Conduite orientée ordonnancement d'un simulateur dynamique hybride : application aux procédés discontinus

Florian Fabre

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par *l'Institut National Polytechnique de Toulouse*
Discipline ou spécialité : *Systèmes Industriels*

Présentée et soutenue par *Florian FABRE*
Le *20 Octobre 2009*

Titre : *Conduite orientée ordonnancement d'un simulateur dynamique hybride : application aux procédés discontinus*

JURY

M. Sandro MACCHIETTO, Professeur à Imperial College London, United Kingdom, Rapporteur
M. Frédéric SEMET, Professeur à l'École Centrale de Lille, Lille, Rapporteur
M. Jean-Marc LE LANN, Professeur à l'INPT-ENSIACET, Toulouse, Membre
Mme Pascale ZARATE, Maître de Conférences à l'INPT-ENSIACET, Toulouse, Membre
M. Gilles HETREUX, Maître de Conférences à l'INPT-ENSIACET, Toulouse, Membre
M. Pierre LOPEZ, Chargé de recherche au LAAS-CNRS, Toulouse, Membre
M. Vincent MOUSSEAU, Professeur à l'École Centrale de Paris, Paris, Membre

---

Ecole doctorale : *École Doctorale Systèmes (EdSYS)*
Unité de recherche : *Laboratoire de Génie Chimique de Toulouse*
Directeur(s) de Thèse : *M. Jean-Marc LE LANN, Mme Pascale ZARATE*
Rapporteurs : *M. Sandro MACCHIETTO, M. Frédéric SEMET*

Année 2009

THESE

Préparée au
Laboratoire de Génie Chimique du CNRS

En vue de l'obtention du titre de
Docteur de l'Université de Toulouse
délivré par l'Institut National Polytechnique de Toulouse

Spécialité
Systèmes Industriels

Par
Florian FABRE
Ingénieur ENSIACET Toulouse

CONDUITE ORIENTEE ORDONNANCEMENT D'UN SIMULATEUR DYNAMIQUE HYBRIDE :

APPLICATION AUX PROCEDES DISCONTINUS

Soutenue le 20 octobre 2009, devant le jury composé de :

Rapporteurs :	Sandro MACCHIETTO Frédéric SEMET
Directeurs de thèses :	Jean-Marc LE LANN Pascale ZARATÉ
Co-directeur de thèse :	Gilles HÉTREUX
Examinateur :	Pierre LOPEZ Vincent MOUSSEAU

À la mémoire de mon grand-père,

À mes parents et à mes frères,

Remerciements

Une Fin pour un Début ! Comme toujours dans ces moments là, on ne garde que les meilleurs moments des aventures passées et quelle aventure ! Je laisserai donc les moments de doute et de découragement à ceux qui ne croyaient pas que je puisse arriver à la fin ; leur doute a été mon catalyseur. Mes remerciements vont plutôt à toutes les personnes qui m'ont permis d'avancer durant cette période de ma vie (un peu trop longue du goût de mes nièces) et avec qui j'ai appris à transformer le découragement en enthousiasme, et le doute en sérénité.

Mes remerciements vont tout d'abord aux rapporteurs de cette thèse, Monsieur Sandro MACCHIETTO, professeur à l'Imperial College of London, et Monsieur Frédéric SEMET, professeur à l'Ecole Centrale de Lille.

J'adresse ensuite mes sincères remerciements à Messieurs Pierre LOPEZ et Vincent MOUSSEAU, d'apporter leur caution scientifique à cette thèse.

J'exprime tout mon respect à mes deux directeurs de thèse, Jean-Marc LE LANN et Pascale ZARATE ainsi qu'à Gilles HETREUX, pour la confiance que vous m'avez accordée, malgré la difficulté et les risques de l'aventure dans laquelle je m'engageais, et des choix personnels que je faisais.

Jean-Marc, comme tu le sais déjà tes venues en fin de journée aux algécos constituaient pour nous ces petits moments qui nous aidaient à avancer et il y a une phrase que je garderai, tu me l'as dite l'autre soir en passant au bureau pendant ma période de rédaction dans ces chers algécos que je retrouvais épisodiquement depuis bientôt trois ans : « C'est pas évident, hein ? », comme toujours, bien résumé !

Merci à Pascale pour m'avoir aidé à garder le cap durant ces quatre longues années, tu m'as aidé à être efficace, à me fixer des jalons et à garder en tête mes priorités, je t'en remercie !

Ensuite, je tiens plus particulièrement à remercier Gilles, mon encadrant durant ces quatre années de thèse. Je te remercie de m'avoir considéré comme ton collègue, je te suis reconnaissant pour toutes ces discussions et ces débats qui nous ont fait avancer dans ces recherches. Cette thèse est le fruit de notre collaboration. Ce fut vraiment très enrichissant de travailler avec toi.

Je remercie la société ProSim pour le suivi et le soutien de ces travaux.

Plus largement, j'exprime toute ma reconnaissance au département PSI pour leur accueil chaleureux et leur gentillesse. Un merci tout particulier à Catherine, qui a toujours suivi mes travaux.
Je tiens plus particulièrement à remercier les équipes AFP et GI, mes anciens professeurs et collègues d'un temps. Merci à Xuan, Xavier, Pascal, Denis, Vincent, Raphaële, Stéphane, Alain, Philippe, Sophie, Jean-Pierre, Christian, Chantal, Karim et Michel. Merci à tous pour votre soutien, pour cette ambiance si particulière, pour ces bons moments passés ensemble.

Cette thèse a également été une aventure humaine faite de rencontres et de partages simples, enrichissants et inoubliables. Merci donc aux thésards de l'algéco à Guillermo, Alejandro, Françoise et Florian.

Je tiens aussi à remercier Didier BERTANA et Jean-Marc JOUANINE pour m'avoir aidé eux-aussi durant cette période, leur compréhension et leur soutien ont été déterminants pour mener à bien cette thèse durant cette dernière année. Encore Merci !

Enfin, cette thèse n'aurait pas pu aboutir sans le soutien inconditionnel de mes amis. Ceux qui ont partagés avec moi ces moments inoubliables dans « nos » algécos et ceux qui partagent ma vie aujourd'hui à Pau.

- Nelly pour nos discussions interminables sur notre avenir, et nos soirées aux algécos, et durant ces deux dernières années pour votre accueil avec Vincent, ce sera toujours un bonheur de vous retrouver, tous les… trois !
- Guillaume, pour ton accueil dans ton « palace ! » durant ces dernières semaines de rédaction, je n'ai jamais du te prévenir plus de deux jours à l'avance de mes arrivées !! Merci !
- Marie et Emmanuel, pour nos discussions interminables sur LA thèse et votre soutien sans faille.
- Jean-Marie pour tes relectures éclairées.

Et tous les autres, les amis de longues dates, qui ont toujours été là dans les bons et les mauvais moments Arnaud, Rémi, Jordane, Adrien, Jocelyn, FabreS , Mike, Etienne, Loac, Maumau…

Pour finir, je tiens à remercier ma famille, présente à tous les instants, bons et mauvais, pour avoir vécu avec moi ces années, pour m'avoir soutenu, écouté, conseillé. Merci à mes parents d'avoir toujours été là, présents. Merci pour votre amour !
Merci à mes frères et à Isabelle, à Arnaud et Manue merci pour ces moments partagés qui me redonnent du courage, pour votre soutien.

Je ne peux finir ces remerciements sans écrire un dernier paragraphe pour Sandra, ces quatre années n'ont pas été faciles, tu as accepté ces contraintes parfois lourdes et je t'en remercie. Tu m'as donné la force et le courage de traverser cette épreuve, c'est « Une Fin pour un Début ! ».

TABLE DES MATIERES

Table des matières

Liste des illustrations

- **Liste des figures**

- **Liste des tableaux**

INTRODUCTION GENERALE

Introduction générale

Depuis plus de cinquante ans, la transformation et la valorisation des matières premières a provoqué une formidable expansion de la chimie industrielle. Durant cette période de croissance, le Génie des Procédés a acquis ses lettres de noblesse dans les domaines de la conception et de l'exploitation des procédés à grande capacité de production. Il est alors devenu synonyme de procédés continus. Le dimensionnement de ces unités de production a été à l'origine du développement des connaissances qui forment aujourd'hui les sciences du Génie des Procédés. Les complexes pétrochimiques constituent une illustration typique de cette catégorie de systèmes de production. Ce monopole historique du procédé continu en régime permanent fait que, par héritage, il est devenu l'élément prédominant de la culture de l'ingénieur de procédé.

Toutefois, tous les systèmes de production relevant du Génie des Procédés ne se limitent pas à cet unique mode de fonctionnement. En effet, si les installations des industries agroalimentaires, biotechnologiques, électroniques ou pharmaceutiques peuvent aussi être décrites et analysées en termes d'opérations unitaires, de bilans, de conditions opératoires, le procédé est rarement de nature exclusivement continue. Le mode de production discontinu est alors quasiment une règle et l'ouverture vers ces secteurs d'application génère une catégorie de problèmes originaux. L'intérêt de la communauté scientifique du Génie des Procédés pour ce type de système est aujourd'hui indéniable. Le fonctionnement des procédés discontinus est calqué sur le principe du traitement par lots. La matière chemine par quantités finies dans le système de production : les « lots ». Lors de son passage dans l'unité de fabrication chaque lot de matière subit une série d'opérations de transformation sur des appareils de traitements spécialisés. Situées sur des marchés fortement concurrentiels, ces industries utilisent préférentiellement ce mode de production en raison d'une rotation importante du nombre de produits nouveaux, ou du manque de temps pour développer une technologie spécifique afin de produire en continu. Cette technique de production exploitée pour des produits à faible demande et forte valeur ajoutée générait alors des profits tels que l'analyse des performances des systèmes de production ne s'imposait pas. Nul besoin de haute technologie ou d'optimisation de conditions opératoires tant que la capacité de production et la qualité du produit étaient atteintes avec des moyens usuels et/ou rudimentaires. Mais l'internationalisation des marchés et les besoins croissants de la société de consommation ont changé les données et engendré de nouvelles stratégies industrielles où le procédé discontinu se distingue par ses qualités de flexibilité. De même, la fonction du procédé se trouve aussi complexifiée par une volonté de regrouper les fabrications, d'alléger les coûts sociaux, de concentrer les investissements. Ces nouvelles contraintes se traduisent par un souci de mieux concevoir et surtout de mieux exploiter ces procédés discontinus.

Parmi les outils d'*Ingénierie des Procédés Assistée par Ordinateur (I.P.A.O.)* exploités au niveau industriel, la simulation dynamique joue aujourd'hui un rôle important pour cet objectif. Durant le processus de développement du procédé, les logiciels de simulation sont utilisés pour réaliser des tâches d'analyse telles que l'établissement des bilans matière et énergie, l'estimation de la taille des équipements, l'estimation de la demande en utilités en fonction du temps, l'estimation des temps de cycle ou l'analyse des coûts. Ces outils offrent donc la possibilité à l'utilisateur « d'expérimenter » différents scénarios sur ordinateur en utilisant différentes initialisations ou conditions opératoires. Ils peuvent ainsi permettre de réduire notablement le travail de laboratoire sur pilote qui s'avère souvent coûteux et consommateur de temps. En phase d'implantation et d'exploitation, disposer d'un bon modèle de simulation améliore la compréhension du procédé entier par les membres de l'équipe et

facilite la communication. En effet, des ajustements sont généralement requis quand une nouvelle unité est implantée. Les équipements sont rarement idéalement dimensionnés dès la conception. Malgré de nombreux facteurs en interaction, même pour des procédés relativement simples, les outils de simulation permettent de décrire en détail le procédé et de réaliser rapidement des analyses *« What-if »* de sensibilité, extrêmement utiles aux ingénieurs dans leur travail quotidien. L'objectif de telles études est d'évaluer l'impact de paramètres critiques sur des indicateurs de performance clefs tels que les coûts de production, les temps de cycle, les taux d'occupation des ressources ou la productivité. La simulation fournit aussi le moyen de suivre l'occupation volumétrique de toutes les cuves durant un cycle et de vérifier que les charges minimum et maximum au cours du temps sont toujours atteintes en tous les points concernés du procédé. De plus, elle permet d'analyser l'impact d'un changement en une poignée de secondes. Par exemple, l'impact de l'augmentation de la taille des batch (qui affecte la durée de toutes les tâches dépendant de la taille des lots) peut être instantanément évalué tant sur le temps de cycle de la recette que sur le nombre de batch qui la compose. Enfin, la simulation permet de définir les conditions opératoires de chaque opération unitaire et de paramétrer les nombreuses boucles de régulation nécessaires au maintien de ces conditions opératoires, l'aspect contrôle étant rarement pris en compte lors de la conception du procédé.

Cependant, les outils de simulation disponibles ne sont pas complètement adaptés à ce type d'étude. En effet, on peut constater que la majorité des applications connues de simulation dynamique de procédés est relative à l'évaluation des performances des opérations unitaires isolées ou des procédés continus en régime transitoire. Or, dans le cas d'un atelier discontinu, cette analyse nécessite :

- la simulation de tous les modes de fonctionnement possibles du système (y compris les régimes transitoires),
- la simulation du procédé complet en vue de satisfaire une liste d'ordres de fabrication sur un horizon d'étude fixé (cet horizon est généralement un paramètre fixé par la *Gestion de Production Assistée par Ordinateur (G.P.A.O.)*).

Concernant le premier point, le comportement multimodal des procédés discontinus qualifie généralement la simulation de tous ces modes de fonctionnement de *systèmes dynamiques hybrides*. Cette catégorie de systèmes nécessite des simulateurs spécifiques (nommés *simulateurs dynamiques hybrides*) capables de traiter de manière aussi rigoureuse les évolutions continues des variables d'état (montée en température, cinétique chimique, etc.) que les évolutions discrètes (marche/arrêt de pompe, ouverture/fermeture de vanne, etc.). Dans ce cadre, l'unification des travaux de recherche en modélisation et simulation de procédés menés depuis de nombreuses années au sein du Laboratoire de Génie Chimique a conduit au développement d'une plate-forme de simulation dynamique hybride nommée *PrODHyS*. Celle-ci s'appuie sur le formalisme *Réseaux de Petri Différentiel Objet (RdPDO)* qui permet de modéliser le comportement hybride (discret/continu) de chaque appareil et les phénomènes physico-chimiques (transformation de la matière) qui s'y déroulent.

Pour le second point, la séquence de passage des lots de matière dans les différents équipements de l'unité peut avoir un impact important sur les performances du système. Pour étudier cet aspect, il faut donc que le simulateur puisse dérouler un scénario correspondant à une recette et une affectation de ressources définies. Ces scénarios correspondent en fait à la réalisation d'un plan de production préalablement établi dans lesquels sont définis, entre autres, le nombre, le volume et la date de lancement de chaque lot ainsi que l'appareil utilisé pour réaliser l'opération. Une utilisation rationnelle de l'unité consiste alors à assurer une synchronisation temporelle et volumique des lots en minimisant autant que possible les temps d'inactivité des appareils. La difficulté réside dans l'évolution myope de la simulation qui ne permet pas de prendre en compte de manière explicite ces contraintes de séquence et peut conduire, dans certains cas, à l'avortement de la simulation pour cause de non respect de certaines contraintes (capacité ou délais par exemple). La coordination des flux ne peut être assurée que par une gestion globale des lots sur tout l'horizon d'étude. Pour cette raison, le simulateur dynamique doit être accompagné d'un module d'ordonnancement chargé d'établir un ou plusieurs scénarios de production faisables satisfaisant au mieux un certain critère afin d'évaluer correctement la performance d'une unité discontinue.

La conception et la mise en place d'un outil satisfaisant les deux points précédents constituent le cœur des travaux de recherche présentés dans ce manuscrit.

Classé dans la catégorie des problèmes *NP-difficiles*, différentes approches d'optimisation sont proposées dans la littérature pour résoudre le problème d'ordonnancement. Notons cependant que, compte tenu de notre objectif, toute méthode permettant d'obtenir une solution faisable satisfaisante en un temps raisonnable est candidate. La qualité du plan initial et l'effort de calcul pour l'obtenir sont donc des paramètres à définir dans cet outil de simulation de procédé discontinu.

Ainsi, dans le cadre du développement de la plate-forme *PrODHyS*, l'objectif est de proposer une extension au simulateur dynamique hybride permettant de construire automatiquement le scénario de simulation d'un procédé discontinu sur la base d'une recette et d'un carnet de commande ou d'une liste d'ordres de fabrication (*OF*). Pour cela, trois points essentiels sont visés dans ces travaux :

- tout d'abord, concevoir et développer des composants réutilisables (*classes de recette*) qui modélisent le déroulement des différentes opérations unitaires et permettent de décrire de manière systématique les recettes à réaliser par assemblage de ces composants,
- ensuite, définir un formalisme de représentation ainsi qu'un modèle mathématique générique d'ordonnancement qui permet de modéliser les principales caractéristiques d'un procédé discontinu et fournit l'ensemble des données d'entrée nécessaires au modèle de simulation,
- enfin, établir l'interface existant entre le modèle d'optimisation et le modèle de simulation, notamment les règles à mettre en place s'il existe une dérive entre les scénarios prévus et simulés.

Afin de présenter les résultats de ce travail, ce manuscrit s'articule autour de six chapitres :

- Le **chapitre I** fixe le cadre de l'étude. Pour cela, les enjeux du pilotage des systèmes de production sont mis en évidence. Les outils de conduite et d'analyse de la performance sont ensuite décrits. Le type de système de production considéré dans ces travaux est présenté : les procédés discontinus. L'intérêt de la simulation dynamique hybride pour l'analyse de ces systèmes est alors montré. Les principes de l'outil développé dans ces travaux sont enfin exposés.

- Dans le **chapitre II**, la plate-forme de simulation dynamique hybride *PrODHyS* est présentée. Les concepts fondamentaux et les potentialités de cet outil sont décrits. Sur cette base, les composants à mettre en place pour étendre ces fonctionnalités sont identifiés. Enfin, l'utilisation de cette plate-forme est illustrée à travers la mise en place d'une opération de réaction utilisée comme exemple didactique.

- Le **chapitre III** propose un état de l'art des méthodes et outils d'ordonnancement généraux et permet d'identifier la solution retenue dans ces travaux : la Programmation Linéaire en Variables Mixtes.

- Le **chapitre IV** est consacré aux formulations de Programmation Linéaire en Variables Mixtes généralement rencontrées dans la communauté Procédés pour la résolution des problèmes d'ordonnancement. Les formalismes graphiques de représentation de ces problèmes sont d'abord présentés. Ensuite, les principales caractéristiques, les points forts et les limites des approches de modélisation et d'optimisation trouvées dans la littérature sont examinés. Le modèle en temps continu mis en place dans notre outil est présenté en détail.

- Le **chapitre V** décrit alors le nouveau module *ProSched* et en particulier le package *Scheduling* qui intègre les développements liés à la conduite de la simulation. Ces derniers visent à formaliser la recette de *ProDHyS* afin de permettre l'identification d'un « Centre de décision » au niveau du simulateur.

- Enfin, le **chapitre VI** propose les principes d'une approche basée sur un couplage Optimisation/Simulation pour le pilotage des procédés discontinus. Nous présentons ensuite deux exemples complets de procédés traités au moyen de l'outil développé. Par ailleurs, plusieurs outils complémentaires qui permettent une exploitation plus aisée du simulateur (inclus dans la bibliothèque *ProSched)* sont décrits dans ce chapitre.

CHAPITRE 1

CHAPITRE 1

CADRE DE L'ÉTUDE

Résumé

Ce chapitre introductif se propose de décrire le contexte dans lequel s'inscrivent ces travaux de recherche. Après une brève présentation du domaine d'application auquel s'est intéressé cette étude : le Génie des Procédés, nous présentons l'évolution des modes de production, pour définir les systèmes continus, discontinus et semi-continus. Ensuite, nous analysons l'importance de la fonction ordonnancement tant au niveau de l'outil de production qu'au sein de l'entreprise. Ce chapitre vise à montrer l'importance de la simulation dynamique hybride pour la conception, la supervision et l'analyse des performances de ces outils de production. Dans ce contexte, les limites des simulateurs actuels sont finalement exposées et la nécessité d'un couplage avec un module d'ordonnancement est abordée.

1.1. PILOTAGE DES SYSTEMES DE PRODUCTION

1.1.1. La démarche de performance

Les entreprises évoluent aujourd'hui au sein de marchés soumis à une forte concurrence et sur lesquels les exigences et les attentes des clients deviennent de plus en plus fortes en termes de qualité, de coût et de délais de mise à disposition. De plus, le développement rapide des NTIC ne fait qu'accroître les contraintes de délais auxquelles sont soumises les entreprises en permettant une relation directe entre entreprises *(BtoB)* et entre les entreprises et leurs clients *(BtoC)*.

Dans [P. Lopez et al. 2001], les deux dimensions permettant de construire la performance d'une entreprise sont décrites de la manière suivante :

- une **dimension technologique**, qui vise à développer les performances intrinsèques des procédés de fabrication et des produits mis sur le marché afin de satisfaire aux exigences de qualité et de réduction des coûts. La recherche (nouveau matériau ou nouvelle molécule, nouvelle technique ou méthode de fabrication) et l'innovation technologique jouent ici un rôle important et peuvent constituer un élément différenciateur déterminant pour la pénétration et le développement d'un marché. Il faut noter à ce niveau que l'évolution rapide des produits et les exigences de personnalisation de ces produits qu'attendent les marchés conduisent les entreprises à abandonner souvent les productions de masse pour s'orienter vers des productions en petites ou moyennes quantités (ou séries), voire même à la commande. Ceci leur impose donc d'avoir des systèmes de production flexibles et évolutifs, capables de s'adapter rapidement et efficacement aux exigences et aux besoins des marchés;
- une **dimension organisationnelle**, qui vise à développer la performance en terme de durée des cycles de fabrication, respect des dates de livraisons prévues, gestion des stocks et des en-cours, adaptation et réactivité face aux variations du carnet commercial ou aux aléas… Cette dimension joue un rôle de plus en plus important, dans la mesure où les marchés sont de plus en plus versatiles et évolutifs et exigent des temps de réponse des entreprises de plus en plus courts. Il faut donc disposer de méthodes et d'outils de plus en plus performants pour l'organisation et la conduite de la production. Cette organisation de la production doit être vue non seulement au niveau de l'entreprise elle-même, mais aussi au niveau de sa position au sein d'une chaîne logistique dont elle constitue l'un des maillons, conduisant ainsi à une entreprise globale «virtuelle» orientée vers la satisfaction du besoin des clients aux meilleures conditions.

1.1.2. Processus de décision pour le pilotage de la production

Un modèle conceptuel classique de l'organisation d'une entreprise est montré sur la Figure 1.1. Celui-ci décompose l'entreprise en 3 sous-systèmes :

- un *sous-système de conception* intégrant la conception du produit (bureau d'étude) et les méthodes de fabrication (bureau des méthodes),
- un *sous-système de gestion* composé de la gestion prévisionnelle de la production et de la conduite de la production qui assure le pilotage temps réel de la production,
- un *sous-système de fabrication* qui regroupe l'ensemble des moyens physiques et humains du système de production.

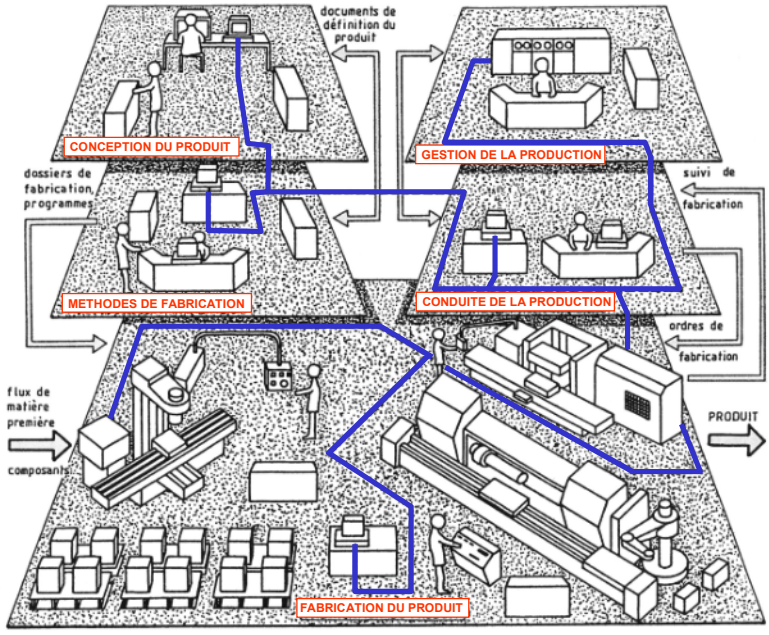


Figure 1.1 : Modèle général d'une entreprise

Au niveau du sous-système de gestion, cinq fonctions essentielles, organisées en étapes hiérarchisées sont classiquement identifiées, chacun des niveaux ayant une capacité de réaction propre. La Figure 1.2 décrit cette hiérarchie. Différentes terminologies peuvent être trouvées dans la littérature pour qualifier ces fonctions. Dans notre cas, nous nous appuyons sur celle donnée dans [Doumeingts G., 1990] :

- *la planification*

  Ce niveau a une vision à moyen et long terme sur la production de l'entreprise. Sur la base de statistiques, études de marché, un *Plan Industriel et Commercial (P.I.C.)* est d'abord établi. Ensuite, le *Plan Directeur de Production (P.D.P.)* est construit en réalisant un compromis entre les objectifs de marketing, financiers et de production. On se situe donc ici dans le niveau de décision tactique du système *entreprise* mais dans le niveau stratégique du système de production *atelier*.

- *la programmation*

  Cette fonction est chargée, à partir du plan directeur, d'établir un *programme prévisionnel de production*, à capacité infinie ou suivant une charge globale admissible par l'atelier. Celui-ci prend en compte les besoins bruts (commandes, délais demandés), les prévisions et calcule les besoins nets en fonction des stocks et des encours. La programmation peut être considérée comme une tâche du niveau tactique car elle consiste essentiellement à traduire les objectifs de la planification et reste dans une logique de définition du *quoi ?* de la production.

- *l'ordonnancement*

  Son objectif est le respect du plan prévisionnel (délais, quantité, qualité, etc.). Il élabore un *planning d'atelier* détaillé cherchant à optimiser l'utilisation des moyens de production en termes de charge, d'encours, de contraintes de succession ou technologiques. Il fournit un calendrier d'organisation du travail pour l'atelier (dates de début et de fin de chaque tâche), par ressource ou groupe de ressources. On se rapproche ici de la définition du *comment ?* de la production, et donc du niveau de décision opérationnel.

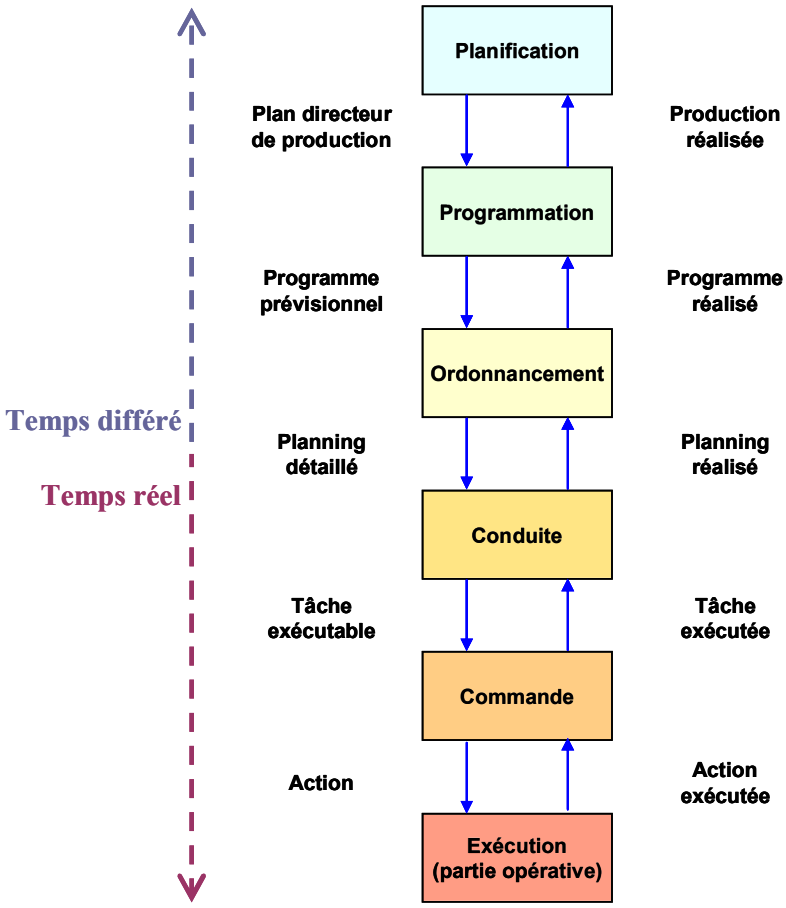


Figure 1.2 Fonctions de la Gestion de Production

- *la conduite*

  La conduite est chargée de réaliser la production prévue. Elle doit régler tous les problèmes non résolus par le niveau prévisionnel (charges ou contraintes locales). Elle doit aussi prendre en compte l'ensemble des contraintes de fabrication (contrôles de la qualité, arrêts liés à la maintenance, niveau de qualification du personnel, etc.), toutes présentes à ce niveau, et réagir aux aléas pour que la production prévue soit possible [Dindeleux E., 1992]. La conduite est donc typiquement une tâche du niveau opérationnel car les problèmes d'affectation de ressources devront être résolus en temps réel (réponses aux questions *quand* et *avec quoi* produire).

- *la commande*

  Ce niveau, directement en relation avec le système de production, a un rôle d'interface et d'interpréteur. Sa tâche essentielle est de traduire un ordre en une séquence d'instructions compréhensibles par la partie opérative. Il est concrétisé soit par un opérateur pilotant une machine et assurant le suivi de réalisation, soit par un automatisme capable d'interpréter un ordre et de renseigner la conduite sur l'état d'avancement de celui-ci.

Notons que la définition grossière des horizons temporels (temps réel et prévisionnel) indiquée sur la Figure 1.3 peut être affinée, la notion de temps réel étant assez ambiguë (d'une manière très générale, on peut considérer qu'un système est temps réel si son temps de réponse est inférieur à une limite imposée par son environnement).

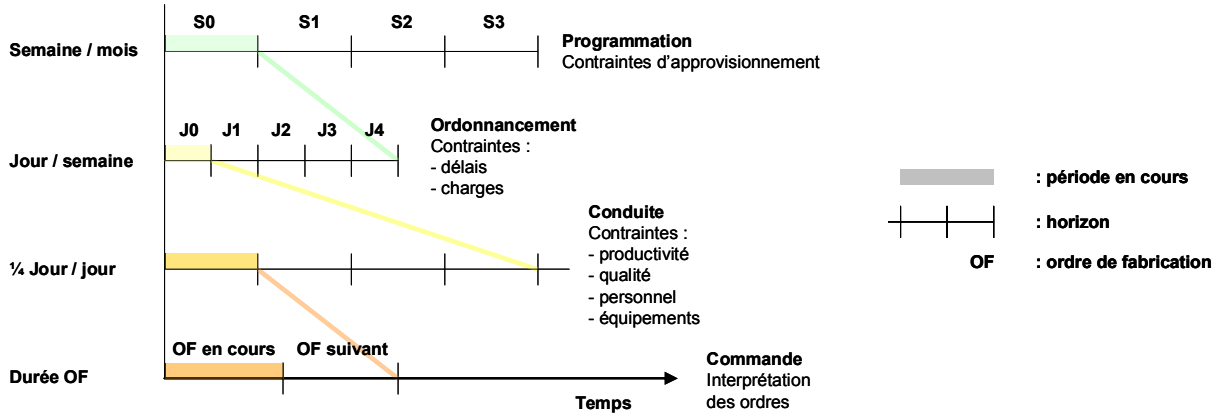


Figure 1.3 Horizons temporels

Dans ce cadre, la structuration des différents niveaux fonctionnels de la gestion de production se traduit par la définition d'intervalles de travail, chacun inclus dans celui du niveau supérieur. Ces intervalles induisent des contraintes de temps de réaction sur les processus décisionnels concernés. Au niveau de la conduite, cet horizon est généralement glissant et est plus ou moins étendu suivant le système et les délais de production. La Figure 1.3 décrit un exemple d'horizons temporels.

1.1.3. Modèle d'intégration des fonctions : le modèle CIM

Comme on l'a vu, chacun des sous-systèmes représentés sur la Figure 1.1 regroupe une ou plusieurs fonctions de l'entreprise qui s'appuient généralement sur un ensemble d'outils informatiques spécifiques (les *X.A.O.*). Dans un univers industriel de plus en plus soumis à la concurrence, la flexibilité et la qualité sont des atouts décisifs. Or, durant les années 80, le manque de cohérence dans la chaîne d'information de l'entreprise a amené une prise de conscience du besoin important d'intégration des systèmes informatiques tout au long de la chaîne de production. Ainsi, issu du développement de l'informatique, le modèle « Computer Integrated Manufacturing » (ou *CIM*) a été définit par un consortium d'industriels et a permis le décloisonnement des fonctions de l'entreprise industrielle.

Ce modèle (Figure 1.4) intègre les fonctions de conception assistée par ordinateur (*CAO* ou *CAD/CAM*, *computer-aided design/computer-aided manufacturing*), de gestion de la production (avec des systèmes de *GPAO* pour *Gestion de Production Assistée par Ordinateur* généralement basés sur la méthode *MRP2* pour *Manufacturing Resource Planning* et plus récemment avec des *ERP* pour *Enterprise Resources Planning*), de fabrication (avec les ateliers flexibles, les centres d'usinage à commande numérique), de stockage et de manutention automatisés ainsi que des méthodologies conceptuelles d'intégration de ces composants à l'intérieur d'un système global d'information de l'entreprise (par exemple, avec la méthode GRAI).

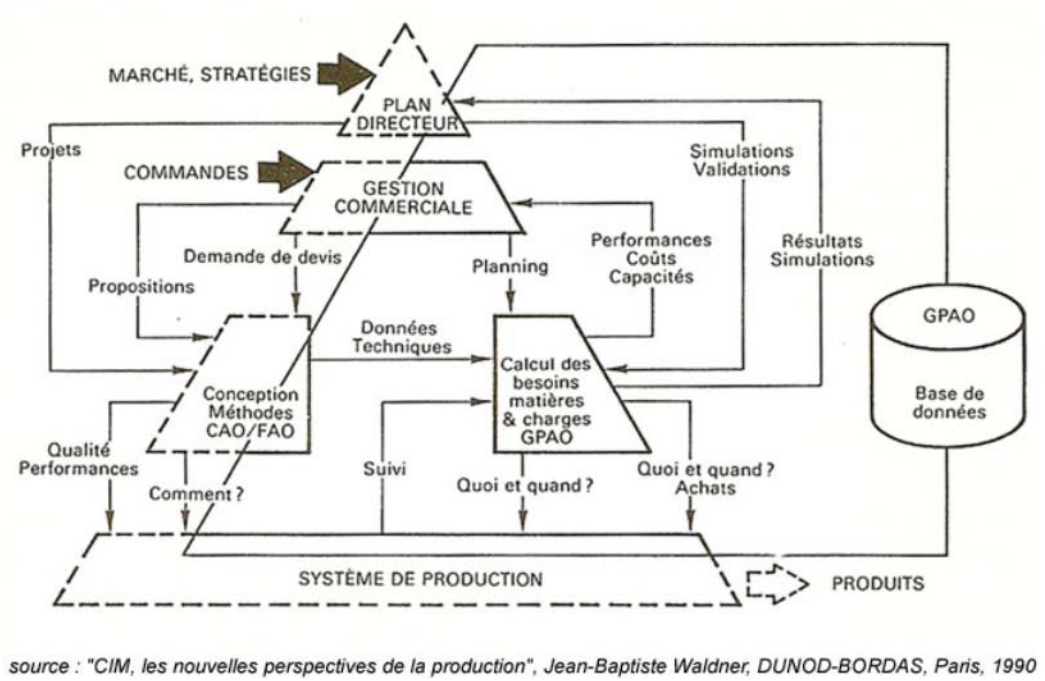


Figure 1.4 Fonctions intégrées dans le modèle CIM *[Waldner, 1990]*

Par ailleurs, la pyramide du *CIM* est une représentation conceptuelle qui comporte 4 niveaux auxquels correspondent des niveaux de décision.

- Le **niveau 3** correspond à la gestion prévisionnelle de la production, c'est-à-dire à la gestion des produits et des stocks, la gestion des approvisionnements, la gestion des clients, des commandes et de la facturation (gérés par les *ERP*)
- Le **niveau 2** est à l'interface entre le temps différé et le temps réel. Il réalise la localisation des produits en stocks, les mouvements physiques et la gestion des lots et assure la fonction de traçabilité (géré par le système de gestion d'entrepôt). Ces fonctions sont aujourd'hui intégrées dans les *MES* décrits dans la section suivante,
- Le **niveau 1** correspond aux équipements assurant la commande (automate programmable et régulateurs) et la surveillance du procédé (console de contrôle)
- Le **niveau 0** est constitué des capteurs et actionneurs.

Plus on s'élève dans cette pyramide, plus le niveau de décision/abstraction est important, plus la visibilité est globale et plus les horizons et cycles opérationnels s'allongent. Un niveau supérieur décide ce qu'un niveau inférieur exécute. Le couplage de ces différents moyens assure le pilotage du système industriel.

En fait, en essayant d'apporter une réponse à la quête de performance des entreprises, le concept *CIM* a mis en exergue les insuffisances de l'automatisation à outrance et la nécessaire communication entre les différents niveaux. La représentation la plus courante faisait apparaître un découpage structurel de l'organisation de l'entreprise, et une communication spécifique entre les niveaux du modèle. Si ce modèle est encore largement répandu et présente une certaine légitimité, il a rapidement mis en évidence un certain nombre de faiblesses, contraintes et limitations, dont notamment :

- *faiblesse structurelle* : les mécanismes d'échange sont construits autour d'un transfert vertical de l'information (niveau *N* vers *N-1* ou *N+1*) nécessitant une remise en forme de l'information à chaque interface.

- *limitation technologique* : l'évolution vers des architectures d'automatismes réparties et/ou distribuées associée à l'augmentation de la capacité de traitement des composants entraînent un accroissement des flux d'informations sur les médiums.

Paradoxalement, durant les années 90, qui ont vu l'émergence puis la croissance des *ERP*, le fossé s'est creusé entre les mondes de l'informatique et de la production. En 2000, les modules *ERM* (comptabilité/ finance, ressources humaines et gestion commerciale) et *GPAO* ont représenté plus des deux tiers des ventes de licences sur le marché des *ERP*. Parallèlement à ce constat, la dernière décennie a vu l'apparition, au travers des technologies Internet et des langages orientés objet, de nouveaux standards et mécanismes d'échange de données dans le monde de l'informatique.

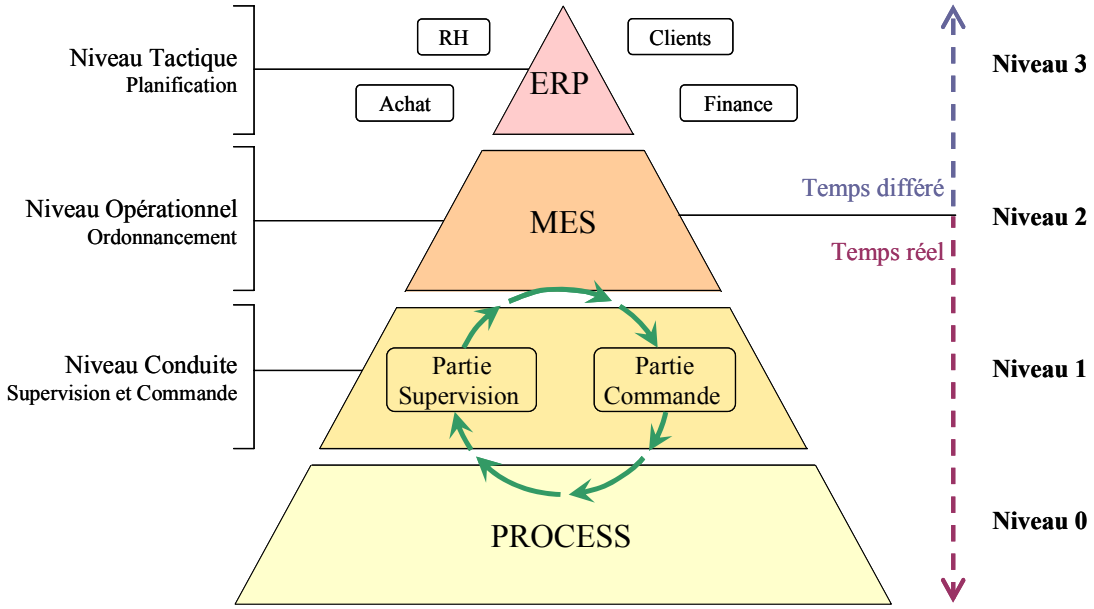


Figure 1.5 Place des MES *dans la pyramide du modèle* CIM

1.1.4. Outils de pilotage pour la maitrise des performances

La performance des entreprises est aujourd'hui indissociable de leur réactivité en terme :

- de réponse aux demandes du marché (il faut passer d'une production de masse à une production personnalisée et flexible),
- de délai de conception et de mise sur le marché de nouveaux produits,
- de prise en compte des normes et réglementations (traçabilité des produits en agroalimentaire),
- de besoin de communication permanent et en temps réel (la bonne information pour la bonne personne en tout lieu et à tout moment).

Pour satisfaire ces multiples exigences, différents outils sont proposés, chacun intervenant aux divers stades du processus de décision. Parmi ceux-ci, les *MES* et les *systèmes de supervision* et la *simulation dynamique* sont abordés dans la suite.

1.1.4.1. Les M.E.S. (Manufacturing Execution System)

L'ensemble des exigences énoncées précédemment a contribué à l'avènement de l'offre *M.E.S.* et a conduit à une modification sensible du modèle pyramidal du *CIM*. En effet, le *MES* par ses fonctionnalités, tant sur le plan du pilotage que de la supervision de l'atelier de production, se positionne en médiateur afin de réduire la fracture entre le monde de l'informatique de gestion et celui de la production.

Le domaine d'application du *MES* se situe donc entre les niveaux Contrôle-Commande (niveaux 1 et 2 du *CIM*), occupé par les automatismes et la supervision, et le niveau Planification (niveau 4 du *CIM*), occupé par les Progiciels de Gestion Industrielle, comme la *GPAO* et plus généralement aujourd'hui les logiciels de type *ERP*. Ces différents domaines se distinguent non seulement par leurs fonctionnalités mais aussi par leurs échelles de temps : alors que la planification travaille au mieux à la journée ou à la demi-journée, le *MES* devra être capable de réagir dans des durées de quelques minutes.

Il devient alors nécessaire de mettre à disposition des utilisateurs des outils d'aide à la décision leur permettant dans un temps limité :

- de prendre en compte les données issues de la supervision,
- d'analyser les scénarios possibles malgré la complexité des outils de production
- de modifier le plan de fabrication par interaction avec le module de supervision

Parmi les 11 fonctionnalités que devrait assurer un *MES*, une des principales fonctions est l'**ordonnancement**, qui se veut de plus en plus réactif. Cette fonction essentielle pour le pilotage de la production vise à organiser l'utilisation des ressources technologiques et humaines présentes dans les ateliers de l'entreprise pour satisfaire soit directement les demandes des clients, soit les demandes issues d'un plan de production préparé par la fonction planification de l'entreprise (cf. section 1.1.2).

1.1.4.2. Les systèmes de supervision

De nos jours, compte tenu du nombre croissant de variables nécessaires pour garantir leur fonctionnement, les unités de production sont de plus en plus instrumentées (capteurs, régulateurs, actionneurs) et sont couplées à un ou plusieurs calculateurs numériques, destinés à l'acquisition de données et à la mise en œuvre de l'automatisation. L'objectif premier de cette automatisation accrue est :

- l'augmentation des performances du système de production,
- la garantie de la qualité du produit fabriqué qui doit satisfaire un certain nombre de normes,
- la diminution des coûts de fabrication,

Face à ces flux d'informations dynamiques et variables, l'opérateur humain a besoin d'outils d'aide à la décision afin d'assurer de manière réactive et sûre la conduite de l'atelier. C'est le rôle tenu par les systèmes de supervision dont les fonctionnalités peuvent être plus ou moins étendues.

La **supervision** est située à l'interface entre la gestion hors ligne et les systèmes de commande. Elle consiste en la conduite d'installations industrielles tant en fonctionnement normal qu'en présence de défaillances et permet de suivre en temps réel des informations disponibles et ainsi de réagir face aux variations observées pour maintenir le procédé dans un mode de fonctionnement optimal.

La supervision englobe généralement des tâches de *pilotage* et de *surveillance* en temps réel de l'atelier, selon un certain degré d'abstraction [Dubois et Gentil, 1990]. Concernant le pilotage, cette fonction induit souvent une volonté d'autonomie vis à vis des opérateurs (source de reproductibilité) et

une réactivité relative au mode de fonctionnement. Ceci est possible grâce à l'utilisation d'automates programmables industriels (*A.P.I.*) et de contrôleurs locaux. Quant à la surveillance, l'aspect décisionnel est encore aujourd'hui souvent du ressort de l'opérateur à travers l'utilisation d'écrans synoptiques, véritables tableaux de bord du système piloté.

Le *pilotage en temps réel* consiste à exécuter la séquence opératoire de chaque recette et assure en temps réel la gestion des ressources nécessaires à cette exécution. Cette fonction est donc située au-dessus des systèmes locaux de commande et est chargée :

- de gérer (lancer, arrêter, suspendre, etc.) des lois de commande prédéfinies (programmes automate, paramètres de régulation, etc.),
- de gérer le transitoire correspondant aux changements de régime du système (démarrage, changement de points de fonctionnement, arrêt).

Pour cela, il met en œuvre en temps réel les ordres de fabrication (consigne) provenant de la fonction ordonnancement en traduisant cette information sous forme de tâche de commande et en la distribuant aux automates et contrôleurs locaux.

Si les objectifs du pilotage résident essentiellement dans le respect des critères de qualité, de productivité et de tenue des délais, la fonction de *surveillance* veille au bon déroulement de la fabrication. Dans ce but, elle vérifie de manière récurrente la conformité de l'évolution dynamique des opérations effectuées sur le système de production en se référant à une situation de fonctionnement normal que l'on cherche à maintenir pour assurer les conditions nominales de fabrication. Ceci induit que la surveillance doit adapter la commande si une déviation est observée [Combacau et Courvoisier, 1990]. Adapter la commande suppose:

- de constater tout d'abord les déviations (ce qui est assuré par la fonction de *détection*),
- d'en trouver la cause (ce qui est géré par la fonction *diagnostic*),
- et de déterminer la réponse adéquate (ce qui est établi par la fonction *reprise*). Cela va de la modification du plan de fabrication par interaction avec le module d'ordonnancement, à la modification de conditions opératoires par interaction avec les niveaux de commande inférieurs.

Compte tenu de ces fonctionnalités, il est fréquent aujourd'hui que la supervision soit aussi chargée :

- de l'amélioration de la sécurité de l'installation industrielle, des hommes qui y travaillent et de son environnement,
- de la traçabilité du produit.

1.2. SYSTÈMES DE PRODUCTION CONSIDÉRÉS DANS CES TRAVAUX

Cette section se propose de présenter et de caractériser les systèmes de production considérés plus spécifiquement dans ces travaux.

1.2.1. Le Génie des Procédés

Apparu au moment où l'industrie chimique a connu son essor le plus important, le Génie des Procédés peut être défini comme « la science des systèmes complexes qui recouvre l'ensemble des connaissances et des méthodes nécessaires à la conception, à la mise en œuvre et au pilotage des procédés industriels de transformation de la matière et de l'énergie ».

Le Génie des Procédés est donc né du besoin de mieux comprendre le fonctionnement des appareils, de les classer en grandes catégories, de dégager des lois dont la connaissance permet de relier entre eux des appareils à première vue fort différents, etc. Ce besoin de rationalisation a conduit au développement de concepts novateurs, avec comme objectif final, une meilleure conception et une meilleure utilisation des appareils.

Cette discipline recouvre des secteurs industriels très variés tels que :

- la chimie lourde (le pétrole, le charbon, l'électrochimie industrielle, la métallurgie, la pétrochimie, le nucléaire, le ciment, le papier, le verre, le plastique, le textile, etc.) ;
- la chimie fine (la pharmacie, la cosmétologie, la photographie, etc.) ;
- la biochimie (l'agroalimentaire, la bio-industrie, etc.).

1.2.2. Opérations unitaires composant une unité de production

Les appareillages nécessaires pour la réalisation industrielle de réactions chimiques sont très variés. Néanmoins, il est apparu qu'ils faisaient appel à un nombre restreint de techniques appelées *opérations unitaires*. Par ailleurs, le processus de fabrication d'un produit chimique est réalisé par un ensemble d'étapes caractéristiques dans lesquelles sont mises en œuvre différentes opérations unitaires (cf. Figure 1.6).

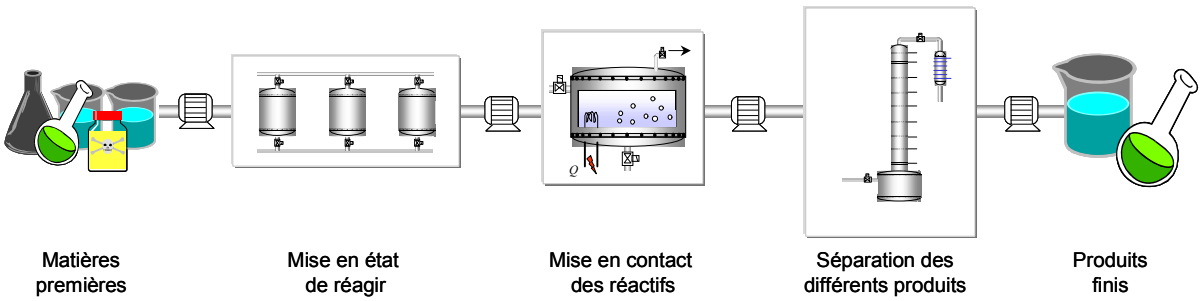


Figure 1.6 Étapes de fabrication d'un produit chimique

Un procédé complet apparaît alors comme une association de plusieurs opérations unitaires.

- *Stockage et transfert de matières premières :*

  Les matières premières sont stockées dans des récipients appropriés puis acheminées par divers mécanismes (ex : différence de pression, gravité, pompes ou moyens mécaniques, etc.) vers le réacteur.

- *Préparation des matières premières*

  Les matières premières doivent être amenées dans un état leur permettant de réagir, par des opérations de broyage, de mélange, de chauffe, de dilution, de dissolution, etc. Des matières solides doivent par exemple être soumises à des opérations de broyage afin d'atteindre le degré de finesse souhaité ; elles peuvent également être mises en suspension dans un solvant ou un diluant.

- *Mise en contact des réactifs dans des conditions opératoires appropriées*

  La mise en contact s'effectue au sein d'un réacteur et nécessite généralement des opérations de réaction, d'agitation, de chauffe, de refroidissement, de fermentation, de pasteurisation, etc. Dans le cas d'un système multiphasique, un agitateur approprié à la nature et à la consistance du milieu s'avère nécessaire. Le réacteur doit par ailleurs être maintenu manuellement ou automatiquement dans des conditions opératoires adaptées (température, pression, pH, etc.). Des réactifs doivent pouvoir également être ajoutés ou soustraits en cours de réaction (par exemple, par distillation). Bien qu'il constitue le maillon essentiel de la chaîne, le réacteur ne représente généralement qu'une faible fraction de l'appareillage.

- *Séparation des produits résultant de la réaction*

  Une fois la réaction terminée, la masse réactionnelle contient le produit recherché, les produits secondaires qui ont pu se former, les matières premières non transformées, des impuretés, les solvants ou diluants utilisés, etc. Il est alors nécessaire de séparer les différents produits obtenus. Chaque problème de séparation est un cas d'espèce, mais il est généralement résolu par la combinaison de plusieurs techniques séparatives. Les opérations de séparation les plus courantes sont les opérations de décantation, de filtration, de sédimentation, de centrifugation, de cristallisation, de distillation, de rectification, d'absorption, d'adsorption, d'extraction par solvant, de séchage, de tamisage, etc.

Pour la plupart de ces opérations, un apport ou une évacuation d'énergie est souvent requise, notamment pour les opérations de séparation. Cette énergie peut se présenter sous plusieurs formes :

- énergie thermique (ex : mise à température, vaporisation, condensation, etc.) ;
- énergie thermique transformée en énergie mécanique (ex : agitation créée par ébullition, transport par vaporisation, etc.) ;
- énergie mécanique ou électrique (ex : broyage, agitation, transport, compression (froid), électrolyse, etc.).

Si tout procédé peut toujours être vu comme une succession d'opérations unitaires, il peut être néanmoins caractérisé par la manière dont ces opérations unitaires sont interconnectées et par la manière dont la matière transite dans l'unité de production. Pour cette raison, on le distingue généralement par :

- le *nombre de produits réalisés* dans l'unité,
- le *mode de production* mis en œuvre dans l'unité qui dépend de la nature des flux la traversant.

1.2.3. Caractérisation des unités selon le nombre de produits fabriqués

Une classification générale des systèmes de production est proposée dans [Andreu 96]. Une classification plus spécifique des systèmes discontinus est proposée par [Daubas 94], inspirée de [Rippin 83]. Celle-ci propose de les classer :

- en *unités mono-produit* : dans ce cas, chaque lot suit la même recette de fabrication. Des variations sont cependant possibles au niveau des procédures et des paramètres afin de compenser par exemple des changements de qualité de matières premières ou d'appareils, ou bien afin d'optimiser le processus.
- en *unités multi-produit* : dans ce cas, l'unité peut réaliser plusieurs produits. Ceux-ci sont fabriqués selon la même procédure opératoire mais en utilisant différentes valeurs de formules (composition en matière première ou durées opératoires différents…). La configuration de l'atelier est unique. Ce type d'atelier est similaire au flow-shop des systèmes manufacturiers : tous les produits suivent la même séquence d'opérations et il y a un unique chemin possible.
- en *unités multi-objectif* : ici, les produits sont fabriqués en utilisant des procédures différentes, notamment au niveau des politiques d'affectation des appareils : cela conduit à des trajectoires de flux physiques différentes pour chaque produit. Ces unités sont analogues au « Job Shop » des systèmes manufacturiers.

1.2.4. Caractérisation des unités selon le mode de production

Un procédé peut être vu comme un système traversé par un ensemble de flux. Parmi ceux-ci, on distingue :

- les *flux physiques* qui permettent un transport de matière ou d'énergie. Au cours des différentes étapes de fabrication du produit chimique, les flux de matière transitent au travers des appareils et subissent des transformations physiques et chimiques. Les flux d'énergie peuvent être apportés au procédé sous forme thermique, électrique ou mécanique ; ils peuvent également résulter des transformations mises en jeu au sein du procédé (réaction exothermique par exemple).
- les *flux d'information* qui permettent un échange d'information entre les différents appareils du procédé. Ce type de flux est largement utilisé au niveau du contrôle et de la régulation du procédé (ex : capteurs, détecteurs, systèmes numériques de contrôle/commande, etc.).

Le plus souvent, la nature des flux physiques caractérise le **mode de production** mis en œuvre dans le procédé.

1.2.4.1. Flux de matière continu ⇔ mode de production continu

Traditionnellement, l'industrie de process fonctionnait plutôt selon un mode de production continu. Ce mode de production concerne généralement des entreprises ayant de grandes capacités de production, fabricant des produits de base assez semblables destinés à alimenter des marchés relativement stables et bien connus (ex : industries pétrochimiques). Les produits ont une nature continue : les matières traitées présentent par ailleurs des caractéristiques voisines et se trouvent principalement à l'état fluide (ex : gaz ou liquide). L'appareil a une fonction unique : il a en charge tout ou partie d'une seule opération unitaire. De même, la séquence des opérations à effectuer est complètement assimilée au flowsheet et l'unité est dédiée à une seule et même production pendant toute la durée de la campagne. La configuration de l'atelier ne change pas et l'état du produit est en général donné par un vecteur d'état en entrée et en sortie de chaque appareil. Il définit les valeurs des variables d'état comme la pression, la température, les

débits ou les compositions chimiques. Ces variables ne varient pas puisque l'unité fonctionne en régime stationnaire. Ce régime permanent est évidemment illusoire, le rôle des boucles de régulation est justement de faire les ajustements nécessaires pour que le système ne s'écarte pas du point nominal de fonctionnement (cf. Figure 1.7).

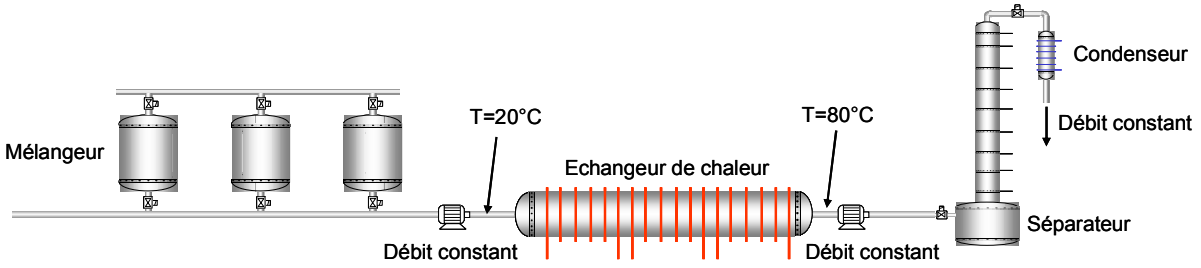


Figure 1.7 Fonctionnement selon un mode de production continu

Un système de production continu possède une phase principale de production en régime stationnaire, et se caractérise donc par (cf. Figure 1.8) :

- une production organisée en longues campagnes et dédiée à des installations spécifiques ;
- des phases de régime permanent largement prédominantes par rapport aux phases de régime transitoire ;
- des opérations unitaires réalisées simultanément dans des appareils distincts ;
- des flux de matière ou d'énergie constants dans le temps qui traversent les appareils composant le procédé de façon ininterrompue et formant un flux permanent (ou voulu comme tel) en sortie d'atelier ;
- peu de cuves tampons.

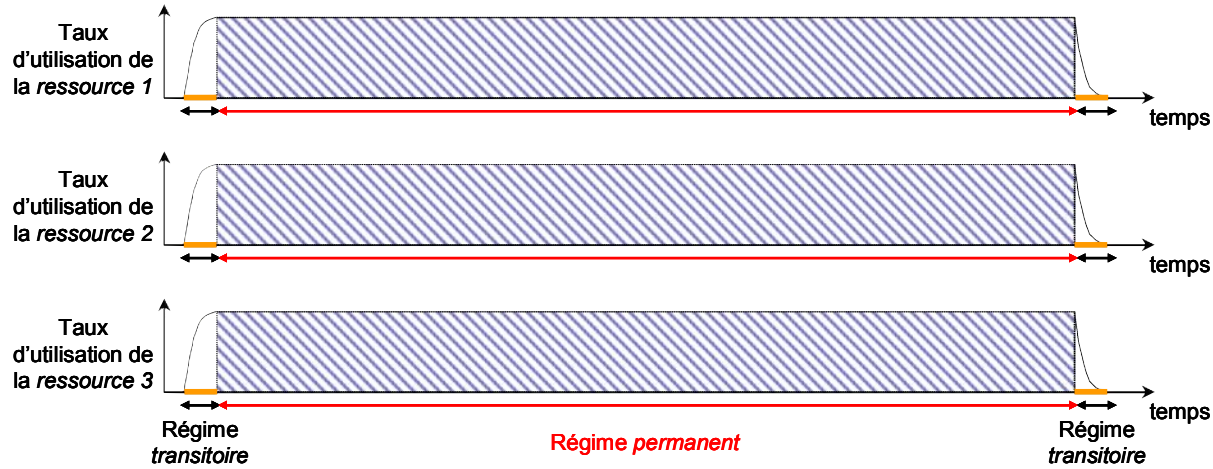


Figure 1.8 Utilisation des ressources en production continue

Le dimensionnement des unités continues implique de déterminer la taille optimale des appareils qui permettra d'atteindre le débit nominal de production souhaité en minimisant certains critères comme les coûts d'investissement, de fonctionnement ou bien l'impact environnemental.

1.2.4.2. Flux de matières discontinus ⇔ mode de production par lot

Un mode de production historiquement du secteur manufacturier

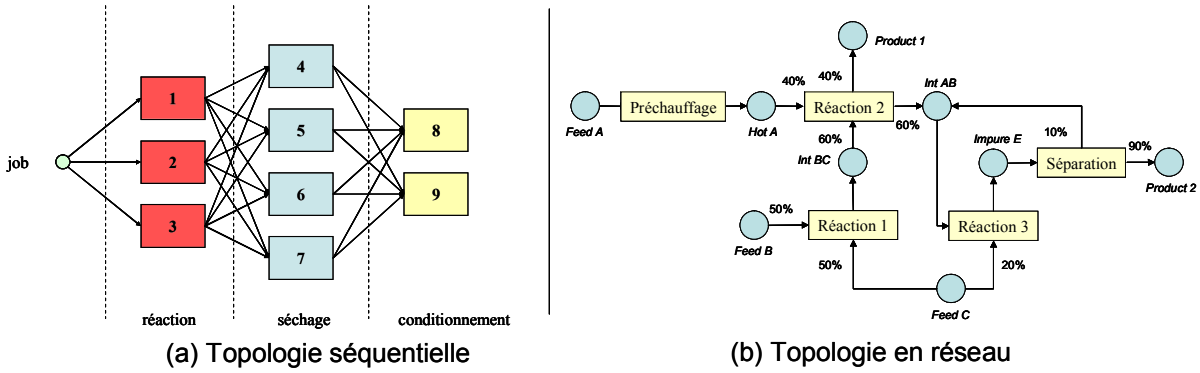
Les secteurs industriels concernés spécifiquement par ce mode de production sont très variés et qualifiés de « manufacturier » (mécanique, automobile, aéronautique). Ici, la matière transformée est de nature discrète. Ce sont des pièces dénombrables, même à fort débit. Les pièces circulent d'une machine à l'autre pour y subir des transformations (usinages) nécessaires à l'obtention du produit fini. Une gamme d'opérations définit la séquence des actions à réaliser pour transformer la matière première en produit fini. Une pièce peut être stockée entre deux opérations (en cours). Son état ne change généralement pas durant cette période. Les lots regroupent des ensembles de pièces ayant une même gamme et qui vont suivre le même cheminement dans l'atelier avec les mêmes traitements. Installations dites « flexibles », elles peuvent produire plusieurs types de pièces ayant des gammes différentes. Cette flexibilité peut être mise en œuvre soit de manière alternée, au travers de campagnes de production durant lesquelles un seul type de pièces est fabriqué, soit de manière simultanée et dans ce dernier cas, plusieurs types de pièces sont usinés en même temps dans l'atelier. Cette flexibilité, où les pièces ont plusieurs cheminements possibles, est un facteur de complexité aussi bien dans la phase de conception de l'atelier, que dans la gestion de la production pendant l'exploitation.

Ces systèmes de production sont en général modélisés comme des *systèmes à évènements discrets*, dont seulement les états les plus importants sont représentés. La conception de ce type d'installation passe par la détermination des capacités de traitement des appareils mais aussi par l'ordonnancement des opérations dans la mesure où plusieurs cheminements sont possibles pour les pièces.

Ateliers de production discontinus

Dans l'industrie de process, ce mode de production par lot s'est plus récemment implanté, conduisant à la mise en œuvre d'*ateliers discontinus*. En effet, au cours des trente dernières années, le contexte économique dans lequel s'inscrit le domaine du Génie des Procédés, s'est montré de plus en plus concurrentiel. Cette évolution se traduit par l'intérêt croissant pour des modes de production batch. En effet, ils permettent généralement de fabriquer plusieurs produits différents en utilisant les mêmes ressources (équipements, opérateurs, cuves de stockage, matières premières, utilités, etc.).

On trouve deux grands types de topologies pour les procédés batch. La plus fréquemment rencontrée est la topologie séquentielle (cf. Figure 1.9.a) pour laquelle un lot suit toujours la même succession d'étapes de fabrication définies dans la recette. Une étape peut être réalisée par une ou plusieurs ressources identiques, disposées en parallèle. Cependant, la complexité croissante des recettes de fabrication, couplée au besoin de conception d'ateliers flexibles conduit à l'utilisation de topologies en réseau (cf. Figure 1.9.b). Ces dernières autorisent un plus grande flexibilité dans le traitement des lots par l'utilisation de cheminements variables (séparation de courants, recyclages…)



(a) Topologie séquentielle (b) Topologie en réseau

Figure 1.9 Topologies des procédés batch

Par ailleurs, ces systèmes sont privilégiés dans les industries pharmaceutiques, cosmétiques, de l'agroalimentaire, des polymères ou de la biochimie car la forte valeur ajoutée de ces produits et leur fragilité nécessitent des productions en lots restreints et bien identifiés.

Les unités de production discontinue permettent aussi de répondre à une demande de plus en plus exigeante en termes de qualité et spécificité des produits auxquelles s'ajoutent aujourd'hui, des contraintes de sécurité et de réduction des délais. Dans ce contexte, la production doit être flexible, dynamique et réactive.

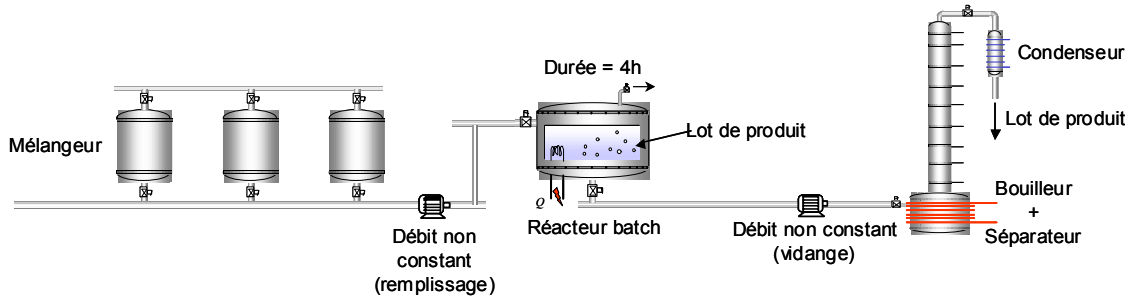


Figure 1.10 Fonctionnement selon un mode de production discontinu

Un mode de production discontinu se caractérise par (cf. Figure 1.10) :

- une production organisée en petits lots de produits diversifiés se partageant les mêmes ressources;
- des opérations unitaires réalisées successivement dans le même appareil sur un lot de produit ;
- des flux de produit variables et discontinus ;
- des phases de régime transitoire non négligeables ;
- de nombreuses cuves tampons.

