42395

Les Sélections
Techniques de l'Ingénieur

ÉLECTRONIQUE - AUTOMATIQUE

EXTRAIT GRATUIT

# Automatique séquentielle

Réf. Internet : 42395

*Actualisation permanente sur*
**www.techniques-ingenieur.fr**

Techniques de l'Ingénieur

Automatique séquentielle

Ti 660

Techniques de l'Ingénieur

# Les Sélections Techniques de l'Ingénieur

La plus importante base scientifique et technique en français

## Une information fiable, claire et actualisée

Validés par un comité scientifique et mis à jour en permanence sur Internet, les articles des Techniques de l'Ingénieur s'adressent à tous les ingénieurs et scientifiques, en poste ou en formation.
Outil d'accompagnement de la formation et de la carrière des ingénieurs, les bases documentaires Techniques de l'Ingénieur constituent le socle commun de connaissances des acteurs de la recherche et de l'industrie.

## Les meilleurs experts scientifiques

Plus de 150 conseillers scientifiques et 3 000 auteurs, industriels, chercheurs, professeurs collaborent aux bases documentaires qui font aujourd'hui de Techniques de l'Ingénieur l'éditeur scientifique et technique de référence.
Les meilleurs spécialistes sont réunis pour constituer une base de connaissances techniques et scientifiques inégalée, vous former et vous accompagner dans vos projets.

## Une collection 100 % en ligne

- Accessibles sur www.techniques-ingenieur.fr, les dernières nouveautés et actualisations de votre base documentaire
- Les articles téléchargeables en version PDF

## Des services associés

Rendez-vous sur votre espace « Mon compte » en ligne pour retrouver la liste de vos services associés aux abonnements et y accéder.

Pour toute information, le service clientèle reste à votre disposition :
Tél : 01 53 35 20 20
Fax : 01 53 26 79 18
Mail : infos.clients@teching.com

Cet ouvrage fait partie du pack **Ingénierie des systèmes et robotique** (Réf. Internet ti660) composé des bases documentaires suivantes :

| | |
|---|---|
| **Modélisation et analyse de systèmes asservis** | Réf. Internet : 42391 |
| **Régulation et commande des systèmes asservis** | Réf. Internet : 42394 |
| **Automatique avancée** | Réf. Internet : 42393 |
| **Automatique séquentielle** | Réf. Internet : 42395 |
| **Supervision des systèmes industriels** | Réf. Internet : 42396 |
| **Systèmes d'information et de communication** | Réf. Internet : 42397 |
| **Robotique** | Réf. Internet : 42398 |

**Sur www.techniques-ingenieur.fr**

- Saisissez la référence Internet pour accéder directement aux contenus en ligne
- Retrouvez la liste complète des bases documentaires

Les Sélections
Techniques de l'Ingénieur

Cette base documentaire fait partie du pack **Ingénierie des systèmes et robotique** (Réf. Internet ti660) dont les experts scientifiques sont :

Retrouvez dans cette base documentaire les contributions de :

**Michel BERTRAND**
*Pour l'article :* S8015

**Michel COMBACAU**
*Pour les articles :* S7572 – S7573

**Isabel DEMONGODIN**
*Pour l'article :* S7252

**Jean-Jacques DUMÉRY**
*Pour les articles :* S7240 – S7241

**Philippe ESTEBAN**
*Pour les articles :* S7572 – S7573

**Christophe HARO**
*Pour l'article :* S7254

**Catherine K. BOUNSAYTHIP**
*Pour l'article :* S7212

**Pierre LADET**
*Pour l'article :* S7252

**Claude LEMARÉCHAL**
*Pour l'article :* S7210

**Salah MAOUCHE**
*Pour l'article :* S7212

**Alexandre NKETSA**
*Pour les articles :* S7572 – S7573

**Ammar OULAMARA**
*Pour l'article :* S7211

**Eldad PERELSTEIN**
*Pour l'article :* S7214

**Marie-Claude PORTMANN**
*Pour l'article :* S7211

**Pascal RICHARD**
*Pour l'article :* S7254

**Gilles ROUSSEL**
*Pour l'article :* S7212

**Igal M. SHOHET**
*Pour l'article :* S7214

Les Sélections
Techniques de l'Ingénieur

## Automatique séquentielle

Réf. Internet 42395

# SOMMAIRE

Référence Internet
S7572

# Commandes à réseaux de Petri

## Modélisation

par **Michel COMBACAU**
*Professeur, université Paul-Sabatier (Toulouse-III), Laboratoire d'analyse et d'architecture des systèmes (LAAS-CNRS)*

**Philippe ESTEBAN**
*Maître de conférences, université Paul-Sabatier (Toulouse-III), LAAS-CNRS*

*et* **Alexandre NKETSA**
*Professeur, université Paul-Sabatier (Toulouse-III), LAAS-CNRS*



*Tous les automatismes logiques ont un comportement pouvant faire l'objet d'une modélisation par réseau de Petri. Cet outil est en effet orienté vers la représentation de systèmes à événements discrets dont les variables d'état évoluent brusquement d'une valeur à l'autre sans qu'il soit nécessaire de représenter les phénomènes transitoires. Entrent dans cette classe, les systèmes de commande comportant ou non des évolutions simultanées, les procédés ou systèmes commandés par modèles à événements discrets, les systèmes automatiques (commande et procédé). Le mode d'évolution asynchrone des réseaux de Petri en fait le modèle par excellence des applications réparties et bien sûr des protocoles de communication. Enfin, ils sont très utilisés pour l'évaluation de performance par simulation ou par calcul formel sur des extensions du modèle comportant des données statistiques ou stochastiques.*

*La capacité de description du modèle par réseau de Petri trouve ses limites lorsque des dates absolues apparaissent dans le cahier des charges. En effet, le temps n'est pas pris en compte explicitement par ce modèle. Il est donc difficile de représenter des applications contenant des recommandations telles que « à la date t, ... ». En revanche, la prise en compte de durée est tout à fait possible grâce aux extensions temporelles du modèle de base. Il est donc tout à fait naturel d'envisager une représentation par réseau de Petri d'un cahier des charges contenant : « après une attente de 10 secondes, ... ». La distinction n'est pas toujours immédiate*

*mais découle d'une référence à un temps absolu (date) ou à un temps relatif (durée). Une autre limitation importante du modèle à réseau de Petri concerne la non-existence de mécanisme d'abstraction. Hormis l'utilisation du concept d'activité décrit dans cet article et qui n'est pas spécifique à ce modèle, aucun mécanisme d'agrégation n'est véritablement proposé par les réseaux de Petri.*

*Bien sûr, d'autres outils de modélisation peuvent être utilisés pour la modélisation à événements discrets. Par exemple, les automates à états finis permettent de capturer sans difficulté le comportement séquentiel d'une application. En revanche, dès que des évolutions simultanées doivent être représentées, surtout si elles doivent être synchronisées de temps à autre, la représentation par graphe d'état devient complexe (on parle d'explosion combinatoire du nombre d'états) et surtout le comportement global du modèle devient difficile à appréhender. Le Grafcet (graphe fonctionnel de commande étape transition, langage normalisé pour les API) peut être utilisé en lieu et place des réseaux de Petri. Il convient de savoir que le Grafcet, qui a été défini par un groupe de travail de l'Adepa dans les années 1980, est en fait un dérivé des réseaux de Petri binaires. Le Grafcet peut être préféré pour des questions de mise en œuvre. En revanche, il devient rapidement difficile à utiliser quand le besoin de comptage (de pièces, de messages, etc.) se fait sentir et l'on aura alors plutôt recours aux réseaux de Petri. Signalons enfin que la règle du Grafcet stipulant que « toutes les transitions franchissables sont franchies simultanément » introduite pour le rendre déterministe transforme, en cas d'erreur de modélisation, un choix en départ en parallèle. Ce problème n'existe pas avec les réseaux de Petri. Une extension des automates à états finis, les statecharts, constitue une alternative à la modélisation des systèmes à événements discrets. Ils intègrent l'abstraction, l'orthogonalité pour le parallélisme et la diffusion pour les communications. Le problème de la validation globale d'une application n'est toutefois pas résolu. À noter que les statecharts permettent de modéliser simplement les mécanismes de préemption sur un groupe d'états et comportent un mécanisme d'historique permettant de replacer le modèle dans le dernier état qui avait été atteint avant une préemption. Les statecharts sont à considérer pour des applications comportant ces mécanismes, mais il faut savoir que la mise en œuvre d'un statechart est un problème non résolu à l'heure actuelle.*

*En résumé, pour une application à caractère distribué, les réseaux de Petri, de par leur fonctionnement asynchrone, s'imposeront assez naturellement, sauf si les traitements de données constituent l'essentiel des opérations à modéliser. En effet, l'interaction avec des données est bien prise en compte par le modèle « réseau de Petri à objets », mais si la structure de contrôle est quasi inexistante, alors elle constituera plutôt une gêne qu'un avantage. Il faut s'orienter dans ces conditions vers des modèles basés sur les flux de données (SADT, SA-RT). Un critère important qui peut conduire au choix des réseaux de Petri est le besoin de pouvoir établir des preuves formelles de propriétés (vivacité ou atteignabilité par exemple) devant absolument être satisfaites par l'application. La représentation formelle est alors un support de la plus grande importance. La possibilité d'effectuer des simulations permet de mesurer avec des données statistiques ou stochastiques les performances que l'on peut espérer du système. Cela est particulièrement utile dans les phases de conception lorsque des choix d'architecture ou de principe doivent être faits. Enfin, la représentation graphique d'un modèle et les règles de fonctionnement simples qui en sont caractéristiques en font un outil privilégié dans le domaine de la didactique. N'importe qui, possédant un niveau scientifique de début d'études supérieures, peut rapidement comprendre le fonctionnement d'un modèle s'il lui est présenté de façon pragmatique. Cet outil peut donc devenir un élément essentiel de communication avec un client, un bureau d'études, etc.*

*L'utilisation de l'outil « réseaux de Petri » pour assurer la commande de systèmes à événements discrets se décline en fait en deux étapes principales : modélisation de la commande à réaliser et analyse du modèle développé, puis mise en œuvre effective de la commande modélisée. Ces deux étapes font chacune l'objet d'un article : ce premier article concerne l'étape « modélisation, analyse », le second [S 7 573] traitant de l'étape de « mise en œuvre ». Un exemple d'application y est également donné.*

# 1. Élaboration du modèle

La théorie des réseaux de Petri peut être consultée dans de nombreux ouvrages [1] [2] [3] [4] [5] [R 7 252] [S 7 254]. Nous ne rappellerons pas ici les éléments graphiques et la représentation formelle de ce modèle mais nous nous focaliserons uniquement sur son utilisation en qualité d'outil de description d'une application exprimée, par exemple, sous forme d'un cahier des charges textuel.

Dresser un modèle à réseau de Petri consiste à choisir un ensemble d'entités, un ensemble d'états de ces entités et à traduire les lois de fonctionnement de l'application considérée.

Un réseau de Petri est constitué d'une structure de contrôle représentant les règles de fonctionnement et donnant une vision statique de l'application modélisée et du marquage, distribution de marques (ou jetons) dans les places qui donnent l'état courant du modèle et permettent donc de suivre ses évolutions. Avant d'aborder les spécificités et les invariants de la modélisation par réseau de Petri, il n'est pas inutile d'énoncer clairement le rôle tenu par chacun des éléments que nous venons d'évoquer.

## 1.1 Rôle des éléments de la structure de contrôle

■ Les **places** d'un réseau de Petri doivent être vues comme une extension du concept d'état des systèmes séquentiels. Cette extension a deux facettes :

— la signification associée à une place est locale ou partielle. Par opposition, dans un graphe d'états, chaque état correspond à une situation précise de l'ensemble du système modélisé. Dans un réseau de Petri au contraire, l'information d'état associée à une place est limitée au sous-ensemble d'éléments choisis par le concepteur. Il est courant de rencontrer une place représentant l'état de repos d'une ressource indépendamment de l'état du reste du système. Seule la vision de l'ensemble des états partiels (des places) donne une vision globale du système modélisé ;

— l'état d'une place d'un réseau de Petri n'est pas astreint à prendre une valeur booléenne. Le marquage d'une place est représenté par un entier naturel. Une place associée à la signification « outils disponibles » peut donc contenir autant de marques qu'il y a effectivement d'outils disponibles.

En tout état de cause, les places servent à modéliser des états partiels qui peuvent être pris par les entités modélisées dans le réseau.

■ Le nom des **transitions** est explicite et véhicule bien le concept de

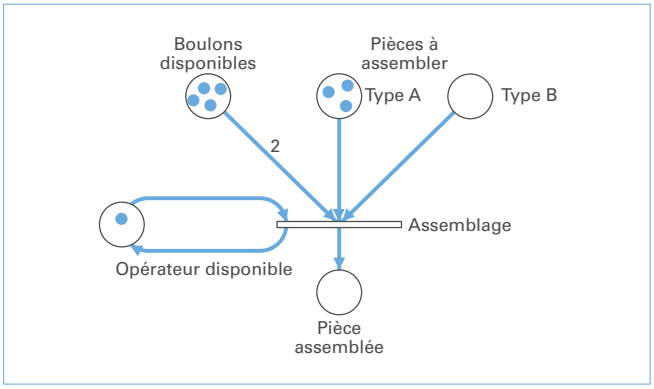



**Figure 1 – Assemblage de deux pièces par un opérateur**

être distingués dans une application. Cette partition s'appuie sur le caractère réversible ou non des changements d'état subis par l'entité considérée au sein du modèle que nous nous proposons d'établir.

Les entités qui subissent des changements d'état réversibles sont appelées des ressources et ne disparaissent jamais du modèle.

Les entités subissant des changements d'état définitifs sont appelées des produits et sont amenées à apparaître et à circuler dans le modèle pour enfin en disparaître. Généralement, tout le but de l'application qui est modélisée consiste à « faire circuler les produits » comme l'impose le cahier des charges. Toutes les évolutions ne mettant pas en jeu un produit sont des évolutions nécessaires pour le bon fonctionnement des ressources, mais sans valeur réelle pour l'application considérée.

Cette distinction ressource/produit n'est pas nécessaire pour établir un modèle à réseau de Petri, mais concourt toutefois à une clarification du rôle des entités gérées par le modèle. Cette partition est une manière d'aborder une des questions fondamentales devant tout problème de modélisation : que doit-on modéliser ?

### 1.1.1 Exemple d'application

Pour illustrer les différents concepts évoqués jusqu'ici, prenons l'exemple d'un **processus d'assemblage de deux pièces par un opérateur à l'aide d'un boulon**. La figure **1** donne le modèle à réseau de Petri correspondant.

Référence Internet
S7573

# Commandes à réseaux de Petri

## Mise en œuvre et application

par **Michel COMBACAU**
*Professeur, université Paul-Sabatier (Toulouse-III), Laboratoire d'analyse et d'architecture des systèmes (LAAS-CNRS)*

**Philippe ESTEBAN**
*Maître de conférences, université Paul-Sabatier (Toulouse-III), LAAS-CNRS*

et **Alexandre NKETSA**
*Professeur, université Paul-Sabatier (Toulouse-III), LAAS-CNRS*



*Les techniques de modélisation ont conduit à l'obtention d'un modèle du comportement de l'application définie par son cahier des charges [1] [3] [4] [5] [S 7 572] [R 7 252] [S 7 254]. La vérification de ce modèle permet de constater (ou non) que son comportement ne trahit pas celui attendu et le décrit correctement [4] [10] [S 7 572]. Des retours sur la modélisation ont peut-être été nécessaires, afin d'ajuster la représentation de l'application et de suivre au plus près le fonctionnement requis. Ces étapes sont détaillées dans [S 7 572].*

*L'étape de mise en œuvre donne une réalité au modèle résultant de tout ce travail de conception et d'analyse : la commande du procédé peut enfin être envisagée. Pour terminer, un exemple d'application est présenté à titre d'illustration.*

# 1. Mise en œuvre

## 1.1 Transformation du modèle

L'application modélisée et vérifiée répond aux attentes, mais sa mise en œuvre peut requérir des modifications. En effet, il suffit de considérer, par exemple, que les activités de base retenues dans la modélisation sont physiquement réparties sur plusieurs matériels distincts. Une mise en œuvre distribuée du réseau de Petri est alors nécessaire pour tenir compte de cette réalité.

La transformation du modèle obtenu peut être envisagée sous plusieurs aspects.

### 1.1.1 Décomposition fonctionnelle du modèle

Le modèle est une représentation du fonctionnement attendu. Dans ce sens, il exprime des fonctions du système, définies explicitement dans le cahier des charges, ou bien conséquentes de la juxtaposition de contraintes élémentaires.

L'utilisateur/concepteur peut décider de découper le modèle afin de mettre en exergue les fonctions qu'il a identifiées, comme par exemple « gestion de la ressource partagée », « évacuation des produits ».

Il décide ainsi de façon pragmatique d'un découpage fonctionnel modulaire regroupant au sein d'un même module les places et transitions du réseau nécessaires à la représentation d'une fonction. Son expérience lui sert de support pour mener la décomposition, et la vision graphique du réseau de Petri est un bon soutien pour isoler ces comportements, dans la mesure où une vision globale est envisageable. Pour cela, le réseau de Petri doit conserver une dimension « lisible », comme cela a été proposé dans [S 7 572] (§ 1.1.3), le format A4 étant une bonne base pour garantir cette lisibilité.

Par ailleurs, le découpage du modèle peut être abordé avec un point de vue « matériel » : la priorité est donnée ici aux matériels du système modélisé. La mise en œuvre du réseau de Petri a pour objectif d'effectuer la commande de ces systèmes matériels, et il est intéressant d'envisager de découper le modèle en les faisant apparaître explicitement.

Dans les deux cas considérés, l'objectif du découpage est de faciliter la mise en œuvre du réseau de Petri. Pour cela, le concepteur doit s'attacher à décomposer le réseau de Petri en effectuant un découpage par rapport aux places. En effet, une place est la représentation d'un état partiel du système, qui reste inchangé tant qu'une évolution n'est pas envisagée. Il est très facile de matérialiser cet état partiel par une bascule ou un compteur pour une mise en œuvre matérielle, ou par une donnée booléenne ou numérique pour une mise en œuvre logicielle.

Un découpage par rapport aux transitions est aussi envisageable, mais il est beaucoup plus délicat de garantir que le fonctionnement global du système est bien respecté. Cela suppose que les éléments séparés par le découpage et liés à une même transition la voient simultanément évoluer. Une transition est la représentation d'un changement d'état, et son caractère événementiel, furtif par nature, en rend la matérialisation difficile. En fait, assez rapidement, la prise en compte de l'évolution de la transition partagée conduit à considérer l'état du marquage de ses places en amont, rejoignant de fait la vision du découpage par rapport aux places.

### 1.1.2 Décomposition systématique en machine à états finis

Le découpage fonctionnel aboutit à isoler des sous-réseaux ayant les caractéristiques générales des réseaux de Petri, pouvant par exemple conserver les notions de parallélisme.

La décomposition peut être abordée avec comme objectif une mise en œuvre par moniteur multitâche. Dans ce cas, chaque élément découpé doit répondre aux caractéristiques définies pour les machines à états finis, et le parallélisme d'états partiels est alors proscrit.

■ **Décomposition manuelle** : la vision que le concepteur a de son modèle lui permet d'envisager la décomposition du réseau de Petri en s'appuyant sur les fonctions mises en jeu ou par rapport aux matériels commandés. Cette décomposition peut être menée de façon manuelle, l'avantage étant de laisser le concepteur associer les fonctions qu'il isole à des machines à états. Mais cette facilité induit aussi un inconvénient : il n'est pas toujours possible de décomposer un réseau de Petri en machines à états communicantes, et seule l'expérience du concepteur peut lui servir pour établir cette limite.

■ **Décomposition automatique** : l'un des avantages des réseaux de Petri est de permettre des opérations mathématiques de transformation du modèle. L'une de ces opérations concerne la recherche des invariants de places. Un invariant de place monomarqué est une machine à états et l'ensemble des invariants de places couvre le réseau de Petri.

L'inconvénient de cette décomposition est que le lien avec les fonctions du système initial n'est plus aussi évident. Cependant, tous les éléments d'une mise en œuvre même distribuée sont fournis par cette décomposition.

## 1.2 Mise en œuvre logicielle

### 1.2.1 En langage structuré

La mise en œuvre logicielle par langage structuré d'un réseau de Petri s'appuie sur les principes généraux des langages de programmation. L'objectif est d'obtenir une application (un code exécutable) réalisant la commande du procédé.

Cette application intègre la description complète du réseau de Petri ordinaire : structure de contrôle (places et transitions) et structure de données (prédicats et données). Elle décrit le réseau de Petri : toute modification du réseau de Petri se répercute par une modification du code qui en assure la description, conduisant ainsi à disposer d'autant d'applications exécutables que de réseaux de Petri différents. Il faut donc disposer de toute la chaîne de développement associée au langage de programmation utilisé. *A contrario*, le joueur de réseau de Petri est une application unique et générale, le réseau de Petri étant décrit dans une structure de données externe à l'application (§ 1.2.3).

L'application dispose en outre des moyens d'accès aux capteurs et actionneurs présents sur le procédé à commander : cela suppose qu'il existe une liaison matérielle entre les ports d'entrée/sortie du calculateur de commande et ce procédé, et un interfaçage logiciel entre ces ports et l'application logicielle mise en œuvre.

Nous supposons ainsi que des fonctions entree() et sortie() sont disponibles avec le principe de fonctionnement suivant.

int entree (int numero)
La fonction renvoie la valeur vraie ou fausse du bit référencé par numero sur les ports d'entrée (par exemple, pour un PPI (*parallel programmable interface*), les 16 bits des ports A et B programmés en entrée, avec numero compris entre 0 et 15).
void sortie (int numero, valeur)
La fonction affecte la valeur vraie ou fausse au bit référencé par numero sur les ports de sortie (par exemple, pour un PPI, les 8 bits du port C programmé en sortie, avec numero compris entre 0 et 7).

