mathcal{S}_J$ separates $\mathcal{S}_I$ from $\mathcal{S}_K$ in $\mathcal{S}$ iff $\mathcal{V}_I|\mathcal{V}_J|\mathcal{V}_K$ on $\mathcal{H}$.*

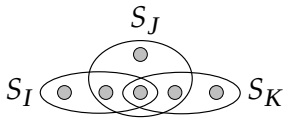


Figure 2: *Component $\mathcal{S}_J$ separates $\mathcal{S}_I$ from $\mathcal{S}_K$.*

The connectivity graph of $\mathcal{S}$ gives a way to read out separation properties between sets of components.

Lemma 3 *Let $\mathcal{G}^{cnx}$ be the connectivity graph of $\mathcal{S}$, and I, J, K be index sets. If $I|J|K$ on $\mathcal{G}^{cnx}$ then $\mathcal{S}_J$ separates $\mathcal{S}_I$ from $\mathcal{S}_K$ in $\mathcal{S}$.*

Proof. This is easily seen on $\bar{\mathcal{H}}$, the bipartite graph associated to $\mathcal{H}$: $\bar{\mathcal{H}}$ has $\mathcal{V} \cup \{1, \ldots, N\}$ as vertices, and there exists an edge between V and i iff variable V belongs to $\mathcal{V}_i$. If $\mathcal{S}_J$ doesn't separate $\mathcal{S}_I$ from $\mathcal{S}_K$, there exists a path $(V_1, c_1, V_2, c_2, \ldots, c_{n-1}, V_n)$ alternating variables and component indexes, such that $V_1 \in \mathcal{V}_I$, $V_n \in \mathcal{V}_K$ and no intermediate variable on the path is in $\mathcal{V}_J$. By removing variables, we get a path $(i, c_1, c_2, \ldots, c_n, k)$ of $\mathcal{G}^{cnx}$ with $i \in I$, $k \in K$, and no intermediate component index is in J, so $I|J|K$ is false on $\mathcal{G}$. □

Communication graphs are more accurate indicators of separation properties.

Lemma 4 *Let $\mathcal{G}^c$ be a communication graph of $\mathcal{S}$, and I, J, K be index sets. If $I|J|K$ on $\mathcal{G}^c$ then $\mathcal{S}_J$ separates $\mathcal{S}_I$ from $\mathcal{S}_K$ in $\mathcal{S}$.*

Proof. We proceed by recursion, removing one redundant edge at a time. Assume the result holds for graph $\mathcal{G}$, and that edge (i,k) is redundant on $\mathcal{G}$. Denote by $\mathcal{G}'$ the graph obtained by removing (i,k). Problems arise for $I|J|K$ holding on $\mathcal{G}'$ but not on $\mathcal{G}$. In that case, any path from I to K and not crossing J on $\mathcal{G}$ must use the edge (i,k). On $\mathcal{G}'$, every path going from i to k must cross J, otherwise $I|J|K$ would not hold on $\mathcal{G}'$. Since (i,k) is redundant on $\mathcal{G}$, there is a path $(i,c_1,\ldots,c_n,k)$ of $\mathcal{G}$ (and thus of $\mathcal{G}'$) such that $\mathcal{V}_i \cap \mathcal{V}_k \subseteq \mathcal{V}_{c_l}, 1 \leq l \leq n$. On that path, one of the c_l is in J, so $\mathcal{V}_i \cap \mathcal{V}_k \subseteq \mathcal{V}_j$ for some $j \in J$. As a consequence, $\mathcal{G}'$ may suggest a smaller set J to separate I from K, but sets $\mathcal{V}_I, \mathcal{V}_J, \mathcal{V}_K$ actually tested for separation on $\bar{\mathcal{H}}$ are identical with $\mathcal{G}$ and $\mathcal{G}'$. In other words, if $\mathcal{V}_I|\mathcal{V}_J|\mathcal{V}_K$ doesn't hold on $\bar{\mathcal{H}}$, this is not due to edge (i,k). □

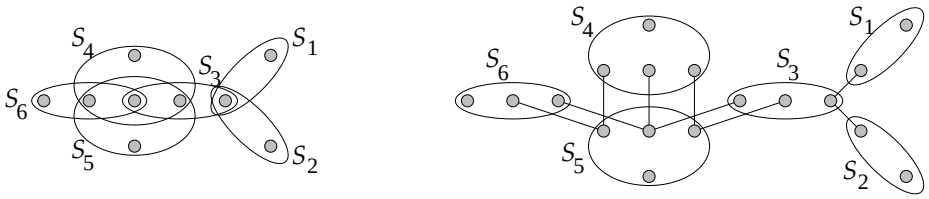


Figure 3: *Left: hypergraph of a system living on a tree. Right: the tree structure is evidenced by duplicating some variables (connected vertices correspond to the same variable), which reveals a possible communication graph.*

Definition 3 *System $\mathcal{S} = \mathcal{S}_1 \wedge \ldots \wedge \mathcal{S}_N$ is said to live on a tree iff it has a communication graph which is a tree.*

In other words, a single component $\mathcal{S}_j$ is enough to separate any two others (provided they are not neighbors). It can be proved that, for such systems, all communication graphs are trees, although they may be different (see theorem 9 in appendix).

3.2 Consequences of separation

The separation criterion between components is crucial to the reduction problem because of the following two consequences. Let I, J, K be pairwise distinct index sets, and assume that $\mathcal{S}_J$ separates $\mathcal{S}_I$ from $\mathcal{S}_K$ in $\mathcal{S}$ (fig. 2), then:

$$\Pi_{\mathcal{V}_J}(\mathcal{S}_{I\cup J\cup K}) \quad = \quad \Pi_{\mathcal{V}_J}(\mathcal{S}_I) \wedge \mathcal{S}_J \wedge \Pi_{\mathcal{V}_J}(\mathcal{S}_K) \tag{12}$$

(12) expresses that if a system $\mathcal{S}_J$ separates two (or more) components, the latter have independent influences on $\mathcal{S}_J$. The proof mostly uses (a3):

$$\Pi_{\mathcal{V}_J}(\mathcal{S}_{I\cup J\cup K}) \quad = \quad \Pi_{\mathcal{V}_J}(\mathcal{S}_I \wedge \mathcal{S}_J \wedge \mathcal{S}_K)$$

$$\begin{aligned} &= \Pi_{\mathcal{V}_J}(\mathcal{S}_J) \wedge \Pi_{\mathcal{V}_J}(\mathcal{S}_I \wedge \mathcal{S}_K) \\ &= \mathcal{S}_J \wedge \Pi_{\mathcal{V}_J}(\mathcal{S}_I) \wedge \Pi_{\mathcal{V}_J}(\mathcal{S}_K) \end{aligned} \tag{13}$$

The second consequence of separation expresses that the influence of $\mathcal{S}_K$ on $\mathcal{S}_I$ can be propagated through the intermediate system $\mathcal{S}_J$:

$$\Pi_{\mathcal{V}_I}(\mathcal{S}_{I\cup J\cup K}) = \mathcal{S}_I \wedge \Pi_{\mathcal{V}_I}[\mathcal{S}_J \wedge \Pi_{\mathcal{V}_J}(\mathcal{S}_K)] \tag{14}$$

The proof uses both (a3) and (a1):

$$\begin{aligned} \Pi_{\mathcal{V}_I}(\mathcal{S}_{I\cup J\cup K}) &= \mathcal{S}_I \wedge \Pi_{\mathcal{V}_I}(\mathcal{S}_J \wedge \mathcal{S}_K) \\ &= \mathcal{S}_I \wedge \Pi_{\mathcal{V}_I}[\Pi_{\mathcal{V}_J\cup\mathcal{V}_K}(\mathcal{S}_J \wedge \mathcal{S}_K)] \\ &= \mathcal{S}_I \wedge \Pi_{\mathcal{V}_I}[\Pi_{\mathcal{V}_J}(\mathcal{S}_J \wedge \mathcal{S}_K)] \\ &= \mathcal{S}_I \wedge \Pi_{\mathcal{V}_I}[\mathcal{S}_J \wedge \Pi_{\mathcal{V}_J}(\mathcal{S}_K)] \end{aligned} \tag{15}$$

The key is that $\Pi_{\mathcal{V}_I} \circ \Pi_{\mathcal{V}_J\cup\mathcal{V}_K} = \Pi_{\mathcal{V}_I} \circ \Pi_{\mathcal{V}_J}$, due to the separation property. Of course, by (a1) and taking into account that $\mathcal{S}_I$ operates on $\mathcal{V}_I$, a term like $\Pi_{\mathcal{V}_J}(\mathcal{S}_I)$ can be replaced by $\Pi_{\mathcal{V}_I\cap\mathcal{V}_J}(\mathcal{S}_I)$.

3.3 Reduction algorithm

Let $\mathcal{S} = \mathcal{S}_1 \wedge \ldots \wedge \mathcal{S}_N$ live on a tree, and $\mathcal{G}^c$ be one of its communication graphs. Let $\mathbf{N}(i)$ denote the neighbors of vertex i on the tree $\mathcal{G}^c$, $1 \leq i \leq N$. The reduction algorithm for $\mathcal{S}$ is based on messages exchanged between neighbors: each system $\mathcal{S}_i$ maintains and updates a message $\mathcal{M}_{i,j}$ for each neighbor $\mathcal{S}_j$, so there are two messages per edge (i,j) of $\mathcal{G}^c$, one in each direction. The idea is that $\mathcal{M}_{i,j}$ collects information about $\mathcal{S}_j$ in systems located on the side of $\mathcal{S}_i$ with respect to edge (i,j). The set of visited systems grows until the whole branch beyond i has been covered.

Algorithm $\mathbf{A}_1$

1. Initialization

$$\mathcal{M}_{i,j} = \Pi_{\mathcal{V}_i\cap\mathcal{V}_j}(\mathbb{I}), \qquad \forall (i,j) \in \mathcal{G}^c \tag{16}$$

2. Until stability of messages, select an edge (i,j) and apply the update rule

$$\mathcal{M}_{i,j} := \Pi_{\mathcal{V}_i\cap\mathcal{V}_j}[\mathcal{S}_i \wedge (\bigwedge_{k\in\mathbf{N}(i)\setminus j} \mathcal{M}_{k,i})] \tag{17}$$

3. Termination

$$\mathcal{S}'_i \;=\; \mathcal{S}_i \wedge (\bigwedge_{k\in\mathbf{N}(i)} \mathcal{M}_{k,i}), \qquad 1\leq i\leq N \tag{18}$$

Although $\Pi_{\mathcal{V}_i\cap\mathcal{V}_j}(\mathbb{I})=\mathbb{I}$, we keep the projector in (16) to stress the fact that message $\mathcal{M}_{i,j}$ only involves shared variables between $\mathcal{S}_i$ and $\mathcal{S}_j$. In practice, and referring to the examples above, $\mathcal{M}_{i,j}$ is represented by a set of allowed states in $\mathcal{D}_{\mathcal{V}_i\cap\mathcal{V}_j}$ (plus a cost in example 2). Message updates in (17) are done in any order: the choice of the oriented edge (i,j) is left unspecified.

Theorem 1 *Let $\mathcal{S}=\mathcal{S}_1\wedge\ldots\wedge\mathcal{S}_N$ live on a tree, and $\mathcal{G}^c$ be a communication graph for $\mathcal{S}$. Then $\mathbf{A}_1$ converges in finitely many steps, and at convergence $\mathcal{S}'_i=\Pi_{\mathcal{V}_i}(\mathcal{S})$, $1\leq i\leq N$.*

"Steps" refer to the number of message updates, where only updates changing the value of a message are counted. This result is well known, but we briefly restate the proof in notations of this paper.