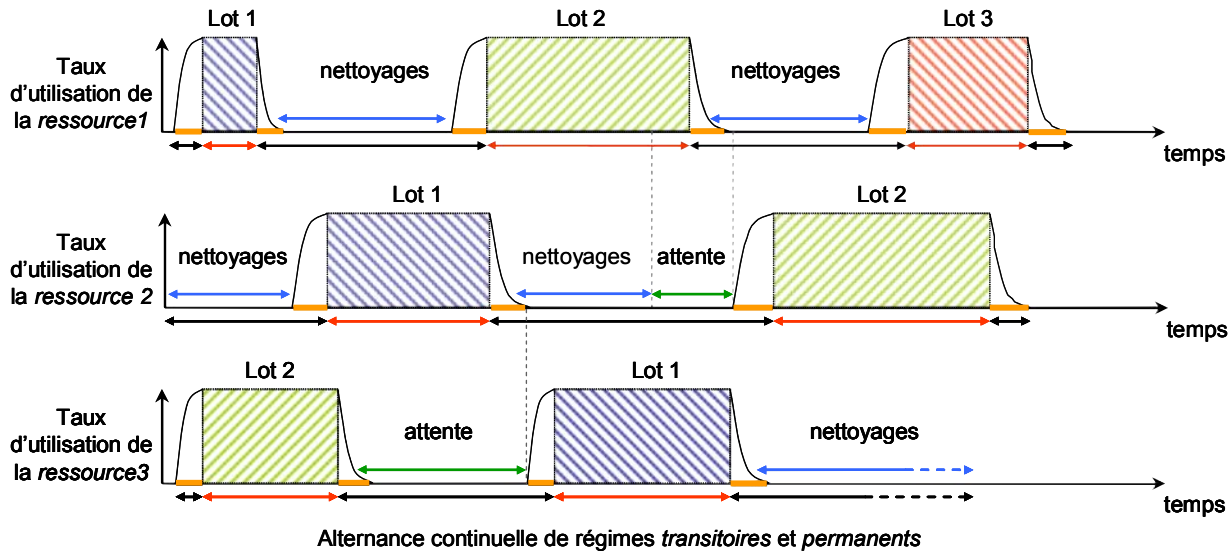


Figure 1.11 Utilisation des ressources en production discontinue

Dans ce mode de production, les lots (ou batchs) sont nombreux mais constituent des volumes de production faibles. De ce fait, les temps d'inutilisation ou les régimes transitoires se multiplient, conduisant globalement à une baisse de la productivité des ressources (cf. Figure 1.11). Les performances de l'installation résident ainsi essentiellement dans :

- l'optimisation de chaque opération unitaire, aussi bien en régime permanent que transitoire,
- et l'établissement d'ordonnancements minimisant les temps d'inactivité et de mise en fabrication.

Ateliers de production semi-continus

Les unités de production semi-continus fonctionnent à la fois selon des modes de production continus et par lots. Elles allient des opérations unitaires impliquant un mode de production continu et des opérations unitaires fonctionnant en mode de production par lots. L'enjeu du pilotage de ces ateliers réside dans la gestion pertinente des phases de transition entre les différents modes de production, notamment afin de limiter les en-cours de production tout en évitant les ruptures de stocks qui peuvent pénaliser fortement une campagne (ex : redémarrage d'une colonne).

De façon générale, la politique de transfert des lots entre appareils est un facteur important de productivité d'une unité batch. En effet, la possibilité de pouvoir accumuler des produits intermédiaires dans des cuves tampon de capacité finie offre un maximum de souplesse dans la gestion des lots en découplant les phases de vidange et de remplissage de deux opérations de transformation successives. Une cuve tampon permet aussi de modifier le volume du lot entre les opérations amont et aval, avec un choix de volume appartenant à une gamme continue. La présence d'un réservoir tampon représente aussi la configuration la plus classique pour la transition de matière entre deux zones de procédés ayant un mode de fonctionnement différent, c'est à dire continu pour l'une et discontinu (ou batch) pour l'autre.

Lorsque la matière circule de la zone discontinue vers la zone continue du procédé, l'activité de la zone aval est cautionnée par une quantité suffisante de matière dans le réservoir. Si le débit de soutirage de la zone continue provoque une baisse de niveau permanente du réservoir qui n'est pas compensée par

l'arrivée des lots de la zone amont, l'occurrence d'une rupture de production de la zone continue est certaine sur un horizon de temps suffisamment long. La gestion de telles situations est particulièrement délicate dans la mesure où la remise en régime d'une installation continue peut être longue. Dans l'hypothèse inverse, si le soutirage continu est trop faible, il y aura dépassement de la capacité de stockage et l'activité de l'atelier discontinu devra être freinée. Cette situation paraît plus facile à gérer sauf quand les lots ne peuvent patienter dans les appareils qu'un temps limité. Dans de telles configurations, le respect des dates de lancement d'opération dans la zone discontinue devient une contrainte prépondérante. Notons que la rupture de production pourrait être évitée en adaptant le régime de marche du continu, mais cela requiert alors un dimensionnement de la ligne très tolérant vis-à-vis des variations de charges possibles, ce qui est rarement le cas.

Pour un sens de circulation allant du continu vers le discontinu, la gestion du réservoir tampon obéit aux mêmes principes pour des causes différentes. La rupture de production du procédé continu est déclenchée par l'atteinte d'une valeur seuil maximale de la capacité du réservoir. Inversement, l'absence de quantité matière suffisante dans le réservoir pour lancer la fabrication d'un lot diffère son démarrage et diminue la productivité de la chaîne discontinue.

1.2.5. Notion de recette

La *recette* est définie comme « une entité comprenant l'ensemble minimal d'informations définissant sans équivoque les prescriptions de fabrication pour un produit spécifique. Les recettes permettent de décrire les produits et la manière de les fabriquer » [IEC 61512-1, 1997].

Le contenu de la recette est exploité à différents niveaux décisionnels avec une granularité adaptée (recette agrégée a un niveau tactique et recette très détaillée à un niveau supervision). Dans le cas de procédés constitués d'un grand nombre d'appareils ou de procédures de fabrication élaborées, la recette peut rapidement devenir très complexe. Afin d'établir une approche standard pour traiter la complexité du contrôle/commande des procédés batch, la norme ISA/SP88 (www.isa.org) propose une modélisation hiérarchisée de la recette.

Elle est constituée de cinq éléments d'information (Figure 1.12) :

- l'*en-tête* qui regroupe les informations administratives (identifiant, auteur, etc.),
- la *formule* qui indique la liste des matières premières et des produits intermédiaires nécessaires ainsi que leur proportion (en pourcentage) et les conditions opératoires à appliquer,
- les *besoins* en équipements, ou type d'équipements,
- la *procédure* définissant la suite d'opérations unitaires ordonnées dans le temps qui décrit la fabrication du produit,
- des informations diverses liées aux contraintes de qualité et de sécurité.

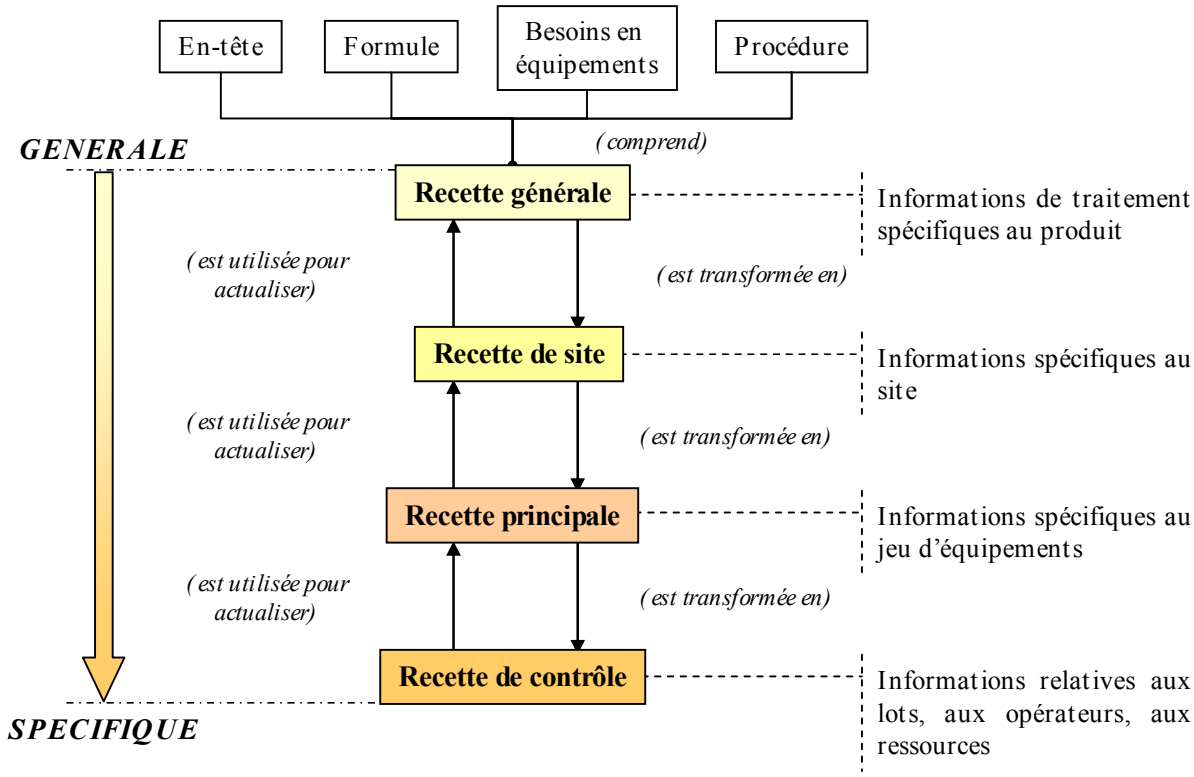


Figure 1.12 Les types de recettes et leurs contenus

Plusieurs types de recettes sont donc définis suivant la granularité de l'information présente (Figure 1.12) :

- la *Recette générique (ou générale)* spécifie la méthode d'obtention du produit indépendamment des équipements utilisés ou d'une campagne particulière. Il s'agit généralement de la recette fournie directement par le *service « industrialisation »*
- la *Recette de site* (en complément de la recette générique) précise les caractéristiques des équipements et des ingrédients requis compte tenu des ressources disponibles sur le site de production. Cette recette est utilisée généralement pour la *planification* ;
- la *Recette principale* est une recette de site adaptée à une campagne particulière (taille des lots définis), l'équipement est décrit par classe (caractéristiques telle que la capacité) mais aucun appareil particulier n'est encore désigné. Ce niveau de recette est mis en œuvre pour la *fonction d'ordonnancement*.
- La *Recette lot* (ou de contrôle) est une recette appliquée à un lot particulier auquel elle est associée pendant tout son processus de fabrication. Elle correspond à l'exécution en temps réel de la recette principale avec une affectation effective des équipements et est utilisée au niveau de la *supervision*.

Cette décomposition illustre le fait qu'une recette de niveau inférieur est tout simplement une *instanciation* de la recette de niveau supérieur dans laquelle des données de production plus spécifiques sont ajoutées. Comme le montre la suite, cet aspect est relativement important, notamment pour la structuration des modèles de simulation.

1.3. SIMULATION DYNAMIQUE DES PROCÉDÉS DISCONTINUS

Si les outils décrits dans la section 1.1.4 sont exploités pour la *gestion en ligne* d'un atelier, d'autres sont utilisés *hors ligne* pour analyser leur comportement dans différents contextes. Parmi les outils d'analyse des systèmes de production, la simulation reste aujourd'hui un des moyens les plus prisés au niveau industriel et cela, tout particulièrement dans le secteur du procédé.

1.3.1. Les différentes utilisations de la simulation

Les outils de simulation ont la particularité d'offrir à l'utilisateur la possibilité de modéliser le système en détail mais de manière relativement simple puis « d'expérimenter » l'unité sur ordinateur en utilisant différentes initialisations ou conditions de fonctionnement.

La simulation a donc un intérêt certain car :

- en *phase de conception*, elle permet d'aider au dimensionnement d'une installation et de réduire notablement le travail sur pilote qui s'avère souvent coûteux et consommateur de temps.
- en *phase de réalisation*, elle peut permettre d'anticiper la formation des opérateurs à la conduite routinière de l'unité, mais aussi de les entraîner à la gestion des phases délicates (démarrage, arrêt, changement de point de fonctionnement, etc.) ou en cas d'incident,
- en *phase d'exploitation*, un modèle de simulation améliore la compréhension du système de production entier par les membres de l'équipe et facilite la communication. Par ailleurs, elle permet de tester hors ligne différentes politiques de gestion ou le comportement de l'atelier dans différentes situations sans avoir à intervenir sur le système physique, point particulièrement intéressant lorsque la production ne peut être stoppée aisément ou qu'elle comporte des risques.

De manière générale, les logiciels de simulation sont utilisés pour réaliser différentes tâches d'analyse statique telles que l'établissement des bilans matière et énergie, l'estimation de la taille des équipements, l'estimation de la demande en utilités en fonction du temps, l'estimation des temps de cycle ou l'analyse des coûts.

Par ailleurs, la simulation permet de réaliser en une poignée de secondes des analyses du type *« What-if »* ou des analyses de sensibilité extrêmement utiles aux ingénieurs dans leur travail quotidien. Cette approche se rattache au concept plus général de Systèmes Interactifs d'Aide à la Décision (SIAD) qui a été introduit par l'école anglo-saxonne et résulte de la traduction de Decision Support Systems (DSSs). Un des tous premiers auteurs à avoir introduit et diffusé la notion de Systèmes de Décision et de Gestion (Management Decision Systems) est Scott Morton [Scott Morton M., 1971]. Le support ainsi fourni se traduit par une amélioration de la qualité de prise de décision plutôt que par une amélioration de son efficacité. Le processus de décision assisté par des SIAD est alors envisagé comme une succession d'allers / retours entre le décideur (utilisateur) et la machine ; l'automatisation de tels processus n'est donc pas envisageable [Zaraté P., 2005].

L'objectif de telles études est d'évaluer l'impact de paramètres critiques sur des indicateurs de performance clefs tels que les coûts de production, les temps de cycle, les taux d'occupation des ressources ou la productivité. Par exemple, l'impact d'un retard sur une tâche ou de l'augmentation de la taille d'un lot (qui affecte la durée de toutes les tâches dépendant de la taille des lots) sur le temps de cycle et sur le nombre de batch peut être évalué instantanément. La simulation fournit aussi le moyen d'évaluer l'occupation des ressources, le niveau des files d'attente devant les appareils durant un cycle. Ce même

type d'étude est utilisé en analyse de risque ou pour tester les procédures de mise en sécurité des équipements et du personnel en cas d'incident.

Au niveau du contrôle, la simulation peut être utilisée pour définir les conditions opératoires de chaque opération et pour paramétrer les boucles de régulation nécessaires au maintien de ces conditions opératoires. Plus globalement, elle permet de tester hors ligne les programmes de commande exécutés par les automates programmables et d'éliminer préventivement certains « bugs ».

Les équipements étant rarement idéalement dimensionnés dès la conception, il n'est pas rare que des ajustements soient requis quand une nouvelle unité est implantée. De même, compte tenu du contexte économique fluctuant, des évolutions dans les objectifs de production sont souvent nécessaires au cours de la vie d'une unité. Dans les deux cas, la simulation est un outil précieux pour ajuster certains paramètres ou pour revamper une installation.

Enfin, notons que certains systèmes de supervision avancée basés sur le principe de redondance analytique utilisent un modèle de simulation comme modèle de référence (en fonctionnement normal ou anormal). Celui-ci est simulé en temps réel et de manière synchrone avec l'évolution du système de production supervisé et permet de détecter les défaillances et les dérives et dans certains cas, d'établir un diagnostic sur le composant défaillant.

1.3.2. Les différents types de simulateur

Si la simulation parait incontournable pour de nombreuses études, il existe néanmoins une vaste gamme d'outils adaptés aux systèmes de production à modéliser et au type d'analyse à mener.

Classiquement, deux types de système à simuler sont distingués : les *systèmes à dynamique continue* et les *systèmes à événements discrets*.

Les *systèmes à dynamique continue* font appel pour leur description à des outils mathématiques tels que les équations différentielles. Ils peuvent donc être vus comme un système multi-variable caractérisé par un vecteur d'état réel de dimension finie. Les variables d'état sont continues et définies sur l'ensemble des réels; leur évolution est continue dans le temps (ex : température ou composition). Ainsi, un système continu peut être décrit par un ensemble d'équations différentielles ordinaires (*EDO*) et/ou algébriques (*EDA*), qui s'écrit sous la forme implicite générale suivante :

$$F\left(x^{(n)}, x, p, \theta\right) = 0$$
$$x_{\theta=0} = x_0$$
$$x^{(n)}_{\theta=0} = x^{(n)}_0$$

avec F : jeu d'équations (différentielles et/ou algébriques) ;
x : ensemble des inconnues du système ;
$x^{(n)}$: ensemble des dérivées à l'ordre n des inconnues par rapport à la variable indépendante ;
p : ensemble des paramètres opératoires du système ;
θ : variable indépendante (généralement le temps).

Les simulateurs permettant de traiter ce type de systèmes sont par exemple, les simulateurs dynamiques de procédés tels *ProSim Batch* ou *Aspen Dynamics* ou des simulateurs dédiés au contrôle tel que *Simulink*.

Les *systèmes à évènements discrets* (ou *SED*) sont classiquement représentés par l'algèbre de Boole combinée à des formalismes états/transitions ayant une représentation graphique (Grafcet, réseau de Petri, automates à états finis, etc.). Ils peuvent être vus comme un système dynamique dont l'espace continu a été discrétisé en un espace d'états discrets (ou symboliques). Pour ces systèmes, la transition permettant le passage d'un état discret à un autre état discret dépend de l'occurrence synchrone ou asynchrone d'événements (franchissements de seuils). Ces évolutions, nommées trajectoires, sont basées sur une

succession d'états et de transitions. Les systèmes de production de type manufacturier, les réseaux informatiques, les systèmes de trafic ferroviaire, routier, aérien en sont des exemples. Ces systèmes sont constitués d'un nombre fini de ressources (ex : machines, connexions entre ordinateurs, voies ferrées) partagées par plusieurs utilisateurs (ex : des produits, des paquets de données, des trains) contribuant tous à un même objectif (ex : fabriquer des produits finis, transmettre des données, transporter des marchandises).

Le comportement dynamique d'un *SED* ne tient pas compte des informations temporelles et ne s'attache qu'à l'ordre d'occurrence des événements. Ces systèmes sont dits *SED non temporisés*. Un simulateur traitant ce type de système est par exemple *ARENA* (Rockwell).

Néanmoins, les systèmes réels sont souvent des systèmes complexes dont la dynamique est modélisée, d'un point de vue macroscopique, par des phénomènes discrets et continus, démontrant ainsi que la frontière entre systèmes continus et systèmes à évènements discrets n'est pas si étanche. Il apparaît alors évident que ces systèmes ne peuvent se contenter d'une représentation homogène à dynamique purement continue ou purement événementielle. Depuis les années quatre-vingt-dix, une attention particulière de la part de la communauté scientifique s'est portée sur l'étude de ces systèmes dits «hybrides» [Alur et al., 1995 ; Branicky, 1995 ; Engell, 1997 ; Guéguen et Lefebvre, 2001] et de nombreuses approches de modélisation traitant à la fois les aspects continus et discrets ont été proposées.

Les *systèmes dynamiques hybrides* caractérisent donc les systèmes faisant intervenir explicitement et simultanément des phénomènes de types continu et évènementiel. Pour leur description, ils nécessitent l'utilisation de fonctions du temps continues par morceaux et de fonctions à valeurs discrètes. Ces différentes fonctions représentent aussi bien l'état interne que les entrées, les sorties et les perturbations.

Les systèmes dynamiques hybrides sont caractérisés par des dynamiques continues propres à chaque état discret composant le système, de telle sorte que l'occurrence d'un événement discret provoque un changement d'état discret e mais également une modification de la dynamique continue du système modélisé par le système différentiel :

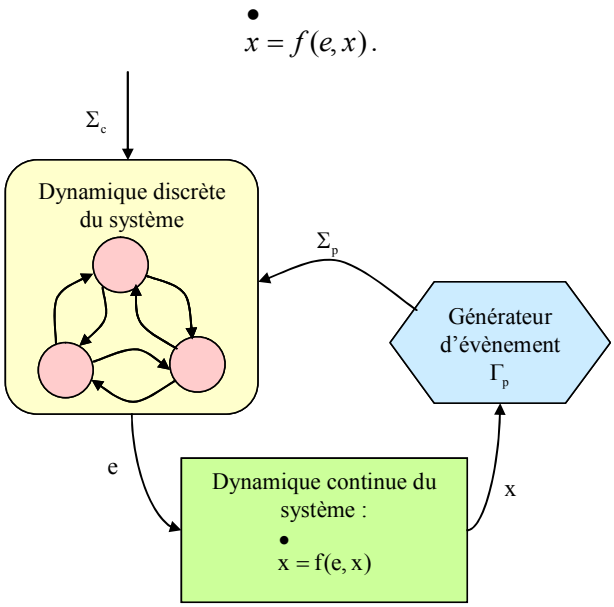


Figure 1.13 La modélisation d'un système dynamique hybride

La Figure 1.13 illustre une modélisation d'un système dynamique hybride d'après [Tittus, 1995]. La détermination de l'état discret e du processus est réalisée en tenant compte des événements de contrôle Σ_c du processus et des événements physiques Σ_p du processus issus du générateur d'événement Γ_p.

Ces vingt dernières années, ces systèmes ont fait l'objet d'une attention particulière. Néanmoins, à l'heure actuelle, les recherches concernant les méthodes et outils formels relatifs à l'analyse du comportement des *SDH* en sont encore à leur début. La simulation reste donc un passage obligé lorsque l'on envisage de les étudier. Par ailleurs, notons qu'au contraire des simulateurs traitant les systèmes à dynamique continue ou les *SED*, peu de logiciels commerciaux traitent explicitement les systèmes dynamiques hybrides en s'appuyant sur un formalisme clairement établi. Il s'agit souvent plutôt d'outils spécifiques à un type de système auquel une extension a été ajoutée pour inclure le comportement manquant. Néanmoins, parmi les outils les plus récents ou les plus couramment rencontrés, nous pouvons citer par exemple *shift* [Deshpande et al., 1998], *Omsim* [Andersson, 1994], *Chi* [Fábián et al., 1998], *BaSiP* [Wöllhaf et al., 1996], *ABACUSS II* (www.yoric.mit.edu/abacuss2/ abacuss2.html.).

1.3.3. Etude des procédés par simulation

L'étude des opérations unitaires offre de nombreux avantages tant au niveau de la conception qu'au niveau de la conduite optimale du procédé. Mais le plus souvent, le procédé à étudier n'offre pas suffisamment d'informations pour comprendre tous les phénomènes mis en jeu. Une étude par simulation devient alors le seul recours. C'est pourquoi les simulateurs sont aujourd'hui des outils incontournables pour la résolution d'un grand nombre de problèmes soulevés au cours de chaque étape du développement, de la conception, de l'exploitation ou de l'amélioration du procédé. Quelle que soit la nature du procédé, la mise en place d'une étude par simulation se déroule généralement en deux temps : la mise en place d'un *modèle mathématique* et sa *résolution*.

1.3.3.1. Etablissement du modèle mathématique

Une représentation mathématique du procédé doit tout d'abord être établie afin de reproduire aussi fidèlement que possible le comportement réel du système. Cette représentation mathématique constitue le modèle du procédé. Ce modèle s'appuie sur les lois fondamentales de la physique et de la chimie telles que la conservation de la masse, le premier et second principe de la thermodynamique, les équilibres entre phases, les lois de transfert… Le modèle qui en résulte conduit alors à un jeu mixte d'équations différentielles ordinaires ou partielles couplées avec des équations algébriques non linéaires. L'évolution rapide de la puissance des calculateurs a permis de valider l'utilisation de ces modèles pour la représentation des opérations unitaires classiques de Génie Chimique.

1.3.3.2. Résolution du modèle mathématique

La deuxième phase de l'étude d'un procédé par simulation consiste à résoudre le système d'équations résultant de la première étape. Compte tenu de la complexité de la plupart des procédés, la résolution analytique de tels systèmes est souvent impossible et le recours à des méthodes numériques s'avère inévitable. L'informatique apparaît alors comme un outil essentiel pour leur mise en œuvre. La résolution du système d'équations équivaut donc à la simulation du procédé : elle permet de suivre l'évolution des variables d'état au cours du temps et d'étudier ainsi le comportement du système représenté.

Deux approches de simulation peuvent être mises en œuvre pour traiter ces systèmes [Mattsson et al., 1993 ; Thévenon, 2000] :

- La **simulation séquentielle modulaire** correspond à une approche où à chaque opération unitaire correspond un sous-programme auquel on fournit des variables d'entrée (courants d'alimentation, conditions opératoires), et où sont calculées les variables de sortie (courants de sortie). Le comportement global du système résulte alors du comportement local de chaque module déterminé séquentiellement selon un ordre bien défini, les entrées des modules étant modifiées par le biais de relations spécifiques entre modules.
- La **simulation globale** constitue l'autre approche. Dans ce cas, le système d'équations complet est obtenu par concaténation des systèmes associés aux modules actifs ; il est ensuite résolu globalement par un solveur puissant. Cette approche permet de lever les difficultés liées aux problèmes de dépendance mutuelle des variables. De plus, la séparation de la phase de modélisation et de la phase de simulation permet une meilleure analyse et manipulation du modèle et induit une simulation plus efficace. Néanmoins, le système complet est généralement de grande taille et son intégration nécessite une méthode de résolution puissante et des moyens de calcul plus importants.

1.3.4. Cas de la simulation des procédés discontinus : La simulation des systèmes dynamiques hybrides

La section 1.2.4.2 a établi que les unités batch fabriquent des produits qui sont par nature continues (fluides) au moyen d'opérations unitaires fonctionnant selon un mode de production discontinu.

Il en résulte que la partie procédurale de la recette comporte une forte composante évènementielle mais dépend aussi de paramètres à valeurs continues (quantités de matière utilisées, conditions opératoires telles que température, pression, concentration, etc.). De ce fait, elle est souvent définie par des événements d'état (seuil de température ou de composition, par exemple) et non par des durées ou des dates d'occurrence fixées a priori. Par conséquent, la simulation des opérations et des transformations physico-chimiques des produits nécessite l'implantation de modèles phénoménologiques.

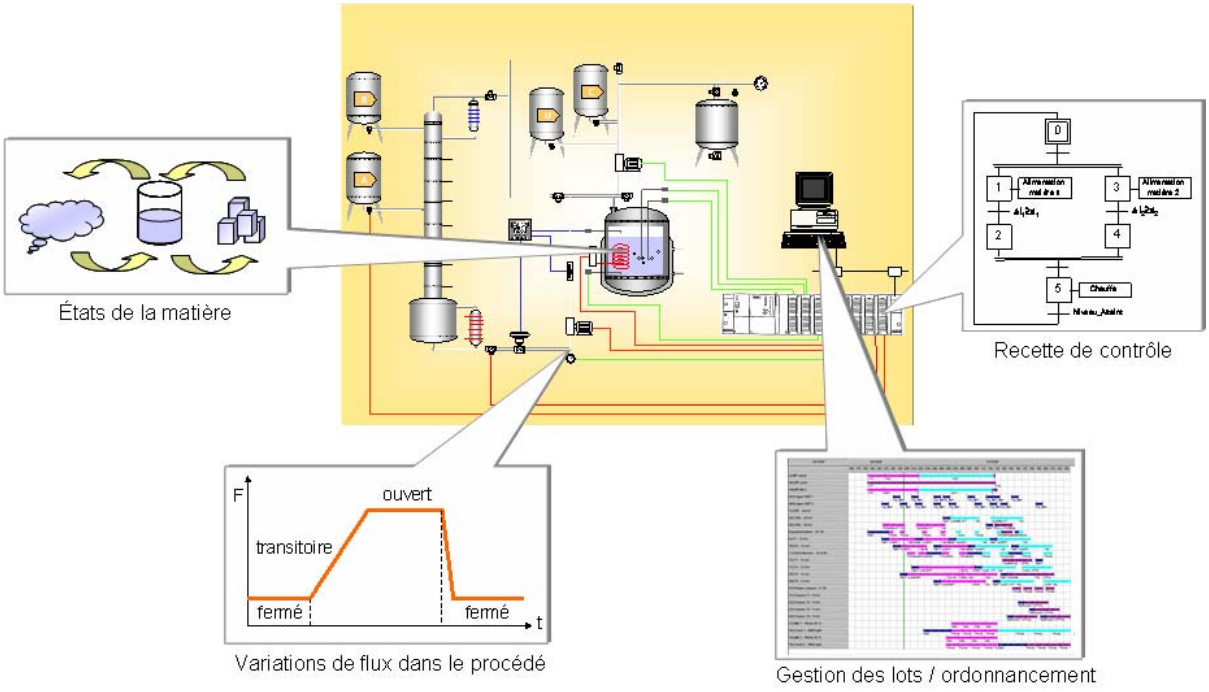


Figure 1.14 Discontinuités intrinsèques à un procédé discontinu

Cependant, la complexité de ces opérations conduit parfois à simplifier ou à négliger certains phénomènes physiques, selon le niveau de finesse ou le temps de réponse exigé pour la simulation (cf. Figure 1.15). Par exemple, si l'évolution de l'état physique de la matière est par nature continue, sa modélisation fait souvent apparaître des discontinuités (dues au changement de phase, notamment). Par ailleurs, comme le montre la Figure 1.14, le mode de production par lots est intrinsèquement un processus comportant des évolutions discrètes ou discontinues.

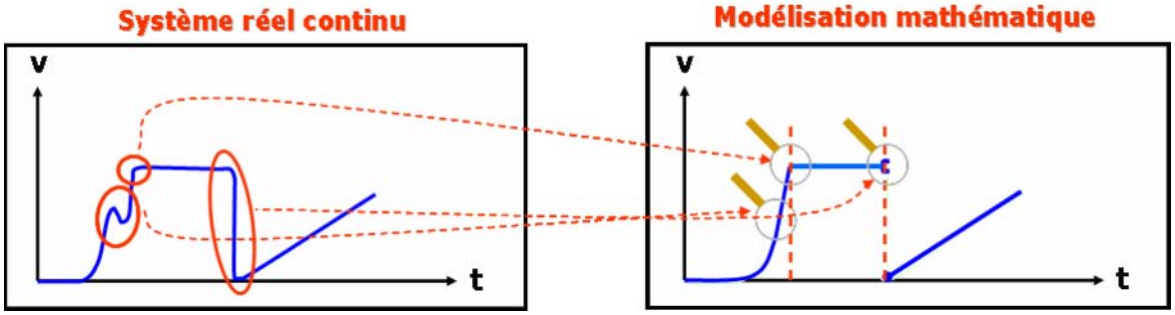


Figure 1.15 Modélisation d'un système dynamique hybride

Il n'est alors plus suffisant de représenter la dynamique continue du procédé, il faut également décrire son comportement discret. Il en résulte que les modèles dynamiques apparaissent ici comme des modèles continus par morceaux ou même discontinus et relèvent alors de la « simulation dynamique hybride » [Zaytoon, 2001]. Une des difficultés majeures liées à la simulation de ces systèmes est la gestion de la structure du modèle. En effet, celle-ci est variable, c'est-à-dire que le nombre d'équations et de variables d'état varie au cours de la simulation. Il est donc nécessaire de disposer d'un noyau de simulation capable de capable de gérer automatiquement ces changements de configuration. Un des moyens pour gérer les séquences légales de commutations entre les différents systèmes d'équations algébro-

différentielles est de mettre en place un modèle à état discret. Dans ce contexte, les outils classiques de simulation dynamique continue ou de simulation à événements discrets d'ateliers s'avèrent mal adaptés à cette classe de problèmes et la mise en œuvre d'un simulateur dynamique permettant de traiter des modèles hybrides paraît donc incontournable.

1.4. LE SIMULATEUR DYNAMIQUE HYBRIDE *PRODHYS*

Les travaux de recherche entrepris depuis plus de quinze ans au sein du Laboratoire de Génie Chimique (*LGC*) ont conduit au développement d'une plate-forme évolutive de modélisation orientée objet et de simulation dynamique hybride de procédés nommée *PrODHyS* (Figure 1.16).

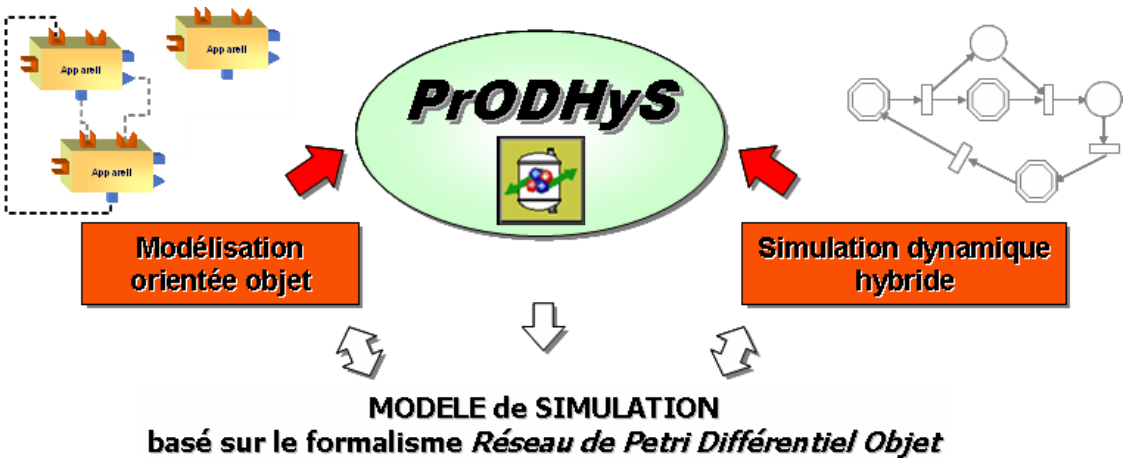


Figure 1.16 Concepts fondamentaux de PrODHyS

Cette section se propose de décrire le contexte dans lequel cette plate-forme a été développée et de présenter brièvement les principaux concepts qui y ont été intégrés. Ces derniers sont repris et décrits de manière plus détaillée dans le chapitre 2.

1.4.1. Modélisation orientée objet des procédés

La section 1.3.3 a établi que toute simulation passe d'abord par une étape de modélisation de l'unité, étape dont la complexité est souvent fortement dépendante de l'outil utilisé. Dans ce cadre, plusieurs approches existent pour construire un modèle de simulation (Figure 1.17).

D'un coté, de nombreux éditeurs proposent des logiciels de *flowsheeting* intégrant dans une même application les outils de modélisation, de simulation et d'analyse. Les progiciels *Prosim Batch* ou *Aspen Dynamics* font partie de cette catégorie de logiciels. Le procédé peut être décrit de manière simple et rapide, soit à travers une *IHM* (*Interface Homme Machine*) graphique et conviviale du type « drag and drop », soit grâce à un langage dédié mais non extensible. Ils offrent à l'utilisateur une panoplie d'appareils prédéfinis qu'il est possible de paramétrer. La contrepartie est souvent un manque de flexibilité lors de la définition de configuration ou de structure particulière d'appareils. En effet, ces applications sont souvent des systèmes propriétaires généralement fermés et monolithiques offrant peu de possibilités d'extension. Or, dans un contexte socio-économique favorable au développement durable, la mise au point de systèmes

innovants (intensification de procédés, bio industries, etc.) est devenue un enjeu majeur du Génie des Procédés. Cette activité d'innovation induit évidemment l'élaboration de nouvelles méthodologies de conception, mais aussi la mise au point de logiciels de modélisation et de simulation adaptés. Notamment, il apparaît nécessaire de disposer d'outils proposant des entités dont la granularité permette la modélisation de prototype de manière flexible et intuitive.

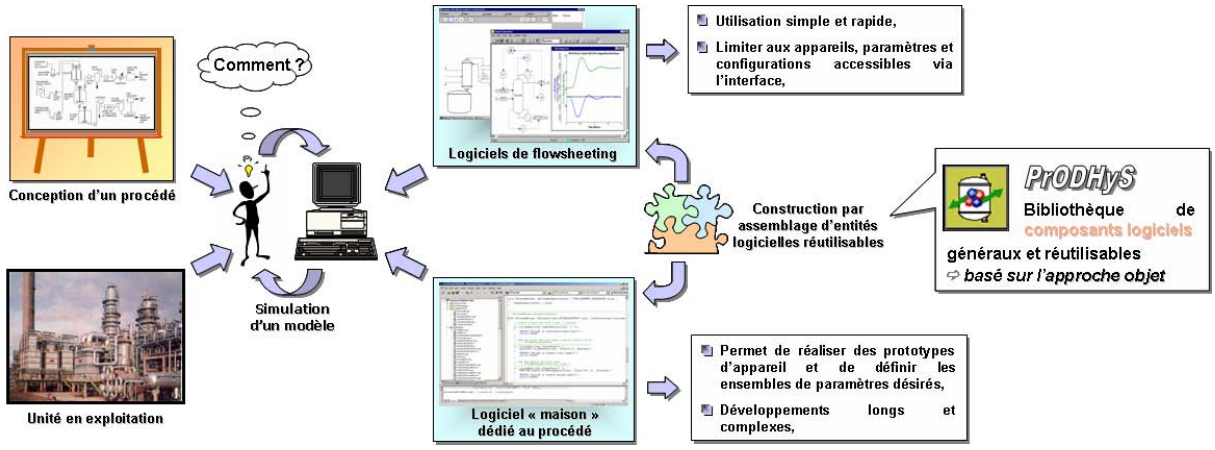


Figure 1.17 Objectifs de PrODHyS

Pour obtenir cette flexibilité, une solution efficace est le développement d'un programme dédié à un procédé particulier pour lequel il est possible de définir tous les paramètres et toutes les configurations souhaités. Dans ce cas, la modélisation consiste à manipuler des variables et des équations et à écrire le système d'équations différentielles associé à chaque état possible du procédé considéré. Quant à la simulation qui réalise l'intégration numérique de ces modèles, elle doit pouvoir gérer la commutation entre ces différents modèles en fonction de l'évolution des variables d'état. Ces développements sont réalisés soit avec des langages de programmation généraux tels que *C++, FORTRAN*, etc., soit avec des langages de modélisation généraux tels que *Omola* (Andersson, 1995), *Ascend* (Allan, 1997), *gProms* (Barton et Pantelides, 1994) ou *Modelica* (Elmqvist *et al.*, 1999). Néanmoins, le développement et la mise au point de tels modèles de simulation peuvent s'avérer relativement longs et complexes. De plus, la maintenance de tels programmes est souvent une tâche délicate. Enfin, le niveau d'abstraction faible des entités manipulées et l'absence de structuration explicite limitent fortement la réutilisation ou l'évolutivité de ces développements.

Afin d'allier la flexibilité d'un langage de modélisation et la rapidité de mise en œuvre d'un outil de *flowsheeting*, une solution consiste à manipuler des composants logiciels élémentaires qui ne représentent pas nécessairement des appareils réels, mais plutôt des entités génériques modélisant une fonction de base particulière. La modélisation qui en résulte se réalise par agrégation ou spécialisation de ces briques élémentaires et offre ainsi à l'utilisateur une approche de construction du modèle de simulation relativement intuitive et très flexible. Cette décomposition du procédé en sous-systèmes plus simples avec une vision systémique s'adapte particulièrement bien à une **approche orientée objet**. La représentation orientée objet des procédés industriels est un thème traité dans la littérature depuis plusieurs années. Ces travaux s'articulent, soit sur la représentation des procédés à un haut niveau [Bourseau, 1993 ; Stephanopoulos et al., 1990], soit sur une représentation structurée mais purement mathématique des modèles [Piela, 1989 ; Nilsson, 1993] (en faisant abstraction d'une représentation moins formelle liée à la structure matérielle des appareils).

C'est précisément dans ce contexte que les bases fondamentales de *PrODHyS* ont été définies [Jourda et al., 1996 ; Sargousse, 1999 ; Moyse, 2000 ; Perret, 2003]. En effet, la conception de cet environnement a cherché à concilier ces deux approches, tout en séparant la représentation de la structure des procédés (c'est-à-dire la constitution de chaque appareil et leur connexion au sein du procédé) de la représentation utile à la simulation (c'est-à-dire les modèles mathématiques formels décrivant leur comportement). Notamment, chaque module de *PrODHyS* repose sur l'identification de composants élémentaires généraux et réutilisables dont l'assemblage permet de construire des modèles de simulation élémentaires, eux-mêmes étant des composants de modèles de simulation plus complexes et spécialisés. C'est pourquoi un des principaux apports de ces travaux réside plus dans la détermination et la conception de ces briques élémentaires initiales que dans l'élaboration d'une bibliothèque riche de modèles. Les objets manipulés sont donc plus fins que les appareils ou la notion d'opération unitaire sans pour autant manipuler explicitement les variables et les équations qui les représentent (même si l'accès à ces éléments est toujours possible). Grâce aux mécanismes de composition, d'héritage et de généricité, il est possible de générer des entités facilement réutilisables pour des développements futurs et conduit à long terme à des temps et coûts de développement réduits. Cette approche s'avère très efficace pour les sociétés d'ingénierie dont le travail est de créer des simulateurs dédiés à des unités spécifiques. En effet, l'utilisation d'outils offrant une réutilisabilité maximale de l'existant est très intéressante car elle évite aux développeurs de recoder entièrement les modèles.

1.4.2. Formalisme de modélisation et de simulation hybride

Comme la section 1.3.4 l'a montrée, les procédés batch ou les procédés continus en régime transitoire relèvent de la catégorie des systèmes dynamiques hybrides *(SDH)*. Dans ce contexte, le modèle doit décrire de manière explicite et rigoureuse autant les évolutions continues que discrètes du système. Pour cela, il faut disposer d'un formalisme combinant simultanément dans une même structure des éléments sémantiques décrivant les composantes continues et les composantes événementielles du modèle.

Dans ce cadre, de nombreux formalismes hybrides ont été définis ou obtenus par extension de formalismes discrets ou continus existants. Nous pouvons citer les plus classiques : les automates hybrides [Alur et al., 1995], les réseaux de Petri prédicats-transitions différentiels [Champagnat et al., 1998], les réseaux de Petri hybrides [Le Bail et al., 1991], les Statecharts hybrides [Kesten et Pnueli, 1992].

Dans *PrODHyS*, le modèle de simulation d'un procédé est décrit au moyen du formalisme *Réseaux de Petri Différentiel Objet* (*RdPDO)*. Les évolutions continues sont décrites par des systèmes d'équations différentielles algébriques (*EDA*) et les changements d'état discrets du système sont représentés par les séquences de places et de transitions de la structure *Réseau de Petri*. La synchronisation des composantes continues et discrètes est assurée grâce à la notion d'événements. Le comportement dynamique de chaque objet de *PrODHyS* est alors décrit au moyen d'un *RdPDO* intégré au sein de celui-ci comme un attribut. Ce *RdPDO* évolue soit de manière autonome, soit en interaction avec d'autres objets du système tels que la recette ou un appareil actif.

1.4.3. Couplage du simulateur *PrODHyS* avec un optimiseur pour l'analyse de performance des procédés discontinus

PrODHyS, comme de nombreux autres outils de simulation disponibles, n'est pas complètement adapté pour réaliser les analyses décrites dans la section 1.3.1. En effet, dans le cas d'un atelier discontinu, ce type d'étude nécessite :

- la simulation de tous les modes de fonctionnement possibles du système (y compris les régimes transitoires),
- la simulation du procédé complet en vue de satisfaire une liste d'ordres de fabrication sur un horizon d'étude fixé (cet horizon étant généralement un paramètre fixé par la *G.P.A.O.*).

Si le premier point est actuellement complètement rempli par *PrODHyS*, il n'en est pas de même pour le second. En effet, la séquence de passage des lots de matière dans les différents équipements de l'unité peut avoir un impact important sur les performances du système. Pour étudier cet aspect, il faut que le simulateur puisse dérouler un scénario correspondant à une recette et une affectation de ressources définies. Ces scénarios correspondent en fait à la réalisation d'un plan de production préalablement établi dans lesquels sont définis, entre autres, le nombre, le volume et la date de lancement de chaque lot ainsi que l'appareil utilisé pour réaliser l'opération. La difficulté est que l'évolution myope de la simulation ne permet pas de calculer de manière explicite ces grandeurs et peut conduire, dans certains cas, à l'avortement du calcul pour cause de non respect de certaines contraintes (capacités ou délais non respectés, par exemple). Pour cette raison, il apparaît que pour évaluer correctement la performance d'une unité discontinue, le simulateur dynamique doit être accompagnée d'un module d'ordonnancement chargé d'établir un (ou des) scénario de production faisable satisfaisant au mieux un certain critère.

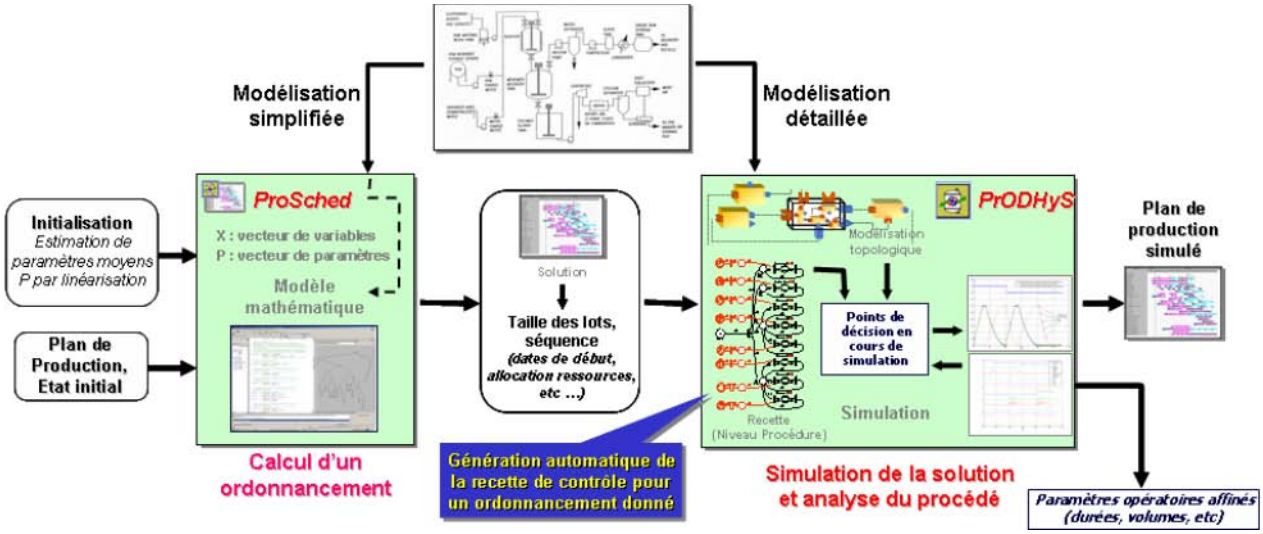


Figure 1.18 Schéma de principe de l'outil développé

La conception et la mise en place de ce module d'ordonnancement ainsi que le couplage avec la simulation constituent le cœur des travaux de recherche présentés ici. La Figure 1.18 montre le schéma de principe de l'outil développé. Sur la base d'une modélisation simplifiée du procédé et en tenant compte de son état initial, un ordonnancement est calculé afin de satisfaire un plan de production donné. Les paramètres caractérisant chaque tâche de production (date de début, ressource utilisée, taille du batch) ainsi que les conditions opératoires de l'opération unitaire associée sont transmis au simulateur dynamique *PrODHyS*. En exploitant ces informations, le *RdPDO* représentant la recette de contrôle est généré automatiquement en assemblant des composants *opérations* prédéfinis et paramétrables. La simulation est alors exécutée jusqu'à la réalisation complète du plan de production. Si la simulation s'achève, cela indique que toutes les contraintes de capacité et de délai sont satisfaites. Le plan de production validé, l'analyse des phénomènes physico-chimiques peut être effectuée et les différents indicateurs de performance sont calculés. Selon le cas, certains paramètres du modèle (simplifié et/ou détaillé) peuvent être modifiés et la

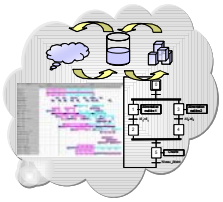
procédure de calcul relancée jusqu'à ce que l'utilisateur estime les résultats satisfaisants. Si la simulation n'aboutit pas, certaines contraintes sont violées et l'utilisateur doit alors analyser les résultats de simulation afin d'entreprendre une action corrective.

1.5. CADRE DE L'ÉTUDE

Les sections précédentes ont permis de dresser un bref panorama des éléments et concepts classiques rencontrés en conduite et simulation des systèmes industriels. L'objectif de cette dernière section est de positionner notre étude dans ce cadre.

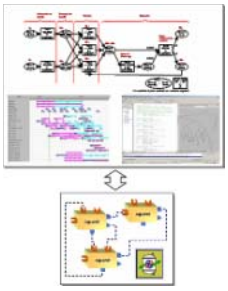
Comme l'a montrée la section 1.2, ce travail s'intéresse principalement au domaine du Génie des Procédés.

En particulier, cette étude se focalise sur la modélisation et la simulation dynamique des procédés discontinus. La section 1.3 s'est attachée à montrer l'intérêt de la simulation pour la conception et surtout, l'analyse de ce type de procédés, notamment pour évaluer ces performances.



Elle a, par ailleurs, mis en évidence le fonctionnement multi-modal caractéristique de ces systèmes industriels ; il en ressort que les modèles proposés pour les représenter sont dits « hybrides ». Ils s'avèrent en effet bien adaptés pour la représentation des procédés batchs, des phases transitoires (comme les démarrages ou les arrêts d'unités) ou tout simplement des phénomènes hybrides intrinsèques au système modélisé comme les changements de phases.

La section 1.4 a mis en avant les principales propriétés du simulateur dynamique hybride *PrODHyS* développé au sein du LGC. Celui-ci se distingue en termes de modularité *(approche objet),* de causalité *(simulation globale)* et de représentation rigoureuse et explicite des composantes continues et discrètes (formalisme *Réseau de Petri Différentiel à Objets (RDPDO)*).



Néanmoins, afin de pouvoir dérouler une simulation sans difficulté, il est nécessaire de fournir un plan de production définissant les tâches à réaliser à chaque instant et les équipements affectés pour les réaliser. Dans ce but, le simulateur doit être associé à un module d'ordonnancement qui calcule automatiquement un scénario sur la base d'une recette et d'une liste d'ordres de fabrication (*OF*). Pour cela, un modèle générique d'ordonnancement doit être établi puis couplé au simulateur *PrODHyS*. Par ailleurs, un formalisme pour représenter la recette doit être étendu afin de décrire de manière systématique les scénarios.

Le travail exposé dans la suite de ce manuscrit s'articule autour de deux parties :

- La première partie est consacrée à un état des lieux de l'existant. Elle est divisée en trois chapitres. Le premier (celui-ci) offre quelques définitions et concepts et fixe l'objectif de ces travaux. Le chapitre 2 dresse un état de la plate-forme *PrODHyS*, telle qu'elle existait avant le démarrage de ces travaux. Enfin, le chapitre 3 commence par réaliser un panorama des différentes techniques et approches trouvées dans la littérature pour résoudre les problèmes d'ordonnancement ; il se poursuit en mettant en évidence les caractéristiques et les spécificités liées à l'ordonnancement des procédés batch ; ce chapitre se termine alors en indiquant la technique d'optimisation retenue dans ces travaux et qui est combinée avec le simulateur dynamique *PrODHyS*.

.

- La deuxième partie est consacrée à la description de l'outil développé et à son exploitation sur quelques exemples typiques de procédé. Au sein de cette partie, le chapitre 4 décrit le modèle d'ordonnancement mis en œuvre dans ces travaux pour fournir le scénario de production suivi par le simulateur ; une extension au formalisme *RTN* est introduite afin de représenter graphiquement et mathématiquement l'aspect multi-modal de certaines opérations (démarrage, arrêt, préparation, etc.) ; ce chapitre se termine avec quelques résultats numériques montrant les qualités et faiblesses de ces modèles. Le chapitre 5 met l'accent sur l'intégration du module *ProSched* au sein de la plate-forme *PrODHyS* ; ce module regroupe les classes nécessaires pour mettre en œuvre la conduite de la simulation dynamique au moyen des informations établies par le modèle d'optimisation. Notamment, la notion de macro-place (pour représenter les différents niveaux hiérarchiques de la recette) et de jeton « lot » (pour identifier chaque batch) sont introduits. Enfin, afin d'illustrer ces potentialités, le chapitre 6 est consacré à l'exploitation de l'outil en présentant deux exemples complets d'application du couplage.

CHAPITRE 2

CHAPITRE 2

LA PLATEFORME DE SIMULATION *PRODHYS*

Résumé

L'unification des travaux de recherche en modélisation et simulation de procédés menés depuis plusieurs années au sein du Laboratoire de Génie Chimique a permis de développer une plateforme de simulation dynamique hybride nommée *PrODHyS (**Pr**ocess **O**bject **D**ynamic **Hy**brid **S**imulator).*

L'objectif de ce chapitre est de décrire les principes fondamentaux de *PrODHyS* notamment le formalisme *Réseau de Petri Différentiel Objet* (*RdPDO*) et la modélisation objet des procédés sur lesquels s'appuie cette plateforme de simulation. L'utilisation de ces concepts est illustrée à travers la mise en place d'un exemple didactique.

Ce chapitre est consacré à la description des principes fondamentaux sur lesquels a été construit l'environnement de modélisation et de simulation de systèmes dynamiques hybrides *PrODHyS*. Après avoir brièvement décrit l'architecture logicielle de *PrODHyS*, nous présentons dans un premier temps le formalisme *Réseau de Petri Differentiel Objet* mis en œuvre dans *PrODHyS* pour modéliser le comportement dynamique de chaque objet. Ensuite, les principales étapes du processus de construction d'un modèle de simulation sont exposées. Enfin, les potentialités de l'environnement sont illustrées à travers la modélisation et la simulation d'un exemple didactique.

2.1. ARCHITECTURE LOGICIELLE DE *PRODHYS*

La plate-forme *PrODHyS (**Pr**ocess **O**bject **D**ynamic **Hy**brid **S**imulator)* est un environnement de modélisation et de simulation dynamique hybride de procédés s'appuyant sur les technologies objet. Il fournit un ensemble de composants généraux, extensibles et réutilisables afin de construire des objets plus spécifiques.

Elle se présente sous la forme d'une bibliothèque de classes destinées à être dérivées en exploitant les mécanismes objets *(polymorphisme, composition, héritage, généricité)*. Son développement s'articule autour d'un processus de conception et de codage unifié, basé sur le formalisme *UML* et le langage objet *C++*.

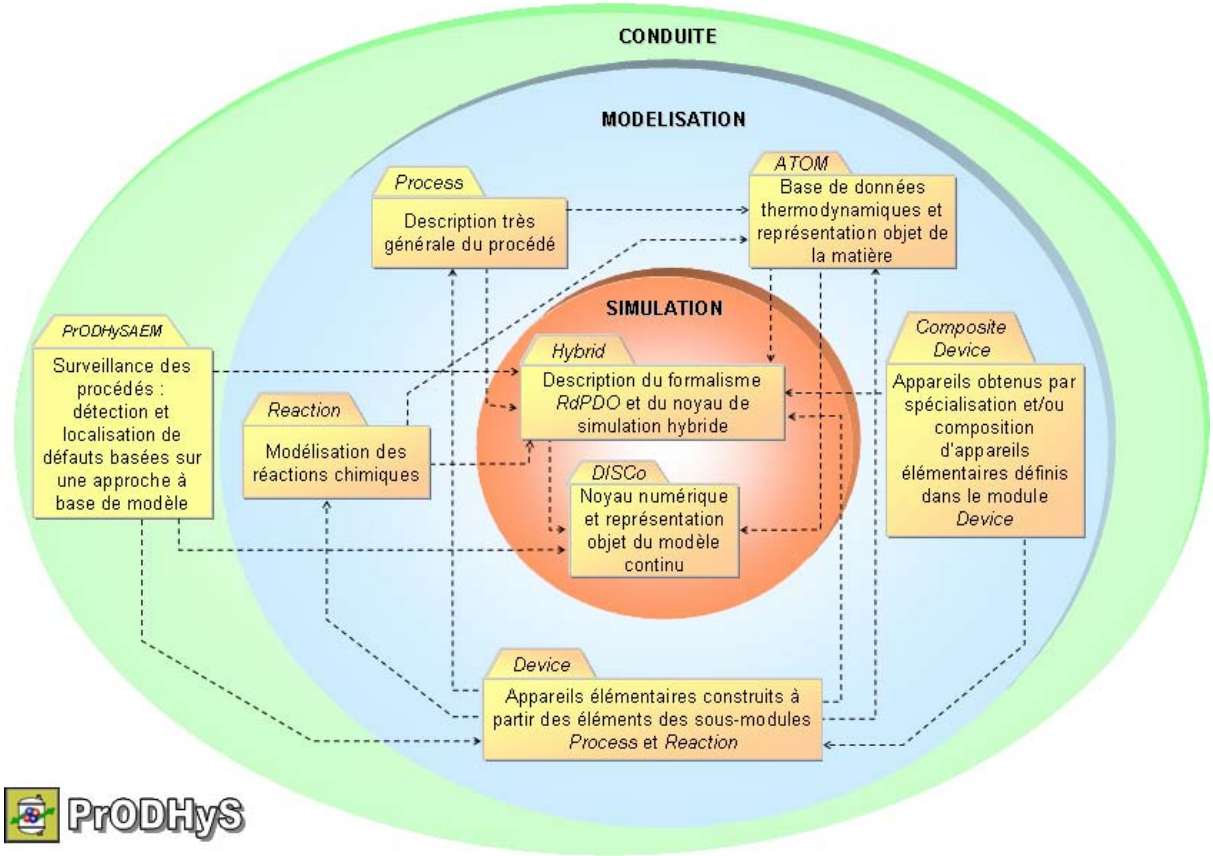


Figure 2.1 Architecture logicielle de PrODHyS

Exploitant la longue expérience du département *Procédés et Systèmes Industriels* (*PSI*) du *LGC* en modélisation et simulation de procédés, cette plate-forme résulte de l'unification de plusieurs travaux de thèse ([Jourda, 1996], [Sargousse, 1999], [Moyse, 2000], [Perret, 2003], [Olivier-Maget, 2007]), de travaux de recherche ([Hétreux et al., 2003]) et de contrats d'étude (*projet OPERA*, *projet SiMoBe*, etc). Actuellement, cette bibliothèque regroupe plus de mille classes réparties en trois couches fonctionnelles indépendantes. Comme le montre la Figure 2.1, chaque couche est décomposée en plusieurs modules.

La couche interne de *PrODHyS* correspond au noyau de *simulation*. Elle fournit les objets de base utiles à la simulation de tout système dynamique, quel que soit le domaine d'application. Elle est composée :

- du module *DISCo* qui constitue le noyau numérique du système ; il offre un ensemble d'intégrateurs d'équations différentielles algébriques (*EDA)* et de solveurs d'équations algébriques non linéaires (*EANL*) ainsi que les entités de base pour la représentation objet des modèles mathématiques continus [Le Lann J.M., 1999].
- du module *Hybrid* qui contient l'ensemble des classes nécessaires à la description du formalisme *Réseau de Petri Différentiel Objet* ainsi que le gestionnaire de simulation qui constitue le cœur du simulateur. Les principes de fonctionnement de ce gestionnaire sont présentés brièvement dans la suite.

Les aspects *modélisation* sont gérés au niveau de la couche intermédiaire. Elle rassemble les classes utilisées pour la représentation de la structure des procédés et des phénomènes physico-chimiques qui s'y déroulent. Cette couche comprend :

- le module *ATOM* qui fournit les classes utiles à la représentation objet de la matière. Il représente la base de données thermodynamiques du système et permet le calcul de propriétés thermodynamiques via le composant *ProPhy32* développé par la société *ProSim SA*. Notons que la connexion au nouveau package *Simulis* proposé par *ProSim SA* est en cours d'implémentation.
- le module *Process* qui regroupe un ensemble de classes, souvent abstraites, correspondant à une description très générale du procédé. En fait, cette classe contient la plupart des classes « mère » de la bibliothèque.
- le sous-module *Device* qui répertorie les appareils élémentaires « concrets » du procédé, construits à partir des éléments du précédent module.
- le module *CompositeDevice* qui rassemble les appareils obtenus par spécialisation et/ou composition d'appareils élémentaires définis dans le module *Device*.
- le sous-module *Reaction* qui permet la modélisation des réactions chimiques,

La couche supérieure, nommée *conduite*, correspond aux applications de la plate-forme. Elle contient toutes classes utiles au développement de systèmes de conduite d'un procédé, ainsi que l'ensemble des tests unitaires utilisés pour la validation de chaque composant (non indiqué sur la Figure 2.1). Actuellement, cette couche comprend :

- le module *PrODHySAEM* récemment développé pour la surveillance des procédés. Il contient les éléments fondamentaux qui interviennent dans la mise en œuvre de la méthode de détection et de diagnostic de défaut *SimAEM* (filtrage, génération des résidus, détection, élaboration de la table de signature, diagnostic, etc).

Durant la phase de développement de cet outil (qui se poursuit encore aujourd'hui), l'environnement *PrODHyS WinTester* a été mis en place à l'usage des développeurs de la plateforme (Figure 2.2). Il permet d'une part, de centraliser et de faciliter le test unitaire de chaque objet de *PrODHyS*. D'autre part, il peut jouer le rôle de pilote de simulation lors de la mise au point d'un simulateur dédié en permettant une introspection réelle à l'intérieur de celui-ci.

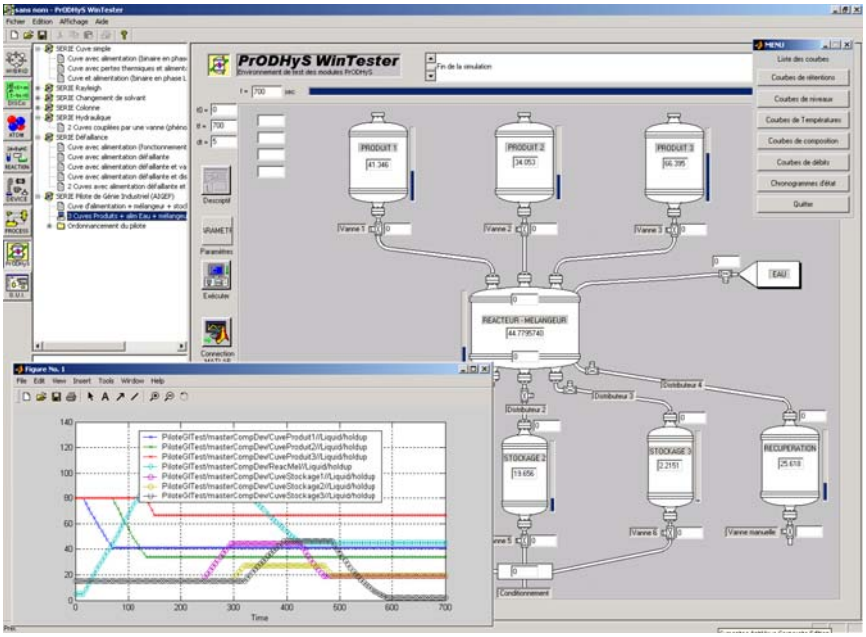


Figure 2.2 Environnement PrODHyS WinTester

Cette architecture en trois couches fonctionnelles rend possible le développement de modèles de simulation dédiés à des domaines d'ingénierie variés (électronique, productique, etc.) simplement en développant la couche *modélisation* adaptée. Enfin, elle permet l'accès à cette plate-forme à différents types d'utilisateurs :

- un *développeur* d'applications orientées objet utilisera directement la couche *simulation* pour créer les composants de base dans un domaine particulier (composants électroniques, pièces mécaniques, éléments de procédés, etc.). Dans notre cas, cela correspond au développement des modules *Process*, *Reaction* et *Device* dédiés au domaine des procédés.
- un *modélisateur* sera plutôt intéressé par la mise en place des systèmes complexes à partir des composants de base en faisant abstraction du comportement interne de ces derniers. Dans notre cas, cela correspond au développement du module *CompositeDevice* qui contient les appareils complexes spécifiques.
- Enfin, un simple *utilisateur* sera désireux de réaliser une étude sans effectuer de développement informatique complexe et en utilisant directement les objets déjà créés à travers une interface graphique de manière analogue à une application de *flowsheeting*. Cette dernière couche logicielle est aussi en cours de développement dans un module nommé *PrODHySGUI*.

2.2. LE FORMALISME RÉSEAU DE PETRI DIFFÉRENTIEL OBJET

Cette section ne vise qu'à rappeler les principes essentiels des *RdPDO* et seuls sont abordés les éléments nécessaires pour comprendre la manière dont les modèles de simulation sont spécifiés au moyen de ce formalisme. Néanmoins, une définition formelle est fournie dans l'annexe B et une

description exhaustive de ce formalisme et son implémentation dans *PrODHyS* peut être trouvée dans [Perret J, 2003], à l'origine de ces travaux.

2.2.1. Objectifs du formalisme *RdPDO*

Un des premiers objectifs a été de mettre en œuvre un formalisme suffisamment général pour permettre de représenter la plupart des systèmes dynamiques hybrides. Pour pouvoir utiliser ce formalisme dans des domaines variés, il se devait de décrire de façon rigoureuse et précise aussi bien les composantes continues que discrètes des systèmes à simuler, en l'occurrence ici les procédés.

Une façon rigoureuse de représenter les aspects continus est d'utiliser des ***systèmes d'équations différentielles algébriques***. Compte tenu de la complexité des phénomènes physico-chimiques mis en jeu, ils ont donc été adoptés pour simuler l'évolution continue des variables d'état. Notons que cette solution implique la présence d'un solveur d'équations différentielles algébriques possédant de nombreuses fonctionnalités (réduction d'index, recherche de conditions initiales cohérentes, détection d'événements, etc.), ce qui est effectivement le cas du module *DISCo* intégré à la plate-forme. La gestion de la partie continue est donc bien établie.

En outre, de nombreux phénomènes discrets intrinsèques aux systèmes eux-mêmes (changements de phase, par exemple) ou résultant du contrôle, de la commande ou de la supervision, ne peuvent être traités de façon simpliste. Par conséquent, un modèle discret doit être mis en place afin de permettre une description aussi fidèle que possible de la composante évènementielle du système. Un modèle discret basé sur les ***réseaux de Petri*** semble bien approprié puisqu'il permet de représenter efficacement le séquencement, et les mécanismes de synchronisation, de parallélisme, d'affectation des ressources ou d'exclusion mutuelle.