Dans un souci de lisibilité du programme mettant en œuvre le réseau de Petri, chaque numéro de capteur et d'actionneur peut être associé à une constante numérique ayant comme nom celui du capteur ou de l'actionneur considéré.

#### 1.2.1.1 Réseau de Petri réduit à une machine à états

Lorsque le réseau de Petri est réduit à une machine à états finis (MEF), le principe de mise en œuvre du réseau peut s'appuyer sur le fait que, dans une telle représentation, le réseau de Petri est binaire et qu'une seule place parmi toutes est marquée à la fois (et il y en a toujours une marquée). De plus, il n'existe pas de conflit de tir de transition, car chaque place ayant plusieurs transitions de sortie ne doit permettre l'évolution du réseau que vers une et une seule place suivante.

Fort de ces caractéristiques, nous pouvons déduire que :

— une variable numérique suffit pour décrire le marquage courant du réseau, sa valeur représentant le numéro de la place marquée ;

— l'instruction switch est l'instruction de base pour envisager l'évolution du réseau à partir de la place courante marquée.

**Structure de données**

marquage_courant est un entier, le numéro de la place marquée en cours.

marquage_suivant est un entier, le numéro de la place marquée à venir.

capteur_i est un entier, la valeur lue du capteur n° *i* ;

il y en a autant que de capteurs utilisés dans le réseau de Petri.

actionneur_j est un entier, la valeur calculée pour commander l'actionneur n° *j* ;

il y en a autant que d'actionneurs utilisés dans le réseau de Petri.

**Algorithme**

```
# initialisation
marquage_courant = numéro de la place du marquage initial
marquage_suivant = marquage_courant
        # (pour le cas où le réseau de Petri n'évoluerait pas)
# évolution
répéter
  # lire l'état des capteurs du procédé
  capteur_i = entree (numero_capteur_i)
  ...     autant que de capteurs utilisés
  # envisager l'évolution du réseau de Petri
  cas où marquage_courant vaut :
   Numéro place p :
     début
        si la condition associée à une transition suivante de p est vraie
        alors marquage_suivant = numéro de la place aval de cette transition
        fin_si
        ... autant de si que de transitions suivantes pour cette place p
     fin_début
   ... autant de cas que de places décrivant le réseau de Petri
  fin_cas
  marquage_courant = marquage_suivant
        # (le marquage du réseau de Petri a peut-être évolué)
  # calculer les valeurs des actions
  actionneur_j = (marquage_courant==numero_place_x)
                        ou (marquage_courant==numero_place_y)
                        ou ... pour toutes les places où cette action est décrite
  ... autant que d'actionneurs utilisés
  # activer les actions vers le procédé
  sortie (numero_actionneur_j, actionneur_j)
  ... autant que d'actionneurs utilisés
jusqu'à toujours
```

#### 1.2.1.2 Réseau de Petri décomposé en plusieurs machines à états

Lorsque le réseau de Petri est décomposé en plusieurs machines à états finis (par exemple *M* machines à états), le mécanisme envisagé précédemment reste valable, mais des précisions liées à la présence de plusieurs MEF sont à apporter :

— il faut d'abord dupliquer *M* fois la paire marquage_courant/marquage_suivant ;

— il faut aussi dupliquer *M* fois le mécanisme basé sur le switch ;

— il faut ensuite considérer que ces *M* MEF évoluent simultanément : pour cela, les *M* instructions assurant la mise à jour du marquage courant (marquage_courant=marquage_suivant) sont toutes regroupées après la description de la dernière MEF ;

— il faut enfin être attentif au fait qu'une transition (ou plusieurs) peut être commune à deux MEF. Dans ce cas, la condition logique associée à la transition doit être complétée par un ET logique sur le test du marquage de la place en amont située dans l'autre MEF (raisonnement extensible à plusieurs MEF).

À partir de ces particularités, l'algorithme précédent, adapté, peut être utilisé.

#### 1.2.1.3 Réseau de Petri non décomposé

Un réseau de Petri non décomposé en machines à états est mis en œuvre en s'appuyant sur le fait que le réseau évolue par le biais des transitions franchies. C'est donc à partir des transitions que le réseau est envisagé, en faisant évoluer son marquage lors du tir d'une transition.

Une transition est franchissable si la condition logique associée est vraie ET le marquage de ses places en amont est suffisant. Le ET logique est commutatif et, pour une meilleure efficacité de mise en œuvre, il est effectué en deux étapes : test du marquage puis, s'il est suffisant, test de la condition logique. Si cette dernière est vraie, le franchissement de la transition conduit à faire évoluer le marquage des places en amont et en aval de la transition.

**Structure de données**

marquage est un vecteur d'entiers, un composant par place du réseau et dont la valeur indique le nombre de jetons dans la place pour le marquage courant.

capteur_i est un entier, la valeur lue du capteur n° *i* ;

il y en a autant que de capteurs utilisés dans le réseau de Petri.

actionneur_j est un entier, la valeur calculée pour commander l'actionneur n° *j* ;

il y en a autant que d'actionneurs utilisés dans le réseau de Petri.

**Algorithme**

```
# initialisation
marquage[place p] = valeur du marquage initial de la place p
... autant que de places du réseau de Petri
# évolution
répéter
 # lire l'état des capteurs du procédé
 capteur_i = entree (numero_capteur_i)
 ...   autant que de capteurs utilisés
 # envisager l'évolution du réseau de Petri
 # évaluation du tir de la transition t
 début
  si (marquage[place_amont_x] ⩾ poids de l'arc x_t associé)
        et (marquage[place_amont_y] ⩾ poids de l'arc y_t associé)
        et... pour toutes les places en amont de la transition t
  alors
   si la condition associée à la transition t est vraie
   alors
     marquage[place_amont_x]=marquage[place_amont_x] – poids de l'arc x_t associé
     marquage[place_amont_y]=marquage[place_amont_y] – poids de l'arc y_t associé
     ... pour toutes les places amont de la transition t
     marquage[place_aval_z] = marquage[place_aval_z] + poids de l'arc t_z associé
     ... pour toutes les places aval de la transition t
   fin_si
  fin_si
 fin_début
 ... autant de début que de transitions du réseau de Petri
 # calculer les valeurs des actions
 actionneur_j = (marquage_courant==numero_place_x)
                    ou (marquage_courant==numero_place_y)
                    ou ... pour toutes les places où cette action est décrite
 ... autant que d'actionneurs utilisés
 # activer les actions vers le procédé
 sortie (numero_actionneur_j, actionneur_j)
 ... autant que d'actionneurs utilisés
jusqu'à toujours
```

Il faut remarquer que cette technique de mise en œuvre est applicable aux réseaux de Petri binaires, mais aussi aux réseaux de Petri ordinaires (pondérés) pour lesquels la borne de marquage est

supérieure à 1. Elle est plus lourde à mettre en œuvre que la technique précédente, mais aussi plus systématique et surtout plus générale.

#### 1.2.1.4 Réseau de Petri décomposé en machines à états plus des places pondérées

Le réseau de Petri peut avoir été réduit à une MEF complétée par des places bornées à une valeur supérieure à 1. Ces places ne peuvent pas faire partie d'une machine à états, mais ne peuvent pas non plus être ignorées dans l'algorithme de mise en œuvre.

Il faut compléter les mécanismes considérés aux paragraphes 1.2.1.1 et 1.2.1.2 pour la prise en compte de ces places particulières.

La place pondérée a une transition en aval (ou plusieurs, sur plusieurs MEF) dont elle conditionne par son marquage la sensibilité au tir et une transition en amont (ou plusieurs, sur plusieurs MEF).

L'évolution de son marquage est liée au tir de ses transitions en amont et en aval. Son nouveau marquage est évalué après celui des places des MEF, et avant la mise à jour des marquages courants des MEF : cela permet de profiter des valeurs des données marquage_courant et marquage_suivant des MEF pour déterminer si les transitions associées à la place pondérée ont été franchies.

Ainsi, pour une place pondérée $P_p$ (figure **1**) :

si marquage_courant_MEF1=$P_1$ et marquage_suivant_MEF1=$P_2$ alors $P_p=P_p + x$

si marquage_courant _MEF2=$P_3$ et marquage_suivant_MEF2=$P_4$ alors $P_p=P_p - y$

Pour toutes les transitions en aval de la place pondérée, l'évaluation de leur condition logique doit être complétée (par un ET logique) par le test du marquage de la place pondérée.

Si tous les conflits de tir associés aux transitions en aval de la place pondérée ne sont pas résolus, l'ordre d'évaluation des MEF induit une priorité implicite. Mais le caractère synchrone de la mise en œuvre retenue impose de mémoriser le fait que la place pondérée est censée avoir perdu du poids (son marquage n'évolue effectivement qu'après l'évaluation de toutes les MEF). Pour cela, il faut autant de variables représentant la place pondérée que de transitions en aval, toutes ayant comme valeur le marquage de la place bornée juste avant d'envisager l'évolution du réseau. Avec $n$ transitions en conflit, la description de la première des MEF utilisant la première transition doit diminuer le marquage des $n-1$ autres représentations de la place bornée si la transition est franchissable. De même, la description de la première des MEF utilisant la deuxième transition doit diminuer le marquage des $n-2$ autres représentations de la place bornée, etc. Le premier groupe de MEF

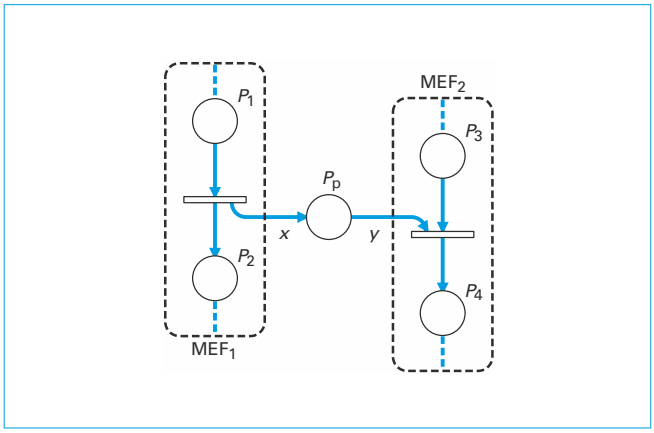



**Figure 1 – Communication entre deux MEF**

### 1.2.2 Par moniteur multitâche

La mise en œuvre par moniteur multitâche permet de contourner élégamment les problèmes de synchronisation rencontrés lors de la mise en œuvre du réseau de Petri après décomposition en machines à états. Nous avons vu (§ 1.2.1.4) qu'un réseau de Petri dont certaines places ne sont pas binaires conduit à une mise en œuvre dans laquelle le traitement des synchronisations n'est pas trivial et doit faire l'objet de la plus grande rigueur. Les moniteurs multitâches fournissent des outils pour créer des tâches correspondant à des processus séquentiels de traitement et des primitives de communication entre les tâches. Après avoir rappelé les principes de base d'un moniteur multitâche, nous montrons comment décomposer un réseau de Petri en processus séquentiels, puis nous donnons une méthode de mise en œuvre systématique. Pour l'exposé, nous utilisons les primitives et la syntaxe de VxWorks, système multitâche temps réel très répandu dans les applications industrielles.

#### 1.2.2.1 Décomposition en processus séquentiels

Un processus séquentiel est constitué par un cycle place-transition monomarqué. La différence avec une machine à états est subtile : dans cette décomposition, aucune transition ne peut être partagée entre deux processus. Les éléments du réseau de Petri qui sont partagés par plusieurs processus sont toujours des places ne faisant pas partie d'un processus et auxquelles ne doit être associée aucune action. Ce choix permet une mise en œuvre systématique

Référence Internet
S7252

# Outils de modélisation des automatismes séquentiels

## Réseaux de Petri

*par* **Pierre LADET**
*Docteur ès Sciences*
*Professeur à l'Institut national polytechnique de Grenoble*
*et* **Isabel DEMONGODIN**
*Docteur ès Sciences*
*Professeur à l'université Paul-Cézanne, Aix-Marseille*



*Depuis leur première définition en 1962 par Carl Adam Petri, les réseaux de Petri sont devenus un paradigme puissant de modélisation et d'analyse, tant dans le monde industriel que dans les laboratoires de recherche. Enseignés dans les écoles d'ingénieurs et les universités, devenus en 2004 norme internationale (ISO/IEC-15909-1) sur les aspects dits « haut niveau », ils sont maintenant largement diffusés et de nombreuses études industrielles utilisent cet outil dans un objectif de conception et/ou d'exploitation de systèmes automatisés.*

*La complexité croissante de nos systèmes de production, notamment dans le domaine manufacturier, a provoqué un appel de la part des concepteurs et des*





Mars 2009

*utilisateurs de systèmes discontinus. Le succès du GRAFCET est dû à ce besoin nouveau d'un outil capable d'exprimer les deux grandes caractéristiques des systèmes séquentiels : le parallélisme et la synchronisation.*

*On sait cependant aujourd'hui que la conception et l'exploitation des systèmes de production manufacturiers, pour ne prendre que cet exemple, requièrent des modèles plus riches en information et plus concis que le GRAFCET, aux fins d'analyse, de simulation et de commande.*

*La conception ou la modification d'une installation industrielle peuvent se résumer en quatre phases :*

- *la **spécification** des fonctions qui la composent et de leurs interactions ;*
- *l'analyse ou la **validation** de la description obtenue ;*
- *la **simulation** qui complète la connaissance du système projeté et permet un dimensionnement et une évaluation de ses performances ;*
- *l'**exploitation** et la **maintenance**.*

*Chacune de ces phases repose sur l'utilisation d'un modèle, donc d'un langage. Trop souvent, les outils de modélisation utilisés ne s'appliquent qu'à l'une ou l'autre de ces phases. Dès lors, le passage d'une étape à la suivante ou le retour en arrière, souvent nécessaire dans cette démarche de conception, entraînent une perte d'acquis et l'introduction d'erreurs, d'ambiguïtés pourtant levées dans la phase précédente. La conception de systèmes qui, de plus en plus, doivent pouvoir s'adapter facilement aux exigences de la production suppose l'utilisation de modèles communs aux différentes étapes de la vie d'une application industrielle. Les réseaux de Petri se proposent de jouer ce rôle.*

# 1. Processus séquentiels

## 1.1 Nature des fonctions séquentielles

Les problèmes de représentation et d'analyse rencontrés dans la conception et la commande des processus ou systèmes discontinus sont fondamentalement différents de ceux traités depuis longtemps dans le domaine des processus continus.

■ L'**opposition processus industriels continus/processus industriels discontinus** apparaît tout d'abord dans la nature même des produits manipulés et élaborés par les uns et les autres : les flots de matière rencontrés en chimie lourde, exemple s'il en est des processus continus, laissent la place dans les processus discontinus aux séries de pièces de l'industrie manufacturière.

Les variables associées à ces produits ou aux machines et instruments qui les traitent, traduisent et formalisent cette différence : variables analogiques telles que les mesures de pression, de température, de débit ou telles que les commandes de puissance et dont l'évolution est, par définition, continue dans le temps ; variables logiques, booléennes, à évolution discontinue qu'illustrent les mesures de fin de course ou les actions dites tout ou rien.

■ La **description des processus discontinus** utilise les règles de l'algèbre de Boole. Le fonctionnement d'une machine dans ses différents modes, le transfert d'une pièce, son chargement ou son déchargement sont autant de fonctions dites logiques. On distingue cependant, parmi ces fonctions, les fonctions combinatoires et les fonctions séquentielles.

Rappelons qu'une **fonction** est dite **combinatoire** si ses sorties (actions) dépendent exclusivement de ses entrées (mesures).

Une **fonction** est dite **séquentielle** si ses sorties à un instant *t* donné dépendent non seulement de la valeur de ses entrées à cet instant, mais également des valeurs antérieures. La connaissance du passé suppose qu'il soit mémorisé, plus précisément que la fonction séquentielle prenne différents états successifs résultats du passé et qui permettent de décider de l'évolution à venir, donc des possibles changements d'état. Nous dirons que les sorties d'une fonction séquentielle à un instant *t* donné dépendent de la valeur de ses entrées à cet instant et de l'état atteint.

■ La **notion d'état** est familière aux automaticiens spécialistes des systèmes continus. Mais à l'état « continu », qui évolue de façon continue, se substitue, dans la conception et la commande des systèmes séquentiels, un **ensemble fini d'états** et, conjointement, un ensemble de changements d'état ou **transitions**. Le temps lui-même n'est plus considéré comme une variable continue, mais il est appréhendé à travers un ensemble d'instants caractéristiques de l'évolution du processus, qui voient la réalisation de conditions (valeurs booléennes), l'apparition d'événements (changements de valeurs booléennes).

Le passage d'un état à l'autre peut être instantané ou temporisé mais, généralement, il dépend de la réalisation d'une condition et/ou de l'apparition d'un événement. Pour un état donné, une occurrence d'événement, une condition vraie, entraînent immédiatement le franchissement d'une transition donc un changement d'état, traduction de l'évolution du processus séquentiel, de la fonction séquentielle.

■ À la succession des états correspond une succession d'**actions élémentaires** associées aux différents états. Dans le cas d'un processus simple, une seule fonction séquentielle suffit à décrire l'évolution. Mais la complexité des processus industriels oblige aujourd'hui à suivre une démarche de décomposition géographique et/ou fonctionnelle. Décrire un processus séquentiel revient à mettre en évidence les différentes fonctions qui le composent, à représenter chacune de ces fonctions et à préciser, si elles existent, les relations entre ces fonctions.

## 1.2 Représentation par graphe d'états

Représenter une fonction séquentielle par un graphe consiste à matérialiser, par l'intermédiaire de deux types de nœuds différents, d'une part les états, d'autre part les transitions entre états. Dans le cas du graphe d'états, les **états** sont représentés par autant de **nœuds** appelés **places**, les **transitions** entre états par autant d'**arcs orientés** qui joignent deux à deux les états en cause.

L'ensemble constitue un modèle du processus séquentiel. Toute place se voit associer un ensemble d'actions (éventuellement vide) et à chaque arc correspond une condition d'évolution, ce que nous résumons comme suit :

État → Place → Ensemble d'actions

Transition → Arc → Condition d'évolution

**Exemple :** cahier des charges n° 1

Soit le chariot *C* représenté sur la figure **1**. *C* peut se déplacer entre les deux fins de course *A* et *B* grâce à un moteur à deux sens de marche commandé par deux relais *D* et *G*.

Lorsque l'opérateur ferme le contact *M*, et à condition que *C* soit en *A*, le chariot se déplace vers la droite jusqu'en *B*. Arrivé en *B*, *C* retourne immédiatement vers la gauche et s'arrête en *A* si *M* est ouvert ; dans le cas contraire, le chariot commence un nouveau cycle.

Le cahier des charges n° 1 laisse apparaître trois états différents représentés par le graphe de la figure **2** :

– $E_0$ – le chariot *C* est au repos : aucune action ;
– $E_1$ – le chariot *C* est en déplacement vers *B* : action *D* ;
– $E_2$ – le chariot *C* est en déplacement vers *A* : action *G*.

$E_0$ sera appelé **état initial** de la fonction séquentielle. La seule évolution possible à partir de l'état initial est celle traduite par la **transition** $E_0 \rightarrow E_1$, transition franchie dès lors que la condition *M.A*, expression combinatoire, est vérifiée. Si la condition *M.A* est vraie pour l'état initial, il y a changement d'état ; le nouvel état est $E_1$.

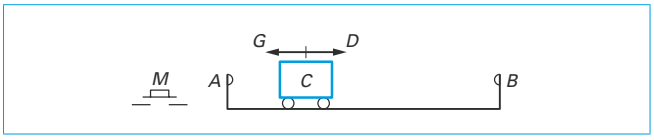



**Figure 1 – Cahier des charges n° 1, synoptique**

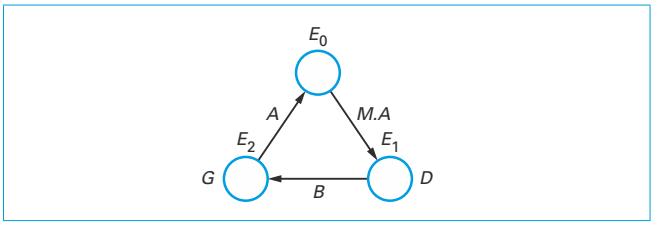



**Figure 2 – Cahier des charges n° 1, modèle graphe d'états**

La transition $E_1 \rightarrow E_2$ portera la condition *B*. La fonction séquentielle restera dans l'état $E_1$ avec action *D* jusqu'à ce que cette condition se réalise. Le nouvel état sera $E_2$, avec action *G*.

Enfin, la transition $E_2 \rightarrow E_0$ sera franchie si la condition *A* est vérifiée pour l'état $E_2$. La fonction séquentielle retrouvera alors son état initial.

## 1.3 Parallélisme et synchronisation

**Exemple :** cahier des charges n° 2

Considérons les deux chariots $C_1$ et $C_2$ (figure **3**). $C_1$ et $C_2$ peuvent se déplacer entre les fins de course $A_1 - B_1$ (respectivement $A_2 - B_2$), grâce à deux moteurs à double sens de marche commandés par les relais $D_1$, $G_1$ (respectivement $D_2$, $G_2$).

Lorsque l'opérateur ferme le contact *M*, et à condition que $C_1$ et $C_2$ soient en $A_1$ et $A_2$, les deux chariots se déplacent simultanément vers la droite. Le retour vers $A_1$ (respectivement $A_2$) sera commandé lorsque les deux chariots seront arrivés en $B_1$ (respectivement $B_2$). $C_1$ et $C_2$ reviennent donc simultanément en $A_1$, $A_2$ après avoir réalisé un « rendez-vous ». On dira aussi que le premier chariot arrivé en $B_1$, $B_2$ s'arrête et attend l'autre.

Le cycle se termine lorsque $C_1$ et $C_2$ sont arrivés respectivement en $A_1$ et $A_2$.

■ Ce deuxième cahier des charges se représente sous la forme du graphe d'états de la figure **4**. On constate que le nombre de places va croissant avec la complexité du système. Dans un graphe d'états, et par définition, le nombre de places est égal au nombre d'états. La complexité d'une fonction, d'un système, se retrouve donc dans le nombre de places nécessaires pour représenter chacun des états.