Proof. For an oriented edge (i,j), we denote by $L_{i<j}$ the vertices (or the indexes of systems) separated from j by edge (i,j) on the tree $\mathcal{G}^c$, and thus lying "behind i" from the standpoint of j. We first show that at any time in the evolution of $\mathbf{A}_1$, messages satisfy

$$\exists L_{i,j}\subseteq L_{i<j}\;:\qquad \mathcal{M}_{i,j} \;=\; \Pi_{\mathcal{V}_i\cap\mathcal{V}_j}(\mathcal{S}_{L_{i,j}}) \tag{19}$$

Observe that the influence of $\mathcal{S}_{L_{i,j}}$ on $\mathcal{S}_j$ is summarized into $\Pi_{\mathcal{V}_j}(\mathcal{S}_{L_{i,j}})$. But since $\mathcal{V}_i$ separates $\mathcal{V}_j$ from $\mathcal{V}_{L_{i,j}}$, one has $\mathcal{V}_j\cap\mathcal{V}_{L_{i,j}}\subseteq\mathcal{V}_i\cap\mathcal{V}_j$ and thus $\mathcal{S}_j$ only needs to know $\mathcal{M}_{i,j}$ as defined above. (19) is true at initialization with $L_{i,j}=\emptyset$. Assume it is true at some point in $\mathbf{A}_1$; hence in the RHS of (17) messages $\mathcal{M}_{k,i}$ are projections of systems $\mathcal{S}_{L_{k,i}}$. $\mathcal{S}_i$ separates the various $\mathcal{S}_{L_{k,i}}$ on $\mathcal{G}^c$, so their projections can be merged as in (12):

$$\begin{aligned}\Pi_{\mathcal{V}_i}[\mathcal{S}_i\wedge(\bigwedge_{k\in\mathbf{N}(i)\setminus j}\mathcal{S}_{L_{k,i}})] \;&=\; \mathcal{S}_i\wedge[\bigwedge_{k\in\mathbf{N}(i)\setminus j}\Pi_{\mathcal{V}_i}(\mathcal{S}_{L_{k,i}})]\\ &=\; \mathcal{S}_i\wedge[\bigwedge_{k\in\mathbf{N}(i)\setminus j}\Pi_{\mathcal{V}_i\cap\mathcal{V}_k}(\mathcal{S}_{L_{k,i}})]\end{aligned}$$

where we have used the fact that $\mathcal{S}_i$ and $\mathcal{S}_{L_{k,i}}$ have only $\mathcal{V}_i\cap\mathcal{V}_k$ in common. This yields

$$\mathcal{S}_i\wedge(\bigwedge_{k\in\mathbf{N}(i)\setminus j}\mathcal{M}_{k,i}) \;=\; \Pi_{\mathcal{V}_i}(\mathcal{S}_{L_{i,j}}) \tag{20}$$

$$\text{for} \qquad L_{i,j} := \{i\} \cup \Big(\bigcup_{k \in \mathbf{N}(i) \setminus j} L_{k,i} \Big) \subseteq L_{i<j} \tag{21}$$

By applying $\Pi_{\mathcal{V}_i \cap \mathcal{V}_j}$ to (20), one gets the update equation (17), which thus satisfies (19) for the new $L_{i,j}$ defined at (21). At any application of (17), either the set $L_{i,j}$ grows, or the update is useless ($\mathcal{M}_{i,j}$ is unchanged). Convergence is characterized by $L_{i,j} = L_{i<j}$ for all messages. This state is reachable in a finite number of useful updates due to the tree structure of $\mathcal{G}^c$. At convergence, (18) expresses another valid merge situation where each $\mathcal{M}_{k,i}$ gathers the influence on $\mathcal{S}_i$ of the whole branch behind $\mathcal{S}_k$, so (18) yields $\mathcal{S}'_i = \Pi_{\mathcal{V}_i}(\mathcal{S}_L)$ for $L = \{i\} \cup (\bigcup_{k \in \mathbf{N}(i)} L_{k<i}) = \{1, \ldots, N\}$, and $\mathcal{S}'_i = \Pi_{\mathcal{V}_i}(\mathcal{S})$. □

Observe that $\mathbf{A}_1$ only relies on (a1,a2,a3,a4). But it can be shown in a similar manner that $\mathbf{A}_1$ converges to the desired reduced systems regardless of the initialization value for messages, so (a4) is actually useless.

The tree property of $\mathcal{G}^c$ is essential to prove convergence of $\mathbf{A}_1$. One could suspect that, provided $\mathcal{S}$ lives on a tree, running $\mathbf{A}_1$ on the connectivity graph instead of $\mathcal{G}^c$ would preserve convergence of messages and not alter the result. As will be shown in the sequel, this is true with involutive systems, but doesn't hold in general.

3.4 Alternate algorithm in an involutive setting

Lemma 5 *Define $\mathcal{S}'_i$ by (18) at any step of $\mathbf{A}_1$. In an involutive setting,* i.e. *assuming (a5), one has $\mathcal{S} = \mathcal{S}'_1 \wedge \ldots \wedge \mathcal{S}'_N$ at any time in $\mathbf{A}_1$.*

Proof. This result is known at convergence of $\mathbf{A}_1$, by lemma 2. But it holds at every step of $\mathbf{A}_1$ since $\mathcal{S}'_1 \wedge \ldots \wedge \mathcal{S}'_N$ has $\mathcal{S}_1 \wedge \ldots \wedge \mathcal{S}_N$ as a factor, and $(\mathcal{S}_1 \wedge \ldots \wedge \mathcal{S}_N) \wedge \mathcal{M}_{i,j} = \mathcal{S}_1 \wedge \ldots \wedge \mathcal{S}_N$ using (19) and involutivity. □

Lemma 5 suggests that in an involutive setting, one could directly base processings on components $\mathcal{S}'_i$. This is what we propose now.

Algorithm $\mathbf{A}_2$

1. Initialization

$$\tilde{\mathcal{M}}_{i,j} \;=\; \Pi_{\mathcal{V}_i \cap \mathcal{V}_j}(\mathbb{I}), \qquad \forall (i,j) \in \mathcal{G}^c \tag{22}$$

2. Until stability of messages, select an edge (i,j) and apply the update rule

$$\tilde{\mathcal{M}}_{i,j} \;:=\; \Pi_{\mathcal{V}_i \cap \mathcal{V}_j}[\mathcal{S}_i \wedge (\bigwedge_{k \in \mathbf{N}(i)} \tilde{\mathcal{M}}_{k,i})] \tag{23}$$

3. Termination

$$\tilde{\mathcal{S}}'_i \;:=\; \mathcal{S}_i \wedge (\bigwedge_{k\in\mathbf{N}(i)} \tilde{\mathcal{M}}_{k,i}) \tag{24}$$

At this point, the only difference with $\mathbf{A}_1$ is the symmetry of (23), which "returns" to $\mathcal{S}_j$ the message $\tilde{\mathcal{M}}_{j,i}$ received from it. Since this message was part of the information already present at $\mathcal{S}_j$, it should not alter subsequent computations when sent back to it, thanks to involutivity (we prove this below). $\mathbf{A}_2$ becomes interesting when a vertex i is chosen instead of an edge (i,j), and messages towards all neighbors of i are updated at once. In that case, messages $\tilde{\mathcal{M}}_{i,j}$ become auxiliary variables, and the heart of computations only involves systems $\tilde{\mathcal{S}}'_i$. One gets

Algorithm $\mathbf{A}_{2bis}$

1. Initialization

$$\tilde{\mathcal{S}}'_i \;=\; \Pi_{\mathcal{V}_i}(\mathbb{I}), \qquad 1\leq i\leq N \tag{25}$$

2. Until stability of reduced systems, select a vertex i and apply the update rule

$$\tilde{\mathcal{S}}'_i \;:=\; \mathcal{S}_i \wedge [\bigwedge_{k\in\mathbf{N}(i)} \Pi_{\mathcal{V}_i\cap\mathcal{V}_k}(\tilde{\mathcal{S}}'_k)] \tag{26}$$

To derive (26) in a direct manner, let us come back to lemma 2. If $\mathcal{S} = \mathcal{S}_1 \wedge \ldots \wedge \mathcal{S}_N$, then $\mathcal{S} = \mathcal{S}'_1 \wedge \ldots \wedge \mathcal{S}'_N$ with $\mathcal{S}'_k = \Pi_{\mathcal{V}_k}(\mathcal{S})$. A similar proof yields $\mathcal{S} = \mathcal{S}_i \wedge (\bigwedge_{k\neq i} \mathcal{S}'_k)$. Grouping terms, one also has $\mathcal{S} = \mathcal{S}_i \wedge (\bigwedge_{k\in\mathbf{N}(i)} \mathcal{S}_{L_{k<i}})$ and so $\mathcal{S} = \mathcal{S}_i \wedge (\bigwedge_{k\in\mathbf{N}(i)} \mathcal{S}'_{L_{k<i}})$. Using (a3) on this equation, we derive

$$\begin{aligned}
\mathcal{S}'_i &= \Pi_{\mathcal{V}_i}(\mathcal{S}) \\
&= \mathcal{S}_i \wedge [\bigwedge_{k\in\mathbf{N}(i)} \Pi_{\mathcal{V}_i}(\mathcal{S}'_{L_{k<i}})] \\
&= \mathcal{S}_i \wedge [\bigwedge_{k\in\mathbf{N}(i)} \Pi_{\mathcal{V}_i\cap\mathcal{V}_k}(\mathcal{S}'_{L_{k<i}})]
\end{aligned}$$

The last equality uses the fact that $\mathcal{V}_i \cap \mathcal{V}_{L_{k<i}} = \mathcal{V}_i \cap \mathcal{V}_k$. Since $\Pi_{\mathcal{V}_k}(\mathcal{S}'_{L_{k<i}}) = \mathcal{S}'_k$, one gets

$$\mathcal{S}'_i \;=\; \mathcal{S}_i \wedge [\bigwedge_{k\in\mathbf{N}(i)} \Pi_{\mathcal{V}_i\cap\mathcal{V}_k}(\mathcal{S}'_k)] \tag{27}$$

Therefore, $\mathbf{A}_{2bis}$ computes a solution of the fix-point equation (27) by means of a Gauss-Seidel procedure. We now prove the convergence of this procedure on trees.

Theorem 2 *Let $\mathcal{S} = \mathcal{S}_1 \wedge \ldots \wedge \mathcal{S}_N$ live on a tree, and $\mathcal{G}^c$ be a communication graph for $\mathcal{S}$. Then $\mathbf{A}_2$ converges in finitely many steps, and at convergence $\tilde{\mathcal{S}}'_i = \Pi_{\mathcal{V}_i}(\mathcal{S})$, $1 \leq i \leq N$.*

Proof. Assume $\mathbf{A}_1$ and $\mathbf{A}_2$ run in parallel, which we denote by $\mathbf{A}_1 \| \mathbf{A}_2$: at any choice of an edge (i,j), both messages $\mathcal{M}_{i,j}$ and $\tilde{\mathcal{M}}_{i,j}$ are updated, together with $\mathcal{S}'_j$ and $\tilde{\mathcal{S}}'_j$. We prove several properties of this joint algorithm.

Let us start with an extra property of $\mathbf{A}_1$: at any time, and on any edge (k,l), $\mathcal{M}_{k,l} \wedge \mathcal{M}^{old}_{k,l} = \mathcal{M}_{k,l}$, where $\mathcal{M}^{old}_{k,l}$ denotes any previous value of $\mathcal{M}_{k,l}$. Assume this is true at some point in $\mathbf{A}_1$; we show it is preserved by the update of $\mathcal{M}_{i,j}$ into $\mathcal{M}^{new}_{i,j}$. The current value $\mathcal{M}_{i,j}$ on edge (i,j) was given by[4] $\mathcal{M}_{i,j} = \Pi_{\mathcal{V}_i \cap \mathcal{V}_j}[\mathcal{S}_i \wedge (\bigwedge_{k \in \mathbf{N}(i) \backslash j} \mathcal{M}^{old}_{k,i})]$. By the recursion assumption

$$[\mathcal{S}_i \wedge (\bigwedge_{k \in \mathbf{N}(i) \backslash j} \mathcal{M}^{old}_{k,i})] \wedge [\mathcal{S}_i \wedge (\bigwedge_{k \in \mathbf{N}(i) \backslash j} \mathcal{M}_{k,i})] \;=\; \mathcal{S}_i \wedge (\bigwedge_{k \in \mathbf{N}(i) \backslash j} \mathcal{M}_{k,i}) \qquad (28)$$

and by lemma 1 one obtains $\mathcal{M}^{new}_{i,j} \wedge \mathcal{M}_{i,j} = \mathcal{M}^{new}_{i,j}$. We conclude by transitivity since any $\mathcal{M}^{old}_{i,j}$ is absorbed by $\mathcal{M}_{i,j}$, which is itself absorbed by $\mathcal{M}^{new}_{i,j}$.

Using this result, we show that $\mathbf{A}_1 \| \mathbf{A}_2$ preserves

$$\tilde{\mathcal{M}}_{k,l} = \mathcal{M}_{k,l} \wedge \mathcal{M}^{old}_{l,k}, \qquad \forall (k,l) \in \mathcal{G}^c \qquad (29)$$

This is obviously true at initialization, so assume it is true at some point in $\mathbf{A}_1 \| \mathbf{A}_2$. As a consequence, for every i:

$$\begin{aligned} \tilde{\mathcal{S}}'_i &= \mathcal{S}_i \wedge (\bigwedge_{k \in \mathbf{N}(i)} \tilde{\mathcal{M}}_{k,i}) \\ &= \mathcal{S}_i \wedge (\bigwedge_{k \in \mathbf{N}(i)} \mathcal{M}_{k,i} \wedge \mathcal{M}^{old}_{i,k}) \\ &= \mathcal{S}_i \wedge (\bigwedge_{k \in \mathbf{N}(i)} \mathcal{M}_{k,i}) \\ &= \mathcal{S}'_i \end{aligned} \qquad (30)$$

The third equality uses the fact that $\mathcal{M}^{old}_{i,k}$ is absorbed by $\mathcal{M}^{new}_{i,k} = \Pi_{\mathcal{V}_i \cap \mathcal{V}_k}[\mathcal{S}_i \wedge (\bigwedge_{l \in \mathbf{N}(i) \backslash k} \mathcal{M}_{l,i})]$, as seen above, and thus is absorbed by $\mathcal{S}_i \wedge (\bigwedge_{l \in \mathbf{N}(i) \backslash k} \mathcal{M}_{l,i})$. Assume $\mathcal{M}_{i,j}$ has just been updated into $\mathcal{M}^{new}_{i,j}$; the update of $\tilde{\mathcal{M}}_{i,j}$ can be written

[4] Notice that one may have $\mathcal{M}^{old}_{k,i} = \mathcal{M}_{k,i}$ in this expression.