Enfin, les systèmes considérés nécessitent de manipuler et de structurer d'abondantes quantités d'informations (éléments constitutifs et géométriques d'une opération unitaire, spécification des conditions opératoires, propriétés de la matière, etc.). Pour cela, l'utilisation de l'***approche objet*** est apparue tout à fait appropriée pour obtenir une description de haut niveau qui réduit la taille du modèle et en améliore la lisibilité.

L'intégration de ces trois concepts a conduit au développement du formalisme nommé ***Réseau de Petri Différentiel Objet*** [Hétreux et al, 2003].

2.2.2. Eléments sémantiques des *RdPDO*

Cette section se propose d'abord de rappeler la définition de chaque élément sémantique introduit dans les *Réseau de Petri Différentiel Objet* puis d'illustrer les deux manières dont *objets* et *réseaux de Petri* interagissent.

2.2.2.1. Définitions des éléments

Formellement, un *RdPDO* initialement marqué est le *n-uplet* :

$$N = \langle C, V, P, T, A, A_{AC}, Pre, Post, X, F, A_T, M_0 \rangle$$

L'ensemble des éléments sémantiques de cette catégorie de réseau de Petri est représenté sur la Figure 2.3, avec le nom de la classe associée dans la bibliothèque *PrODHyS*.

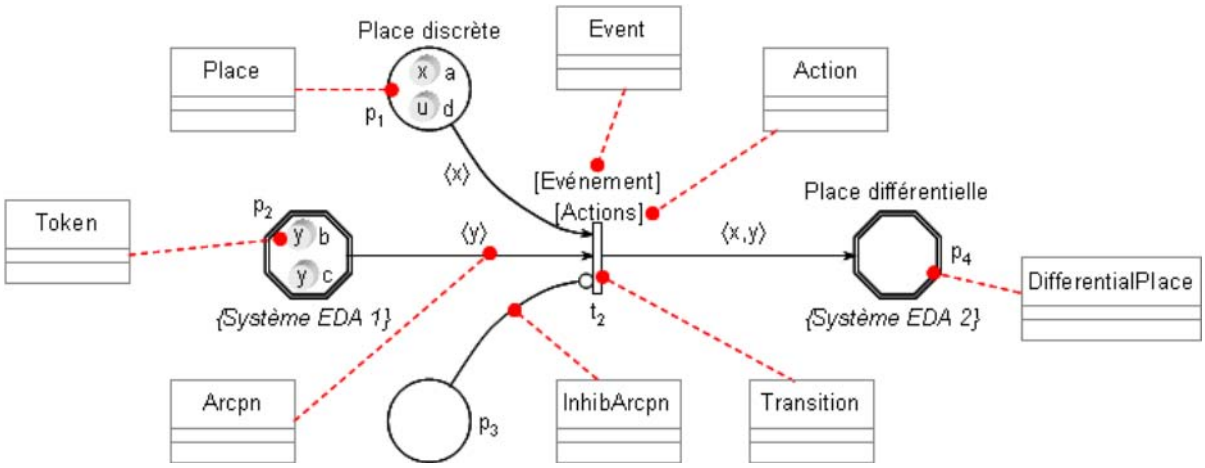


Figure 2.3 Sémantique du formalisme RdPDO

Définition 2.1 : Classe

Une *classe* c_k est associée à chaque objet composant le système et est définie dans l'ensemble $C = \{c_1, c_2, \ldots, c_k\}$

Le but de l'intégration des concepts objet dans le modèle *RdPDO* est de faciliter la décomposition et la structuration du système à simuler, conduisant ainsi à une meilleure gestion de sa complexité.

Définition 2.2 : Variable formelle

Une *variable formelle* est définie dans l'ensemble $V = \{v_1, v_2, \ldots, v_k\}$ et est typée par des éléments de l'ensemble *C*. Une variable formelle $v_i = <c_i>$ spécifie le type des jetons qui peuvent lui être substitués est c_i.

Soit une classe c_i ; une variable formelle de type c_i est notée $v_i = \langle c_i \rangle$; un *n-uplet* de variables formelles est noté $\langle c_1, c_2, \ldots, c_n \rangle$. Lorsqu'un *n-uplet* contient n_i variables formelles de type c_i, elles sont notées $\langle c_i^{(k)} \rangle$ pour k allant de *1* à n_i.

Les *RdPDO* sont constitués d'un ensemble fini de places $P = \{p_1, p_2, \ldots, p_k\}$ représentant les différents états du système. Néanmoins, deux types de places doivent être différenciés (Figure *2.7*):

- des places dites *discrètes* représentées par le sous-ensemble P_D
- des places dites *différentielles* représentées par le sous-ensemble P_H

Par conséquent, l'ensemble P est tel que : $P = P_H \cup P_D$

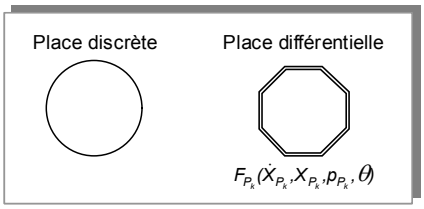


Figure 2.4 Représentation des places

Définition 2.3 : Place discrète

Une *place discrète* p_k représente un état dans lequel il n'y a pas d'évolution au cours du temps des variables continues du système.

Représentée par un simple cercle, une *place discrète* permet généralement de modéliser un contrôle, une commande, la disponibilité d'une ressource, etc.

Définition 2.4 : Place différentielle

Une *place différentielle* p_k représente un état dans lequel il y a une évolution continue au cours du temps des variables d'état du système. Cette évolution est régie par le système d'équations différentielles algébriques $F_{P_k}(\dot{X}_{P_k}, X_{P_k}, p_{P_k}, \theta) = 0$ associé à la place p_k.

Représentée par deux octogones concentriques (Figure 2.4), une *place différentielle* permet donc de modéliser un comportement continu (d'une opération, de la matière, etc.) au moyen d'un système d'équations différentielles algébriques. Ce système est composé d'équations liant des sous-ensembles de variables d'état du modèle continu global. Il s'écrit comme suit :

$$F_{P_k}(\dot{X}_{P_k}, X_{P_k}, p_{P_k}, \theta) = \begin{pmatrix} f_{P_k 1}(\dot{X}_{P_k}, X_{P_k}, p_{P_k}, \theta) \\ \vdots \\ f_{P_k n_f}(\dot{X}_{P_k}, X_{P_k}, p_{P_k}, \theta) \end{pmatrix} = 0$$

avec :

$f_{P_k i}$: fonction i associée à la place P_k ;

n_f : nombre de fonctions associées à la place P_k ;

X_{p_k} : ensemble des variables d'état associées à la place P_k tel que : $X_{P_k} = X_{P_k}^G \cup X_{P_k}^L \cup X_{P_k}^F$ où :

$X_{P_k}^G$: ensemble des variables globales au réseau de Petri possédant la place différentielle P_k ;

$X_{P_k}^L$: ensemble des variables locales à la place différentielle P_k ;

$X_{P_k}^F$: ensemble des variables définies comme attributs des classes associées aux variables formelles, ces dernières appartenant à l'ensemble $Post(P_k, \bullet)$.

p_{p_k} : ensemble des paramètres associés à la place P_k tel que : $p_{P_k} = p_{P_k}^G \cup p_{P_k}^F$ où :

$p_{P_k}^G$: ensemble des paramètres globaux au réseau de Petri possédant la place différentielle P_k ;

$p_{P_k}^F$: ensemble des paramètres définis comme attributs des classes associées aux variables formelles, ces dernières appartenant à l'ensemble $Post(P_k, \bullet)$.

Définition 2.5 : Transition

Une *transition* t_k est définie dans l'ensemble fini $T = \{t_1, t_2, \ldots, t_k\}$ et permet le passage d'un état à un autre. A chaque transition sont associés un *événement* et un ensemble d'*actions*.

Définition 2.6 : Jeton

Un *jeton* j_k définit le marquage d'une place. Une place marquée indique que l'état correspondant est actif. Chaque jeton correspond à une instance de classe d'objet.

S'agissant d'une instance de classe, chaque jeton est constitué d'attributs *(paramètres de l'objet, variables continues, réseau de Petri, etc.)* et de méthodes. Ce mécanisme, avec individualisation et typage du jeton, permet de rendre le réseau plus compact, sans perte d'information.

Définition 2.7 : Arc

Un *arc* a_k est défini dans l'ensemble fini $A = \{a_1, a_2, \ldots, a_k\}$ et permet d'établir une relation d'accès entre une place et une transition. Chaque arc porte une ou plusieurs variables formelles permettant de spécifier les classes de jetons autorisés à transiter sur cet arc.

L'*arc* est donc le ligand entre la place et la transition. Afin d'assurer la cohérence du modèle, les arcs d'entrée et de sortie sont explicitement typés. Ainsi, tout jeton autorisé à transiter par un arc se substitue à la variable formelle de même type. S'agissant de classes, les propriétés relatives à l'héritage s'appliquent. En particulier, toute classe de jeton héritière du type inscrit pourra aussi transiter par cet arc.

Les *RdPDO* peuvent aussi contenir des arcs particuliers appelés *arcs inhibiteurs*. Au niveau de la représentation, ils se différencient par un cercle situé à l'une de ses extrémités (celle reliée à la transition).

Définition 2.8 : Arc inhibiteur

Un *arc inhibiteur* est un arc qui ne consomme pas et qui permet de tester l'absence de jeton sur la place située en amont.

L'arc inhibiteur est, par exemple, très bien adapté pour tester l'absence de stock dans une unité de production. Dans le cadre du formalisme *RdPDO*, l'arc inhibiteur permet de tester l'absence de jeton du (ou des) type(s) indiqué(s) sur l'arc.

Définition 2.9 : Evènement

A toute transition peut être associé un *évènement* unique, constitué d'une conjonction de *conditions*. L'évènement est occurrent lorsque toutes les conditions sont satisfaites.

L'*évènement* conditionne le tir d'une transition. Dans le modèle *RdPDO*, deux types de conditions peuvent être associés à une transition :

- des conditions dites *discrètes*,
- des conditions dites *continues*, encore appelées *fonctions de sensibilisation*

Ces conditions sont des expressions construites avec les méthodes et attributs portés :

- par les jetons sensibilisant la transition,
- par l'objet dont le comportement est décrit par ce réseau de Petri.

Définition 2.10 : Condition discrète

Une *condition discrète* est une expression dont le résultat est un booléen.

Définition 2.11 : Condition continue ou fonction de sensibilisation

Une *condition continue* ou *fonction de sensibilisation* e_i associée à la transition t_i est définie comme étant la première solution en θ de :

$$e_i\left(\dot{X}_{in_i}, X_{in_i}, p_{in_i}, \theta\right) = 0$$

Une fonction de sensibilisation permet de vérifier une condition portant soit sur les variables d'état continues, soit sur la variable indépendante, le temps. Cette fonction appartient donc au domaine du « continu ».

Lors du franchissement d'une transition, un certain nombre d'*actions* peut être exécuté. Ces actions consistent :

- d'une part à initialiser les variables d'état continues et leurs dérivées. Dans ce cas, on les qualifie de *fonctions de jonctions*,
- d'autre part, à exécuter des méthodes offertes par les jetons franchissant la transition ou appartenant à l'objet incluant ce réseau ; elles permettent de modifier l'état de l'objet référencé par le jeton ou l'état de l'objet qui intègre ce réseau.

Définition 2.12 : Fonction de jonction

La *fonction de jonction* j_i associée à la transition t_i calcule à la date θ (date de franchissement de t_i) les valeurs des variables ainsi que de leurs dérivées, en accord avec l'état suivant, comme suit :

$$j_i : \begin{array}{l} X_{out_i}(\theta^+) = j_{iX}(\dot{X}_{in_i}, X_{in_i}, p_{in_i}, \theta^-) \\ \dot{X}_{out_i}(\theta^+) = j_{i\dot{X}}(\dot{X}_{in_i}, X_{in_i}, p_{in_i}, \theta^-) \end{array}$$

Les valeurs des variables et de leurs dérivées, juste après θ (à θ^+), sont calculées à partir des valeurs des variables et de leur dérivées juste avant θ (à θ^-).

2.2.2.2. Interactions entre *objets* et *réseaux de Petri*

Dans le formalisme *RdPDO*, *Objet* et *Réseau de Petri* peuvent interagir de deux manières :

- **introduction des objets dans les réseaux de Petri**

  La première manière de faire interagir *objet* et *Réseau de Petri* est de considérer les jetons comme des objets qui se déplacent sur le réseau de Petri. La philosophie sous-jacente consiste à

modéliser les états d'un ensemble d'entités identiques par un réseau de Petri unique dans lequel chaque jeton représente une entité particulière. Ces jetons doivent donc être individualisés et porteurs d'informations. La structure des entités modélisées est définie au moyen d'une classe caractérisée par un ensemble d'attributs (incluant les variables d'état) et un ensemble de méthodes qui traitent ces données. Chaque jeton correspond alors à une instance de cette classe.

- **introduction des réseaux de Petri dans les objets**

  Une autre façon de faire interagir *objet* et *Réseau de Petri* est d'utiliser les réseaux de Petri pour décrire le comportement dynamique interne des objets. Considérons, par exemple, un objet caractérisé par un ensemble d'attributs et de méthodes. Le marquage courant du réseau indique alors l'état interne de l'objet, le franchissement des transitions se traduit par l'exécution d'une ou plusieurs méthodes de l'objet et la structure globale du réseau spécifie les séquences légales d'exécution des méthodes.

Les deux approches énoncées ci-dessus peuvent évidemment se combiner et ainsi, introduire plusieurs niveaux d'intégration des *objets* dans les *RdP* et des *RdP* dans les *objets*. En effet, un objet intègre un réseau de Petri sur lequel transitent des jetons. Certains de ces jetons peuvent être un autre objet dont le comportement est décrit aussi par un réseau de Petri et ainsi de suite (cf. Figure 2.5).

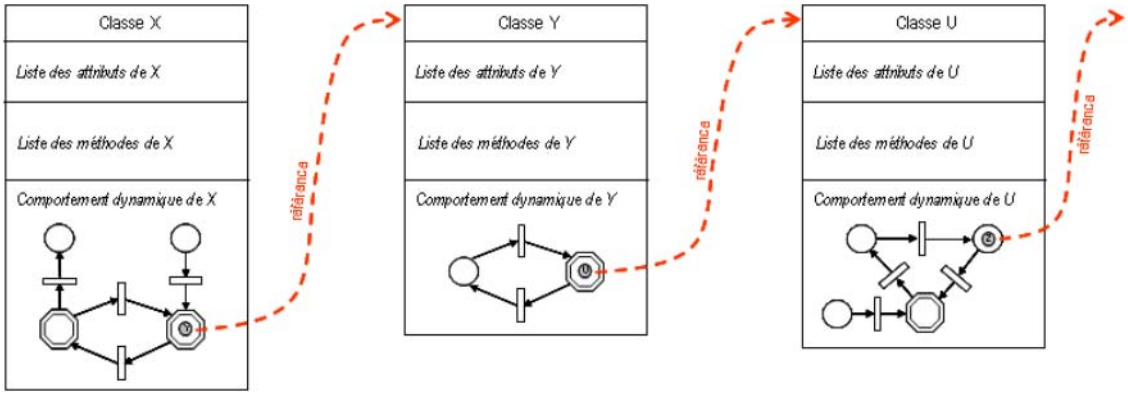


Figure 2.5 Approche combinée

2.2.3. Règles d'évolution dans les *RdPDO*

Après avoir défini les éléments sémantiques du formalisme, cette section se propose de rappeler les principales règles d'évolution dans les *RdPDO*.

Règle 2.1 : Arc sensibilisé

Un arc est dit *sensibilisé* si et seulement si on peut substituer à chaque variable formelle, au moins un jeton distinct parmi les jetons de la place située en amont.

De même, la notion d'*arc inhibiteur sensibilisé* est introduite :

Règle 2.2 : Arc inhibiteur sensibilisé

Un arc inhibiteur est dit *sensibilisé* si et seulement si la place située en amont ne contient pas de jetons pouvant être substitués aux variables formelles associées à l'arc.

Exemple *:*

Sur Figure 2.6.a, l'arc $a(p_1,t_1)$ *est caractérisé par le n-uplet de variables formelles* $\langle a,c \rangle$ *et la place située en amont possède deux jetons* a_1 *et* c_1 *portant les mêmes inscriptions ; par conséquent, l'arc* $a(p_1,t_1)$ *est sensibilisé. En revanche, l'arc* $a(p_2,t_2)$*, caractérisé par le même n-uplet, n'est pas sensibilisé car la place* p_2 *ne contient pas de jeton de type* c *; il est donc impossible de substituer un jeu de jetons au n-uplet de variables formelles* $\langle a,c \rangle$*.*

Sur la Figure 2.6.b, l'arc $a(p_1,t_1)$ *est caractérisé par la variable formelle* $\langle a \rangle$ *et la place située en amont possède un seul jeton* c_1 *de type* c *; il est donc impossible de substituer un jeton de la place* p_1 *à la variable formelle* $\langle a \rangle$ *; l'arc inhibiteur* $a(p_1,t_1)$ *est par conséquent sensibilisé. Au contraire, la présence d'au moins un jeton de type* a *sur la place* p_2 *(jetons* a_1 *et* a_2*) rend l'arc inhibiteur* $a(p_2,t_2)$ *non sensibilisé.*

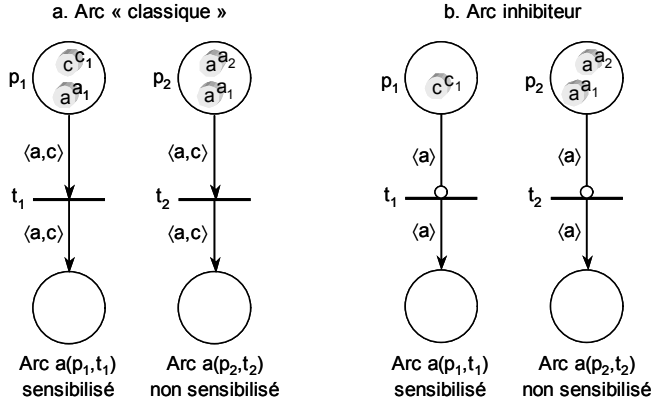


Figure 2.6 Sensibilisation des arcs

Des deux règles précédentes, on déduit :

Règle 2.3 : Transition sensibilisée

Une transition est dite *sensibilisée* si et seulement si tous les arcs situés en amont sont sensibilisés.

Les conditions de sensibilisation d'une transition étant définies, une deuxième règle peut être établie concernant les jetons :

Règle 2.4 : Jeton sensibilisé

Un jeton marquant la place p_i est dit *sensibilisé* si la transition située en aval de la place p_i est sensibilisée et s'il peut se substituer à l'une des variables formelles associées aux arcs de sortie de la place p_i.

Il reste maintenant à définir les conditions permettant de valider une transition. Pour cela, une règle a été établie et s'écrit comme suit :

Règle 2.5 : Transition validée

Une transition est dite *validée* si et seulement si :

- elle est sensibilisée,
- l'évènement associé est occurrent.

Lorsqu'un jeton marque une place différentielle, le système *EDA* associé est instancié par substitution des variables formelles par les attributs du jeton.

Exemple :

Soit A , l'instanciation d'un objet de type A (cf.Figure 2.7). Cet objet est caractérisé par un paramètre w et deux variables d'état u et v. Par ailleurs, elle possède trois méthodes. Son comportement dynamique est décrit par un RdPDO constitué de deux places discrètes p2 et p3 et d'une place différentielle p1. Les transitions t1 et t2 sont conditionnées par des conditions discrètes alors que la transition t3 est associée à une condition continue. Le franchissement de chaque transition induit l'exécution d'une action. La place différentielle est marquée par le jeton b1 de type b. La classe b possède une variable d'état z, deux méthodes et un RdPDO de deux places discrètes.

Le système algébro-différentiel associé à la place différentielle p_1 est construit à partir des deux variables d'état u et v propre à la classe A et de la variable d'état z portée par le jeton b_1 marquant la place. Il s'écrit de la façon suivante :

$$F_{p_1} = \begin{pmatrix} \dfrac{du}{d\theta} - \langle b \rangle.z \\ \dfrac{dv}{d\theta} + u - \dfrac{d\left(\langle b \rangle.z\right)}{d\theta} \end{pmatrix} = 0$$

Le marquage de la place p_1 induit l'intégration du système d'équations associé. La résolution induit l'évolution continue des variables d'état de l'objet A , ainsi que l'évaluation de la condition c_1 de la transition t_1. Comme pour le système d'équation, cette condition est évaluée en remplaçant la variable formelle <b> par le jeton b_1. Lorsque la transition est validée (cf. règle 4.5), l'action a_1 est exécutée et le jeton b_1 franchit la transition t_1. Cette action peut modifier la valeur des attributs de la classe A ou la valeur des attributs portés par le jeton b_1.

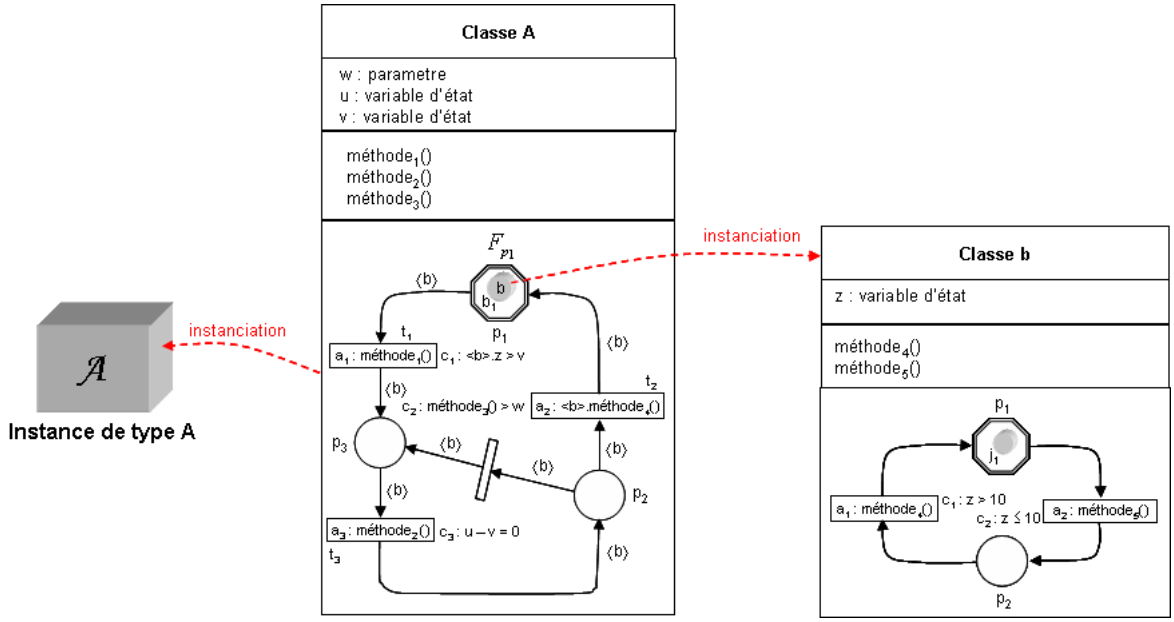


Figure 2.7 Marquage d'une place différentielle

Plusieurs jetons de même type peuvent marquer la place, déclenchant alors l'intégration du même système *EDA* à partir de la date de marquage. Par ailleurs, l'instanciation du système *EDA* d'une place différentielle peut nécessiter une combinaison de jetons de même type et/ou de types différents. Dans ce cas, la cohérence de l'ensemble doit être garantie par la modélisation ou validée a posteriori par la simulation.

Exemple *:*

Soit un objet de type CUVE (cf.Figure 2.8)

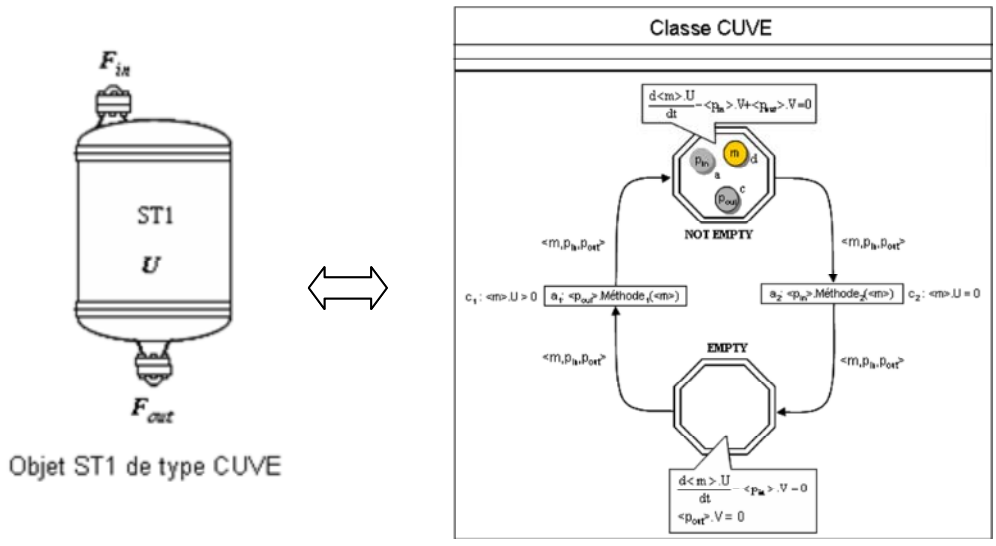


Figure 2.8 Modèle très simplifié d'un objet CUVE

On suppose que le débit traversant le port d'entrée est toujours nul. Dans l'intervalle [0..1[, aucun débit de sortie n'existe. A partir de t=1, la variable d'état V du jeton de type p_{out} devient différente de zéro, induisant une vidange de la cuve. Lorsque la rétention U du jeton de type m devient égale à zéro, la condition est atteinte et la transition est franchie. La cuve passe alors de l'état NOT EMPTY à l'état EMPTY, induisant la mise place d'un nouveau système EDA.

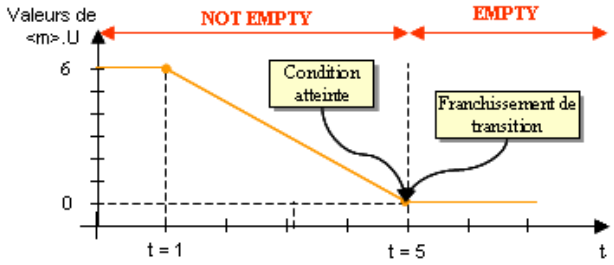


Figure 2.9 Changement d'état de la CUVE

2.2.4. Le gestionnaire de simulation

Afin de pouvoir « jouer » les *RdPDO*, le noyau du simulateur se décompose en trois modules : le *solveur discret* (joueur de *RdP*), le *solveur continu* (basé sur une extension de la méthode de Gear [Gear C.W. et al., 1971] ⇨ prédicteur/correcteur à pas variable robuste pour les systèmes raides) et le *gestionnaire de simulation*. Ce dernier est responsable de la simulation globale du système et est chargé de gérer les interactions entre les deux solveurs. Il donne à tour de rôle la main au joueur de réseau de Petri jusqu'à ce que son évolution conduise à un état stable (c'est-à-dire, un état où plus aucune transition ne puisse être franchie) et au solveur *EDA* qui intègre le modèle continu courant jusqu'à l'occurrence d'un événement.

Le cycle de fonctionnement du simulateur hybride est présenté sur la Figure 2.10 et est constitué de cinq phases essentielles :

1. Le gestionnaire de simulation construit le modèle continu global par concaténation de tous les systèmes *EDA* associés aux places différentielles initialement marquées et initialise les variables d'état.
2. Le modèle discret est ensuite exécuté jusqu'à ce que plus aucune transition ne puisse être franchie *(configuration stable)*. Les actions associées aux transitions franchissables sont exécutées et un nouveau marquage est établi.
3. Le gestionnaire de simulation concatène alors les systèmes *EDA* de chaque place différentielle marquée, ainsi que les fonctions de sensibilisation de chaque transition située en aval d'une place marquée.
4. Si cela s'avère nécessaire, l'intégrateur recalcule automatiquement de nouvelles conditions initiales cohérentes pour les variables d'état et leurs dérivées. Le gestionnaire de simulation reprend ensuite le contrôle de la simulation qui impose alors une nouvelle mise à jour des réseaux de Petri. Si aucune évolution n'est constatée et si le temps final de la simulation n'est pas atteint alors l'intégration du système algébro-différentiel global résultant est lancée pendant une durée égale à un temps de cycle préalablement établi. Notons que les conditions discrètes ne sont évaluées qu'à la fin de ces cycles.
5. L'intégration provoque l'évolution des variables continues au cours du temps. Celle-ci s'arrête dès qu'une fonction de sensibilisation est satisfaite (indiquant l'occurrence d'un événement) ou que le fin d'un cycle est atteint.
6. Le contrôle est repris par le solveur discret qui exécute les fonctions de jonction des transitions franchissables et établit le nouveau marquage.

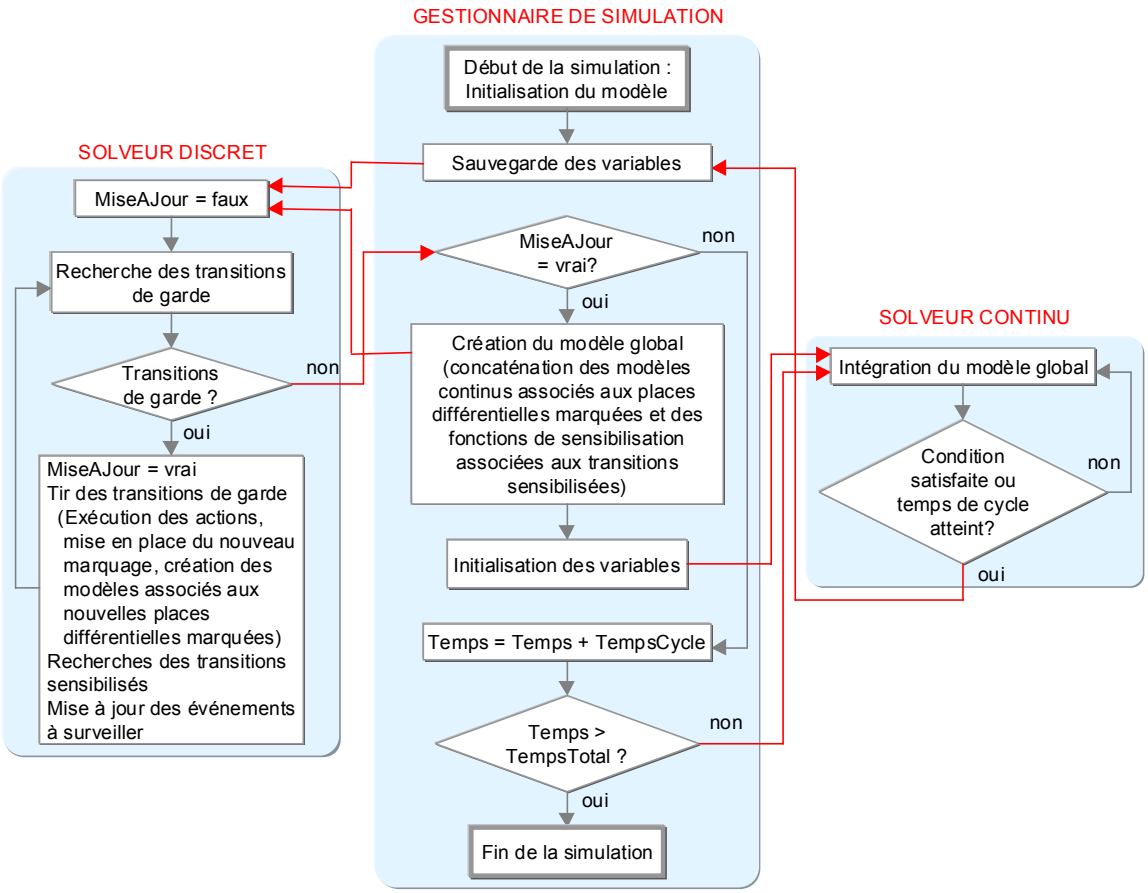


Figure 2.10 Phases du processus de simulation hybride

Outre la résolution des systèmes *EDA* et des modèles discrets, le noyau de simulation doit effectuer d'autres traitements fondamentaux tels que le calcul exact des instants de commutation, le recalage des états, la vérification de la cohérence des nouveaux modèles continus générés après commutation, la réinitialisation des variables d'état et de leurs dérivées.

Le gestionnaire de simulation peut être vu comme un moniteur d'*EDA* conduisant à la création dynamique d'un unique modèle de simulation dont la dimension et la structure fluctuent entre deux évènements. En effet, la modélisation adoptée repose sur une approche modulaire et privilégie la décomposition du modèle global en plusieurs sous-modèles. En revanche, le mode de simulation adopté est global. Cela signifie donc que, lorsque le solveur discret est lancé, tous les réseaux de Petri doivent être joués. De même, pour construire le modèle de simulation global, tous les sous-systèmes *EDA* des places différentielles marquées doivent être assemblés.

Afin de pouvoir parcourir tous les réseaux, une structure arborescente a été adoptée. Celle-ci permet, à partir d'un réseau donné, d'accéder à tous les réseaux de Petri de niveau inférieur. Point de départ de l'analyse, la procédure spécifique à la recette constitue le niveau *commande* et a le statut de réseau de Petri *maître*. Cette analyse se propage ensuite dans les réseaux de Petri de niveau inférieur, dits *esclaves*, jusqu'à atteindre les feuilles de l'arborescence. Cet ensemble de réseaux de Petri constitue le niveau *procédé*. Cette séparation entre *commande* et *procédé* est détaillée dans la section suivante.

Par ailleurs, l'arborescence est parcourue dans deux directions. En effet, le parcours évolue d'une part, selon une séquence définie par les relations *Composant/Composés* qui caractérisent les réseaux de Petri des appareils composés (vers le bas) tels qu'une colonne (cf. exemple ci-après) et d'autre part,

il évolue pour un sommet donné, selon une séquence définie par les relations *Contenant/Contenus* qui lient un réseau de Petri à ceux intégrés dans les jetons de ses places (dans le sens de la profondeur) tels que la matière ou les ports (cf. exemple ci-après). L'exploitation de méthodes récursives s'avère particulièrement efficace pour explorer cette structure arborescente. Afin d'illustrer ces propos, considérons l'exemple suivant.

Exemple *:*

Soit un procédé, constitué d'une alimentation et d'une colonne et piloté par un réseau de Petri recette donné (Figure 2.11).

L'alimentation constitue un appareil élémentaire spécifique. La colonne, plus complexe, est décomposée en plusieurs appareils :

- un bouilleur décomposé en un système thermique et une cuve,

- un condenseur également décomposé en un système thermique et une cuve,

- et un ensemble de zones à garnissage ou à plateaux, décomposées elles-mêmes en étages.

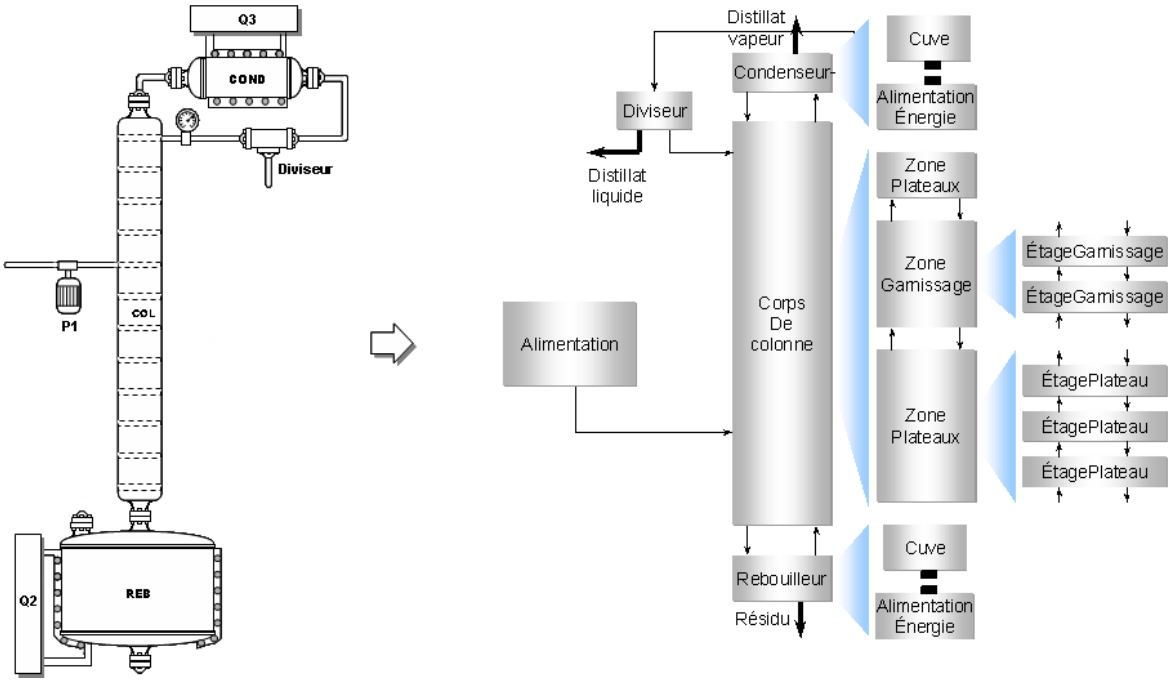


Figure 2.11 Décomposition des appareils du procédé en appareils élémentaires

Les réseaux de Petri associés à ces différents appareils sont organisés de façon hiérarchique selon la structure indiquée sur la Figure 2.12. La recette constitue le réseau de Petri dit maître. Tous les autres constituent les réseaux de Petri dits esclaves.

Dans cet exemple, ces derniers se décomposent :

- dans le sens Composant /Composés, sur quatre niveaux.

- dans le sens Contenant/Contenus, où seules les feuilles initient une arborescence dans le sens de la profondeur. En effet, elles correspondent aux réseaux de Petri d'appareils élémentaires spécifiques qui sont les seuls à être dotés de jetons intégrant d'autres réseaux de Petri. Il s'agit des jetons Port (qui intègrent le réseau de Petri associé au port) et Matière (qui intègrent le réseau de Petri associé à la matière). Le réseau de Petri Matière possède lui-même des jetons Phase qui intègrent un réseau de Petri spécifique associé à la phase.

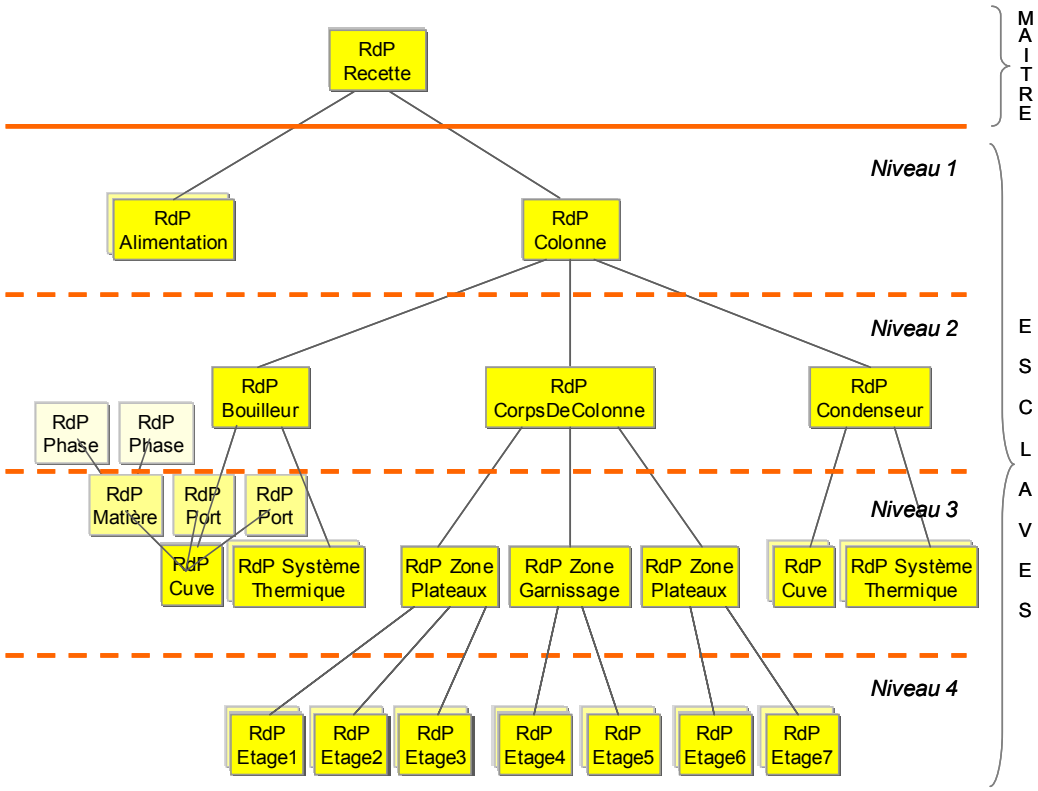


Figure 2.12 Structure hiérarchique du modèle de l'exemple

2.3. MODÉLISATION OBJET D'UN PROCÉDÉ

Après avoir décrit le formalisme utilisé pour modéliser le comportement dynamique (ou *phénoménologique*) du système à simuler, cette section montre la manière dont est modélisée sa structure statique (ou *topologique)*. S'appuyant sur une approche objet, cet aspect consiste à définir l'ensemble des éléments constituant le système et les connexions qui existent entre ces éléments. Néanmoins, cette section se concentre essentiellement sur le niveau *commande* et ses interactions avec le niveau *procédé*, ce dernier niveau ayant été largement décrit dans les travaux de [Moyse, 2000], [Perret, 2003] et [Olivier-Maget, 2007].

2.3.1. Structure générale du modèle de simulation

Comme la section 1.4.3 l'a montré, la simulation d'un procédé discontinu nécessite de modéliser à la fois la partie recette (la *commande*) et la partie opérative (le *procédé*). Dans ce cadre, une hiérarchisation des modèles a donc été introduite (cf. section 2.2.4). Le niveau *commande* décrit l'aspect procédural de la recette à exécuter alors que le niveau *procédé* décrit le comportement du système en réponse aux signaux de commande (Figure 2.13).

Cette structuration est en accord avec la philosophie objet introduite dans *PrODHyS* qui consiste à générer des entités réutilisables et autonomes. Ici, il s'agit de faire en sorte que la modélisation de la commande (le superviseur en quelque sorte) n'ait aucun impact direct sur la modélisation de la partie opérative du procédé. Ainsi, différentes recettes peuvent être implémentées et testées sans modifier aucunement les modèles associés aux appareils. Pour cela, les appareils ne sont connectés au niveau commande que par une interface limitée décrite dans la suite. De même, la matière est modélisée comme un objet indépendant dont le système d'équations représentant son comportement est découplé du modèle de l'appareil qui la contient. Cet objet est alors réutilisable quelque soit le contexte dans lequel il est employé.

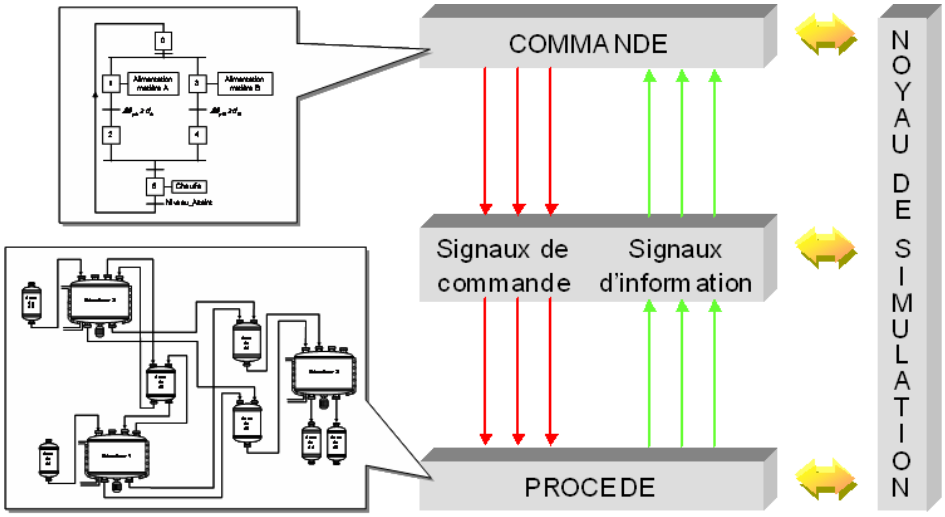


Figure 2.13 Structure générale d'un modèle de simulation

2.3.2. Modélisation de la matière

Afin de limiter la combinatoire liée au comportement de la matière, la philosophie adoptée dans *PrODHyS* est de faiblement coupler celle-ci avec l'appareil qui la contient. Ainsi, plutôt que de confondre le comportement de la matière avec celui de l'appareil, chaque élément est décrit séparément (cf. exemple ci-dessous).

> ***Exemple*** *:*
>
> *Supposons qu'un appareil soit décrit par n états et que deux états de la matière soient seulement considérés (Liquide et Liquide/Vapeur). En couplant les deux systèmes, le modèle résultant possèderait alors potentiellement 3x n états. En les dissociant, on obtient au contraire un modèle de 2+n états.*

Ainsi, chaque sous-système possède son propre réseau de Petri. Néanmoins, le comportement de la matière reste intégré au modèle de l'appareil grâce à un jeton objet représentant la Matière (et indirectement son réseau de Petri) et se déplaçant sur le réseau de Petri de l'appareil. Le modèle continu global de l'ensemble *appareil/matière* est alors obtenu par concaténation du système algébro-différentiel associé à l'état courant de la matière avec le système algébro-différentiel associé à l'état courant de l'appareil. Enfin, soulignons par ailleurs que cette décomposition facilite l'extension des modèles (cf. exemple ci-dessous).

> ***Exemple** :*
>
> *Reprenons l'exemple précédent dans lequel pour la matière, on souhaite considérer en plus l'état Vapeur seul. Pour cela, il suffit d'ajouter cette configuration dans le RdP de la matière, soit un état. Cette prise en compte dans une représentation monolithique du système aurait induit n états et de nombreuses transitions supplémentaires.*

Dans *PrODHyS*, la matière est considérée comme un ensemble de phases, chacune étant caractérisée par des grandeurs spécifiques. En effet, la *phase* est généralement décrite par des propriétés physiques qualitatives telles que l'enthalpie, le volume molaire, la viscosité, etc. et par des grandeurs quantitatives telles que la rétention molaire ou volumique. Elle contient, par conséquent, les équations permettant de déterminer ces variables. Quant au *système de phases*, il est supposé homogène et à l'équilibre thermodynamique. Les variables qui lui sont propres sont par conséquent la température et la pression et les équations qu'il possède concernent les relations d'équilibre entre phases. Le modèle relatif à la matière est décomposé :

- une partie regroupe les équations et variables communes à l'ensemble des phases et au système de phases. Par exemple, une place correspondant à un état *liquide/vapeur* contient les équations traduisant l'équilibre liquide/vapeur ; en revanche, une place représentant un état *monophasique* ne possède aucune équation particulière.
- l'autre partie du modèle regroupe les équations et variables permettant le calcul des propriétés physiques des phases ; il est, cette fois-ci, directement associé aux places différentielles du *RdP* des phases.

Ainsi, un jeton objet de type Phase a été créé. Les équations relatives à la phase sont ainsi associées aux places du *RdP* de la phase. Le modèle de la phase reste cependant intégré à celui de la matière grâce au jeton Phase qui se déplace sur le *RdP* de la matière. La classe Phase est en fait dérivée en une classe phase vapeur φ_v ou liquide φ_l, ce qui permet de prendre en compte les différents états de la matière.

> ***Exemple** :*
>
> *Soit un objet Appareil dont le comportement est décrit par 8 places différentielles, chacune décrivant un état continu (Figure 2.14). Le jeton Matière se déplace sur le réseau de l'appareil et son comportement est lui-même décrit par un réseau de Petri. Celui-ci possède deux places : une place p_1 symbolisant l'état monophasique de la matière dont la nature dépend du jeton marquant la place et une place p_2 symbolisant l'état liquide/vapeur indiquant la présence des phases liquide et vapeur dans le système de phases. En ce qui concerne les jetons Phase, ils se déplacent sur le RdP du système de phases et sont dotés d'un RdP particulier. Celui-ci comporte deux places correspondant à deux configurations distinctes.*

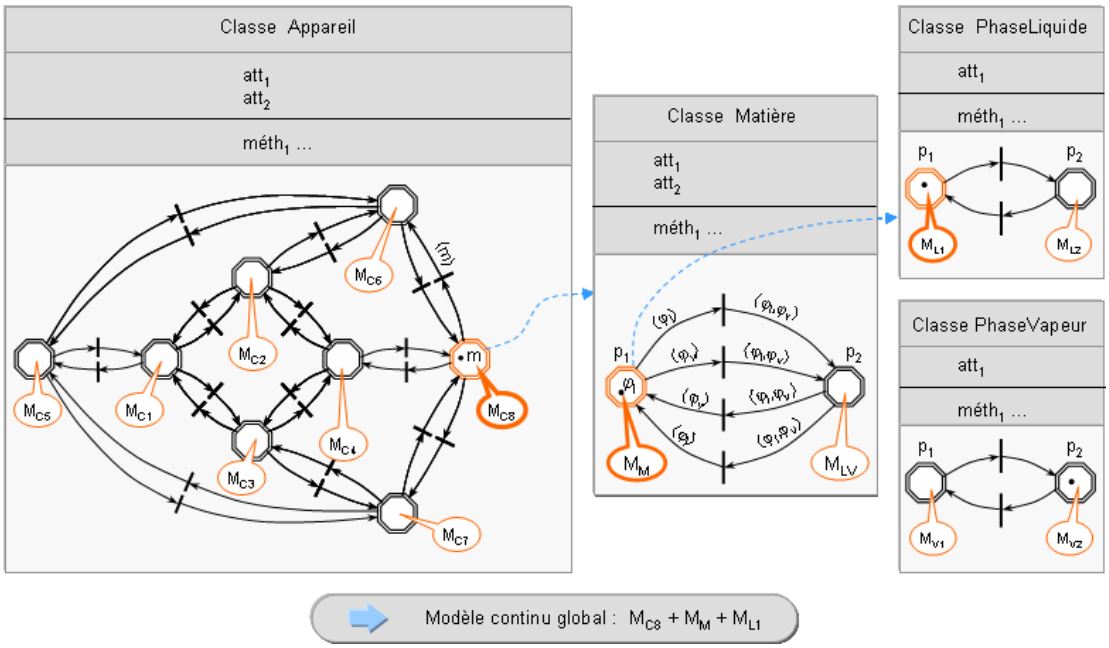


Figure 2.14 Modèles Appareil/Matière/Phases

Si l'on considère que le marquage est celui indiqué sur la Figure 2.14, le modèle continu qui en résulte est issu de la concaténation du système EDA M_{C8} associé à la place marquée p_8 du RdP de l'appareil, du système EDA M_M associé à la place marquée p_1 du RdP de la matière et enfin du système EDA M_{L1} associé à la place marquée p_1 du RdP de la phase liquide. Notons que si la matière est dans l'état liquide, seul le jeton Phase liquide φ_l *marque la place différentielle monophasique du système de phases. Le jeton* Phase *portant la phase vapeur n'existe pas.*

2.3.3. La modélisation de la partie opérative

2.3.3.1. Topologie d'un appareil élémentaire

La partie opérative d'un système est constituée d'un ensemble d'appareils. De manière générale, un *appareil élémentaire* est défini comme une enceinte de bilan qui peut échanger de la matière, de l'énergie ou de l'information. Afin de formaliser ces échanges, un élément d'interface nommé *port* a été introduit. Compte tenu de la nature des échanges, deux types de port ont été identifiés :

- les ports de communication qui permettent un échange d'information,
- et les ports de transport qui permettent un échange physique de matière ou d'énergie. Dans ce dernier cas, le transfert est caractérisé par un flux et un potentiel.

Définir la topologie d'un appareil consiste donc à déterminer le type, le sens (*entrant* ou *sortant*) et le nombre de ports qu'il possède et dont il est responsable (Figure 2.15).

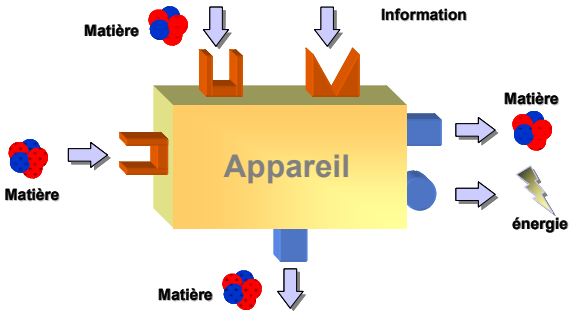


Figure 2.15 Notion d'appareil

Toutes les classes modélisant un appareil « réel » sont dérivées (*par héritage*) de la classe `ElementaryDevice`. Cette classe caractérise les plus petites entités manipulables dans *PrODHyS* (*entités non décomposables*) et fixe la granularité de décomposition du procédé. En fait, cette classe définit une portion de procédé conceptuellement isolable et pas nécessairement une unité de génie chimique. Il s'agit d'un concept beaucoup plus général qui permet de représenter autant un composant d'une colonne que d'un réacteur.

2.3.3.2. Topologie d'un procédé

Il s'agit ici de représenter la structure du procédé selon une vision système, c'est à dire en termes de connexions entre diverses structures composées hiérarchiquement au travers desquelles circulent matière, énergie et information. En conséquence, dans *PrODHyS,* la spécification structurelle de tout procédé est toujours définie selon deux axes :

- un **axe connexion** : cet axe décrit les connexions entre les éléments constitutifs du procédé (cf. Figure 2.16). Il se généralise à tout type de communication, en éliminant totalement la notion de circulation : il n'y a donc aucun transfert, ni duplication d'information, simplement une visibilité offerte à un élément vers un autre (principe d'encapsulation) par l'intermédiaire des ports. Ce principe induit qu'il n'existe aucune variable globale, que le partage d'une variable entre plusieurs sous-systèmes se fait toujours de manière explicite et enfin que la synchronisation est implicite puisque l'intégration des systèmes *EDA* actifs est effectuée de façon globale. Une connexion s'établit par la mise en relation de deux ports mais reste totalement acausale. La notion de port d'entrée ou de sortie permet seulement de fixer le sens conventionnel du flux (positif si *sortie* ⇨ *entrée*). Selon cet axe, un procédé est représenté par un graphe orienté.

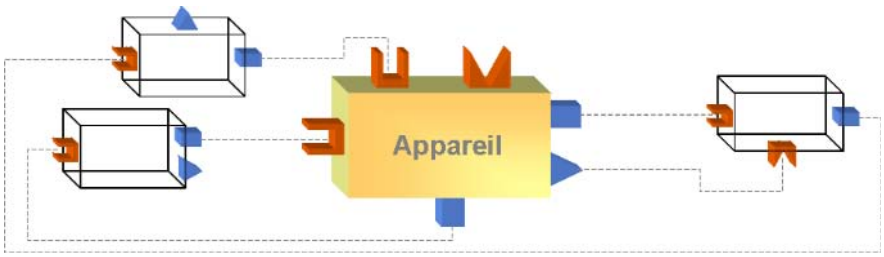


Figure 2.16 Connexion entre appareils

- un **axe décomposition/composition** : cet axe décrit la structure hiérarchique interne de chaque entité (Figure 2.17) et s'appuie sur la classe `CompositeDevice`. Selon cet axe, un procédé apparaît alors sous la forme d'un arbre dont les feuilles sont les appareils élémentaires et dont la racine est un objet général de type `Flowsheet`.

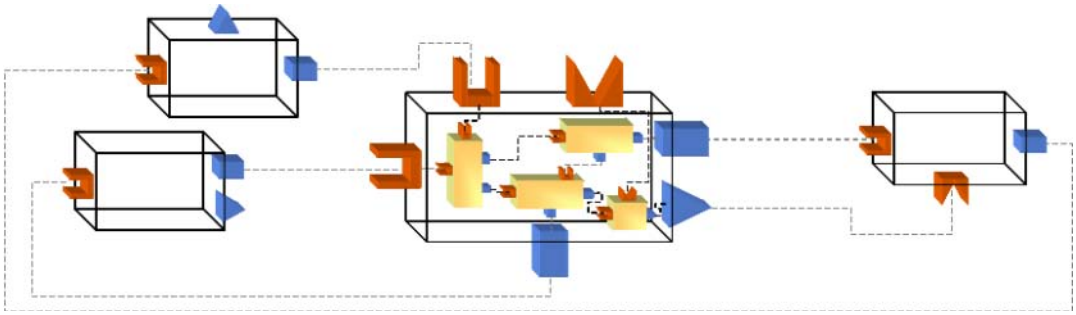


Figure 2.17 Structure interne d'un appareil composé

Les classes de fondation associées aux appareils s'organisent selon le diagramme de classe de la Figure 2.18

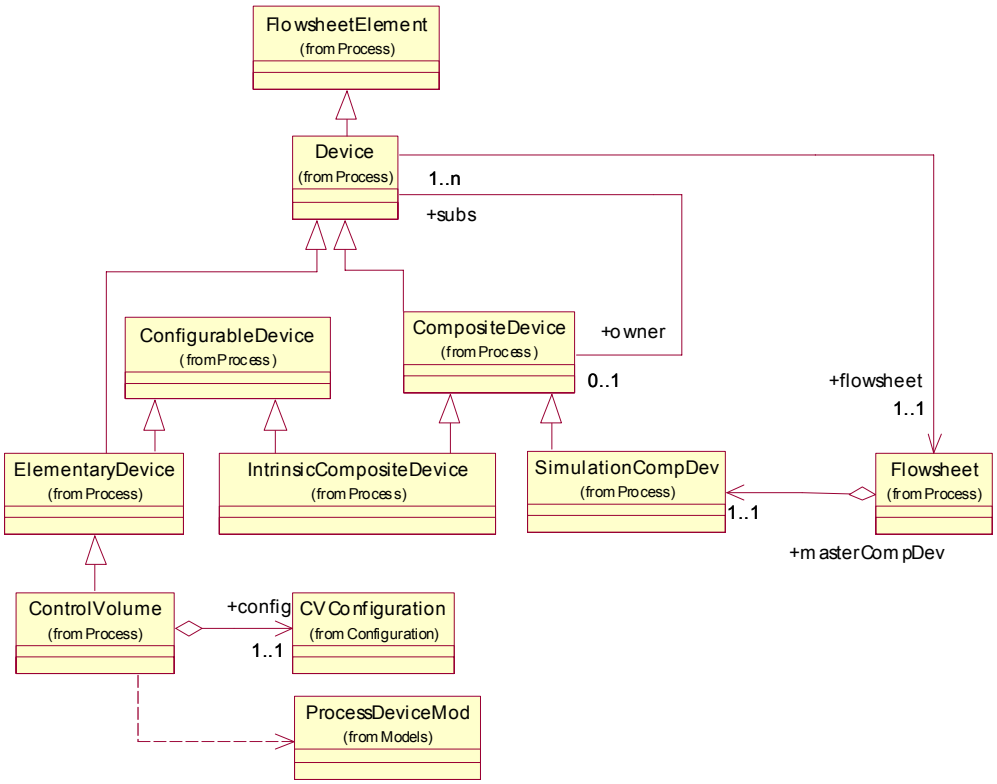


Figure 2.18 Classes de fondation pour la modélisation des appareils

La construction d'un procédé s'appuie donc seulement sur deux types initiaux d'éléments : les *appareils élémentaires* (classe `ElementaryDevice`) non-décomposables et qui peuvent être spécialisés (cf. section 2.3.3.1) et les *appareils composés* (classe `CompositeDevice`) qui décrivent

récursivement la composition des appareils plus complexes. Cela signifie qu'il peut contenir d'autres appareils composés et que le niveau de décomposition est donc infini. Notons qu'un objet de type `CompositeDevice` n'est responsable ni de la création, ni de la destruction des appareils le constituant ; il s'agit simplement d'un regroupement d'appareils. De même, il possède des ports mais ces ports ne lui appartiennent pas; il s'agit des ports des appareils le constituant non connectés entre eux (Figure 2.19).

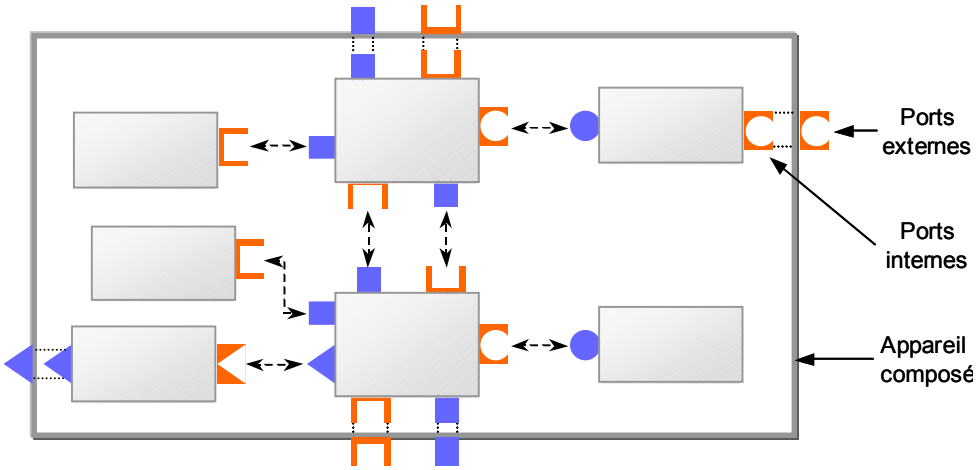


Figure 2.19 Ports d'un appareil composé

La classe `CompositeDevice` est elle-même spécialisée en deux autres classes:

- Tout d'abord, la classe `IntrinsicCompDev` permet de modéliser des appareils complexes réels (des colonnes par exemple) dont le concepteur a figé la construction. Cet objet est ici responsable de la construction et de la destruction des appareils qui constituent l'appareil composé.
- L'autre classe d'appareil composé est la classe `SimulationCompDev`. Elle définit un appareil composé de simulation représentant tout le procédé. Sachant que la stratégie de simulation adoptée repose sur une approche globale orientée équations, cet objet est responsable de la création du modèle global du procédé et est associé au noyau de simulation.

Exemple *:*

A titre d'illustration, la Figure 2.20 montre la structure adoptée pour modéliser un procédé constitué d'une colonne, de deux alimentations et d'un automate pour la commande.

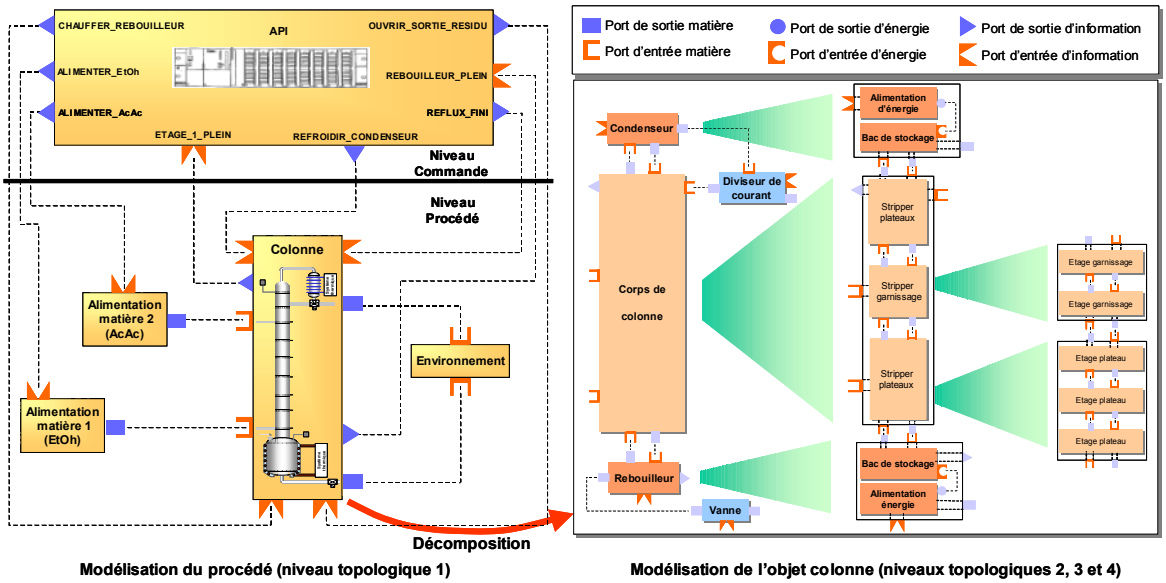


Figure 2.20 Modélisation topologique d'un procédé

Tout échange de matière, énergie ou information entre ces appareils apparaît explicitement via les différents ports (niveau topologique 1). La colonne étant un appareil complexe, celle-ci est construite par composition d'appareils plus simples répartis sur trois niveaux topologiques. Conceptuellement, une colonne peut être vue comme un élément composé d'un condenseur, d'un rebouilleur (association pour tous deux, d'une cuve de stockage et d'une alimentation énergie), d'un diviseur de courant, d'une vanne et d'un corps de colonne. Ce dernier élément est lui-même constitué de plusieurs sections, chacune contenant un ensemble d'étages de garnissages ou de plateaux interconnectés. Notons qu'une section de colonne est un objet générique (type générique ≡ type d'étage présent) et que c'est un appareil composé intrinsèque (classe `IntrinsicCompDev`*): cela signifie qu'il construit et configure lui-même les étages le constituant à partir d'une configuration que l'utilisateur doit renseigner.*

Rappelons que ces classes sont conçues indépendamment des phénomènes physiques que l'on cherche à simuler. C'est l'introduction de jetons de type matière et de type port dans le *RdPDO* associé à l'élément de procédé qui permet de simuler les phénomènes physico-chimiques qui s'y déroulent, ainsi que d'effectuer des bilans (matière et énergie). Chaque élément de procédé, quel qu'il soit, est alors considéré comme une enceinte de bilans pour lequel les ports définissent les flux échangés avec l'extérieur et le volume interne définit les rétentions de matière et d'énergie. Nous parlons alors de *volume de contrôle* (classe `ControlVolume`) pour caractériser l'aspect conteneur de phénomène. Conceptuellement, un volume de contrôle représente donc une portion finie de matière en équilibre, dans un état thermodynamique donné (température, pression, composition) et constitue l'ancêtre de nombreuses classes utilisées pour la description d'appareils contenant de la matière (Figure 2.19). La quantité de matière associée au volume de contrôle est définie comme un objet de type `ReactivePhaseSystem`. Enfin, notons que ce système de phases réactif peut être caractérisé par un ensemble de réactions chimiques (réactions contrôlées ou équilibrées).