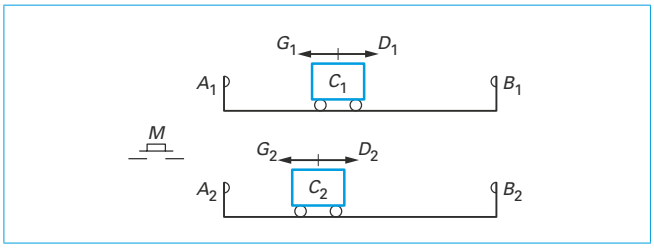



**Figure 3 – Cahier des charges n° 2, synoptique**

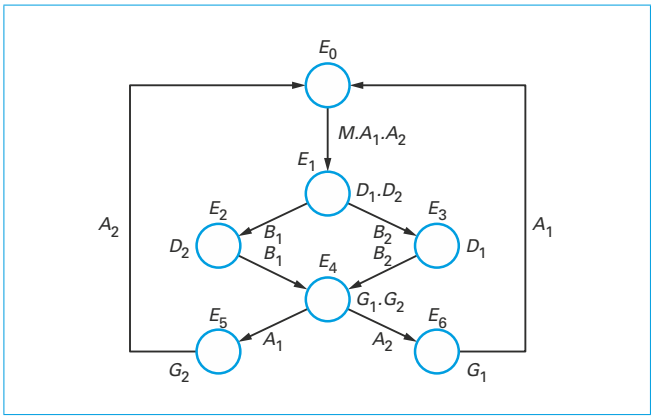



**Figure 4 – Cahier des charges n° 2, modèle graphe d'états**

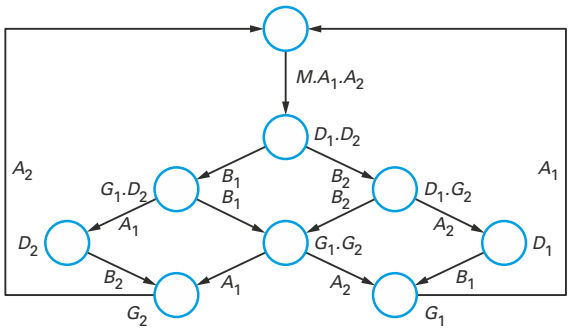



**Figure 5 – Cahier des charges n° 3, modèle graphe d'états**

Ainsi, l'exemple précédent appliqué à un groupe de $N$ chariots conduirait à un graphe de $2^{N+1} - 1$ places, soit 2 047 places pour $N = 10$.

Cette complexité se retrouve également si l'on considère le nombre de transitions. Chaque place représente un état « global » ; pour chacune d'elles, il est donc nécessaire de considérer toutes les évolutions possibles, ce qui revient à multiplier le nombre d'arcs.

■ Le **principe de base du graphe d'états**, *un état = une place*, s'oppose à toute décomposition d'un système complexe et par conséquent heurte la démarche utilisée par les ingénieurs dans toute analyse de processus. Dans notre exemple, la lecture du graphe ne laisse pas apparaître la décomposition naturelle qui s'impose à la vue de la figure **3**. Deux fonctions séquentielles apparaissent en effet nettement dans ce schéma et dans le cahier des charges qui l'accompagne. Ces deux fonctions sont liées aux deux chariots et présentent un fort parallélisme avec deux points de synchronisation qui correspondent au départ simultané des deux chariots et au rendez-vous en $B_1$, $B_2$.

La complexité des graphes d'états et leur impossibilité à représenter le parallélisme et la synchronisation apparaissent également dans l'exemple suivant.

**Exemple :** cahier des charges n° 3

Considérons à nouveau les deux chariots $C_1$ et $C_2$ (figure **3**). Lorsque l'opérateur ferme le contact $M$, et à condition que $C_1$ et $C_2$ soient en $A_1$ et $A_2$, les deux chariots partent simultanément mais se déplacent librement sur $A_1 - B_1 - A_1$ et $A_2 - B_2 - A_2$. Le deuxième point de synchronisation (rendez-vous en $B_1$, $B_2$) est supprimé, ce qui signifie que de nouveaux états sont à prendre en considération, comme l'indique le graphe de la figure **5**.

## 1.4 Conception et commande des processus séquentiels

La conception et la commande d'un processus séquentiel et, par conséquent, sa représentation passent par une méthode d'analyse descendante qui doit faire apparaître les différentes fonctions séquentielles et les relations entre fonctions.

La décomposition en fonctions séquentielles distinctes repose sur la recherche de **séquences parallèles**, qui s'exécutent en parallèle dans le temps mais qui peuvent, au cours de leur évolution, se synchroniser entre elles ou sur une ressource partagée.

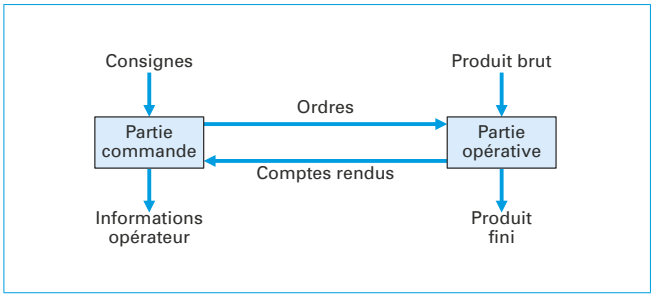



**Figure 6 – Décomposition d'un système en partie commande et partie opérative**

L'outil de représentation d'un système séquentiel doit permettre la visualisation du parallélisme et de la synchronisation entre fonctions. Nous avons montré (§ 1.3) que cet outil de modélisation ne pouvait être le graphe d'états. Par ailleurs, tout système peut se représenter en deux parties coopérantes (figure **6**) appelées :

– **partie commande** ;
– **partie opérative**.

La décomposition d'un système en une partie commande et une partie opérative facilite sa description et son analyse. Elle permet en effet d'aborder des problèmes de natures différentes liés au sous-système d'exécution (partie opérative) ou au sous-système de décision (partie commande). On dit aussi que la partie opérative représente l'environnement de la partie commande, environnement équivalent au processus physique réel ou environnement image de ce processus.

L'analyse de la partie commande fait apparaître les différentes fonctions séquentielles ou séquencements d'opérations traduits en termes d'actions dans la partie opérative, laquelle renvoie les informations nécessaires aux décisions prises dans la partie commande.

# 2. Réseaux de Petri autonomes

## 2.1 Définitions. Exemples

### 2.1.1 Définition informelle des réseaux de Petri

**Un réseau de Petri est un graphe composé de deux types de nœuds :**

– les **places** qui permettent de décrire les **états** du système modélisé ;
– les **transitions** qui représentent les **changements d'état**.

Places et transitions sont reliées par des arcs orientés (figure **7**). On dit qu'un réseau de Petri est un graphe biparti orienté.

Une place peut contenir un nombre entier de **jetons** ou **marques**. L'ensemble des marques présentes à un instant donné dans les places constitue le **marquage** du réseau à cet instant et représente l'état du système.

Le **marquage** dit « **initial** » décrit l'état initial du système modélisé. Pour le réseau de la figure **7**, le marquage initial est donné par 3 jetons dans la place $P_1$, 0 jeton dans la place $P_2$ et 0 jeton dans la place $P_3$.

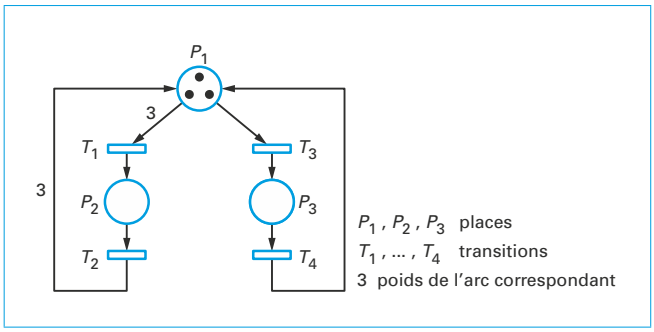



**Figure 7 – Exemple de réseau de Petri**

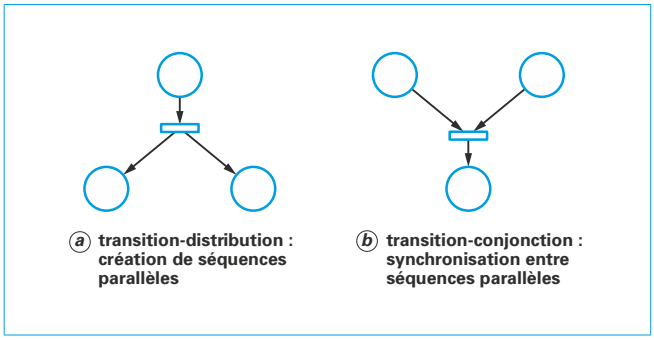
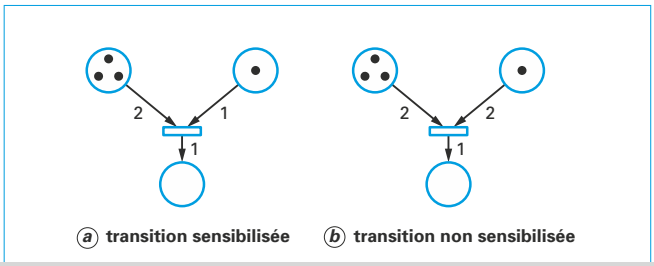



**Figure 8 – Séquences parallèles et synchronisation**



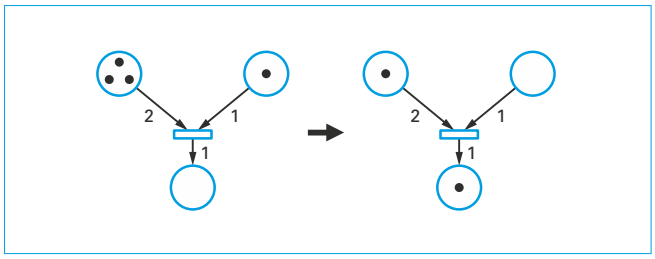




**Figure 10 – Franchissement d'une transition**

**Franchir une transition** consiste à prendre, dans chaque place d'entrée, un nombre de jetons égal au poids de l'arc joignant cette place à la transition et à déposer, dans chaque place aval ou place de sortie, un nombre de jetons égal au poids de l'arc joignant la transition à chacune de ces places (figure **10**). Le franchissement des transitions et les modifications de marquage qu'il entraîne permettent d'analyser la dynamique du système modélisé.

**Exemple**

■ **Cahier des charges n° 1** (figure **1**) : le cahier des charges n° 1, décrit au § 1.2 par le graphe d'états de la figure **2**, ne conduit pas à une décomposition fonctionnelle. Les trois états que peut prendre le système de commande réduit à une seule fonction séquentielle seront représentés par un réseau de Petri composé de trois places, selon la structure indiquée sur la figure **11**. On retrouve sur le réseau de Petri obtenu l'association place-action et l'association transition-condition introduites avec les graphes d'états.

■ **Cahier des charges n° 2** (figure **3**) : la lecture du cahier des charges n° 2 et le schéma qui l'accompagne laissent apparaître non plus une seule fonction mais deux fonctions séquentielles de commande, appliquées, l'une au chariot $C_1$, l'autre au chariot $C_2$. On relève cependant deux points de synchronisation, le premier pour traduire le départ simultané des deux chariots vers la droite, le second pour prendre en compte le rendez-vous des chariots en $B_1$, $B_2$.

Le réseau de Petri de la figure **12** met en évidence le parallélisme

Référence Internet
S7254

# Applications des réseaux de Petri

*par* **Pascal RICHARD**
*Maître de conférences en Informatique à l'Université de Poitiers (LISI/ENSMA)*

*et* **Christophe HARO**
*Ingénieur en Informatique*
*Enseignant-chercheur à l'École d'ingénieurs en informatique pour l'industrie (EIII)*
*Chercheur au laboratoire d'informatique (LI), Université de Tours*



*L'article présente l'utilisation de techniques utilisant les réseaux de Petri (RdP) pour aider les ingénieurs de production dans leur travail. Après une présentation générale, une première partie aborde, par l'étude d'un cas concret, la modélisation, le dimensionnement et l'évaluation des performances d'un atelier de fabrication de type job-shop. Une deuxième partie présente un cas industriel emprunté à la production verrière et décrit la mise en œuvre d'un système de planification de production.*

*Les réseaux de Petri, inventés par Carl Adam Petri en 1962, suscitent toujours aujourd'hui de nombreux travaux de recherche dans beaucoup de pays. Cet engouement est dû à l'élégante théorie mathématique sous-jacente à cette technique de modélisation graphique dont les fondements théoriques sont présentés dans l'article « Outils de modélisation des automatismes séquentiels. Les réseaux de Petri [R 7 252] ». Malgré cette effervescence dans le domaine de la recherche, peu d'applications de taille industrielle voient le jour. Cela s'explique en partie par l'absence de normalisation internationale des réseaux de Petri, contrairement au grafcet. Cet inconvénient est en passe d'être corrigé puisque la norme ISO 15909 sur les réseaux de Petri de haut niveau est actuellement en cours d'élaboration. L'objectif de cet article est de présenter les applications des réseaux de Petri dans un contexte industriel. La suite de l'introduction établit une synthèse des types d'applications privilégiées des réseaux de Petri et replace dans ce cadre les deux cas d'étude qui seront ensuite présentés.*





Septembre 2001

# 1. Types d'application

Deux grandes applications des réseaux de Petri émergent dans la littérature suivant que le réseau est utilisé durant le cycle de conception ou bien comme outil de mise en œuvre d'un système. Dans le premier cas, le réseau spécifie le système et constitue le **point de départ du cycle de conception**. Ces spécifications permettent ensuite des analyses qualitatives et quantitatives du système à concevoir. Le non-respect de certaines clauses du cahier des charges engendre des modifications du modèle dont les répercussions seront évaluées en rejouant les analyses. La seconde catégorie d'applications consiste à utiliser le réseau comme **structure interne d'un système de contrôle ou d'aide à la décision**.

Un réseau de Petri est un outil graphique de modélisation de système complexe. Sa puissance d'expression permet d'étudier des systèmes composés de sous-systèmes fonctionnant en parallèle, communiquant et partageant des ressources. Le modèle représente les différents états possibles du système.

Afin d'étendre leurs champs d'application aux systèmes continus et semi-continus, des extensions des réseaux de Petri ont été proposées dans la littérature : il s'agit, respectivement, des réseaux continus et des réseaux hybrides. Nous n'en parlerons pas dans cet article, bien qu'ils constituent une branche très vivante au niveau de la recherche universitaire.

L'apprentissage de la modélisation par réseau de Petri n'est généralement pas aisé pour un débutant. Le concepteur du modèle doit raisonner en termes d'états et de transitions d'états ce qui n'est pas une démarche naturelle. Il est nécessaire tout d'abord de maîtriser la modélisation des structures de base comme l'enchaînement d'opérations, les communications et le partage de ressources. On peut effectuer le parallèle avec l'apprentissage de la programmation des ordinateurs et la maîtrise des structures algorithmiques de base. Toutefois, la modélisation par réseau de Petri se prête très bien à la conception modulaire des systèmes. Ainsi chaque sous-système est modélisé séparément, puis les interactions de ceux-ci reviennent à interconnecter les réseaux par de nouvelles places et transitions. Nous pouvons noter qu'à l'usage, la modélisation par réseau de Petri s'impose très rapidement aux autres outils de modélisation pour étudier des systèmes complexes.

La puissance d'expression des réseaux de Petri permet aux concepteurs de modéliser un même système de plusieurs façons. Par exemple, les activités d'un système peuvent être modélisées par des **places** ou des **transitions**. Dans le premier cas, le traitement de l'opération est symbolisé par la présence d'un **jeton** dans la place la modélisant et dans le second elle est modélisée par le **tir** de la transition la modélisant. Mais il faut avoir présent à l'esprit que le modèle sera plus ou moins facile à analyser suivant la stratégie de modélisation adoptée. Par exemple, si l'analyse à effectuer est fondée sur le calcul des P-semiflots, les activités du système seront modélisées par des places. C'est notamment le cas lorsque l'exclusion mutuelle d'accès de processus à des ressources doit être validée. L'absence de méthode systématique de modélisation est conjointement la limite de l'approche réseau de Petri en terme de facilité d'utilisation dans un contexte industriel et sa richesse en terme de facilité d'analyse de tels systèmes. A contrario, le grafcet est une méthode de modélisation systématique, mais dont les capacités d'analyse sont limitées.

L'apport des réseaux de Petri dans le cycle de conception est méthodologique puisque le modèle n'est pas utilisé uniquement pour exécuter ou implémenter les spécifications du système. La démarche complète suit les étapes suivantes :

— **conception du modèle** : le concepteur construit un réseau de Petri répondant à la description du système à concevoir ;
— **analyse qualitative** : cette analyse permet de valider les spécifications fonctionnelles du système ; par exemple, vérifier la bornitude d'une place permet de valider le respect de la capacité du buffer qu'elle modélise ;
— **analyse quantitative** : cette analyse permet de valider les performances du système ;
— **génération de code** : la génération de code permet d'établir en général le squelette du programme contrôlant le procédé, par exemple dans le langage de programmation d'un automate programmable.

Ce programme est ensuite complété pour tenir compte de l'environnement industriel des entrées/sorties.

Pour illustrer cette première catégorie d'applications, nous présenterons la conception d'un atelier de fabrication de perforatrices de bureau. La modélisation modulaire du système sera décrite, puis l'analyse des performances sera étudiée afin de satisfaire les commandes prévisionnelles de perforatrices. Dans cette étude de cas, l'analyse quantitative et la génération de code ne seront pas abordées. La seconde catégorie d'applications des réseaux de Petri réside dans la mise en œuvre d'un système industriel utilisant un réseau. Dans ce cas, le réseau de Pétri n'est plus un outil méthodologique de conception et de dialogue entre concepteurs mais devient le modèle interne d'un procédé automatique ou d'un système de décision. Nous distinguons trois sous-classes d'utilisation dans la mise en œuvre suivant qu'elle dépend de l'interprétation du réseau (joueur de jetons), de l'utilisation du graphe de marquages ou de l'utilisation des relations d'algèbre linéaire sous-jacentes au réseau. En pratique, ces trois techniques visent à résoudre des problèmes différents.

Dans la première sous-classe d'utilisation d'un réseau de Petri pour implémenter un système, un joueur de jeton interprète la règle de tir du réseau en tenant compte des interactions avec le procédé contrôlé ou l'utilisateur utilisant un système d'aide à la décision. Ce type de réseau devient dépendant de son contexte d'utilisation et est appelé pour cela **réseau interprété**. L'analogie avec le grafcet et les automates programmables est alors immédiate.

Dans la seconde sous-classe, le réseau de Petri permet d'exprimer la dépendance et l'indépendance (le parallélisme potentiel) entre les actions ou les événements d'un système. L'**arbre de marquages** du réseau permet de décrire l'enchaînement des états possibles. Ils décrit précisément toutes les séquences d'actions ou d'événements réalisables. Cette description détaillée du système permet d'envisager la recherche d'une séquence de transitions particulière en optimisant un ou plusieurs critères de performance. Par exemple, la séquence la plus courte en nombre de transitions permet d'effectuer une reprise d'erreur efficace entre un état d'erreur et un état sain. De même, si les séquences représentent les ordonnancements réalisables des activités d'un atelier de fabrication, la recherche de la séquence de plus courte durée permet d'établir le plan de fabrication optimal de l'atelier. Le graphe de marquages est alors le moyen d'explorer l'ensemble des solutions réalisables pour un problème complexe. L'implémentation reprendra alors logiquement le réseau comme structure de contrôle et de commande, ou bien comme structure de données informatiques mémorisant le problème à résoudre.

Enfin, dans la troisième sous-classe, les réseaux de Petri sont utilisés pour mettre en œuvre un système. Cela revient à utiliser le réseau en modèle interne comme précédemment, mais cette fois en utilisant les relations de l'algèbre linéaire sous-jacentes au graphe du réseau ou des outils de programmation mathématique. Ce type d'application se pose dans la conception de système informatique d'aide à la décision en interaction, non plus avec un environnement industriel, mais avec un utilisateur. La seconde étude que nous aborderons sera dédiée à ce problème dans le contexte de l'aide à la planification de production dans l'industrie verrière. Nous verrons dans ce cas les apports des réseaux de Petri par rapport aux approches classiques de type recherche opérationnelle.

## 2. Analyse d'un système en cours de conception

Dans cette partie, nous présentons un **système à fonctionnement cyclique de type job-shop** à partir de l'étude d'un cas concret, mais cependant suffisamment simple pour permettre de ne pas accumuler les niveaux de difficultés. Nous montrons comment étudier son comportement dynamique et comment en optimiser les performances.

### 2.1 Présentation du cas simplifié

Un atelier doit produire des **perforatrices de bureau**. Celles-ci sont constituées à partir de l'assemblage de plusieurs types de pièces.

Le corps de la perforatrice est constitué de l'assemblage de **deux demi-coquilles** rigoureusement identiques. Chaque demi-coquille est obtenue par une opération de tournage réalisée sur un tour à commandes numériques, suivie d'un ensemble d'opérations réalisées sur une fraiseuse.

Le **poinçon** est obtenu par une opération de tournage, suivie d'une opération de fraisage. Il passe ensuite sur une rectifieuse qui termine sa préparation. Le poinçon est emmanché à force dans un **bouton** permettant d'actionner la perforatrice. Le bouton subit une opération de tournage puis une opération de fraisage.

Les pièces entrent dans l'atelier sur un tapis roulant. Un robot $R_1$ les saisit et charge le tour $M_1$. Lorsque l'opération est terminée, un second robot $R_2$ décharge $M_1$ et charge la fraiseuse $M_2$ avec la pièce intermédiaire. L'opération de fraisage terminée, un robot $R_3$ décharge $M_2$. Si la pièce intermédiaire est un poinçon, il la charge sur la rectifieuse $M_3$. Sinon, il la dépose sur un tapis qui conduit les pièces à l'atelier d'assemblage. À la fin d'une opération de rectification, le robot $R_3$ décharge la rectifieuse $M_3$ et dépose la pièce sur le tapis roulant.