$\tilde{\mathcal{M}}_{i,j} := \Pi_{\mathcal{V}_i \cap \mathcal{V}_j}(\tilde{\mathcal{S}}'_i)$, hence

$$\begin{aligned}
\tilde{\mathcal{M}}_{i,j} &:= \Pi_{\mathcal{V}_i \cap \mathcal{V}_j}(\mathcal{S}'_i) \\
&= \Pi_{\mathcal{V}_i \cap \mathcal{V}_j}[\mathcal{S}_i \wedge (\bigwedge_{k \in \mathbf{N}(i)} \mathcal{M}_{k,i})] \\
&= \Pi_{\mathcal{V}_i \cap \mathcal{V}_j}[\mathcal{S}_i \wedge (\bigwedge_{k \in \mathbf{N}(i) \setminus j} \mathcal{M}_{k,i}) \wedge \mathcal{M}_{j,i}] \\
&= \Pi_{\mathcal{V}_i \cap \mathcal{V}_j}[\mathcal{S}_i \wedge (\bigwedge_{k \in \mathbf{N}(i) \setminus j} \mathcal{M}_{k,i})] \wedge \mathcal{M}_{j,i} \\
&= \mathcal{M}^{new}_{i,j} \wedge \mathcal{M}_{j,i} \qquad (31)
\end{aligned}$$

So $\tilde{\mathcal{M}}^{new}_{i,j} = \mathcal{M}^{new}_{i,j} \wedge \mathcal{M}_{j,i}$ which proves (29) for the updated messages on edge (i,j).

(29) reveals that $\mathbf{A}_1 \| \mathbf{A}_2$ preserves (19) augmented with the extra property[5] :

$$\exists \tilde{L}_{i,j} \subseteq \{1,\dots,N\} : \quad \tilde{\mathcal{M}}_{i,j} = \Pi_{\mathcal{V}_i \cap \mathcal{V}_j}(\mathcal{S}_{\tilde{L}_{i,j}}) \qquad (32)$$

$$\text{with} \quad \tilde{L}_{i,j} \cap L_{i<j} = L_{i,j} \qquad (33)$$

$$\text{and} \quad \tilde{L}_{i,j} \cap L_{j<i} \subseteq L_{j,i} \qquad (34)$$

After the update of $\tilde{\mathcal{M}}_{i,j}$, $\tilde{L}_{i,j}$ is changed into $\tilde{L}^{new}_{i,j} = \{i\} \cup (\bigcup_{k \in \mathbf{N}(i)} L_{k,i})$. The end of the proof follows that of theorem 1 : sets $\tilde{L}_{i,j}$ can only grow, and reach their maximal value $\{1,\dots,N\}$ in a finite number of useful updates, due to the tree structure of $\mathcal{G}^c$. At convergence, one has $\tilde{\mathcal{S}}'_i = \mathcal{S}'_i = \Pi_{\mathcal{V}_i}(\mathcal{S})$, and $\tilde{\mathcal{M}}_{i,j} = \tilde{\mathcal{M}}_{j,i} = \Pi_{\mathcal{V}_i \cap \mathcal{V}_j}(\mathcal{S}) = \mathcal{M}_{i,j} \wedge \mathcal{M}_{j,i}$. □

Remark : thanks to (30), lemma 5 holds also for components $\tilde{\mathcal{S}}'_i$ defined by (24) along $\mathbf{A}_2$.

3.5 Reconnecting reduced components

We finally mention a composition property of reduced components $\mathcal{S}'_i = \Pi_{\mathcal{V}_i}(\mathcal{S})$ for involutive systems living on a tree.

Lemma 6 *In an involutive setting, let $\mathcal{S} = \mathcal{S}_1 \wedge \ldots \wedge \mathcal{S}_N$ live on a tree, $\mathcal{G}^c$ be one of its communication graphs, and $\mathcal{S}'_i$ be the reduced components. Let $J \subset \{1,\dots,N\}$ define a connected subtree $\mathcal{G}^c{}_{|J}$ of $\mathcal{G}^c$, then $\bigwedge_{i \in J} \mathcal{S}'_j \triangleq \mathcal{S}'_J = \Pi_{\mathcal{V}_J}(\mathcal{S})$.*

Lemma 2 states that the composition of all reduced components $\mathcal{S}'_i$ yields $\mathcal{S}$; here we prove the same kind of property at an intermediate scale : composing part of the reduced components yields a part of $\mathcal{S}$.

[5] Recall that $L_{i<j} \cup L_{j<i}$ is a partition of $\{1,\dots,N\}$ which identifies the two sets of components separated by edge (i,j).

Proof. Let $K = \{1, \ldots, N\} \setminus J$ be the set of removed vertices, and assume $\mathcal{G}^c{}_{|K}$ is a connected branch of $\mathcal{G}^c$. Assume the connection between K and J on $\mathcal{G}^c$ is done by the edge (k, j). Replacing $\mathcal{S}_j$ by $\mathcal{S}_j \wedge \Pi_{\mathcal{V}_j}(\mathcal{S}_K)$ and running $\mathbf{A}_1$ or $\mathbf{A}_2$ on $\mathcal{G}^c{}_{|J}$ yields the same reduced systems $\mathcal{S}'_i$ for $i \in J$. So, by lemma 2, $\mathcal{S}'_J = \mathcal{S}_J \wedge \Pi_{\mathcal{V}_j}(\mathcal{S}_K)$. But $\Pi_{\mathcal{V}_J}(\mathcal{S}) = \Pi_{\mathcal{V}_J}(\mathcal{S}_J \wedge \mathcal{S}_K) = \mathcal{S}_J \wedge \Pi_{\mathcal{V}_J}(\mathcal{S}_K) = \mathcal{S}_J \wedge \Pi_{\mathcal{V}_j}(\mathcal{S}_K)$, whence the result. The general case where several branches must be cut around J can be proved in a similar manner, or by recursion. □

Notice that the connectivity assumption for $\mathcal{G}^c{}_{|J}$ is crucial; the result doesn't hold without it (counter-examples are easy to build).

4 Reduction algorithm for cyclic systems

4.1 Approximate reduction algorithms

Obviously, not all compound systems live on trees, see for example fig. 4. To address the reduction problem in such situations, there are essentially two families of methods: exact methods, and approximations.

On the side of exact methods, the idea is to get back to the tree situation. A popular way to do so consists in aggregating components involved in a cycle, in order to obtain a *junction tree* between groups of components. On fig. 4, for example, one could build $\dot{\mathcal{S}}_1 = \mathcal{S}_1 \wedge \mathcal{S}_2$ and $\dot{\mathcal{S}}_2 = \mathcal{S}_3 \wedge \mathcal{S}_4$, interacting through variables A and C. The major drawback lies in the size of the aggregated components, which penalizes local computations of $\mathbf{A}_1$ and $\mathbf{A}_2$. The less known *conditioning method* does a similar thing. It freezes one (or several) variable(s) to a particular value, which amounts to removing it (them) from $\mathcal{S}$, and may thus open a cycle (for example variable A on fig. 4). The reduction problem can be solved easily on the remaining "conditioned system," if it lives on a tree. The complexity is similar to the junction tree method (all combinations of values must be explored for the conditioning variables). And a technical difficulty remains in the deconditioning step, which combines all reduced components obtained for all values of the frozen variables. In practice, the complexity of exact reduction methods explodes with the number of cycles in the communication graph of the system, and approximate methods are generally preferred.

On the side of approximate methods, the choice is larger (mean field, iterated conditional modes, Gibbs sampling, generalized belief propagation [3], etc.). The objective is to propose a reasonable approximation of the true reduced components, with a reasonable computational effort. We focus here on the so-called turbo procedures, which basically amounts to running the algorithms of the previous section as

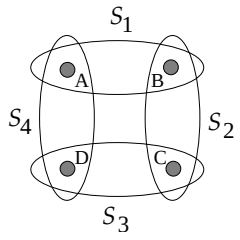


Figure 4: *A compound system $\mathcal{S} = \mathcal{S}_1 \wedge \ldots \wedge \mathcal{S}_4$ not living on a tree.*

if the system was living on a tree. Specifically, in $\mathbf{A}_1$, one had at each step

$$\mathcal{S}_i' \quad = \quad \mathcal{S}_i \wedge [\bigwedge_{k \in \mathbf{N}(i)} \Pi_{\mathcal{V}_i}(\mathcal{S}_{L_{k,i}})] \tag{35}$$

where indexes of $L_{k,i}$ point to components located in the branch of the tree containing k (denoted by $L_{k<i}$). Elements $\Pi_{\mathcal{V}_i}(\mathcal{S}_{L_{k,i}})$ precisely define the messages $\mathcal{M}_{k,i}$ exchanged between components $\mathcal{S}_k$ and $\mathcal{S}_i$. If $\mathcal{S} = \mathcal{S}_1 \wedge \ldots \wedge \mathcal{S}_N$ doesn't live on a tree, $\mathcal{S}_i$ may not separate the connectivity graph into disconnected branches. As a consequence, since the separation criterion is violated, equality $\Pi_{\mathcal{V}_i}(\bigwedge_{k \in \mathbf{N}(i)} \mathcal{S}_{L_{k,i}}) = \bigwedge_{k \in \mathbf{N}(i)} \Pi_{\mathcal{V}_i}(\mathcal{S}_{L_{k,i}})$ necessary to derive (35) doesn't hold in general. It may hold for the first steps of the algorithm, but soon the sets $L_{k,i}$ will overlap, and messages arriving at $\mathcal{S}_i$ will carry correlated information. (Note that this is the main motivation for using a communication graph instead of the connectivity graph: one avoids correlated messages arriving at some system, or, equivalently, a communication graph displays more separation properties than the connectivity graph.) On the example of fig. 4, information from $\mathcal{S}_3$ will reach $\mathcal{S}_1$ both through $\mathcal{S}_2$ and through $\mathcal{S}_4$. Information from $\mathcal{S}_1$ will even come back to $\mathcal{S}_1$ itself through the cycle, whence the name "turbo." As we will see, this is not bothering for involutive systems, but it can cause a divergence in systems with cost functions. Therefore, a renormalization operation on cost functions is usually introduced after projection and composition. In the $(\min, +)$ setting for example, with positive weight functions, one may require $\min_{(\mathbf{v},c) \in \mathcal{S}} \; c = 1$ at any time in any system $\mathcal{S}$, and in the $(+, *)$ setting that $\sum_{(\mathbf{v},c) \in \mathcal{S}} \; c = 1$. These renormalizations can be shown to preserve axioms (a1,...,a4).

Experimentally, turbo procedures have proved to converge and to provide good approximations of the true reduced components $\Pi_{\mathcal{V}_i}(\mathcal{S})$. At least sufficiently good for efficient decoding, in the case of some error correcting codes. A standard interpretation is that the "independence" of messages arriving at a node is approximately preserved provided the cycles are long enough[6]. Nevertheless, convergence is not

[6]For random systems, specifically Markov random fields, this argument is based on the fact that the correlation between components decays geometrically with their distance on the graph, if the

guaranteed in general, unless extra axioms are satisfied, as shown in section 4.2. Now, given a stationary point of algorithms $\mathbf{A}_1$ or $\mathbf{A}_2$ and the corresponding systems $\mathcal{S}_i'$ and $\tilde{\mathcal{S}}_i'$ obtained at this point, how good are these approximations of the true reduced components ? This is explored next, in subsections 4.3 and 4.4.

Remark. In the sequel, we use notations $\mathcal{S}_i'$ and $\tilde{\mathcal{S}}_i'$ to represent results (or stationary points) of algorithms $\mathbf{A}_1$ and $\mathbf{A}_2$ respectively, which may differ from the true reduced components $\Pi_{\mathcal{V}_i}(\mathcal{S})$.

4.2 Axioms ensuring convergence

To study the convergence of reduction algorithms, we need to introduce some notion of "topology" on systems, together with extra axioms defining its relations with $\wedge$ and Π. We thus assume the existence of a partial order $\sqsubseteq$ on systems, where $\mathcal{S} \sqsubseteq \mathcal{S}'$ can be read as *$\mathcal{S}$ contains more information than $\mathcal{S}'$*. We require $\sqsubseteq$ satisfy the following properties :

$$\forall \mathcal{S}, \quad \mathcal{S} \sqsubseteq \mathbb{I} \tag{a6}$$

$$\forall \mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3, \qquad \mathcal{S}_1 \sqsubseteq \mathcal{S}_2 \quad \Rightarrow \quad \mathcal{S}_1 \wedge \mathcal{S}_3 \sqsubseteq \mathcal{S}_2 \wedge \mathcal{S}_3 \tag{a7}$$

$$\forall \mathcal{S}, \mathcal{S}', \forall \mathcal{V} \subseteq \mathcal{V}_{max}, \qquad \mathcal{S} \sqsubseteq \mathcal{S}' \quad \Rightarrow \quad \Pi_{\mathcal{V}}(\mathcal{S}) \sqsubseteq \Pi_{\mathcal{V}}(\mathcal{S}') \tag{a8}$$

Intuitively, (a6) makes $\mathbb{I}$ the least informative system, (a7) states that composition incorporates the same "amount" of information to all systems, and (a8) means that reduction doesn't introduce information.