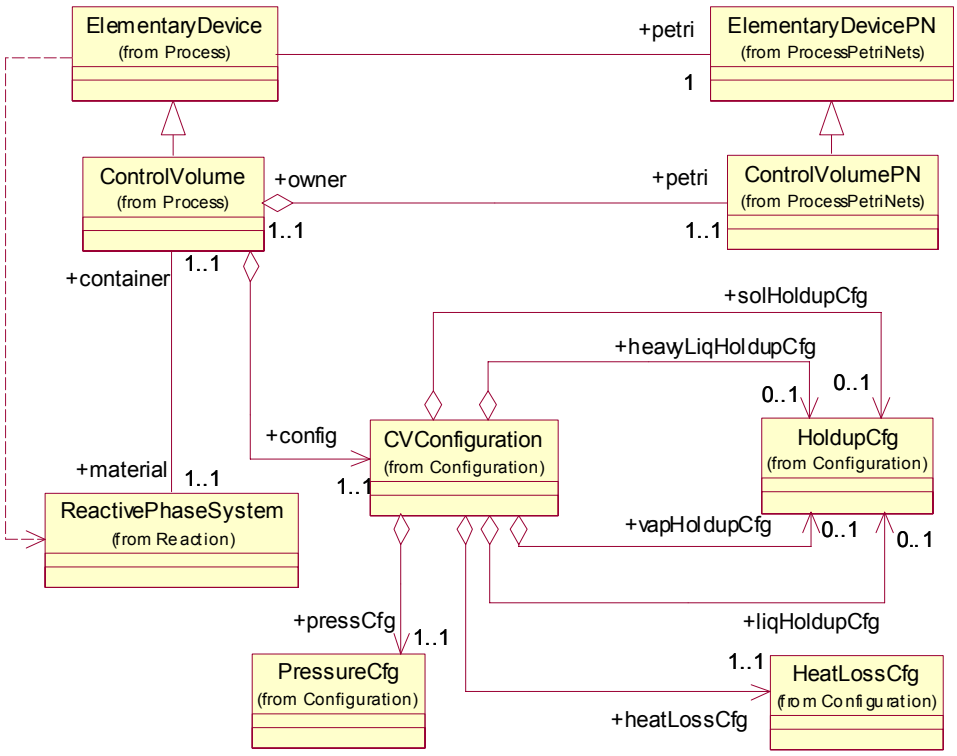


Figure 2.21 Classe ControlVolume

Les modèles continus correspondants dépendent alors du marquage des places différentielles. Les variables et paramètres nécessaires sont obtenus via les jetons marquant la place. Les modèles sont ainsi formulés à l'aide des variables formelles ⟨m⟩, ⟨φ⟩, ⟨p⟩, qui sont substituées respectivement par les jetons Matière, Phase et Port du réseau.

Exemple *:*

Pour illustrer ces propos, considérons un appareil possédant n_{pme} ports matière d'entrée, n_{pms} ports matière de sortie et contenant de la matière multiphasique de n_φ phases. Le bilan matière global et les n_c bilans matière partiels s'écriraient de la manière suivante :

$$\frac{d\left(\sum_{k=1}^{n_\varphi} \langle m\rangle.\langle \varphi^{(k)}\rangle.U\right)}{d\theta} - \sum_{k=1}^{n_{pme}} \langle p^{(k)}\rangle.F + \sum_{k=1}^{n_{pms}} \langle p^{(k)}\rangle.F = 0$$

$$\frac{d\left(\sum_{k=1}^{n_\varphi} (\langle m\rangle.\langle \varphi^{(k)}\rangle.U)(\langle m\rangle.\langle \varphi^{(k)}\rangle.z_i)\right)}{d\theta} - \sum_{k=1}^{n_{pme}} (\langle p^{(k)}\rangle.F)(\langle p^{(k)}\rangle.z_i) + \sum_{k=1}^{n_{pms}} (\langle p^{(k)}\rangle.F)(\langle p^{(k)}\rangle.z_i) = 0 \qquad (i = 1, n_c)$$

L'instanciation de ce modèle induit que la place différentielle doit être marquée avec n_{pme} jetons de type <p_{in}>, n_{pms} jetons de type <p_{out}> et un jeton de type <m> dont le RdP est lui-même marqué de n_φ jetons de type <φ>(ou classe dérivée).

2.3.4. Modélisation du niveau commande

En procédé, la *recette* rassemble les données techniques nécessaires à la fabrication d'un produit. La section 1.2.5 a montré que celle-ci est hiérarchisée en quatre niveaux et est structurée autour de cinq éléments d'information. Dans *PrODHyS*, le *niveau commande* du modèle de simulation correspond à la modélisation de la partie *procédure* au niveau *recette de contrôle*. Celle-ci s'applique à un lot particulier et pilote l'ensemble des séquences opératoires qui le suivent pendant tout son processus de fabrication.

2.3.4.1. Modélisation de la procédure avec les *RdPDO*

Le formalisme adopté pour décrire la procédure repose évidemment sur le formalisme *RdPDO*. Notons toutefois qu'au niveau *commande*, les aspects continus sont quasiment inexistants et sont essentiellement liés à l'expression explicite de durées opératoires fixées (temps réactionnel, par exemple). Dans ce cadre, la notion de *place temporisée* est introduite. L'exemple ci-dessous illustre leur utilisation.

***Exemple** :*

Soit la séquence ci-dessous qui modélise un appareil en état d'alimentation pendant une durée d (Figure 2.22). Pour cela, une place temporisée est introduite (classe `TimerPlace`*). Cette place dérive d'une place différentielle pour laquelle est associée automatiquement un paramètre* **d** *(durée de la temporisation) et l'équation différentielle indiquée sur la figure. La variable θ comptabilise la durée de marquage de la place. Une action particulière associée à la transition qui précède la place temporisée initialise à 0 cette variable θ. Le jeton binaire est alors sensibilisé dès que la condition de la transition qui succède à la place temporisée est validée.*

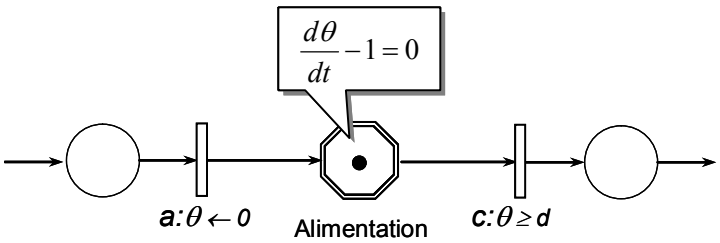
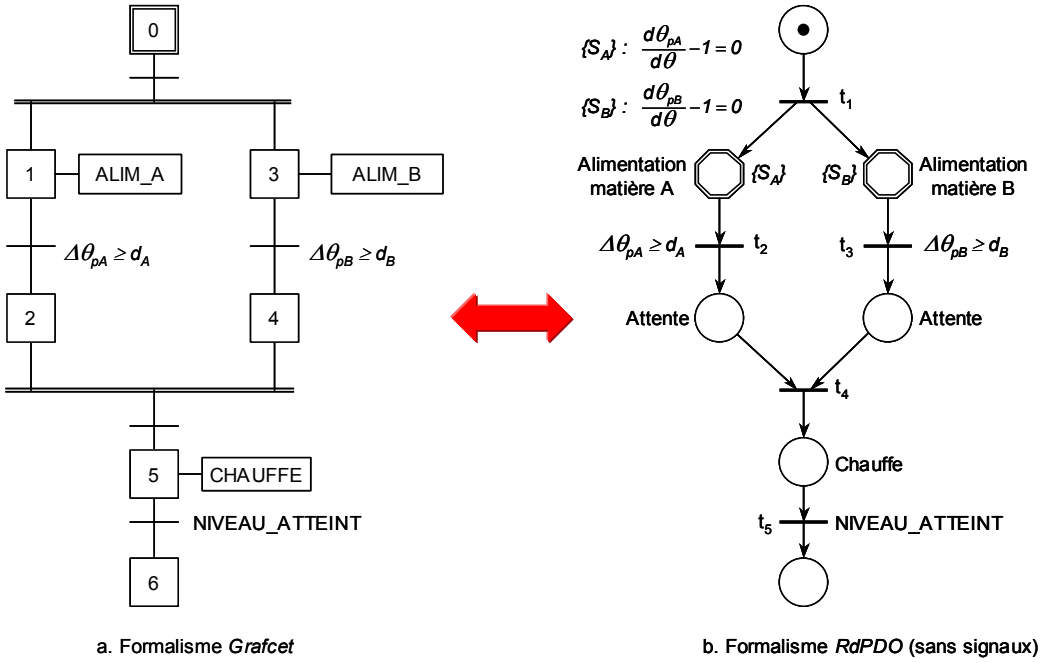


Figure 2.22 Place temporisée

Le formalisme *RdPDO* étant une extension des réseaux de Petri autonomes, il permet de gérer aisément les contraintes de parallélisme, de synchronisation, de séquencement ou d'affectation de ressources largement présentes au niveau *commande*. Par ailleurs, il est facile de traduire en *RdPDO* des lois de commande modélisées par d'autres formalismes comme par exemple le ***GRAFCET***, couramment utilisé dans le secteur industriel pour la programmation des automates industriels (*API*).

Exemple :

La Figure 2.23 propose la traduction dans le formalisme RdPDO d'une séquence en GRAFCET.



a. Formalisme *Grafcet* b. Formalisme *RdPDO* (sans signaux)

Figure 2.23 Modélisation du niveau commande

La séquence opératoire à effectuer est la suivante : il s'agit d'alimenter une cuve avec deux matières simultanément, puis à chauffer le mélange jusqu'à ce que celui-ci soit descendu en-dessous d'un niveau défini au préalable (parallélisme, synchronisation).

La Figure 2.24 donne le diagramme de classe correspondant à la classe centrale `PetriNet`. En effet, cette classe modélise la structure générale d'un *RdPDO* et contient l'ensemble des méthodes permettant de gérer et de faire évoluer le réseau.

La classe générale `RecipePN` (dérivé de la classe `PetriNet`) a été définie afin de regrouper les éléments généraux qui caractérisent une procédure de recette. De part sa fonction, ce réseau de Petri se situe hiérarchiquement au niveau le plus élevé (cf. section 2.2.4). Par conséquent, la valeur de son attribut `master` (défini dans la classe mère `PetriNet`) est fixée à vrai, ce qui le qualifie de réseau de Petri *maître*. La classe `RecipePN` est également responsable d'initier la simulation de la partie discrète. Dans ce contexte, la méthode `simulate()` est redéfinie.

A ce niveau, le réseau de Petri représenté par la classe `RecipePN` ne contient encore aucun élément en termes de places et transitions. En effet, la procédure suivie par un procédé est généralement spécifique et doit donc être décrite par l'utilisateur. La spécification d'une procédure particulière consiste alors à spécialiser la classe `RecipePN` et à surcharger un certain nombre de ces méthodes.

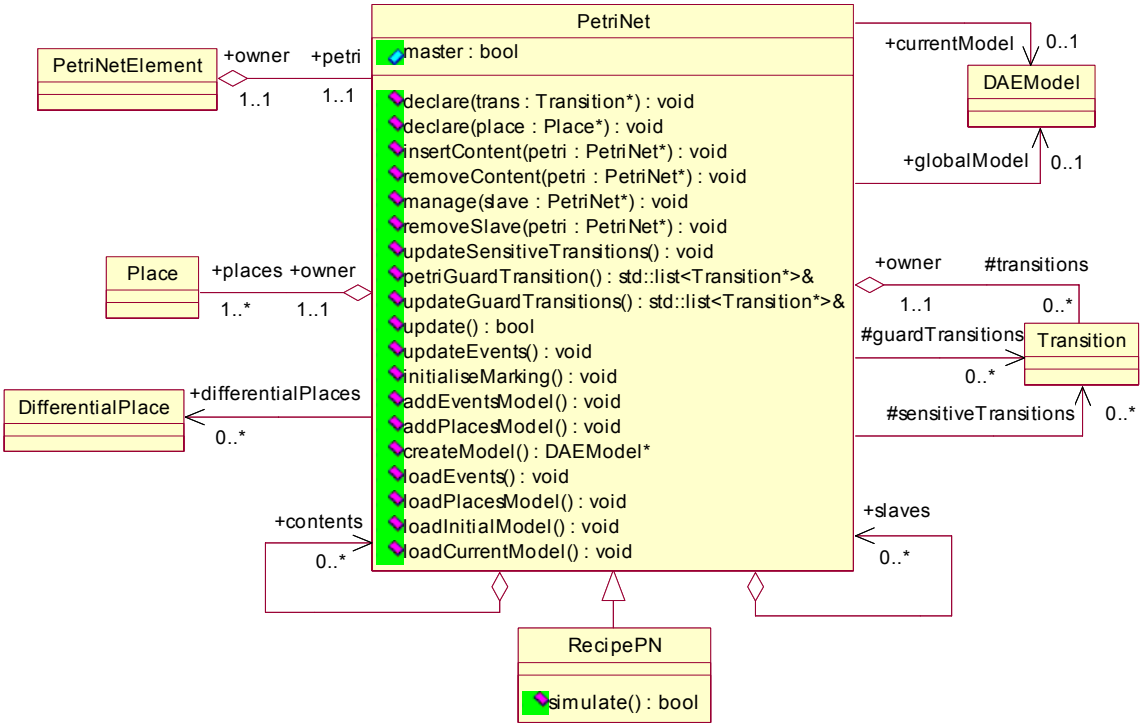


Figure 2.24 Diagramme de classes de PetriNet et RecipePN

2.3.4.2. Connexion des *RdPDO* du niveau *procédé* et *commande*

Chaque entité de la partie opérative (appareils élémentaires ou composés) évolue en réponse à deux types distincts d'évènement :

- D'une part, des évènements dits *externes* qui provoquent des *commutations commandées*. Il s'agit de tous les signaux échangés entre le niveau *commande* et le niveau *procédé* : les commandes émises depuis le *RdP recette*, l'occurrence d'un évènement d'état (détection d'un seuil) ou d'un évènement temporel (délai écoulé). Spécifiés par l'utilisateur, ces évènements apparaissent explicitement sur le *RdP recette*.
- D'autre part, des évènements dits *intrinsèques* dont l'occurrence dépend uniquement de l'évolution spontanée du procédé. Ces *commutations autonomes* correspondent au changement d'état de la matière (passage de l'état liquide à l'état liquide/vapeur lorsque la température d'ébullition est atteinte, par exemple) ou d'un appareil non commandé. Ceux-ci n'apparaissent donc pas explicitement sur le *RdP recette* (l'utilisateur n'a donc pas à les spécifier) et sont traités exclusivement au sein du modèle de l'entité concernée.

Comme indiqué, les signaux échangés entre la partie *procédé* et la partie *commande* relèvent donc de la première catégorie d'évènements. En effet, ils correspondent soit à l'émission d'une commande émanant du *RdP* recette, soit à la réception d'une information provenant du procédé. Dans les deux cas, la nature du signal échangé est binaire, discrète ou réelle.

La section 2.3.3.2 a montré que les variables de chaque entité sont encapsulées et rendues inaccessibles depuis l'extérieur. C'est pourquoi les flux d'information échangés apparaissent de façon *explicite*. Dans *PrODHyS*, deux possibilités sont offertes pour réaliser cet échange :

- soit au travers de *ports d'information* (cf. section 2.3.3.1),
- soit en utilisant une place particulière dite *signal* qui n'accepte que des jetons binaire ou d'information. L'état du signal est alors associé au marquage de cette place.

Dans *PrODHyS*, cette deuxième solution a été privilégiée (Figure 2.25), notamment pour les commandes tout ou rien (*TOR*). Le *signal de commande* est généralement émis à l'aide de deux arcs : *Set()* et *Reset()* ; ils sont liés à des transitions du *RdP* de niveau *commande* et permettent respectivement la mise à *un* et la mise à *zéro* du signal. Le marquage de la place discrète indique ainsi l'émission d'une commande, qui est transmise aux éléments commandés grâce aux arcs *Test()*. Ces derniers permettent en effet de vérifier l'état « signalé » ou « non signalé » de la place *signal* (utilisation d'un arc inhibiteur dans ce dernier cas). Dans le cas du *signal d'information*, le marquage de la place discrète informe le niveau commande d'un évènement particulier survenu au niveau du procédé et l'autorise à poursuivre la séquence opératoire.

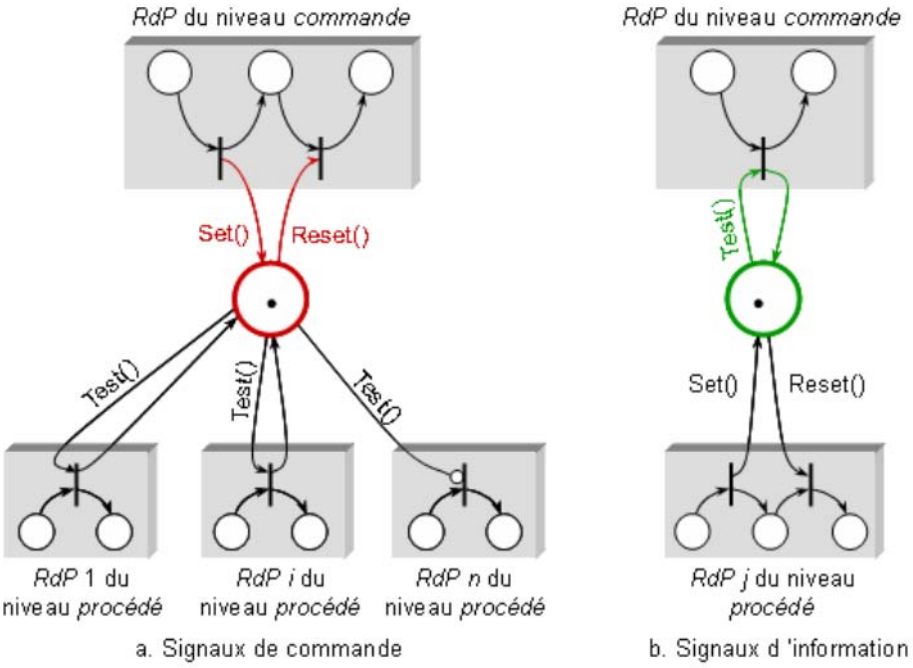


Figure 2.25 Modélisation des signaux échangés entre niveau commande *et* procédé

Ces places *signal* sont donc le seul lien entre les appareils commandés et la procédure d'exécution. Considérés comme des boîtes noires, on distingue de ce fait deux types de dispositifs :

- les *dispositifs actifs* : il s'agit des appareils dont le *RdP* possède une ou plusieurs places *signal* (de commande et/ou d'information) tels que les vannes, les pompes, les alimentations d'énergie, les capteurs, etc.
- les *dispositifs passifs* : ce sont les entités dont le *RdP* ne possède aucun lien direct avec le *RdP recette* tels que les cuves, les réacteurs ou la matière.

Les interactions entre *RdP* du niveau *commande* et *RdP* du niveau *procédé* sont illustrées sur l'exemple suivant.

Exemple *:*

Soit le procédé de la figure ci-contre. Celui-ci est constitué d'une cuve, d'une alimentation pilotée par une vanne TOR, d'un capteur de niveau et d'un automate pour la commande. La séquence opératoire à exécuter est d'alimenter un réacteur jusqu'à ce qu'un volume désiré soit atteint. Ce système se modélise sous PrODHyS tel que le montre la Figure 2.27.

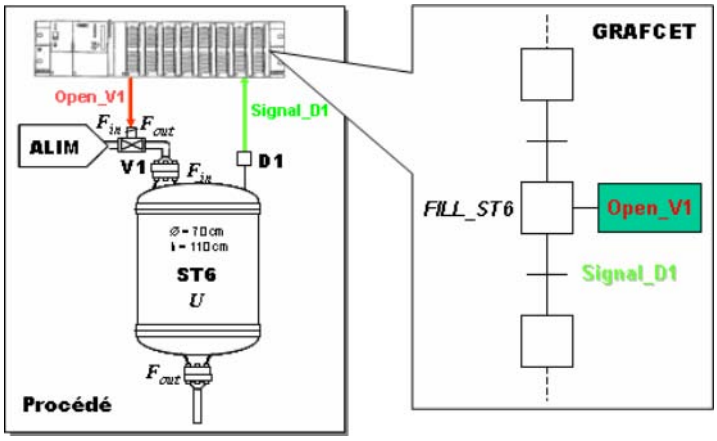


Figure 2.26 Elements constitutifs de l'exemple

L'objet Cuve n'ayant aucune place signal, celui-ci est donc un dispositif passif. Au contraire, les objets Vanne et Détecteur sont des dispositifs actifs car ils possèdent respectivement une place de commande Open_V1 et une place d'information Signal_D1. Cependant, notons que la rétention U dans la cuve est transmise au détecteur via un port d'information.

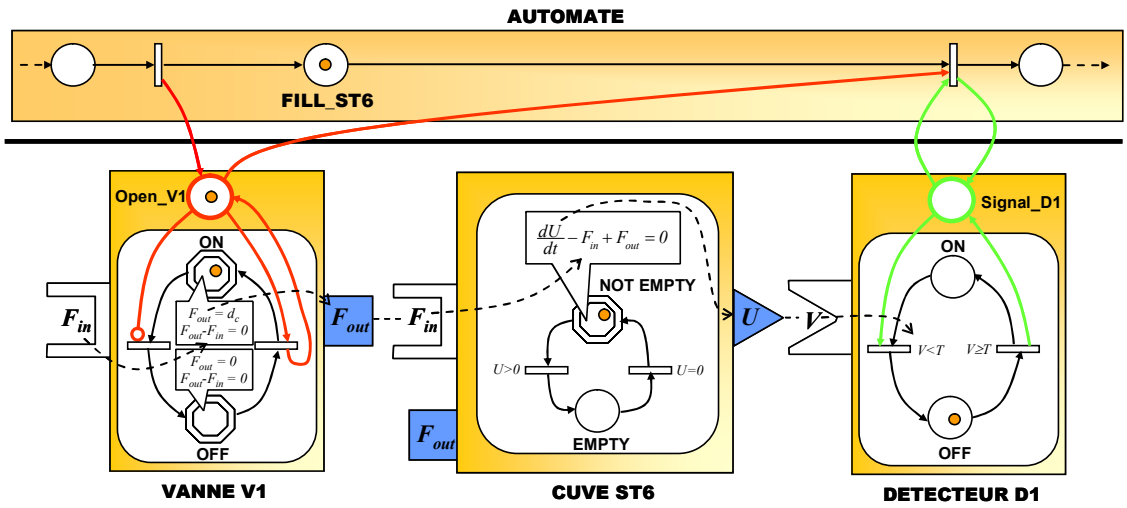


Figure 2.27 Modélisation du système

L'opération de remplissage FILL_ST6 *de la cuve est déclenchée par le RdP recette qui envoie un jeton sur la place signal de l'objet* Vanne. *L'alimentation est maintenue ouverte tant que cette place de commande reste marquée. L'intégration du système d'équations résultant provoque l'évolution de la variable de rétention* U *de la cuve. Transmise à l'objet* Détecteur *via le port d'information, cette variable valide une transition de cet objet lorsque la valeur seuil est atteinte. Cette commutation provoque alors le marquage de la place signal* Signal_D1 *de l'objet* Détecteur, *indiquant au RdP recette que le volume de produit a atteint la valeur cible. La transition qui succède à la place* FILL_ST6 *est validée et tirée. L'absence de jeton sur la place signal* Open_V1 *provoque alors la fermeture de la vanne.*

L'exemple précédent montre aussi que le marquage d'une place signal (place de commande ou d'information) d'un dispositif actif induit en général l'évolution de son *RdP*, lui-même provoquant éventuellement en cascade l'évolution des dispositifs passifs au travers du réseau constitué par la connexion des différents ports matière ou énergie.

Par souci de clarté, nous utiliserons dans la suite de ce manuscrit une vision réduite du *RdPDO* global du modèle de simulation. En effet, comme ces travaux s'intéressent essentiellement au niveau *commande*, nous ne ferons apparaître que les places *signal* des appareils et non leurs *RdPDO* complets puisque du point de vue de la commande, seules ces places sont significatives.

Exemple *:*

Si on ne s'intéresse qu'au niveau commande du modèle de simulation, le réseau de Petri de la Figure 2.27 peut être représenté tel que le montre la Figure 2.28 :

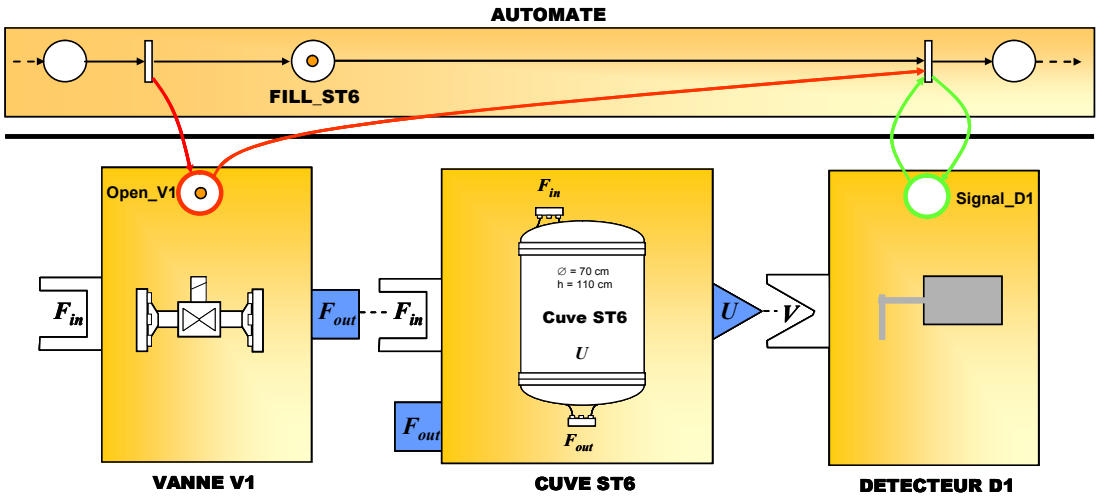


Figure 2.28 Vision simplifiée du RdPDO du modèle de simulation (niveau commande)

Ce schéma peut encore être simplifié en ne faisant apparaître que les dispositifs actifs du système. On obtient dans ce cas le réseau de Petri de la Figure 2.29.

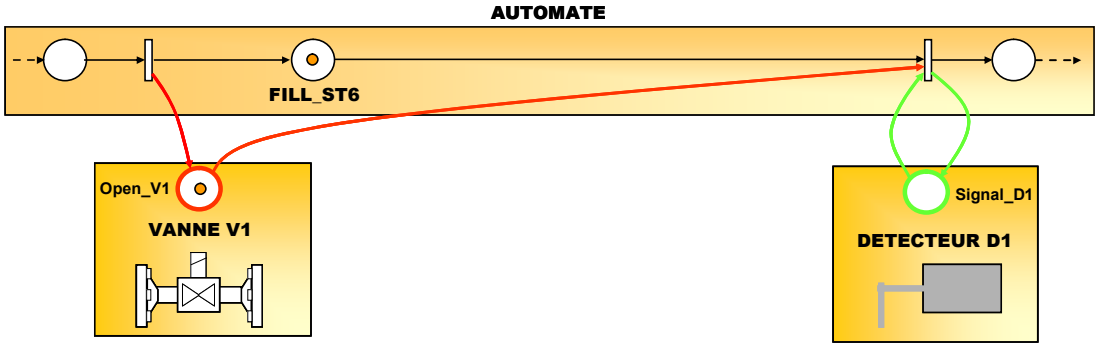


Figure 2.29 Vision simplifiée du RdPDO du modèle de simulation (niveau commande)

2.4. EXEMPLE D'APPLICATION

Afin d'illustrer les potentialités de *PrODHyS* mais aussi d'identifier ses limites actuelles, une application est traitée dans cette section. Il s'agit d'une opération de réaction que nous avons volontairement voulue très simple. Par ailleurs, nous nous focalisons sur le niveau commande du modèle de simulation, le modèle des différents appareils de ce procédé pouvant être trouvé dans [Perret, 2003].

2.4.1. Caractéristiques du procédé considéré

Le procédé considéré dans cet exemple est décrit en s'appuyant sur la structuration de recette proposée par la norme *ISA SP88* (cf. section 1.2.5).

- **Formule**

Le but est de fabriquer un produit *P* par une opération de réaction. Il s'agit d'une réaction équilibrée endothermique impliquant deux réactifs *R1* et *R2* et un produit *P* :

$$(R): \; R1 + R2 \longleftrightarrow P$$

Elle suit une loi cinétique d'ordre 2 de la forme :

$$r_R = k_1^{\circ}.e^{-\frac{Ea_1}{RT}}.x_{R1}.x_{R2} - k_{-1}^{\circ}.e^{-\frac{Ea_{-1}}{RT}}.x_P \quad \text{avec :} \quad \begin{array}{ll} k^{\circ}_{1} = 3.10^{9} & E_{a1} = 21300 \\ k^{\circ}_{-1} = 10^{6} & E_{a-1} = 18766 \end{array}$$

Comme le montre la Figure 2.31, la fraction molaire de produit *P* ne peut excéder 0,84 et un temps de séjour infini serait nécessaire pour atteindre cette composition. Afin de maximiser le taux de conversion de la réaction sans pour autant pénaliser le temps de cycle du procédé, la réaction est stoppée dés que la composition en produit *P* atteint la valeur de 0,8. La Figure 2.30 montre les proportions massiques de chaque état dans cette opération.

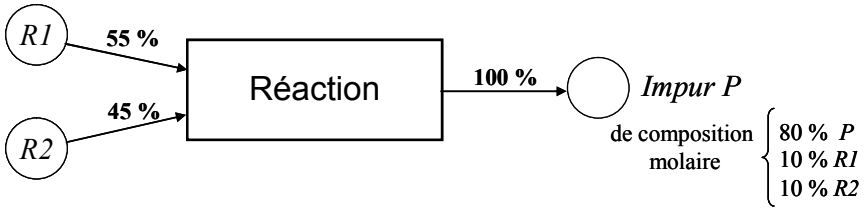


Figure 2.30 Caractéristiques de l'opération de réaction

Par ailleurs, compte tenu de la loi de vitesse considérée, la réaction *(R)* est d'autant plus rapide que la température *T* est élevée. La température retenue pour la réaction doit garantir une réaction rapide tout en maintenant les constituants dans l'état liquide. Une température de 372 K permet de satisfaire ces deux contraintes.

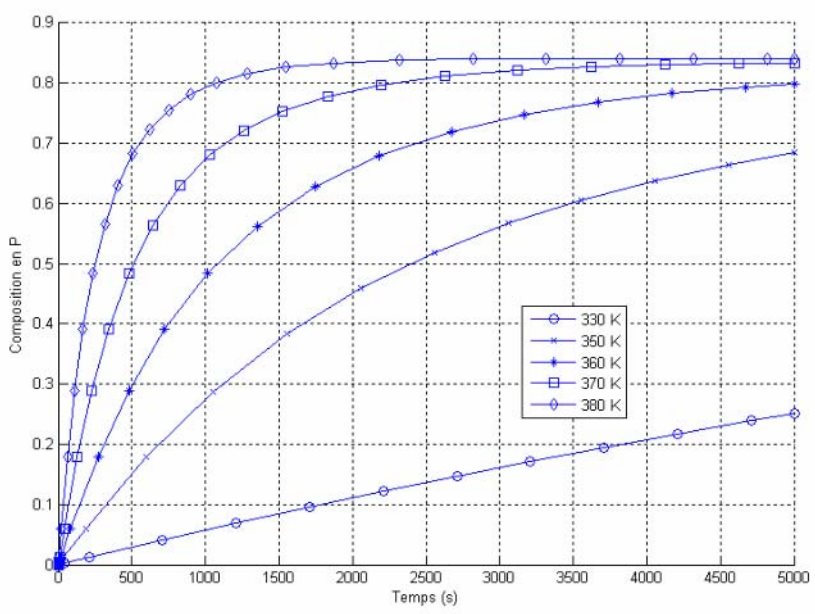


Figure 2.31 Vitesse de réaction en fonction de la température

- **Equipements disponibles**

La topologie du procédé mis en œuvre est montrée sur la Figure 2.32. Il comporte deux cuves d'alimentation en produits *R1 (cuve ST1)* et *R2 (cuve ST2)*, un réacteur *BR1* et une cuve de stockage du produit de la réaction *(cuve ST5)* en sortie de *BR1*. L'ensemble des transferts est effectué via le même type de vannes et par gravité. De plus, le procédé est instrumenté de plusieurs capteurs de niveau et de température.

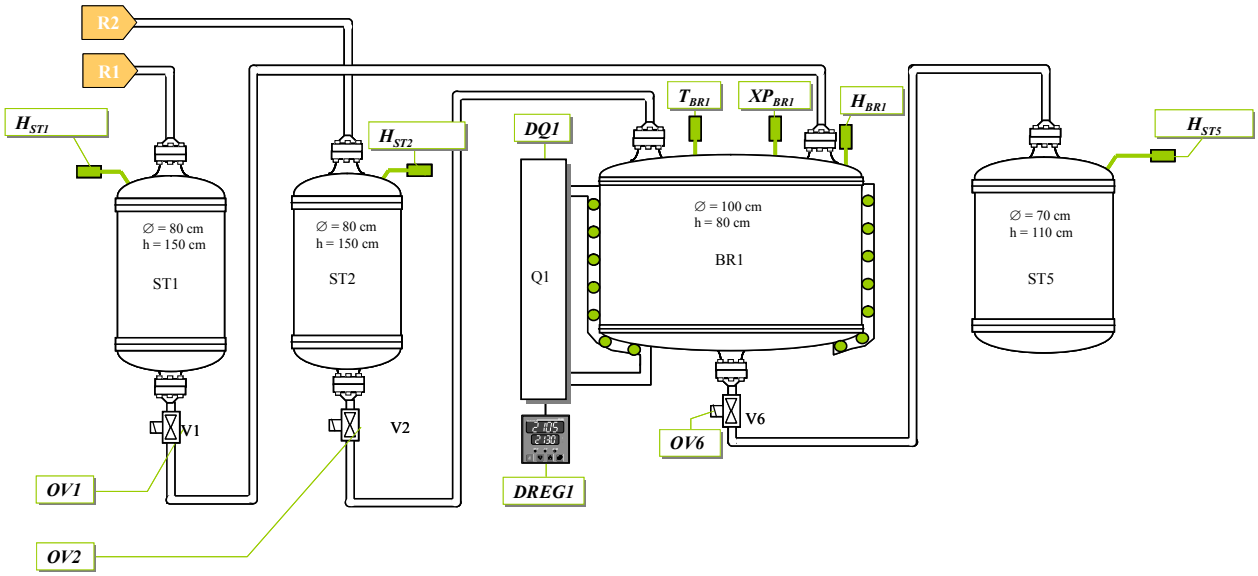


Figure 2.32 topologie du procédé mis en œuvre

Les dimensions de chaque appareil sont indiquées sur la Figure 2.32. L'élévation de chaque cuve est répertoriée dans le Tableau 2.1.

Élévation Cuve **ST1** (e_{ST1})	250 cm
Élévation Cuve **ST2** (e_{ST2})	250 cm
Élévation Réacteur **BR1**(e_{BR1})	150 cm
Élévation Cuve **ST5** (e_{ST5})	30 cm
Section Conduite ∅**V1** = ∅**V2**	0,3 cm^2
Masse molaire **R1**	56 $g.mol^{-1}$
Masse molaire **R2**	46 $g.mol^{-1}$
Masse molaire **P**	102 $g.mol^{-1}$

Tableau 2.1 Les données techniques de l'exemple

- **Procédure**

La fabrication de *n* moles de produit *P* se déroule en 5 phases principales :

- le chargement du réacteur avec *n*x*(9/8)* moles de produit *R1*,
- le préchauffage à 372 K,
- le chargement du réacteur de *n*x*(9/8)* moles de produit *R2* tout en régulant la température à 372 K,
- la réaction jusqu'à ce que la composition en produit *P* atteigne la valeur 0.8
- et la vidange totale ou partielle du mélange réactionnel.

2.4.2. Modélisation du procédé

La Figure 2.33 montre la structure objet mise en œuvre pour modéliser le procédé décrit dans la section précédente.

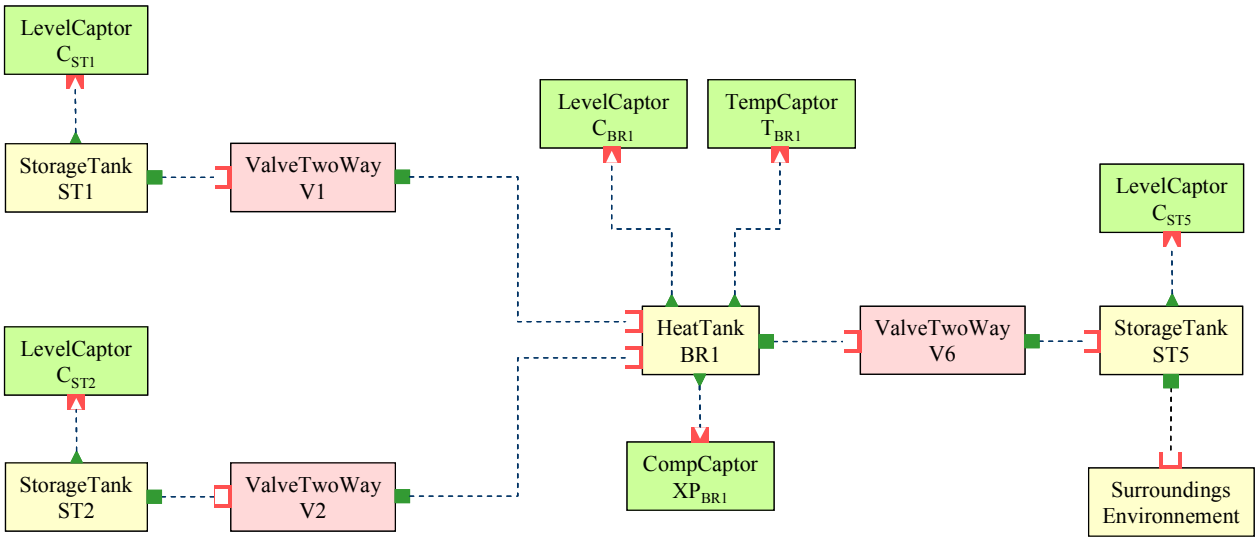


Figure 2.33 Modélisation du procédé sous PrODHyS

Le schéma bloc donne pour chaque élément, son type et la classe d'objet qui le décrit. Celui-ci comporte :

- trois instances de type `StorageTank` pour représenter les cuves *ST1*, *ST2* et *ST5*,
- une instance de type `HeatTank` qui modélise le réacteur *BR1*,
- trois instances de type `ValveTwoWay` pour représenter les trois vannes *V1*, *V2* et *V6*,
- un ensemble d'instances de capteurs de composition (de type `CompCaptor`), de niveau (de type `LevelCaptor`) et de température (de type `TempDetector`),
- une instance de type `Surroundings` qui a un objet virtuel auquel sont connectés tous les ports de sortie non connectés.

Ce modèle est évidemment constitué de différents dispositifs actifs et passifs. Comme le montre la Figure 2.34, les détecteurs et les vannes sont des dispositifs actifs possédant des places *signal* : place de commande mOpen pour les vannes et place d'information mOn pour les détecteurs. Seules ces places apparaissent au niveau du *RdPDO* de la recette. Ainsi, si une instance de type `ValveTwoWay` est nommée V1, la place de commande de cette instance est nommée V1.mOpen au niveau du *RdP* de la recette.

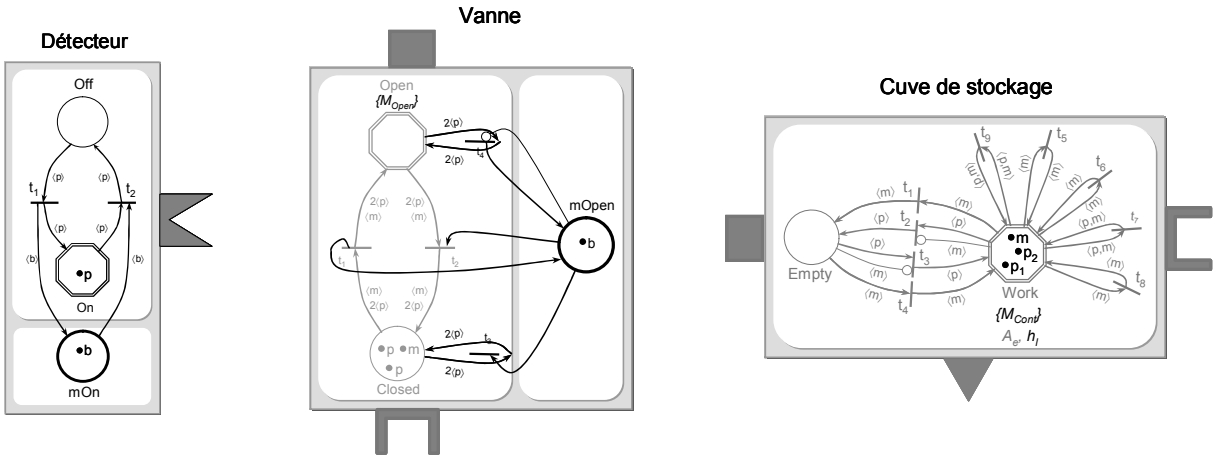


Figure 2.34 Modèle (simplifié) des objets Détecteur, Vanne *et* Cuve de stockage

L'objet *détecteur* est conçu pour suivre une grandeur physique du procédé. Plus exactement, cette grandeur (transmise par référence) est comparée (=, ≤, ≥, ≠, etc) par rapport à une valeur seuil spécifiée au moment de la construction de l'objet. Ainsi, la place *signal* mOn est marquée lorsque cette condition est satisfaite et est non marquée dans le cas contraire.

L'objet *vanne* est Tout Ou Rien (*TOR*) et possède donc deux états : ouvert ou fermé. Le marquage de la place mOpen induit l'ouverture de la vanne et l'existence d'un débit (fixe ou dépendant de la pression aux bornes de la vanne), sinon celle-ci est fermée.

Dispositif passif, l'objet *cuve de stockage* est un volume de contrôle particulier caractérisé par une forme géométrique spécifique et une hauteur de liquide. La hauteur de liquide permet d'une part, de calculer l'aire d'échange entre la cuve et l'extérieur et d'autre part, de déterminer les pressions hydrauliques associées aux ports hydrauliques du bac. Les variables concernant les propriétés qualitatives et quantitatives des phases sont portées par les jetons Phase ; les variables de *pression* et *température* sont portées par le jeton Matière ; les variables de *flux* et éventuellement de *pression* et *d'énergie hydraulique* sont portées par les jetons Port. Enfin, les variables décrivant les pertes thermiques éventuelles et l'aire d'échange sont spécifiques à l'objet. Enfin, l'état vide et occupé de l'appareil constitue une information intéressante en conduite de la production où le problème de l'occupation

des ressources est important. C'est pourquoi les places Empty et Work sont créées afin de représenter respectivement l'état *vide* et l'état *occupé* de la ressource.

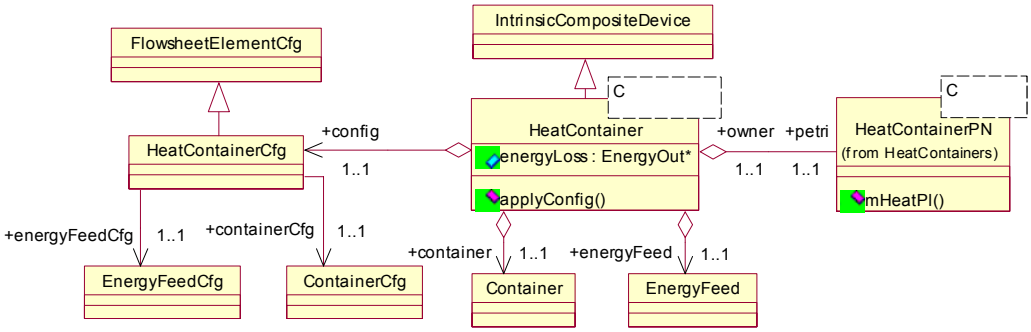


Figure 2.35 : Les bacs composés

L'objet *réacteur* équipé de son système thermique est un appareil composé intrinsèque qui associe un objet *bac* avec un objet *alimentation énergie* (cf. Figure 2.35). Celui-ci est donc responsable de sa configuration et de la construction des appareils qui le constituent.

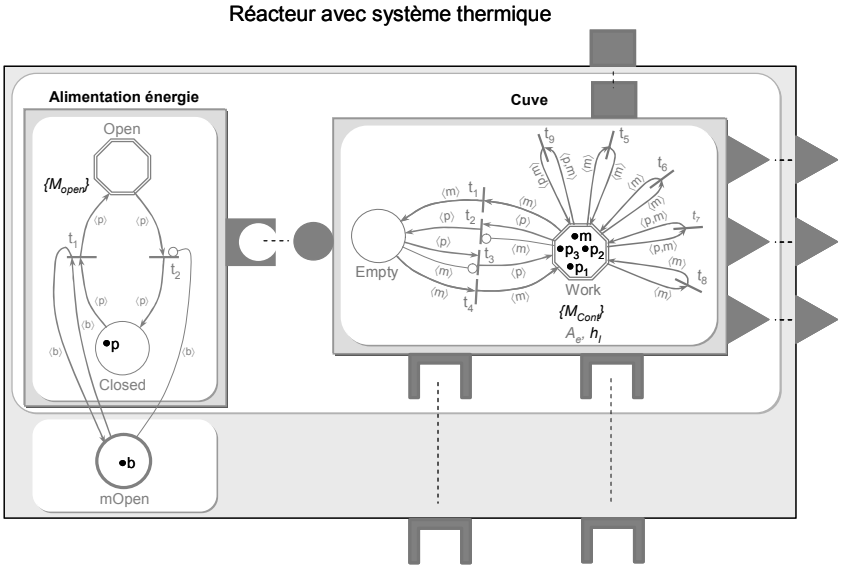


Figure 2.36 Modèle (simplifié) de l'objet Réacteur

La Figure 2.36 montre le modèle de l'objet *réacteur*. Celui-ci ne définit pas d'élément particulier, il regroupe simplement les *RdP* des appareils le constituant. Bien que similaire à un objet *cuve*, il s'agit d'un dispositif actif du fait de la présence de la place de commande mOpen de l'alimentation énergie.

2.4.3. Modélisation de la recette

La section 1.2.5 a montré que la recette est hiérarchisée en quatre niveaux. Dans le cadre de la simulation, la procédure à exécuter est décrite au niveau de la *recette de contrôle*. Cette dernière est une instanciation de la recette principale pour un lot donné.

Dans le cadre de cet exemple, on souhaite qu'une quantité OF_P = 550 moles de *P* soit fabriquée. Le Tableau 2.2 indique par ailleurs les conditions initiales dans chaque cuve :

ST1	U_0 = 2000 moles de *R1*
ST2	U_0 = 1500 moles de *R2*
BR1	U_0 = 5 moles de *P impur*
	x_P = 0.8, x_{R1} = 0.1, x_{R2} = 0.1
ST5	U_0 = 10 moles de *P impur*
	x_P = 0.8, x_{R1} = 0.1, x_{R2} = 0.1

Tableau 2.2 Conditions initiales dans chaque cuve

La température initiale dans chaque cuve est de T_0 = 293 K. Enfin, la cuve *ST5* qui sert de cuve tampon avec une autre partie du procédé possède une capacité de *HMAX* = 8000 moles.

La procédure générique a été établie dans la section 2.4.1. Celle-ci se décline en la séquence opératoire suivante :

- La première étape consiste à ouvrir la vanne *V1* afin d'alimenter en réactif *R1* le réacteur *BR1*. Cette opération est maintenu jusqu'à atteindre une rétention *VLOT* = 618,75 moles dans *BR1*.
- Une phase de chauffe du contenu de *BR1* est alors lancée jusqu'à ce que celui-ci atteigne la température *TREAC* = 372K.
- L'ouverture de la vanne *V2* permet alors d'alimenter le réacteur *BR1* de *VLOT* = 618,75 moles de réactif *R2*, provoquant le démarrage immédiat de la réaction contrôlée en température.
- Enfin, l'obtention d'une concentration en produit *P* de *XP* = 0.8 dans le réacteur *BR1* déclenche la vidange complète de ce dernier par l'ouverture de la vanne *V6* (*HMIN* = 5 moles).

En supposant que l'opération commence immédiatement, la modélisation de cette séquence opératoire conduit au *RdPDO* montré sur la Figure 2.38.

2.4.4. Simulation du procédé

Les paramètres *VLOT*, *TREAC*, *XP*, *HMIN* et *HMAX* ayant été fixés au moment de la construction du *RdPDO* de la recette, la simulation de ce modèle peut être lancée. Les figures Figure 2.37, Figure 2.39 et Figure 2.40 montrent quelques résultats de la simulation.

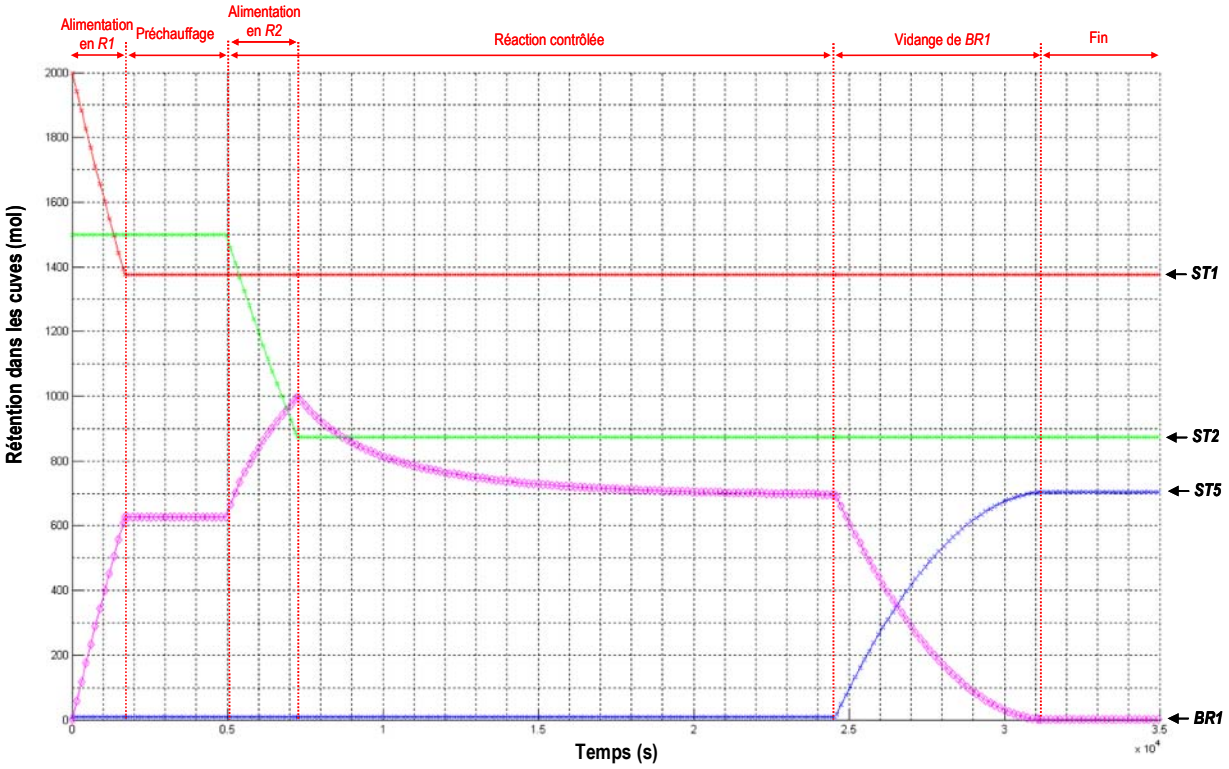


Figure 2.37 Evolution de la rétention molaire dans les différentes cuves

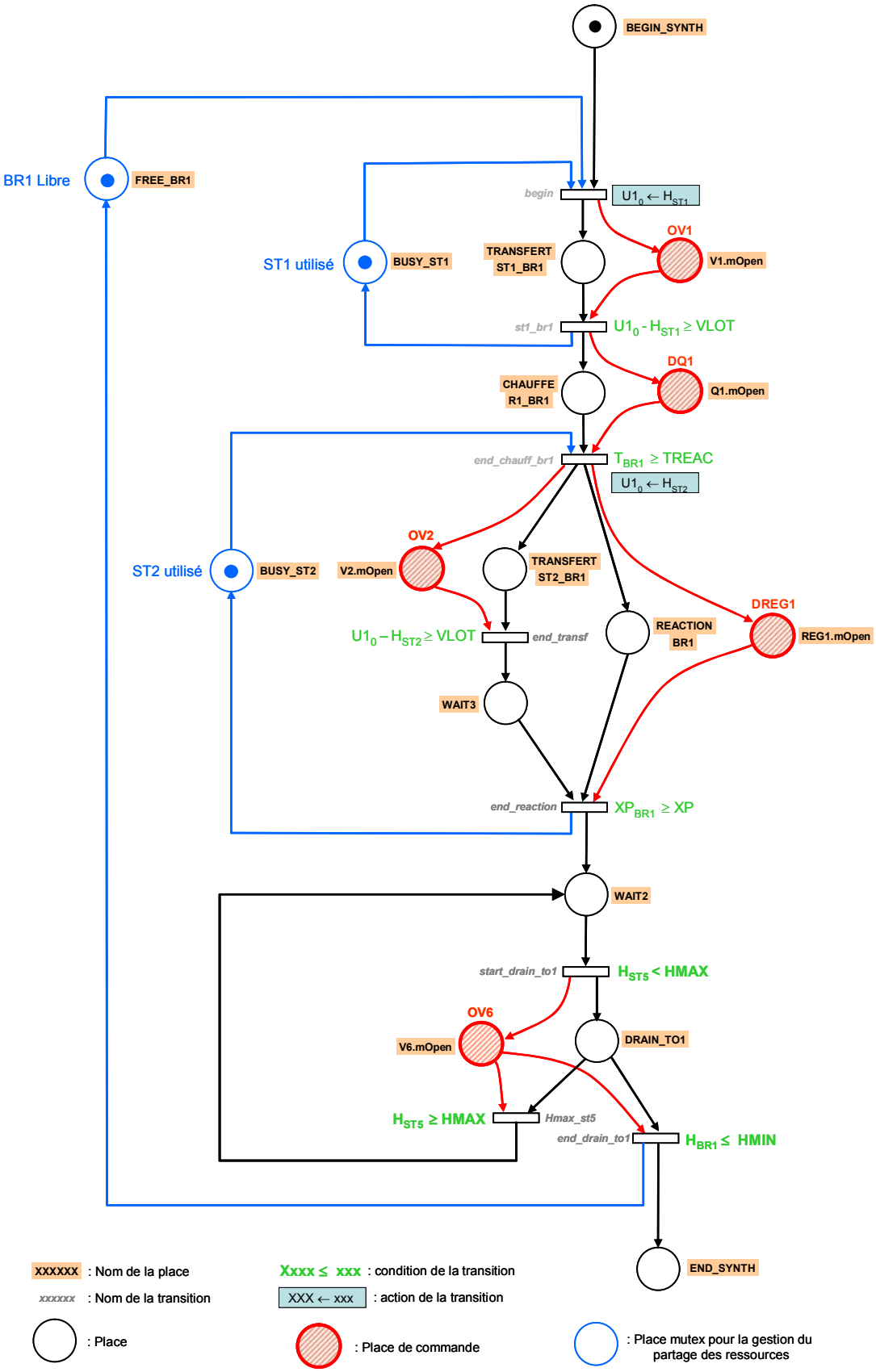


Figure 2.38 Procédure de l'opération de réaction

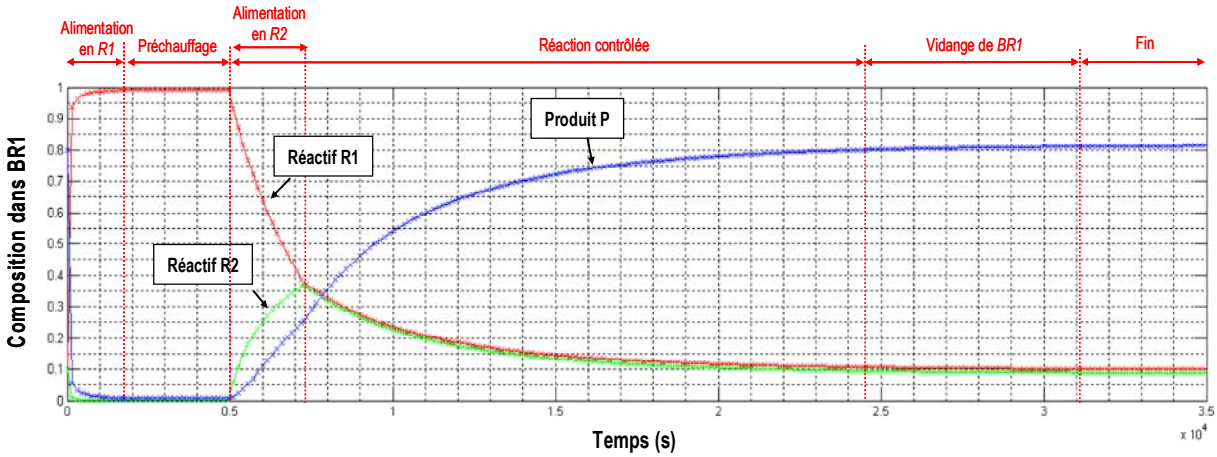


Figure 2.39 Evolution de la composition dans le réacteur BR1

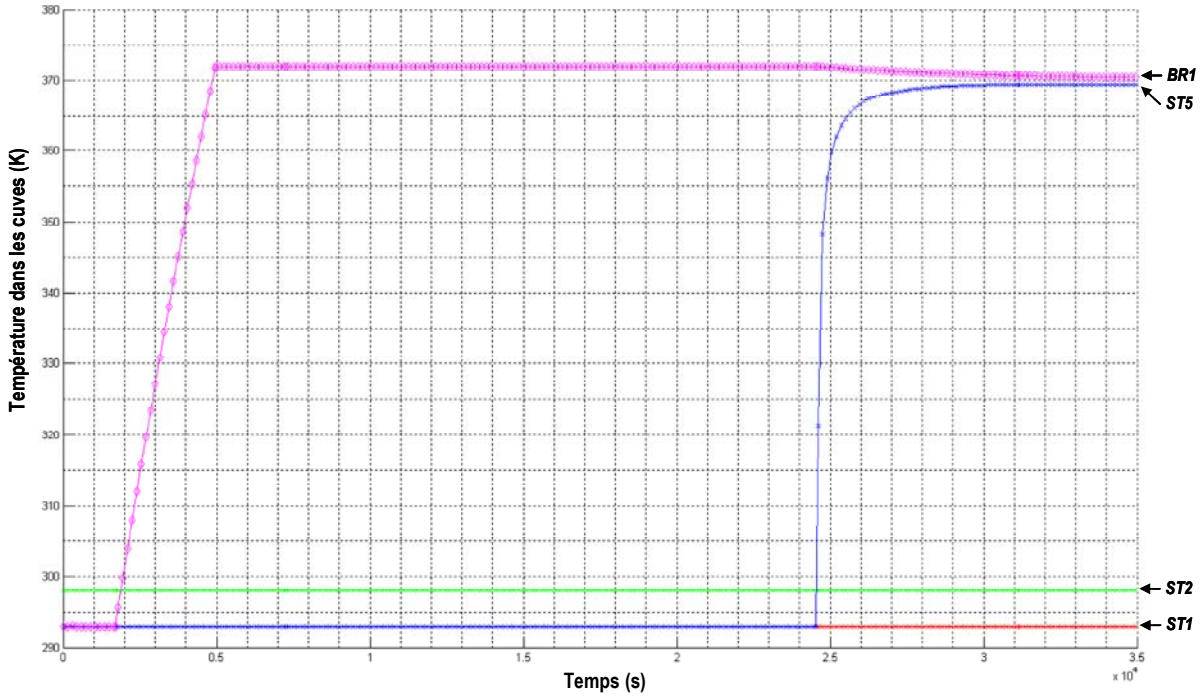


Figure 2.40 Evolution de la température dans les différentes cuves

Comme le montre la Figure 2.37, la réaction démarre dès l'introduction du deuxième réactif. Une partie des réactifs est convertie durant la phase d'alimentation en *R2*, ce qui explique que la rétention dans la cuve n'atteint jamais 2x618.75 = 1237.5 mol.

Cette exemple montre qu'il est possible de simuler de manière complète la production d'un lot (quelque soit l'opération unitaire mise en œuvre). La production de *n* lots de taille identique et d'un même produit peut facilement être prise en marquant avec *n* jetons la place BEGIN_SYNTH du *RdPDO*. Comme le lancement d'un lot est conditionné à la disponibilité du réacteur (place FREE_BR1), chaque lot est lancé en séquence dès que le lot précédent s'est achevé.

De même, si on souhaite fabriquer *n* lots identiques en introduisant un délai de lancement spécifique pour chaque lot, ceci peut être réalisé au moyen d'un *RdPDO* ayant une structure analogue à celui de la Figure 2.41(construit ici pour *n* = 2 lots)

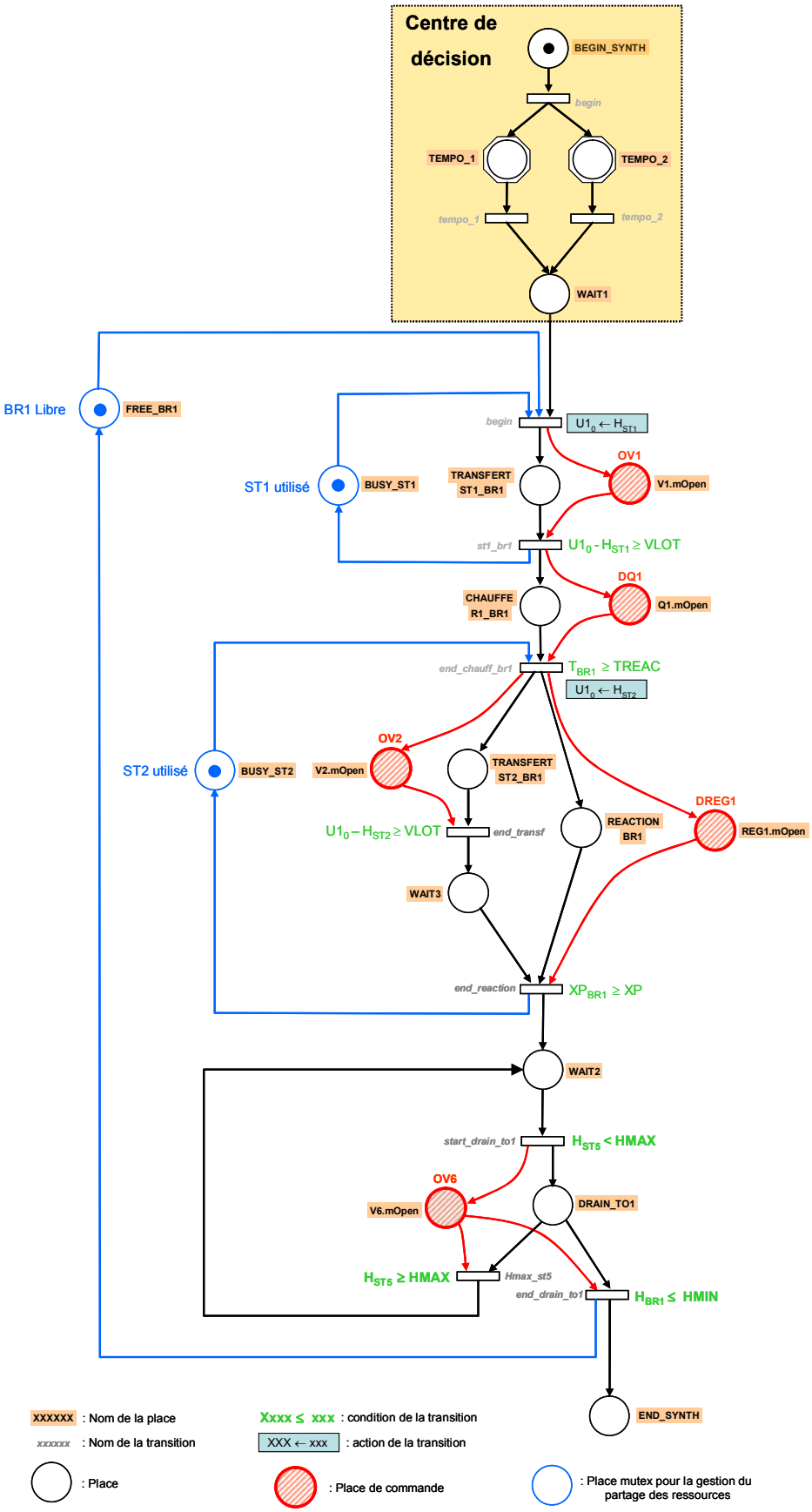


Figure 2.41 Gestion du lancement de plusieurs lots identiques dans BR1

Une place temporisée est associée à chaque lot, la temporisation étant égale à la date de lancement du lot. Par conséquent, pour *n* lots, il faut introduire *n* places temporisées. La Figure 2.42 montre la réalisation de deux lots identiques (OF_1 = OF_2 = 550 moles de *P*) lancés successivement aux dates t_1 = 1000 s et t_2 = 40000 s.

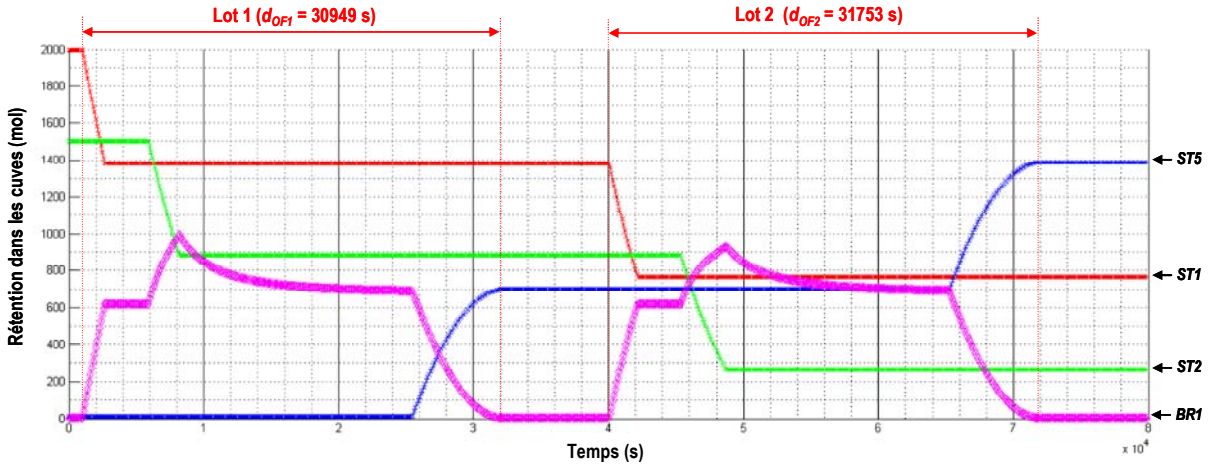


Figure 2.42 Evolution de la rétention molaire dans les différentes cuves

Evidemment, si la date de lancement de l'OF_2 avait été de t_2=25000 s (c'est-à-dire, avant la fin du l'OF_1), l'OF_2 aurait été lancé juste après la fin de l'OF_1, c'est-à-dire dès que le réacteur *BR1* ait été à nouveau libre. On peut noter aussi un allongement de la durée de l'opération, bien que les deux *OFs* soient totalement identiques. Ce constat est encore plus net sur la simulation de la Figure 2.43.