Pour simplifier l'étude de ce cas, nous confondrons les opérations de chargement/déchargement des machines avec les opérations réalisées par ces machines. Ainsi, par exemple, le travail de $R_1$ sera intégré à l'opération de tournage réalisée par $M_1$. Le système se simplifie donc et devient celui représenté sur la figure **1**.

Dans la suite, on désigne par $P_1$ le produit brut en entrée de l'atelier destiné à être usiné pour réaliser le poinçon ou le poinçon

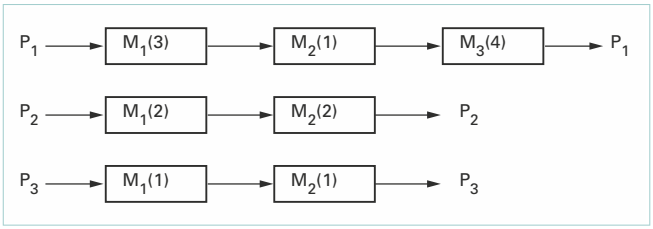



**Figure 2 – Gamme de fabrication**

par $P_3$ celui utilisé pour produire les demi-coquilles formant le corps de l'objet fini. La perforatrice étant constituée des deux demi-coquilles, d'un poinçon et d'un bouton, il est clair que l'atelier devra produire à partir de 25 % de $P_1$, 25 % de $P_2$ et de 50 % de $P_3$.

On suppose que les pièces entrent dans l'atelier selon la séquence $\sigma_0 = < P_3, P_1, P_2, P_3 >$. Le séquencement sur les machines, c'est-à-dire l'ordre de passage des pièces sur chacune d'elles, est donné par :

$$\sigma(M_1) = < P_3, P_1, P_2, P_3 > \ ; \ \sigma(M_2) = < P_3, P_1, P_2, P_3 > \ ; \ \sigma(M_3) = < P_1 >$$

Cette ligne de fabrication est associée à une gamme linéaire. Les pièces visitent les machines dans l'ordre donné par les gammes suivantes :

$$P_1 : < M_1(3), M_2(1), M_3(4) >$$
$$P_2 : < M_1(2), M_2(2) >$$
$$P_3 : < M_1(1), M_2(1) >$$

Les nombres entre parenthèses précisent la durée de l'opération considérée sur la machine correspondante. Ces gammes sont représentées sur la figure **2**. Dans l'atelier, les machines sont alignées dans l'ordre de la gamme.

### 2.2 Modélisation

La modélisation d'un job-shop se fait usuellement en **trois étapes**.

■ On commence par modéliser le **processus de fabrication de chaque type de pièce**. Pour cela, chaque opération réalisée sur un produit dans sa gamme de fabrication est représentée par une transition *t*. Les différentes transitions associées aux opérations successives que subit le produit sont séparées les unes des autres par des places symbolisant des stocks tampons ou plus généralement

Référence Internet
S7210

# Optimisation continue

par **Claude LEMARÉCHAL**
*Ingénieur de l'École nationale supérieure d'Électronique, d'Électrotechnique, d'Informatique et d'Hydraulique de Toulouse (ENSEEIHT)*
*Docteur ès sciences*
*Directeur de recherche à l'Institut national de recherche en Informatique et en Automatique (INRIA)*



*En tant que branche des mathématiques appliquées, l'optimisation est maintenant omniprésente. C'est à la fin de la dernière guerre mondiale qu'elle est devenue vraiment opérationnelle, avec l'apparition de la programmation linéaire pour organiser les convois américains vers l'Europe (les « liberty ships »). Elle s'est ensuite fortement développée à partir des années 1960, pendant lesquelles les problèmes non linéaires ont pu être abordés efficacement, grâce principalement aux méthodes de « quasi-Newton ».*

*Les problèmes traités dans cet article appartiennent au domaine de l'**optimisation continue**, dans laquelle les variables à optimiser peuvent prendre tout un **continuum de valeurs**. Ceci s'oppose aux problèmes **combinatoires**, dans les-*

*quels il s'agit de trouver la meilleure parmi un **ensemble fini** de possibilités. Nous ne parlons pas dans cet article de ces derniers.*

*Les méthodes d'optimisation continue relèvent toutes de l'analyse des fonctions de plusieurs variables réelles, et consistent toutes à construire une suite itérative de solutions approchées. C'est ce type de méthodes qui fait l'objet du présent article.*

# 1. Exemples de problèmes d'optimisation

## 1.1 Calcul d'un équilibre chimique

On considère le système formé par le mélange d'un certain nombre d'éléments chimiques (par exemple trois éléments : le soufre, l'oxygène et le fluor). Dans des conditions données de pression $P$ et de température thermodynamique $T$, un équilibre s'établit entre ces éléments et tous les composés qu'ils sont susceptibles de former (ici il y a 35 corps chimiques possibles : $SO_2$, $SF_2$, etc.). On désire **calculer les concentrations des divers constituants à l'équilibre**. Pour cela, on exprime qu'à l'équilibre l'énergie du système est minimale.

### Modèle mathématique

Ce problème illustre les éléments essentiels présents dans une optimisation.

- Appelons $x_i$ le nombre de moles du composé $i$. Les $x_i$, $i = 1, ..., n$ (ici $n = 35$) sont inconnus, et le problème, justement, est d'en calculer les bonnes valeurs. Néanmoins, si on leur attribue des valeurs arbitraires, on peut alors calculer l'énergie du système. C'est pourquoi ces inconnues sont appelées **variables de commande** : elles gouvernent l'état du système cherché.
- A priori, les $x_i$ sont des nombres réels : l'**espace des commandes** est $\mathbb{R}^n$, l'espace vectoriel des vecteurs à $n$ coordonnées.
- La **fonction coût** à minimiser est ici l'enthalpie libre de Gibbs, qui peut être calculée dès que des valeurs (arbitraires) sont attribuées aux variables de commande. Elle a pour expression :

$$G(x) = \sum_{i=1}^{n} e_i x_i + RT \sum_{i \in \Gamma} x_i \ln\left(\frac{P x_i}{\sum_{j=1}^{n} x_j}\right)$$

avec $e_i$ potentiel chimique (connu) du composé $i$ dans les conditions normales,

$R$ constante des gaz parfaits,

$\Gamma$ ensemble des produits gazeux dans les conditions de l'expérience (les liquides sont supposés non miscibles).

- Enfin, les $x_i$ ne peuvent varier dans $\mathbb{R}$ librement et indépendamment les uns des autres : le problème possède des **contraintes**, qui sont ici de deux types : d'une part $x_i \geqslant 0$ (bien entendu) et, d'autre part, les contraintes de conservation de la matière. Ces dernières expriment que le nombre total de moles de chaque élément, sous forme libre ou combinée, est invariant ; ce sont des contraintes linéaires d'égalité. Ici, il y a trois contraintes, relatives au soufre, à l'oxygène et au fluor.

Appelons $b_j$ le nombre de moles (donné) de l'élément $j$ dans le mélange, et $a_{ij}$ la fraction molaire de l'élément $j$ dans le composé $i$ (donnée par la formule chimique de chaque composé : dans $SO_2$, il y a un demi $S_2$, un $O_2$ et zéro $F_2$). Le problème est donc :

$$\begin{cases} \min G(x) & \\ x_i \geqslant 0 & i = 1,..., n \ (n = 35) \\ \sum_{i=1}^{n} a_{ij} x_i = b_j & j = 1,..., m \ (m = 3) \end{cases}$$

c'est un problème d'optimisation tout à fait standard. Remarquons toutefois que les contraintes $x_i \geqslant 0$ ne sont pas vraiment des contraintes car $G$ augmente si un $x_i$ tend vers 0. On les appelle plutôt des **barrières**.

## 1.2 Trajectoire d'un objet volant

Notons $X$, $V$, $A$ les vecteurs position, vitesse et accélération (trois coordonnées chacun) d'un objet volant. Ces trois vecteurs sont reliés par le système différentiel :

$$\dot{X}(t) = V(t) \ ; \quad \dot{V}(t) = A(t) \text{ (six équations scalaires)} \qquad \textbf{(1)}$$

On peut alors se poser le problème simple suivant : **partant de conditions initiales données $X_0$, $V_0$, arriver le plus près possible d'un objectif $X_T$ donné, en un temps $T$ donné.**

### Modèle mathématique

La fonction à minimiser est la distance entre $X(T)$ et $X_T$. Les variables de commande et la mise en équation dépendent de l'objet volant.

- S'il s'agit d'un hélicoptère, il est raisonnable de considérer que le pilote peut choisir à tout instant son accélération (inférieure à une accélération maximale donnée $G$), c'est-à-dire que $A$ peut être choisi comme commande. Alors, l'espace des commandes est un ensemble de triplets de fonctions du temps :

$$A(t) = [a_1(t), a_2(t), a_3(t)].$$

Compte tenu des conditions initiales, le système (1) a une solution unique $[X(t), V(t)]$ lorsque la commande $A$ est fixée entre 0 et $T$. Par conséquent, $X(T)$ (et donc la fonction coût) ne dépend que de $A$. Le problème peut alors se noter comme suit :

$$\left.\begin{array}{l} \min J(A) = |X(T) - X_T|^2 \\ \dot{X}(t) = V(t) \ ; \quad \dot{V}(t) = A(t) \ ; \quad X(0) = X_0 \ ; \quad V(0) = V_0 \\ -G \leqslant a_i(t) \leqslant G \quad i = 1, 2, 3 \quad t \in [0, T] \end{array}\right\} \qquad \textbf{(2)}$$

où la fonction coût a été notée $J$.

● Si maintenant l'objet volant est un avion, $A$ ne peut plus être pris comme commande car il faut tenir compte de la dynamique de l'avion. Un modèle possible est le suivant : on repère $X = (x, y, z)$ en coordonnées cartésiennes et $V = (v, \theta, \varphi)$ en coordonnées sphériques. Les commandes sont maintenant la poussée instantanée $p$, l'incidence $i$ (angle entre $V$ et l'axe longitudinal de l'avion) et la gîte $\mu$ (angle entre le plan de l'avion et le plan horizontal). Ce sont ces trois paramètres que le pilote peut fixer à tout instant. Le vol est alors (en supposant que l'avion a une masse unité et en notant $g$ l'accélération due à la pesanteur) :

$$\begin{cases} \dot{X}(t) = V(t) \quad \text{(trois équations scalaires)} \\ \dot{v}(t) = p(t)\cos i(t) - B - g\sin\theta(t) \\ \dot{\theta}(t) = \dfrac{[p(t)\sin i(t) + C]\cos\mu(t) - g\cos\theta(t)}{v(t)} \\ \dot{\varphi}(t) = \dfrac{[p(t)\sin i(t) + C]\sin\mu(t)}{v(t)\cos\theta(t)} \end{cases}$$

où $B$ et $C$ (la traînée et la portance) sont des fonctions connues des conditions de vol à l'instant $t$ (éventuellement données par des abaques).

De plus, des contraintes de la forme :

$$h_j[X(t), V(t), p(t), i(t), \mu(t)] \leqslant 0$$

peuvent être incluses, exprimant que la poussée est limitée, que l'altitude doit être positive, que l'avion ne doit pas se désintégrer, etc.

Quelle que soit la complexité du modèle, il importe de remarquer que, là encore, l'objectif et les contraintes dépendent uniquement des commandes, par l'intermédiaire de l'intégration d'un système différentiel. De ce point de vue, il n'y a pas de différence essentielle entre un avion et un hélicoptère.

Les problèmes de ce type sont appelés **problèmes de commande optimale**. Les équations différentielles caractérisant la dynamique du système sont les **équations d'état** ; leurs solutions sont les **variables d'état**, caractérisant l'état du système à chaque instant. L'espace $H$ des commandes est un ensemble de fonctions du temps, c'est-à-dire, essentiellement, un espace vectoriel de dimension infinie.

## 1.3 Un problème typique d'asservissement

On se limite d'habitude au feedback le plus simple, une fonction affine de l'observation. La commande optimale est donc cherchée sous la forme :

$$u(t) = K(t)\ y_2(t) + r(t).$$

Pour concilier les deux exigences a priori contradictoires – suivre une trajectoire et limiter les oscillations de la commande –, on choisit comme fonction coût une combinaison de

— la distance entre $y_1(t)$ et $\tilde{y}_2(t)$ d'une part ;
— la dérivée de la commande par rapport au temps, d'autre part.

Il s'agit donc d'un **problème de commande optimale** où les commandes sont $K(t)$ et $r(t)$ :

$$\begin{cases} \min \displaystyle\int_0^T \left\{ p[y_1(t) - \tilde{y}_1(t)]^2 + \dot{u}^2(t) \right\} dt \\ u(t) = K(t)y_2(t) + r(t) \\ \dot{y}(t) = Ay(t) + Bu(t) + f(t) \ ; \quad y(0) \text{ donné} \end{cases}$$

$p$ étant un coefficient positif judicieusement choisi.

Une fois ce problème résolu, la connaissance de $K$ et de $r$ permet de câbler la boucle de contre-réaction. Dans cet exemple, le modélisateur doit opérer certains choix au niveau qualitatif (forme du *feedback*, fonction coût, etc.) et cela implique un dialogue actif entre l'automaticien posant le problème et le mathématicien censé le résoudre.

## 1.4 Un problème d'identification

Le problème est le suivant : en observant à l'aide d'un sismographe l'effet d'une explosion, déterminer l'impédance acoustique du sous-sol.

### Modèle mathématique

Sans entrer dans les détails, dégageons les idées essentielles. On suppose que l'impédance acoustique $s$ ne dépend que de la profondeur. La pression acoustique $y$ vérifie alors le système

$$s(x)\ y''_{tt}(x, t) - [s(x)y'_x(x, t)]'_x = 0$$
$$y(x, 0) = y'_t(x, 0) = 0 \ ; \quad -s(0)y'_x(0, t) = g(t)$$

avec $'$ le symbole représentant la dérivation partielle,
$t$ le temps,

Référence Internet
S7211

# Optimisation discrète

*par* **Marie-Claude PORTMANN**
*Docteur ès sciences mathématiques, Université Nancy 1*
*Professeur à l'École Nationale Supérieure des Mines de Nancy, INPL*

*et* **Ammar OULAMARA**
*Docteur en informatique de l'Université Joseph Fourier, Grenoble*
*Maître de Conférences à l'École Nationale Supérieure des Mines de Nancy, INPL*



*Résoudre un problème d'optimisation, c'est rechercher, parmi un ensemble de solutions qui vérifient des contraintes données, la ou les solutions qui rendent minimale (ou maximale) une fonction mesurant la qualité de cette solution. Cette fonction est appelée* ***fonction-objectif****. En général, pour modéliser un problème d'optimisation, on commence par définir les éléments qui composent les* ***contraintes*** *et la fonction objectif. Parmi ces éléments, certains sont connus et sont appelés* ***paramètres*** *du problème. On lit leur valeur dans des bases de données ou on les fournit dans des fichiers ou encore en les tapant au clavier d'un ordinateur. D'autres éléments sont inconnus et sont appelés* ***inconnues*** *ou* ***variables****. Les contraintes et la fonction objectif s'expriment à l'aide de formules mathématiques qui combinent les paramètres connus et les variables du problème. Les variables correspondent souvent à des décisions à prendre de manière à obtenir l'optimum souhaité. On parle d'optimisation continue (cf. [S 7 210], si les variables représentant les décisions prennent leur*





Décembre 2006

*valeur sur un ensemble continu de valeurs : par exemple, tous les réels contenus entre deux limites. On parle d'**optimisation discrète** si les variables prennent leur valeur dans un ensemble fini ou dans un ensemble dénombrable, comme par exemple l'ensemble des entiers. Dans le cas le plus général, une partie des variables sont continues et une autre partie des variables sont discrètes. C'est la difficulté des problèmes contenant des variables discrètes qui nous intéressent ici.*

*Nous présentons ici quatre problèmes concrets et nous les modélisons en utilisant des équations linéaires ou éventuellement quadratiques. Nous introduisons ensuite les notions de complexité d'algorithmes et de problèmes. La suite du dossier est composée de deux grandes parties.*

*Une partie est consacrée à la présentation de méthodes de résolution exacte, certaines sont **polynomiales**, spécifiques du problème « facile » considéré et donc utilisables même pour des problèmes de grandes tailles ; d'autres sont **pseudo-polynomiales** et encore utilisables pour des problèmes de tailles importantes ; enfin, des méthodes exponentielles, construites sur des schémas généraux, appelés **procédures par séparation et évaluation** ne peuvent être utilisées que sur des problèmes de taille relativement restreinte. Ce sont des méthodes exponentielles de ce type qui sont utilisées dans les **solveurs** généraux, à base de programmation linéaire ou de propagation de contraintes, qui sont actuellement disponibles sur le marché des progiciels. L'amélioration progressive de ces solveurs et l'augmentation spectaculaire de la rapidité des ordinateurs permettent de résoudre, avec ces progiciels, des problèmes de taille relativement importante. Néanmoins, la durée des exécutions augmente toujours exponentiellement avec la taille des problèmes.*

*Une deuxième partie est consacrée aux méthodes de résolution approchées, quand on accepte de ne plus avoir la preuve d'obtenir une solution optimale, mais que l'on espère utiliser des approches suffisamment astucieuses pour obtenir de très bonnes solutions. Nous présentons, en particulier, les **méta-heuristiques** les plus connues telles que le recuit simulé, la méthode Taboue et les algorithmes génétiques. Nous incitons également le lecteur à croiser les méthodes de manière à essayer d'obtenir les meilleurs résultats.*

# 1. Exemples de problèmes d'optimisation discrète

Les exemples concrets sur lesquels vont s'appuyer l'ensemble du dossier sont les suivants :

1. un problème d'affectation de personnel ;
2. un problème d'agencement de bureaux ;
3. un problème de choix d'investissements ;
4. des problèmes d'ordonnancement à une machine.

Dans ce paragraphe, nous décrivons l'application concrète correspondante et proposons une modélisation pour chacun d'entre eux.

## 1.1 Problème d'affectation de personnel

### Description du problème

Dans un atelier de conditionnement où toutes les tâches sont des tâches manuelles, $m$ tâches de la durée d'un poste (7 h) ont été constituées par le gestionnaire d'atelier la veille au soir. En début de poste, les employés pointent et certains d'entre eux sont absents pour différentes raisons dont certaines sont non prévisibles (maladie, enfant malade...). Lorsque tous les employés ont pointé, on connaît leur nombre exact $n$ et on les désigne par un indice $j$ variant de 1 à $n$. En fonction de sa qualification, l'employé désigné par l'indice $j$ peut faire ou non la tâche désignée par l'indice $i$. On note $A_{i,j}$ une valeur entière connue qui vaut 1 si l'employé $j$ peut faire la tâche $i$ et 0 dans le cas contraire. Les tâches non exécutées sont reportées au poste suivant et si on n'arrive pas à occuper un employé parce que sa qualification ne lui permet pas de faire une des tâches non exécutées, alors on l'occupe à des tâches non rentables (nettoyage, par exemple). L'intérêt de l'entreprise est d'exécuter un maximum de tâches pendant le poste.

### Modélisation du problème

Nous modélisons le problème sous forme de programmation linéaire en variables entières de type 0-1, en utilisant l'inconnue entière $x_{i,j}$ qui vaut 1 si l'employé $j$ exécute la tâche $i$ et 0 dans le cas contraire.

La **première famille de contraintes** exprime le fait que tout employé $j$ ne peut pas exécuter plus d'une tâche :

$$\sum_{i=1}^{m} (x_{i,j} \leqslant 1), \quad j = 1,\ldots,n \qquad \textbf{(1)}$$

La **deuxième famille de contraintes** exprime le fait que toute tâche $i$ ne doit être exécutée que par au plus un employé :

$$\sum_{j=1}^{n} x_{i,j} \leqslant 1, \quad i = 1,\ldots,m \qquad \textbf{(2)}$$

La **troisième famille de contraintes** exprime le fait qu'un employé $j$ ne peut être affecté qu'à une tâche qu'il sait faire :

$$x_{i,j} \leqslant A_{i,j}, \quad i = 1,\ldots,m, \quad j = 1,\ldots,n \tag{3}$$

Les inconnues sont entières de type 0-1 :

$$x_{i,j} \in \{0, 1\}, \quad i = 1,\ldots, m, \quad j = 1,\ldots, n \tag{4}$$

L'équation linéaire à maximiser représente le nombre total de tâches exécutées :

$$\max \sum_{i=1}^{m} \sum_{j=1}^{n} x_{i,j} \tag{5}$$

Les familles de contraintes (1) et (2) sont appelées des **contraintes d'affectation**. Elles mettent en correspondance les éléments de deux ensembles, mais la correspondance peut ne pas être biunivoque au sens où certains éléments d'un ensemble peuvent ne pas avoir de correspondant dans l'autre ensemble, mais tout élément a au plus un correspondant.

## 1.2 Problème d'agencement de bureaux

### Description du problème

Le département administratif d'une grande entreprise déménage dans un nouveau bâtiment. Tous les bureaux sont situés au même étage et ont la même taille comme, par exemple, sur le plan représenté sur la figure **1**. Il y a $n$ bureaux. Les rectangles noirs représentent les portes et, pour deux bureaux $j_1$ et $j_2$, on connaît la distance entre les portes des bureaux, notée $D_{j_1,j_2}$. On a découpé le département administratif en $n$ services pouvant chacun être abrité par l'un des bureaux de taille identique. Dans les anciens locaux, on a payé une stagiaire pour faire des statistiques et en particulier pour donner une valeur estimée moyenne du nombre de fois où une personne se déplace du service $i_1$ au service $i_2$ au cours d'un mois standard ; cette valeur est notée $M_{i_1,i_2}$. Le problème consiste à attribuer les $n$ bureaux aux $n$ services de manière à minimiser la distance parcourue par l'ensemble des personnes qui vont se déplacer à l'intérieur du département administratif, sachant que les déplacements correspondent à des temps improductifs.