Example. In the case of constraint systems, where $\mathcal{S}$ is defined by a subset of possible states $\mathcal{O} \subseteq \mathcal{D}_{\mathcal{V}_{max}}$, we define $\mathcal{S} \sqsubseteq \mathcal{S}'$ by $\mathcal{O} \subseteq \mathcal{O}'$. The verification of (a6,a7,a8) is immediate. Observe that $\mathcal{S} \sqsubseteq \mathcal{S}'$ implies $\mathcal{V}_{\mathcal{S}} \supseteq \mathcal{V}_{\mathcal{S}'}$ in this example.

Counter-example. For systems with cost functions, simple attempts to define this topology fail[7]. Let us consider systems with positive cost functions, in the $(\min, +)$ setting for example. A system $\mathcal{S}$ is defined by a subset $\mathcal{O}$ of $\mathcal{D}_{\mathcal{V}_{max}} \times \mathbb{R}^+$, such that $(\mathbf{v}, c) \in \mathcal{O}, (\mathbf{v}, c') \in \mathcal{O} \Rightarrow c = c'$. Let systems $\mathcal{S}, \mathcal{S}'$ be given by $\mathcal{O}, \mathcal{O}'$, and define $\sqsubseteq$ by

$$\mathcal{S} \sqsubseteq \mathcal{S}' \quad \Leftrightarrow \quad \forall (\mathbf{v}, c) \in \mathcal{O}, \ \exists (\mathbf{v}, c') \in \mathcal{O}' \ \text{ and } \ \mathrm{c} \leq \mathrm{c}' \tag{36}$$

global system is far from a phase transition. For constraint systems, similar results are provided by percolation theory.

[7] This is not surprising. No criterion minimized by the turbo algorithm for systems with positive cost functions has been evidenced up to now. To the knowledge of the author, it has only been shown that, in the $(+, *)$ setting, the turbo algorithm converges to an extremal point of a distance between the cost function of $\mathcal{S}$ and a Kikuchi approximation of this cost function. However, whether this point is a local minimum or a saddle point remains unknown [2].

In words, $\mathcal{S} \sqsubseteq \mathcal{S}'$ if $\mathcal{S}$ allows less local states than $\mathcal{S}'$ (*i.e.* gathers more constraints), and if the remaining states have a lower cost value than in $\mathcal{S}'$. With this definition, (a7) is immediate. A problem appears with (a6) since all states of $\mathbb{I}$ must have zero cost to ensure (a4). But this problem could be solved by shifting cost values to $\mathbb{R}^-$ in all systems but $\mathbb{I}$. More seriously, (a8) fails as indicated by the following counter-example. Consider $\mathcal{V}_{max} = \{A, B\}$ and $\mathcal{O} = \{(ab, c_0)\}$, $\mathcal{O}' = \{(ab, c_1), (ab', c_2)\}$. Then $\mathcal{S} \sqsubseteq \mathcal{S}'$ as soon as $c_0 \leq c_1$, whatever the value of c_2. However, reducing systems to variable A, one gets $\Pi_A(\mathcal{S}) = \{(a, c_0)\}$ and $\Pi_A(\mathcal{S}') = \{(a, \min(c_1, c_2))\}$, so $\Pi_A(\mathcal{S}) \sqsubseteq \Pi_A(\mathcal{S}')$ fails for $c_2 \leq c_0$. Notice that choosing $c \geq c'$ in the definition of $\sqsubseteq$ would erase all these problems, but this choice is meaningless in a $(\min, +)$ setting.

Lemma 7 *Let* $\mathbf{A}_1$ *or* $\mathbf{A}_2$ *be applied to* $\mathcal{S} = \mathcal{S}_1 \wedge \ldots \wedge \mathcal{S}_N$*, whatever the structure of its communication graph, with messages initialized to* $\mathbb{I}$*. If axioms (a6,a7,a8) are satisfied, then messages have a decreasing evolution for* $\sqsubseteq$*.*

Proof. Consider the update equation (17) in $\mathbf{A}_1$ for example. Assume some $\mathcal{M}_{k,i}$ has been changed into $\mathcal{M}'_{k,i}$ since the last update of $\mathcal{M}_{i,j}$, with $\mathcal{M}'_{k,i} \sqsubseteq \mathcal{M}_{k,i}$. Then, by (a7,a8), the new message on edge (i, j), denoted by $\mathcal{M}'_{i,j}$, satisfies $\mathcal{M}'_{i,j} \sqsubseteq \mathcal{M}_{i,j}$. Since one has $\mathcal{M}_{i,j} = \mathbb{I}$ at initialization, the first update also yields $\mathcal{M}'_{i,j} \sqsubseteq \mathcal{M}_{i,j} = \mathbb{I}$ by (a6). □

Theorem 3 *Under conditions of lemma 7, the update equation of* $\mathbf{A}_1$ *(resp.* $\mathbf{A}_2$*) has at most one accessible stationary point. If the number of possible states is bounded, (*i.e. $|\mathcal{D}_{\mathcal{V}_{max}} \times \mathcal{D}_{\mathcal{W}}| < \infty$*), this point is reached, whatever the ordering of updates.*

Proof. In case of a bounded number of states, convergence is ensured by lemma 7, so the only point to prove is the uniqueness of accessible stationary points. Consider $\mathbf{A}_1$ for example (the same argument holds for $\mathbf{A}_2$). Let $\bar{\mathcal{M}}_{i,j}$ and $\dot{\mathcal{M}}_{i,j}$ be two stationary points of (17), accessible from initialization $\mathcal{M}_{i,j} = \mathbb{I}$. Let us consider an evolution scheme starting from $\mathcal{M}_{i,j} = \mathbb{I}$ and leading to $\bar{\mathcal{M}}_{i,j}$. At initialization, one has $\dot{\mathcal{M}}_{i,j} \sqsubseteq \mathcal{M}_{i,j} = \mathbb{I}$ on every edge (i, j). By (a7),

$$\mathcal{S}_i \wedge \Big(\bigwedge_{k \in \mathbf{N}(i) \setminus j} \dot{\mathcal{M}}_{k,i} \Big) \quad \sqsubseteq \quad \mathcal{S}_i \wedge \Big(\bigwedge_{k \in \mathbf{N}(i) \setminus j} \mathcal{M}_{k,i} \Big) \tag{37}$$

so by (a8), and using the stationarity of $\dot{\mathcal{M}}_{i,j}$, the updated message $\mathcal{M}_{i,j}$, denoted by $\mathcal{M}_{i,j}^{new}$, satisfies $\dot{\mathcal{M}}_{i,j} \sqsubseteq \mathcal{M}_{i,j}^{new}$. At convergence, one gets $\dot{\mathcal{M}}_{i,j} \sqsubseteq \bar{\mathcal{M}}_{i,j}$. By symmetry, $\dot{\mathcal{M}}_{i,j} \sqsupseteq \bar{\mathcal{M}}_{i,j}$ also holds, whence equality since $\sqsubseteq$ is a partial order. □

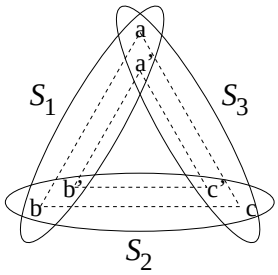


Figure 5: *Detailed view of $\mathcal{S} = \mathcal{S}_1 \wedge \mathcal{S}_2 \wedge \mathcal{S}_3$ where values of A, B, C are displayed. Values related by a dashed line define legal tuples of a system.*

This result deserves several comments. First, observe that (17) may have several stationary points, but at most one of them is accessible. Consider $\mathcal{V}_{max} = \{A, B, C\}$ and constraint systems $\mathcal{S}_1 = \{ab, a'b'\}, \mathcal{S}_2 = \{bc, b'c'\}, \mathcal{S}_3 = \{ca, c'a'\}$ (fig. 5). These components correspond to the unique accessible stationary point of $\mathbf{A}_1$ (by the way, they are also the true reduced components of $\mathcal{S}$). Nevertheless, $\mathcal{M}_{1,2} = \{b\} = \mathcal{M}_{2,1}$, $\mathcal{M}_{2,3} = \{c\} = \mathcal{M}_{3,2}$, $\mathcal{M}_{3,1} = \{a\} = \mathcal{M}_{1,3}$ defines another stationary point which is not accessible (*idem* with primed letters). Secondly, to get convergence, the strong assumption of a bounded number of states, to get convergence is necessary to compensate the weakness of the "topology" defined by $\sqsubseteq$. This captures only special forms of systems with cost functions, but discards usual ones. A strategy could be to define $\mathcal{S} \sqsubseteq \mathcal{S}'$ by $\phi(\mathcal{S}) \geq \phi(\mathcal{S}')$, where ϕ is a measure of the information present in a system. But in that case, $\sqsubseteq$ would only be a pre-order (unless ϕ is injective), which breaks the uniqueness of the accessible point. Moreover, for general random systems, such increasing information measure on messages of turbo algorithms have not been evidenced up to now.

4.3 Properties of stationary points

We now put aside conditions for convergence of $\mathbf{A}_1$ and $\mathbf{A}_2$, and rather consider properties of stationary points of their update equations.

4.3.1 Expanded systems

The first result states that $\mathbf{A}_1$ and $\mathbf{A}_2$ are insensitive to the global structure of $\mathcal{G}^c$, but only captures its *local* structure.

Definition 4 *Let $\mathcal{G} = (V, E)$ be a graph. $\hat{\mathcal{G}} = (\hat{V}, \hat{E})$ is a locally isomorphic expansion of $\mathcal{G}$ iff there exists an onto morphism[8] $\lambda : \hat{V} \to V$ which is bijective between*

[8] *i.e.* λ preserves edges

$\{\hat{v}\} \cup \partial(\hat{v})$ *and* $\{\lambda(\hat{v})\} \cup \partial(\lambda(\hat{v}))$ *for every* $\hat{v} \in \hat{V}$, *where* $\partial(.)$ *denote neighbors of a vertex.*

Expansions of $\mathcal{G}$ can be easily constructed by superimposing several copies of $\mathcal{G}$ and crossing edges between columns associated to neighboring vertices of $\mathcal{G}$ (see fig. 6). There exists a unique (up to an isomorphism) tree being an expansion of $\mathcal{G}$, named the universal covering tree. It is isomorphic to $\mathcal{G}$ if $\mathcal{G}$ is a tree, and is infinite as soon as $\mathcal{G}$ has a cycle.

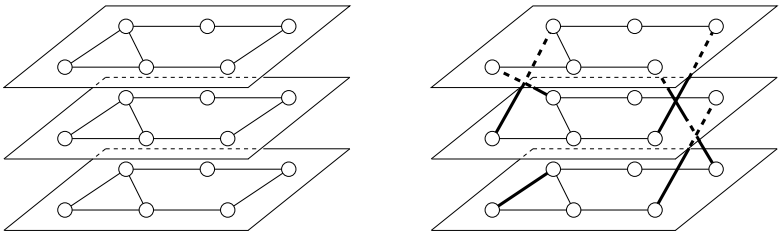


Figure 6: *Construction of a locally isomorphic expansion of* $\mathcal{G}$, *by taking several copies of* $\mathcal{G}$ *and crossing edges between them.*

This notion extends to systems. Let us first say that systems $\mathcal{S}$ and $\hat{\mathcal{S}}$ are isomorphic iff they are identical up to a renaming of variables (we denote by $\phi : \mathcal{V}_{\mathcal{S}} \to \mathcal{V}_{\hat{\mathcal{S}}}$ this bijective mapping), then

Definition 5 *Let* $\mathcal{S} = \mathcal{S}_1 \wedge \ldots \wedge \mathcal{S}_N$ *and* $\hat{\mathcal{S}} = \hat{\mathcal{S}}_1 \wedge \ldots \wedge \hat{\mathcal{S}}_M$ *be two systems, and let* $\mathcal{G}^c$ *and* $\hat{\mathcal{G}}^c$ *denote generic communication graphs for* $\mathcal{S}$ *and* $\hat{\mathcal{S}}$. $\hat{\mathcal{S}}$ *is a locally isomorphic expansion of* $\mathcal{S}$ *iff*

1. *there exist communication graphs* $\mathcal{G}^c$ *and* $\hat{\mathcal{G}}^c$ *and an onto mapping* $\lambda : \{1, \ldots, M\} \to \{1, \ldots, N\}$ *which makes* $\hat{\mathcal{G}}^c$ *a locally isomorphic expansion of* $\mathcal{G}^c$,
2. *there exists a mapping* $\phi : \mathcal{V}_{\hat{\mathcal{S}}} \to \mathcal{V}_{\mathcal{S}}$ *such that components* $\hat{\mathcal{S}}_i$ *and* $\mathcal{S}_{\lambda(i)}$ *are isomorphic through* ϕ, $1 \leq i \leq M$.

Observe that ϕ is onto, and for every edge (i, j) of $\hat{\mathcal{G}}^c$, neighboring components share the same variables as their images by λ, *i.e.* $\phi(\mathcal{V}_{\hat{\mathcal{S}}_i} \cap \mathcal{V}_{\hat{\mathcal{S}}_j}) = \mathcal{V}_{\mathcal{S}_{\lambda(i)}} \cap \mathcal{V}_{\mathcal{S}_{\lambda(j)}}$. The following result generalizes the "periodic assignment lemma" mentioned in [1].