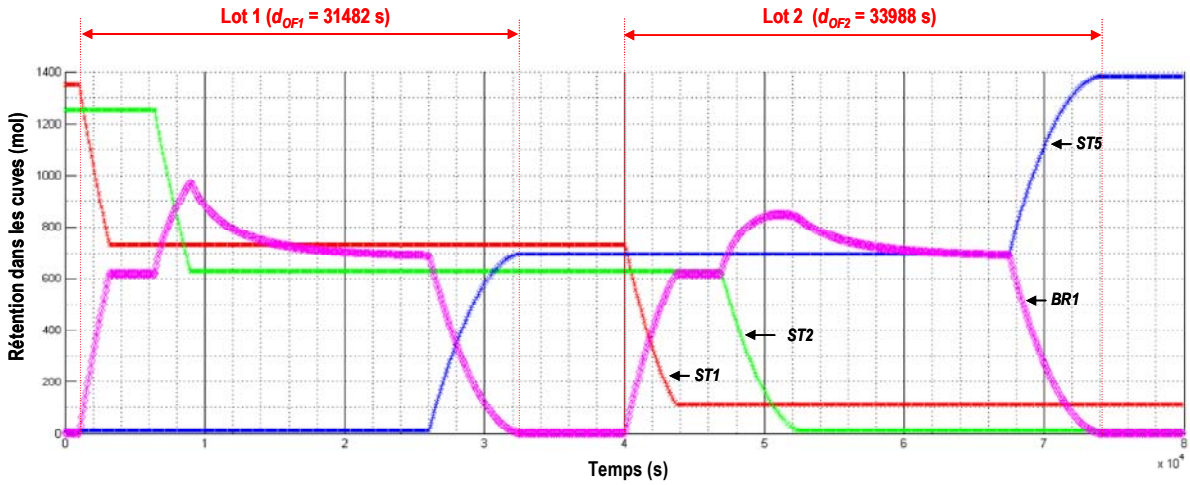


Figure 2.43 Evolution de la rétention molaire dans les différentes cuves

Il s'agit toujours de la production des deux mêmes *OFs* mais les rétentions initiales dans les cuves *ST1* et *ST2* sont respectivement de $U0_{ST1}$ = 1350 mol et $U0_{ST2}$ = 1250 mol. On constate un écart de durée Δd_{OF} = 2506 s entre ces deux *OFs* et un écart de Δd_{OF} = 3039 s entre l'OF_1 de Figure 2.42 et l'OF_2 de la Figure 2.43. Les transferts étant réalisés par gravité, on voit ici l'influence de la pression hydraulique sur les débits d'alimentation des réactifs *R1* et *R2*. Ce phénomène fait partie des

nombreuses situations pouvant entraîner une différence notable entre un plan de production établi par optimisation (où ce phénomène n'est généralement pas pris en compte) et un plan de production établi par simulation (où le modèle est plus proche du système réel). Ceci renforce aussi le fait qu'une solution optimale « mathématiquement », souvent obtenue avec un effort de calcul important, n'a pas réellement d'intérêt dans la pratique car cette solution est rarement utilisable sans ajustement.

2.5. VERROUS ACTUELS DE LA SIMULATION

Dans l'état actuel du simulateur *PrODHyS*, il est possible de simuler les différentes phases d'une opération unitaire. Il est aussi possible de gérer aisément l'enchaînement de plusieurs lots identiques via l'utilisation de places temporisées (pour les dates de lancement) et/ou de places *mutex* (pour la gestion du partage des ressources).

Néanmoins, comme dans beaucoup de simulateurs dynamiques, la mise en œuvre devient plus délicate dès que plusieurs lots aux caractéristiques différentes doivent subir successivement plusieurs opérations unitaires sur un procédé constitué de plusieurs appareils distincts. Plusieurs verrous apparaissent à ce niveau :

- D'abord, dans la description des phases d'une opération unitaire, on peut distinguer deux types d'information. Les données telles que température de réaction sont propres à l'opération, au contraire de la taille du lot qui dépend de la tâche à réaliser. Or, ces informations (utilisées notamment pour franchir des transitions) sont figées sur le *RdPDO* de la recette. Il faudrait donc que certaines de ces données (comme la taille du lot) deviennent des paramètres attachés au jeton. L'utilisation de simple jeton binaire qui identifie l'état d'avancement de la procédure n'est plus suffisante et l'exploitation de jeton objet apparaît donc nécessaire.

- L'enchaînement de plusieurs opérations unitaires dans un même procédé complexifie considérablement la description du *RdDPO* de la recette. En effet, si la *recette de contrôle* est adaptée pour la conduite de la simulation, la gestion plus globale des opérations nécessite de raisonner plutôt à un niveau *recette principale*. Il apparaît alors souhaitable de hiérarchiser la construction du *RdPDO* utile à la simulation en introduisant les notions d'*opérations* et de *phases* comme des objets paramétrables et composables. La mise en œuvre de cet aspect suppose la mise en place de nouveaux éléments sémantiques dans les *RdPDO* pour représenter des entités agrégées.

- dans les exemples traités à la section 2.4, la taille des lots et les dates de lancement sont des données du modèle de simulation. Or, dans un contexte plus général, ces informations ne sont pas directement disponibles. En effet, la taille et le nombre de lots sont calculés en fonction des *OFs* à réaliser et des caractéristiques physiques des moyens de production. Les dates de lancement résultent elles-mêmes d'un calcul d'ordonnancement qui effectue l'affectation des divers lots sur les différents appareils. La vision myope de la simulation et le lancement « au plus tôt » des lots ne permettent en aucun cas d'évaluer efficacement ce type de données, notamment si des contraintes temporelles (dates de livraison, délais d'attente maximum, etc.) doivent être prises en compte. Cela peut alors conduire à des dysfonctionnements du type débordement de cuve ou rupture d'alimentation qui pénalisent fortement les performances du système de production. Encore une fois, il apparaît nécessaire d'introduire dans *PrODHyS* un prétraitement des plans de production (liste d'*OFs* et dates de livraisons) de façon à obtenir les données nécessaires à la simulation (nombre, taille et date de lancement des lots). Ce prétraitement (ou « preprocessing ») consiste finalement à résoudre un problème d'ordonnancement.

- l'obtention des données de base identifiées au point précédent permet d'établir un plan conducteur et un calendrier d'événement pour la simulation. Néanmoins, comme l'a montré l'exemple précédent, même l'obtention de ces données peut ne pas être suffisante pour mener à terme une simulation. En effet, l'écart existant entre le modèle utilisé pour l'ordonnancement et celui utilisé dans *PrODHyS* peut générer des indéterminations à certains points critiques de la simulation. Dans ces conditions, des décisions locales doivent être prises, introduisant alors la notion de *centre de décisions* tel que le montre la Figure 2.44.

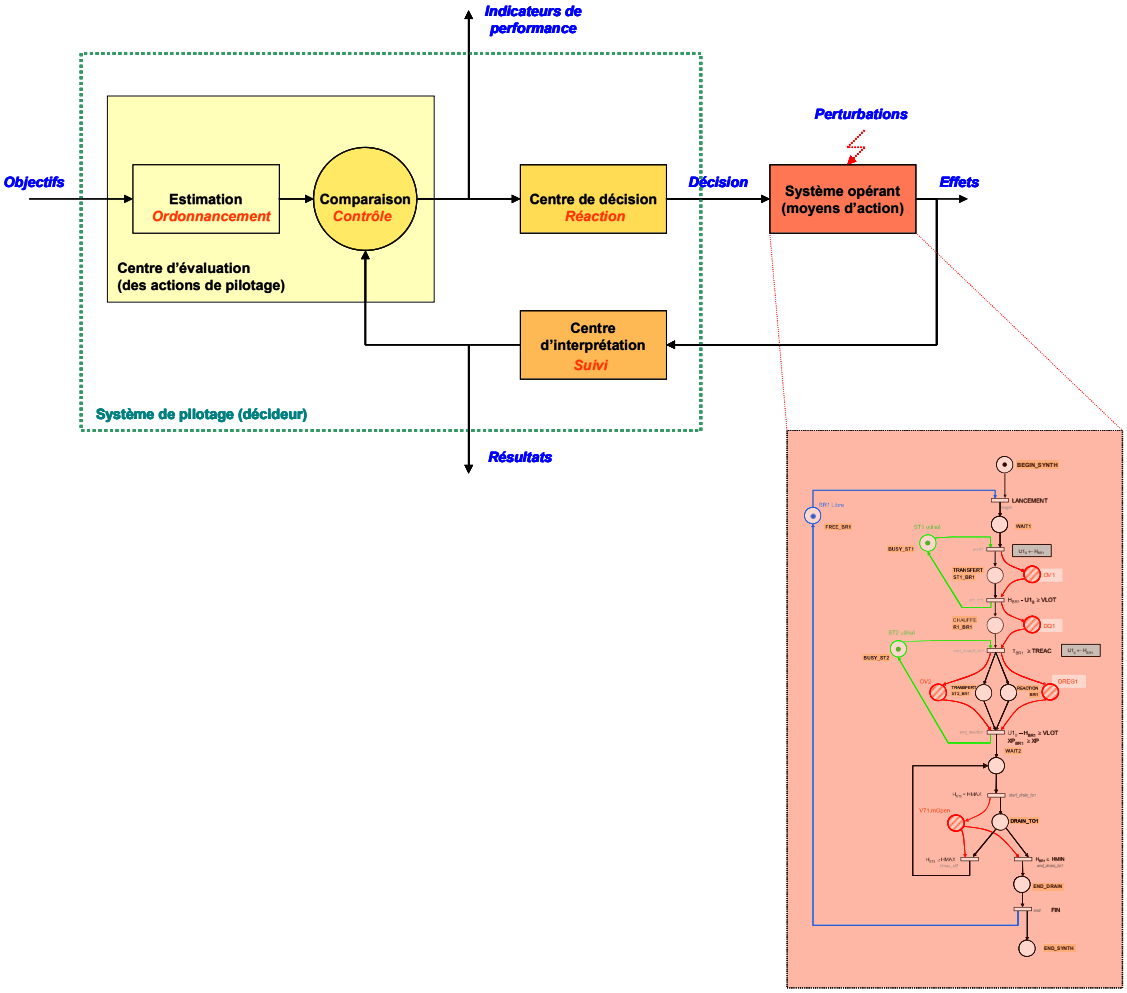


Figure 2.44 Système de pilotage décisionnel sous PrODHyS

Les travaux présentés dans ce manuscrit visent à lever ces différents verrous. Pour cela, une structuration de la partie commande du simulateur *PrODHyS* est proposée. Sa mise en œuvre nécessite :

- de proposer une formalisation graphique permettant de représenter sans ambiguïtés et de manière systématique la *recette principale*,
- d'en déduire un modèle générique d'ordonnancement basé sur une approche par programmation linéaire en variable mixte,
- d'étendre le formalisme *RdPDO* afin de représenter de manière hiérarchisée la *recette de contrôle* des systèmes complexes et de pouvoir la générer de manière automatique,
- d'intégrer un outil de pilotage décisionnel au simulateur *PrODHyS* pour l'exécution des simulations.

CHAPITRE 3

CHAPITRE 3

METHODES ET OUTILS D'ORDONNACEMENT

Résumé

Les problèmes d'ordonnancement constituent un thème d'étude important de la recherche opérationnelle. Fonction complexe de la gestion de la production, les ouvrages et les articles traitant de cette problématique sont nombreux. Dans ce contexte, ce chapitre vise tout d'abord à rappeler les définitions et concepts généraux relatifs aux problèmes d'ordonnancement. Il propose ensuite un état de l'art des méthodes et outils d'ordonnancement généraux et permet enfin d'identifier la solution retenue dans ces travaux : la Programmation Linéaire en Variables Mixtes

Les problèmes d'ordonnancement constituent un thème d'étude important de la recherche opérationnelle. Fonction complexe de la gestion de la production, les ouvrages et les articles traitant de cette problématique sont nombreux. Dans ce contexte, ce chapitre rappelle les définitions et concepts généraux relatifs aux problèmes d'ordonnancement. Puis, une revue des méthodes de résolution est réalisée. Pour finir, l'approche retenue dans le cadre de notre étude est indiquée.

3.1. NOTION D'ORDONNANCEMENT

3.1.1. Définitions générales

De façon générale, un *problème d'ordonnancement* se définit ainsi : organiser dans le temps la réalisation d'un ensemble de tâches, compte tenu des contraintes temporelles (délais, contraintes d'enchaînement, etc.) et de contraintes portant sur l'utilisation et la disponibilité des ressources requises par les tâches [Baker 1974, Bellmann et al. 1982, Carlier et al. 1988, Conway et al. 1967, Esquirol et al. 1999, French 1982, GOTHA 1993, Pinedo 1995].

Un *ordonnancement* (souvent appelé *planning d'atelier*) constitue une solution au problème d'ordonnancement. Il décrit l'exécution des tâches et l'allocation des ressources au cours du temps, et vise à satisfaire un ou plusieurs objectifs. Plus précisément, on parle de problème d'ordonnancement lorsqu'on doit déterminer les dates de début et les dates de fin des tâches, alors qu'on réserve le terme de problème de séquencement au cas où l'on cherche seulement à fixer un ordre relatif entre les tâches qui peuvent être en conflit pour l'utilisation des ressources.

Enfin, le problème d'ordonnancement peut être vu comme un problème d'optimisation combinatoire, auquel il faut trouver une bonne solution, éventuellement optimale au regard d'un ou de plusieurs critères d'évaluation.

3.1.2. Éléments d'un problème d'ordonnancement

D'après la définition précédente, un ordonnancement est constitué principalement des quatre éléments suivants : les *tâches*, les *ressources*, les *contraintes* et les *objectifs* ou *critères* d'optimisation. De plus, la description de la décomposition en tâches constitue en général une *gamme de fabrication*, encore appelée *procédure* dans l'industrie des procédés.

3.1.2.1. Les tâches

Une *tâche* i est une entité élémentaire de travail localisée dans le temps (cf. Figure 3.1) par une *date de début* t_i et une *date de fin* c_i ou par l'une de ces dates et une *durée* d_i. Un ordre de fabrication décomposable en tâches peut avoir une *date de livraison* souhaitée l_i (correspondant généralement aux jalons fixés par la planification).

Une tâche utilise une (ou plusieurs) ressource k avec une intensité a_i^k, souvent supposée constante pendant l'exécution de la tâche.

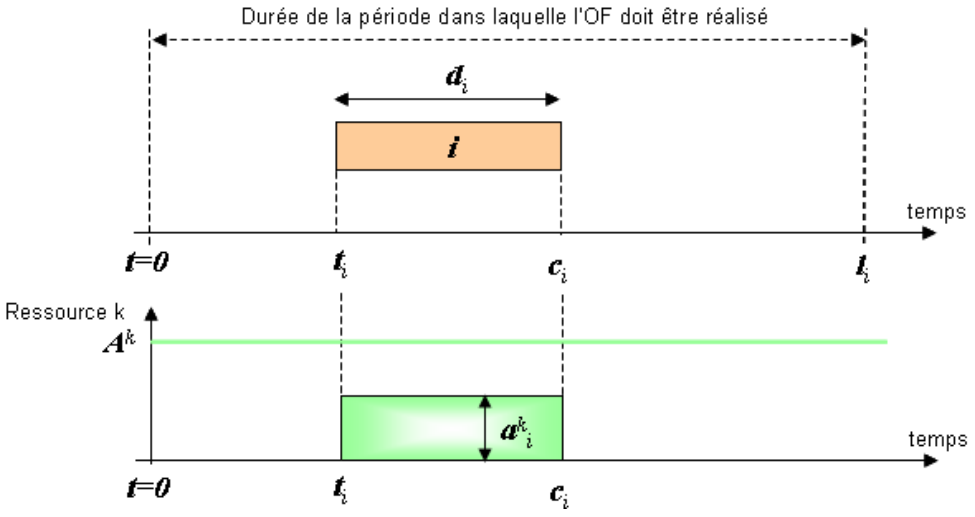


Figure 3.1 Tache et ressource

3.1.2.2. Les ressources

Une ressource k est un moyen technique ou humain requis pour la réalisation d'une tâche, et disponible en quantité limitée : sa capacité A^k.

Une ressource est *renouvelable* si, lorsqu'elle est libérée par une ou plusieurs tâches, elle est à nouveau disponible en même quantité (les hommes, les machines, l'espace, l'équipement en général). Dans le cas contraire, elle est *consommable* (matières premières, budget, énergie etc.).

On distingue les ressources *disjonctives* ou non partageables (machine outil) qui ne peuvent exécuter qu'une tâche à la fois et les ressources *cumulatives* ou partageables (équipe d'ouvriers).

Le *temps de préparation* d'une ressource afin d'exécuter une tâche peut dépendre ou non de la tâche effectuée précédemment. Enfin, les durées des tâches ne sont pas toujours connues mais peuvent être fonction de la quantité de moyens utilisés pour leur exécution, elle-même liée à la vitesse ou la performance de la ressource.

3.1.2.3. Les contraintes

Les contraintes expriment des restrictions sur les valeurs que peuvent prendre conjointement les variables de décision. Leur prise en compte permet d'avoir un ordonnancement réalisable. Deux classifications des contraintes existent : la première est classique, elle comporte les contraintes temporelles et celles liées aux ressources. La seconde présentée par Kacem [Kacem 2003] distingue les contraintes endogènes qui sont liées directement au système de production et à ses performances, des contraintes exogènes indépendantes du système.

Nous considérons la première classification pour décrire les différents types de contraintes qui conditionnent l'admissibilité d'un problème d'ordonnancement.

3.1.2.3.1. Les contraintes temporelles

Les contraintes temporelles se décomposent en plusieurs types :

- **contraintes de temps alloué**, issue généralement d'impératifs de gestion et relatives aux dates limites des tâches (délais de livraisons, disponibilité des approvisionnements, respect d'une date au plus tôt) ou à la durée totale d'un projet ou d'une campagne de production,
- **contraintes d'antériorité** et plus généralement les contraintes de cohérence technologique qui décrivent le positionnement de certaines tâches par rapport à d'autres (par exemple : les contraintes de gammes dans le cas des problèmes d'ateliers)
- **contraintes de calendrier** : ces dernières sont liées au respect des horaires de travail ou des périodes planifiées d'inactivité (maintenance préventive, vacances, etc)

3.1.2.3.2. Les contraintes de ressources

Ces contraintes traduisent le fait que les ressources sont disponibles en quantité limitée (leur capacité). On parle également de contraintes de partage. Dons ce cadre, deux types de contraintes de ressources sont distingués :

- **contraintes disjonctives** : les tâches doivent être effectuées l'une après l'autre. En effet, il s'agit de ressources qui ne peuvent être utilisées que par une tâche à la fois. Dans la plupart des problèmes, il s'agit des appareils et/ou des opérateurs.
- **contraintes cumulatives** : la somme des besoins en ressources de tâches cumulées doit être inférieure à la capacité des ressources. Dans un problème de procédé, il s'agit souvent des matières et des utilités (énergie).

3.1.2.4. Les objectifs et critères d'évaluation

Lorsqu'on aborde la résolution d'un problème d'ordonnancement, deux grands types de stratégies peuvent être choisis, visant respectivement à l'*optimalité* des solutions par rapport à un ou plusieurs critères, ou à leur *admissibilité* vis-à-vis des contraintes.

L'approche par optimisation suppose que les solutions candidates à un problème puissent être ordonnées de manière rationnelles selon un ou plusieurs critères d'évaluation numérique permettant d'apprécier la qualité des solutions. On cherchera à minimiser ou maximiser de tels critères [Lopez et al., 2001]. Le credo de l'ordonnancement d'atelier reste le tryptique qualité/coût/délais.

Néanmoins, il est parfois difficile de traduire l'ensemble des objectifs de résolution par un ou plusieurs critères numériques. On peut dans ce cas avoir recours à une approche par satisfaction de contraintes. L'ensemble des contraintes regroupe alors à la fois des contraintes intrinsèques au problème (cohérence technologique par exemple) et des objectifs de type seuil à atteindre ou à ne pas dépasser (durée maximale, stocks plafond/plancher…). On pourra dans ce cas, se contenter d'une solution quelconque, pourvu qu'elle soit admissible [Lopez et al., 2001].

3.2. DIFFÉRENTS TYPES DE PROBLÈMES D'ORDONNANCEMENT

La fonction *ordonnancement* peut être mise en œuvre dans différents contextes et pour différents objectifs. Ces deux aspects sont relativement importants car ils donnent souvent lieu à des méthodologies de modélisation et de résolution spécifiques.

3.2.1. Ordonnancement de projet / de production

L'ordonnancement est un processus de décision qui apparaît dans la plupart des systèmes de production et de transport [Semet, 1995], ainsi que dans la gestion de projet [Pinedo, 1995]. Dans le cadre de ces travaux, nous nous intéressons spécifiquement à l'ordonnancement d'ateliers de production.

3.2.1.1. Ordonnancement de projet

La particularité de l'ordonnancement de projet est d'étudier un projet unique, pour lequel on cherche à minimiser la durée totale de réalisation, sans tenir compte des contraintes de limitations de ressources.

Initialement limité à la gestion de grands projets, l'emploi de ces techniques, souvent fédérées sous le célèbre acronyme *PERT (Program Evaluation and Review Technique)*, s'est sensiblement répandu aujourd'hui dans les entreprises.

3.2.1.2. Ordonnancement de la production

La fonction *ordonnancement* occupe une place particulière dans la gestion informatisée des flux de production au sein de l'entreprise. C'est généralement le point de rencontre entre un système informatisé et hiérarchisé de production et le système de production lui-même. C'est le lieu de l'interface entre l'élaboration de la commande et la partie opérationnelle. La planification de la production crée les ordres de fabrication qui déterminent globalement ce qui doit être fait dans une période donnée (généralement la semaine). L'ordonnancement consiste à prévoir l'enchaînement de toutes les opérations élémentaires nécessaires à la réalisation de ces ordres de fabrication sur les ressources de production, tout en tenant compte des ressources secondaires (telles que les opérateurs, les outillages…), des contraintes extérieures (maintenance préventive, calendrier de travail…) et de l'existant (tâches en cours de réalisation…).

Cette fonction trouve sa place essentiellement dans les ateliers gérés en flux poussés ayant un mode de production par lot ; mode défini par des ordres de fabrication. En effet, dans les systèmes de production manufacturiers mono-produits ou les ateliers gérés en *Kanban* (flux tirés), les flux de produits s'écoulent naturellement et doivent en principe s'autoréguler. Notons toutefois que pour les ateliers de type procédés, le couplage des différents modes de production (continu/discontinu) nécessite aussi une fonction ordonnancement, même pour des systèmes de production mono-produits.

Les problèmes d'ordonnancement d'ateliers peuvent être classés en deux catégories en fonction du nombre de machines nécessaires pour réaliser chaque tâche. La première regroupe les problèmes pour lesquels chaque tâche nécessite une seule machine, la deuxième, ceux pour lesquels chaque tâche demande plusieurs machines pour son exécution.

La première catégorie de problèmes concerne les ateliers à machines parallèles ou non dédiées. Ce type d'atelier se caractérise par le fait que plusieurs machines sont disponibles pour l'exécution d'un travail qui n'en nécessite qu'une seule. Il s'agit d'une généralisation du problème à une machine. Il est possible de distinguer trois types d'ateliers selon la durée d'exécution d'une tâche sur ces machines :

- les *ateliers à machine identiques* : toute tâche peut s'exécuter sur n'importe quelle machine avec une même durée opératoire (à condition que la machine soit libre).
- les *ateliers à machines uniformes* : chaque machine possède une vitesse d'exécution propre et ceci, indépendamment de la tâche à réaliser.
- les *ateliers à machines indépendantes* (ou non reliées): la durée d'exécution d'une tâche sur une machine donnée dépend de la tâche à réaliser.

La deuxième classe de problèmes, quant à elle, englobe les ateliers composés de *m* stations différentes. Une station comporte une ou plusieurs machines en parallèles. Les travaux à réaliser sont constitués d'un ensemble de *n* opérations élémentaires, chacune doit s'exécuter sur une machine différente de celles des autres opérations. En fonction du mode de passage des opérations sur les machines, on retrouve les trois classes d'ateliers classiquement définis, à savoir le flow shop, le job shop et l'open shop.

3.2.2. Ordonnancement statique / dynamique

Deux catégories de problèmes d'ordonnancement peuvent être distinguées en fonction du degré de connaissance que l'on a du problème à résoudre au moment de sa résolution. On parle de :

- **problème statique** lorsque les tâches à ordonner sur une période ainsi que l'état initial de l'atelier sont connus au début de la période ;
- **problème dynamique** lorsque les décisions sont à prendre sur la période mais toutes les tâches à réaliser sur cette période ne sont pas totalement connues au début de la période.

3.2.3. Ordonnancement prédictif / réactif

La fonction ordonnancement peut tenir deux rôles toujours présents dans un système de production, mais pouvant revêtir une importance relative variable :

- l'**ordonnancement prédictif** consiste à prévoir a priori un certain nombre de décisions en fonction des données prévisionnelles et d'un modèle de l'atelier;
- l'**ordonnancement réactif** consiste à adapter les décisions prévues en fonction de l'état courant du système et des déviations entre la réalité et le modèle (ce qui est prévu en théorie). Dans ce cas, la fonction ordonnancement doit être en relation avec le système de supervision qui fait remonter les alarmes qui ne peuvent pas être traitées localement (cf. section 1.1.4.2).

Comme nous l'avons souligné dans le chapitre 1, l'ordonnancement est une étape cruciale dans le processus de décision liée au pilotage d'un système de production. Ce processus de décision peut être décrit par un cycle d'ordonnancement, de contrôle, de suivi et de réaction (Figure 3.2). Il s'agit d'une application aux systèmes de production de la théorie plus générale liée pilotage décisionnel des systèmes [Sénéchal O., 2004 ; Trentesaux et al. 2002].

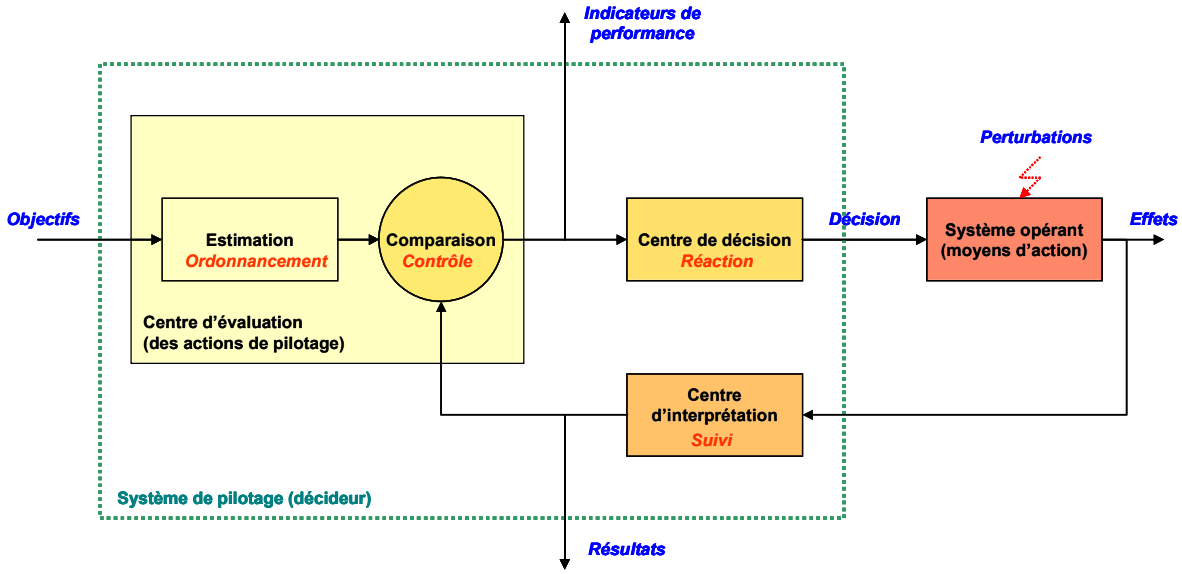


Figure 3.2 Pilotage décisionnel des systèmes : application aux systèmes de production

3.2.4. Ordonnancement préemptif / non préemptif

Si les tâches peuvent être interrompues, on parle d'ordonnancement préemptif (non préemptif dans le cas contraire).

L'ordonnancement d'atelier relève en général de cas non préemptif car il est rare qu'une tâche de production soit interrompue une fois lancée, sauf en cas de dysfonctionnement avéré du système.

Le cas préemptif est plus souvent rencontré en informatique, notamment dans la gestion des processus ou des threads au sein de systèmes d'exploitation multitâches (tels que *VxWorks* ou *Unix*).

3.3. VISUALISATION GRAPHIQUE D'UN ORDONNANCEMENT

Le **diagramme de Gantt** est certainement la représentation la plus ancienne, la plus répandue et la plus simple pour visualiser graphiquement l'exécution des tâches et/ou l'occupation des ressources au cours du temps. Le diagramme de Gantt consiste à placer les tâches sur un graphique où le temps est en abscisse et les ressources en ordonnée. A chaque tâche est associée un segment ou une barre horizontale, de longueur proportionnelle à la durée de traitement.

A titre d'illustration, la Figure 3.3 visualise le diagramme de Gantt d'un ordonnancement réalisable du problème présenté ci-dessous :

Exemple *:*

Soit un problème d'ordonnancement composé de trois ressources notées A, B et C et de 8 tâches numérotées de 1 à 8. La gamme opératoire associée à chaque OF est montrée ci-dessous et indique la ressource et la durée de chaque tâche. On peut constater que les tâches 1, 5 et 6 doivent se réaliser sur la même ressource, il en est de même pour les tâches 2,4 et 7 ainsi que pour les tâches 3 et 8.

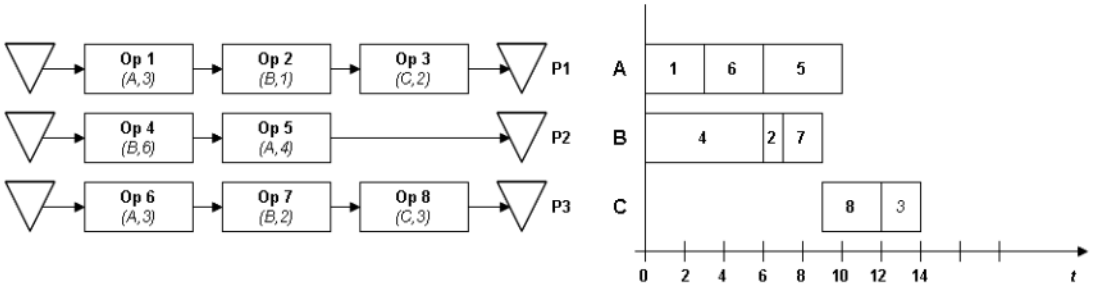


Figure 3.3 Diagramme de Gantt

3.4. PRINCIPALES CARACTÉRISTIQUES DES PROBLÈMES D'ORDONNANCEMENT DE PROCÉDÉS DISCONTINUS

De nombreux aspects doivent être pris en compte lors du développement de modèles d'ordonnancement de procédés batch. Une caractérisation systématique, est d'abord présentée à travers une classification générale qui permet de répertorier la plupart des caractéristiques pertinentes d'un problème d'ordonnancement. Cette classification est résumée sur la Figure 3.4 et ne prend pas seulement en compte les équipements et le matériel, mais aussi les contraintes liées au temps et à la demande.

Tout d'abord, le procédé mis en œuvre et ses implications *topologiques* ont une influence significative sur la complexité des problèmes. Comme la section 1.2.4.2, les procédés batch se décomposent souvent en un traitement séquentiel, impliquant une ou plusieurs étapes. Chaque étape peut être réalisée par un ou plusieurs équipements en parallèle. Chaque lot doit être traité d'après une séquence d'étapes définies dans la *recette* (produit/lot). Dans un souci d'optimisation, les ateliers de fabrication en Génie de Procédés deviennent de plus en plus complexes et il est de même de leur topologie, les recettes doivent alors prendre en compte des opérations de mélange et de séparation, de recyclage ….

Les *contraintes sur les équipements* en termes d'affectation et de connectivité étant étroitement liées aux *considérations topologiques*, les interconnexions limitées entre les équipements imposent des contraintes strictes sur l'allocation des unités.

Un autre aspect important des ateliers de Génie des Procédés réside dans l'importance des *politiques de stockage*. Ces ateliers nécessitent parfois la prise en compte de contraintes de stockage dans des cuves dédiées à capacité finie, bien que de nombreux cas d'étude s'appuient sur des hypothèses simplificatrices comme celles *de type zero-wait* qui impose un transfert immédiat entre deux tâches successives sans politique de stockage, ou à l'inverse les hypothèses de stockage à capacités illimitées qui masquent complètement les problèmes liés à la gestion des stocks.

Le *transfert de matière* est souvent supposé instantané, cependant dans certains cas comme par exemple les transferts par gravité (dont les durées peuvent être longues en comparaison avec des transferts par pompes), sa durée est importante et doit être prise en compte dans les approches de modélisation correspondantes.

Un autre facteur important est la manipulation des *tailles de lots*. Dans les usines pharmaceutiques, les tailles de lots sont généralement fixes pour des raisons de traçabilité (pas de mélange ou de fractionnement de lots), tandis que pour la production de solvants ou de polymères végétaux, les tailles de lots peuvent être variables et les séparations de courant ou les mélanges sont fréquents.

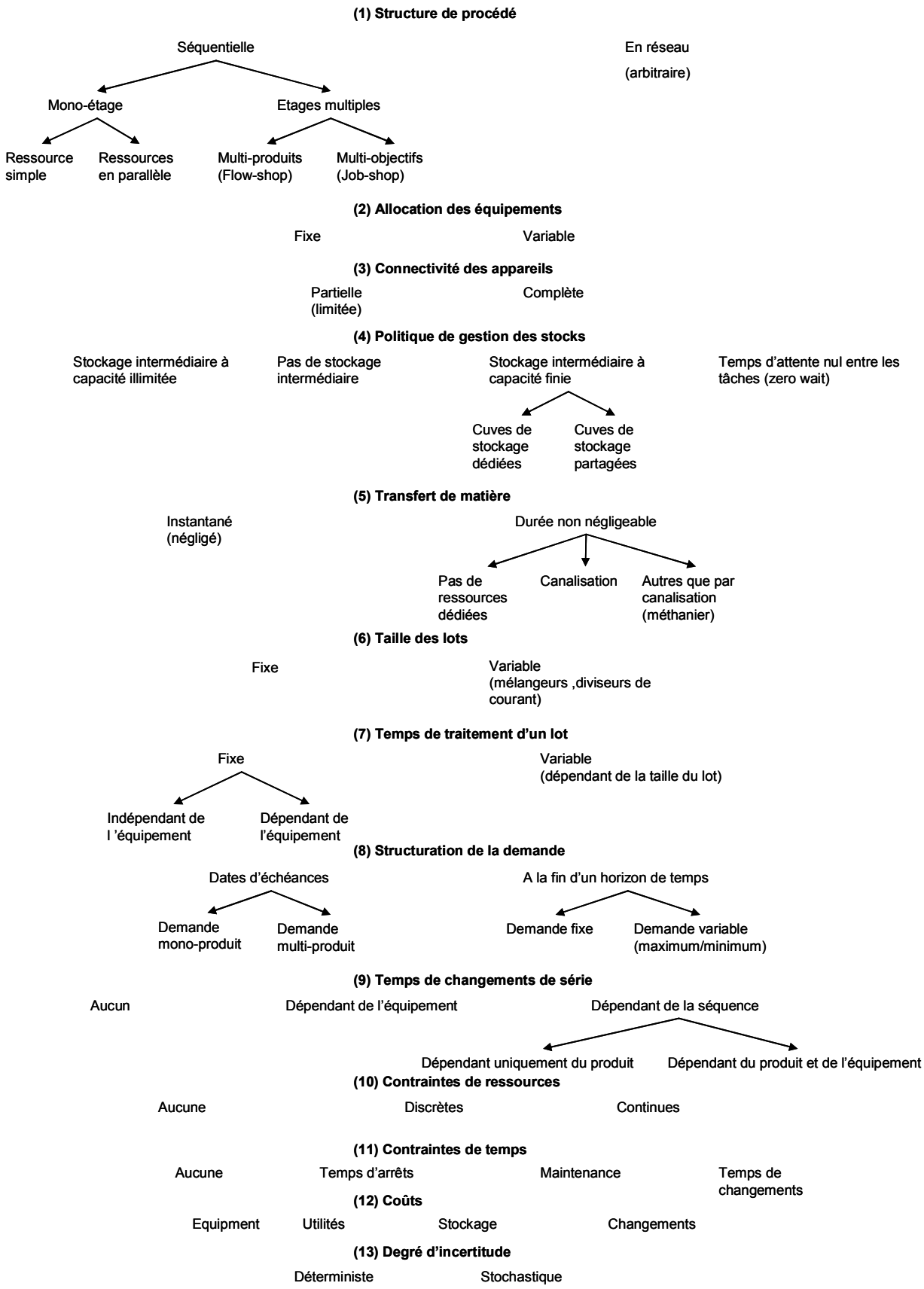


Figure 3.4 Caractéristiques des problèmes d'ordonnancement des procédés batch

De même, des exigences différentes sur les *temps de traitement* peuvent être trouvées dans divers secteurs d'activité en fonction des caractéristiques des procédés. Par exemple, les procédés dont les applications sont règlementées, comme la pharmacie, pourraient impliquer des temps de traitement fixes, tandis que la production de solvants ou de polymères peut être ajustée et optimisée avec les modèles de procédés.

La *structuration de la demande* peut aussi varier considérablement, allant des cas où les dates d'échéance doivent être respectées aux cas où les objectifs de production doivent être remplis sur un horizon temporel donné (fixe ou dates d'échéance).

Les *temps de changements de série* sont également un facteur très important qui est particulièrement critique dans les cas de *phases transitoires dépendantes de l'ordre des produits*, par opposition au simple *set-up* qui ne sont dépendants que de l'unité. Cela correspond au problème de nettoyage ou de « montée en couleur ».

Les *contraintes de ressources*, en dehors de l'allocation des équipements, (par exemple, les *utilités* comme l'eau, la vapeur de différente pression, l'électricité...etc.), sont également souvent d'une grande importance et peuvent être entièrement discrètes ou continues. Les pratiques d'exploitation donnent souvent lieu à des contraintes de temps, comme les week-ends ou les périodes de maintenance.

En outre, même si l'ordonnancement est souvent considéré comme un problème de faisabilité, les *coûts* associés à l'utilisation des équipements, aux stocks, aux changements et aux utilités peuvent avoir un impact significatif dans la définition d'un calendrier optimal.

Enfin, il y a la question du *degré d'incertitude dans les données* qui doit être pris en compte, et qui est plus particulièrement critique lorsque l'horizon de temps alloué à la production est réduit.

La classification de la Figure 3.4 montre qu'il existe une très grande diversité de facteurs qui doit être prise en compte dans le problème d'ordonnancement à court terme (ici, identification de 13 catégories principales), ce qui rend assez difficile la tâche de développement de méthodes unifiées de résolution. Dans le même temps, un bon compromis peut être d'avoir un certain nombre de méthodes spécialisées qui peuvent traiter des cas de cette classification d'une manière plus efficace.

3.5. MÉTHODES DE RESOLUTION DES PROBLEMES D'ORDONNANCEMENT

Quelque soit le secteur industriel considéré et la modélisation mise en œuvre, un problème d'ordonnancement se ramène généralement à la résolution d'un problème d'**optimisation combinatoire**. Aujourd'hui, différentes approches sont proposées dans la littérature pour traiter ce type de problème. Même si les méthodes de résolution sont presque aussi variées que les problèmes d'ordonnancement, on distingue néanmoins trois grandes classes :

- les **approches mathématiques** basées essentiellement sur les techniques de la **recherche opérationnelle** telles que la théorie des graphes, la programmation mathématique (*linéaire*, *mixte*, *non linéaire*, *etc.*), les heuristiques souvent basées sur un algorithme optimisé pour un type spécifique de problème de production et les méta-heuristiques,
- les **approches** rattachées au vaste domaine de l'**intelligence artificielle**,
- les **approches par simulation** de tout type (*continue*, *hybride*, *à événements discrets*, *etc.*) qui mettent en œuvre des règles plus ou moins sophistiquées de placement (pour le séquencement des tâches) et de priorité (pour le partage des machines).

Notons de plus, que chacune de ces approches regroupe aussi bien :

- des **méthodes exactes** qui assurent l'obtention de la solution optimale mais souvent au prix de temps de calcul relativement important selon la complexité du problème (*polynômiale*, *NP-difficile*, etc.),
- que des **méthodes approchées** qui permettent de trouver une solution proche de l'optimal en un temps raisonnable.

Enfin, les techniques de modélisation utilisées dans la plupart des méthodes de résolution précédentes s'appuient sur une **représentation graphique** du problème d'ordonnancement. Le caractère visuel de ces graphes facilite l'interprétation des solutions et surtout, la sémantique associée permet généralement de construire de manière systématique et non ambigüe le modèle du problème. Par ailleurs, les modèles s'appuyant sur une représentation graphique se prêtent bien au développement d'outils graphiques intégrant une *IHM* et permettent de définir un problème de manière plus intuitive et/ou plus proche du métier en masquant la complexité des outils mathématiques sous-jacents. Ces aspects sont abordés dans le chapitre 4.

La suite de cette section dresse un rapide panorama des différentes techniques exploitées aujourd'hui pour résoudre le problème d'ordonnancement. Ne pouvant pas être exhaustif, nous nous intéressons plus spécifiquement aux méthodes permettant de résoudre des problèmes d'ordonnancement de n ordres de fabrication (ou *jobs*) sur m (m>2) machines, problèmes couramment rencontrés en Génie des Procédés.

3.5.1. Approches mathématiques

Un grand nombre de méthodes basées sur une approche mathématique s'appuient sur les outils de la **recherche opérationnelle**. Dans ce cadre, différentes techniques sont mises en œuvre selon la complexité du problème et les objectifs visés. Cette section présente brièvement celles qui sont le plus couramment rencontrées.

3.5.1.1. Méthodes exactes

3.5.1.1.1. Procédures de séparation et d'évaluation

Une procédure par séparation et évaluation (PSE, aussi appelée Branch and Bound) est une méthode d'exploration par énumération implicite de l'espace de recherche. Elle utilise une représentation de l'ordonnancement sous forme d'une arborescence dont les sommets représentent chacun un sous-problème et les arcs issus d'un même sommet représentent chacun une décomposition du problème situé au sommet de l'arbre en sous-problèmes de taille réduite. Cette arborescence est explorée de façon à éviter les branches ne contenant pas de solutions réalisables et les branches n'amenant pas à des solutions meilleures que la solution courante. D'après [Lopez et al. 2001], les *PSE* reposent sur quatre composantes essentielles :

- La technique de séparation, qui permet de décomposer un problème en le partitionnant en sous-problèmes de taille réduite,
- La méthode d'évaluation qui associe une borne au critère d'optimisation sur l'ensemble des solutions d'un sous-problème (borne supérieure dans le cas d'une minimisation). Cette borne permet d'éviter l'exploration d'un sommet dont la solution partielle correspondante a une valeur du critère dépassant la borne.

- La méthode de sondage, qui permet de déterminer si un sommet est terminal, (c'est-à-dire s'il ne contient pas de solution admissible, ou si c'est une solution optimale, ou si l'on peut obtenir de façon polynomiale la solution optimale de ce sous-problème), ou s'il mérite d'être séparé.
- La méthode de sélection ou stratégie d'exploration, qui décrit comment choisir le sous-problème à séparer, lorsque plusieurs sont candidats. On peut distinguer deux stratégies d'exploration, celle qui favorise les meilleurs d'abord appelée aussi procédure par séparation et d'évaluation progressive, et celle de type profondeur d'abord et retour arrière (procédure par séparation et évaluation séquentielle).

Cette méthode est optimale, mais son temps de réponse peut être important. En pratique, il est difficile d'imaginer de traiter de manière efficace des cas dépassant quelques dizaines d'ordres de fabrications et quelques machines, ce qui explique que cette méthode soit peu utilisée dans les logiciels industriels.

3.5.1.1.2. Théorie des Graphes

La théorie des graphes apporte une aide incontestable à la manipulation d'un ensemble de données numériques. Elle est un support rigoureux à la fois pour la vérification de la cohérence du problème posé et pour sa résolution. Les formulations courantes liées au problème d'ordonnancement passent par la définition d'un graphe manipulant des inégalités de potentiels [Lopez et al., 2001].

Un graphe de précédence est constitué d'un ensemble de nœuds et de deux types d'arcs. Les nœuds représentent les tâches à réaliser. Les arcs conjonctifs valués (dont la valuation représente la durée de réalisation de la tâche d'où l'arc sort), représentent les contraintes de précédences. Les arcs disjonctifs à deux sens représentent les contraintes de ressources. Pour représenter le début et la fin de l'ordonnancement, deux nœuds fictifs sont rajoutés au graphe.

Ainsi l'exemple de la section 3.3 peut être représenté sous forme de graphe de la manière suivante :

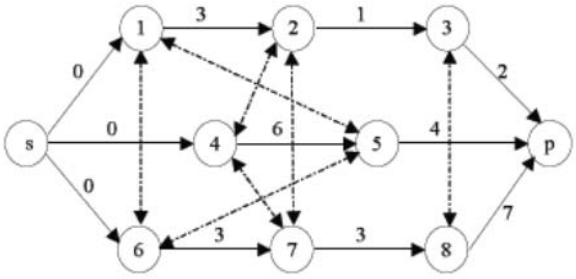


Figure 3.5 Exemple de graphe de précédence

Comme nous l'avons vu précédemment, un problème d'ordonnancement peut en général être exprimé sous forme de graphe. Pour résoudre un tel problème (Figure 3.5), il s'agira d'orienter les arcs matérialisant les conflits (ce qui donne la priorité à une des opérations) en valuant ces arcs par la valeur de l'attente nécessaire pour lever le conflit, ce qui peut être fait par exemple en utilisant des règles de type *SPT (Shortest Processing Time)* ou *EDD (Earliest Due Date)*.

De manière plus générale, de nombreux algorithmes ont été développés en théorie des graphes pour déterminer des chemins optimaux (algorithmes de Dijkstra, Sedewick et Vitter, Bellmann, Floyd, etc.)

Ces différents algorithmes traitent des cas légèrement différents, et la difficulté principale de leur utilisation est de modéliser un problème pratique sous forme d'un graphe permettant d'utiliser l'un d'eux de manière efficace. Cette difficulté nous paraît les réserver à des personnes ayant des compétences aussi bien en ordonnancement qu'en théorie des graphes.

Il existe deux types de modélisation d'un problème d'ordonnancement. La modélisation sous forme de graphes de précédences et la représentation analytique sous forme de programme mathématique. Le premier type de modélisation présente un caractère visuel facilitant l'interprétation des solutions. De plus ce type de modélisation se prête bien au développement d'outils graphiques d'aide à la décision.

3.5.1.1.3. Programmation dynamique

La programmation dynamique est une méthode d'optimisation opérant par phases (ou encore par séquences). Cette méthode de résolution est adaptée aux problèmes qui satisfont au principe d'optimalité de Bellmann [Bellmann et al., 1974] : une sous-solution d'une solution optimale est elle-même optimale pour la fonction objectif restreinte aux solutions partant de cette solution. Ce principe permet une résolution ascendante, qui détermine une solution optimale d'un problème à partir des solutions de tous les sous-problèmes. Chaque étape correspond à un sous-problème à résoudre de manière optimale en tenant compte des informations obtenues des étapes précédentes. Ceci nécessite une formulation du critère sous forme d'une relation de récurrence liant deux niveaux successifs.

Cette méthode est destinée à résoudre des problèmes d'optimisation à vocation plus générale que la méthode de séparation et évaluation. Par contre, la taille des problèmes qu'elle permet d'aborder est plus limitée.

3.5.1.1.4. Programmation linéaire

La programmation linéaire est une des méthodes classiques de la recherche opérationnelle. Elle permet la modélisation d'un problème d'optimisation d'une fonction objectif Z de plusieurs variables en présence de contraintes sous la forme d'un programme mathématique.

Le programme est dit *linéaire* si la fonction et les contraintes sont toutes des combinaisons linéaires de variables. Il comporte *n* variables non négatives (1.3), *m* contraintes d'égalité ou d'inégalité (1.2) et la fonction *objectif* à optimiser (1.1).

$$\max ou \min Z = \sum_{j=1}^{n} c_j x_j \qquad (1.1)$$

$$\begin{cases} \forall i = 1..m : \sum_{j=1}^{n} a_{ij} x_j \leq ou \geq b_i & (1.2) \\ \forall j = 1..n : x_j \geq 0 & (1.3) \end{cases}$$

Le coefficient de coût ou de profit de la variable x_j est noté c_j, celui de la variable x_j dans la contrainte i est noté a_{ij}. La contrainte i a un second membre constant b_i. Les contraintes simples de positivité ne sont pas incluses dans les m contraintes car elles sont générées par les algorithmes.

Si les variables sont astreintes à être entières, on a un programme *linéaire en nombres entiers (PLNE)*. Un programme linéaire en 0-1 est un cas particulier des *PLNE* dont les variables ne peuvent prendre que deux valeurs 0 ou 1 ; ces variables sont dites booléennes, binaires ou de décision. Enfin, à partir du moment où au moins une des contraintes ou la fonction objectif n'est pas une combinaison linéaire de variables, on parle alors d'un programme *non linéaire (PNL)*. Les *PLNE* et les *PL* en 0-1 sont plus difficiles à résoudre que les *PL* classiques. Les *PNL* sont encore plus difficiles.

Une formulation classique du problème d'ordonnancement de base est la suivante [Van Hulle 1991] : soit O l'ensemble des opérations, M l'ensemble des machines et d_{ik} la durée opératoire de l'opération i sur la machine k. On cherche à minimiser (ou maximiser) C (critère retenu) avec $\{i,p\} \in O, \{k,h\} \in M \text{ et } K \approx \infty \text{ (très grand en pratique)}$:

$$\begin{cases} \text{date de début des opérations :} & t_{ik} \geq 0 \\ \text{contraintes de précédence :} & t_{ik} - t_{ih} \geq d_{ih} \text{ si } O_{ih} \text{ précède } O_{ik} \\ \text{contraintes disjonctives :} & t_{pk} - t_{ik} + K(1 - y_{ipk}) \geq d_{ik}\, y_{ipk} = 1 \text{ si } O_{ik} \text{ précède } O_{pk} \\ & t_{ik} - t_{pk} + K(y_{ipk}) \geq d_{pk}\, y_{ipk} = 1 \text{ autrement} \end{cases}$$

Ces méthodes d'optimisation, à la puissance d'expression importante, sont fréquemment utilisées en Génie des Procédés depuis les problèmes d'agencement d'ateliers [Georgiadis M.C. and Macchietto S., 1997], au pilotage de ces derniers [Mendez CA, 2006]. Des contraintes très variées peuvent être ajoutées de la même façon. La méthode *Simplexe* est une méthode classique de résolution de programme linéaire. Dans des cas complexes, la résolution de programmes linéaires se fera à l'aide de solveurs du marché.

3.5.1.2. Méthodes de résolution approchées

Malgré l'évolution permanente des calculateurs et les progrès permanents de l'informatique, il existe pour plusieurs problèmes d'optimisation combinatoire une taille critique de l'espace de solutions admissibles. La méthode permettant d'obtenir une solution optimale est bien évidemment celle de l'énumération complète de l'espace de recherche. Cette dernière est dans la plupart des cas prohibitive. Compte tenu de ces difficultés, la plupart des spécialistes de l'optimisation combinatoire ont orienté leur recherche vers le développement de méthodes heuristiques. Une méthode heuristique est souvent définie comme une procédure exploitant au mieux la structure d'un problème, dans le but de trouver une solution de qualité raisonnable en un temps de calcul aussi faible que possible [Lopez et al. 2001].

Bien que l'obtention d'une solution optimale ne soit pas garantie, l'utilisation d'une méthode heuristique offre de multiples avantages par rapport à une méthode exacte :

- La recherche d'une solution optimale peut être totalement impossible dans certaines applications pratiques en raison du nombre de variables et de contraintes étudiées, de l'antagonisme entre les objectifs à atteindre et parfois même de l'imprécision des données récoltées.

- L'exécution des méthodes heuristiques est souvent rapide. En effet, ces dernières fournissent en un temps polynômial une ou plusieurs solutions admissibles et de bonne qualité.
- L'application des méthodes heuristiques est à la portée des utilisateurs non expérimentés. En effet, une méthode heuristique se base sur des règles et des principes simples et « intelligents » ce qui leur permet d'être compréhensibles.
- L'adaptation ou la combinaison d'une méthode heuristique avec d'autres méthodes est souvent facile. Cette flexibilité permet d'augmenter la performance de la méthode hybride résultante et de garantir parfois l'obtention de bonnes solutions.

Avant de présenter une typologie des *métaheuristiques* développées pour résoudre un problème d'ordonnancement, nous définissons les termes *heuristique* et *métaheuristique*.

Une **heuristique** est une méthode de résolution de problèmes non fondée sur un modèle formel et qui n'aboutit pas nécessairement à une solution (Journal Officiel du 16 septembre 1989). Il s'agit généralement de méthodes basées sur le déroulement d'un algorithme. Le principal inconvénient est que ces méthodes sont souvent spécifiques à un type de problème donné. Les **métaheuristiques** sont quant à elles, des stratégies d'optimisation combinatoire basées elles-mêmes sur des heuristiques. On peut les considérer comme une réponse au problème de manque d'assurance dans les performances d'une heuristique : une heuristique permettra en général d'arriver rapidement à une solution au problème, tandis que la métaheuristique conduira une exploration plus exhaustive de l'espace des solutions afin de s'assurer que la solution n'est pas un minimum local.

Une classification des différents types de métaheuristique est présentée par [Widmer et al., 2001]. Les auteurs considèrent qu'il existe trois types de métaheuristique : les *métaheuristiques constructives* basées sur l'idée de diminution progressive de la taille du problème, les *métaheuristiques basées sur la recherche locale* et les *métaheuristiques évolutives* inspirées des phénomènes réels, telles que les algorithmes génétiques et les algorithmes de colonies de fourmis, etc. Afin de pouvoir expliquer le principe de certains types d'approches, nous allons utiliser les notations suivantes : soit X, l'ensemble des solutions admissibles du problème traité (nous supposons que cet ensemble est fini). Soit f, une fonction d'évaluation des solutions admissibles. Résoudre un problème d'optimisation combinatoire d'une manière optimale en respectant les contraintes qui lui sont associées (dans le cas de minimisation), consiste à déterminer une solution $s^* \in X$ vérifiant :

$$f(s^*) = \min_{s \in X} f(s)$$

3.5.1.2.1. Approches constructives

L'idée principale d'une approche de résolution constructive est de diminuer progressivement la taille de l'espace de recherche en fixant à chaque étape la valeur d'une variable du problème. Notons que la construction d'une solution réalisable est statique, par conséquent, les prises de décisions intermédiaires ne sont jamais remises en cause. La majorité des méthodes constructives sont de type glouton et sont souvent considérées comme myopes. A chaque étape, la solution courante est complétée de la meilleure façon sans tenir compte des conséquences que cela entraîne au niveau de la qualité de la solution finale.

Les méthodes constructives se distinguent par leur rapidité et leur grande simplicité. On obtient en effet très rapidement une solution admissible sans avoir recours à des connaissances approfondies dans le domaine de l'optimisation combinatoire. Le principal défaut de ces méthodes réside dans la qualité des solutions fournies. En effet, le fait de vouloir opérer rapidement sans tenir

compte du contexte global de problème a souvent des conséquences négatives sur le coût de la solution obtenue. Il est donc préférable de mettre au point des procédures anticipant les effets secondaires occasionnés par les décisions prises lors de la construction d'une solution admissible.

3.5.1.2.2. Approches de recherche locale

Les méthodes de recherche locale sont des algorithmes itératifs qui explorent l'espace d'états X en partant d'une solution admissible s_O choisie arbitrairement ou à l'aide d'une heuristique. Le principe de la recherche locale est de la modifier légèrement à chaque itération k, à l'aide d'un opérateur de solution s_k. Ce procédé prend fin lorsqu'une solution d'arrêt est satisfaite. Les conditions d'arrêt peuvent être un seuil d'itération ou l'évaluation de la qualité de la solution courante. Dans ce dernier cas, si l'application de l'opérateur à la solution courante ne permet pas de l'améliorer, l'algorithme s'arrête.

Nous allons détailler par la suite trois approches de recherche locale :

- la méthode de descente
- le recuit simulé
- la méthode tabou

a. Méthode de descente

La méthode de descente est l'une des approches de recherche locale. Cette méthode explore l'espace X, en choisissant à chaque fois la meilleure solution voisine de la solution courante. Ce procédé continue aussi longtemps que la valeur de la fonction objectif diminue. La recherche s'interrompt dès lors qu'un minimum local de f est atteint. Historiquement, les méthodes de descente ont connu un grand succès pour le traitement des problèmes d'optimisation combinatoire. Toutefois, elles comportent deux obstacles majeurs qui limitent considérablement leur efficacité :

- suivant la taille et la structure du voisinage d'une solution $s \in X$, la recherche de la meilleure solution voisine est un problème qui peut être aussi difficile à résoudre que le problème initial,
- une méthode de descente peut fournir un minimum local. En effet, le processus s'arrête dès qu'un minimum est atteint. Or, les problèmes d'optimisation combinatoire comportent typiquement de nombreux optima locaux pour lesquels la fonction objectif peut être fort éloignée de la valeur optimale.

La Figure 3.6 montre l'inconvénient de la méthode de descente se basant sur un choix aléatoire d'un voisin meilleur et s'arrêtant s'il n'en existe plus.

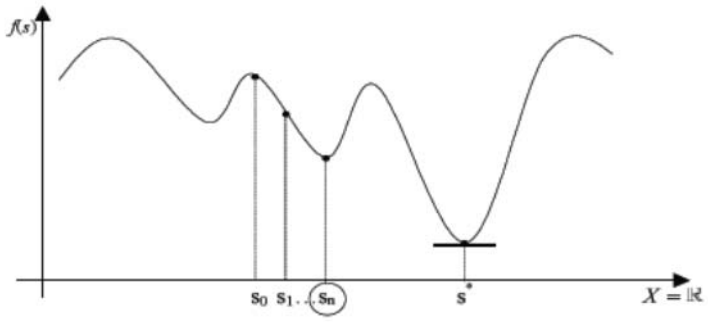


Figure 3.6 Blocage dans un optimum local de f

Pour faire face à ces carences, d'autres méthodes de recherche locale plus sophistiquées ont été développées au cours de ces dix dernières années. Ces méthodes analysent des solutions voisines moins bonnes afin d'éviter les optima locaux.

Les méthodes les plus connues seront introduites aux paragraphes suivants. Il y a deux différences majeures entre ces méthodes :

- la manière d'explorer les solutions voisines d'une solution,
- le critère d'arrêt de la recherche (qui est souvent difficile à choisir puisque la solution optimale est rarement connue).

b. Recuit simulé

Le recuit simulé est une méthode de recherche locale dont les origines remontent aux expériences de Metropolis en 1953 [Metropolis et al., 1953]. Leurs travaux consistaient à étudier la stabilité thermique d'un système physique. Cette méthode de recherche utilise une analogie avec le processus de recuit d'un matériau : par refroidissement lent après une élévation de température, on peut obtenir à la fin du processus un état d'énergie minimal.

Le recuit simulé démarre par une solution initiale admissible et continue l'exploration de l'espace d'états en effectuant des perturbations mineures à la solution courante. Si la solution voisine obtenue améliore le critère cherché alors elle est retenue. Si au contraire, elle provoque une dégradation ΔE du critère, elle est retenue avec une probabilité $P = \exp(-\Delta E / T)$, où T est un paramètre inversement proportionnel au nombre d'itérations. La diversification est donc privilégiée en début de processus, l'intensification étant ensuite dominante.

Les avantages du recuit simulé sont nombreux. Tout d'abord, cette approche est en mesure de fournir des résultats satisfaisants dès que l'on sait trouver une solution initiale admissible et une manière d'explorer les solutions voisines. De plus, cette méthode permet de traiter les problèmes d'optimisation combinatoire pour lesquels il n'existe aucune méthode de résolution. Le fait de garder les solutions moins bonnes permet de couvrir un espace de recherche plus grand et d'éviter une convergence prématurée vers un optimum local. Le recuit simulé a été abondamment utilisé pour résoudre des problèmes pratiques d'optimisation tel que le job shop.

c. Méthode tabou

Cette méthode a été créée dans les années 1970 et a été présentée pour la première fois par Glover en 1986 [Glover F., 1986]. La formulation finale de cette méthode est apparue en deux parties [Glover F., 1989], et [Glover F., 1990]. Cette méthode est basée sur deux principes. Le premier consiste à améliorer à chaque itération la valeur de la fonction objectif en choisissant à chaque fois la meilleure solution voisine si elle existe. Si aucune amélioration n'existe dans le voisinage, le choix se fait sur le moins mauvais des voisins. Ce principe utilisé seul, présente un inconvénient majeur. Si un optimum local se trouve dans un pic très grand (éventuellement ou au fond d'une vallée profonde s'il s'agit d'un problème de minimisation), il sera impossible de l'atteindre (ou d'en sortir). C'est pour cette raison que la méthode Tabou s'appuie sur un deuxième principe qui consiste à garder en mémoire les dernières solutions visitées et à interdire le retour vers celles-ci pendant un nombre d'itérations fixé.

La méthode Tabou est une métaheuristique souvent utilisée pour résoudre des problèmes industriels car elle présente une amélioration importante des algorithmes de plus forte pente. On parle également de recherche guidée. Plusieurs problèmes de types flow shop ou job shop sont traités par cette méthode.

3.5.1.2.3. Approches évolutives

De tout temps, les sciences de la vie et les processus naturels ont constitué des bases d'inspiration et d'imitation pour les chercheurs et les ingénieurs. Les mécanismes du monde vivant sont à l'origine des systèmes artificiels utilisables dans des contextes variés. Les méthodes évolutives qui sont présentées dans cette section constituent la base d'un nouveau champ de la programmation informatique en pleine effervescence.

Contrairement aux méthodes constructives et de recherche locale qui font intervenir une solution unique, les méthodes évolutives traitent un ensemble de solutions admissibles (appelé aussi population d'individus). L'idée principale consiste à appliquer des opérateurs spécifiques à certaines solutions distinguables pour faire évoluer la population vers un niveau acceptable de performance moyenne. En général, la taille n de la population reste constante tout au long du processus cyclique d'exploration de l'espace X. Après avoir généré une population initiale, généralement de manière aléatoire, une méthode évolutive tente d'améliorer la qualité moyenne de la population courante en ayant recours à des principes d'évolution naturelle. Le processus cyclique qui est à la base d'une méthode d'évolution est composé d'une phase de coopération et d'une phase d'adaptation individuelle qui se succèdent à tour de rôle.

Lors de la phase de coopération, les solutions de la population courante sont comparées puis combinées entre elles dans le but de produire des solutions inédites et de bonne qualité. L'échange d'information qui en résulte, se traduit par l'apparition de nouvelles solutions admissibles qui héritent des caractéristiques prédominantes contenues dans les solutions de la population courante. Dans la phase d'adaptation individuelle, les solutions reçoivent sans aucune interaction entre elles des modifications mineures en restant évidemment admissibles. Une nouvelle génération de solutions est créée au terme de chaque phase d'adaptation individuelle.

a. Algorithmes génétiques

Les algorithmes génétiques constituent à ce jour, l'approche la plus utilisée parmi les méthodes évolutives. Cette méthode donne souvent de très bons résultats aux problèmes d'optimisation, mais un grand nombre de solutions sont susceptibles d'être évaluées, ce qui peut demander un temps de traitement important.
Les paramètres à régler sont nombreux et influent beaucoup sur la qualité des solutions finales :

- taille de la population initiale,
- probabilités des individus d'accéder à la population suivante en fonction de leur adaptation,
- pourcentage respectif de croisement, mutation et recopie dans l'élaboration de la génération suivante.

b. Colonies de fourmis

Les colonies de fourmis [Dorigo et al. 1999] ont été à l'origine d'une certaine vogue de méthodes basées sur le comportement animal (essaims particulaires, etc.). Le paradigme des colonies de fourmis repose sur une analogie entre le problème d'optimisation et celui de la recherche d'un plus court chemin. Les fourmis en quête de nourriture déposent en effet des phéromones permettant à leurs consœurs de les suivre. Ces phéromones s'évaporant avec le temps, et une fourmi ayant tendance à suivre la trace la plus forte, les chemins les plus courts, moins sujets à évaporation et renforcés par le passage de nombreuses fourmis, auront tendance à être privilégiés avec le temps.

Les paramètres principaux à régler seront ici :

- le nombre de fourmis et leur rythme de sortie ;
- le mode de dépôt et d'évaporation des phéromones ;
- la probabilité qu'une fourmi suive un chemin en fonction de son degré de «marquage».

3.5.2. Approches basées sur les techniques de l'intelligence artificielle

Pour continuer, nous citerons quelques méthodes rattachables au domaine de l'intelligence artificielle, une présentation plus détaillée de ces méthodes peut être consultée dans [Grabot B., 2006].

3.5.2.1. Approches par contraintes

Les techniques de propagation de contraintes visent à réduire l'espace des solutions à un problème en appliquant des contraintes restreignant l'ordre avec lequel les variables sont sélectionnées, et la séquence avec laquelle les valeurs possibles leur sont assignées. Ces techniques ont par exemple été utilisées sous la forme voisine de l'analyse sous contraintes [Erschler J., 1976], dans le logiciel d'ordonnancement industriel Ordo.

Notons que ces techniques doivent souvent être complétées par d'autres pour aboutir à un ordonnancement précis, la propagation de contraintes seule n'aboutissant le plus souvent pas à une solution unique.