### Modélisation du problème

On peut utiliser des inconnues semblables à celles utilisées pour le problème précédent. Ici, l'inconnue entière $x_{i,j}$ vaut 1 si le service $i$ est placé dans le bureau $j$ et 0 dans le cas contraire.

La **première famille de contraintes** exprime le fait que tout bureau $j$ contient un et un seul service :

$$\sum_{i=1}^{n} x_{i,j} = 1, \quad j = 1,\ldots,n \tag{6}$$

La **deuxième famille de contraintes** exprime le fait que tout service $i$ est placé dans un et un seul bureau :

$$\sum_{j=1}^{n} x_{i,j} = 1, \; i = 1,\ldots,n \tag{7}$$

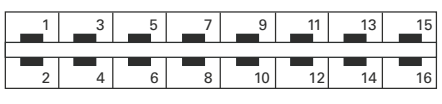



**Figure 1 – Plan possible pour l'implantation de 16 bureaux de taille identique**

Les inconnues sont entières de type 0-1 :

$$x_{i,j} \in \{0, 1\}, \quad i = 1, \ldots, n, \quad j = 1, \ldots, n \tag{8}$$

L'équation quadratique à minimiser représente le nombre total de déplacements de personnes entre les services :

$$\max \sum_{i_1=1}^{n} \sum_{\substack{i_2=1 \\ i_2 \neq i_1}}^{n} \sum_{j_1=1}^{n} \sum_{\substack{j_2=1 \\ j_2 \neq j_1}}^{n} M_{i_1,i_2} D_{j_1,j_2} x_{i_1,j_1} x_{i_2,j_2} \tag{9}$$

Les familles de contraintes (6) et (7) sont des familles de contraintes très souvent utilisées en optimisation discrète. Elles mettent en correspondance biunivoque deux ensembles de même taille de telle sorte qu'à tout élément du premier ensemble corresponde un et un seul élément du deuxième ensemble. Ce sont des **contraintes d'appariement** : il s'agit d'un cas particulier de contraintes d'affectation déjà vues dans le § 1.1 et dont nous verrons d'autres exemples.

## 1.3 Problème de choix d'investissement

### Description du problème

Dans une grande entreprise découpée en branches d'activités, on dispose de $K$ millions d'euros pour réaliser des investissements. Une étude préalable a permis de tabuler une fonction $f_j(x_j)$ qui représente le bénéfice (supplémentaire) obtenu par l'entreprise sur un horizon donné si elle investit $x_j$ millions dans la branche d'activités désignée par l'indice $j$. Une telle fonction tabulée est représentée dans le tableau **1**. Le nombre de millions que l'on peut placer dans la branche d'activité $j$ est au plus $N_j$. Le problème consiste à placer au mieux les $K$ millions dans les $n$ branches d'activité de manière à maximiser le bénéfice total (supplémentaire) ainsi obtenu.

### Modélisation du problème

On utilise ici des inconnues entières positives ou nulles notées $x_i$ qui indiquent tout simplement le nombre entier de millions investis dans la branche d'activité $i$.

La **première contrainte** interdit de dépasser le nombre de millions disponibles :

$$\sum_{j=1}^{n} x_j \leqslant K \tag{10}$$

Les inconnues sont entières et ont un domaine de définition limité :

$$0 \leqslant x_j \leqslant N_j, \quad j = 1,\ldots,n \tag{11}$$

La fonction à maximiser représente le total des bénéfices (supplémentaires) obtenus dans l'ensemble des branches d'activité :

$$\max \sum_{j=1}^{m} f_j(x_j) \tag{12}$$

**Tableau 1 – Exemple de bénéfices à réaliser pour un plan d'investissement** (cf. § 3.2)

| $x_j$ | $j$ | | | |
|---|---|---|---|---|
| | **1** | **2** | **3** | **4** |
| **1** | 9 | 8 | 10 | 9 |
| **2** | 20 | 16 | 19 | 18 |
| **3** | 25 | 30 | 27 | 27 |
| **4** | 27 | | 34 | |

## 1.4 Problèmes d'ordonnancement à une machine

### Description du problème

Un atelier de production ne comporte qu'une seule machine. Cette machine est toujours disponible. Au début de l'horizon, $n$ pièces à traiter sont arrivées et on suppose qu'il n'en arrivera pas d'autres sur l'horizon considéré. Le traitement de la pièce désignée par l'indice $i$ nécessite $p_i$ unités de temps consécutives sur la machine. Si la pièce $i$ n'est pas terminée à l'instant $d_i$, elle est considérée comme en retard. Le problème considéré consiste à choisir le meilleur ordre possible de passage des pièces sur la machine de manière à satisfaire au mieux l'entreprise et/ou ses clients. Cette satisfaction peut être exprimée par différents critères [3].

Le **critère de la durée totale**, c'est-à-dire la date de fin de réalisation de la dernière pièce, est sans intérêt ici, car les traitements sont consécutifs sur la machine (on a tout à perdre à laisser la machine sans traitement avant d'avoir tout fini), alors la durée totale est exactement égale à la somme des durées des traitements et est donc constante.

Le **critère de la somme totale des temps de présence** consiste à faire sortir, en moyenne, les pièces de l'atelier le plus tôt possible. Ce critère est intéressant si on peut faire payer le client au moment où il reçoit la pièce ; il minimise également le nombre de pièces présentes en moyenne dans l'atelier, mais ce critère n'est intéressant pour minimiser les encours moyens que si les pièces arrivent de manière dynamique dans l'atelier à des dates différentes.

Les critères qui permettent de mieux satisfaire les clients consistent à minimiser une fonction qui dépend du **retard des pièces**. Cela peut être, le plus grand retard, le nombre de pièces terminées en retard, l'importance des pièces terminées en retard ou encore la somme des pénalités de retard proportionnelles au retard de la pièce et à la valeur de la pénalité par unité de retard. On parle de retard moyen si toutes les pièces ont la même importance et de retard pondéré si les pénalités de retard dépendent des pièces.

### Modélisation du problème pour différents critères

Une solution de ce problème est décrite par l'ordre de passage des pièces sur la machine. Si on numérote de 1 à $n$ les positions possibles pour les pièces dans la séquence de traitement, il s'agit à nouveau de mettre en correspondance biunivoque l'ensemble des positions et l'ensemble des pièces. On peut donc utiliser les mêmes inconnues et les mêmes familles de contraintes que pour le problème du § 1.3, seule la formulation de l'objectif à minimiser pièce à la position $j$ que l'on note $C_j$. La famille d'équations (16) expriment le lien entre les inconnues $x_{i,j}$ et les inconnues $C_j$.

$$C_j = \sum_{k=1}^{j} \sum_{i=1}^{n} p_i x_{i,k}, \quad j = 1,\dots,n \tag{16}$$

Le critère somme totale des temps de présence s'exprime au moyen de l'équation linéaire (17) :

$$\min \sum_{j=1}^{n} C_j \tag{17}$$

— Pour exprimer les critères liés au retard, il est préférable d'introduire une famille d'inconnues redondantes, notées $T_j$, de même type que les $C_j$, mais positives ou nulles et qui représentent le retard des pièces non terminées avant leur délai $d_j$. En supposant que les critères utilisés imposent que l'on minimise autant que faire se peut les $C_j$, on peut exprimer le calcul des $T_j$ à partir des $C_j$ au moyen de deux familles de contraintes linéaires (18) et (19) :

$$C_j - \sum_{i=1}^{n} x_{i,j} d_i \leqslant T_j, \quad j = 1,\dots,n \tag{18}$$

$$0 \leqslant T_j, \quad j = 1,\dots,n \tag{19}$$

Le critère somme des retards s'exprime au moyen de l'équation linéaire (20) :

$$\min \sum_{j=1}^{n} T_j \tag{20}$$

— Si on s'intéresse au nombre de pièces en retard, il convient d'ajouter une nouvelle famille d'inconnues redondantes, notées $U_j$, de type entier à valeur 0 ou 1, qui vaut 1 si la pièce en position $j$ est terminée strictement en retard et 0 dans le cas contraire. Avec la même hypothèse que ci-dessus, le calcul des $U_j$ à partir des $T_j$ se fait au moyen de la famille de contraintes linéaires (21) :

$$MU_j \geqslant T_j, \quad j = 1,\dots,n \tag{21}$$

avec $M$ nombre plus grand que le plus grand retard possible (par exemple, la somme de toutes les durées des traitements des pièces).

L'équation (22) permet d'exprimer le critère somme des tâches en retard :

Référence Internet
S7212

# Optimisation du placement des formes irrégulières

par **Salah MAOUCHE**
*Professeur à l'Université des sciences et technologies de Lille (USTL)*
*Laboratoire d'automatique $I^3D$ (Interaction, Image et Ingénierie de la Décision)*

**Catherine K. BOUNSAYTHIP**
*Docteur de l'USTL*
*VTT Information Technology, Finlande*

et **Gilles ROUSSEL**
*Maître de conférences à l'université du Littoral-Côte-d'Opale,*
*Laboratoire d'analyse des systèmes du Littoral (LASL)*



*Le **placement** fait partie du problème de **découpe** rencontré chaque fois que la fabrication d'un objet manufacturé est réalisée par **transformation de la matière**. De ce fait, il concerne un grand nombre d'industries, dont les industries du métal, du bois, du verre, de la confection et du cuir.*





Décembre 2000

*La fabrication d'un objet est souvent conduite avec l'objectif d'une réduction du coût de revient à travers la réduction de la consommation en matière. Cette tendance est en particulier liée aux augmentations de prix des matières premières. Aussi, en général, l'**exploitation optimale des ressources** constitue une préoccupation croissante dans les industries manufacturières.*

*Une formulation industrielle du problème de découpe en deux dimensions peut se présenter comme suit : « Étant donné une matière première présentée sous forme de plusieurs unités de dimensions et formes éventuellement différentes, comment produire une quantité de pièces, en fonction de la demande et des niveaux de stocks, en utilisant le minimum de matière, et ce dans un temps compatible avec les délais fixés par le client. »*

*La quantité de pièces à produire peut être connue ou non à l'avance. Si l'objectif principal consiste à minimiser la consommation en matière première, il est d'actualité de produire vite pour satisfaire des délais de livraison impératifs et de plus en plus courts, au « juste-à-temps » pour éviter les frais de stockage.*

*Une demande établie à partir d'une liste de pièces et d'une certaine quantité de matière première constitue la donnée initiale du processus. Les contraintes sont définies comme étant les restrictions imposées au processus pour tenir compte des propriétés de la matière, de la qualité des pièces, du mode de découpe, de l'état des stocks, etc.*

*Le **problème du placement** consiste à rechercher le meilleur amalgame au sens des objectifs du placement, et dans le problème de découpe, il s'agit de trouver un ensemble d'amalgames pour satisfaire les demandes. Un amalgame est la manière de découper une unité d'une matière première. Le placement constitue la partie la plus importante du problème de découpe et on ne peut résoudre efficacement un problème de découpe sans résoudre efficacement celui du placement.*

*Le placement fait partie des problèmes d'**optimisation combinatoire** qui suscitent beaucoup d'intérêt. Malgré les progrès considérables de l'outil informatique, les méthodes d'énumération, exhaustive ou partielle, sont encore peu satisfaisantes en termes de temps d'exécution ou d'efficacité. Comme ces problèmes contiennent souvent beaucoup de solutions à intérêts pratiques acceptables, les recherches sont orientées vers le développement des **méthodes heuristiques**. Le but est de trouver une solution de qualité satisfaisante en un temps de calcul raisonnable, d'autant plus que pour des problèmes réels, il n'est pas toujours impératif de trouver la solution optimale, mais des solutions dont la qualité et le temps mis pour les obtenir restent acceptables.*

# 1. Considérations générales

## 1.1 Problème de placement

Le domaine du placement de formes quelconques reste largement ouvert. La difficulté majeure de ce dernier tient à l'incapacité actuelle à trouver des caractéristiques générales aux formes, incapacité qui contraint les utilisateurs à adopter des critères souvent subjectifs et donc restrictifs. Il apparaît dès lors que les phases les plus importantes dans la résolution d'un problème de placement de formes quelconques sont celles du codage et de la représentation.

### Codage des formes

Le placement automatique des formes irrégulières pose la question importante de la représentation des objets à placer. Nous présentons le **codage par peignes de contour**. Les peignes représentent la déficience rectangulaire d'une forme. Cette déficience est utilisée pour assembler les formes qui se marient le mieux : nous parlons alors de **complémentarité des formes**.

### Représentation du problème

La complexité du problème nous a amenés à décomposer le problème du placement de plusieurs formes en **placement par bandes**. Chaque bande est construite par imbrication de formes deux à deux. Ensuite, nous présentons les outils nécessaires à la construction d'un placement, en particulier l'opérateur de concaténation de deux formes qui, lorsqu'il est utilisé itérativement, permet la construction d'un placement en général ou d'une bande quand une contrainte sur les dimensions du placement est introduite. Le problème du placement est représenté par un **arbre** dont le parcours permet de trouver des solutions respectant les contraintes. Les critères d'évaluation des solutions sont fournis par les objectifs d'optimisation ; ils permettent de caractériser et de définir la meilleure solution.

Dans le cas des entreprises de confection, les pertes en matière ne sont pas contrôlées rigoureusement, alors qu'elles peuvent représenter jusqu'à 25 % des dépenses sur la matière. On estime qu'une diminution des pertes de 2 % peut faire augmenter de 10 % le bénéfice de l'entreprise.

## 1.2 Résolution

Nous proposons l'étude de trois méthodes heuristiques pour résoudre le problème du placement de formes irrégulières. Le but final est de constituer une bibliothèque d'algorithmes à utiliser selon le choix de l'utilisateur (carnet de commande, critère d'optimalité, temps de calcul...). Les analyses des résultats et les comportements des différents algorithmes permettront de conclure sur l'efficacité de chaque algorithme en fonction des critères à optimiser.

Un algorithme de placement doit être en mesure de fournir le placement d'un sous-ensemble de formes. Il cherche à optimiser une fonction de but visant à minimiser les pertes de matière, tout en respectant un temps de calcul acceptable. Les trois types d'approches étudiées ici sont **les recherches en arbre**, **le recuit simulé** et **les méthodes évolutionnistes**.

**Recherches heuristiques en arbre**

Les recherches en arbres sont des méthodes déterministes issues de l'algorithme $A^*$ (§ 4.3). La recherche est basée sur le principe du « meilleur d'abord » dans lequel la sélection d'un chemin à parcourir repose sur l'estimation de son coût futur. La difficulté de cette stratégie réside dans cette estimation. En effet, la condition pour que la recherche aboutisse à une solution optimale est que celle-ci soit inférieure ou égale au coût optimal. Or, le coût optimal n'est pas connu à l'avance. Il existe alors deux possibilités.

- La première opte pour le chemin de coût toujours largement inférieur au coût optimal estimé, c'est le meilleur coût obtenu depuis le début de la recherche (il est mis à jour au fur et à mesure des nouvelles découvertes) ; l'utilisateur peut alors fixer un écart de tolérance par rapport à cet optimal estimé. C'est le principe de l'algorithme $A^*_\varepsilon$ (§ 4.4).
- La deuxième autorise le choix d'un chemin dont le coût surestime légèrement l'optimal sans trop s'éloigner en valeur supérieure, dont le seuil est fixé par un paramètre $\delta$ ; c'est la variante $R^*_{\delta\varepsilon}$.

Ces méthodes nécessitent une évaluation systématique à chaque étape de la recherche. Parfois, lorsqu'il n'est pas possible d'énumérer tous les états, les préférences sont orientées vers les méthodes aléatoires. Mais ces dernières peuvent ne jamais aboutir à une solution, c'est pourquoi il est nécessaire de les *informer*. Dès lors, elles ne sont plus tout à fait aléatoires, mais « semi-aléatoires » puisqu'elles utilisent les informations enregistrées au cours de la recherche. Les méthodes semi-aléatoires ne font pas un examen systématique de toutes les possibilités à chaque étape et leur recherche est guidée soit par une probabilité d'acceptation d'un état (recuit simulé), soit par le maintien d'un ensemble de points potentiels (algorithmes évolutionnistes).

**Algorithme du recuit simulé**

Le recuit simulé peut être comparé à la méthode de descente du gradient avec possibilité de sortir des optima locaux. L'algorithme est basé sur le principe thermodynamique dont un des paramètres de contrôle est assimilé à la température. À haute température, les molécules ont une plus grande mobilité et peuvent occuper plusieurs configurations, même non stables. Au fur et à mesure que la température diminue, la structure des molécules se fige peu à peu pour être définitivement bloquée dans une configuration quelconque. Un nouveau réchauffement, le *recuit*, permet de déformer cette configuration. Le processus de refroidissement doit être très lent afin de permettre au système d'atteindre un état stable à énergie minimale, car si la température décroît trop rapidement, la recherche s'arrête sans avoir eu le temps de converger vers un optimum ; la recherche peut ne pas atteindre un état minimal global.

**Algorithme évolutionniste**

La méthode évolutionniste est une métaphore du principe de la sélection naturelle et de la loi de survie du plus fort de Darwin. Elle utilise un ensemble (ou *population*) de solutions, la loi de sélection et les opérateurs de transformation qui sont, entre autres, le croisement et la mutation. Une solution au problème est *codée* en terme de *chromosome*, appelé parfois aussi *génotype*, dont le décodage donne le *phénotype*. Les chromosomes sont croisés entre eux ou mutés pour générer d'autres chromosomes. Pour éviter la génération de solutions invalides, l'idée est d'incorporer dans les opérateurs de croisement ou de mutation des contraintes ou des heuristiques.

# 2. Description des formes irrégulières

Comment passer d'un espace où le placeur humain manipule des gabarits en carton à une **représentation numérique pour calculateur** par exemple, dans le cadre du placement de patrons de vêtements ? Plusieurs étapes de traitement sont nécessaires pour résoudre ce problème.

## 2.1 Discrétisation du contour

En dehors des éventuels points de raccords imposés, seul le contour est nécessaire à la connaissance des objets à placer. Le contour est alors défini à l'aide d'un outil d'édition de dessin ou extrait de la base de données d'un système de CAO (conception assistée par ordinateur) de l'atelier de conception. Il peut également être le résultat d'un algorithme de recherche de contour à partir de l'image d'un gabarit issu d'une caméra ou d'un scanner.

Dans tous les cas, le contour est discrétisé relativement au pavage (carré, hexagonal...) du support discret dans lequel il a été défini. Le contour d'une forme $F_k$ est alors considéré comme une courbe fermée discrète 8-connexe (c'est-à-dire chaque point du support possède huit voisins) codée par une liste $C_k$ de $n$ points $\pi_q\,(x_q,\ y_q)$ exprimés dans le repère d'édition. La liste $C_k = ((x_0, y_0), (x_1, y_1), (x_q, y_q) \ldots (x_n, y_n))$ respecte le principe d'unilatéralité par rapport au contour continu initial.

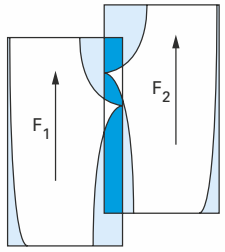
## 2.2 Choix d'un modèle de description

Le modèle établi précédemment constitue un codage de base pour l'affichage, mais ne présente pas une structuration adéquate pour effectuer les traitements géométriques. En outre, les codages de contour potentiellement utilisables doivent être analysés en tenant compte de la réversibilité de la transformation et de l'adéquation aux opérateurs de traitement. Un codage exact est préférable à un codage « approximant », mais ce dernier peut être acceptable s'il n'altère pas le respect des contraintes ni les rendements de placement de façon significative. De nombreux types de codages de contours sont proposés dans la littérature. Bien qu'étant appropriés pour la reconnaissance de formes, l'infographie ou la caractérisation, tous ne sont pas utilisables pour la réalisation du « placement constructif » (c'est-à-dire par ajouts successifs de formes), en raison de l'absence d'opérateurs géométriques d'imbrication.

Lorsque toutes les formes sont parallèles à un axe privilégié, le placement est dit *paraxial*. Même si les contours sont disjoints, leurs rectangles circonscrits respectifs peuvent se chevaucher (figure **1**). Cette propriété rencontrée dans le placement textile suggère un type de code particulier permettant une décomposition des formes selon quatre côtés : les *peignes de contour*.

### 2.2.1 Peignes de contour

Ce code est d'une grande efficacité en placement constructif paraxial. Le principe en est ici résumé, mais il est décrit complètement dans [1].




Les flèches indiquent le droit-fil

**Figure 1 – Chevauchement entre rectangles circonscrits**

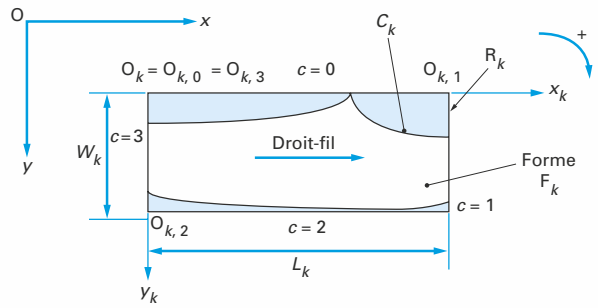
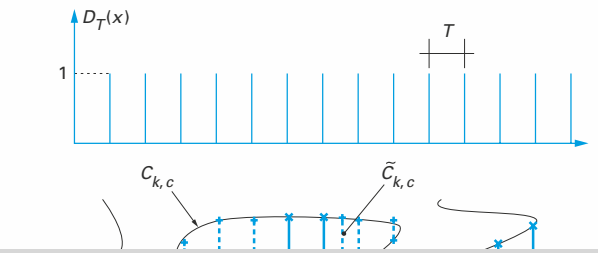



**Figure 2 – Paramètres caractéristiques de forme**




**Étape 2 : détermination du peigne complet**

Le peigne $D_{k,c}$ du sous-contour $c$ de la forme $F_k$ correspond à l'échantillonnage spatial de la courbe analogique du sous-contour visible $\tilde{C}_{k,c}$ (c'est-à-dire le contour $C_{k,c}$ sans les concavités non visibles) par un peigne de Dirac $D_T(x)$ de période $T$ (figure **3**) :

$$D_{k,c} = \{D_{k,c}(1), \ldots, D_{k,c}(\ell_{k,c})\} = \text{Int}\,[(\tilde{C}_{k,c}(x)\, D_T(x)]$$

avec Int [ ] la partie entière,

$\ell_{k,c}$ la longueur du côté $c$ du rectangle circonscrit,

$D_{k,c}(j)$ la hauteur de la dent d'indice $j$.