Lemma 8 *Let* $\hat{\mathcal{S}}$ *be a locally isomorphic expansion of* $\mathcal{S}$ *through mappings* λ, ϕ. *If messages* $\mathcal{M}_{i,j}$ *are a stationary point of the update equation (17) of* $\mathbf{A}_1$ *(resp. (23) of* $\mathbf{A}_2$*) on* $\mathcal{G}^c$, *their inverse image by* λ, ϕ *define messages* $\hat{\mathcal{M}}_{k,l}$ *which are a stationary point of* $\mathbf{A}_1$ *(resp.* $\mathbf{A}_2$*) on* $\hat{\mathcal{G}}^c$. *Moreover, if the* $\mathcal{M}_{i,j}$ *are an accessible stationary point on* $\mathcal{G}^c$, *then the* $\hat{\mathcal{M}}_{k,l}$ *are also accessible on* $\hat{\mathcal{G}}^c$.

Proof. Stationarity is obvious: define messages $\hat{\mathcal{M}}_{k,l}$ on $\hat{\mathcal{G}}^c$ as the image by ϕ^{-1} of $\mathcal{M}_{\lambda(k),\lambda(l)}$. If the $\mathcal{M}_{i,j}$ are a stationary point of, say, (17) on $\mathcal{G}^c$, the $\hat{\mathcal{M}}_{k,l}$ still satisfy (17) on $\hat{\mathcal{G}}^c$, thanks to the local isomorphism property. To prove accessibility, run $\mathbf{A}_1$ simultaneously on $\mathcal{G}^c$ and $\hat{\mathcal{G}}^c$, and each time an update is performed on an edge (i,j) of $\mathcal{G}^c$, update also messages on all edges (k,l) of $\lambda^{-1}[(i,j)]$ on $\hat{\mathcal{G}}^c$. This preserves $\hat{\mathcal{M}}_{k,l} = \phi^{-1}(\mathcal{M}_{i,j})$ on these edges. □

Notice that, conversely, stationary messages $\hat{\mathcal{M}}_{k,l}$ on $\hat{\mathcal{G}}^c$ yield stationary messages $\mathcal{M}_{i,j}$ on $\mathcal{G}^c$ only if all edges (k,l) in $\lambda^{-1}[(i,j)]$ carry isomorphic messages, for every (i,j) of $\mathcal{G}^c$. This property is not guaranteed however; counter examples are easy to build.

4.3.2 Extendibility to every nested tree

Let the $\mathcal{S}'_i$ be obtained at a stationary point of $\mathbf{A}_1$. $\mathbf{A}_1$ being an approximation, a local state $\mathbf{v}_i$ in $\mathcal{S}'_i$ may not be the projection of a global state $\mathbf{v}$ of $\mathcal{S}$. Nevertheless, we show that $\mathbf{v}_i$ is part of a larger state covering not all, but several components of $\mathcal{S}$.

Theorem 4 *Let $\mathcal{G}^c$ be a communication graph for $\mathcal{S} = \mathcal{S}_1 \wedge \ldots \wedge \mathcal{S}_N$, let the $\mathcal{M}_{i,j}$ be a stationary point of (17) on $\mathcal{G}^c$, and build approximate reduced systems $\mathcal{S}'_i$ by (18). Select $J \subseteq \{1,\ldots,N\}$ such that subgraph $\mathcal{G}^c{}_{|J}$ is a tree, and define $\bar{\mathcal{S}}_i = \mathcal{S}_i \wedge (\bigwedge_{k\in\mathbf{N}(i)\setminus J} \mathcal{M}_{k,i})$, $1 \le i \le N$ then $\forall j \in J$, $\mathcal{S}'_j = \Pi_{\mathcal{V}_j}(\bar{\mathcal{S}}_J)$.*

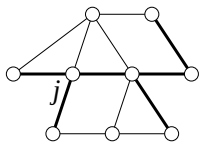


Figure 7: *A communication graph $\mathcal{G}^c$ and a nested tree $\mathcal{G}^c{}_{|J}$ (thick edges) around vertex j.*

This theorem expresses that every local state $\mathbf{v}_j \in \mathcal{S}'_j$ can be extended into a larger state $\mathbf{v}_J$ over *any tree* around j ($\mathbf{v}_J$ may not involve all variables of $\mathcal{S}$, however). In terms of constraint systems, this means that only a cycle of $\mathcal{G}^c$ could determine that a $\mathbf{v}_j$ in some system $\mathcal{S}'_j$ doesn't belong to the true $\Pi_{\mathcal{V}_j}(\mathcal{S})$ (see the example of fig. 8). In other words, $\mathbf{A}_1$ is blind to cycles.

Proof. Let us replace $\mathcal{S}_i$ by $\bar{\mathcal{S}}_i$ on J. $\bar{\mathcal{S}}_i$ gathers the influence of messages coming from outside the tree $\mathcal{G}^c{}_{|J}$. By the stationarity assumption, on every edge (i,j) of

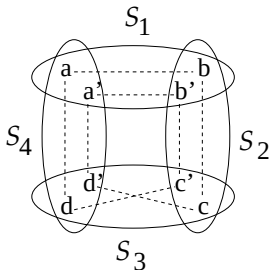


Figure 8: *Detailed view of components in* $\mathcal{S} = \mathcal{S}_1 \wedge \ldots \wedge \mathcal{S}_4$. *The drawing expresses, for example, that* $\mathcal{S}_1 = (\{A, B\}, \{(a, b), (a', b')\})$. $\mathbf{A}_1$ *doesn't reduce these components. In every* $\mathcal{S}'_i$, *a local state can be extended to the tree formed with its two neighbors. But not further,* i.e. *not to the entire* $\mathcal{S}$ *:* $\mathcal{S}$ *is actually empty.*

$\mathcal{G}^c{}_{|J}$ one has

$$\mathcal{M}_{i,j} \quad = \quad \bar{\mathcal{S}}_i \wedge \Big(\bigwedge_{k \in (\mathbf{N}(i) \setminus j) \setminus J} \mathcal{M}_{k,i} \Big) \tag{38}$$

and the corresponding approximate reduced systems satisfy

$$\mathcal{S}'_i \quad = \quad \bar{\mathcal{S}}_i \wedge \Big(\bigwedge_{k \in \mathbf{N}(i) \setminus J} \mathcal{M}_{k,i} \Big) \tag{39}$$

These equations characterize the only accessible stationary point of $\mathbf{A}_1$ on tree $\mathcal{G}^c{}_{|J}$ for system $\bar{\mathcal{S}}_J = \bigwedge_{j \in J} \bar{\mathcal{S}}_j$, whence the result by theorem 1. □

4.3.3 Local optimality

The extension result of theorem 4 has a local optimality interpretation in the case of probabilistic systems, which was first mentioned in [1].

We consider systems with positive weight functions, in the $(\max, *)$ setting. Let $\alpha(\mathcal{S})$ represent a normalization operation on the weight function of $\mathcal{S}$ ensuring

$$\max_{(\mathbf{v}, w) \in \mathcal{S}} w \quad = \quad 1 \tag{40}$$

where $w \in \mathcal{D}_W = \mathbb{R}^+$ and $\mathbf{v}$ ranges over $\mathcal{D}_{\mathcal{V}_{max}}$, or equivalently over $\mathcal{D}_{\mathcal{V}_\mathcal{S}}$. Replacing projection $\Pi_\mathcal{V}$ by $\dot{\Pi}_\mathcal{V} \triangleq \alpha \circ \Pi_\mathcal{V}$, and redefining composition as $\mathcal{S}_1 \dot{\wedge} \mathcal{S}_2 \triangleq \alpha(\mathcal{S}_1 \wedge \mathcal{S}_2)$ preserves the validity of axioms[9] (a1) to (a4).

[9] Moreover, one gets $\forall \mathcal{S}$, $\dot{\Pi}_\emptyset(\mathcal{S}) = \mathbb{I}$, where $\mathbb{I}$ is composed of all states $\mathcal{D}_{\mathcal{V}_{max}}$ with weight value 1, *i.e.* $\mathbb{I} = (\emptyset, \{(*, 1)\})$.

For $\mathcal{S} = \mathcal{S}_1 \dot{\wedge} \ldots \dot{\wedge} \mathcal{S}_N$ living on a tree, $\mathbf{A}_1$ converges, by theorem 1, and yields reduced systems $\mathcal{S}'_i = \dot{\Pi}_{\mathcal{V}_i}(\mathcal{S})$. Let us denote by $\mathbf{v}^\star$ (resp. $\mathbf{v}_i^\star$) a state of $\mathcal{S}$ (resp. $\mathcal{S}'_i$) having weight 1, *i.e.* a most likely state of $\mathcal{S}$ (resp. $\mathcal{S}'_i$) in the probabilistic interpretation of weights. Observe that normalized systems directly have normalized projections, *i.e.* $\mathcal{S} = \alpha(\mathcal{S})$ induces $\mathcal{S}'_i = \dot{\Pi}_{\mathcal{V}_i}(\mathcal{S}) = \Pi_{\mathcal{V}_i}(\mathcal{S})$. Moreover, by definition of $\Pi_{\mathcal{V}_i}$, the restriction of a $\mathbf{v}^\star$ to variables $\mathcal{V}_i$ necessarily yields a $\mathbf{v}_i^\star$, and conversely a $\mathbf{v}_i^\star$ is necessarily[10] part of at least one $\mathbf{v}^\star$. So, if components $\mathcal{S}'_i$ contain a single $\mathbf{v}_i^\star$ at convergence of $\mathbf{A}_1$, these local states can be composed to form the *unique* optimal state $\mathbf{v}^\star$ of $\mathcal{S}$: $\{\mathbf{v}^\star\} = \bigwedge_i \{\mathbf{v}_i^\star\}$.

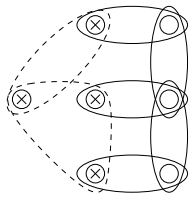


Figure 9: *In this compound system, solid lines identify components selected to form the tree* $\mathcal{G}^c{}_{|J}$. *Variables represented with a cross are frozen to their value in* $\mathbf{v}$. *Changing the value of remaining variables* $\mathcal{V}'_J$ *(internal to* $\mathcal{S}_J$*) cannot give a more likely state than* $\mathbf{v}$.

This result can be partially extended to a general compound system. Let $\mathcal{G}^c$ be a communication graph of $\mathcal{S} = \mathcal{S}_1 \dot{\wedge} \ldots \dot{\wedge} \mathcal{S}_N$. Assume $J \subseteq \{1, \ldots, N\}$ defines a subtree $\mathcal{G}^c{}_{|J}$ of components of $\mathcal{S}$. At a stationary point[11] of $\mathbf{A}_1$, by theorem 4, one has $\mathcal{S}'_j = \dot{\Pi}_{\mathcal{V}_j}(\bar{\mathcal{S}}_J)$, with $\bar{\mathcal{S}}_J$ defined as in theorem 4. Hence, with the same reasoning as above, any $\mathbf{v}_j^\star$ of $\mathcal{S}'_j$ extends into a locally optimal $\mathbf{v}_J^\star$ of $\bar{\mathcal{S}}_J$, where local optimality refers to the weight function of $\bar{\mathcal{S}}_J$. By generalization, a $\mathbf{v}_j^\star$ of $\mathcal{S}'_j$ extends into a locally optimal $\mathbf{v}_J^\star$ over any tree $\mathcal{G}^c{}_{|J}$ around j.

The weight function of $\bar{\mathcal{S}}_J$ is different from W, the weight function of $\mathcal{S}$. To make the connection, we need two extra observations.

First, let $I = \{1, \ldots, N\} \setminus J$ and denote by $\mathcal{V}_{I,J} = \mathcal{V}_I \cap \mathcal{V}_J$ the set of shared variables between $\bar{\mathcal{S}}_J$ and remaining components in system $\mathcal{S}$. In the same way, define sets of "private" variables as $\mathcal{V}'_I = \mathcal{V}_I \setminus \mathcal{V}_J$ for $\mathcal{S}_I$, and symmetrically $\mathcal{V}'_J$ for $\bar{\mathcal{S}}_J$. By construction, the weight functions $\bar{W}_J$ and W_J of systems $\bar{\mathcal{S}}_J$ and $\mathcal{S}_J$ respectively

[10] Strictly speaking, this only holds for systems with a finite number of elements.

[11] In practice, the renormalization α is central to avoid divergence of weights when $\mathbf{A}_1$ is applied on a general graph, although convergence has not been proved, as mentioned earlier.

are related by[12]

$$\bar{W}_J(\mathbf{v}'_J, \mathbf{v}_{I,J}) \quad = \quad W_J(\mathbf{v}'_J, \mathbf{v}_{I,J})\mu(\mathbf{v}_{I,J}) \tag{41}$$

for some function μ due to messages appearing in the definition of components $\bar{\mathcal{S}}_j$.