On trouvera une synthèse de l'application de ces techniques à l'ordonnancement, dont la mise en œuvre demande une nouvelle fois une certaine expertise, dans [Lopez et al. 2001].

3.5.2.2. Systèmes experts

Les systèmes experts visent à imiter le raisonnement humain, considéré comme décomposable en parties élémentaires dénommées « règles de production », constituées d'une partie « condition » et d'une partie « conséquence » (Si A alors B). Un système expert est alors composé d'une base de règles, renfermant la connaissance sur le problème à traiter, et d'une base de faits regroupant les propositions « vraies ». Un « moteur d'inférence » permet alors de déterminer les parties conditions des règles qui sont vérifiées, et les conséquences qui peuvent en être déduites et ajoutées à la base de faits.

La combinatoire du problème d'ordonnancement a fait que, très tôt, des tentatives ont été faites pour « mettre en boîte » des connaissances sur le domaine ou sur un atelier donné afin d'orienter l'élaboration d'un ordonnancement. Ces expériences se sont heurtées à la difficulté du problème : peu de connaissances génériques semblent exister, et le développement d'une base de connaissances relative à un atelier donné demande un effort très important. De plus, les connaissances mises en jeu en ordonnancement semblent peu se plier à un schéma aussi binaire que celui de règles de production «simples». Dans la plupart des applications, seules des connaissances locales sont modélisées pour gérer quelques choix au sein de méthodes plus performantes (propagation de contraintes, par exemple).

3.5.2.3. Logique floue

La logique floue (voir par exemple [Kaufmann A., 1992]), reposant sur la notion de sous-ensembles flous, vise essentiellement à formaliser de manière « informatisable » l'imprécision et l'incertitude, considérées comme inhérentes à beaucoup de connaissances humaines. Au lieu d'être traduite par une valeur de vérité « Vrai » ou « Faux » supposant un seuil précis, une proposition comme « la commande X est urgente » se verra par exemple affecter une valeur de vérité comprise entre 0 et 1 après que la notion d'urgence ait été traduite par une « fonction d'appartenance » décrivant l'évolution de cette valeur de vérité sur le référentiel concerné.

Deux grandes catégories d'utilisation de la logique floue à l'ordonnancement peuvent être différenciées :

- utilisation pour intégrer des connaissances imprécises dans des systèmes experts.
- utilisation pour décrire des contraintes «flexibles» en propagation de contraintes [Lopez et al. 2001], ce qui ne pose plus le problème de décrire une base de connaissance complexe.

3.5.2.4. Réseaux de neurones

Les réseaux de neurones appartiennent à une autre branche de l'intelligence artificielle, puisqu'il s'agit non plus d'imiter le raisonnement humain mais la structure du cerveau, et en particulier ses capacités d'apprentissage.

Pour cela, on considère comme entité de base un neurone artificiel, dont l'état de sortie dépend de la somme pondérée de ses entrées par l'intermédiaire d'une fonction de transfert. Dans les cas simples, les entrées (variables) du problème seront représentées par une première couche de neurones, et les sorties par une autre (solution). Entre les deux seront implantées une à plusieurs «couches cachées». Les réseaux des différentes couches seront complètement interconnectés, la «connaissance» étant stockée dans les poids des connections.

Pour pouvoir être utilisé, un réseau de neurones devra d'abord suivre une phase d'apprentissage pendant laquelle des exemples résolus lui seront présentés (entrées et sorties désirées). À partir de l'écart entre la sortie actuelle et la sortie désirée, les poids internes du réseau seront progressivement ajustés pour permettre, au terme de la phase d'apprentissage, d'arriver à un système de poids unique permettant une réponse correcte à tous les exemples appris.

Un réseau de neurones permettant d'approximer n'importe quelle relation entre entrées et sorties, une réponse pourra ensuite être fournie à des exemples non appris mais situés dans le domaine des exemples appris (ce qui pose le problème de la représentativité des exemples servant à l'apprentissage). Pour cela, le réseau ne réalise néanmoins qu'une extrapolation simple entre les exemples appris : l'apprentissage d'un problème présentant un espace de solutions tourmenté demandera donc plus de neurones, mais aussi de nombreux exemples autour des singularités, dont la localisation est en général inconnue en ordonnancement. D'autres architectures, utilisant d'autres types de neurones, sont aussi possibles.

Comme les systèmes experts, les réseaux de neurones ne servent en général pas en ordonnancement à bâtir directement une solution, mais à réaliser des choix au sein d'autres méthodes : choix d'une règle de priorité adaptée à un contexte donné par exemple. Dans ce cadre, leur intérêt réside principalement dans le fait qu'il peut être plus facile de trouver des exemples résolus à un problème complexe que de modéliser sous forme utilisable l'expertise permettant cette résolution. C'est en particulier le cas en ordonnancement.

3.5.2.5. Raisonnements à partir de cas

Le raisonnement à partir de cas *(RàPC – Case-based reasoning)*, dont on pourra trouver un intéressant panorama d'applications dans [Napoli A., 1999], a des points communs avec les réseaux de neurones. En effet, on stockera aussi des cas résolus pour une transposition à des cas comparables mais pouvant être un peu différents. La différence fondamentale est que les réseaux de neurones utilisent une représentation numérique de l'information, tandis que le raisonnement à partir de cas utilise une représentation symbolique.

Des cas, stockés dans une «base de cas», sont décrits par les caractéristiques du problème, de son environnement et de la solution trouvée. Lorsqu'un problème nouveau est posé, la ressemblance entre les cas stockés et le nouveau cas est tout d'abord analysée. Le cas le plus proche est sélectionné et sa solution modifiée en fonction des différences entre les deux problèmes (incluant leur contexte).

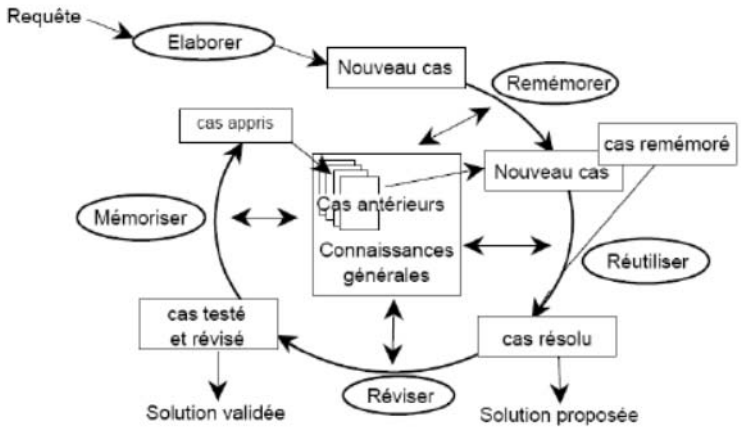


Figure 3.7 Le processus du RàPC [Fuch B. et al., 1997]

Cette technique suppose donc :

- d'être capable de recenser à l'avance les paramètres pertinents pour décrire un cas,
- de déterminer une fonction permettant d'évaluer la « distance » entre deux cas,
- de déterminer un opérateur permettant l'adaptation d'un cas.

Des applications encourageantes du *RàPC* à l'ordonnancement ont déjà été réalisées. Comme dans les cas précédents, il semble que les applications les plus prometteuses visent davantage à imiter le comportement de l'expert humain lors de certains choix ponctuels, ou à paramétrer certaines variables générales du problème d'ordonnancement, plutôt qu'à générer complètement un ordonnancement.

3.5.2.6. Systèmes multi-agents

Les systèmes multiagents appartiennent à la branche de l'intelligence artificielle distribuée. Plutôt que de modéliser la résolution d'un problème sous forme centralisée, il s'agit de créer une communauté d'entités autonomes (les agents) permettant de résoudre des sous-problèmes du problème générique. La résolution du problème global doit donc émerger de la résolution des problèmes élémentaires. Il s'agit donc d'une forme de résolution de problème par décomposition, les agents représentant en général les différentes entités impliquées dans le problème.

Les applications des systèmes multiagents à la production sont répandues depuis de nombreuses années, beaucoup d'entre elles étant orientées vers l'ordonnancement. Les applications typiques visent à substituer au paradigme classique de l'ordonnancement centré « machine » un paradigme « mixte » où :

- les machines sont modélisées par des agents et ont pour objectif de trouver de la charge ;
- les ordres de fabrication sont eux aussi modélisés par des agents, et tentent d'être élaborés au moindre coût et dans les meilleurs délais.

Cette formulation du problème est particulièrement intéressante pour intégrer les degrés de liberté pouvant exister dans l'élaboration des produits : les agents-ordres de fabrication émettent en effet des « offres » auxquelles les agents-machines répondent en termes de délai et de coût, un agent-OF pouvant émettre des propositions correspondant à ses différents modes d'élaboration possibles.

Ce type de fonctionnement permet aussi d'aborder de manière originale le problème de l'optimisation multiobjectif, des agents pouvant être dédiés à la satisfaction d'un objectif donné, le compromis final émergeant des interactions entre agents. On trouvera par exemple dans [Archimède B. et al., 2001] une application typique.

Malgré des avantages théoriques séduisants, l'utilisation des systèmes multiagents présente aussi des difficultés importantes, en particulier au niveau de leur mise en œuvre :

- développements informatiques importants pour implanter une communauté d'agents,
- problèmes de validation et de performance intrinsèques à la notion d'émergence, en particulier au niveau de la convergence et de la validation du comportement global du système,
- temps de traitements, dus en particulier aux négociations entre agents,
- problèmes liés à une optimisation essentiellement locale car distribuée, alors que la plupart des objectifs d'ordonnancement ont un caractère global (minimiser les retards, les temps de cycle, les en-cours, etc.).

3.5.3. Les approches par simulation

L'approche simulatoire fait partie des méthodes dites à construction progressive d'un ordonnancement. Ce type de méthodes permet de déterminer rapidement un ordonnancement respectant les contraintes techniques.

Le chapitre 1 a montré que différents outils et techniques de simulation existent, ceux-ci étant adaptés aux différents types de systèmes généralement rencontrés :

- la simulation des systèmes à dynamique continue,
- la simulation à évènements discrets,
- la simulation des systèmes dynamiques hybrides.

Cette dernière catégorie de système couvre de nombreux domaines allant des technologies de l'information et de la communication aux procédés chimiques. Notamment, les procédés batch sont intrinsèquement hybrides, puisqu'ils rassemblent de nombreuses unités de traitement continu interconnectées et partagées. Par ailleurs, avec un mode de production par lot, la procédure associée à la recette comporte une forte composante évènementielle mais dépend aussi de paramètres à valeurs continues (conditions opératoires, température, pression, etc.).

Le comportement des systèmes dynamiques est traditionnellement décrit par un modèle dont la représentation, continue ou discrète, est directement liée à la nature des variables d'état et temporelles qui le caractérisent [Zaytoon, 2001]. Les variables d'état d'un système sont :

- soit *continues* : la variable prend alors ses valeurs dans l'ensemble des réels (température, concentration),
- soit *discrètes* : la variable prend ses valeurs dans un ensemble dénombrable, sous-ensemble des entiers naturels (nombre de pièces),
- soit *symboliques* : la variable prend ses valeurs dans un ensemble fini non structuré (par exemple la vanne TOR est caractérisée par ses états *ouvert* et *fermé*).

Les études menées pour concilier les composantes continue et discrète ont conduit à de nombreux formalismes. Il est évidemment impossible de passer en revue la totalité des approches proposées. Par conséquent, nous invitons le lecteur à lire les articles référencés pour plus d'informations. Selon l'approche de modélisation adoptée, une classification a été mise en place [Chombart *et al.*, 1996 ; Flaus, 1998 ; Zaytoon, 2001] :

- les approches qui peuvent être vues comme une extension des modèles des systèmes à dynamique continue pour intégrer le comportement discret (cas des bond-graphs [Buisson et Cormerais, 1998] à commutation par exemple). Ces approches consistent à introduire au sein du modèle continu des variables booléennes ou entières (par exemple VRAI/FAUX, 0/1).
- les approches qui peuvent être vues comme une extension des approches discrètes classiques pour inclure le comportement continu, telles que les réseaux de Petri hybrides [Alla *et al.*, 1992], les réseaux de Petri temporisés, temporels et stochastiques [Sifakis, 1977 ; Merlin et Segall, 1976 ; Florin *et al.*, 1991], les automates temporisés [Alur et Dill, 1994]. Ces formalismes tiennent compte des aspects quantitatifs du temps en ajoutant par exemple des contraintes temporelles.
- les approches mixtes qui combinent les modèles du discret et du continu dans une même représentation (par exemple, les automates hybrides [Alur *et al.*, 1995], les statecharts hybrides [Kesten et Pnueli, 1992], les réseaux de Petri prédicats-transitions différentiels [Champagnat *et al.*, 1998]). Chacune des deux parties (continue et discrète) est représentée de façon rigoureuse et explicite et leur collaboration est traitée au sein de l'interface qui les relie. Notons que la structuration se fait généralement à partir du discret.

Si la technique de résolution par simulation possède de nombreux avantages pour l'analyse, elle présente aussi des limitations génériques, quel que soit le type de simulation mise en œuvre.

Tout d'abord, le simulateur a une vue locale du problème du point de vue temporel. A une date t donnée pour l'horloge de simulation, on ne connaît pas les évènements qui vont se produire à $(t + \Delta t)$. Les décisions prises en cours de simulation sont liées à l'état du système, ou à des informations sur le passé de la simulation, mais pas sur les évènements à venir.

De plus, vu la complexité des modèles, les retours arrières ne sont pas possibles. Il sera donc nécessaire de faire une nouvelle simulation pour lever les violations de contraintes.

Dans ces conditions, quelle que soit la complexité des règles, la faisabilité et encore moins l'optimalité des solutions proposées ne peuvent être garanties [Pekny J.F., 1998]. La qualité du résultat obtenu dépend grandement du travail du concepteur, qui va essayer divers scénarios. Ce travail peut devenir fastidieux ; enfin les règles édictées sont spécifiques à un problème particulier.

3.6. LOGICIELS D'ORDONNANCEMENT

L'offre en logiciels d'ordonnancement nous paraît pouvoir être caractérisée par trois types de principaux de produits :

- des outils d'ordonnancement « de base » inclus dans des outils de *GPAO* (gestion de production assistée par ordinateur) ou des *ERP* (Enterprise Resource Planning),
- des logiciels d'ordonnancement spécialisés,
- des « solveurs » génériques permettant d'aborder les problèmes d'ordonnancement.

Ce paragraphe qui présente principalement les logiciels d'ordonnancement spécialisés est largement inspiré de [Grabot B., 2006], des informations complémentaires à ce rapide panorama ainsi que d'autres références bibliographiques peuvent y être trouvées.

3.6.1. Les outils d'ordonnancement des systèmes de GPAO et des ERP

Si la plupart des outils de gestion industrielle proposent en standard des modules *ordonnancement*, il ne s'agit la plupart du temps que d'outils de base, permettant de générer un ordonnancement traduit par un diagramme de Gantt figé sur lequel peu d'actions correctrices seront directement possibles. La méthode d'ordonnancement utilisée est souvent simple (jalonnement ou règles de priorité simples) et seules des contraintes de base seront prises en compte. Ces modules seront tout à fait suffisants pour des entreprises ayant des besoins basiques en ordonnancement, comme jalonner plus précisément les *OF* placés dans le plan de charge et préparer le lancement du travail dans un atelier simple.

Il convient toutefois de ne pas généraliser, certaines *GPAO* pouvant proposer en option, ou plus rarement en standard, un ordonnancement assuré par un logiciel spécialisé avec lequel elles s'interfacent.

3.6.2. Les logiciels d'ordonnancement spécialisés

En France, l'offre en logiciels d'ordonnancement est principalement caractérisée par des produits nationaux, souvent élaborés par des petites entreprises. Les spécificités de l'ordonnancement obligent en effet souvent à adapter un produit aux besoins précis du client, ce qui impose à la fois une certaine proximité et une grande flexibilité souvent caractéristique des petites structures.

Le pendant est que ces sociétés n'ont accès qu'à un marché limité (il est rare de voir un logiciel d'ordonnancement spécialisé diffusé à plusieurs centaines d'exemplaires, même au cours d'une longue carrière) et en conséquence les prix restent relativement élevés.

Nous évoquerons ici quelques logiciels particulièrement répandus en France et qui nous paraissent représentatifs des tendances du marché.

3.6.2.1. SipaPlus™

Issu de travaux universitaires réalisés au Laboratoire *GRAI* [Bérard Ch., 1983], *SipaPlus* est certainement le pionnier des logiciels d'ordonnancement spécialisés apparus sur *PC* dans les années 1980 et reste relativement utilisé en France. Ses caractéristiques originales à sa sortie (interface graphique interactif et structure de données en mémoire vive lui permettant un ordonnancement très rapide par règles de priorité) en font l'initiateur de l'ordonnancement « interactif » : un premier ordonnancement est proposé à l'utilisateur, qui dispose ensuite de points d'action au niveau du

diagramme de Gantt lui permettant d'améliorer la solution (déplacement d'opérations, extension du calendrier d'utilisation des ressources, etc.). Le logiciel fut par la suite complété par un plan de charge.

3.6.2.2. Ordo™(Ordo Software)

Ordo Software est lui aussi issu de travaux universitaires [Erschler J., 1976]. Son originalité est d'utiliser une technique d'ordonnancement par analyse sous contraintes lui permettant de générer des groupes d'opérations permutables sur les machines. Ces permutations possibles peuvent être utilisées au niveau du pilotage d'atelier pour réagir par rapport à l'occurrence d'aléas.

Les éditeurs parlent ainsi d'« ordonnancement temps réel ». On peut donc considérer ce logiciel comme caractéristique d'un souci d'ordonnancement « robuste » ou réactif.

3.6.2.3. Ortems™(Ortems)

Découlant une nouvelle fois de travaux de thèse [Al Kazzaz Y., 1989], *Ortems* est basé sur une technique d'ordonnancement originale combinant théorie des graphes (un graphe des opérations ordonnançables est bâti et maintenu au cours du processus d'ordonnancement) et gestion de priorités (des règles différentes pouvant être associées à chaque machine ou groupe de machine). Le support de ce graphe donne au produit une grande souplesse dans l'expression de contraintes très diverses, ce qui lui a permis des applications dans des domaines dépassant largement les ateliers manufacturiers (agroalimentaire, pharmaceutique, chimie, etc.). Le logiciel peut désormais être livré avec un logiciel additionnel appelé «Optimizer» permettant d'aider à son paramétrage [Talbi D. et al., 2004]

L'offre logicielle autour d'*Ortems* s'est considérablement accrue depuis sa sortie, tant vers le bas (*MES - Manufacturing Execution System*), ce qui est le cas de la plupart des logiciels, que vers le haut avec la proposition d'un *APS* (*Advanced Planning System*).

Tant du point de vue de ses fonctionnalités que de son prix, *Ortems* se situe actuellement dans la fourchette haute des logiciels d'ordonnancement.

3.6.2.4. CadPlan™(CadPlan Software)

Initialement implanté sur station de travail, *CadPlan* est depuis plusieurs années disponible sur *PC*. Il propose un calcul de charge et un ordonnancement par règles de priorités, ses principales caractéristiques étant :

- l'aspect « multiressources » et ressources « multicompétentes »,
- la gestion de tâches en réseau : capacité de gérer démontage (maintenance) et assemblage (production),
- la synchronisation avec les contraintes d'approvisionnement (contraintes de type dates au plus tôt),
- l'ordonnancement décentralisé : sur profil et droit d'accès, il est possible de prendre en compte des demandes de modifications par Intranet.

Il est à noter que *CadPlan* peut être directement relié à *Microsoft Project*, ce qui permet d'ordonnancer l'utilisation des ressources nécessaires à un projet en cohérence avec le découpage des tâches réalisé dans l'outil de gestion de projet.

3.6.2.5. Io™(Cesium)

Io est lui aussi un logiciel basé sur des règles de priorité. Les contraintes temporelles entre opérations sont toutefois gérées d'une manière originale qui élargit le domaine possible d'utilisation du

logiciel au domaine pharmaceutique ou à la chimie. L'ordonnancement peut être effectué de manière «régénérative» (on replanifie toutes les opérations), ou non-régénérative (certaines opérations placées a priori seront respectées). Le logiciel peut donc ponctuellement utiliser une stratégie de « placement d'ordres de fabrication ».

Par ailleurs, le parti pris d'interactivité avec l'utilisateur est ici basé sur une large possibilité de paramétrage de la visualisation de l'ordonnancement : Gantt-machine classique mais aussi visualisation de la charge, des en-cours, Gantt-ressources, etc. Chaque utilisateur peut ainsi paramétrer la manière avec laquelle l'ordonnancement est visualisé pour mieux diagnostiquer les problèmes et choisir plus efficacement les actions d'amélioration.

3.6.2.6. Preactor™(Preactor International)

Preactor est une exception dans cet échantillon de logiciels, puisqu'il a été développé en Grande-Bretagne. Basé sur des règles de priorité, *Preactor* se veut être un produit relativement simple mais ouvert pour être adapté à des utilisations particulières. Il est ainsi tout à fait envisageable de remplacer les règles d'ordonnancement implantées par d'autres au moyen de programmes extérieurs écrits en *C++* ou en *Visual Basic*.

3.6.3. Solveurs génériques

Il ne s'agit pas ici de logiciels dédiés à l'ordonnancement, mais d'outils permettant de résoudre un problème formulé de manière adéquate. Les outils les plus utilisés en ordonnancement sont à base de programmation linéaire, comme *Xpress-MP* de *Dash Optimization*, ou de propagation de contraintes : *Xpress-Kalis* de *Dash Optimization*, *CHIP* de *Cosytec* ou les produits d'*Ilog, Ilog Solver* et *ILOG Schedulers*, ce dernier intégrant déjà des primitives adaptées à l'ordonnancement.

Le contexte d'application de ces logiciels est très différent de celui des logiciels dédiés: ils seront soit utilisés pour résoudre un problème ponctuel (au prix d'un effort de modélisation relativement important), soit intégrés dans des progiciels comme moteur de calcul (la suite d'*Ilog* est par exemple très présente dans les *APS* du marché). Leur mise en œuvre demande bien sûr une expertise importante.

Comme on le voit (et même si ce panorama est loin d'être exhaustif), les logiciels dédiés d'ordonnancement, confrontés à la nécessité de proposer des solutions génériques, sont pour l'instant beaucoup plus orientés vers un ordonnancement interactif, offrant à l'expertise des chefs d'ateliers un cadre d'expression adéquat, que vers une tentative d'optimisation qui reste encore bien problématique dans des cas industriels. Au contraire, les solveurs génériques permettent d'arriver à des solutions optimales ou quasi optimales sur des problèmes spécifiques, et sont souvent utilisés pour des études d'ordonnancement ponctuelles, menées par des spécialistes.

3.7. ANALYSE COMPARATIVE DES MÉTHODES D'ORONNANCEMENT : TABLEAU RÉCAPITULATIF

Type	Approche	Méthode (exemple)	Référence	Principes	Avantages	Inconvénients
Méthodes Exactes	Procédure de Séparation et d'Evaluation	PSEP	[Lopez et al., 2001]	-méthode d'exploration par énumération implicite de l'espace de recherche - statégie d'exploration qui favorise les meilleures solutions d'abord	- méthode optimale	- modélisation du problème complexe et nécessité de décomposer le problème initial en sous-problèmes - temps de résolution important
		PSES	[Lopez et al., 2001]	- méthode d'exploration par énumération implicite de l'espace de recherche - statégie d'exploration de type profondeur d'abord et retour arrière		
	Théorie des Graphes	PERT	[Giard V., 1988]	- méthode de modélisation du problème d'ordonnancement sous forme de graphe faisant apparaître les conflits sous forme d'arcs à double direction - résolution du problème en fixant une orientation aux arcs identifiant les conflits et en les valuant	- méthode optimale - aide à la manipulation d'un ensemble important de données numériques - de nombreux algorithmes existent pour déterminer des chemins optimaux	- complexité de la modélisation du problème sous forme de graphe - difficulté dans le choix de l'algorithme à utiliser
	Programmation Dynamique		[Bellmann R. et al., 1974]	-méthode d'exploration par énumération implicite de l'espace de recherche - résolution ascendante grace au principe d'optimalité de Bellam	- méthode à vocation plus générale que les PSE	- taille des problèmes plus limitée que PSE
	Programmation Linéaire / Non Linéaire	PL PLNE (variables entières) PL-01 (variables binaires) PNL	[Guéret C. et al., 2000] [Van Hulle 1991]	- méthode classique de la recherche opérationnelle qui permet la modélisation d'un problème d'optimisation d'une fonction objectif de plusieurs variables en présence de contraintes sous la forme d'un programme mathématique - Le programme est dit linéaire si la fonction et les contraintes sont toutes des combinaisons linéaires de variables, non linéaire si non	- en formalisant un problème de manière générique, il est possible de rendre la saisie des données du problème relativement simple, via une interface graphique adaptée. - l'ajout de contraintes spécifiques à un problème peut se faire simplement sans modifier l'ensemble des autres contraintes	- l'étape de modélisation du problème peut s'avérer difficile pour des néophytes - la résolution d'un programme linéaire complexe peut demander un temps relativement long (de quelques dizaines de minutes à quelques heures). - Les PLNE et les PL en 0-1 sont plus difficiles à résoudre que les PL classiques. Les PNL sont encore plus difficiles.

Type	Approche	Méthode (exemple)	Référence	Principes	Avantages	Inconvénients
Méthodes Approchées 1/2	Approche Constructive	Méthodes de type Glouton	[Nawaz M. et al., 1983]	- méthode dont le principe est de diminuer progressivement la taille de l'espace de recherche en fixant à chaque étape la valeur d'une variable du problème. Notons que la construction d'une solution réalisable est statique, par conséquent, les prises de décisions intermédiaires ne sont jamais remises en cause.	- rapidité et simplicité	- faible qualité des solutions fournies
	Recherche Locale	Méthodes de descente	[Lopez et al., 2001]	- méthode qui explore l'espace X des solutions admissibles, en choisissant à chaque fois la meilleure solution voisine de la solution courante. Ce procédé continue aussi longtemps que la valeur de la fonction objectif diminue. La recherche s'interrompt dès lors qu'un minimum local de f est atteint.	- rapidité et simplicité	- suivant la taille et la structuration du voisinage résolution en des temps très variables - une méthode de descente peut fournir un minimun local
		Recuit Simulé	[Lopez et al., 2001] [Kirkpatrick et al., 1983]	- méthode qui démarre par une solution initiale admissible et continue l'exploration de l'espace d'états en effectuant des perturbations mineures à la solution courante. Si la solution voisine obtenue améliore le critère cherché alors elle est retenue. Si au contraire, elle provoque une dégradation du critère, elle est retenue avec une probabilité dont T est un paramètre inversement proportionnel au nombre d'itérations.	- résultats satisfaisants - permet de traiter des problèmes de grande taille	- difficulté à formaliser des modèles génériques pour la résolution de problèmes
		Recherche Tabou	[Glover F., 1986] [Glover F., 1989] [Glover F., 1990]	- méthode basée sur deux principes. Le premier consiste à améliorer à chaque itération la valeur de la fonction objectif en choisissant à chaque fois la meilleur solution voisine si elle existe. Si aucune amélioration n'existe dans le voisinage, le choix se fait sur le moins mauvais des voisins; le deuxième principe consiste à garder en mémoire les dernières solutions visitées et à interdire le retour vers celles-ci pendant un nombre d'itérations fixé	- résultats satisfaisants - permet de traiter des problèmes de grande taille	- difficulté à formaliser des modèles génériques pour la résolution de problèmes
	Approche Evolutive	Algorithmes Génétiques	[Holland J.H., 1975]	- méthode à base de population qui vise à imiter le processus d'évolution des espèces, dans lequel les individus les mieux adaptés à leur milieu ont une probabilité plus forte d'accéder à la reproduction et donc de transmettre leurs caractéristiques à leurs descendants	- très bons résultats aux problèmes d'optimisation	- difficulté à formaliser des modèles génériques pour la résolution de problèmes - un grand nombre de solutions sont susceptibles d'être évaluées -> temps de résolution important - paramètres de réglages nombreux et influant beaucoup sur la qualité des solutions finales - formalisation très contraignante souvent réservée à des spécialistes
		Colonies de fourmis	[Dorigo et al. 1999]	- méthode basée sur le comportement animal. Le paradigme des colonies de fourmis repose sur une analogie entre le problème optimisation et celui de la recherche d'un plus court chemin. Les fourmis en quête de nourriture déposent en effet des phéromones permettant à leurs consoeurs de les suivre. Ces phéromones s'évaporant avec le temps, et une fourmi ayant tendance à suivre la trace la plus forte, les chemins les plus courts, moins sujets à évaporation car renforcés par le passage de nombreuses fourmis, auront tendance à être privilégiés avec le temps.	- très bons résultats aux problèmes d'optimisation	- difficulté à formaliser des modèles génériques pour la résolution de problèmes - formalisation très contraignante souvent réservée à des spécialistes - paramètres de réglages nombreux et influant beaucoup sur la qualité des solutions finales

Type	Approche	Méthode (exemple)	Référence	Principes	Avantages	Inconvénients
Méthodes Approchées 2/2	Techniques issues de l'Intelligence Artificielle	Approches par contraintes	[Lopez et al., 2001]	- les techniques de propagation par contraintes visent à réduire l'espace des solutions à un problème en appliquant des contraintes restreignant l'ordre avec lequel les variables sont sélectionnées, et la séquence avec laquelle les valeurs possibles leur sont assignées	- simplifie la résolution d'un problème ou permet de démontrer l'absence de solutions	- difficulté à formaliser des modèles génériques pour la résolution de problèmes - techniques qui doivent souvent être complétées par d'autres pour aboutir à un ordonnancement précis, la propagation seule n'aboutissant le plus souvent pas à une solution unique
		Systèmes Experts	[Grabot B., 2006]	- méthode visant à imiter le raisonnement humain, considéré comme décomposable en parties élémentaires dénommées "règles de production", constituées dune partie "condition" et d'une partie "conséquence" (Si A alors B).	- utilisée pour modéliser des connaissances locales pour gérer quelques choix au sein de méthodes plus performantes (propagation de contraintes par exemple)	- la connaissance mise en jeu en ordonnancement semble peut se plier à un schéma aussi binaire que celui des règles de production - difficulté à formaliser des modèles génériques pour la résolution de problèmes - ne servent pas à bâtir directement une solution
		Logique Floue	[Lopez et al., 2001]	- méthode reposant sur la notion de sous-ensembles flous, visant essentiellement à formaliser de manière « informatisable » l'imprécision et l'incertitude, considérées comme inhérentes à beaucoup de connaissances humaines.	- utilisée pour intégrer des connaissances imprécises dans les systèmes experts - utilisation pour décrire les contraintes "flexibles" en propagation de contrainte	- ne sert pas à bâtir directement une solution
		Réseaux de Neurones	[Grabot B., 2006]	- méthodes appartenant à une autre branche de l'intelligence artificielle, puisqu'il s'agit non plus d'imiter le raisonnement humain mais la structure du cerveau, et en particulier ses capacités d'apprentissage	- Un réseau de neurones permettant d'approximer n'importe quelle relation entre entrées et sorties, une réponse pourra ensuite être fournie à des exemples non appris mais situés dans le domaine des exemples appris	- phase d'apprentisssage nécessaire - problème de la représentativité des exemples servant à l'apprentissage - ne servent pas à bâtir directemet un solution
		Raisonnement à Partir de Cas	[Grabot B., 2006] [Napoli A., 1999]	- méthode basée sur des cas stockés dans une «base de cas», décrits par les caractéristiques du problème, de son environnement et de la solution trouvée. Lorsqu'un problème nouveau est posé, la ressemblance entre les cas stockés et le nouveau cas est tout d'abord analysée. Le cas le plus proche est sélectionné et sa solution modifiée en fonction des différences entre les deux problèmes (incluant leur contexte).	- les applications les plus prometteuses visent davantage à imiter le comportement de l'expert humain lors de certains choix ponctuels, ou à paramétrer certaines variables générales du problème d'ordonnancement, plutôt qu'à générer complètement un ordonnancement	- ne servent pas à bâtir directemet un solution
		Systèmes Multi-Agents	[Archimède B. et al., 2001] [Shen W. , 1999]	- Les systèmes multiagents appartiennent à la branche de l'intelligence artificielle distribuée. Plutôt que de modéliser la résolution d'un problème sous forme centralisée, il s'agit de créer une communauté d'entités autonomes (les agents) permettant de résoudre des sous-problèmes du problème générique. La résolution du problème global doit donc émerger de la résolution des problèmes élémentaires. Il s'agit donc d'une forme de résolution de problème par décomposition, les agents représentant en général les différentes entités impliquées dans le problème.	- permet d'aborder de manière originale le problème de l'optimisation multiobjectifs, des agents pouvant être dédiés à la satisfaction d'un objectif donné, le compromis final émergeant des interactions entre agents	- développements informatiques importants pour implanter une commaunté d'agents ; - problèmes de validation et de performance intrinsèques à la notion d'émergence, en particulier au niveau de la convergence et de la validation du comportement global du système - temps de traitements, dus en particulier aux négociations entre agents ; - problèmes liés à une optimisation essentiellement locale car distribuée, alors que la plupart des objectifs d'ordonnancement ont un caractère global (minimiser les retards, les temps de cycle, les en-cours, etc.)
	Approches Simulatoires	Simulation à Evènements Discrets (SED) / Simulation Dynamique Hybride (SDH)	[Tittus, 1995] [Hétreux et al., 2003] [Lopez et al., 2001]	- L'approche simulatoire fait partie des méthodes dites à construction progessive d'un ordonnancement. Ce type de méthodes permet de déterminer rapidement un ordonnancement respectant les contraintes techniques.	- permet d'étudier la dynamique du comportement d'un atelier - amélioration itérative	- modèles simplifiés d'ateliers dans le cadre des SED - décisions prises en cours de simulation liées à l'état du système, ou à des informations sur le passé de la simulation, mais pas sur les évènements à venir. - dans le cadre des SDH vu la complexité des modèles, les retours arrières ne sont pas possibles. Il sera donc nécessaire de faire une nouvelle simulation pour lever les violations de contraintes. - quelle que soit la complexité des règles, la faisabilité et encore moins l'optimalité des solutions proposées ne peuvent être garanties [Pekny 98]. La qualité du résultat obtenu dépend grandement du travail du concepteur, qui va essayer divers scénarios.

3.8. MÉTHODE DE RÉSOLUTION RETENUE DANS CES TRAVAUX

Cette section se propose d'abord de réaliser un bilan sur les différentes méthodes de résolution présentées. Sur cette base, l'approche retenue dans ses travaux est indiquée.

3.8.1. Bilan sur les différentes méthodes de résolution

- Les méthodes exactes

L'apport des méthodes exactes à la résolution des problèmes d'ordonnancement reste important, malgré la simplification que nécessite l'utilisation des méthodes les plus simples. En effet, savoir simplifier à bon escient un problème d'ordonnancement est certainement une première condition pour arriver à une résolution efficace et rapide, même si cela peut demander une compréhension profonde du système à traiter. Par ailleurs, le progrès des techniques informatiques, tant au niveau matériel que programmation, a permis d'aborder la résolution de problème de taille qui semblait inabordable il y a quelques années. Ainsi, après avoir été quelques temps négligées, les méthodes optimales sont de nouveau à considérer dans la résolution des problèmes d'ordonnancement réels.

- Les méthodes approchées

Les métaheuristiques ont depuis longtemps été utilisées pour la résolution de problèmes d'ordonnancement. Dans tous les cas, deux problèmes principaux doivent être résolus :

- tout d'abord, celui de la modélisation du problème; en effet, toute métaheuristique est sous tendue par une formulation du problème qui peut s'avérer très contraignante, et réserve la plupart du temps leur emploi sinon à des spécialistes, du moins à des personnes « initiées ».
- Ensuite, comme nous l'avons déjà mentionné, le paramétrage des méthodes peut être long et difficile, alors qu'il conditionne grandement le résultat final.

Si les métaheuristiques commencent à être présentes dans les logiciels industriels de planification, c'est davantage au niveau moyen terme qu'à celui de l'ordonnancement. Il semble par exemple que plusieurs *APS* utilisent des algorithmes génétiques afin d'optimiser la structure ou le pilotage d'une chaîne logistique.

Les logiciels industriels utilisent de plus en plus de paramètres dont la valeur conditionne fortement la qualité de l'ordonnancement obtenu dans un cas d'application donné. L'implantation des métaheuristiques utilisées dans ce cas combine choix et réglage de paramètres afin de procurer des performances optimales. Les métaheuristiques en ordonnancement visent essentiellement à optimiser la qualité de l'ordonnancement obtenu, au prix d'un effort de modélisation et d'un temps de calcul pouvant être important. Même si de nombreux succès ont été constatés, leur utilisation reste délicate : les problèmes d'ordonnancement sont en effet connus comme peu stables, ce qui laisse présager dans des cas généraux des configurations « tourmentées » des espaces de solutions. Dans des cas complexes pour lesquels il est difficile de bâtir une borne inférieure de référence « réaliste », il est alors très facile de passer à côté d'un optimum, tout en améliorant de manière importante une solution initiale.

Rappelons enfin que, même si les stratégies d'intensification et de diversification peuvent être subtiles, les métaheuristiques n'intègrent habituellement pas de connaissances sur le domaine de l'ordonnancement, qui pourraient conduire plus rapidement à l'exploration de zones de solution prometteuses. L'hybridation des méthodes d'une part, mais aussi l'intégration de connaissances aux stratégies d'exploration semblent ainsi deux voies intéressantes dans ce domaine.

3.8.2. Approche retenue dans ces travaux

Dans le cadre de nos travaux, nous souhaitons mettre en place un module d'optimisation permettant d'établir des scénarios exploités pour la conduite de la simulation dynamique hybride des procédés batch. La méthode d'optimisation retenue doit permettre de proposer un cadre générique pour la résolution des problèmes d'ordonnancement des procédés, de faciliter la saisie des paramètres du problème au travers d'un graphe et d'inclure facilement des contraintes additionnelles comme la prise en compte de recyclages ou de démarrages de colonnes à distiller.

Une méthode comme la programmation linéaire, à la puissance d'expression importante puisque très flexible par rapport aux contraintes exprimables, ne souffre plus autant de ses limitations anciennes, au point d'être reprise dans les outils actuels de gestion de chaînes logistiques (APS-Advanced Planning Systems). Dans le cadre de nos travaux de recherche, nous nous appuierons aussi sur des **modèles de programmation linéaire en variables mixtes.** En effet, cette approche permet à la fois de proposer une modélisation générique, tout en laissant la possibilité d'ajouter des contraintes particulières par l'intermédiaire d'une formalisation graphique générique. L'intérêt d'un tel formalisme est de pouvoir renseigner les nombreux paramètres d'un problème via une interface graphique adaptée. Un premier prototype de logiciel permettant la saisie de ces données via une *IHM* est proposé dans ces travaux.

Relevant de la catégorie des méthodes exactes, la résolution d'un programme linéaire complexe peut demander un temps relativement long (de quelques dizaines de minutes à quelques heures). Cependant, l'objectif étant d'améliorer une solution vis-à-vis du procédé réel par la simulation, l'obtention d'une solution non optimale mathématiquement par rapport à un critère n'est pas rédhibitoire. Ce point est abordé dans le chapitre 6.

Enfin précisons que la résolution des programmes linéaires est réalisée avec un solveur du marché, *Xpress-MP* version *1.6.3* de *Dash Optimization.*

CHAPITRE 4

CHAPITRE 4

PROGRAMMATION LINÉAIRE ET ORDONNANCEMENT DES PROCÉDÉS BATCH

Résumé

Ce chapitre est consacré aux méthodes de Programmation Linéaire utilisées en industrie des procédés pour la résolution de problèmes d'ordonnancement. L'objectif principal de ce chapitre est de fournir un état de l'art de ce domaine complexe. Les méthodes de modélisation graphique de ces problèmes sont présentées. Les principales caractéristiques, les points forts et les limites des techniques existantes de modélisation et d'optimisation sont examinés. La modélisation des problèmes d'optimisation est abordée : les deux principales approches de modélisation en temps discret et en temps continu sont présentées en détail. Le choix du modèle utilisé dans les chapitres suivants est détaillé. Les extensions des éléments sémantiques et des apports en modélisation sont introduits.

Au cours des vingt dernières années, l'intérêt croissant pour les procédés batch s'est traduit d'une part, par un recours accru à la simulation dynamique pour mieux appréhender les phénomènes transitoires et d'autre part, par le développement de modèles d'ordonnancement de plus en plus sophistiqués pour en maîtriser la conduite.

En effet, les performances effectives de ces unités sont fortement liées à la fonction ordonnancement. Classés dans la catégorie des problèmes *NP*-difficiles, le chapitre 3 a présenté différentes approches pour les résoudre. Parmi celles-ci, la programmation linéaire en variables mixtes tient une place importante et plusieurs review peuvent être citées dans lesquelles différentes modélisations sont proposées [Mendez CA, 2006, Kallrath, 2002, Pekny et Reklaitis, 1998, Pinto et Grossmann, 1998, et Shah, 1998].

Méthode retenue dans nos travaux pour sa flexibilité et sa généricité, l'objectif de ce chapitre est de décrire les modèles mis en place dans notre outil. Pour cela, les formalismes graphiques sur lesquels s'appuie la modélisation des problèmes sont d'abord présentés. Les principales caractéristiques, les points forts et les limites de ces représentations sont abordées et une extension est proposée.

Ensuite, les deux principales approches de modélisation (temps discret et temps continu) sont décrites en détail.

Enfin, différentes extensions de ces modèles sont examinées pour aboutir au modèle retenu dans la suite de ces travaux.

4.1. MODÉLISATION DE LA RECETTE PRINCIPALE

4.1.1. Différentes notations selon la nature du procédé

La notion générale de recette a été introduite dans le chapitre 1 (section 1.2.5). S'appuyant sur une structure hiérarchisée définie par la norme *ISA SP88*, la recette peut être représentée de différentes manières selon la nature du procédé.

En effet, dans le domaine des procédés continus, la représentation d'une unité de production passe par la représentation d'un flowsheet, c'est-à-dire un graphe représentant les appareils et les flux de matière entre eux (les connexions). Le mode de fabrication étant continu, l'enchaînement des opérations unitaires correspond strictement à l'enchaînement des appareils. L'aspect procédural de la recette est assimilé au flowsheet et les contraintes de précédence sont alors relativement simples. Dans de tels procédés (Figure 4.1), l'ensemble des opérations unitaires se déroule par nature en parallèle et les flux de matière traversent les appareils de manière continue. Les nœuds représentent alors seulement des flux de matières qui se séparent ou se mélangent dans des appareils.

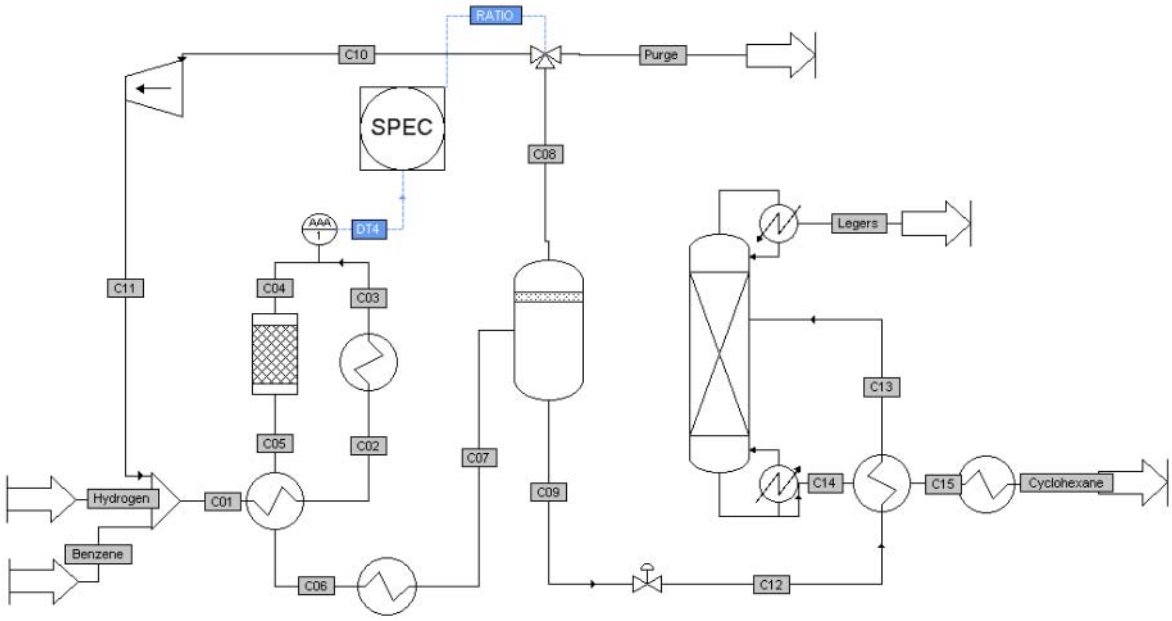


Figure 4.1 Flowsheet d'un procédé de production de cyclohexane sous Prosim

Par contre, dans le cadre d'un procédé discontinu, le flowsheet n'est plus suffisant. En effet, les opérations unitaires sont exécutées en séquence sur un lot donné selon un ordre qui peut être différent de la structure du procédé.

La partie procédurale devient alors un élément clé de la recette. A ce niveau, la norme *ISA SP88* préconise une hiérarchisation des informations selon cinq niveaux (cf. Figure 4.2) :

- Le *niveau procédure* définit un ensemble d'opérations à exécuter dans une ligne de production pour fabriquer un produit particulier.
- Le *niveau opération* définit l'ensemble des phases exécutées en séquence ou en parallèle dans une unité de production. Ce niveau correspond à la notion d'opération unitaire induisant une modification des propriétés physico-chimiques d'un produit (réaction, séparation, décantation, chauffe, etc.).
- Le *niveau phase* définit une succession de pas permettant de réaliser une fonction élémentaire. Une phase est limitée par des frontières définies par les instants où le déroulement des pas peut être interrompu en toute sécurité pour l'installation et sans dommage pour le produit.
- Le *niveau pas* désigne une séquence temporelle ou événementielle d'instructions à exécuter dans un module d'équipement en vue d'accomplir une tâche spécifique.
- Enfin le *niveau instruction* définit une commande élémentaire orientée équipement adressée à un élément ou à un composant afin qu'il effectue un traitement donné.

En ce qui concerne l'ordonnancement, le niveau de description requis correspond généralement au niveau procédure. Ainsi, seul est considéré l'enchaînement des opérations pour lesquelles les phases sont agrégées à travers des durées. Différentes représentations graphiques sont présentées dans la suite.

Notons que nous nous intéressons ici à la représentation graphique de la procédure en faisant abstraction du modèle mathématique associé originellement à ces représentations.

Notons dès à présent, que le formalisme utilisé pour modéliser la procédure dépend aussi du niveau de recette dans lequel on se situe. Ceci induit que des mécanismes doivent être définis afin de transformer ou traduire un formalisme dans un autre.

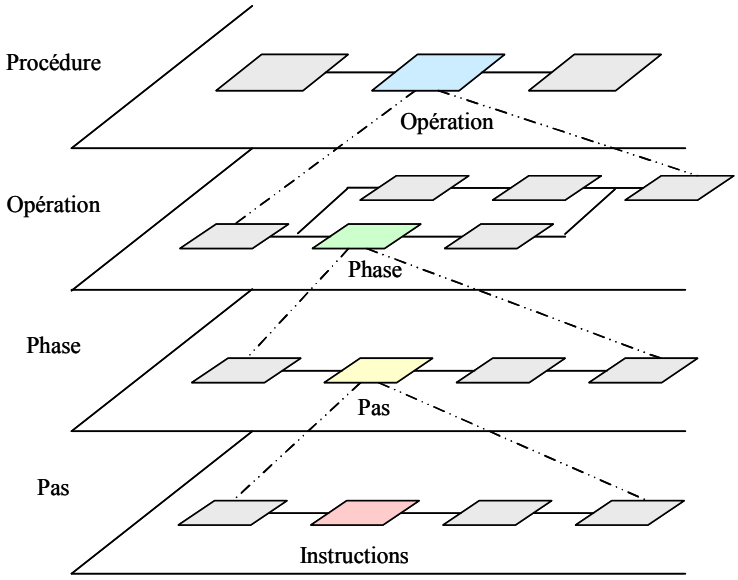


Figure 4.2 Hiérarchisation de la procédure

4.1.2. Représentation par graphe de précédence

Les graphes constituent un support rigoureux à la fois pour la vérification de la cohérence du problème posé et pour sa résolution. Les formulations courantes liées au problème d'ordonnancement passent par la définition d'un graphe de précédence manipulant des inégalités de potentiels [Lopez et al., 2001].

Ce graphe est constitué d'un ensemble de nœuds et de deux types d'arcs. Les nœuds représentent les tâches à réaliser. Les arcs conjonctifs valués de la durée de réalisation de la tâche d'où l'arc sort, représentent les contraintes de précédences. Les arcs disjonctifs à deux sens représentent les contraintes de ressources. Pour représenter le début et la fin de l'ordonnancement, deux nœuds fictifs sont rajoutés au graphe.

Le début et la fin de l'ordonnancement sont représentés respectivement par les nœuds fictifs « s » et « p ». Le fait de fixer un sens à chaque arc disjonctif réalise un ordonnancement [Roy B. et al., 1964].

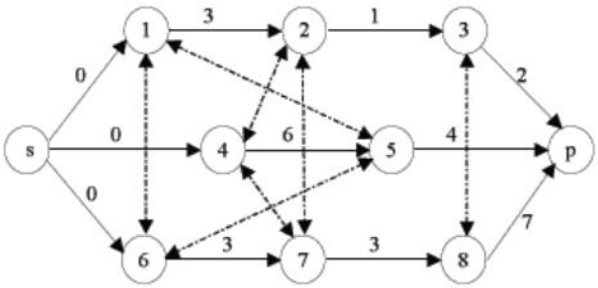


Figure 4.3 Exemple de graphe de précédence

Si un graphe de précédence est bien adapté pour représenter des contraintes temporelles, il ne permet pas de représenter de manière explicite les flux de matière transitant dans le système de production.

4.1.3. Représentation par un graphe de flux

Une représentation permettant d'expliciter les flux est la notion de graphe de flux. Prenons un exemple de graphe de flux (Figure 4.4). Les flux sortants d'une opération deviennent les flux entrant des opérations qui lui succèdent.

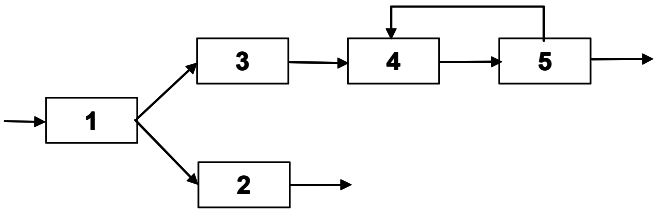


Figure 4.4 Représentation classique par graphe de flux

Cependant, cette représentation introduit un certain nombre d'ambigüités. Notamment,

- La tâche 1 fournit-elle deux produits différents, formant les entrées des tâches 2 et 3 ?
  ou alors
  y a t'il seulement un produit qui est partagé entre les tâches 2 et 3 ?
- La tâche 4 nécessite-t-elle deux produits différents, fournis respectivement par les tâches 3 et 5 ?
  ou alors
  nécessite-t-elle un seul produit qui peut être fourni soit par la tâche 2, soit par la tâche 3 ?

Une telle représentation ne permet pas de répondre à ces questions et est donc mal adaptée pour modéliser la procédure de procédés de type réseau.

4.1.4. Le formalisme *State Task Network*

4.1.4.1. Eléments sémantiques du formalisme *STN*

Le *State-Task Network* (*STN*) est une représentation graphique introduite par [Kondili 93] et est utilisée dans le cadre de gBSS (general Batch Scheduling Software) [Shah 95]. Un *STN* est un graphe possédant deux types de nœud qui alternent l'un avec l'autre. La Figure 4.5 propose une représentation graphique de ces éléments. Les ronds représentent l'*état* de la matière (*state*) et les rectangles, une *opération* de transformation de la matière (*task*). Les tâches sont nécessairement de type discontinu. Les flèches qui relient les états aux tâches définissent les *flux matières* utilisés en alimentation de la tâche et leur pourcentage dans la composition du lot de matière associé à la tâche.

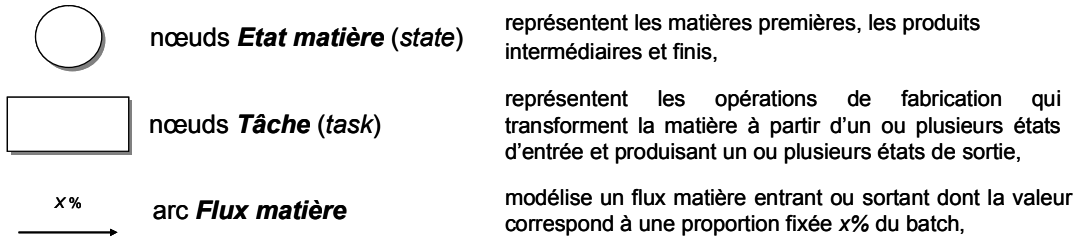


Figure 4.5 Arcs et nœuds définissant un STN

Cette représentation est intéressante car elle permet de présenter les états successifs, les tâches et les bilans matières de façon explicite par le graphe. Ces trois notions sont essentielles dans l'ordonnancement des procédés discontinus. La Figure 4.5 donne un exemple de procédure décrit dans le formalisme *STN*.

Les éléments sémantiques du *STN* permettent ainsi de lever les ambigüités du graphe de flux. Reprenons le graphe de la Figure 4.4, la représentation *STN* permet de différencier deux situations :

- **Interprétation 1**

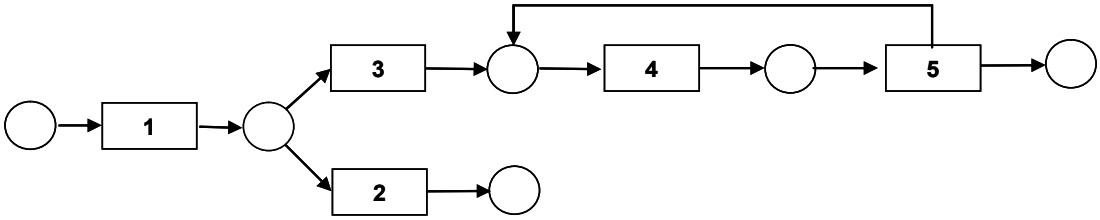


Figure 4.6 Première interprétation possible

Dans ce cas, le *STN* de la Figure 4.6 permet de représenter que :

- La tâche 1 fournit seulement un produit partagé par les tâches 2 et 3,
- La tâche 4 requiert seulement un produit d'entrée qui est fabriqué à la fois par les tâches 3 et 5.

- **Interprétation 2**

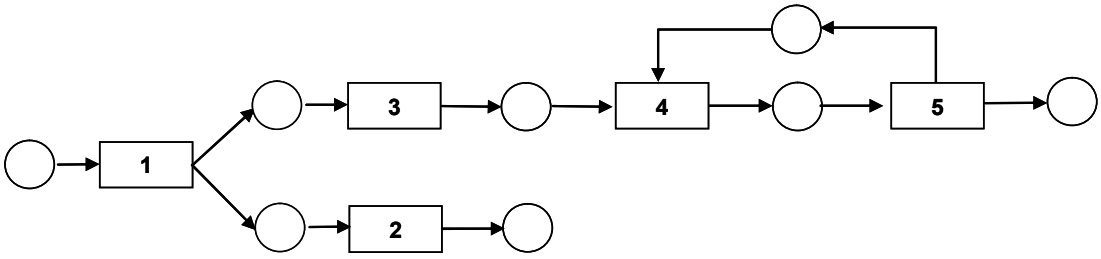


Figure 4.7 Deuxième interprétation possible

Ici, le *STN* de cette hypothèse permet de modéliser que :

- La tâche 1 fabrique deux produits différents, utilisés respectivement par les tâches 2 et 3.
- La tâche 4 requiert deux produits d'entrée, fabriqués respectivement par les tâches 3 et 5.

4.1.4.2. Règles de modélisation avec le formalisme *STN*

Soit le *STN* de la Figure 4.8 sur lequel sont illustrées les règles de modélisation.

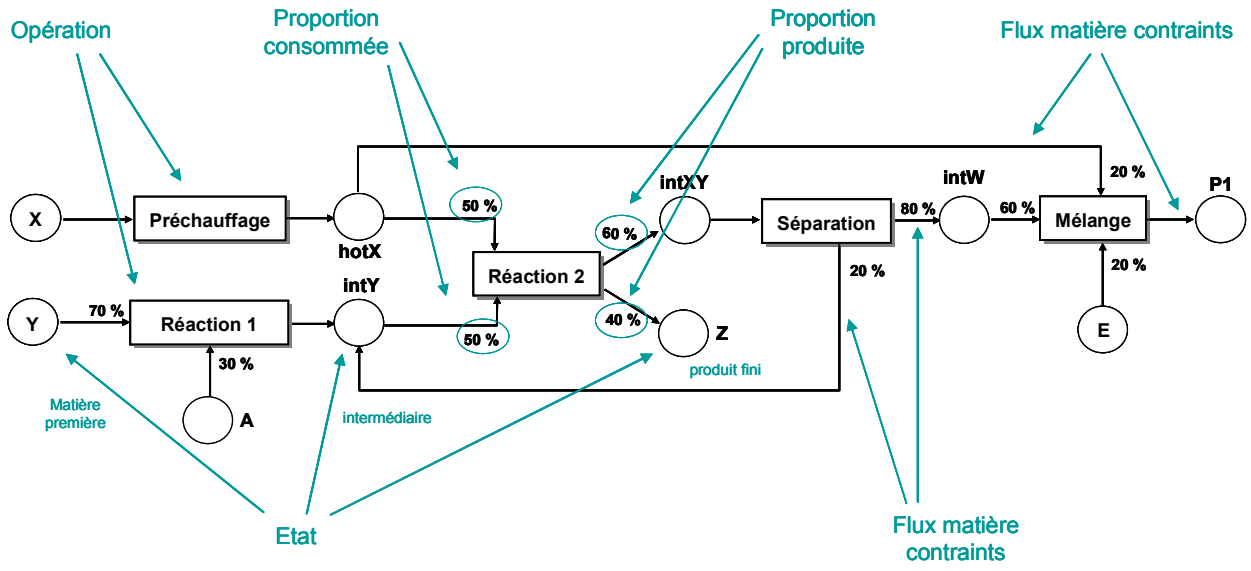


Figure 4.8 Exemple modélisation de STN

Règle 4.1 :

Une tâche en modélisation *STN* est un couple unique (opération unitaire / appareil).

Cette règle peut s'étendre de manière plus générale aux procédés. Dans ce cas, il est possible de définir le *batch* comme la décomposition d'une *tâche* en un ensemble de couples : *tâche* / *quantité de matière*.

Corollaire 4.1.1 : Une opération peut être représentée par une seule tâche si une seule ressource est utilisée. Elle doit être dupliquée en *n* tâches si *n* ressources peuvent réaliser cette même opération.

Règle 4.2 :

Un *STN* est un graphe orienté qui débute (respectivement se termine) toujours par un ou plusieurs nœuds de type *état*.

Règle 4.3 :

Un nœud de type *état* représente un état bien défini de la matière.

Dans un *STN*, un nœud de type *état* (cf. Figure 4.8) peut représenter :

- les matières premières (*X*, *Y*, *A*)
- les produits intermédiaires (*hotX*, *intY*, *intXY*, *intW*)
- les produits finis (*Z*, *E*, *P1*)

Corollaire 4.3.1 : Deux ou plusieurs flux entrant dans le même état (resp. sortant du même état) sont de même nature.

Par exemple : L'état *intY* est produit à la fois par l'opération Séparation et par l'opération Réaction 1.

Corollaire 4.3.2 : Si un mélange de différentes matières est nécessaire, alors cette opération constitue une tâche spécifique.

Par exemple : Les états *hotX*, *intW* et *E* sont mélangés par une opération de mélange spécifique.

Règle 4.4 :

Une *tâche* est un nœud représentant les opérations unitaires qui transforment un ou plusieurs *états* d'entrée dans un ou plusieurs *états* de sortie.

Une tâche a autant d'*états* d'entrée (resp. de sortie) que de types de matière d'entrée (resp. de sortie).

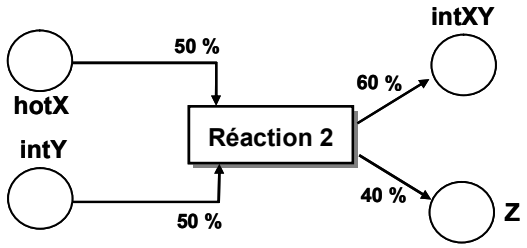


Figure 4.9 Représentation de la règle définissant les entrées / sorties d'une tâche

Règle 4.5 :

Les arcs portent la proportion apportée par chaque flux d'entrée (resp. de sortie) dans la masse d'un batch. La somme des proportions en entrée (resp. en sortie) doit être égale à 1.

Corollaire 4.5.1 : La valeur du flux d'entrée (resp. de sortie) entre une tâche et un état est égale à la valeur de la taille du lot (en masse) traité par la tâche multipliée par la proportion portée par l'arc d'entrée (resp. de sortie) considéré.

Dans de nombreux cas, un *état* peut être assimilé à la présence d'une cuve de stockage de capacité finie. Notamment, ces états peuvent représenter les cuves tampons qui stockent les intermédiaires réactionnels communs à plusieurs recettes. Il suffit de fixer la capacité de stockage à zéro pour définir une politique sans stockage intermédiaire.

4.1.4.3. Utilisation de la représentation STN

Le formalisme *STN*, peut être utilisé pour décrire la procédure au niveau de la recette générique, dans laquelle il n'y a aucune notion d'équipements. On ne s'intéresse alors qu'aux opérations de la recette. Mais s'il faut tenir compte des contraintes liées à l'utilisation des ressources la représentation *STN* est moins bien adaptée.

Dans l'exemple de la Figure 4.10, la recette est composée de *4 opérations* de réaction et permet la fabrication de *2 produits finis*. Le procédé est constitué de deux réacteurs : le réacteur 1 ne peut effectuer les réactions 1 et 3 et le réacteur 2, les réactions 2 et 4.

Ce type de contraintes de ressources n'apparaît aucunement sur le *STN*. Pourtant, elles ont un impact important sur le séquencement des tâches.

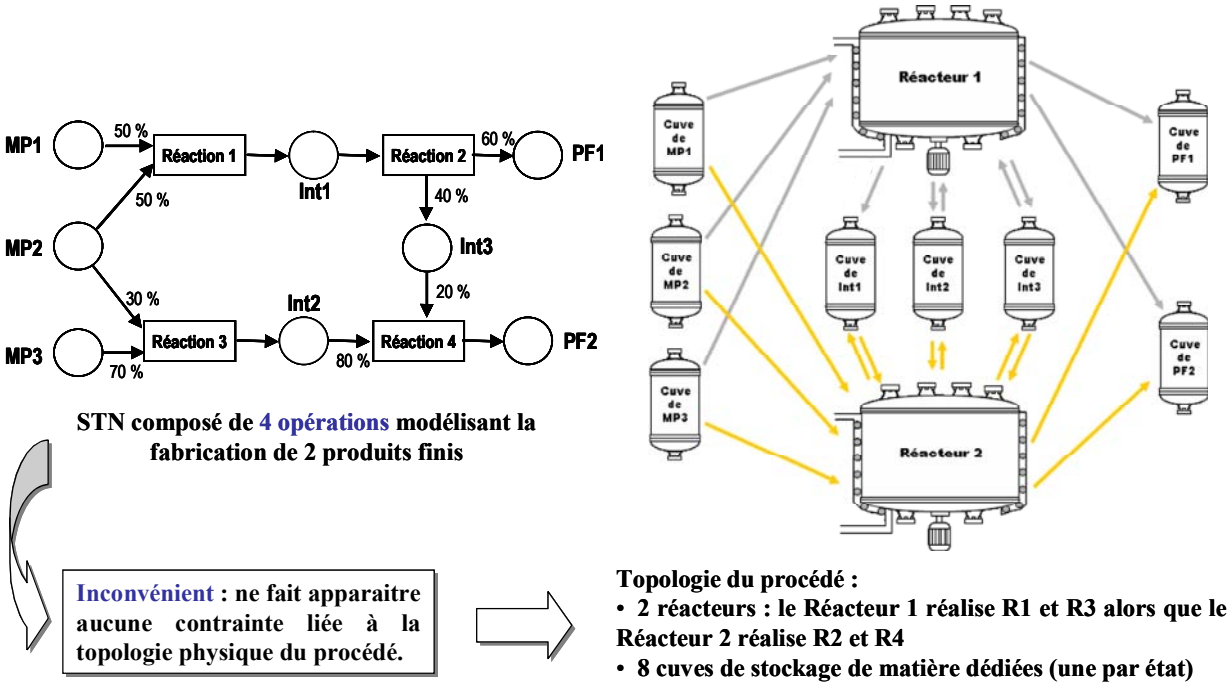


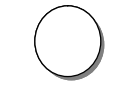
Figure 4.10 Contrainte de ressource non explicite par la modélisation STN

4.1.5. Le formalisme *Resource Task Network*

Afin de pouvoir représenter de manière explicite les contraintes liées à l'utilisation des ressources, une évolution du *STN* a été proposée par [Pantelides, 1994] : le Resource-Task Network (*RTN*).

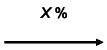
4.1.5.1. Eléments sémantiques du formalisme *RTN*

La représentation *RTN* est basée sur un traitement uniforme de l'ensemble des ressources nécessaires aux tâches. Il repose sur l'hypothèse que les tâches de traitement et de stockage consomment et/ou produisent des ressources renouvelables ou non, au cours de leur activité (voir Figure 4.12). Dans ce contexte, les cercles représentent non seulement les états de la matière, mais aussi d'autres ressources nécessaires à la réalisation d'une tâche comme les utilités et les appareils. La gestion des appareils est ajoutée ainsi que la modélisation des contraintes de ressources disjonctives.



nœuds ***Etat matière*** (*state*) — représentent des ressources cumulatives de type « matière » (MP, intermédiaires, PF) ou de type « utilité »,

nœuds ***Tâche*** (*task*) — représentent les opérations de fabrication qui transforment la matière à partir d'un ou plusieurs états d'entrée et produisant un ou plusieurs états de sortie,



arc ***Flux matière*** — modélise un flux matière entrant ou sortant dont la valeur correspond à une proportion fixée *x%* du batch,

nœuds ***Appareils*** (*Resource*) — modélisent les ressources disjonctives tels que les équipements sur lesquels les tâches peuvent être réalisées,

Arc disjonctif ***Utilise*** — modélise l'utilisation unitaire et exclusive d'une ressource pour la réalisation d'une tâche,

Figure 4.11 Arcs et nœuds définissant un RTN

La traduction de l'exemple de la Figure 4.11 au formalisme *RTN* est montré sur la Figure 4.12, les contraintes de partage de ressources apparaissent alors explicitement sur la représentation graphique.