**Étape 3 facultative : détermination du peigne réduit**

Cela consiste à *fusionner* les sommets de concavité (c'est-à-dire supprimer le point B pour que l'arc [ABC] devienne le segment [AB]. On supprime alors les dents correspondant aux points à fusionner jusqu'à ce que l'augmentation de surface du peigne atteigne un seuil fixé. Seuls les points de concavité sont supprimés pour conserver le respect de la contrainte de disjonction des contours analogiques. Cette étape permet de limiter la quantité de données associées à chaque forme, et même de réduire le traitement mathématique des opérateurs géométriques définis dans la suite. L'absence de description des parties concaves non visibles peut cependant représenter un obstacle dans certains cas de placement de contours en puzzle.

## 2.2.2 Opérateurs géométriques d'imbrication

Les deux opérateurs présentés dans cette partie, la concaténation et l'imbrication locale optimale, représentent le noyau de calcul d'optimisation géométrique locale. Ces opérateurs sont appliqués en réponse à une décision prise au niveau d'un algorithme d'optimisation combinatoire globale. Ce dernier, guidé par un objectif de minimisation des pertes, construit progressivement le placement en ajoutant itérativement une forme au placement courant.

C'est le principe du *placement constructif*. Chaque décision d'ajout d'une forme au placement à l'ordre $n$ s'accompagne donc de la séquence suivante :

— étape 1 : recherche de la position optimale de $F_k$ par rapport au placement courant $P\ell_n$. On applique la fonction :

$$\text{Position\_opt} = \text{Imbrication}\,(F_k, P\ell_n)\ ;$$

— étape 2 : Concaténation de la forme $F_k$ avec $P\ell_n$ dans la position optimale, en appliquant la fonction :

$$P\ell_{\ldots} = \text{Concaténation}\,(F_k, P\ell_{\ldots}, \text{Position\_opt})$$

Référence Internet
S7214

# Optimisation multicritères

# Application dans l'industrie du bâtiment

*par* **Igal M. SHOHET**
*et* **Eldad PERELSTEIN**
*Faculty of Civil Engineering*
*Technion, Israel Institute of Technology*

*Traduction de l'anglais par* Anne-Marie GAULIER



*Un des problèmes majeurs des organismes institutionnels possédant une quantité importante de bâtiments est la contrainte d'un budget limité pour la maintenance et la réhabilitation. Ils rencontrent le dilemme de* ***répartir, entre différents projets, des ressources financières limitées****. Ce problème peut être formulé comme suit :*

*— parmi un grand nombre de projets potentiels de réhabilitation : lequel choisir ?*

*— quel est le niveau « optimal » d'intervention dans chaque projet qui maximiserait le profit total (en termes de coût, de durée de vie, de niveau de performance et de logistique) ?*

*Le problème a été décomposé en deux phases :*

*— développer une méthodologie pour déterminer les ratios coût/profit de plusieurs modes d'intervention différents dans chaque projet considéré séparément (****étude intraprojet****) ;*

*— développer un algorithme d'optimisation pour déterminer les priorités de l'allocation des ressources parmi différents projets, avec une fonction d'objectif qui mènera à maximiser les profits totaux (****étude interprojets****).*

*Le modèle proposé pour la solution de ce problème comprend quatre niveaux :*

*(1) évaluation systématique de l'état de l'installation (adéquation physique et fonctionnelle) ;*

*(2) génération de différentes solutions alternatives de réhabilitation ou de revalorisation basées sur les résultats de l'évaluation de cet état ;*

*(3) établissement de tables coût/profit pour chaque projet avec mise en évidence des différents niveaux d'intervention, de leurs coûts et des profits correspondants ;*

*(4) répartition des ressources financières entre les alternatives les plus intéressantes, c'est-à-dire celles qui garantissent la maximisation du profit total pour un budget total donné. Le mécanisme de recherche proposé utilise la* ***programmation dynamique*** *pour déterminer les configurations proches de l'optimum.*

*Nous conclurons l'article avec un exemple représentatif qui décrit le module en 4 phases et les critères composites de détermination des priorités.*

# 1. Domaine d'application de l'article

Le domaine de la maintenance, de la réhabilitation, de la rénovation et de la revalorisation des installations présente un large champ d'applications de critères décisionnels, aux différentes phases de la vie du projet.

Une installation-type occupée par un organisme voit sa maintenance habituellement en accord avec le budget disponible et le niveau souhaité de performance. À partir d'un certain niveau de délabrement, les coûts de maintenance pour faire fonctionner l'installation s'élèvent, tandis que l'état physique et fonctionnel de cette installation se détériore, diminuant son niveau de performance. Les grands organismes privés et institutionnels possédant un parc important d'installations (c'est-à-dire des universités, des hôpitaux, etc.) se trouvent face au problème de **fixer des priorités aux projets de maintenance et de réhabilitation**, cela à deux niveaux :

1) parmi une grande quantité de projets potentiels : lesquels choisir afin de maximiser (en termes de performance fonctionnelle) le profit total ?

2) quel est le niveau d'investissement pour chaque projet qui maximiserait le profit total ?

Ces problèmes incluent l'optimisation multivariables qui associe : (1) l'investissement financier, (2) le choix des constituants du bâtiment à maintenir ou à réhabiliter, (3) la détermination de la durée du projet, (4) la logistique temporaire, (5) le niveau de service.

La solution souhaitée doit aussi satisfaire un large spectre de contraintes : (1) des limitations de budget, (2) un cadre temporel, (3) le niveau de performance requis pour chaque installation, (4) le cycle de vie requis du projet pour chaque installation, (5) tout en restant dans un budget de maintenance annuel limité.

Les méthodes PERT (*Program Evaluation and Review Technique*) et CPM (*combinatorial pattern matching*) fournissent des outils pour résoudre les problèmes dans lesquels des ressources multiples (telles que le temps, le budget, etc.) sont allouées à des activités de projets multiples. Ces techniques n'ont cependant pas la propriété générale d'allouer **simultanément** différents types de ressources à de nombreux projets, le but étant d'optimiser une fonction d'objectif [9]. La méthode présentée ici est formulée dans le cadre de problèmes d'optimisation ; ce type de solution a été développé dans [1], [5], [11]. Cependant, toutes ces solutions ont traité le problème seulement dans la première phase, c'est-à-dire la sélection d'une méthode au niveau d'un projet pris individuellement. Un autre travail mené sur ce thème a traité le problème seulement au second niveau, c'est-à-dire la définition des priorités entre différents types d'installation [5].

Le présent article présente une formulation d'optimisation résolvant le problème **simultanément** dans les deux phases. La solution est basée sur un **algorithme hiérarchisé multiphase** :

— évaluation de l'installation (ce module est décrit dans [10]);
— génération d'alternatives de réhabilitation réalisables au niveau d'un projet pris séparément (cela est décrit dans [8]);
— attribution de ressources limitées aux différents projets de façon à maximiser le profit total résultant de la stratégie d'attribution.

Le présent article est consacré à la troisième phase de la solution. Il est basé sur les analyses des deux étapes antérieures, et utilise la stratégie d'élimination et la programmation dynamique pour déterminer le domaine des solutions possibles, puis la sélection d'une solution quasi optimale suivant des critères composites.

# 2. Détermination des priorités dans la maintenance et réhabilitation de bâtiments

Le développement continu de la construction et la part croissante des systèmes électriques et mécaniques dans les bâtiments ont augmenté la complexité des bâtiments et leur confort d'occupation. On constate une tendance constante, au niveau mondial, pour restreindre les budgets de maintenance, et cette situation soulève le problème de fixer des priorités aux travaux de maintenance et de réhabilitation. Ce sujet a été le thème essentiel des recherches dans ce domaine depuis les dix dernières années. Les méthodes et la recherche dans ce domaine peuvent être classées de deux façons principales : (a) les outils heuristiques et (b) les outils quantitatifs. Nous donnons ci-dessous une liste de trois méthodes sélectionnées.

### ■ Système expert à logique floue [4]

Dans cette méthode, les utilisateurs doivent fournir au système deux types d'informations :

a) donner la probabilité d'existence des facteurs choisis, qui composent les critères de priorité (cette probabilité est basée sur l'expérience personnelle de l'utilisateur) ;

b) fournir l'information sur les activités candidates pour l'attribution de ressources.

Cette méthode ne peut pas être implémentée pour la fixation des priorités de maintenance puisque la solution n'est pas basée sur des mesures quantitatives.

**Modèle d'indice multicomposite** [3]

Ce modèle se propose de considérer différentes options de réhabilitation, en donnant à chaque option un score composé d'un certain nombre de critères.

Ce modèle convient à la fois pour des critères quantitatifs et qualitatifs. Il est cependant compliqué et nécessite un haut niveau d'expertise de la part des utilisateurs, s'il n'y a pas de programme informatique approprié.

**Approche multiattributs** [11]

Dans cette approche, chaque travail de maintenance dans le bâtiment est enregistré et un indice de priorité lui est donné suivant les critères acceptés par l'utilisateur. L'indice de priorité d'un certain travail de maintenance est déterminé en ajoutant les produits du poids relatif du critère et du score donné. Après qu'un indice de priorité a été donné à chaque activité, ces indices sont classés, puis le coût cumulé des activités est calculé.

Quand le coût cumulé des activités dans la liste des priorités atteint le montant alloué, une ligne est tirée et toutes les activités au-dessus de ce trait devront être menées suivant leur priorité. Cette approche souffre de deux défauts majeurs :

a) L'utilisation de poids relatifs peut être justifiée quand l'estimation est faite au niveau d'un bâtiment unique. Cependant, si l'estimation inclut plusieurs bâtiments, avec la nécessité de les comparer entre eux, les poids doivent être classés, suivant une échelle uniforme pour toutes les unités de bâtiments.

b) Parfois, il peut y avoir un cas où deux activités de maintenance, dépendant l'une de l'autre, sont classées – l'une au début de la liste et l'autre à la fin. Cette approche ne permet pas de lier ensemble ces deux activités dans le processus de décision.

Cette liste de méthodes met en évidence le fait qu'une recherche complémentaire est nécessaire dans le développement de critères quantitatifs pour fixer les priorités de maintenance. Elle montre aussi que ce problème doit être traité avec une perception globale du bâtiment, fournissant ainsi une vision complète de la maintenance du bâtiment dans son entier plutôt que traiter les activités comme un ensemble d'éléments indépendants. Un autre point qui doit être défini, lors de la résolution du problème de fixation des

Cette dernière définition est centrée sur la minimisation du coût et convient aux situations où l'objectif principal de l'organisation ou de la division est de minimiser les coûts totaux à cause des incertitudes ou des réductions de budget.

La première catégorie mentionnée convient bien à une situation plus courante où le propriétaire de l'installation cherche à maximiser les profits, à l'intérieur d'un budget total donné.

L'approche présentée ici suit la première formulation, bien que la seconde forme puisse aussi être utilisée, selon les objectifs principaux du propriétaire.

## 4. Modèle mathématique

La formulation mathématique du problème de fixation des priorités est la suivante.

Soit $X_i$ le montant de l'investissement dans le $i^{ème}$ projet et soit $F^*$ l'allocation optimale de ressources dans le $n^{ième}$ projet qui autorise la maximisation du profit de la totalité des projets. Le problème d'optimisation peut être formulé comme suit : trouver $X_n(x_1,x_2,x_3, \ldots, x_n)$ qui maximise le profit total en termes de performance, avec les contraintes suivantes :

1) limitations du budget des projets de réhabilitation ;

2) budget de maintenance : une situation typique dans les divisions de maintenance des organisations dans le monde entier est la disponibilité de budgets limités dans le cadre desquels les décideurs doivent opérer ;

3) niveau de performance : c'est une autre exigence qui doit être remplie relativement à la destination de l'installation. De façon à atteindre un ratio profit /coût élevé, le niveau de performance doit être adapté à l'utilité et à l'importance de l'installation ;

4) espérance de vie du projet : l'espérance du cycle de vie de l'installation rénovée ne devrait pas excéder l'espérance de vie de l'installation telle que déduite des plans comme la stratégie globale ;

5) temps limité pour l'achèvement du projet : l'utilisateur a aussi besoin que l'installation rénovée soit achevée dans un temps limité pour répondre à des nécessités opérationnelles ou commerciales.

Le problème est donc formulé de la façon suivante :

fonction d'objectif :

$$F^* = \max\left\{ \sum_{n=1}^{N} f_n(X_n) \right\} \quad (1)$$

Référence Internet
S7240

# GRAFCET

# Concepts de base

*par* **Jean-Jacques DUMÉRY**
*Docteur de l'École Centrale Paris*



*On observe que, pour décrire le comportement des systèmes, les représentations graphiques sont de plus en plus privilégiées par rapport aux représentations littérales et algébriques, du fait de leur aptitude à servir de support de communication entre tous les intervenants (spécificateur, concepteur, réalisateur, exploitant, mainteneur). Le fait que tous ces intervenants puissent dialoguer et exprimer leur point de vue dans le cadre d'une démarche de développement d'un système automatique est en soi une aide considérable, d'abord pour spécifier, puis pour effectuer des validations (partielles) de cette spécification. C'est*





Mars 2007

*dans cet esprit que le GRAFCET (Graphic fonctionnel de commande étapes/transitions) a été conçu. Ce langage de spécification permet la description fonctionnelle du comportement de la* ***partie séquentielle des systèmes de commande****. Il est normalisé, enseigné dans l'enseignement secondaire et supérieur, et utilisé dans l'industrie. Cette norme a fait l'objet d'une révision publiée en 2002 pour adapter celle-ci aux nouveaux besoins des utilisateurs.*

*Ce présent texte est la nouvelle édition du dossier rédigé par Paul BRARD, dont il reprend certains passages.*

*Ce sujet fait l'objet de deux dossiers, le lecteur doit donc consulter également le fascicule [S 7 241] qui complète le présent dossier.*

# 1. Évolution

## 1.1 Création du GRAFCET

En 1975, un groupe de travail de l'AFCET (nota **1**) animé par Michel Blanchard, constitué d'industriels et d'universitaires, intitulé « Systèmes Logiques », créa une commission « Normalisation de la représentation du cahier des charges d'un automatisme logique ». Il s'agissait de définir un « formalisme » simple, adapté à la représentation des évolutions séquentielles d'un système, compréhensible à la fois par les spécificateurs et les exploitants, et fournissant potentiellement des facilités de passage à une réalisation matérielle et (ou) logicielle de l'automatisme. Au début, le travail consista à dresser un état de l'art des différentes approches de modélisation du comportement de tels automatismes. Ces travaux aboutirent à la définition du GRAFCET, ainsi nommé pour, à la fois marquer l'origine de ce nouvel outil de modélisation (grAFCET) et son identité (GRAphe Fonctionnel de Commande Étapes-Transitions).

**Nota 1 :** Association Française pour la Cybernétique Économique et Technique, renommé en Association Française des Sciences et Technologies de l'Information et des Systèmes. Association créée en 1968 et dont l'objectif est d'aider aux développements de ces nouvelles techniques.

## 1.2 Diffusion et normalisation

La première publication officielle fut réalisée dans la revue « Automatique et Informatique Industrielle » en décembre 1977, date considérée par la communauté comme date de naissance effective du GRAFCET. Dès 1979, ce travail a été diffusé auprès de nombreux universitaires, enseignants et industriels, et présenté à une commission de normalisation AFNOR. Le GRAFCET fit alors l'objet d'une norme AFNOR en 1982 (NFC 03-190) et devint un standard de fait dans l'enseignement.

Le groupe « Systèmes Logiques » s'est ensuite intéressé à la prise en compte des premiers retours d'expériences industrielles d'utilisation de ce langage de spécification, il proposa alors quelques extensions. Le GRAFCET devint une norme internationale en 1988 (CEI 848) sous le nom de **« Sequential Function Chart » (SFC)** ce qui contribua à renforcer sa diffusion à l'étranger.

# 2. Intérêt de la dernière révision de la norme

Depuis de nombreuses années, les spécificateurs étaient confrontés à de nombreuses difficultés liées :

— au nombre croissant d'entrées et de sorties à gérer par les systèmes de commande ;

— aux comportements séquentiels complexes nécessitant la définition de nombreux états de fonctionnement ;

— aux interfaces hommes – machines très développées ;

— aux nombreuses procédures d'exploitation et de maintenance (commandes locales et distantes, changement de campagne de production, différentes recettes dans les procédés continus et batch, procédures liées à la sûreté de fonctionnement, ...) ;

— aux comportements associant du combinatoire, du séquentiel et du continu.

Ainsi, une révision de la norme GRAFCET fut demandée par les utilisateurs pour clarifier certaines notions et pour enrichir cet outil de spécification normalisé par de nouveaux concepts permettant une description structurée et hiérarchisée du comportement de la partie séquentielle des systèmes de commande.

Cette révision (CEI 60848) publiée en février 2002 [1] a permis également d'ajouter, aux aspects descriptifs et fonctionnels de la première édition, les aspects formels et comportementaux essentiels à la définition d'un véritable langage de spécification.

Le lecteur pourra se reporter aux références [5] [6] [7] [8] pour compléter son information sur l'évolution des différents concepts depuis la parution de la première norme GRAFCET.

# 3. Domaine d'application et objet de la norme

La norme internationale CEI 60848 est destinée principalement aux spécificateurs, concepteurs, réalisateurs, agents d'exploitation et de maintenance qui ont besoin de spécifier le comportement d'un système (commande d'une machine automatique, composant de sûreté, etc). Ce langage de spécification est un support de communication privilégié entre les concepteurs et les utilisateurs de systèmes automatisés.

Le langage de spécification GRAFCET permet la description fonctionnelle du comportement de la partie séquentielle des systèmes de commande.

La norme définit les symboles et les règles nécessaires à la représentation graphique de ce langage, ainsi que l'interprétation qui en est faite. Elle ne donne pas de méthodes d'utilisation des notions et concepts définis.

Elle a été établie pour les systèmes automatiques qui relèvent d'applications industrielles ou grand public, mais aucun champ d'application n'est exclu.

Les méthodes permettant le passage d'une spécification utilisant le GRAFCET à une réalisation câblée et (ou) programmée ne font pas partie du domaine d'application de cette norme.

Dans le cas des systèmes de commande intégrant un automate programmable, il est important de remarquer que depuis 1993, la norme CEI 61131-3 [2] définissant un ensemble de langages de programmation destinés aux automates programmables peut être utilisée. Il est possible alors de faire appel au langage « SFC » décrit dans cette norme comme méthode de réalisation de la spécification GRAFCET.

# 4. Généralités et définitions

## 4.1 Description d'un système automatique

La fonction globale d'un système automatique (nota **2**) est d'apporter une valeur ajoutée à un ensemble de biens dans un environnement donné (figure **1**).

**Nota 2 :** la notion de système automatisé de production (systèmes mécanisés puis automatisés) est progressivement remplacée par celle de système automatique, dans la mesure où les systèmes considérés sont conçus dès le départ du projet de développement comme intégrant des automatismes. Par ailleurs, ils ne concernent pas que la production.

Sa définition en conception ou son analyse nécessite la description :

— des fonctions qu'il assure ;
— de sa structure matérielle et logicielle ;
— de son comportement ;
— des données traitées et échangées ;
— ainsi que de son environnement physique et humain.

Les fonctions assurées peuvent être partiellement ou totalement réalisées automatiquement, donc avec ou sans intervention humaine.

Les biens sur lesquels la valeur ajoutée est apportée peuvent être : des matières premières, des biens intermédiaires ou matières d'œuvre, des articles manufacturés, de l'énergie sous de multiples formes, de l'information (créée, mise en forme, traitée ou gérée, distribuée ou reproduite), ou encore des services.

Tous ces biens sont appelés produits. Un produit est, selon la norme NF EN 1325 [3] [4] ce qui est, ou sera, fourni à un utilisateur pour répondre à son besoin.

### Structure d'un système automatique

Il est possible de distinguer dans tout système automatique deux parties interdépendantes (figure **2**) :

— la **partie opérative** (PO) qui est le processus physique automatisé et qui réalise les opérations sur le flux de produits permettant l'apport de la valeur ajoutée ;
— la **partie commande** (PC) qui coordonne la succession des actions de la partie opérative, permet la communication avec les utilisateurs et les autres parties commandes.

La description du système (fonctions, comportement, structure, données) en spécification et en conception nécessite au préalable la définition d'une **frontière d'isolement** entre la partie opérative et la partie commande. Dans le cadre d'un projet de développement d'un système automatique, ce choix délimite la prestation.

Cette frontière est affinée au cours des activités de conception, elle caractérise les différents points de vue portés sur le système. Selon le niveau de description adopté : un robot pourra être considéré comme une partie opérative pour le concepteur de sa commande globale, mais un axe particulier pourra également être vu comme la partie opérative de sa commande d'axe.

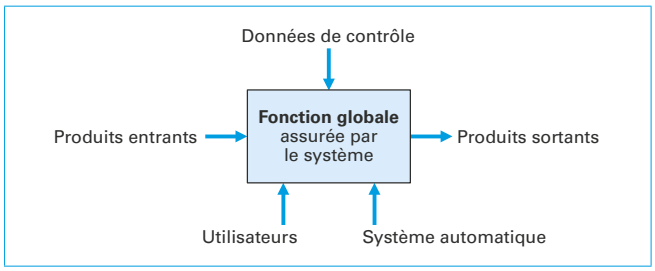



**Figure 1 – Système automatique**

Il est communément admis que la partie commande assure les fonctions suivantes :

— acquérir l'information (issue de la partie opérative, des autres parties commandes et des utilisateurs) ;
— traiter l'information ;
— communiquer l'information ;
— commander (les préactionneurs et actionneurs).

De même la partie opérative permet :

— de distribuer l'énergie grâce aux préactionneurs tels que les distributeurs, les contacteurs, relais, variateurs ;
— de la convertir (par exemple par les moteurs, les vérins, les résistances) ;
— de la transmettre (par exemple dans une chaîne d'action électromécanique : chaîne cinématique, effecteur).