Secondly, considering factorization $\mathcal{S} = \mathcal{S}_I \dot{\wedge} \mathcal{S}_J$, and denoting by W_I the weight function in $\mathcal{S}_I$, observe that

$$W(\mathbf{v}'_I, \mathbf{v}_{I,J}, \mathbf{v}'_J) \quad \propto \quad W_I(\mathbf{v}'_I, \mathbf{v}_{I,J}) W_J(\mathbf{v}'_J, \mathbf{v}_{I,J}) \tag{42}$$

for elements $(\mathbf{v}'_I, \mathbf{v}_{I,J}, \mathbf{v}'_J)$ of $\mathcal{S}$. Therefore, for a fixed value of $(\mathbf{v}'^{\star}_I, \mathbf{v}^{\star}_{I,J})$, maximizing W over $\mathbf{v}'_J$ in $\mathcal{S}$ amounts to maximizing $W_J(\mathbf{v}'_J, \mathbf{v}^{\star}_{I,J})$ or any related weight function $W_J(\mathbf{v}'_J, \mathbf{v}^{\star}_{I,J}) \cdot \mu(\mathbf{v}^{\star}_{I,J})$. In $\bar{\mathcal{S}}_J$, we have already underlined that $\bar{W}_J(\mathbf{v}'^{\star}_J, \mathbf{v}^{\star}_{I,J})$ cannot be improved by changing $\mathbf{v}'^{\star}_J$ into any other possible value $\mathbf{v}'_J$. Using (41), this holds also for $W_J(\mathbf{v}'^{\star}_J, \mathbf{v}^{\star}_{I,J})$, and by consequence for $W(\mathbf{v}'^{\star}_J, \mathbf{v}^{\star}_{I,J}, \mathbf{v}'^{\star}_I)$ (see fig. 9). This proves the following corollary to theorem 4:

Corollary 1 *Let the $\mathcal{S}'_i$ be derived from a stationary point of $\mathbf{A}_1$ on $\mathcal{G}_c$, a communication graph of $\mathcal{S} = \mathcal{S}_1 \dot{\wedge} \ldots \dot{\wedge} \mathcal{S}_N$. Let $\mathbf{v}$ be a state of $\mathcal{S}$ such that $\mathbf{v}^{\star}_i$, its restriction to variables $\mathcal{V}_i$, is an element of weight 1 in $\mathcal{S}'_i$. Then $\mathbf{v}$ is locally optimal in $\mathcal{S}$ in the following sense: $\forall J \subseteq \{1, \ldots, N\}$ such that $\mathcal{G}^c{}_{|J}$ is a tree, let $I = \{1, \ldots, N\} \setminus J$ and $\mathcal{V}'_J = \mathcal{V}_J \setminus \mathcal{V}_I$, any other state $\mathbf{v}'$ differing from $\mathbf{v}$ only on variables of $\mathcal{V}'_J$ has lower weight that $\mathbf{v}$ in $\mathcal{S}$.*

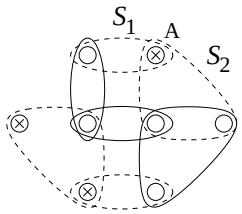


Figure 10: *Same notations as fig. 9. If J is such that $\mathcal{H}_{|\mathcal{V}_J}$ has a tree as communication graph, changing the value of* all *variables $\mathcal{V}_J$ (not only private variables $\mathcal{V}'_J$) cannot give a more likely state than $\mathbf{v}$. Here, taking $J \cup \{1\}$ would violate this condition: variable A would close a cycle through $\mathcal{S}_2$.*

In [1], this result is stated in a stronger form. Namely, all variables of $\mathcal{V}_J$ can actually be modified without improving $W(\mathbf{v})$, provided the hypergraph $\mathcal{H}_{|\mathcal{V}_J}$ has a tree as communication graph (fig. 10). The proof of $W(\mathbf{v}^{\star}_J, \mathbf{v}'^{\star}_I) \geq W(\mathbf{v}_J, \mathbf{v}'^{\star}_I)$ can be obtained in a similar manner using a locally isomorphic system in which cycles have been expanded, and relying on lemma 8.

[12] This relation holds for elements $(\mathbf{v}'_J, \mathbf{v}_{I,J})$ lying in both systems $\bar{\mathcal{S}}_J$ and $\mathcal{S}_J$, of course.

4.4 Specific properties of involutive systems

4.4.1 Comparison of stationary points of $\mathbf{A}_1$ and $\mathbf{A}_2$

Lemma 9 *In an involutive setting, let $\mathcal{G}^c$ be a communication graph for $\mathcal{S} = \mathcal{S}_1 \wedge \ldots \wedge \mathcal{S}_N$. Let the $\mathcal{M}_{i,j}$ and $\mathcal{S}'_i$ be a stationary point of $\mathbf{A}_1$ on $\mathcal{G}^c$, then the $\tilde{\mathcal{M}}_{i,j} \triangleq \mathcal{M}_{i,j} \wedge \mathcal{M}_{j,i}$, $\tilde{\mathcal{S}}'_i \triangleq \mathcal{S}'_i$ define a stationary point of $\mathbf{A}_2$.*

Proof. Let us define $\tilde{\mathcal{S}}'_i$ as $\mathcal{S}'_i$, and $\tilde{\mathcal{M}}_{i,j}$ as $\mathcal{M}_{i,j} \wedge \mathcal{M}_{j,i}$. On every edge (i,j) one has

$$\Pi_{\mathcal{V}_i \cap \mathcal{V}_j}(\tilde{\mathcal{S}}'_i) = \Pi_{\mathcal{V}_i \cap \mathcal{V}_j}[\mathcal{M}_{j,i} \wedge \mathcal{S}_i \wedge (\bigwedge_{k \in \mathbf{N}(i) \setminus j} \mathcal{M}_{k,i})] = \mathcal{M}_{j,i} \wedge \mathcal{M}_{i,j} \tag{43}$$

using stationarity for $\mathbf{A}_1$, so $\tilde{\mathcal{M}}_{i,j} = \Pi_{\mathcal{V}_i \cap \mathcal{V}_j}(\tilde{\mathcal{S}}'_i)$ holds. Moreover,

$$\mathcal{S}_i \wedge (\bigwedge_{k \in \mathbf{N}(i)} \tilde{\mathcal{M}}_{k,i}) = \mathcal{S}_i \wedge (\bigwedge_{k \in \mathbf{N}(i)} \mathcal{M}_{k,i}) \wedge (\bigwedge_{k \in \mathbf{N}(i)} \mathcal{M}_{i,k}) = \mathcal{S}'_i \wedge (\bigwedge_{k \in \mathbf{N}(i)} \mathcal{M}_{i,k}) = \mathcal{S}'_i \tag{44}$$

since $\mathcal{S}'_i \wedge \mathcal{M}_{i,k} = \mathcal{S}'_i$ using involutivity and stationarity of $\mathbf{A}_1$. So $\tilde{\mathcal{S}}'_i = \mathcal{S}_i \wedge (\bigwedge_{k \in \mathbf{N}(i)} \tilde{\mathcal{M}}_{k,i})$, which proves stationarity of the update equations in $\mathbf{A}_2$. □

The converse result cannot be proved so directly : there is no closed form expression of messages $\mathcal{M}_{i,j}$ in terms of messages $\tilde{\mathcal{M}}_{i,j}$, so it is difficult to build a stationary point of $\mathbf{A}_1$ from one of $\mathbf{A}_2$. Nevertheless

Theorem 5 *In an involutive setting, let $\mathcal{G}^c$ be a communication graph for $\mathcal{S} = \mathcal{S}_1 \wedge \ldots \wedge \mathcal{S}_N$. Algorithms $\mathbf{A}_1$ and $\mathbf{A}_2$ have the same accessible stationary point on $\mathcal{G}_c$ (in terms of components $\mathcal{S}'_i$ and $\tilde{\mathcal{S}}'_i$), if this point exists for one of them.*

The proof relies on the following lemma

Lemma 10 *In an involutive setting, define relation $\sqsubseteq$ by*

$$\mathcal{S}_1 \sqsubseteq \mathcal{S}_2 \quad \Leftrightarrow \quad \mathcal{S}_1 \wedge \mathcal{S}_2 = \mathcal{S}_1 \tag{45}$$

Then $\sqsubseteq$ is a partial order relation which satisfies axioms (a6,a7,a8).

Proof. Transitivity, reflexivity and antisymmetry of $\sqsubseteq$ are obvious, so we have a partial order. (a6) is a transcription of (a4), (a7) is obvious with $\mathcal{S}_3 \wedge \mathcal{S}_3 = \mathcal{S}_3$, and (a8) has been proved in lemma 1. □

Proof of theorem 5. By application of theorem 3, $\mathbf{A}_1$ and $\mathbf{A}_2$ each have at most one accessible stationary point (this point exists if components are finite). So we only

have to prove they are identical. We proceed as for theorem 2, by running $\mathbf{A}_1 \| \mathbf{A}_2$. Properties (29,30) still hold: they don't need any assumption on the underlying graph of the algorithm. (But closed form characterizations (19,32) of messages in terms of components are not valid anymore.) We conclude by observing that $\tilde{\mathcal{S}}'_i = \mathcal{S}'_i$ is true all along $\mathbf{A}_1 \| \mathbf{A}_2$, so in particular when both $\mathbf{A}_1$ and $\mathbf{A}_2$ have converged to their unique accessible stationary point. □

Remark: the proof also reveals that $\mathbf{A}_1$ and $\mathbf{A}_2$ converge at the same time in $\mathbf{A}_1 \| \mathbf{A}_2$, still in terms of components $\tilde{\mathcal{S}}'_i$ and $\mathcal{S}_i$.

4.4.2 Invariance by graph change

For systems living on a tree, the result of $\mathbf{A}_1$ or $\mathbf{A}_2$ is optimal, and in particular insensitive to the choice of graph $\mathcal{G}^c$. For general systems, whether accessible stationary points depend on the communication graph chosen to run the turbo algorithm remains an open question. However, this invariance can be proved for involutive systems.

Lemma 11 *In an involutive setting, let $\mathcal{S} = \mathcal{S}_1 \wedge \ldots \wedge \mathcal{S}_N$. Let graphs $\mathcal{G}_1, \mathcal{G}_2$ satisfy $\mathcal{G}_c \subseteq \mathcal{G}_1 \subseteq \mathcal{G}_2 \subseteq \mathcal{G}^{cnx}$, where $\mathcal{G}^c$ is a communication graph of $\mathcal{S}$, and $\mathcal{G}^{cnx}$ its connectivity graph. We assume $\mathcal{G}_2 = \mathcal{G}_1 \cup \{(i,j)\}$, where (i,j) is a redundant edge on $\mathcal{G}_2$. Then $\mathbf{A}_1$ (resp. $\mathbf{A}_2$) run on $\mathcal{G}_1$ or on $\mathcal{G}_2$ yields the same accessible stationary point, in terms of reduced components $\mathcal{S}'_k$ (resp. $\tilde{\mathcal{S}}'_k$).*

Proof. Observe first that theorem 5 remains valid for any support graph of $\mathbf{A}_1$ and $\mathbf{A}_2$. So it is enough to prove the lemma for $\mathbf{A}_2$.

We denote by $(i = k_0, k_1, k_2, \ldots, k_L = j)$ a redundancy path for edge (i,j) on $\mathcal{G}_2$. Let the $\tilde{\mathcal{S}}'_k$ (and consequently the $\tilde{\mathcal{M}}_{k,l}$) define an accessible stationary point of $\mathbf{A}_2$ run on $\mathcal{G}_1$; we must show that these $\tilde{\mathcal{S}}'_k$ are also accessible to $\mathbf{A}_2$ run on $\mathcal{G}_2$, which allows to conclude by unicity of accessible stationary points, as stated in theorem 5.

Let us define $\tilde{\mathcal{M}}_{i,j}$ on the missing edge (i,j) by $\tilde{\mathcal{M}}_{i,j} = \Pi_{\mathcal{V}_i \cap \mathcal{V}_j}(\tilde{\mathcal{S}}'_i) = \Pi_{\mathcal{V}_i \cap \mathcal{V}_j}(\tilde{\mathcal{S}}'_{k_0})$. Using the fact that $\mathcal{V}_i \cap \mathcal{V}_j \subseteq \mathcal{V}_{k_0} \cap \mathcal{V}_{k_1}$ and $\tilde{\mathcal{M}}_{k_0,k_1} = \Pi_{\mathcal{V}_{k_0} \cap \mathcal{V}_{k_1}}(\tilde{\mathcal{S}}'_{k_0})$, one has $\tilde{\mathcal{M}}_{i,j} \wedge \tilde{\mathcal{M}}_{k_0,k_1} = \tilde{\mathcal{M}}_{k_0,k_1}$, and consequently $\tilde{\mathcal{M}}_{i,j} \wedge \tilde{\mathcal{S}}'_{k_1} = \tilde{\mathcal{S}}'_{k_1}$. By recursion on the same scheme, one has $\tilde{\mathcal{M}}_{i,j} \wedge \tilde{\mathcal{S}}'_{k_l} = \tilde{\mathcal{S}}'_{k_l}$ for $0 \leq l \leq L$. So the new message $\tilde{\mathcal{M}}_{i,j}$ doesn't change $\tilde{\mathcal{S}}'_j$. And conversely, the new message $\tilde{\mathcal{M}}_{j,i}$ in the reverse direction doesn't change $\tilde{\mathcal{S}}'_i$. We have proved that the $\tilde{\mathcal{S}}'_k$ form a stationary point for $\mathbf{A}_2$ on $\mathcal{G}_2$. Accessibility of this point on $\mathcal{G}_2$ is obvious: simply run $\mathbf{A}_2$ on $\mathcal{G}_2$ ignoring the presence of edge (i,j), which is equivalent to running it on $\mathcal{G}_1$. Then update messages $\tilde{\mathcal{M}}_{i,j}$ and $\tilde{\mathcal{M}}_{j,i}$ once the stationary systems $\tilde{\mathcal{S}}'_k$ are obtained. □

Theorem 6 *In an involutive setting, let $\mathcal{S} = \mathcal{S}_1 \wedge \ldots \wedge \mathcal{S}_N$, and $\mathcal{G}^{cnx}$ be the connectivity graph of $\mathcal{S}$. Let graph $\mathcal{G} \subset \mathcal{G}^{cnx}$ be obtained by removing some redundant edges of $\mathcal{G}^{cnx}$. Whatever the choice of $\mathcal{G}$, $\mathbf{A}_1$ (resp. $\mathbf{A}_2$) yields the same result, in terms of components $\mathcal{S}'_i$ (resp. $\tilde{\mathcal{S}}'_i$).*

Proof. Direct, by recursive application of lemma 11. □

This result allows to replace the use of a communication graph, which requires some global knowledge on the structure of $\mathcal{S}$, by the connectivity graph, a purely local information. It also evidences the strength of the involutivity assumption.