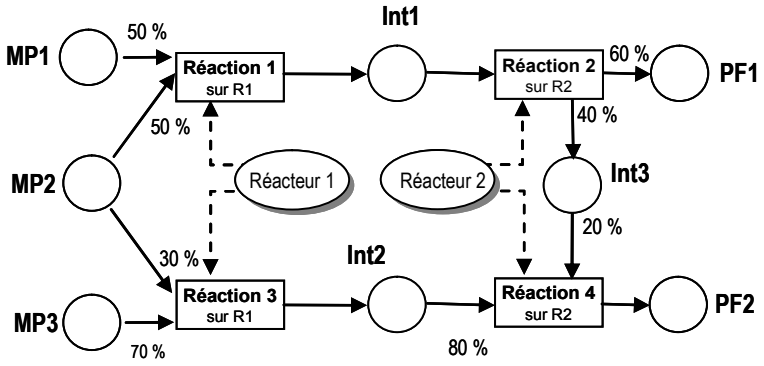


Figure 4.12 Représentation RTN de l'exemple détaillé Figure 3.9

Le formalisme *RTN* paraît donc particulièrement bien adapté pour décrire la procédure au niveau recette de site. En effet, le *STN* défini au niveau de la recette générique peut être instancié en un *RTN* en prenant en compte la configuration et la topologie de l'unité de production du site considéré.

Les caractéristiques des appareils et des équipements étant alors connus, des informations supplémentaires apparaissent sur les symboles des tâches et des states. Ces informations sont les suivantes :

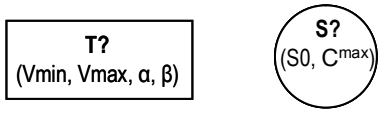


Figure 4.13 State et Task en RTN

4.1.5.2. Règle de modélisation avec le formalisme *RTN*

Les règles définies précédemment dans le cadre du *STN* restent valables. Néanmoins, une règle est ajoutée pour la gestion des ressources.

Règle 4.6 :

Une ressource (équipement) peut être utilisée par différentes tâches.

Corollaire 4.6.1 : Une tâche en modélisation *RTN* est l'élément unique de représentation du couple opération/ressource. La définition d'une tâche inclut implicitement la ressource utilisée (représentée via les *arcs disjonctifs Utilise*).

4.1.5.3. Instanciation d'un *STN* en *RTN*

Compte tenu de la définition du formalisme *RTN*, on peut montrer qu'à un *STN* donné peuvent correspondre différents *RTN* selon la topologie et la composition de l'unité de production.

Pour cela, le procédé décrit dans l'article de [Kondili *et al.,* 1993] est utilisé comme exemple illustratif.

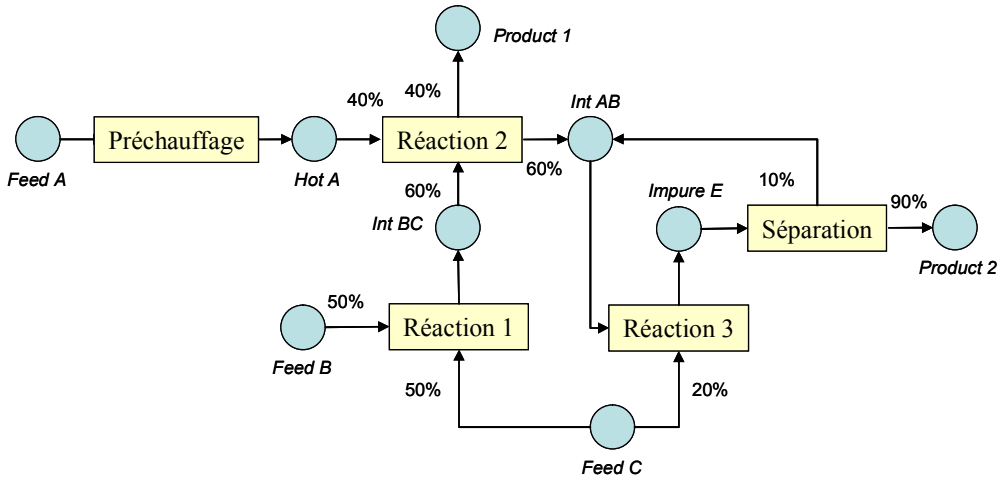


Figure 4.14 Exemple de STN

La recette générique de ce procédé est constituée de cinq opérations (Figure 4.14)

- un préchauffage
- trois réactions
- une séparation

Dans la suite l'impact de la topologie dans la modélisation RTN est montré. De plus, les notions de ressources disjonctives et de duplication de tâches, de modélisation RTN sont mises en œuvre conformément aux règles de modélisation *RTN*.

4.1.5.3.1. Topologie 1 : Cas d'un appareil utilisable pour réaliser plusieurs opérations différentes

Dans ce premier exemple de configuration du procédé (Figure 4.15), un seul réacteur réalise l'ensemble des *opérations* de *Réaction 1, 2 et 3*. Le partage de la ressource *Réacteur 1* se fait par l'intermédiaire de trois arcs disjonctifs *Utilise* qui indiquent que les tâches T2, T3, T4 (cf. Figure 4.16) sont en exclusion mutuelle, c'est-à-dire qu'à un instant t donné, seule une de ces tâches peut être réalisée sur la ressource Réacteur 1. Pour les *opérations* de *Séparation* et *Préchauffage*, une ressource est dédiée à chaque opération.

Par ailleurs, chaque opération ne peut être réalisée que par une seule ressource. Par conséquent, nous n'avons pas besoin de dupliquer les tâches car les opérations ne pourront pas être réalisées en parallèle sur des ressources différentes (cf. Règle 4.6). Nous obtenons finalement 5 tâches en modélisation *RTN* (Figure 4.16).

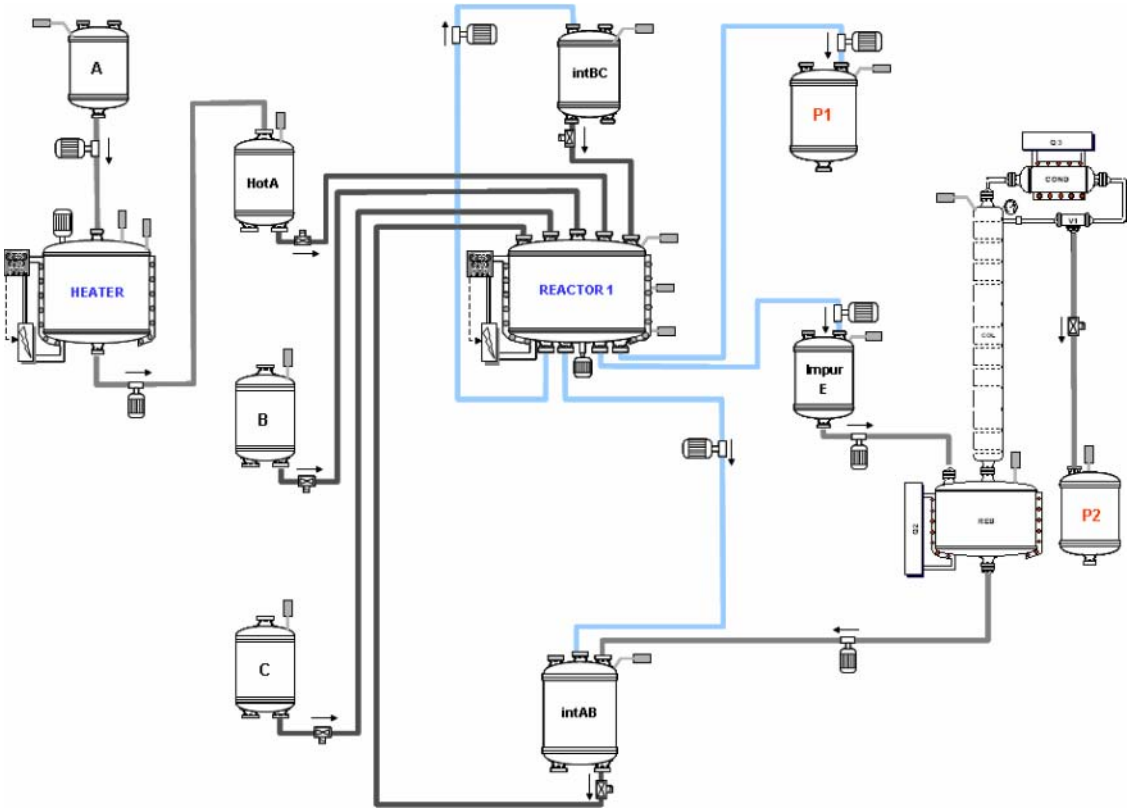


Figure 4.15 Topologie 1 de procédé

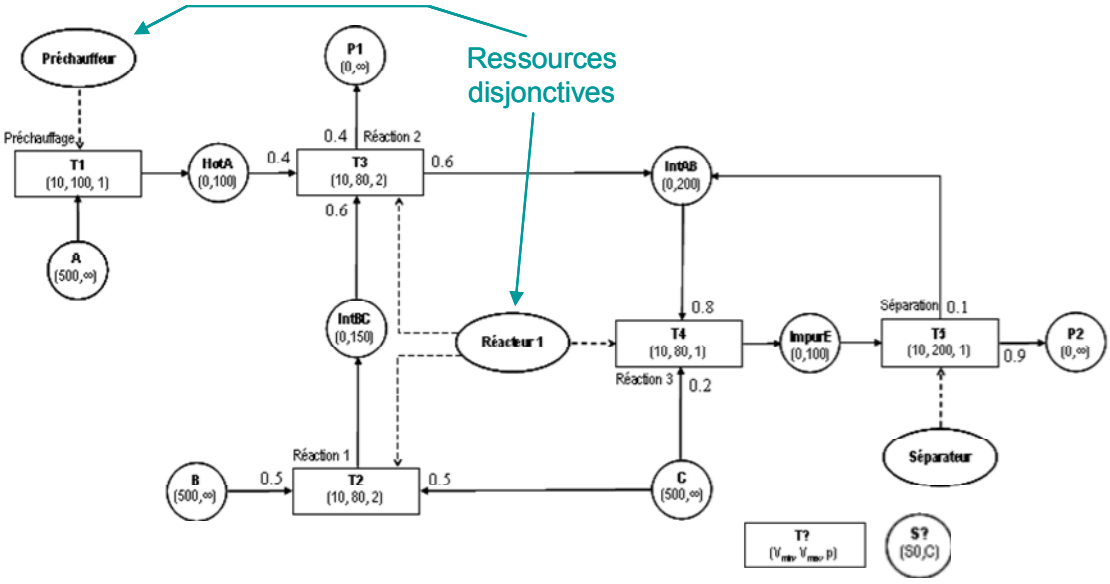


Figure 4.16 RTN *obtenu par instanciation du* STN *avec l'unité de la figure*

4.1.5.3.2. Topologie 2 : Cas de deux appareils utilisables pour réaliser plusieurs opérations différentes

Dans ce deuxième exemple de configuration de procédé (Figure 4.17), un premier réacteur réalise les *opérations* de *Réaction 1 et 2* tandis qu'un second réacteur réalise l'*opération* de *Réaction 3*. Le partage de la ressource *Réacteur 1* entre les deux opérations de *Réaction 1 et 2* se fait par l'intermédiaire de deux arcs *Utilise*. Pour les *opérations* de *Réaction 3*, *Séparation* et *Préchauffage*, une ressource est dédiée à chaque opération.

En appliquant les règles de construction, nous obtenons finalement 5 tâches en modélisation *RTN* (Figure 4.18). Cependant, le *RTN* est bien différent de celui de la topologie 1.

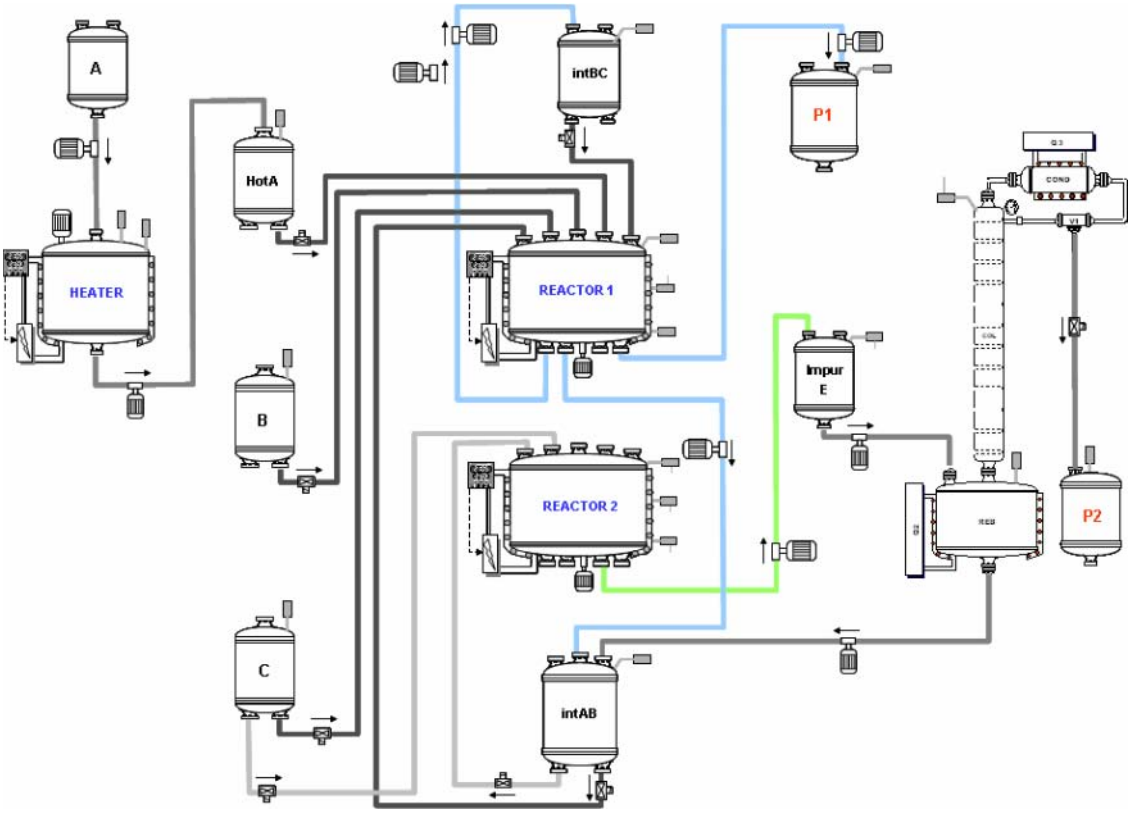


Figure 4.17 Topologie 2 de procédé

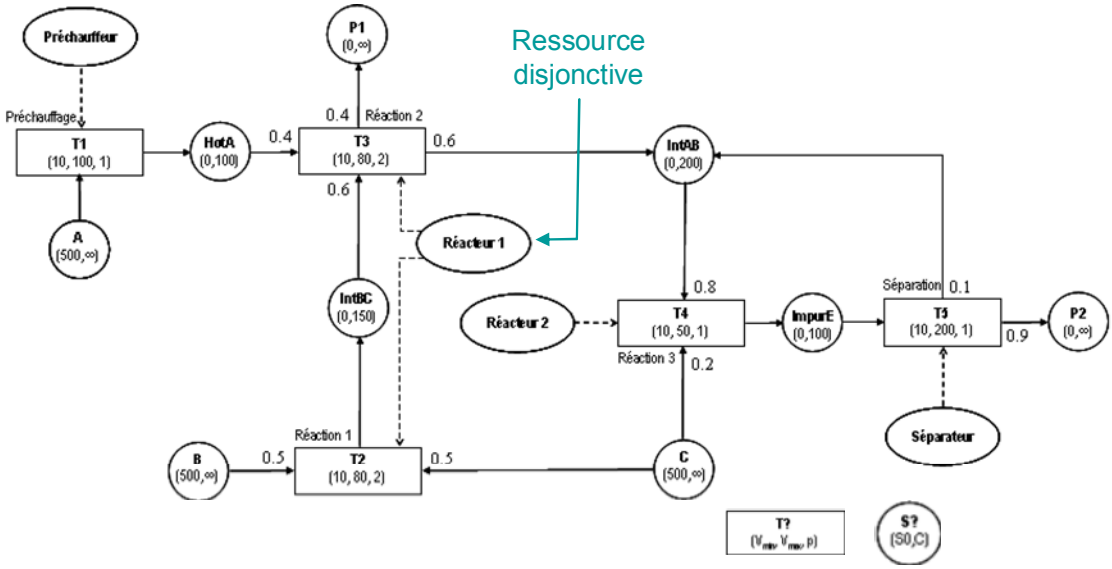


Figure 4.18 RTN *obtenu par instanciation du* STN *avec l'unité de la figure*

4.1.5.3.3. Topologie 3 : Cas d'opérations réalisables dans plusieurs appareils différents

Dans ce troisième exemple de configuration de procédés (Figure 4.19), la connectivité est plus complexe, les deux ressources *Réacteurs 1 et 2* peuvent réaliser les trois *opérations* de *Réaction 1,2 et 3*.

Dans ce cas, une tâche est définie pour chaque combinaison (Opérations, Ressources) de Réaction 1, 2 et 3, ce qui conduit à la duplication des ces trois tâches.

Une tâche en *RTN* correspond à une opération sur un appareil. Dans le cas présent chaque opération de réaction peut être réalisée par deux ressources. Par conséquent, nous obtenons au final 8 tâches en modélisation *RTN* (Figure 4.20).

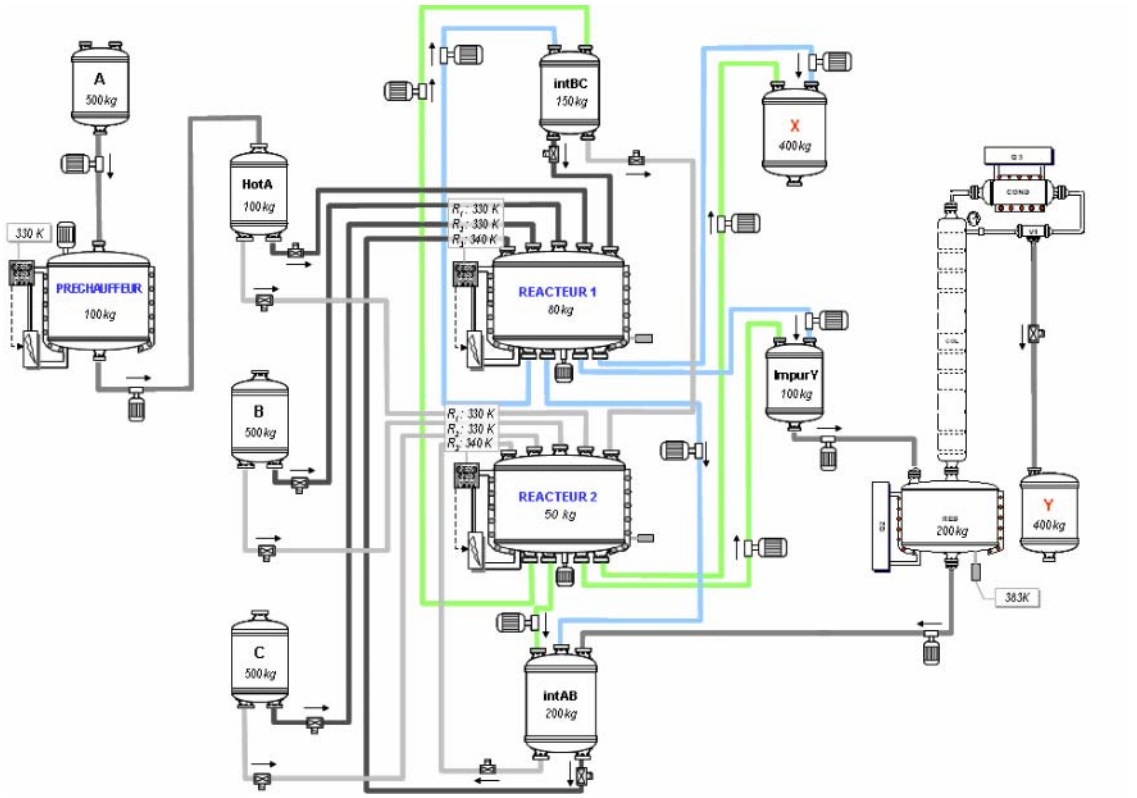


Figure 4.19 Topologie 3 de procédé

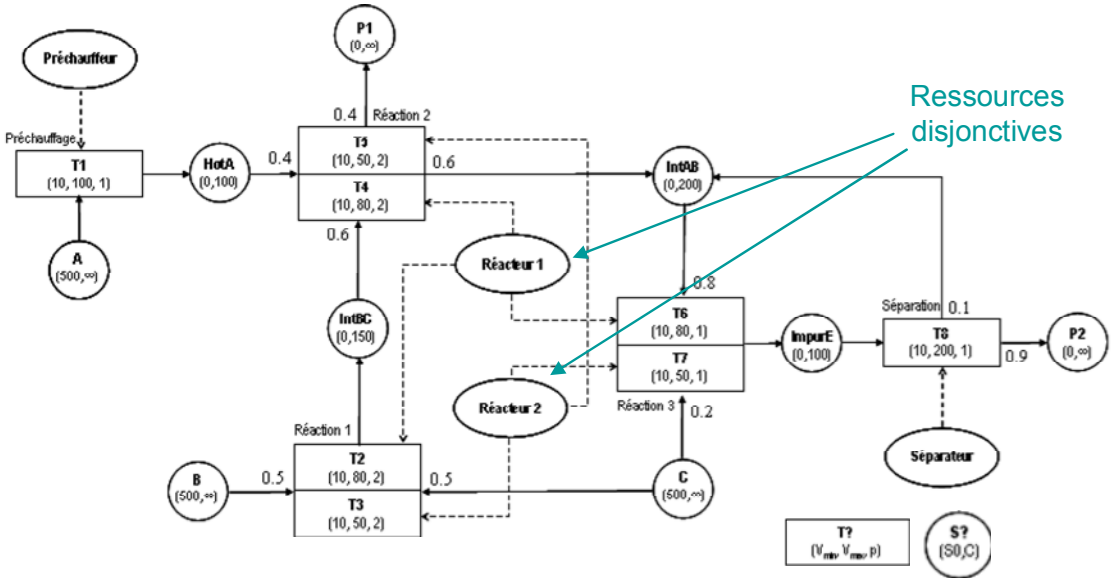


Figure 4.20 RTN *obtenu par instanciation du* STN *avec l'unité de la figure*

4.2. MODÈLES MATHÉMATIQUES DE BASE POUR L'ORDONNANCEMENT DES PROCÉDÉS BATCH

Les différentes contraintes à prendre en compte dans les modèles d'ordonnancement de procédés batch, ont été présentées dans la section 3.4.

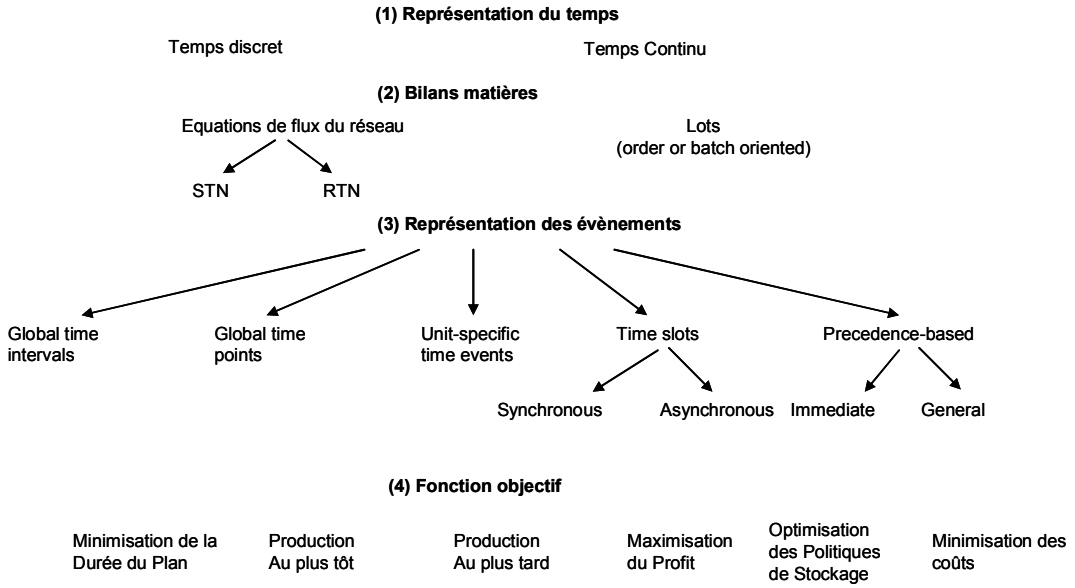


Figure 4.21 Roadmap des caractéristiques des modèles d'ordonnancement de procédés batch

La Figure 4.21 donne une classification des différents éléments qui caractérisent un modèle d'optimisation *MILP*. Ces différentes alternatives de formulation du même problème ont généralement

un impact direct sur les performances de calcul, les capacités et les limites résultant du modèle d'optimisation [Mendez CA, 2006].

Chacune des options de modélisation qui est présentée est en mesure de répondre à un sous-ensemble de caractéristiques décrites dans la Figure 3.3 du chapitre 3.

Les quatre points principaux de cette *roadmap*, à savoir la *représentation du temps*, les *bilans matière*, la *représentation des évènements* et la *fonction objectif* sont décrits en détail dans la suite de cette section.

4.2.1. Représentation du temps

La représentation du temps est un des points les plus importants dans l'écriture d'un programme linéaire. Selon que les événements du calendrier ne peuvent avoir lieu qu'à certains moments prédéfinis, ou peuvent survenir à tout moment sur l'horizon de temps, les formulations des modèles d'optimisation peuvent être classées en *formulation à temps discret* ou en *formulation à temps continu*.

- **Formulation en temps discret**

Les modèles en temps discret sont basés sur :

- Une division de l'horizon de temps de l'ordonnancement en un nombre fini d'intervalles de temps avec une durée prédéfinie,
- Une synchronisation des évènements comme le début ou la fin de tâches aux frontières de ces périodes de temps.

Par conséquent, les contraintes d'ordonnancement doivent seulement être contrôlées à des dates connues dans le temps, ce qui réduit la complexité du problème et rend le modèle de structure plus simple et plus facile à résoudre, en particulier lorsque des contraintes de ressources et de stockage doivent être prises en compte. Cependant, ce type de modélisation du problème a deux inconvénients majeurs :

- Tout d'abord, la taille du modèle mathématique ainsi que son efficacité de calcul dépendent fortement du nombre d'intervalles de temps choisi. Ce nombre d'intervalles est défini en fonction du problème, des données et de la précision désirée de la solution.
- De plus, il peut générer une solution sous-optimale, ou rendre le problème infaisable (sans solution), du fait de la réduction des intervalles de décisions.

En représentation *STN*, la notion d'opération, dans le sens ensemble d'étapes se faisant au sein d'un même appareil, n'existe pas. Les ressources utilisables doivent être spécifiées sur chaque « tâche », même si ce sont les mêmes que la tâche précédente. Par conséquent, dans les modèles en temps discret, les transferts ne peuvent pas être représentés : une seule ressource peut être affectée à chaque tâche alors qu'un transfert en requiert au moins trois simultanément (une cuve amont, un organe de transfert et une cuve aval). Les transferts sont alors considérés avoir une durée nulle. Toute tâche est supposée être une activité productive qui change systématiquement l'état de la matière entrante, ce qui pose un problème de représentation pour les nettoyages par exemple.

En dépit d'être une version simplifiée du problème d'ordonnancement initial, les formulations en temps discret se sont révélées être très efficaces, adaptables et pratiques pour une grande variété d'applications industrielles, en particulier dans les cas où un nombre raisonnable d'intervalles est suffisant pour obtenir une bonne représentation du problème.

Un exemple de formulation en temps discret peut être consulté en Annexe C, il s'agit du modèle « Global time intervals » en formulation *STN*.

- **Formulation en temps continu**

Afin de surmonter les limites des formulations précédentes, une grande variété de modèles d'optimisation emploie une représentation continue du temps. Dans ces formulations, les « *dates de décision* » sont explicitement présentées comme un ensemble de variables continues définissant les dates exactes d'occurrence des événements (début ou fin d'une tâche).

Dans le cas général, l'utilisation de ces variables « *date de décision* » permet d'obtenir une réduction significative du nombre de variables du modèle et en même temps, des solutions plus flexibles en termes de dates peuvent être générées. En contre-partie, les problèmes d'allocation de ressources et de capacité des stocks sont traduits par des contraintes plus complexes impliquant de nombreuses équations formulées avec des « *Big-M* ». Ces dernières tendent à accroître la complexité du modèle et donc à limiter les capacités de résolution de ces méthodes.

4.2.2. Bilans matière

Le traitement des lots et les tailles de lot donnent lieu à deux catégories de modèle d'optimisation. La *première catégorie* se réfère aux *approches monolithiques*, qui déterminent grâce à un même modèle le nombre et la taille des lots, l'allocation des ressources de fabrication et l'ordonnancement des tâches. La *deuxième catégorie* se réfère à une *approche séquentielle* dans laquelle le nombre et la taille des lots sont calculés séparément de l'allocation des ressources et de l'ordonnancement.

Ces deux catégories de modèles peuvent être considérées comme l'un des modules d'une approche plus globale de la planification de la production, largement utilisée dans l'industrie, qui décompose le problème en deux étapes, la *planification* et l'*ordonnancement* de la production.

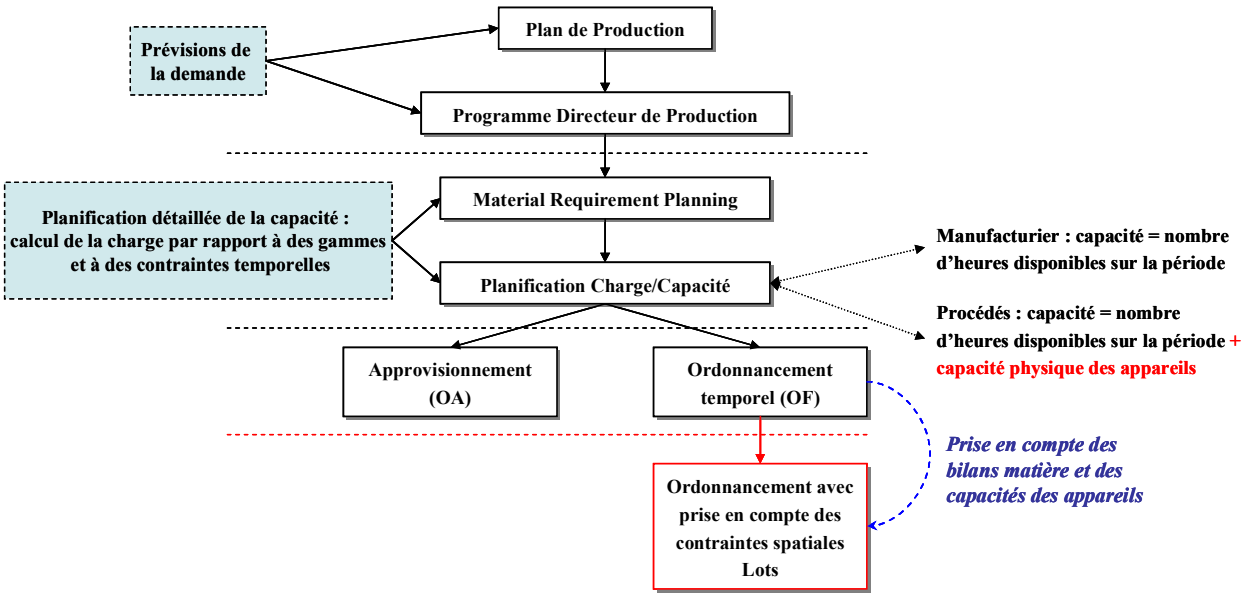


Figure 4.22 Intégration du MRP avec les contraintes spatiales des procédés dans l'organisation hiérarchique d'un système de production

Le *problème de planification* traduit les exigences principales de production des lots et s'attache à optimiser certains critères propres à l'atelier de production. Cette étape de planification (Figure 4.22) se base sur un calcul de besoin communément appelé MRP et établit un *plan prévisionnel de production* (liste d'Ordres de Fabrication ou OF), un *plan d'approvisionnement* (liste d'Ordres d'Approvisionnement ou OA) ainsi qu'un *Plan de charge* par Poste de charge. Chaque OF définit un volume de produit à fabriquer sur une période donnée. L'approche MRP2 permet d'obtenir directement les lots à produire (OF) à condition de se trouver dans un cas similaire aux ateliers manufacturiers pour lesquels les contraintes de temps suffisent à garantir la faisabilité d'un *Plan de charge*. Dans ce cas les *n* tâches de l'OF correspondent aux *n* phases de la gamme opératoire. En effet, les contraintes temporelles seules permettent de gérer la réalisation d'un lot dans le cas de production manufacturière.

Dans le cas des procédés, une approche MRP (Material Requirement Planning) uniquement basée sur une approche temporelle de l'outil de production ne permet pas d'obtenir directement la taille des lots. Il est en effet nécessaire de coupler une deuxième étape prenant en compte les contraintes spatiales des outils de production considérés via des bilans matière. Dans ce cas, les *n* tâches de l'OF correspondent aux *n* phases de la gamme opératoire qui doivent être redécoupées en lots de volume compatible avec la capacité physique des appareils.

La deuxième étape de résolution permet donc :

- de redécouper les *lots « temporels »* en *lots « physiques »*, en effet les dates définies dans les lots temporels correspondent aux dates de disponibilité d'un volume de produit donné,
- d'organiser l'ordonnancement des *lots « physiques »* en tenant compte des contraintes externes de disponibilité des produits calculées par le MRP et des contraintes spatiales internes du procédé.

Ainsi, lors d'un ordonnancement, une même tâche peut être exécutée plusieurs fois sur une même campagne de production. Par exemple, si un OF de 1200kg doit être réalisé sur une ressource de capacité maximale de 600kg alors au minimum 2 batch doivent être lancés (avec par exemple 2 batch de 600kg. Néanmoins, une autre solution peut être de lancer 4 tâches avec des batch respectivement de 200kg, 500kg, 300kg et 200kg. Ceci explique pourquoi, il est nécessaire de prendre en compte les bilans matière dans le modèle d'ordonnancement. Il y a, en effet, une contrainte temporelle (durée du plan) mais aussi spatiale (capacité des cuves).

4.2.2.1.1. Approches séquentielles

La section 1.1.2 a montré qu'une approche souvent utilisée dans l'industrie consiste à décomposer le problème de production en deux étapes, la *planification* et *l'ordonnancement*.

Pour répondre à la double contrainte de l'ordonnancement des procédés, les approches séquentielles calculent d'abord le nombre et la taille des lots (heuristique de lot sizing) puis dans un second temps réalisent le calcul du séquencement et l'allocation des ressources.

4.2.2.1.2. Approches monolithiques

Ces méthodes permettent de résoudre des problèmes d'ateliers très généraux et mettant en jeu des recettes complexes (topologie en réseau). Ces approches s'appuient généralement sur des représentations de type *STN* ou *RTN*.

L'approche retenue dans ces travaux se base sur une approche monolithique basée sur une représentation *RTN*.

4.2.3. Fonction Objectif

Il existe différentes manières d'évaluer la qualité d'une solution d'un ordonnancement. Cependant, le critère sélectionné pour l'optimisation a une incidence directe sur les temps de résolution du modèle. De plus, certaines fonctions objectif peuvent être très difficiles à implémenter pour une représentation donnée des évènements, nécessitant un grand nombre de contraintes additionnelles.

4.2.4. Représentation des évènements

Les modèles d'ordonnancement reposent non seulement sur une représentation temporelle et des bilans matière, mais aussi sur la manière de gérer les évènements.

Un évènement est généralement le début et/ou une fin de tâche qu'il faut placer dans le temps, le principal objectif étant de garantir que la capacité maximale des ressources partagées n'est jamais dépassée.

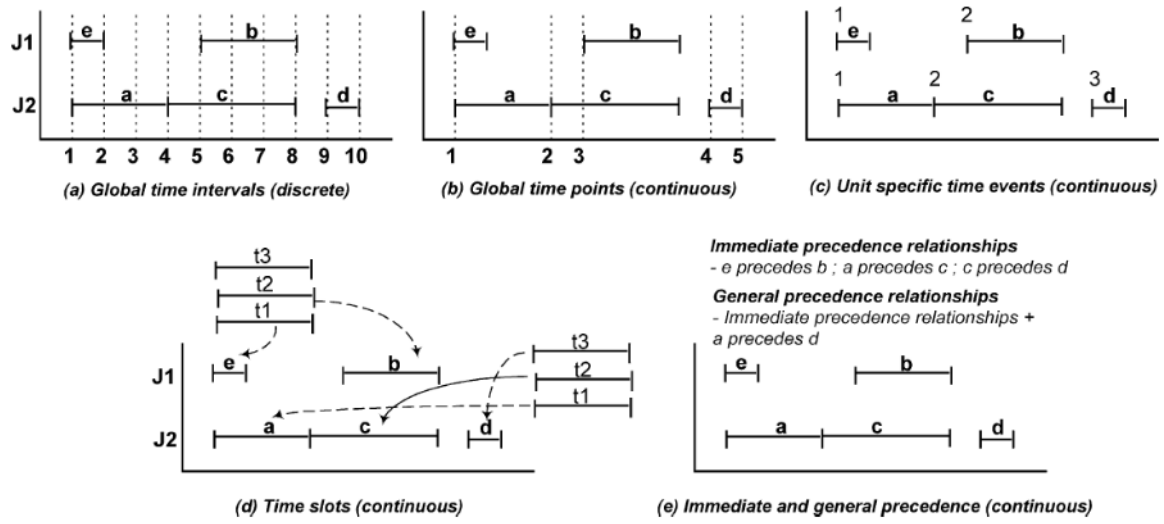


Figure 4.23 Différentes méthodes de gestion des évènements

La Figure 4.23 illustre une représentation schématique du même ordonnancement obtenu par des représentations différentes des évènements. L'exemple considéré concerne cinq lots (a, b, c, d, e) répartis en deux unités (J1, J2).

Pour représenter cette solution, les différentes formulations imposent:

(a) 10 fixed time intervals,
(b) 5 variable global time points,
(c) 3 unit-specific time events,
(d) 3 asynchronous time slots pour chaque unité,
(e) 3 immediate precedence relationships ou 4 general precedence relationships

Ces concepts peuvent être regroupés en cinq types différents de représentation des évènements (cf. Figure 4.21 et Figure 4.23). Ces types de représentation ont été largement utilisés pour développer un grand nombre de formulations mathématiques des problèmes d'ordonnancement de procédés batch.

Bien que certaines représentations soient plus globales que d'autres, elles sont généralement orientées vers la résolution, soit d'ateliers de fabrications en réseau nécessitant des équations de flux, soit d'ateliers séquentiels. Mais toutes ont une approche commune orientée *production par lot*. Le Tableau 4.1 [Mendez CA, 2006] résume les principaux critères de modélisation et les caractéristiques des problèmes liés au choix de représentation des évènements. La ligne *Critical modelling issues* fait référence aux paramètres de modélisation qui ont un impact fort sur la taille du modèle et la difficulté de résolution. De même, la ligne *Critical problem features* fait référence à des typologies de problèmes qu'il serait maladroit de modéliser avec les concepts de base d'une modélisation donnée.

Characteristic	Discrete time models	Continuous time Models					
Event representation	Global time intervals	Global time points	Unit-specific time events	Time slots*	Unit-specific immediate precedence*	Immediate precedence*	General precedence*
Main decisions	Lot-sizing, allocation, sequencing, timing				Allocation, sequencing, timing		
Key discrete variables	W_{ijt} defines if task I starts in unit j at the beginning of time interval i.	Ws_{in} / Wf_{in} define if task i starts/ends at time point n. $W_{inn'}$ defines if task i starts at time point n and ends at time point n'.	Ws_{in} /W_{in} / Wf_{in} define if task i starts/is active/ends at event point n.	W_{ijk} define if unit j starts task i at the beginning of time slot k.	$X_{ii'j}$ defines if batch i is processed right before of batch i' in unit j. XF_{ij} defines if batch i starts the processing sequence of unit j.	$X_{ii'}$ defines if batch i is processed right before of batch i'. XF_{ij} / W_{ij} defines if batch i starts/is assigned to unit j.	$X'_{ii'}$ define if batch i is processed before or after of batch i'. W_{ij} defines if batch i is assigned to unit j
Type of process	General network				Sequential		
Material balances	Network flow equations (STN or RTN)	Network flow equations (STN or RTN)	Network flow equations (STN)		Batch-oriented		
Critical modeling issues	Time interval duration, scheduling period (data dependent)	Number of time points (iteratively estimated)	Number of time events (iteratively estimated)	Number of time slots (estimated)	Number of batch tasks sharing units (lot-sizing) and units	Number of batch tasks sharing units (lot-sizing)	Number of batch tasks sharing resources (lot-sizing)
Critical problem features	Variable processing times, sequence-dependent changeovers	Intermediate due dates and raw-material supplies	Intermediate due dates and raw-material supplies	Resource limitations	Inventory, resource limitations	Inventory, resource limitations	Inventory

* Batch-oriented formulations assume that the overall problem is decomposed into the lot-sizing and the short-term scheduling issues. The lot-sizing or "batching" problem is solved first in order to determine the number and size of "batches" to be scheduled.

Tableau 4.1 Caractéristiques générales des principaux modèles d'optimisation d'après [Mendez C.A. et al.*, 2006]*

4.2.4.1. Représentation en temps discret

Pour les *représentations en temps discret*, la modélisation des évènements selon la formulation *Global time intervals* est l'unique possibilité pour les procédés séquentiels ou en réseau. Dans ce cas, une échelle de temps fixe, partagée entre toutes les ressources, est prédéfinie et on force la synchronisation des débuts et fins de tâche sur les points de cette échelle. Par conséquent, le problème initial est réduit à un problème d'allocation dans lequel les décisions consistent à attribuer à un intervalle de temps, les tâches qui doivent être lancées. Cette allocation est modélisée au travers de la variable discrète W_{ijt} telle qu'elle est définie dans le tableau. Il y a un avantage certain à utiliser une échelle de temps fixe, car certaines contraintes du problème qui sont dépendantes du temps peuvent être modélisées de manière simple sans compromettre la linéarité du modèle. Certains de ces aspects comprennent des contraintes strictes de temps, de coûts d'utilités dépendants du temps [Hait, 2007], de coûts de stockage, de demandes de produits échelonnées sur l'horizon de temps, etc.

4.2.4.2. Modèles en temps continu pour les topologies de procédés en réseau

A l'inverse, les formulations en temps continu autorisent un grand nombre de possibilités pour la gestion des évènements. A chaque mode de gestion des évènements correspondent des types de topologie d'ateliers bien précis. En effet, pour les topologies en réseau, les modèles basés sur les *Global time points* et *Unit-specific time event* peuvent être utilisés ; tandis que pour les procédés séquentiels, les différentes combinaisons possibles imposent l'utilisation des *time slots* ou de différentes approches basées sur des *contraintes de précédences*. La modélisation sous la forme de *Global time points* correspond à la généralisation des *Global time intervals* où la durée des intervalles de temps est traitée comme une nouvelle variable du modèle. Dans ce cas, une échelle de temps variable est partagée par l'ensemble des ressources partagées.

Les modèles continus basés sur les représentations *STN* ou *RTN* sont bien adaptés à la représentation de topologies d'ateliers en réseau, cependant il est nécessaire de surveiller les limites de capacités des ressources pour un petit nombre de dates afin de garantir la faisabilité de la solution. Les modèles de ce type sont relativement simples à mettre en œuvre, même pour des problèmes d'ordonnancement complexes. A l'inverse des *Global time points*, le principe des modélisations *Unit-specific time event* repose sur la définition d'une échelle de temps propre à chaque ressource partagée, ce qui permet à différentes tâches de démarrer à des dates différentes pour un même point d'évènement. Ces modélisations utilisent la représentation au format *STN*, grâce à la localisation flexible des évènements, le nombre d'évènements requis est généralement plus faible que dans le cas de *Global time points*. Cependant, le manque de points de référence pour le contrôle des limites de capacité rend la formulation plus compliquée. Des contraintes spécifiques et des variables additionnelles doivent être définies pour prendre en compte les contraintes de capacité.

L'adaptabilité des modèles et l'efficacité des méthodes de calcul associées aux modélisations du type *Global time points* et *Unit-specific time event* dépend fortement du nombre d'évènements prédéfinis. En effet, si la résolution globale du modèle nécessite la définition d'au moins n évènements, un nombre inférieur d'évènements conduira à une solution sous-optimale ou à une résolution impossible. Tandis qu'un nombre trop élevé d'évènements ne sera pas significatif non plus et entraînera des temps de résolution élevés. Ce nombre n'étant pas connu a priori, une méthode consiste à le déterminer de manière itérative. Le nombre d'évènements est incrémenté de un jusqu'à ce qu'il n'y ait plus d'amélioration de la fonction objectif. Cela signifie que pour chaque problème, un nombre significatif d'itérations doit être réalisé, ce qui peut mener à une augmentation du temps CPU. Il est important de préciser que l'arrêt d'une résolution ne permet pas de garantir que la solution globale du problème a été trouvée mais que l'optimum correspond au nombre d'évènements définis.

4.2.4.3. Modèles en temps continu pour les procédés avec des structures séquentielles

Il existe différentes formulations de modèles en temps continu adaptées spécialement aux procédés séquentiels, bien que certains aient été étendus de manière récente aux topologies en réseau.

L'un des premiers modèles développés pour résoudre ces types de problèmes était basé sur le concept de *time-slots* qui définit un certain nombre d'intervalles de temps dont la durée est inconnue. Le concept de base de ce modèle repose sur la définition d'un nombre de *time- slots* pour chaque opération unitaire et l'allocation de ces derniers aux lots qui doivent être réalisés. Le choix du nombre de *time-slots* est non trivial et correspond à un compromis entre optimalité et temps de résolution.

Les représentations basées sur les *time-slots* peuvent être divisées en deux grandes catégories : synchrones et asynchrones.

- La *représentation synchrone*, basée sur des concepts similaires aux *global time points*, définit des *time-slots* communs pour toutes les unités, ce qui simplifie la manipulation des ressources partagées notamment pour des topologies de procédés en réseau.
- La représentation asynchrone autorise la définition de *time-slots* différents d'une unité à l'autre, ce qui donne plus de flexibilité à l'ordonnancement dans le choix des dates. Cette représentation est plus proche de la modélisation *unit-specific time event* et est mieux adaptée aux procédés ayant une structure séquentielle.

D'autres approches existent pour la résolution des procédés avec une structure séquentielle, notamment celles basées sur les contraintes de précédence entre les lots : *batch precedence*. Dans ces formulations, les variables et les contraintes forçant l'utilisation séquentielle des ressources sont explicitement définies. Par conséquent, les problèmes basés sur des temps de préparation dépendants de la séquence peuvent être résolus. Afin de déterminer la séquence optimale des tâches dans chaque unité, le concept de *bacth precedence* peut être appliqué soit au *lot « immédiatement précédent »* soit à *l'ensemble des lots précédents*.

4.3. FORMULATION RETENUES DANS NOS TRAVAUX

La section précédente a montré que diverses formulations *MILP* pour le problème d'ordonnancement des procédés batch peuvent être trouvées dans la littérature [Kondili *et al.,* 1993], [Kallrath, 2002], [Maravelias et Grossmann, 2003] et plusieurs reviews récentes sur le sujet ont été proposées [Floudas et Lin, 2004], [Burkard et Hatzl, 2005], [Shaik *et al*, 2005], [Mendez et al, 2006].

Par ailleurs, une classification des modèles applicables à chaque catégorie de problèmes a ensuite été détaillée.

Dans le cadre de ces travaux, le besoin de prendre en compte des **topologies de procédé en réseau**, élimine les modèles basés sur des contraintes de précédence. De même, la nécessité de prendre en compte des **durées de tâches dépendantes de la taille des lots** impose une modélisation en temps continu. Il reste donc trois grands types de formulations possibles, à savoir : *Global time points*, *Unit-specific time event, Time slots*.

Une comparaison détaillée de ces types de modèles peut être consultée dans [Shaik *et al.,* 2005]. A titre d'illustration, la résolution de l'exemple relativement complexe de [Sundaramoorthy and Karimi, 2005] est présentée ci-dessous. La Figure 4.24 présente le graphe de flux de ce problème.

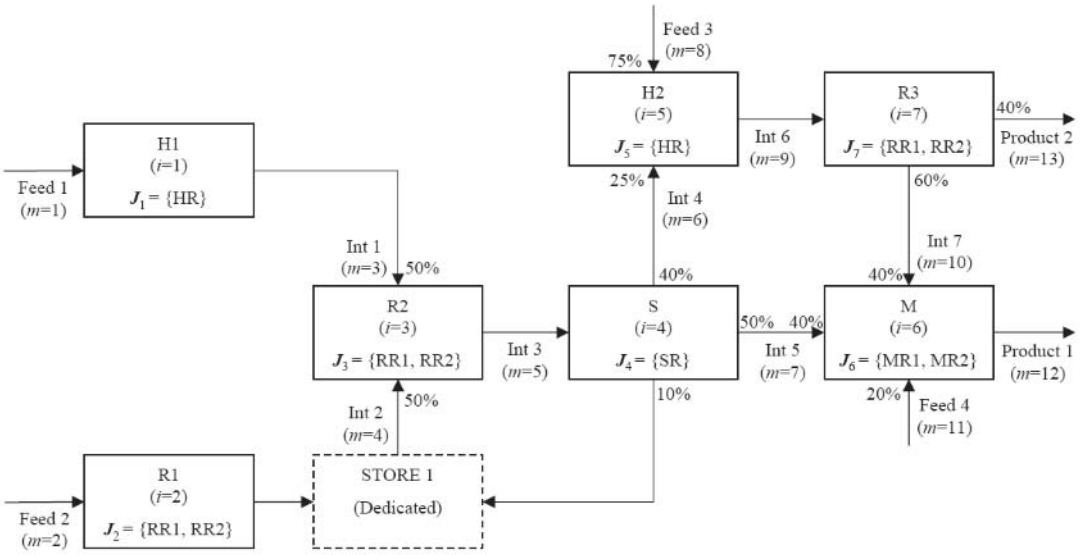


Figure 4.24 Graphe de flux de l'exemple 3 de [Sundaramoorthy and Karimi, 2005]

Les données de l'exemple sont les suivantes :

Recipe Task	Label i	Unit	Label j	τ_{ij}	α_{ij}	β_{ij}	B_{ij}^{L} (mu)	B_{ij}^{U} (mu)
Example 3								
Heating-1	H1	Heater	HR	1	0.667	0.00667	—	100
Heating-2	H2	Heater	HR	1.5	1.000	0.01000	—	100
Reaction-1	R1	Reactor 1	RR1	2	1.333	0.01333	—	100
		Reactor 2	RR2	2	1.333	0.00889	—	150
Reaction-2	R2	Reactor 1	RR1	1	0.667	0.00667	—	100
		Reactor 2	RR2	1	0.667	0.00445	—	150
Reaction-3	R3	Reactor 1	RR1	2	1.333	0.01330	—	100
		Reactor 2	RR2	2	1.333	0.00889	—	150
Separation	S	Separator	SR	3	2.000	0.00667	—	300
Mixing	M	Mixer 1	MR1	2	1.333	0.00667	20	200
		Mixer 2	MR2	2	1.333	0.00667	20	200

Tableau 4.2 Durées opératoires des tâches et bornes sur la taille des lots sur chaque unité

Material m	Example 3		
	Storage capacity (mu)	Initial inventory (mu)	Revenue ($/mu)
1	UL	AA	0
2	UL	AA	0
3	100	0	0
4	100	0	0
5	300	0	0
6	150	50	0
7	150	50	0
8	UL	AA	0
9	150	0	0
10	150	0	0
11	UL	AA	0
12	UL	0	5
13	UL	0	5

UL = Unlimited; AA = Available as and when required.

Tableau 4.3 Capacités de stockage, niveaux de stocks initiaux et bénéfices par unité de masse

model	events	H	CPU time (s)	nodes	RMILP (h)	MILP (h)	binary variables	continuous variables	costraints	nonzeros
				Example 3a (D_{12} = 100 mu, D_{13} = 200 mu)						
S&K	8	--	0.28	36	12.317	14.366	119	690	689	2425
	9	--	13.32	5156	11.621	13.589	136	783	793	2789
	10	--	226.83	53789	11.417	13.532	153	876	897	3153
	11	--	4340.65	821194	11.335	13.367	170	969	1001	3517
	12	--	>80000[1]	11858901	11.295	13.367	187	1062	1105	3881
M&G	8	50	1.15	316	12.317	14.366	154	831	1937	6905
	9		126.77	21366	11.621	13.589	176	944	2201	8208
	10		3949.36	605450	11.417	13.532	198	1057	2465	9592
	11		>80000[2]	7481387	11.335	13.367	220	1170	2729	11057
CBM	8	--	0.62	68	10.941	14.366	385	541	1064	4044
	9	--	5.89	2762	8.941	13.589	484	659	1309	5090
	10	--	53.42	22452	6.941	13.532	594	788	1578	6254
	11	--	2514.97	673460	4.941	13.367	715	928	1871	7536
	12	--	>80000[3]	20380858	3.825	13.367	847	1079	2188	8936
CBMN(Δt=2)	8	--	0.23	67	12.192	14.366	143	307	402	1776
(Δt=2)	9	--	2.23	2566	10.192	13.589	165	349	459	2044
(Δt=2)	10	--	14.73	17426	8.192	13.532	187	391	516	2312
(Δt=2)	11	--	312.07	326752	6.192	13.532	209	433	573	2580
(Δt=3)	11	--	20230.89	16842943	6.192	13.367	297	521	725	3494
(Δt=3)	12	--	11547.29	5054232	4.635	13.367	330	574	801	3877
G&G	7	--	0.25	338	11.066	14.616	77	336	558	1902
	9	--	3.36	3960	10.167	14.616[a]	99	428	712	2448
I&F	**7**	**50**	**0.38**	**458**	**11.066**	**13.367**	**52**	**225**	**452**	**1413**
	8		2.89	3506	10.000	13.367	63	260	526	1677
				Example 3b (D_{12} = D_{13} = 250 mu)						
S&K	11	--	981.01	226238	14.535	17.357[a]	170	969	1001	3517
M&G	11	100	62724.36	5802875	14.535	17.357[a]	220	1170	2729	11057
CBM	11	--	38.14	9627	10.722	17.357[a]	715	928	1871	7536
CBMN(Δt=2)	11	--	31.57	28079	12.494	17.357[a]	209	433	573	2580
G&G	10	--	59.26	84970	12.763	18.978[a]	110	474	789	2721
I&F	**10**	**100**	**2.50**	**2668**	**12.500**	**17.025**	**85**	**330**	**674**	**2205**
	11		396.58	424617	12.500	17.025	96	365	748	2469

[a] Suboptimal solution; Relative Gap: 1.01 %[1], 3.055 %[2], 3.29 %[3]

Tableau 4.4 Tableau comparatif des résultats de l'exemple pour la minimisation de la Durée du Plan (d'après [Shaik et al.*, 2005])*

En conclusion ces résultats montrent sur plusieurs exemples de benchmarks le bon compromis entre **temps de résolution** et **robustesse** (voir Tableau 4.4 et de manière plus complète [Shaik *et al.,* 2005]) de la formulation *Unit-Specific Event* proposée dans [Ierapetritou et Floudas, 1998]. C'est pourquoi nous avons retenu cette formulation dans nos travaux. En effet, outre ces qualités, elle permet la résolution des problèmes à structure en réseau avec des durées de tâches dépendantes des tailles de lots. De plus, basé sur une approche monolithique (cf. section 4.2.2), ce modèle d'optimisation permet de calculer simultanément les dates de début (ainsi que leur ordre de passage sur les équipements) et la taille de lots de chaque tâche.

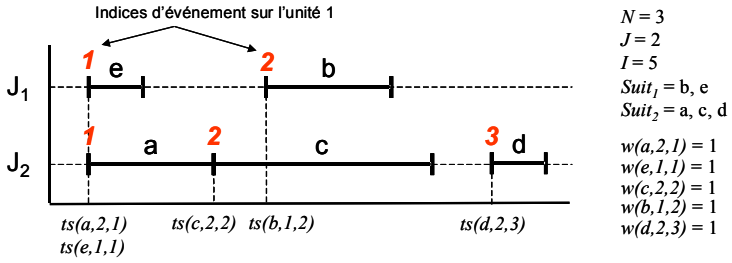


Figure 4.25 Formulation Unit-specific time event

Précisons enfin, que la version du modèle effectivement implémentée correspond à la plus récente qui limite l'emploi des contraintes de type « Big-M » et favorise l'agrégation des contraintes de séquences. La Figure 4.25 montre un schéma de principe d'un ordonnancement selon cette modélisation.

4.4. ANALYSE DU MODÈLE D'ORDONNANCEMENT DE BASE

Le modèle d'ordonnancement à établir doit déterminer :

- la date de lancement au plus tard *Ts* de chaque lot (et implicitement le nombre de ces lots),
- la taille du batch *B* associé à chaque lot

de manière à réaliser chaque *OF* défini sur la campagne de production considérée, en :

- respectant la gamme opératoire de chaque produit ainsi que les bilans matière,
- respectant les contraintes de capacité des appareils,
- respectant les contraintes de capacité des cuves de stockage,

avec pour objectif :

- de minimiser le *makespan* (ou les retards de livraison) et les en-cours.

4.4.1. Formulation du modèle d'ordonnancement de base et formalisation *RTN*

Comme indiqué dans la section précédente, le problème est modélisé sous forme d'un programme linéaire en variable mixte (*MILP*) s'appuyant sur une formulation en temps continu nommée *Unit Specific Event* (basée sur des points d'évènement).

Le modèle d'ordonnancement considéré s'appuie sur une *représentation graphique RTN*, pour laquelle il est possible de faire correspondre un jeu de contraintes à chaque entité du formalisme.

Ce modèle nécessite les indices, ensembles, paramètres et variables indiqués dans la nomenclature. Sur la base de cette notation, le modèle mathématique de base pour l'ordonnancement court-terme de procédés batch est constitué des contraintes suivantes :

- **Contraintes d'allocation :**

$$\sum_{i \in I_j} W_{i,n} \leq 1 \qquad \forall j \in J,\ \forall n \in N \tag{3.1}$$

Ces contraintes expriment le fait que pour chaque appareil *j* et à chaque point *n*, seulement une des tâches *i* qui peut être réalisée dans cet appareil peut y prendre place.

Formalisation graphique 1 : Contraintes de ressources disjonctives (3.1)

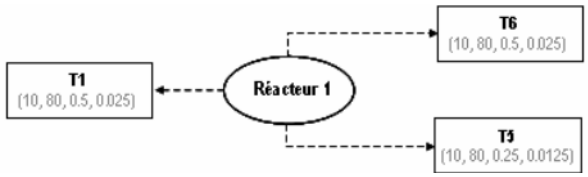


Figure 4.26 Ressource disjonctive et contraintes d'allocations

- **Contraintes de capacité des appareils de production**

$$V_i^{\min} W_{i,n} \leq B_{i,n} \leq V_i^{\max} W_{i,n} \qquad \forall i \in I,\ \forall n \in N \tag{3.2}$$

Si $W_{i,n} = 1$, alors les contraintes (3.2) définissent les bornes minimales et maximales pour la taille des batch $B_{i,n}$, sinon elles forcent la variable $B_{i,n}$ à zéro.

Formalisation graphique 2 : Contraintes de taille de lots (3.2) (seuil de lancement V_i^{min}, capacité des appareils V_i^{max})

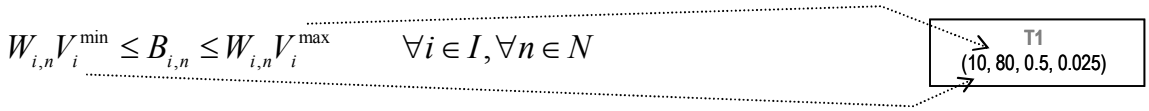


Figure 4.27 Contraintes de capacité

- **Bilan matière sur les états :**

$$S_{s,n} = S_{s,n-1} + \sum_{i \in I_s^p} \rho_{i,s}^p B_{i,n-1} - \sum_{i \in I_s^c} \rho_{i,s}^c B_{i,n} \qquad \forall s \in S,\ \forall n \in N \mid n>1 \qquad (3.3)$$

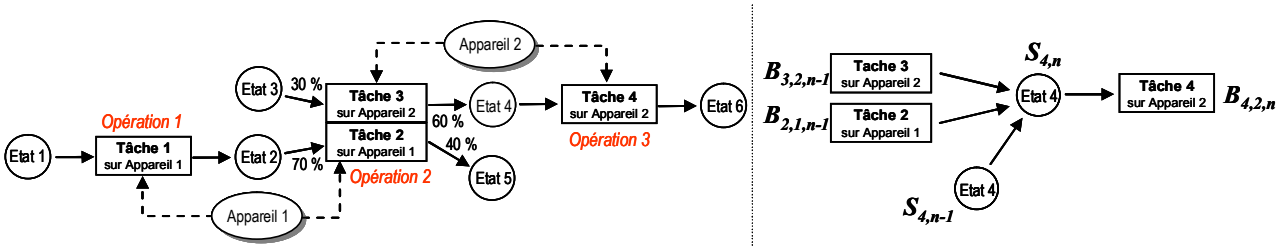


Figure 4.28 Bilans matière

Selon ces contraintes, la quantité de matière dans l'état *s* au point *n* est égale à celle existant au point *n-1* augmentée des quantités produites au point *n-1* et réduite des quantités consommées au point *n*.

Etat initial :

$$S_{s,1} = S0_s + \sum_{i \in I_s^c} \rho_{i,s}^c B_{i,1} \qquad \forall s \in S \qquad (3.4)$$

La quantité de matière dans l'état *s* au premier point est égale au stock initial $S0_s$, réduit des quantités consommées par les batch lancés au point 1.

Contraintes de capacité des cuves de stockage :

$$S_{s,n} \leq C_s \qquad \forall s \in S,\ \forall n \in N \qquad (3.5)$$

Ces contraintes fixent une borne supérieure sur les niveaux de stock de chaque état *s*.

Formalisation graphique 3 : Contraintes liées au Bilan matière sur les états (3.3), (3.4) et (3.5)

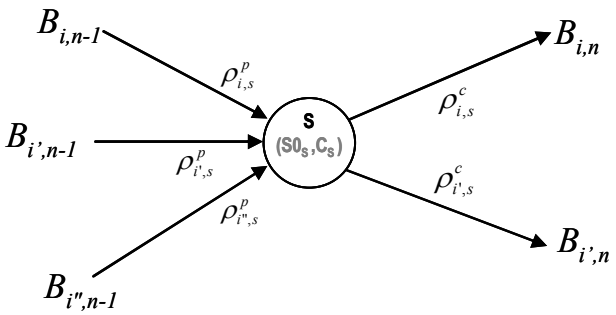


Figure 4.29 Etat initial et capacités de stockage des états en RTN

- **Satisfaction des OF :**

$$SF_s = S_{s,N} + \sum_{i \in I_s^p} \rho_{i,s}^p B_{i,N} \qquad \forall s \in S \tag{3.6}$$

La quantité totale SF_s de matière dans l'état s à la fin du dernier point est égale à la quantité en stock au début du dernier point $S_{s,n}$ augmentée des quantités produites par les batch lancés au dernier point N.

$$SF_s \geq D_s \qquad \forall s \in S \tag{3.7}$$

La quantité totale SF_s de matière dans l'état s en stock à la fin du dernier point doit être supérieure à la quantité définie par l'OF en matière dans l'état s.

Cette contrainte est écrite sous forme d'une inégalité (et non d'une égalité) à cause du couplage pouvant exister entre deux procédures. Par exemple, dans l'unité considérée Figure 4.28, la fabrication des produits Etat 5 et Etat 6 est couplée via l'intermédiaire de l'Etat 4. En effet, selon les volumes respectifs de chaque OF à réaliser, on peut être amené à produire plus de produit Etat 5 que demandé, simplement parce que la demande en Etat 6 impose la production de l'Etat 4 et par voie de conséquence, induit la coproduction du produit Etat 5.

- **Durée opératoire des tâches :**

$$pt_{i,n} = ft_i W_{i,n} + vt_i B_{i,n} \qquad \forall i \in I,\ \forall n \in N \tag{3.8}$$

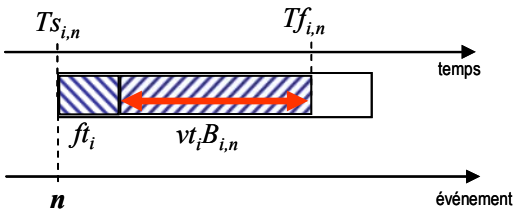


Figure 4.30 Durée opératoire d'une tâche

La contrainte (3.8) définit la durée d'une tâche comme étant la somme d'une durée fixe (temps de préparation, temps opératoire indépendant de la taille du lot) et d'une durée variable (temps opératoire dépendant de la taille du lot, temps de transfert, etc.). La détermination des valeurs ft et vt est décrite dans la suite.

$$Tf_{i,n} = Ts_{i,n} + pt_{i,n} \qquad \forall i \in I,\ \forall n \in N \tag{3.9}$$

Les contraintes (3.9) représentent la relation entre la date de début et la date de fin de la tâche i lancée au point n.

Formalisation graphique 4 : Contraintes sur la durée d'une tâche (3.9)

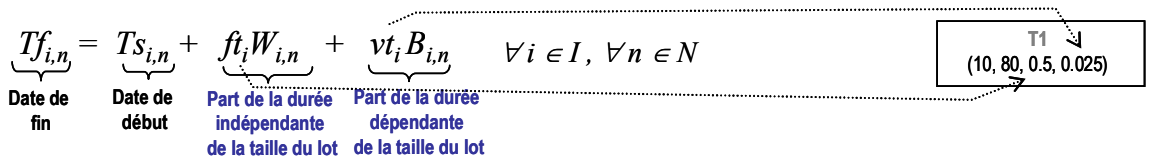


Figure 4.31 Contraintes de durée opératoire

- **Contraintes de séquence des tâches :**

Enchaînement d'une même tâche sur le même appareil

$$Ts_{i,n} \geq Tf_{i,n-1} \qquad \forall i \in I,\ \forall n \in N \mid n>1 \qquad (3.10)$$

Ces contraintes de séquence établissent qu'une nouvelle tâche *i* commençant au point *n* ne peut démarrer qu'après la fin d'une éventuelle autre tâche *i* réalisée sur le même appareil *j*.