## 4.2 GRAFCET langage de spécification du comportement de la partie commande

**Convention d'écriture**

La convention généralement admise est :

— GRAFCET lorsqu'il s'agit du langage de spécification ;
— grafcet lorsqu'il s'agit du modèle élaboré grâce au langage GRAFCET.

La frontière de description du comportement d'un système automatique étant définie, il est alors possible de faire un bilan des entrées et des sorties traitées par la partie commande. Ces entrées et ces sorties (figure **3**) peuvent être logiques, analogiques ou numériques.

Le langage de spécification GRAFCET permet d'établir un grafcet décrivant le comportement attendu de la partie séquentielle d'un système de commande. Il s'agit donc de l'aspect logique d'un système auquel l'accès se fait par des **variables d'entrée et des variables de sortie booléennes**, le comportement indiquant la relation de causalité qui existe entre les entrées et les sorties.

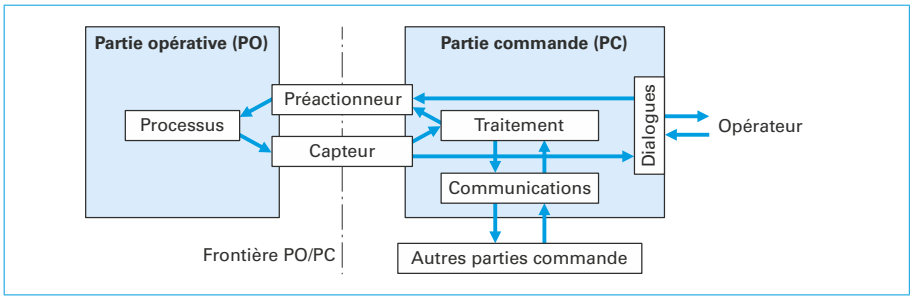



**Figure 2 – Structure d'un système automatisé**

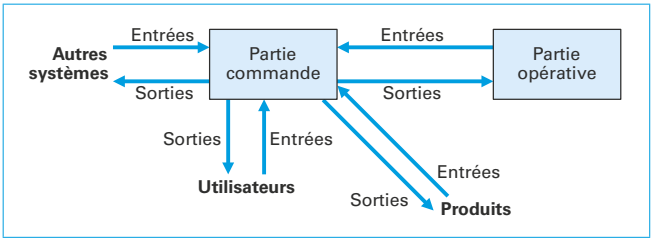



**Figure 3 – Entrées et sorties de la partie commande**

Un grafcet est donc une description graphique du comportement de la partie séquentielle du système de commande qui établit les relations entre ses entrées et ses sorties logiques.

Définir la frontière de modélisation et faire le bilan des entrées et des sorties, qui seront gérées par le grafcet, sont un préalable à toute conception de ce même grafcet.

Dans le cas de la commande d'ouverture de porte (figure **4**), toutes les entrées et les sorties ne sont pas logiques. En effet, l'entrée « couple » (nécessaire à la fermeture) est non booléenne. Un opérateur « test » permet de réaliser la comparaison de la valeur de la variable d'entrée avec une valeur donnée (limite). Le résultat de cette comparaison est une variable logique (comparaison vraie ou fausse) : [couple < limite].

La partie séquentielle ne comporte que des variables d'entrées et de sorties booléennes, toutefois le langage de spécification GRAFCET permet par extension (exemple : évaluation d'un prédicat ou affectation d'une valeur numérique à une variable) de décrire le comportement de variables non booléennes.

## 4.3 Présentation sommaire du langage graphique GRAFCET

Le langage GRAFCET est défini par un ensemble constitué :

— d'éléments graphiques de base : les étapes, les transitions et les liaisons orientées, formant la partie graphique du grafcet et sa **structure** qui décrit l'évolution possible entre les situations du système ;

— d'une **interprétation**, traduisant les comportements de la partie commande vis-à-vis de ses entrées/sorties et caractérisée par les actions associées aux étapes et les réceptivités associées.

Les éléments graphiques (figure **5**) offrent une représentation synthétique du comportement reposant sur une description indirecte de la situation du système.

### 4.3.1 Étape

Une étape caractérise un comportement invariant d'une partie ou de la totalité de la partie commande à un instant donné. Elle est utilisée pour définir la situation de la partie séquentielle d'un système.

Une étape est soit **active**, soit **inactive**. L'ensemble des étapes actives d'un grafcet à un instant donné représente la **situation** de ce grafcet à l'instant considéré.

Une ou plusieurs actions peuvent être associées à une étape afin de traduire « ce qui doit être fait » chaque fois que cette étape est active. L'**action** indique, dans un rectangle, comment agir sur la variable de sortie (RECUL dans la figure **6**).

### 4.3.2 Transition

Une transition indique la possibilité d'évolution d'activité entre deux ou plusieurs étapes (figure **7**). Cette évolution s'accomplit par le franchissement de la transition.

Une transition est dite **validée** lorsque toutes les étapes immédiatement précédentes reliées à cette transition sont actives.

Associée à chaque transition, la **réceptivité** est une condition logique qui est soit vraie, soit fausse, et qui est composée de variables d'entrées et/ou de variables internes (figure **8**).

Une réceptivité qui est toujours vraie se note : 1.

### 4.3.3 Liaisons

Une liaison orientée relie soit une ou plusieurs étapes à une transition, soit une transition à une ou plusieurs étapes (figure **5**).

### 4.3.4 Règle de syntaxe

L'alternance étape-transition et transition-étape doit toujours être respectée quelle que soit la séquence (nota **3**) parcourue.

**Nota 3** : une séquence est un bloc d'étapes successives où chaque étape est suivie d'une seule transition et chaque transition n'est validée que par une seule étape.

En conséquences :

— deux étapes ou deux transitions ne doivent jamais être reliées

Référence Internet
S7241

# GRAFCET

# Structuration des descriptions. Applications

*par* **Jean-Jacques DUMÉRY**
*Docteur de l'École Centrale Paris*



*L'étude du langage de spécification GRAFCET pour diagrammes fonctionnels en séquence fait l'objet de deux dossiers. Le lecteur doit donc consulter également le fascicule [S 7 240] qui complète le présent dossier.*

# 1. Structuration des descriptions par niveaux de description

L'un des avantages principaux de la révision de la norme CEI 60848 [1] est de disposer de moyens pour structurer la spécification. Cette structuration, assistée ou non par des méthodologies adaptées, peut se limiter à une partition du grafcet ou utiliser des macro-étapes, intégrer des notions de hiérarchie par forçage ou encore utiliser l'encapsulation.

Certaines de ces notions (macro-étape et forçage), apparaissant dans diverses publications, étaient utilisées par la communauté mais elles n'étaient pas normalisées.

Les représentations de type macro-étapes, partition de grafcet et étapes encapsulantes ont pour avantages :

— d'obtenir des représentations homogènes du comportement facilement concevables ou analysables ;
— de pouvoir décomposer la description globale en un ensemble de documents fonctionnels de format réduit (A4 par exemple) ;
— d'expliciter par une approche progressive, la compréhension des différents fonctionnements ;
— de faciliter la mise à jour des documents.

La notion de forçage est plus souvent réservée à l'expression de contraintes liées aux modes de marche et d'arrêt (forçage et figeage de situations par exemple).

## 1.1 Utilisation de macro-étapes

Pour améliorer leur compréhension, les descriptions élaborées sous forme de grafcets, qui contiennent de nombreuses séquences et étapes, peuvent intégrer avantageusement des macro-étapes qui permettent une **abréviation d'écriture** de grafcets. Il s'agit d'exprimer une fonction à remplir sans se soucier de tous les détails superflus de fonctionnement, les détails étant repris dans l'expansion de la macro-étape concernée.

Une **macro-étape** est une représentation unique d'une partie détaillée de grafcet, appelé **expansion de la macro-étape**.

La macro-étape ne possède pas toutes les propriétés des autres types d'étapes, car seule l'étape de sortie de son expansion valide ses transitions aval.

L'expansion d'une macro-étape Mi est une partie de grafcet munie d'une étape d'entrée Ei et d'une étape de sortie Si (i étant le repère de la macro-étape).

L'étape d'entrée Ei devient active lorsque l'une des transitions amont de la macro-étape est franchie. La macro-étape Mi est active lorsque l'une au moins des étapes de son expansion est active.

Dans l'exemple de la figure **1**, lorsque la transition (1) est franchie, simultanément l'étape d'entrée E2 et la macro-étape M2 sont activées. La situation évolue alors dans l'expansion, M2 restant active. Lorsque l'étape de sortie S2 est activée, la transition (2) devient franchissable. Elle est alors franchie car la réceptivité associée est toujours vraie, M2 et S2 sont alors désactivées simultanément.

**Remarques** : l'expansion d'une macro-étape peut comporter une ou plusieurs étapes initiales (figure **2**). Il est recommandé que ces étapes initiales ne soient pas l'étape d'entrée ou de sortie.

L'expansion d'une macro-étape peut comporter une ou plusieurs macro-étapes. Pour éviter les incohérences, les repères numériques de ces macro-étapes seront pris dans l'ordre croissant (figure **3**).

La figure **4** donne une macroreprésentation d'une application de perçage/taraudage.

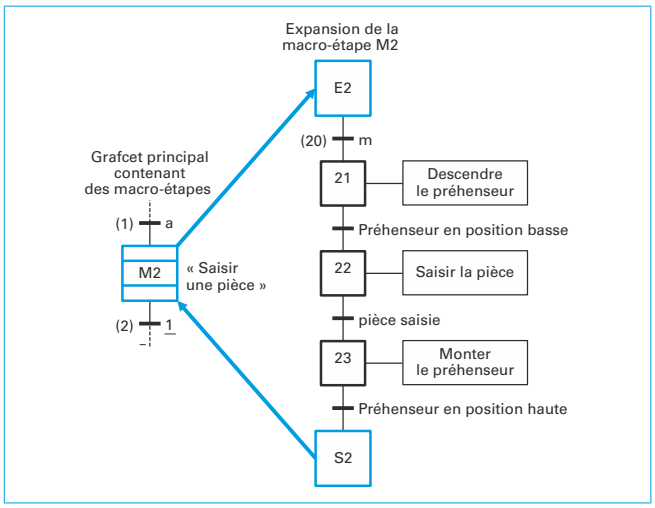



**Figure 1 – Exemple d'une macro-étape et de son expansion**

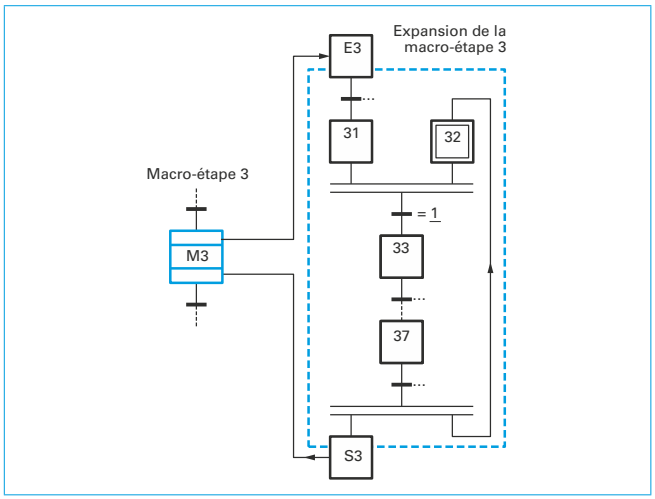



**Figure 2 – Représentation d'une macro-étape**

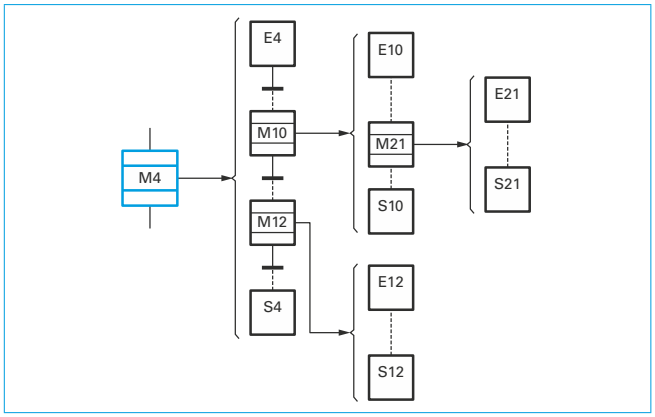



**Figure 3 – Niveaux successifs de plusieurs macro-étapes**

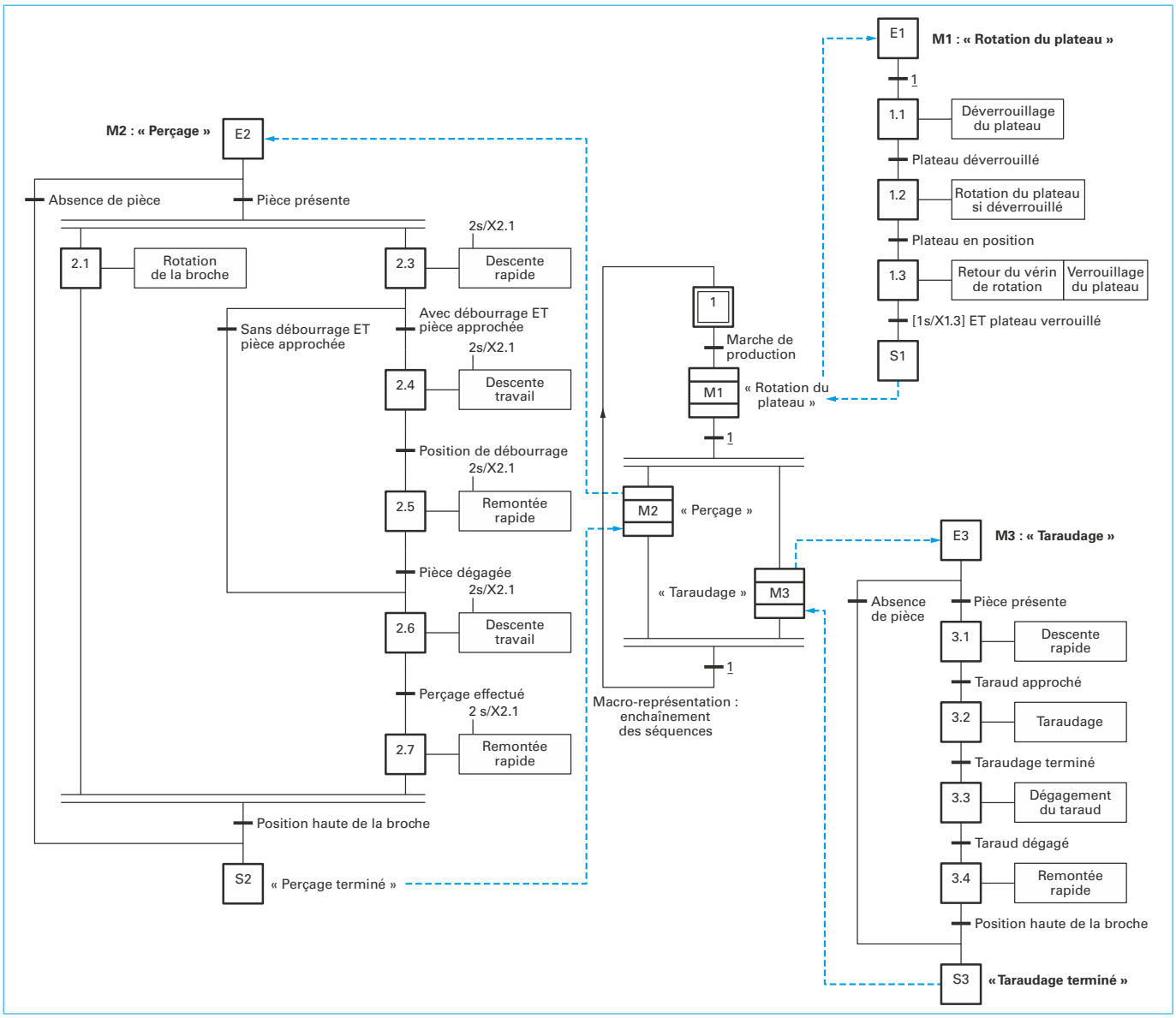



**Figure 4 – Macroreprésentation du comportement d'une unité de perçage/taraudage**

## 1.2 Partition d'un grafcet en grafcets connexes et partiels

Dans le cas d'une application complexe, il est très rare de pouvoir décrire l'ensemble du comportement de la partie séquentielle du système de commande à l'aide d'un seul grafcet. Les spécificateurs utilisent souvent une possibilité qui consiste à réaliser une partition du grafcet en grafcets assurant une fonction particulière dans l'automatisme :

- gestion de la sûreté de fonctionnement ;
- surveillance ;
- gestion des modes de marche et d'arrêt ;
- gestion d'interfaces de dialogue homme-machine ;
- coordination de tâches pour optimiser des temps de cycle ;
- gestion de la commande d'une tâche particulière dévolue à un poste ...

**Remarque** : la méthode utilisée pour réaliser cette partition fait partie du savoir-faire des concepteurs de grafcets. Le langage de spécification GRAFCET décrit uniquement les concepts de grafcet global, de grafcet connexe et de grafcet partiel (encadré **1**) nécessaires à la description du résultat de la partition.

Dans le cas où la spécification des modes de marche et d'arrêt a été réalisée à l'aide du GEMMA [3], la partition en grafcets peut s'appuyer sur celui-ci.

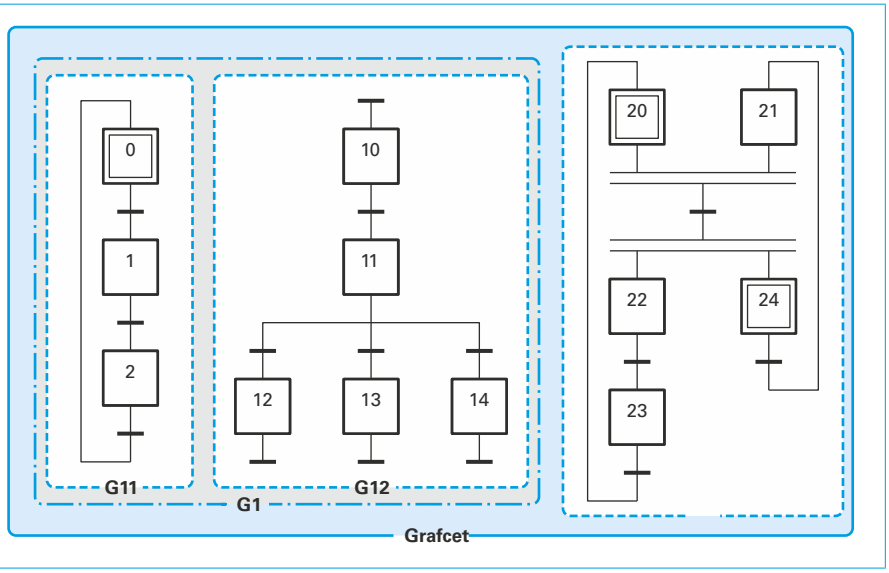


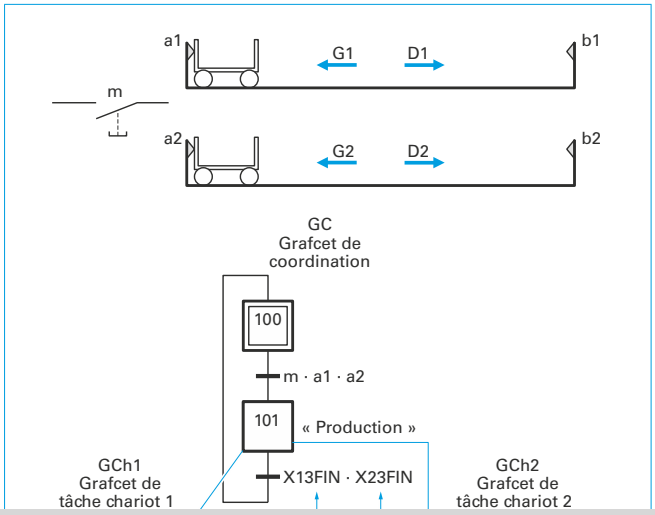

**Figure 5 – Exemple de partition d'un grafcet global**

### Encadré 1 – Définition des différents concepts de grafcets

- **Grafcet connexe** : un grafcet connexe est une structure de grafcet telle qu'il existe toujours une suite de liens (alternance d'étapes et de transitions) entre deux éléments quelconques, étape ou transition, de ce grafcet.
- **Grafcet partiel** : constitué d'un ou plusieurs grafcets connexes, un grafcet partiel résulte d'une partition, selon des critères méthodologiques, du grafcet global décrivant le comportement de la partie séquentielle d'un système.
- **Grafcet global** : le grafcet global est l'ensemble des grafcets décrivant le comportement du système isolé.

L'exemple de la figure **5** (où seule la partie structure a été représentée) illustre une partition d'un grafcet global en deux grafcets partiels G1 et G2, le grafcet partiel G1 étant lui-même partitionné en deux grafcets connexes G11 et G12.