4.4.3 Refinements of the extendibility property

Lemma 12 *In an involutive setting, let $\mathcal{G}^c$ be a communication graph for $\mathcal{S} = \mathcal{S}_1 \wedge \ldots \wedge \mathcal{S}_N$, and $\mathcal{S}'_i$ (resp. $\tilde{\mathcal{S}}'_i$) be obtained at a stationary point of $\mathbf{A}_1$ (resp. $\mathbf{A}_2$). Select $K \subseteq J \subseteq \{1, \ldots, N\}$ such that subgraph $\mathcal{G}^c{}_{|J}$ is a tree, and $\mathcal{G}^c{}_{|K}$ a connected subtree of $\mathcal{G}^c{}_{|J}$. Then $\mathcal{S}'_K = \Pi_{\mathcal{V}_K}(\mathcal{S}'_J)$ (resp. $\tilde{\mathcal{S}}'_K = \Pi_{\mathcal{V}_K}(\tilde{\mathcal{S}}'_J)$).*

This lemma expresses that every state $\mathbf{v}_K$ of $\mathcal{S}'_K$ can be extended to any tree around K. It generalizes theorem 4, which states the same property for K reduced to a single vertex. A recursive use of lemma 12 shows also that a local state $\mathbf{v}_K$ can be progressively enlarged by "connecting" one element $\mathbf{v}_j$ at a time, where $\mathbf{v}_j$ is taken in a neighboring system. This extension can progress as long as no cycle is closed on $\mathcal{G}^c$.

Proof. As for theorem 4, define $\bar{\mathcal{S}}_i = \mathcal{S}_i \wedge (\bigwedge_{l \in \mathbf{N}(i) \setminus J} \mathcal{M}_{l,i})$, $1 \leq i \leq N$, where the $\mathcal{M}_{l,i}$ are stationary messages of $\mathbf{A}_1$. Running $\mathbf{A}_1$ on the tree $\mathcal{G}^c{}_{|J}$ for components $\bar{\mathcal{S}}_j$ yields the same $\mathcal{S}'_j$, so one readily has $\bar{\mathcal{S}}_J = \mathcal{S}'_J$ by lemma 2. But lemma 6 also applies on $\mathcal{G}^c{}_{|J}$ and gives $\mathcal{S}'_K = \Pi_{\mathcal{V}_K}(\bar{\mathcal{S}}_J)$ whence $\mathcal{S}'_K = \Pi_{\mathcal{V}_K}(\mathcal{S}'_J)$. Similar arguments prove the result for $\mathbf{A}_2$. □

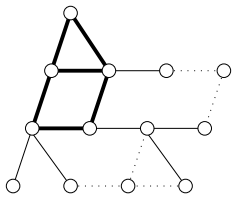


Figure 11: *A graph $\mathcal{G}^c$ (all edges) with subgraph $\mathcal{G}^c{}_{|K}$ (thick edges) and $\mathcal{G}^c{}_{|J}$ (solid edges).*

This result can be further refined : any state of an $\mathcal{S}'_K$ can be extended to branches outside K, even if $\mathcal{G}^c{}_{|K}$ is not a tree. Notice however that $\mathcal{S}'_K$ may very well be empty if it captures cycles (see fig. 8 with $K = \{1, \ldots, 4\}$ for an example) : for $k \in K$, $\Pi_{\mathcal{V}_k}(\mathcal{S}'_K) = \mathcal{S}'_k$ is not guaranteed any more if $\mathcal{G}^c{}_{|K}$ is not a tree.

Theorem 7 *Under assumptions of lemma 12, select $K \subseteq J \subseteq \{1, \ldots, N\}$ such that $\mathcal{G}^c{}_{|K}$ is a connected subgraph of $\mathcal{G}^c$, and all cycles of $\mathcal{G}^c{}_{|J}$ are contained in $\mathcal{G}^c{}_{|K}$. Then $\mathcal{S}'_K = \Pi_{\mathcal{V}_K}(\mathcal{S}'_J)$ for $\mathbf{A}_1$, and $\tilde{\mathcal{S}}'_K = \Pi_{\mathcal{V}_K}(\tilde{\mathcal{S}}'_J)$ for $\mathbf{A}_2$.*

Proof. As for lemma 12, running $\mathbf{A}_1$ on $\mathcal{G}^c{}_{|J}$ with components $\bar{\mathcal{S}}_j$ yields the same stationary systems $\mathcal{S}'_j$, so we can restrict ourselves to $\mathcal{G}^c{}_{|J}$, or equivalently assume $J = \{1, \ldots, N\}$. For simplicity, assume vertices of $L = J \setminus K$ form a single branch, connected to $\mathcal{G}^c{}_{|K}$ by the edge (l, k). Then

$$\Pi_{\mathcal{V}_K}(\mathcal{S}'_J) = \Pi_{\mathcal{V}_K}(\mathcal{S}'_K \wedge \mathcal{S}'_L) = \mathcal{S}'_K \wedge \Pi_{\mathcal{V}_K}(\mathcal{S}'_L) = \mathcal{S}'_K \wedge \Pi_{\mathcal{V}_k}(\mathcal{S}'_L) \tag{46}$$

Applying lemma 2 on the tree $\mathcal{G}^c{}_{|L}$, one has $\mathcal{S}'_L = \bar{\mathcal{S}}_L \wedge \mathcal{M}_{k,l}$, whence $\Pi_{\mathcal{V}_k}(\mathcal{S}'_L) = \Pi_{\mathcal{V}_k}(\bar{\mathcal{S}}_L) \wedge \mathcal{M}_{k,l} = \mathcal{M}_{l,k} \wedge \mathcal{M}_{k,l}$. But both $\mathcal{M}_{l,k}$ and $\mathcal{M}_{k,l}$ are absorbed by $\mathcal{S}'_k$, so also by $\mathcal{S}'_K$, and finally $\Pi_{\mathcal{V}_K}(\mathcal{S}'_J) = \mathcal{S}'_K$. The case of several branches connected to $\mathcal{G}^c{}_{|K}$ is proved similarly.

Again, a similar reasoning yields the result for $\mathbf{A}_2$. □

4.4.4 Progressive reduction

The following result extends lemma 5 to any graph structure.

Theorem 8 *In an involutive setting, whatever the structure of $\mathcal{G}^c$ for $\mathcal{S} = \mathcal{S}_1 \wedge \ldots \wedge \mathcal{S}_N$, algorithm $\mathbf{A}_1$ preserves $\mathcal{S} = \mathcal{S}'_1 \wedge \ldots \wedge \mathcal{S}'_N$ at any time, for $\mathcal{S}'_i$ defined by (18), and similarly algorithm $\mathbf{A}_2$ preserves $\mathcal{S} = \tilde{\mathcal{S}}'_1 \wedge \ldots \wedge \tilde{\mathcal{S}}'_N$ at any time, for $\tilde{\mathcal{S}}'_i$ defined by (24).*

Proof. Consider $\mathbf{A}_1$. The proof differs from lemma 5 since there is no closed characterization of messages at every step. We rather proceed by recursion and show that messages $\mathcal{M}_{i,j}$ are still absorbed by $\mathcal{S}_1 \wedge \ldots \wedge \mathcal{S}_N$ at any time, which induces $\mathcal{S} = \mathcal{S}'_1 \wedge \ldots \wedge \mathcal{S}'_N$. This holds at initialization since $\mathcal{M}_{i,j} = \mathbb{I}$, so assume it holds at some point in $\mathbf{A}_1$. The updated message $\mathcal{M}^{new}_{i,j} = \Pi_{\mathcal{V}_i \cap \mathcal{V}_j}[\mathcal{S}_i \wedge (\bigwedge_{k \in \mathbf{N}(i) \setminus j} \mathcal{M}_{k,i})]$ is absorbed by $\mathcal{S}_i \wedge (\bigwedge_{k \in \mathbf{N}(i) \setminus j} \mathcal{M}_{k,i})$ which is itself absorbed by $\mathcal{S}_1 \wedge \ldots \wedge \mathcal{S}_N$ using the recursion assumption. A similar proof yields the result for $\mathbf{A}_2$. □

This result means that $\mathbf{A}_1$ and $\mathbf{A}_2$ progressively reduce components $\mathcal{S}_i$, borrowing constraints from their neighbors, but do not introduce "wrong" constraints. At convergence of the algorithm (if achieved), one gets partially reduced components, where

constraints specifically due to cycles have not been considered to discard states. This can be evidenced if a partial order $\sqsubseteq$ is available on systems (for example the one defined in lemma 10) : the true reduced system satisfies $\Pi_{\mathcal{V}_i}(\mathcal{S}) = \Pi_{\mathcal{V}_i}(\mathcal{S}'_1 \wedge \ldots \wedge \mathcal{S}'_N) = \mathcal{S}'_i \wedge \Pi_{\mathcal{V}_i}(\bigwedge_{j \neq i} \mathcal{S}'_j) \sqsubseteq \mathcal{S}'_i \sqsubseteq \mathcal{S}_i$.

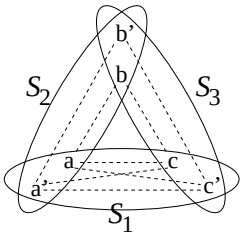


Figure 12: *Reduced components are not a stationary point of* $\mathbf{A}_2$.

Remark. The true reduced components $\mathcal{S}''_i \triangleq \Pi_{\mathcal{V}_i}(\mathcal{S})$ are "contained" in the partially reduced components given by $\mathbf{A}_1$ or $\mathbf{A}_2$, but in general do not themselves define a stationary point of $\mathbf{A}_2$, nor of [13] $\mathbf{A}_1$. Consider the following counter-example (displayed fig. 12). We take $\mathcal{V}_{max} = \{A, B, C\}$ and $\mathcal{S}_1 = \{(a,c),(a,c'),(a',c),(a',c')\}$, $\mathcal{S}_2 = \{(a,b),(a',b')\}$, $\mathcal{S}_3 = \{(b,c),(b',c')\}$, so $\mathcal{S} = \mathcal{S}_1 \wedge \mathcal{S}_2 \wedge \mathcal{S}_3 = \{(a,b,c),(a',b',c')\}$. Reduced components satisfy $\mathcal{S}''_1 = \{(a,c),(a',c')\}$ and $\mathcal{S}''_2 = \mathcal{S}_2$, $\mathcal{S}''_3 = \mathcal{S}_3$. If stationarity holds for $\mathbf{A}_2$, we must have $\tilde{\mathcal{M}}_{i,j} = \tilde{\mathcal{M}}_{j,i} = \Pi_{\mathcal{V}_i \cap \mathcal{V}_j}(\mathcal{S}''_i)$, and check that $\mathcal{S}''_i = \mathcal{S}_i \wedge (\bigwedge_{k \in \mathbf{N}(i)} \tilde{\mathcal{M}}_{k,i})$. But one has $\tilde{\mathcal{M}}_{2,1} = \{a, a'\}$ and $\tilde{\mathcal{M}}_{3,1} = \{c, c'\}$, so $\mathcal{S}_1 \wedge \tilde{\mathcal{M}}_{2,1} \wedge \tilde{\mathcal{M}}_{3,1} = \mathcal{S}_1 \neq \mathcal{S}''_1$.

5 Conclusion

We have proposed an algebraic framework allowing the derivation of turbo like reduction algorithms for a general family of modular systems. Its main advantage is to rely on very few ingredients : two simple operations on systems, composition and reduction, and on a small set of axioms on these operations. Despite this parsimonious setting, many properties of turbo algorithms can be obtained. In particular, local extendibility and local optimality properties of solutions obtained at a stationary point of a turbo procedure appear as "algebraic" in nature. Convergence results, on the contrary, would require extra material in order to define an appropriate topology on systems. In this framework, involutive systems form a notable exception. This strong property allows to prove convergence of turbo algorithms, and uniqueness of

[13] This is more difficult to prove since one has to construct messages $\mathcal{M}_{i,j}$, but the result holds in an involutive setting thanks to theorem 5.

the limit, with a very simple form of topology. Further, it also provides insensitivity to the choice of the graph on which the turbo algorithm is run. The class of involutive systems may essentially contain constraint systems, however, but it remains interesting as a "benchmark" for properties of turbo algorithms. Finally, let us mention that results presented here have a natural extension to dynamic systems. Specifically, one can replace static components $\mathcal{S}_i$ by components changing in time, or even by automata. In this setting, turbo procedures naturally become distributed on-line algorithms, and allow to address problems like distributed state estimation for example [15, 20].

Acknowledgement : the author would like to thank Aline Roumy and Albert Benveniste for fruitful discussions that helped improving early versions of the manuscript.

A Property of systems living on a tree

We prove the following result, mentioned at the end of section 3.1 :

Theorem 9 *If a compound system $\mathcal{S}$ has at least one communication graph which is a tree, then* all *its communication graphs are trees.*

The proof relies on the following lemmas.

Lemma 13 *Let $\mathcal{G}^{cnx}$ be the connectivity graph of $\mathcal{S}$; if $\mathcal{G}^{cnx}$ has a single connected component, every communication graph $\mathcal{G}^c$ for $\mathcal{S}$ contains at least a spanning tree of $\mathcal{G}^{cnx}$.*

Proof. This is obviously due to the fact that an edge can be removed only if it belongs to a cycle, by definition of redundancy. As a consequence, if one gets a tree in an erosion scheme of $\mathcal{G}^{cnx}$, then the process terminates and this tree *is* a communication graph. □

Lemma 14 *In the process of building a communication graph $\mathcal{G}^c$ for $\mathcal{S}$, the redundant edges recursively taken away from the connectivity graph $\mathcal{G}^{cnx}$ can be removed in any order.*

Proof. It is enough to prove that one can reverse the order in which two redundant edges (i, j) and (k, l) are removed: permutations of two successive elements in a sequence generate all possible permutations of that sequence.

Assume (i, j) and (k, l) are removed in this order, which corresponds to the sequence of graphs $\mathcal{G}_0, \mathcal{G}_1, \mathcal{G}_2$. (k, l) is obviously redundant on $\mathcal{G}_0$, and could be removed first. So we only have to show that (i, j) is still redundant once (k, l) has

been removed from $\mathcal{G}_0$, which defines graph $\mathcal{G}'_1$. There is a redundancy path for (k,l) on $\mathcal{G}_1$ (and $\mathcal{G}_2$), which doesn't use edge (i,j) ; we denote a generic element on this path by n. In the same way, there is a redundancy path for (i,j) on $\mathcal{G}_0$ (and $\mathcal{G}_1$), the generic nodes of which are denoted by m. If this path doesn't use edge (k,l), it is still present on $\mathcal{G}'_1$, and thus (i,j) is redundant on $\mathcal{G}'_1$. Otherwise, the edge (k,l) can be bypassed through vertices n on $\mathcal{G}'_1$, which defines another redundancy path for (i,j) : $\mathcal{V}_i \cap \mathcal{V}_j \subseteq \mathcal{V}_k, \mathcal{V}_l$, so $\mathcal{V}_i \cap \mathcal{V}_j \subseteq \mathcal{V}_k \cap \mathcal{V}_l$ and by assumption $\mathcal{V}_k \cap \mathcal{V}_l \subseteq \mathcal{V}_n$, so $\mathcal{V}_i \cap \mathcal{V}_j \subseteq \mathcal{V}_n$ for every n on the bypass. □

Lemma 15 *Let $\mathcal{G}^c$ be a tree-shaped communication graph for $\mathcal{S}$, and (i,j) be an edge of its connectivity graph $\mathcal{G}^{cnx}$ that is not in $\mathcal{G}^c$. Then the single path linking i to j on $\mathcal{G}^c$ is a redundancy path for edge (i,j).*

Proof. We rely on the previous lemma: there exists an erosion scheme from $\mathcal{G}^{cnx}$ to $\mathcal{G}^c$ in which (i,j) is the last edge removed. On $\mathcal{G}^c \cup \{(i,j)\}$, there is a single cycle, containing (i,j), and (i,j) is redundant. □

Proof of theorem 9. Let ${\mathcal{G}^c}_1$ and ${\mathcal{G}^c}_2$ be two communication graphs for $\mathcal{S}$, such that ${\mathcal{G}^c}_1$ is a tree, and ${\mathcal{G}^c}_2$ has cycles. For the convenience of the reasoning, edges of the tree ${\mathcal{G}^c}_1$ are said to be red, and edges of ${\mathcal{G}^c}_2$ are said to be black. The objective is to prove that there exists a redundant (black) edge on ${\mathcal{G}^c}_2$.

We now distinguish three cases.

Case 1. Since ${\mathcal{G}^c}_2$ has cycles, observe that there is at least one black edge (i,j) which is not red. There exists also a unique red path $(i = k_0, k_1, k_2, \ldots, k_{L-1}, k_L = j)$ from i to j (by lemma 13), which makes edge (i,j) redundant by lemma 15. If all edges of this path are also black, then (i,j) is also redundant on ${\mathcal{G}^c}_2$ and we are done.

Case 2. Otherwise, there exists edges of the red path which are not black (fig. 13). Let (k_l, k_{l+1}) be such an edge. By lemma 14, (k_l, k_{l+1}) could be the last edge removed to get ${\mathcal{G}^c}_2$, so there exists a black path P_l from k_l to k_{l+1} which makes (k_l, k_{l+1}) redundant. Let m be a generic node on this path P_l, one has $\mathcal{V}_{k_l} \cap \mathcal{V}_{k_{l+1}} \subseteq \mathcal{V}_m$, but since $\mathcal{V}_i \cap \mathcal{V}_j \subseteq \mathcal{V}_{k_l}, \mathcal{V}_{k_{l+1}}$ (on the red path), one has also $\mathcal{V}_i \cap \mathcal{V}_j \subseteq \mathcal{V}_{k_l} \cap \mathcal{V}_{k_{l+1}} \subseteq \mathcal{V}_m$. Assume it is possible to select the black paths P_l such that none of them uses edge (i,j). Then, we recursively build a black path from i to j by either keeping edge (k_l, k_{l+1}) if it is black, or replacing it by the bypass P_l. By construction, this black path doesn't use edge (i,j) and defines a redundancy path for (i,j), so we are done.

Case 3. The construction above fails if, for some red edge (k_l, k_{l+1}), all black redundancy paths P_l use the edge (i,j). In that case, we have $\mathcal{V}_{k_l} \cap \mathcal{V}_{k_{l+1}} \subseteq \mathcal{V}_i \cap \mathcal{V}_j$ using black edges, and $\mathcal{V}_i \cap \mathcal{V}_j \subseteq \mathcal{V}_{k_l} \cap \mathcal{V}_{k_{l+1}}$ using red edges, whence $\mathcal{V}_i \cap \mathcal{V}_j =$

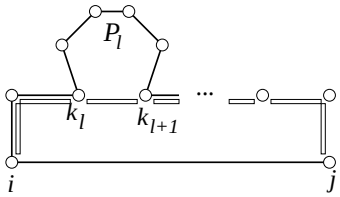


Figure 13: *Red edges are represented as rectangles, black edges as solid lines. A red path from i to j allows to build a black redundancy path for edge (i,j).*

$\mathcal{V}_{k_l} \cap \mathcal{V}_{k_{l+1}}$. So, by adding edge (i,j) to ${\mathcal{G}^c}_1$, one closes a (unique) cycle containing edges (k_l, k_{l+1}) and (i,j), and *both* of these edges are redundant on this extended graph. In other words, one can replace edge (k_l, k_{l+1}) by edge (i,j) in ${\mathcal{G}^c}_1$: this yields a new spanning tree ${\mathcal{G}^{c\prime}}_1$, which is therefore a communication graph. This operation replaces a red edge, by another edge which is both red and black. We can thus go back to the beginning of the proof, with ${\mathcal{G}^c}_1$ replaced by ${\mathcal{G}^{c\prime}}_1$. Since case 3 can only apply a limited number of times, one necessarily discovers a redundant black edge either by case 1 or by case 2. For example, in the worst case, all red and non-black edges of ${\mathcal{G}^c}_1$ are turned into red and black edges, and case 1 necessarily applies. □

References

[1] Y. Weiss, W. T. Freeman, "On the Optimality of Solutions of the Max-Product Belief-Propagation Algorithm in Arbitrary Graphs," IEEE Trans. on Information Theory, vol. 47, no. 2, pp. 723-735, Feb. 2001.

[2] J. S. Yedidia, W. T. Freeman, Y. Weiss, "Bethe free energy, Kikuchi approximations, and belief propagation algorithms," available at www.merl.com/papers/TR2001-16/

[3] J. S. Yedidia, W. T. Freeman, Y. Weiss, "Generalized Belief Propagation," Advances in Neural Information Processing Systems (NIPS), Vol 13, pp. 689-695, December 2000, also as MERL Tech. Rep. no. TR2000-26.

[4] R. J. McEliece, D. J. C. MacKay, J.-F. Cheng,"Turbo Decoding as an Instance of Pearl's Belief Propagation Algorithm," IEEE Journal on Selected Areas in Communication, vol. 16(2), Feb. 1998, pp. 140-152.

[5] S. M. Aji, R. J. McEliece, "The Generalized Distributive Law," IEEE Trans. Inform. Theory, vol. 46, no. 2 (March 2000), pp. 325-343.

[6] S. M. Aji, R. J. McEliece, "The Generalized Distributive Law and Free Energy Minimization," 39th Allerton Conference, October 4, 2001.

[7] S.-Y. Chung, T. Richardson, R. Urbanke, "Analysis of Sum-Product Decoding of Low-Density Parity-Check Codes Using a Gaussian Approximation," IEEE Trans. Inform. Theory, 47 (2001), pp. 657-670.

[8] T. Richardson, R. Urbanke, "The Capacity of Low-Density Parity Check Codes Under Message-Passing Decoding," IEEE Trans. Inform. Theory, 47 (2001), pp. 599-618.

[9] F. R. Kschischang, B. J. Frey, "Iterative Decoding of Compound Codes by Probability Propagation in Graphical Models," IEEE J. on Selected Areas in Communications, vol. 16(2), Feb. 1998, pp. 219-230.

[10] F. R. Kschischang, B. J. Frey, H.-A. Loeliger, "Factor Graphs and the Sum-Product Algorithm," IEEE Transactions on Information Theory, vol. 47(2), Feb. 2001, pp. 498-519.

[11] J. Pearl, "Fusion, Propagation, and Structuring in Belief Networks," Artificial Intelligence, vol. 29, pp. 241-288, 1986.

[12] S. L. Lauritzen, D. J. Spiegelhalter, "Local computations with probabilities on graphical structures and their application to expert systems," J. Royal Statistical Society, Series B, vol. 50(2), pp. 157-224, 1988.

[13] S. L. Lauritzen, *Graphical Models*, Oxford Statistical Science Series 17, Oxford University Press, 1996.

[14] E. Fabre, "Compositional Models of Distributed and Asynchronous Dynamical Systems," 41st Conf. on Decision and Control, Las Vegas, Dec. 2002, pp. 1-6.

[15] E. Fabre, V. Pigourier, "Monitoring distributed systems with distributed algorithms," 41st Conf. on Decision and Control, Las Vegas, Dec. 2002, pp. 411-416.

[16] E. Fabre, A. Benveniste, C. Jard, "Distributed diagnosis for large discrete event dynamic systems," in Proc. of the IFAC congress, July 2002.

[17] E. Fabre, "Distributed Diagnosis for Large Discrete Event Dynamic Systems," in preparation.

[18] E. Fabre, "Factorization of Unfoldings for Distributed Tile Systems," INRIA research report no. 4829, May 2003.

[19] A. Benveniste, E. Fabre, C. Jard, S. Haar, "Diagnosis of asynchronous discrete event systems, a net unfolding approach," IEEE Trans. on Automatic Control, vol. 48(5), May 2003. Preliminary version available at www.irisa.fr/sigma2/benveniste/pub/IEEE_TAC_AsDiag_2003.html

[20] A. Benveniste , S. Haar , E. Fabre , C. Jard, "Distributed monitoring of concurrent and asynchronous systems," (extended version) IRISA research report no. 1540, available at www.irisa.fr/bibli/publi/pi/2003/1540/1540.html

Unité de recherche INRIA Lorraine, Technopôle de Nancy-Brabois, Campus scientifique,
615 rue du Jardin Botanique, BP 101, 54600 VILLERS LÈS NANCY
Unité de recherche INRIA Rennes, Irisa, Campus universitaire de Beaulieu, 35042 RENNES Cedex
Unité de recherche INRIA Rhône-Alpes, 655, avenue de l'Europe, 38330 MONTBONNOT ST MARTIN
Unité de recherche INRIA Rocquencourt, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex
Unité de recherche INRIA Sophia-Antipolis, 2004 route des Lucioles, BP 93, 06902 SOPHIA-ANTIPOLIS Cedex

Éditeur
INRIA, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex (France)
http://www.inria.fr
ISSN 0249-6399