Une partition du grafcet global étant réalisée, il est possible de coordonner les différents grafcets par l'intermédiaire de variables



Référence Internet
S8015

# Automates programmables industriels

par **Michel BERTRAND**
*Docteur-ingénieur*



*Source de méfiance et d'inquiétude lors de son apparition dans un monde industriel où l'électronique était rare, l'automate est devenu un produit banal ne donnant pas lieu à des annonces technico-commerciales spectaculaires. Certains aujourd'hui le voient comme une simple parcelle particulière du monde du PC, ou au mieux comme un composant transparent pour l'utilisateur. En fait, paradoxalement, vu sa grande diffusion, il s'est fait oublier, et est finalement méconnu.*

*Il possède pourtant des qualités propres qui le rangent dans une catégorie spécifique d'organes de commande, à côté notamment des PC, des microcontrôleurs, des systèmes numériques de contrôle-commande (SNCC), des contrôleurs d'automatismes programmables (PAC) : robustesse dans des conditions industrielles, positionnement temps réel, variété de configurations, de tailles et de présentations, facilité de programmation par des non-spécialistes, intégration aisée à des ensembles-métier dans nombre d'applications. Ses limites en mémoire et capacité de traitement peuvent être dépassées par une judicieuse association avec d'autres systèmes numériques,* via *des réseaux locaux industriels.*

*Le développement de nombreuses fonctions métier, sous forme de modules logiciels et/ou matériels, celui concomitant de nouveaux moyens de communication et d'outils logiciels appropriés facilitant sa mise en réseau, entraînant ainsi une diminution du câblage et la possibilité d'une programmation à plus haut niveau, en font un bon organe de la production automatisée flexible, prenant en compte les exigences actuelles en matière de sûreté de fonctionnement.*

Décembre 2010

# 1. Automatisation et automates

## 1.1 Caractérisation de l'automate

L'automate programmable, souvent appelé automate programmable industriel (API, en anglais PLC pour *Programmable logic controller*) pour rappeler son domaine privilégié d'utilisation, l'industrie, est apparu voici 40 ans et s'est rapidement répandu dans la production, la logistique, le conditionnement, la gestion technique de bâtiments, etc. Son développement a accompagné celui de l'automatisation de la production, la faisant passer du stade de la ***machine automatisée*** à celui du ***système automatisé de production*** (SAP), et il en est devenu, avec le robot, un composant majeur, le « fantassin de l'automatisation industrielle », suivant l'expression de C. Laurgeau, l'un des auteurs du premier ouvrage français sur la question. Il s'est vendu à des millions d'exemplaires, et seul le PC, né plus tard, avec un champ d'application plus large, a fait mieux.

La question de le ramener à cet outil de communication universel qu'est le PC s'est alors rapidement posée. Lorsque l'on cerne bien la définition rappelée ci-après et le domaine concerné, l'industrie et les services gérant du matériel, que l'on prend en compte l'évolution des moyens de transmission de l'information, ces systèmes à processeur ne s'emploient pas de manière optimale au même niveau. Ils sont complémentaires dans une optique de traitement numérique de plus en plus poussé, intégrant toutes les étapes du processus de production [R 8 022].

Il faut donc définir l'automate, et ce n'est pas si simple. « Automate » a d'ailleurs différents sens, dans le langage courant et dans le langage scientifique [1]. La norme française NF 61131, Automates programmables, dans sa partie 1, le fait ainsi : « *Système électronique fonctionnant de manière numérique, destiné à être utilisé dans un environnement industriel, qui utilise une mémoire programmable pour le stockage interne des instructions orientées utilisateur aux fins de mise en œuvre de fonctions spécifiques, telles que des fonctions de logique, de mise en séquence, de temporisation, de comptage et de calcul arithmétique, pour commander au moyen d'entrées et de sorties tout-ou-rien ou analogiques divers types de machines ou de processus. L'automate programmable et ses périphériques associés sont conçus pour pouvoir facilement s'intégrer à un système d'automatisme industriel et être facilement utilisés dans toutes leurs fonctions prévues* ».

La définition donnée par la norme NFC 63-850 « *Appareil électronique qui comporte une mémoire programmable par un utilisateur automaticien (et non informaticien) à l'aide d'un langage adapté, pour le stockage interne des instructions composant les fonctions d'automatisme comme* :

– *logique séquentielle et combinatoire* ;
– *temporisation, comptage, décomptage, comparaison* ;
– *calcul arithmétique* ;
– *réglage, asservissement, régulation, etc., pour commander, mesurer et contrôler au moyen d'entrées et de sorties (logiques, numériques ou analogiques) différentes sortes de machines ou de processus, en environnement industriel* » était voisine mais légèrement différente.

D'après le site de J.L. Hû, « *Un automate programmable industriel (API) est une machine électronique programmable par un personnel non informaticien et destiné à piloter en ambiance industrielle et en temps réel des procédés ou parties opératives* ».

À partir de là, et en respectant les caractéristiques essentielles qui ressortent de ces définitions souvent un peu laborieuses, nous considérerons plus simplement comme automate programmable un système :

– construit autour d'un processeur numérique, spécifique ou non ;
– pouvant être relié à de nombreux signaux physiques ;
– fonctionnant dans des conditions industrielles grâce à une protection adaptée ;
– doté d'un logiciel de programmation permettant un traitement rapide des fonctions logiques (tout-ou-rien, TOR dans le monde industriel) ;
– doté de possibilités d'échanges avec d'autres processeurs.

Ceci constitue un « noyau » minimal susceptible de nombreuses variantes et extensions propres à un automate donné.

## 1.2 Rôles de l'API dans un système de production

L'API est donc d'abord un composant de commande, envoyant des signaux vers les actionneurs, en fonction des informations reçues de l'instrumentation du système matériel, la Partie Opérative, et des ordres reçus (consignes), selon une algorithmique appropriée définie par le programme.

La figure **1** explicite ce fonctionnement ; la Partie Commande – pour nous l' API – agit dans le cadre d'un système bouclé sur les éléments matériels de l'installation, en vue d'une production, au sens large du terme : fabrication bien sûr, mais aussi emballage, autorisation d'accès, etc. De plus en plus, elle gère aussi l'énergie, incluant donc celle-ci, s'il s'agit d'élaborer matières premières ou objets, dans une Partie Opérative élargie.

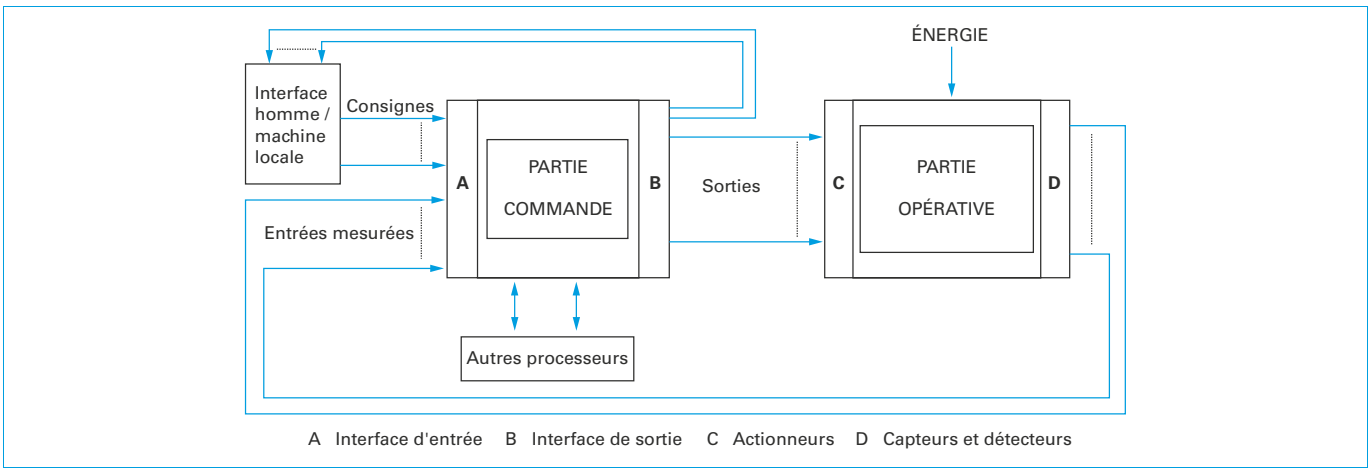



**Figure 1 – Automate en commande**

**Nota :** dans le schéma de la figure **1**, outre les consignes opérateur, il peut en exister d'autres provenant d'un niveau supérieur *via* « Autres processeurs » ; la notion d'entrées et de sortie se détermine par rapport à la Partie Commande.

Les flux d'informations qui apparaissent sur la figure **1** véhiculent les consignes, les signaux de commande et ceux émis par la Partie Opérative (mesures), mais aussi les informations en provenance ou à destination d'autres systèmes numériques. Au travers de la multiplicité des échanges possibles entre l'automate et son environnement apparaissent d'autres rôles éventuels de l'API : supervision locale, gestion de réseau, conditionnement de mesures, etc. mais aussi la possibilité de modifier de l'extérieur la commande, au niveau consignes comme à celui du paramétrage, voire de l'algorithme lui-même.

L'API s'intègre dans un ensemble de processeurs spécialisés et interconnectés, suivant une architecture décentralisée qui en facilite la conception et l'installation en permettant de fractionner les études, la mise en place, les tests, et qui en améliore aussi la maintenance (modification aisée des programmes, de parties du système automatisé). Elle entraîne toutefois, du fait des multiples sous-ensembles fonctionnels, un fort accroissement des besoins de communication et de gestion pour assurer une bonne coordination de cet ensemble.

# 2. Propriétés fondamentales

La meilleure façon de concrétiser la définition de l'API est d'examiner les éléments matériels et logiciels qu'il doit comporter, et la manière dont ils s'agrègent pour constituer un ensemble opérationnel.

## 2.1 Constituants

Quel que soit le type d'automate – compact (monobloc), modulaire, c'est-à-dire formé de modules fonctionnels en boîtiers enfichables sur un châssis spécifique, ou encore en cartes à monter dans des supports standard (rack 19 pouces) – il comporte les éléments détaillés ci-après.

### 2.1.1 Unité centrale

Fondamentalement un automate est un système numérique. Il comprend donc une unité centrale bâtie autour d'un processeur. Ce dernier peut être spécifique ; c'est le plus souvent aujourd'hui un processeur de PC. La course aux performances globales présente peu d'intérêt ici, et il peut s'agir d'un composant considéré comme obsolète pour des applications imagerie, voire bureautique. L'automate PLC102 de Wöhrle comporte ainsi deux processeurs 80386 d'Intel, un pour l'unité centrale, l'autre pour l'interface homme/machine (IHM) contenue dans le même boîtier. Ce n'est pas là le dernier cri en matière de processeurs.

Le processeur proprement dit est donc un circuit de conception ancienne, éprouvé et bon marché. L'amélioration des performances en temps d'exécution viendra plutôt des circuits d'interfaçage, les coupleurs, gérant les échanges selon les différents modes étudiés au § 4, de l'incorporation de microcontrôleurs et processeurs spécialisés, voire du remplacement partiel par des circuits spécifiques (*Application specific integrated circuit* : ASIC) si le facteur temps est primordial.

### 2.1.2 Mémoires

Elles sont étroitement associées à l'unité centrale.

Le système d'exploitation est contenu dans une mémoire morte (*Read only memory* : ROM), que le processeur ne peut que lire ; une mémoire vive (*Random access memory* : RAM) contient programme et données de l'application ; elle est sauvegardée par batterie pour protéger ces données lors de coupures de l'alimentation électrique. Le programme, une fois au point, est stocké dans une mémoire reprogrammable avec un matériel adapté (EPROM), automatiquement transférée dans la RAM après une coupure secteur. La taille de cette mémoire, exprimée en kilooctets (1 024 groupes de 8 bits), est faible par rapport à celles rencontrées en bureautique, même si elle s'est nettement accrue suite au développement de la communication numérique et de ses besoins. L'utilisation de mémoires flash de 1 024 kilooctets ou davantage, qui conservent leurs données en cas de perte de l'alimentation électrique, facilite la sauvegarde.

La taille des mémoires programme et données n'est qu'une indication de la capacité de l'API dans ce domaine. Le système d'exploitation impose aussi des limites par type de langage pour le programme, de variables pour les données : la mémoire d'un micro-automate pourra ainsi recevoir au plus 256 mots variables de 16 bits, 64 mots constants, 128 bits internes.

### 2.1.3 Entrées et sorties (E/S)

Un automate ne se conçoit qu'en liaison avec des grandeurs physiques caractérisant le comportement de la Partie Opérative, c'est-à-dire le substrat du système de production.

Au minimum on aura des E/S tout-ou-rien (TOR), aussi appelées logiques, binaires, numériques, digitales.

#### ■ En modules d'entrées ou de sorties

Ils regroupent des voies de liaison filaire de mêmes caractéristiques physiques, conformes à la norme NF 61131, Partie 2, dans un boîtier, souvent selon un nombre puissance de 2 pour faciliter l'adressage, le plus courant étant 16. Il faut noter la nécessité de les protéger contre les nuisances de l'environnement industriel : parasites, surtensions, court-circuits, etc., et la variabilité des standards de liaison électrique. La figure **2** montre un mode usuel de protection des entrées, avec optocouplage et circuits de conditionnement électrique assurant filtrage et limitation du courant pour assurer une tension $V_e$ adéquate entre la borne d'entrée et la masse.

Pour les sorties, protégées contre les surcharges et l'inversion de polarité, certaines sont à relais statiques, d'autres à relais électromagnétiques :

– modules à sorties statiques : relais statiques intégrant des composants spécialisés (transistors bipolaires, IGBT [*Insulated gate bipolar transistor*], thyristors, triacs. Ils ne subissent pas d'usure mécanique et leurs caractéristiques de commutation se maintiennent dans le temps) ;

– modules à relais électromagnétiques, dits simplement sorties relais. Ils assurent le découplage entre signaux de commande et de travail grâce à l'existence de deux circuits électriques : circuit d'excitation (bobine) et circuit de puissance (contacts). D'une durée de vie plus limitée que les relais statiques (moins de 100 000 cycles pour des contacts soumis à 10 A sous 125 V alternatif) et plus lents, les relais électromagnétiques ont aussi des avantages : faible résistance de contact, faible capacité de sortie, faible coût. Il faut leur adjoindre un circuit anti-rebond pour obtenir une commutation unique lors du changement d'état de la sortie correspondante.

#### ■ Intégrées

C'est le cas des automates compacts type PS4 de Moeller ou TSX07 de Schneider. Les caractéristiques des E/S sont alors choisies parmi les standards les plus répandus (entrées 24 V isolées, sorties transistorisées à alimentation continue, ou entrées 110 V alternatif

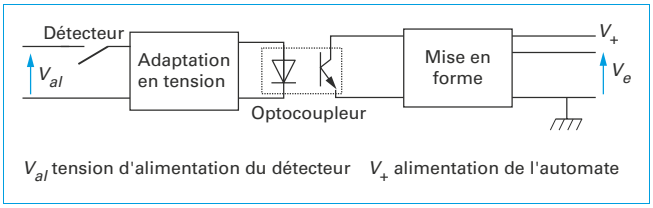



**Figure 2 – Circuit d'entrée TOR d'API avec protection**

isolées, sorties relais, par exemple). Le nombre de voies des modules intégrés reflète le rapport moyen entre entrées et sorties dans les problèmes industriels, qui se traduit par un déséquilibre au profit des entrées. Le micro-automate programmable NAIS FP-e de Matsushita possède ainsi huit entrées et six sorties TOR (cinq transistorisées et une à relais). La possibilité de configurer des voies d'accès en entrée ou en sortie est rarement utilisée, pour des raisons de sécurité. La distinction modulaire/compact est en partie gommée par l'existence, sur plusieurs API monobloc, de modules d'extension renforçant si besoin la faible configuration initiale.

**Déportées**

Cette dernière solution est en progrès car elle minimise le câblage. Le bloc d'E/S doit comporter un processeur et communiquer avec le processeur central, grâce à l'un des modules décrit au § 2.1.4. Les modules d'E/S sans fil se rattachent à cette catégorie.

Les modules d'E/S possèdent un voyant (diode électroluminescente) par voie ; cette visualisation est fort utile pour la mise au point et la maintenance, même s'il existe aujourd'hui d'autres outils de diagnostic, et participe à la sûreté de fonctionnement de l'API. Il ne faut pas oublier par contre l'existence d'un commun électrique entre les différentes voies, au moins par groupe de quatre par exemple.

Les API ayant besoin de mesures, pour le contrôle et/ou la régulation, disposent d'E/S analogiques. Elles sont groupées par quatre, quelquefois huit, avec, là encore, une grande variété de gammes de liaison électriques :

– en tension 0-5 V, 1-5 V, 0-10 V, ± 5 V, ± 10 V ;
– en courant 0-20 mA, 4-20 mA, ± 20 mA.

Les plus répandues sont 0-10 V et 4-20 mA ; ce dernier standard assure une transmission plus fiable que les liaisons en tension, et fournit une détection immédiate d'une rupture de la liaison ou du capteur.

Il existe encore d'autres types d'E/S, notamment des entrées adaptées à des capteurs particuliers (sondes à résistances de platine PT100, thermocouples, jauges de contrainte), à associer à des fonctions métier. On les trouve même parfois sur des API monoblocs : le NAIS FP-e déjà cité a ainsi deux entrées thermocouple.

**Exemple :** Modules d'E/S TOR pour Modicon M340 de Schneider.
En entrées TOR, on dispose des éléments suivants à bornier débrochable :
– 16 E 24 V continu en logique positive,
– 16 E 48 V continu en logique positive,
– 16 E 24 V alternatif en logique positive ou négative,
– 16 E 48 V alternatif en logique positive ou négative,
– 16 E 110 V alternatif en logique positive ou négative.

– les sorties à relais permettent le passage de courants supérieurs à ceux supportés par les sorties statiques, mais nécessitent une protection extérieure contre les court-circuits ;
– les entrées peuvent être en logique positive (1 correspond au contact fermé, le commun est à 0 V) ou négative ;
– différents modes de raccordement mécanique sont offerts ; ils sont accompagnés de dispositifs de placage (vis étriers, vis à cage, ressorts, pour les borniers débrochables, 2 vis pour les connecteurs) garantissant un bon contact entre module et châssis. C'est un point important pour la fiabilité de l'API. La possibilité d'embrochage et débrochage à chaud (*plug and play*) se développe, facilitant la maintenance.

### 2.1.4 Modules de communication

Un API a aujourd'hui plusieurs possibilités de communication numérique :

– par le port série du processeur (liaison RS232 ou RS485) ;
– par un port USB (*Universal serial bus*) ;
– par un coupleur Ethernet ;
– par d'autres canaux : CAN (*Controller area network*), AS-I (*Actuator sensor interface*), etc. dont le coupleur est quelquefois intégré.

Remarquons qu'il s'agit là de possibilités physiques ; le problème de la communication effective sera abordé au § 4.

### 2.1.5 Alimentation électrique

À partir du secteur, plus rarement d'une source de 24 V continu, l'alimentation principale fournit l'énergie nécessaire à l'UC et aux interfaces (sous des tensions de 5, 12, 15, 24 V continu). Elle ne peut que rarement alimenter des sorties, et pas toujours les entrées. Elle doit faire face aux microcoupures du réseau. Un onduleur évite les risques de coupure si celles-ci risquent de dépasser les tolérances admises, 10 ms au plus à 50 Hz avec au moins 1 seconde entre deux coupures.

L'alimentation fait l'objet d'une protection avec notamment un disjoncteur magnéto-thermique. Il est parfois nécessaire, pour lutter contre les perturbations électriques du secteur, d'introduire un transformateur d'isolement. C'est notamment le cas pour les raccordements à un réseau électrique à « neutre flottant ».

Il ne faut pas oublier que les châssis d'extension, les E/S déportées, voire certaines E/S directes, exigent aussi une alimentation. Son dimensionnement doit prendre en compte les ajouts possibles de modules et/ou de liaisons fil à fil directes lors de modifications ultérieures de l'installation.

Techniques de l'Ingénieur

**Cet extrait vous a plu ?**
Pour consulter les articles dans leur intégralité…

# SOUSCRIVEZ
aux Techniques de l'Ingénieur

## 3 BONNES RAISONS DE CHOISIR TECHNIQUES DE L'INGÉNIEUR

- Une **actualisation permanente** du fonds documentaire
- Un **comité d'experts** scientifiques et techniques reconnus
- Une **collection scientifique et technique incontournable** sur le marché français

Actualisées en permanence, les ressources documentaires profitent aujourd'hui à **plus de 300 000 utilisateurs** et sont la référence pour tout ingénieur, bureau d'études, direction technique et centre de documentation.

En souscrivant à l'une de nos offres pack, vous bénéficiez d'un **droit d'accès à tous les articles du pack ainsi qu'à un bouquet de services associés**. Pour encore plus d'avantages, vous pouvez également obtenir un droit d'accès pluriannuel. Pour cela, il vous suffit de demander un devis en remplissant le formulaire sur notre site. Vous serez alors contacté par l'un de nos chargés d'affaire qui vous présentera les avantages associés à cette offre.

## LES SERVICES ASSOCIÉS AUX PACKS

### Service Questions-Réponses
Besoin de complément d'information ou de validation pour mieux prendre vos décisions ? Posez des questions techniques, scientifiques, réglementaires aux plus grands spécialistes des domaines couverts par vos bases documentaires.

### Les articles Découverte
Accédez gratuitement à des articles Techniques de l'Ingénieur hors de votre abonnement dans l'ensemble de notre ressource documentaire.

### Le dictionnaire technique multilingue
60 000 termes scientifiques et techniques en français, anglais, allemand et espagnol.

### Les archives
Accédez aux versions antérieures de vos articles, comme à ceux qui traitent des technologies plus anciennes.

RESSOURCES DOCUMENTAIRES | FORMATION | CONSEIL

Techniques de l'Ingénieur

Pour disposer d'un panorama complet sur une thématique donnée

# DECOUVREZ
## nos offres de packs !

### LES + DES OFFRES PACK

- Un large choix de **+ de 60 thématiques** pour des besoins de contenu plus larges
- Des **tarifs préférentiels sur mesure** adaptés à vos besoins

### NOS UNIVERS DOCUMENTAIRES

- Plus de 425 bases documentaires et plus de 8 000 articles en 14 univers

**ARTICLES TECHNIQUES & SCIENTIFIQUES**

- Sciences fondamentales
- Environnement - Sécurité
- Énergies
- Technologies de l'information
- Mécanique
- Procédés chimie - bio - agro
- Matériaux
- Mesures - Analyses
- Électronique - Automatique
- Construction
- Transports
- Innovations
- Génie industriel
- Biomédical - Pharma

### POUR EN SAVOIR PLUS SUR NOS OFFRES PACKS...

... contactez notre service Relation clientèle
qui se chargera de vous rediriger vers un chargé d'affaire :

Tél : +33 (0)1 53 35 20 20
Email : infos.clients@teching.com
www.techniques-ingenieur.fr