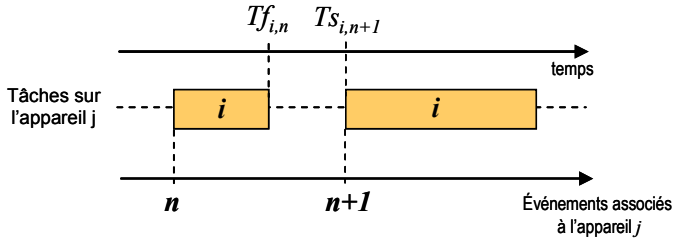


Figure 4.32 Satisfaction des OF

Enchaînement de tâches différentes sur le même appareil

$$Ts_{i,n} \geq Tf_{i',n-1} + \tau_{i,i'} W_{i',n} - M(1 - W_{i',n-1})$$

$$\forall j \in J, \forall i \in Ij, \forall i' \in Ij \mid i \neq i', \forall n \in N \mid n>1 \qquad (3.11)$$

Les contraintes (3.11) sont écrites pour des tâches *i* et *i'* exécutées consécutivement dans le même appareil *j* respectivement au point *n-1* et *n*. Ainsi, si une tâche *i'* a été lancée au point *n-1*, alors la date de début d'une tâche *i* suivante doit être supérieure à la date de fin de cette tâche *i'* (c.a.d, $Ts_{i,n} \geq Tf_{i',n-1}$). Par contre, s'il n'y a pas de tâche *i'* lancée au point *n-1*, alors la contrainte est relaxée. Un temps de nettoyage $\tau_{i',i}$ dépendant de la séquence peut aussi être pris en compte par cette contrainte.

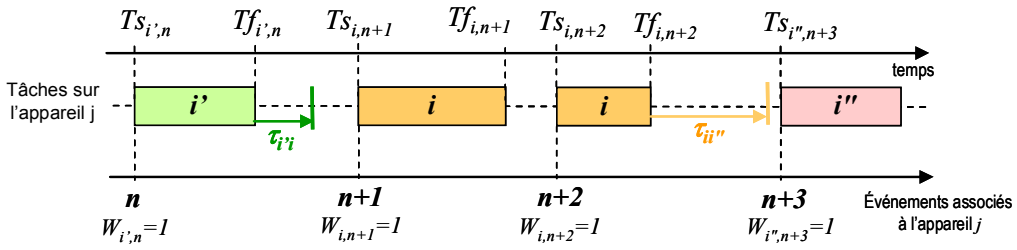


Figure 4.33 Enchaînement de tâches différentes sur le même appareil

Enchaînement de tâches différentes sur des appareils différents

$$Ts_{i,n} \geq Tf_{i',n-1} - M(1 - W_{i',n-1})$$

$$\forall s \in S, \forall i \in I_s^c, \forall i' \in I_s^p, \forall j \in Ji, \forall j' \in Ji' | j \neq j', \forall n \in N | n > 1 \qquad (3.12)$$

En relation avec les bilans matière (cf. contraintes (3.3)), ces contraintes relient des tâches *i* et *i'* exécutées dans des appareils distincts *j* et *j'*, mais situées de manière consécutive dans le temps à cause de la recette de production. Notamment, elles relient des tâches productrices et consommatrices d'un même état de la matière *s*. Une tâche *i* située dans l'appareil *j* et consommant l'état *s* doit commencer au point *n* après la fin de la tâche *i'* située sur l'appareil *j'* et produisant l'état *s* lancée au point *n-1*. Dans ce cas, la contrainte devient $Ts_{i,n} \geq Tf_{i',n-1}$. Dans le cas contraire, la contrainte est relaxée.

Notons ici que ces contraintes induisent une synchronisation des numéros de points d'événement des 2 appareils *j* et *j'* concernés.

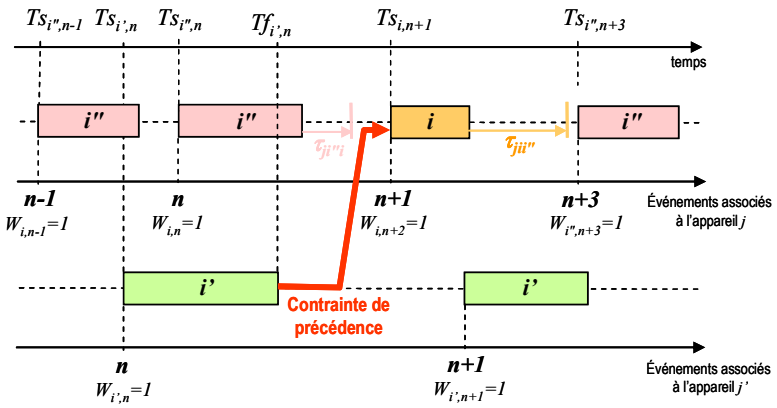


Figure 4.34 Enchaînement de tâches différentes sur des appareils différents

- **Contraintes de durée du plan :**

$$Tf_{i,n} \leq DureePlan \qquad \forall i \in I, \ \forall n \in N \qquad (3.13)$$

Cette contrainte fixe simplement une borne supérieure aux dates de fin des tâches, la variable *DureePlan* étant soit une variable à minimiser, soit une donnée correspondant au jalon fixé par le niveau *Planification*.

- **Critère à optimiser :**

$$\min\left(a.DureePlan + \sum_{s \in S} h_s SF_s + \sum_{s \in S} \sum_{n \in N} h_s S_{s,n} \right) \qquad (3.14)$$

Cette fonction *objectif* minimise la durée du plan, ainsi que les niveaux de stock. Cela s'avère nécessaire afin de lancer des batch de volume « juste nécessaire ». En effet, pour les tâches dont la durée est indépendante de la taille du lot, un volume supérieur pourrait être produit sans pour autant allonger la durée du plan.

4.4.2. Nomenclature

Indices :

i : indice de tâche dans la représentation *R.T.N,*
j : indice d'appareil,
n : indice de point d'événement,
s : indice d'état de la matière,

Ensembles :

I : ensemble des tâches de la recette,
J : ensemble des appareils du procédé,
N : ensemble des points d'événement,
S : ensemble des états de la matière,
I_j : ensemble des tâches pouvant être réalisées sur l'appareil *j,*
I_s^c : ensemble des tâches consommant l'état de la matière *s,*
I_s^p : ensemble des tâches produisant l'état de la matière *s,*
J_i : ensemble des appareils pouvant réaliser la tâche *i*,

Variables binaires :

$W_{i,n}$: vaut 1 si une tâche *i* est lancée au point d'événement *n*,

Variables continues :

$B_{i,n}$: quantité de matière traitée par la tâche *i* au point d'événement *n*,
$S_{s,n}$: quantité de matière dans l'état *s* au point d'événement *n*,
SF_s : quantité finale de matière dans l'état *s* en fin d'horizon,
$Ts_{i,n}$: date à laquelle démarre la tâche *i* au point d'événement *n*,
$Tf_{i,n}$: date à laquelle se finit la tâche *i* au point d'événement *n*,
$pt_{i,n}$: durée opératoire de la tâche *i* au point d'événement *n*,
DureePlan : définit la durée du plan

Paramètres :

D_s : Ordre de Fabrication de matière dans l'état *s* à satisfaire sur la campagne de production
$V_i^{\min}, V_i^{\max}$: volume minimum et maximum de matière pour la tâche *i*,
C_s : capacité de la cuve de stockage de l'état *s*,
$\rho_{i,s}^c$: proportion de matière dans l'état *s* consommée par la tâche *i*,
$\rho_{i,s}^p$: proportion de matière dans l'état *s* produite par la tâche *i*,
$S0_s$: stock initial de matière dans l'état *s*,
h_s : coût de stockage de l'état *s*,
a : poids associé à la durée du plan dans le critère,
$\tau_{i',i}$: délai de nettoyage sur l'appareil *j* si on enchaîne une tâche *i* après une tache *i'*,
ft_i : partie fixe de la durée opératoire de la tâche *i*,
vt_i : partie variable de la durée opératoire de la tâche *i*,

4.4.3. Exemples de scénarios de production

4.4.3.1. Description du procédé considéré

L'exemple traité est l'exemple inspiré de [Joglekar *et al.,* 1985], dans un premier temps, nous considérons la phase de séparation comme une opération discontinue. Cette hypothèse nous conduit à la représentation *STN* de la Figure 4.35.

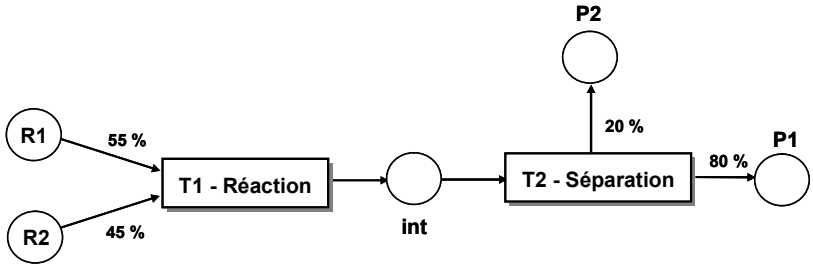


Figure 4.35 STN *de l'exemple de procédé*

La topologie du procédé détaillée sur la Figure 4.36 nous permet de déduire la représentation *RTN* de la Figure 4.37.

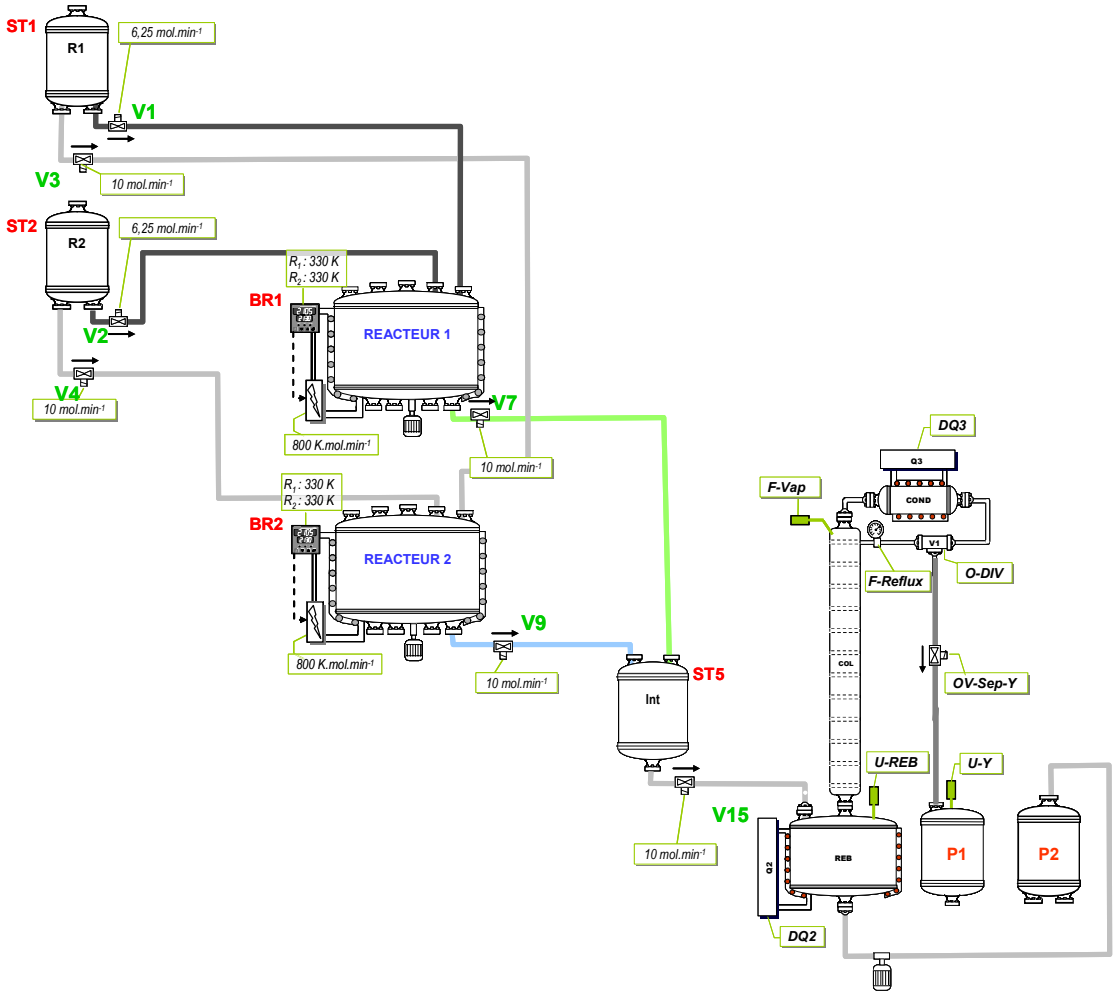


Figure 4.36 Topologie de l'exemple de procédé

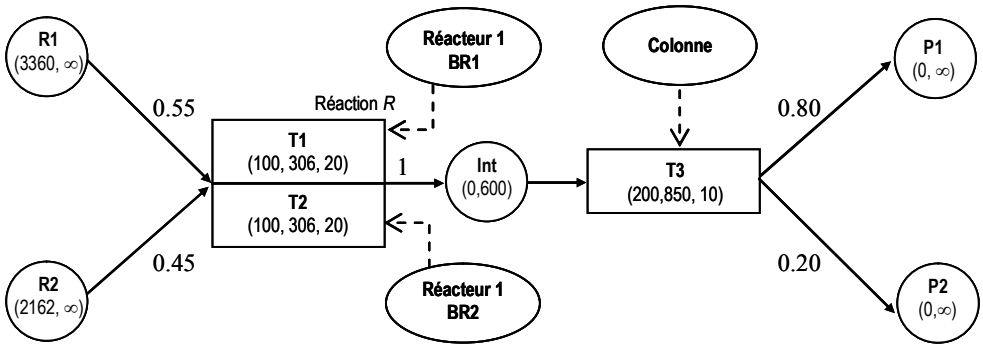


Figure 4.37 RTN *obtenu par instanciation du* STN *avec l'unité de la figure*

Dans la suite, différentes instances sont proposées correspondant à un scénario particulier de production réalisé sur le procédé décrit précédemment.

Les valeurs des paramètres $V_i^{\min}, V_i^{\max}$, C_s, $\rho_{i,s}^c$ et $\rho_{i,s}^p$ sont celles indiquées sur le *RTN* correspondant. Par ailleurs, le poids associé à la durée du plan vaut *a=10* et les coûts de stockage h_s sont indiqués ci-dessous :

n° Etat	1	2	3	4	5
Etat matière	R1	R2	Int	P2	P1
h	0	0	100	30	40

Tableau 4.5 Coûts de stockage

Les valeurs de stocks initiaux sont les suivantes :

n° Etat	1	2	3	4	5
Etat matière	R1	R2	Int	P2	P1
SO (kg)	3360	2162	0	0	0

Tableau 4.6 Stocks initiaux pour l'Instance 1

Toutes les instances sont résolues avec le logiciel *XPRESS-MP (version 1.6.3),* sur un processeur *INTEL Pentium Centrino 1,76 GHz (1 Go RAM).*

4.4.3.2. Instance 1 : Durées opératoires fixes

Paramètres :

- Nombre de points d'évènement : *N=6* points.
- Ordres de fabrication à réaliser : *600 kg* de *P2*.
- Pas de délais de nettoyage des appareils.
- **Durées opératoires fixes** et indépendantes de la taille du lot et de l'appareil utilisé :

Les durées ont tout d'abord été estimées par la simulation d'un batch de taille moyenne, sans réaliser une analyse fine, puis une marge a été ajoutée pour la prise en compte de l'ensemble des tailles de lots admissibles.

n° Tâche	1	2	3
Ressource	BR1	BR2	Colonne
df (h)	20	20	10

Tableau 4.7 Durées opératoires fixes

Solution : solution optimale obtenue en 2,9 secondes

Tâche	t	Durée	Machine	Produit	Batch
T1	0.00	20.00	1	1	306.00
T1	0.00	20.00	2	1	244.00
T1	20.00	20.00	2	1	200.00
T2	20.00	10.00	3	2	550.00
T2	40.00	10.00	3	2	200.00

Tableau 4.8 Tableau de résultats de l'ordonnancement de l'instance 1

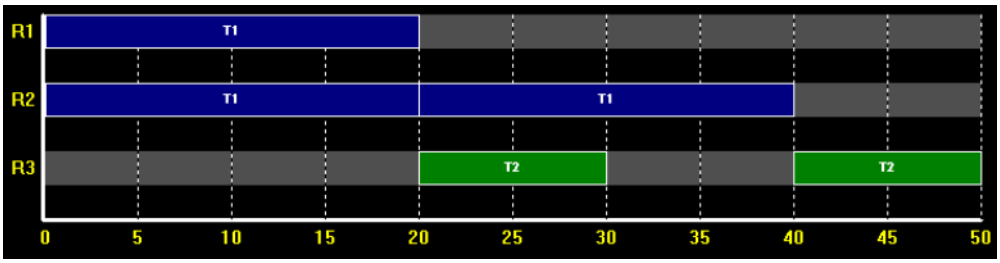


Figure 4.38 Diagramme de Gantt de l'instance 1 (durées en heures)

Les tâches affichées par le diagramme de Gantt correspondent au tâches identifiées dans le *STN* (cf. Figure 4.35).

Légende pour la lecture du Gantt :

- les appareils : *R1* : Réacteur *BR1*, *R2*: Réacteur *BR2*, *R3*: Séparateur (*Colonne*)
- les tâches : Dans la notation *T(n)*, *n* correspond au n° d'opération. Par conséquent, *T(1)* sur la ressource *R2* correspond à l'opération 1 (réaction) réalisée sur le *réacteur BR2*, soit la tâche notée T1 sur la représentation *STN* de la Figure 4.35 (cette tâche correspond à la tâche nommée *T2* sur le *RTN* de la Figure 4.36).

4.4.3.3. Instance 2 : Durées opératoires variables

Paramètres :

- Nombre de points d'évènement : *N=5* points.
- Ordres de fabrication à réaliser : *600 kg* de *P2*.
- Pas de délais de nettoyage des appareils.
- **Durées opératoires variables** et dépendantes de la taille du lot et de l'appareil utilisé ne comprenant qu'une partie variable :

n° Tâche	1	2	3
Ressource	BR1	BR2	Colonne
dv (h/kg)	0,0535	0,0535	0,006

Tableau 4.9 Durées opératoires variables

Solution : solution optimale obtenue en 4,1 secondes

Tâche	t	Durée	Machine	Produit	Batch
T1	0.00	14.71	1	1	275.00
T1	14.71	5.35	1	1	100.00
T1	0.00	14.71	2	1	275.00
T1	14.71	5.35	2	1	100.00
T2	14.71	3.30	3	2	550.00
T2	20.06	1.20	3	2	200.00

Tableau 4.10 Tableau de résultats de l'ordonnancement de l'instance 2

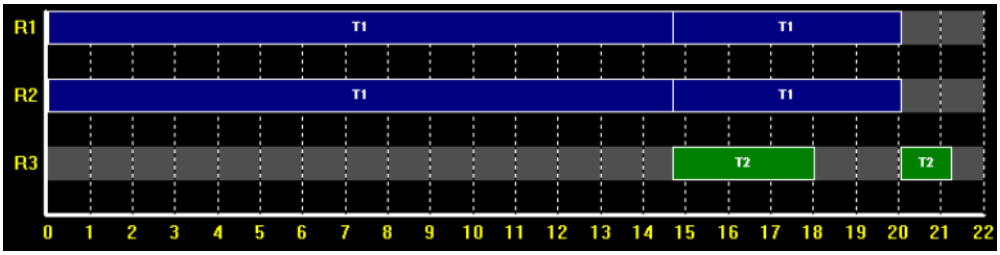


Figure 4.39 Diagramme de Gantt de l'instance 2 (durées en heures)

4.4.3.4. Instance 3 : Durées opératoires comprenant une partie fixe et une partie variable

Paramètres :

- Nombre de points d'évènement : *N=7* points.
- Ordres de fabrication à réaliser : *600 kg* de *P2*.
- Pas de délais de nettoyage des appareils.
- **Durées opératoires variables** et dépendantes de la taille du lot et de l'appareil utilisé comprenant une partie fixe et une partie variable :
-

n° Tâche	1	2	3
Ressource	BR1	BR2	Colonne
df (h)	7,9	7,9	9.56
dv (h/kg)	0,0535	0,0535	0,006

Tableau 4.11 Durées opératoires

Ces durées opératoires ont été évaluées par simulation, à l'aide du simulateur *PrODHyS* (cf. chapitre 2).

Ces durées opératoires peuvent être divisées en trois parties :

- les durées opératoires fixes : temps de réaction lié à la température contrôlée
- les temps de chauffe proportionnels aux tailles de lots
- les remplissages et vidanges par gravité (fonctions non linéaires des tailles de lots)

En première approximation, nous avons identifié une partie fixe et une partie variable pour l'opération de réaction, en traçant la durée opératoire (en heures) en fonction de la taille des lots (en kg) sur la Figure 4.40, ce qui nous a permis de remplir le Tableau 4.11. En effet, les tailles de lots étant fortement contraintes (entre 100 et 300 kg) les aspects non linéaires peuvent être négligés.

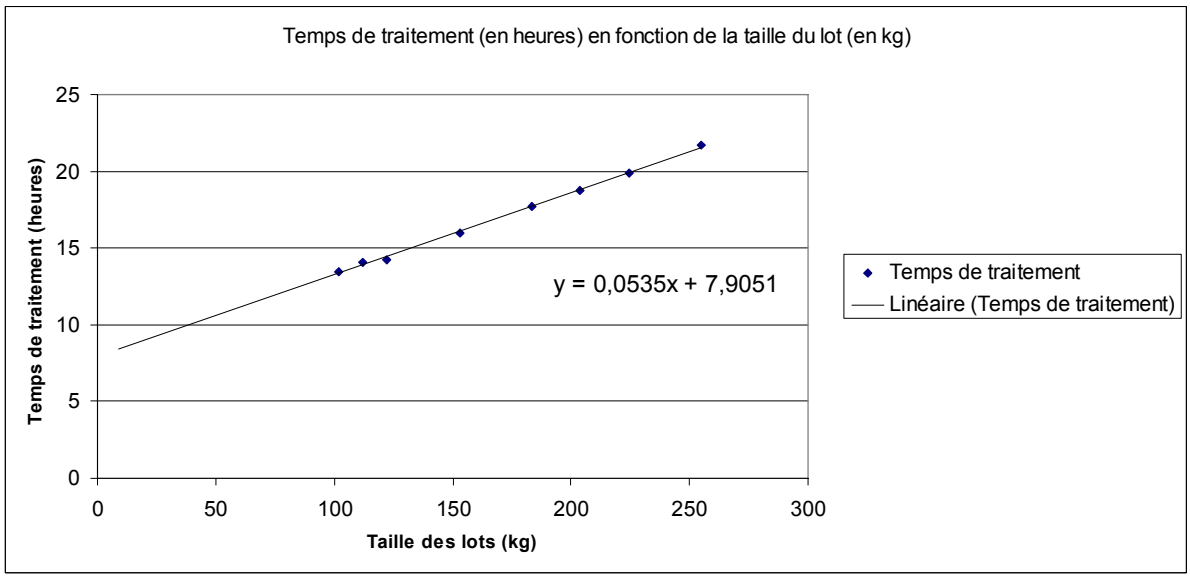


Figure 4.40 Durées de traitement de l'opération de réaction dans BR1 et BR2 (température de réaction 365 K)

Solution : solution optimale obtenue en 4,1 secondes

Tâche	t	Durée	Machine	Produit	Batch
T1	0.00	22.61	1	1	275.00
T1	22.61	13.25	1	1	100.00
T1	0.00	22.61	2	1	275.00
T1	22.61	13.25	2	1	100.00
T2	22.61	12.86	3	2	550.00
T2	35.86	10.76	3	2	200.00

Tableau 4.12 Tableau de résultats de l'ordonnancement de l'instance 3

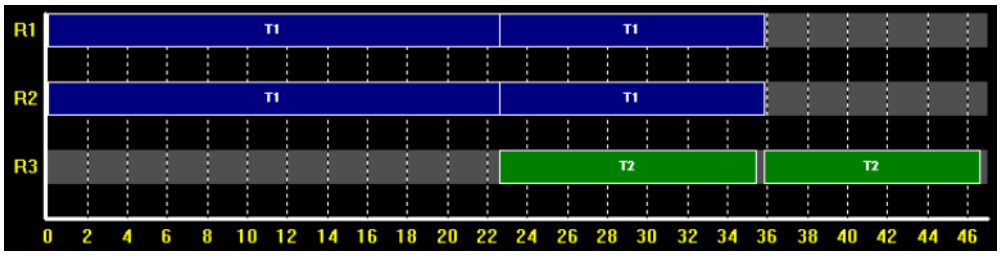


Figure 4.41 Diagramme de Gantt de l'instance 3 (durées en heures)

4.5. EXTENSION DU MODÈLE À LA PRISE EN COMPTE DE PARAMÈTRES NON LINÉAIRES

Pour illustrer ce type de contraintes, le cas de vidange par gravité est considéré. Pour cela, de nouvelles variables doivent être intégrées.

4.5.1. Modélisation de paramètres non linéaires

La durée fixe de réaction est estimée à : 5h

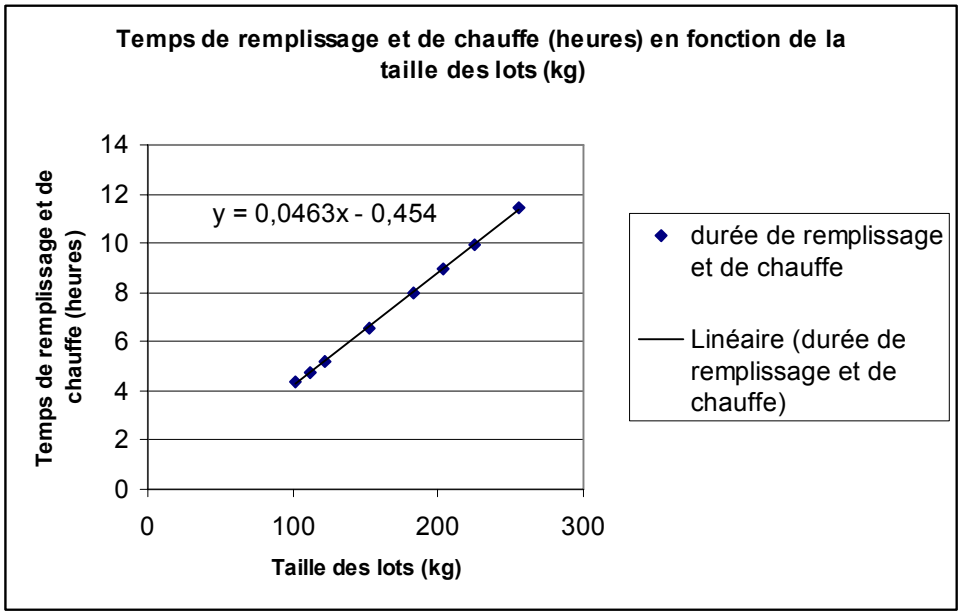


Figure 4.42 Estimation des variables dépendantes des tailles de lots (remplissage et chauffe)

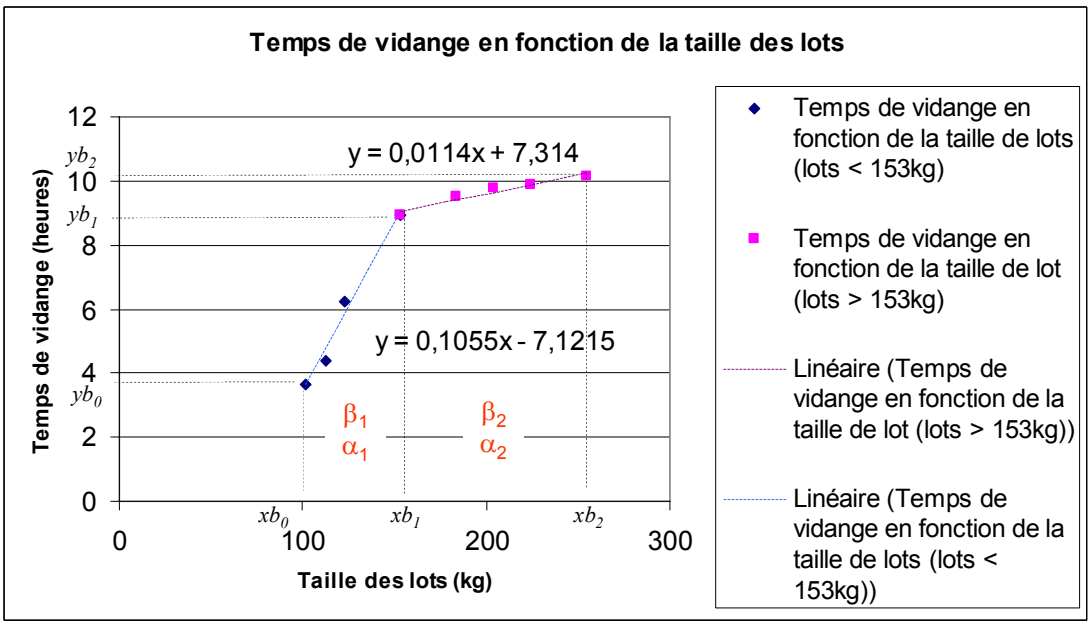


Figure 4.43 Prise en compte fine de la vidange par gravité

Soit une fonction f approximée par *n* segments de droite. Le segment *i* est délimité par les points de coordonnées (xb_{i-1}, yb_{i-1}) et (xb_i, yb_i).

En notant :

$\beta_{k,i,n}$: variable binaire qui vaut 1 si la valeur de la variable *x* est située dans l'intervalle (xb_{i-1}, xb_i).

$\alpha_{k,i,n}$: variable réelle telle que $\alpha_{k,i,n} = \dfrac{B_{i,n} - xb_{k-1}}{xb_k - xb_{k-1}}$

On obtient les contraintes :

$$\sum_{k=1}^{K} \beta_{k,i,n} \leq 1$$

$$0 \leq \alpha_{k,i,n} \leq \beta_{k,i,n}$$

Seulement un couple $(\beta_{k,i,n}, \alpha_{k,i,n})$ peut être différent de *0* avec :

$$B_{i,n} = \sum_{k=1}^{K} \beta_{k,i,n}.xb_k + \alpha_{k,i,n}.(xb_{k+1} - xb_k)$$

On obtient finalement :

- **Durée opératoire des tâches :**

$$pt_{i,n} = ft_i W_{i,n} + vt_i B_{i,n} + \sum_{k=1}^{K} \beta_{k,i,n}.yb_k + \alpha_{k,i,n}\,(yb_{k+1} - yb_k) \qquad \forall i \in I,\ \forall n \in N$$

4.5.2. Représentation RTN

Du point de vue de la représentation RTN, la prise en compte de fonctions non linéaires se fait en transformant la fonction non linéaire en fonction continue par morceaux :

$y = f(x)$ où f non linéaire ⇨ si $x \in [xb_{i-1}, xb_i]$, alors $y = \alpha_i\, x + \beta_i$ avec $i = 1,..,n$

Puis en associant une tâche à chaque morceau de la manière suivante :

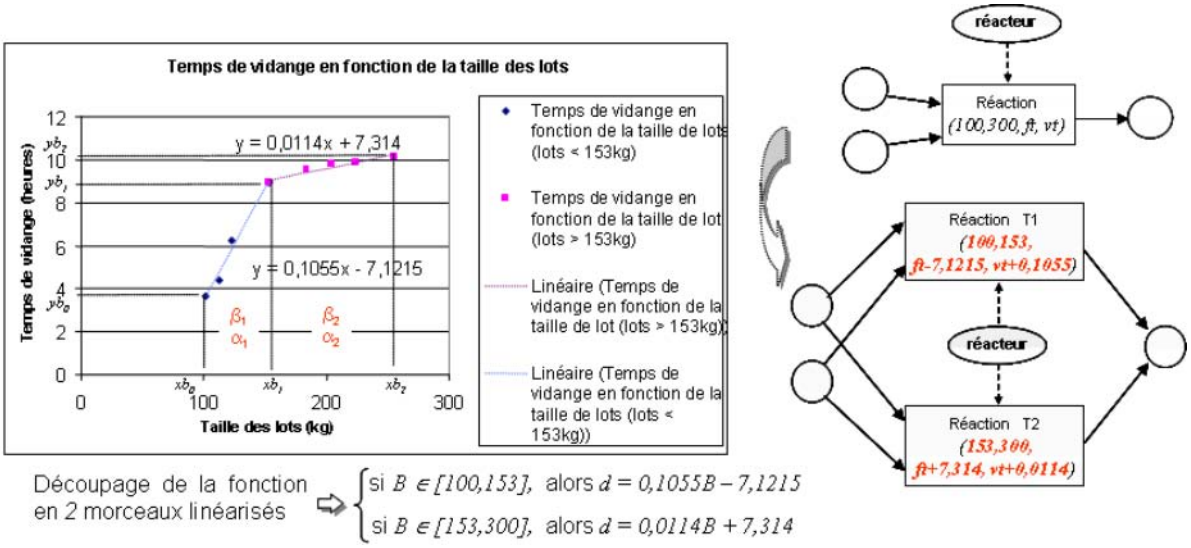


Figure 4.44 Exemple : Application à une réaction avec vidange par gravité

4.5.3. Exemple : Instance 4 - Le transfert par gravité

Paramètres :

- Nombre de points d'évènement : *N= 7* points.
- Ordres de fabrication à réaliser : *600 kg* de *P2*.
- Pas de délais de nettoyage des appareils.
- **Durées opératoires variables** et dépendantes de la taille du lot et de l'appareil utilisé comprenant une partie fixe, une partie variable et une partie continue par morceaux.

Ces durées opératoires peuvent être divisées en trois parties :

- les durées opératoires fixes : temps de réaction lié à la température contrôlée,
- les durées de remplissages proportionnelles aux tailles de lots (débits fixes),
- les temps de chauffe et de vidange (fonctions non linéaires des tailles de lots).

Nous avons identifié ces trois parties (Figure 4.42 et Figure 4.43) et les avons intégrées au modèle de l'opération de réaction.

Solution : solution sous-optimale obtenue en 1891 secondes

Tâche	t	Durée	Machine	Produit	Batch
T1	0.00	26.85	1	1	273.35
T1	26.85	12.84	1	1	101.65
T1	0.00	26.85	2	1	273.35
T1	26.85	12.84	2	1	101.65
T2	26.85	12.84	3	2	546.70
T2	39.69	10.78	3	2	203.30

Tableau 4.13 Tableau de résultats de l'ordonnancement de l'instance 4

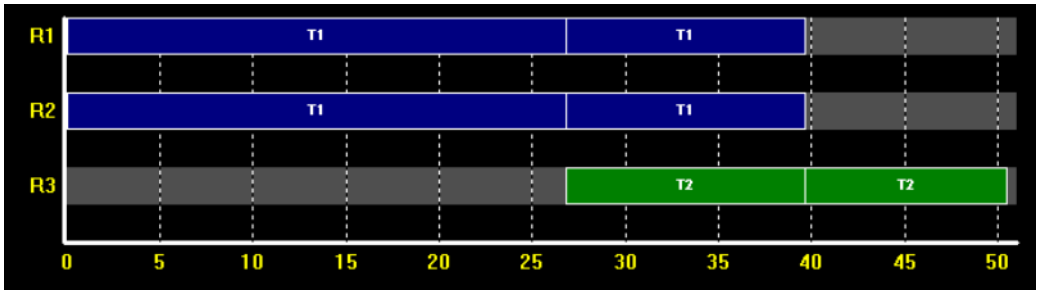


Figure 4.45 Diagramme de Gantt de l'instance 4 (durées en heures)

4.6. EXTENSION DU MODÈLE À LA GESTION AFFINÉE DES STOCKS

4.6.1. Modélisation affinée du pilotage des tâches

Le modèle mathématique décrit précédemment ne permet de prendre en compte que partiellement la limite sur la capacité des cuves de stockage (cf. contrainte (3.5)). En effet, dans cette formulation, les bilans matière (contraintes (3.3)) ne sont calculés qu'au lancement d'une tâche. Or, il faudrait qu'ils soient évalués également à la fin des tâches (notamment, lorsque plusieurs tâches produisent le même état *s*).

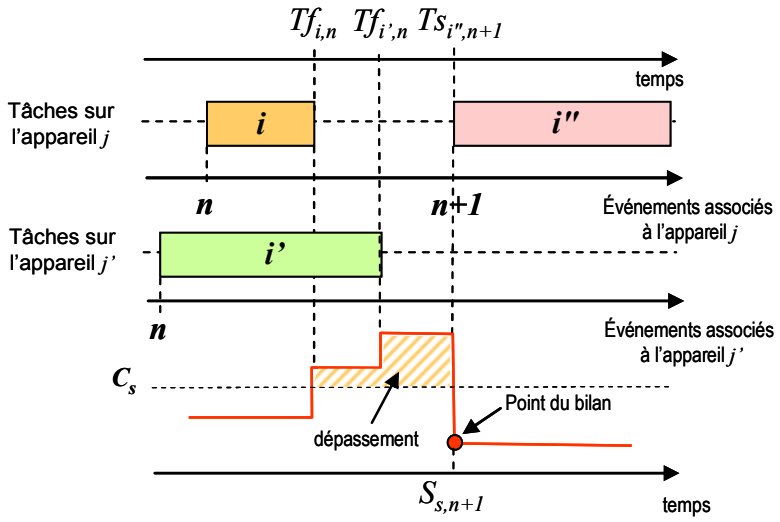


Figure 4.46 Exemple de dépassement de stock

La Figure 4.46 illustre ce point. On suppose que les tâches *i* et *i'* produisent le même état s et que la tâche *i''* consomme cet état. Si la date de fin des tâches *i* et *i'* lancées au point d'événement *n* n'est pas synchronisée avec la date de début de la tâche consommatrice *i''*, alors les variables $S_{s,n}$ ne permettent pas de représenter le niveau réel de stock et celui-ci peut transitoirement dépasser la capacité des cuves de stockage, même avec la présence de la contrainte (3.5). De ce fait, ce dépassement ne pourrait être détecté qu'au moment de la simulation de ce plan de production.

4.6.1.1. Contraintes additionnelles liées à la gestion affinée du pilotage des tâches de production

Des variables sont ajoutées permettant de suivre les démarrages et arrêts des tâches de manière plus fine, le modèle ci-dessous est alors obtenu :

- **Contraintes de base d'allocation, identiques aux contraintes (3.1)**

$$\sum_{i \in I_j} W_{i,n} \leq 1 \qquad \forall j \in J,\ \forall n \in N \tag{4.1}$$

Ces contraintes expriment le fait que pour chaque appareil *j* et à chaque point d'évènement *n*, seulement une des tâches *i* qui peuvent être réalisées dans cet appareil peut y prendre place.

- **Contraintes additionnelles d'allocation**

$$W_{i,n} = \sum_{n' \le n} Ws_{i,n'} - \sum_{n' < n} Wf_{i,n'} \qquad \forall i \in I,\ \forall n \in N \tag{4.2}$$

Les contraintes (4.2) permettent de relier la variable continue, $W_{i,n}$ aux variables binaires $Ws_{i,n}$ et $Wf_{i,n}$. Cette formulation permet d'exécuter des tâches sur plusieurs évènements, au lieu de restreindre leur fonctionnement à un seul et même évènement.

$$\sum_{n \in N} Ws_{i,n} = \sum_{n \in N} Wf_{i,n} \qquad \forall i \in I \tag{4.3}$$

Les contraintes (4.3) obligent les tâches qui ont été démarrées à s'arrêter au cours de l'horizon de temps.

$$Ws_{i,n} \le 1 - \sum_{n' \le n} Ws_{i,n'} + \sum_{n' < n} Wf_{i,n'} \qquad \forall i \in I,\ \forall n \in N \tag{4.4}$$

Les contraintes (4.4) empêchent tout lancement de tâches à l'évènement *n* qui auraient été lancées à des évènements précédents et non arrêtées lors de l'évènement *n*.

$$Wf_{i,n} \le \sum_{n' \le n} Ws_{i,n'} - \sum_{n' < n} Wf_{i,n'} \qquad \forall i \in I,\ \forall n \in N \tag{4.5}$$

Les contraintes (4.5), imposent qu'une tâche *i*, ne se termine pas à l'évènement *n*, bien qu'elle ait démarré à un évènement antérieur *n'*.

- **Contraintes de base de capacités des appareils de production, identiques aux contraintes (3.2)**

$$V_i^{\min} W_{i,n} \le B_{i,n} \le V_i^{\max} W_{i,n} \qquad \forall i \in I,\ \forall n \in N \tag{4.6}$$

Si $W_{i,n} = 1$, alors les contraintes (3.2) définissent les bornes minimales et maximales pour la taille des batch $B_{i,n}$, sinon elles forcent la valeur de la variable $B_{i,n}$ à zéro.

- **Contraintes additionnelles de cohérence des tailles de lots**

$$B_{i,n} \le B_{i,n-1} + V_i^{\max}[1 - W_{i,n-1} + Wf_{i,n-1}] \qquad \forall i \in I,\ \forall n \in N \mid n > 1 \tag{4.7}$$

$$B_{i,n} \ge B_{i,n-1} - V_i^{\max}[1 - W_{i,n-1} + Wf_{i,n-1}] \qquad \forall i \in I,\ \forall n \in N \mid n > 1 \tag{4.8}$$

Les contraintes (4.7) et (4.8) représentent les relations qui lient les tailles de lots d'une tâche *i*, pour deux évènements consécutifs *n-1* et *n*. Ces contraintes sont nécessaires car les tâches peuvent s'étendre sur plusieurs évènements dans cette formulation. Aussi, la cohérence des tailles de lots d'une même tâche pour des évènements consécutifs doit être assurée. Néanmoins, si une tâche est active et se termine à l'évènement *n-1*, alors les tailles de lots des évènements *n* et *n-1* ne sont en aucun cas liées.

$$B_{i,n}^{s} \le B_{i,n} \qquad \forall i \in I,\ \forall n \in N \tag{4.9}$$

$$B_{i,n}^{s} \le V_i^{\max} Ws_{i,n} \qquad \forall i \in I,\ \forall n \in N \tag{4.10}$$

$$B_{i,n}^{s} \ge B_{i,n} - V_i^{\max}[1 - Ws_{i,n}] \qquad \forall i \in I,\ \forall n \in N \tag{4.11}$$

Les contraintes (4.9) à (4.11), permettent de relier les variables $B_{i,n}$ et $B_{i,n}^{s}$, où $B_{i,n}^{s}$ représente la quantité de matière présente au démarrage de la tâche *i* à l'évènement *n*. Si la tâche *i,*

démarre à l évènement *n* alors $B^s_{i,n} = B_{i,n}$ sinon $B^s_{i,n} = 0$. De la même manière, les contraintes (4.12) à (4.14) relient les variables $B_{i,n}$ et $B^f_{i,n}$.

$$B^f_{i,n} \le B_{i,n} \qquad \forall i \in I,\ \forall n \in N \tag{4.12}$$

$$B^f_{i,n} \le V_i^{\max} Wf_{i,n} \qquad \forall i \in I,\ \forall n \in N \tag{4.13}$$

$$B^f_{i,n} \ge B_{i,n} - V_i^{\max}[1 - Wf_{i,n}] \qquad \forall i \in I,\ \forall n \in N \tag{4.14}$$

- **Bilans matière sur les états, en remplacement des contraintes (3.3), (3.4) et (3.5)**

Les contraintes de bilans sont adaptées afin de prendre en compte les nouvelles variables représentant les évolutions de quantités de matière dans les tâches.

$$S_{s,n} = S_{s,n-1} + \sum_{i \in I^p_s} \rho^p_{i,s} B^f{}_{i,n-1} - \sum_{i \in I^c_s} \rho^c_{i,s} B^s{}_{i,n} \qquad \forall s \in S,\ \forall n \in N \mid n>1 \tag{4.15}$$

Etat initial en remplacement de la contrainte (3.4)

$$S_{s,1} = S0_s + \sum_{i \in I^c_s} \rho^c_{i,s} B^s{}_{i,1} \qquad \forall s \in S \tag{4.16}$$

Contraintes de capacité des cuves de stockage

$$S_{s,n} \le C_s \qquad \forall s \in S,\ \forall n \in N \tag{4.17}$$

Ces contraintes fixent une borne supérieure sur les niveaux de stock de chaque état *s*.

- **Durées opératoires des tâches de production, en remplacement des contraintes (3.8) et (3.9)**

$$Tf_{i,n} \ge Ts_{i,n} \qquad \forall i \in I,\ \forall n \in N \tag{4.18}$$

$$Tf_{i,n} \le Ts_{i,n} + H W_{i,n} \qquad \forall i \in I,\ \forall n \in N \tag{4.19}$$

Les contraintes (4.18) et (4.19) représentent les relations entre les dates de début et de fin des tâches *i* à l'évènement *n*. Comme les tâches peuvent s'étendre sur plusieurs évènements, la date de fin n'est pas attribuée à l'évènement dès le lancement de la tâche mais doit être supérieure ou égale à la date de lancement (contraintes (4.18)).Si la tâche *i* n'est pas réalisée à l'évènement *n* alors les dates de début et de fin sont égales (contraintes (4.19)) ; dans le cas contraire, la contrainte est relaxée.

$$pt_{i,n} = ft_i Ws_{i,n} + vt_i B^s{}_{i,n} \qquad \forall i \in I,\ \forall n \in N \tag{4.20}$$

$$Tf_{i,n'} - Ts_{i,n} \ge pt_{i,n} - H[1 - Ws_{i,n}] - H[1 - Wf_{i,n'}] - H[\sum_{n \le n'' < n'} Wf_{i,n''}]$$

$$\forall i \in I,\ \forall n \in N \tag{4.21}$$

$$Tf_{i,n'} - Ts_{i,n} \le pt_{i,n} + H[1 - Ws_{i,n}] + H[1 - Wf_{i,n'}] + H[\sum_{n \le n'' < n'} Wf_{i,n''}]$$

$$\forall i \notin Ips,\ \forall n \in N \tag{4.22}$$

Les contraintes (4.20) et (4.21) permettent de relier la date de début d'une tâche *i* à l'évènement *n* avec sa date de fin à un évènement *n'*. Si la tâche *i* commence à l'évènement *n* et se termine à l'évènement *n'* alors $Tf_{i,n'} = Ts_{i,n} + pt_{i,n}$. Cependant, si la tâche *i* se termine à l'évènement *n''* où $n \le n'' \le n'$ ou bien si elle ne se termine pas à l'évènement *n'*, alors ces contraintes sont relaxées. Il est important de noter que la contrainte (4.22) ne s'applique qu'à des tâches permettant à la fois de réaliser des opérations de réaction et de stocker de la matière. Par conséquent, la date de fin n'est pas fixée pour les opérations permettant à la fois de faire de la production et du stockage.

- **Contraintes de base de séquence des tâches, adaptées de (3.10), (3.11) et (3.12)**

Enchaînement d'une même tâche sur le même appareil

$$Ts_{i,n} \ge Tf_{i,n-1} \qquad \forall i \in I,\ \forall n \in N \mid n>1 \tag{4.23}$$

$$Ts_{i,n} \le Tf_{i,n-1} + H[1 - W_{i,n-1} + Wf_{i,n-1}] \qquad \forall i \in I,\ \forall n \in N \mid n>1 \tag{4.24}$$

Ces contraintes de séquence établissent qu'une nouvelle tâche *i* commençant au point *n* ne peut démarrer qu'après la fin d'une éventuelle autre tâche *i* réalisée sur le même appareil *j*. Ces contraintes forcent ces dates à être égales si la tâche active à l'évènement précédent se termine. Cependant, les contraintes (4.24) sont relaxées si les tâches actives à l'évènement *n-1* ne se terminent pas à la fin de cet évènement.

Enchaînement de tâches différentes sur le même appareil

$$Ts_{i,n} \ge Tf_{i',n-1} + \tau_{i,i'} W_{i',n} - H(1 - W_{i',n-1})$$

$$\forall j \in J, \forall i \in Ij, \forall i' \in Ij \mid i \ne i', \forall n \in N \mid n>1 \tag{4.25}$$

Les contraintes (4.25) sont écrites pour des tâches *i* et *i'* exécutées consécutivement dans le même appareil *j* respectivement au point *n-1* et *n*. Ainsi, si une tâche *i'* a été lancée au point *n-1*, alors la date de début d'une tâche *i* suivante doit être supérieure à la date de fin de cette tâche *i'* (c.a.d, $Ts_{i,n} \ge Tf_{i',n-1}$). Par contre, s'il n'y a pas de tâche *i'* lancée au point *n-1*, alors la contrainte est relaxée. Un temps de nettoyage $\tau_{i',i}$ dépendant de la séquence peut aussi être pris en compte par cette contrainte.

Enchaînement de tâches différentes sur des appareils différents

$$Ts_{i,n} \ge Tf_{i',n-1} - H[1 - Wf_{i',n-1}]$$

$$\forall s \in S, \forall i \in I_s^c, \forall i' \in I_s^p, \forall j \in Ji, \forall j' \in Ji' \mid j \ne j', \forall n \in N \mid n>1 \tag{4.26}$$

Les contraintes (4.26) sont écrites pour des tâches *i* et *i'* exécutées consécutivement dans des appareils différents *j* et *j'* respectivement au point *n-1* et *n*. Ainsi, si une tâche *i'* a été lancée au point *n-1*, alors la date de début d'une tâche *i* suivante doit être supérieure à la date de fin de cette tâche *i'* (c.a.d, $Ts_{i,n} \ge Tf_{i',n-1}$). Par contre, s'il n'y a pas de tâche *i'* lancée au point *n-1*, alors la contrainte est relaxée et les deux dates ne sont pas liées.

- **Contraintes de séquence : pas de temps d'attente entre les tâches (Politique Zero-Wait)**

$$Ts_{i,n} \le Tf_{i',n-1} + H[2 - Wf_{i',n-1} - Ws_{i,n}]$$

$$\forall s \in S^Z, \forall i \in I_s^c, \forall i' \in I_s^p, \forall j \in Ji, \forall j' \in Ji' \mid j \ne j', \forall n \in N \mid n>1 \tag{4.27}$$

Les contraintes (4.27) sont écrites pour des tâches *i* et *i'* qui doivent s'exécuter consécutivement avec des conditions de type « Zero-Wait ». Combinées avec les contraintes (4.26), ces contraintes forcent la tâche *i* à s'exécuter immédiatement après la fin de la tâche *i'* à condition que les deux tâches soient actives.

- **Contraintes additionnelles pour borner les variables continues**

$$Tf_{i,n} \leq H \qquad \forall i \in I,\ \forall n \in N \tag{4.28}$$

$$Ts_{i,n} \leq H \qquad \forall i \in I,\ \forall n \in N \tag{4.29}$$

$$0 < W_{i,n} < 1 \qquad \forall i \in I,\ \forall n \in N \tag{4.30}$$

- **Satisfaction des OF adaptation des contraintes (3.6)**

$$SF_s = S_{s,N} + \sum_{i \in I_s^p} \rho_{i,s}^p B^f{}_{i,N} \qquad \forall s \in S \tag{4.31}$$

- **Critère à optimiser (cf. (3.14)):**

$$\min \quad \left(a.DureePlan + \sum_{s \in S} h_s SF_s + \sum_{s \in S} \sum_{n \in N} h_s S_{s,n} \right) \tag{4.32}$$

Cette fonction *objectif* minimise la durée du plan, ainsi que les niveaux de stock. Cela s'avère nécessaire afin de lancer des lots de volume « juste nécessaire ». En effet, pour des tâches dont la durée est indépendante de la taille du lot, un volume supérieur pourrait être produit sans pour autant allonger la durée du plan.

4.6.1.2. Nomenclature complémentaire

Ensembles :

I : ensemble des tâches de la recette,
S^f : ensemble des états de la recette avec une politique de stockage en quantité finie (FIS),
S^z : ensemble des états de la recette avec des contraintes de type Zero-Wait,
S^n : ensemble des états de la recette sans stockage intermédiaire,

Variables binaires :

$Ws_{i,n}$: vaut 1 si une tâche *i* est lancée au point d'évènement *n*,
$Wf_{i,n}$: vaut 1 si une tâche *i* est arrêtée au point d'évènement *n*,

Variables continues :

$B_{i,n}$: quantité de matière stockée par la tâche *i* au point d'évènement *n*,
$Bs_{i,n}$: quantité de matière consommée par la tâche *i* au point d'évènement *n*,
$Bf_{i,n}$: quantité de matière libérée par la tâche *i* au point d'évènement *n*,
$W_{i,n}$: vaut 1 si une tâche *i* est en cours d'exécution au point d'évènement *n*,

Paramètres :

H : Horizon de temps,

4.6.1.3. Application à l'exemple

Paramètres :

- Nombre de points d'évènement : *N=7* points.
- Ordres de fabrication à réaliser : *1200 kg* de *P2*.
- Pas de délais de nettoyage des appareils.
- **Durées opératoires variables** et dépendantes de la taille du lot et de l'appareil utilisé comprenant une partie fixe et une partie variable :
-

n° Tâche	1	2	3
Ressource	BR1	BR2	Colonne
df (h)	7,9	7,9	9.56
dv (h/kg)	0,0535	0,0535	0,006

Tableau 4.14 Durées opératoires variables

Solution : solution optimale obtenue en 121,7 secondes

Tâche	t	Durée	Machine	Produit	Batch
T1	0.00	24.27	1	1	306.00
T1	24.27	24.27	1	1	306.00
T1	48.54	15.28	1	1	138.00
T1	0.00	24.27	2	1	306.00
T1	24.27	24.27	2	1	306.00
T1	48.54	15.28	2	1	138.00
T2	27.80	11.40	3	2	306.00
T2	39.20	11.40	3	2	306.00
T2	50.59	13.23	3	2	612.00
T2	63.83	11.22	3	2	276.00

Tableau 4.15 Tableau de résultats de l'ordonnancement

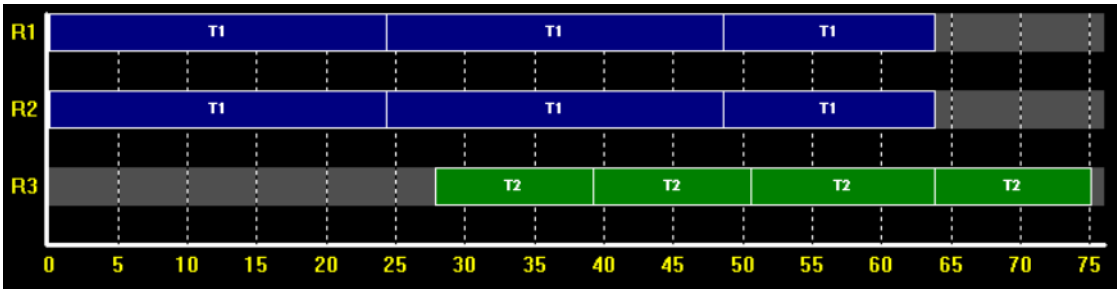


Figure 4.47 Diagramme de Gantt

Les dépassements sont observés (Figure 4.48) aux dates $t = 24{,}27\ h$ et $t = 48{,}54\ h$. En effet, les dates de fin des tâches T1 et les dates de début des tâches T2 ne sont pas synchronisées.

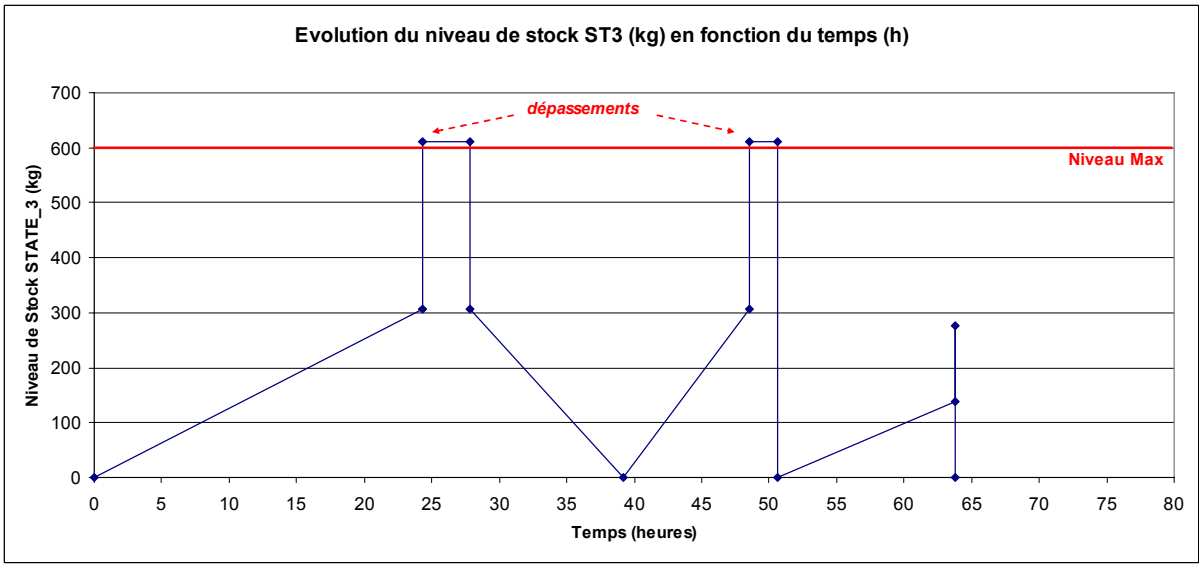


Figure 4.48 Evolution du niveau de stock de l'état 3 (en kg) en fonction du temps (en h)

4.6.2. Modélisation des tâches de stockage

Afin de gérer de manière plus rigoureuse cette limite, des contraintes supplémentaires doivent être ajoutées au modèle précédent (basées sur le modèle décrit dans [Janak *et al.,* 2004]).

Le principe est d'introduire dans le modèle des tâches fictives dites « de stockage ». Celles-ci sont lancées dés qu'une tâche de production se termine. De cette façon, des points d'évènement supplémentaires sont générés et la valeur des batch de ces tâches « de stockage » correspond à la quantité d'état s en stock entre 2 points d'évènement successifs.

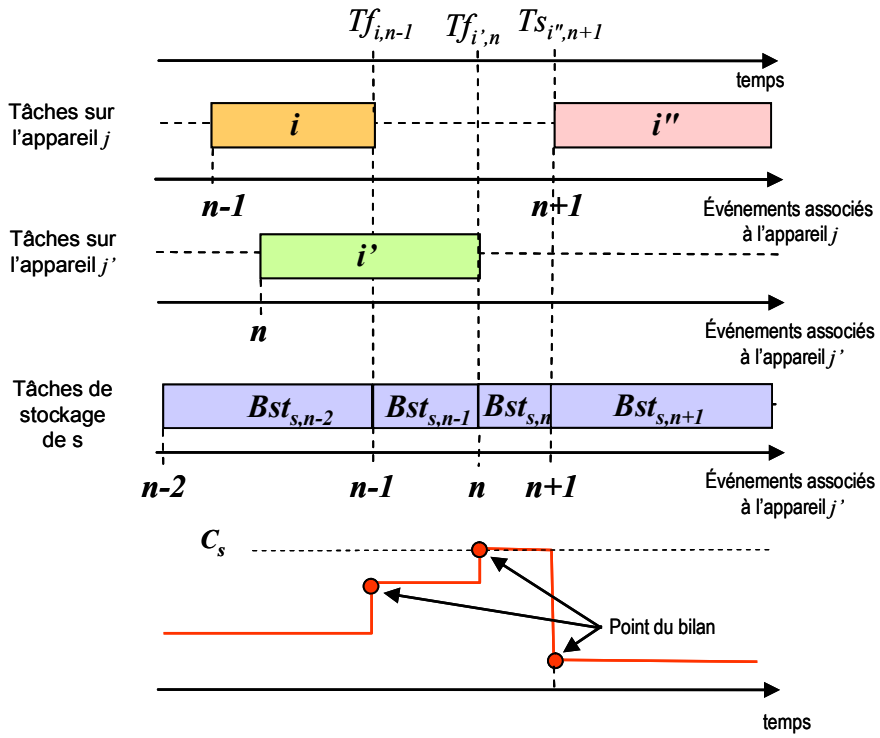


Figure 4.49 Gestion affinée des stocks

Certaines contraintes du modèle sont adaptées et de nouvelles contraintes ajoutées afin de prendre en compte ces tâches fictives de « stockage » de la manière suivante :

- **Contraintes de capacités : Tâches fictives de « stockage »**

$$Bst_{s,n} \le C_s \qquad \forall s \in S,\ \forall n \in N \tag{5.1}$$

Les contraintes (5.1) représentent les quantités maximales de matière qui peuvent être stockées par les *tâches de stockage s* à l'évènement *n*. Elles représentent les bornes maximales de quantités de matière qui peuvent être stockées pour chaque état *s* au travers d'une tâche de stockage.

- **Bilans matière sur les états en remplacement de la contrainte (4.15)**

Les contraintes de bilans matière sont modifiées afin d'intégrer les tâches fictives de stockage, l'évolution des quantités de matière dans les états est gérée au moyen des différentes tâches fictives de stockage.

$$S_{s,n} = S_{s,n-1} + \sum_{i \in I_s^p} \rho_{i,s}^p B^f{}_{i,n-1} - \sum_{i \in I_s^c} \rho_{i,s}^c B^s{}_{i,n} + \sum Bst_{s,n-1} - \sum Bst_{s,n}$$

$$\forall s \in S,\ \forall n \in N \mid n>1 \tag{5.2}$$

$$S_{s,n} = 0 \qquad \forall s \in S \tilde{z},\ S f,\ S n,\ \forall n \in N \tag{5.3}$$

Etat initial en remplacement de la contrainte (4.16)

$$S_{s,1} = S0_s + \sum_{i \in I_s^c} \rho_{i,s}^c B^s{}_{i,1} - \sum Bst_{s,1} \qquad \forall s \in S \tag{5.4}$$

- **Contraintes de durées : Tâches fictives de « stockage »**

$$Tst^f_{s,n} \geq Tst^s_{s,n} \qquad \forall s \in S,\ \forall n \in N \tag{5.5}$$

Comme pour des tâches classiques, ces contraintes lient les dates de début des tâches à leurs dates de fin. Les dates de fin doivent toujours être supérieures ou égales aux dates de début des tâches.

- **Contraintes de séquence : Tâches fictives de « stockage »**

$$Ts_{i,n} \geq Tst^f_{s,n-1} \qquad \forall s \in S, \forall i \in I^c_s,\ n \in N \mid n>1 \tag{5.6}$$

$$Ts_{i,n} \leq Tst^f_{s,n-1} + H[1 - Ws_{i,n}] \qquad \forall s \in S\, f, \forall i \in I^c_s,\ n \in N \mid n>1 \tag{5.7}$$

Les contraintes (5.6) et (5.7) conditionnent la date de début d'une *tâche de production i* à l'évènement *n* à la date de fin de la *tâche fictive de stockage s* à l'évènement précédent *n-1*. La contrainte (5.7) est seulement écrite pour les états de types *Stockage Intermédiaire Fini* (Finite Intermediate Storage - FIS), si une tâche *i* démarre à l'évènement *n* et consomme un état avec un FIS, alors nous obtenons $Ts_{i,n} = Tst^f_{s,n-1}$ pour l'état *s*. Cependant, si la tâche *i* ne démarre pas à l'évènement *n* ou si l'état *s* ne nécessite pas de FIS alors les dates pour les deux tâches ne sont pas forcées. Aussi, les dates de fin des tâches de stockage des états avec FIS et les dates de début des tâches qui les consomment sont reliées.

$$Tst^s_{s,n} \geq Tf_{i',n-1} - H[1 - Wf_{i',n-1}] \qquad \forall s \in S, \forall i' \in I^p_s,\ n \in N \mid n>1 \tag{5.8}$$

$$Tst^s_{s,n} \leq Tf_{i',n-1} + H[1 - Wf_{i',n-1}] \qquad \forall s \in S, \forall i' \in I^p_s,\ n \in N \mid n>1 \tag{5.9}$$

De la même façon que pour les contraintes (5.6) et (5.7), les contraintes (5.8) et (5.9) relient les dates de début des tâches de stockage avec FIS aux dates de fin des tâches qui les produisent. Aussi, les dates de fin des tâches qui produisent des états avec FIS et les dates de début des tâches de stockages de ces états sont reliées.

$$Tst^s_{s,n} = Tst^f_{s,n-1} \qquad \forall s \in S,\ n \in N \mid n>1 \tag{5.10}$$

Les contraintes (5.10) relient les dates de début et de fin des tâches de stockage à deux évènements consécutifs *n-1* et *n*. Ces contraintes assurent en liaison avec les contraintes (5.6) et (5.10), que les dates pour les tâches de stockage des états avec FIS soient forcées afin d'assurer le respect des contraintes de limitation de stockage pour ces états.

- **Satisfaction des OFs en remplacement de la contrainte (4.30)**

La contrainte (4.30) est modifiée pour prendre en compte la dernière tâche fictive de stockage dans le bilan matière final.

$$SF_s = S_{s,N} + \sum_{i \in I^p_s} \rho^p_{i,s} B^f_{i,N} + \sum Bst_{s,N} \qquad \forall s \in S \tag{5.11}$$

- **Contraintes de resserrement**

Les contraintes de resserrement reprises de [Maravelias et Grossmann, 2003] et de [Janak *et al.,* 2004] permettent d'accélérer la convergence.

$$\sum_{i \in I_j} \sum_{n \in N} pt_{i,n} \leq H \qquad \forall j \in J \tag{5.12}$$

La première contrainte (5.12) force la somme des durées des tâches allouées à un appareil à être inférieure à l'horizon de temps.

De plus, cette condition est aussi forcée pour chaque appareil *j* à chaque évènement *n*, dans les contraintes (5.13) et (5.14), pour lesquelles $tt^s_{j,n}$ et $tt^f_{j,n}$ représentent respectivement les dates de début et de fin de la tâche active sur l'appareil *j* à l'évènement *n* et définies suivant les contraintes (5.15) à (5.18).

$$\sum_{i\in I_j}\sum_{n'\geq n} pt_{i,n'} \leq H - tt^s_{j,n} \qquad \forall j\in J,\ \forall n,n'\in N, \tag{5.13}$$

$$\sum_{i\in I_j}\sum_{n'<n} pt_{i,n'} \leq tt^f_{j,n-1} + H[1-\sum_{i\in I_j} Wf_{i,n-1}] \qquad \forall j\in J,\ \forall n,n'\in N \mid n>1, \tag{5.14}$$

Ainsi, les contraintes (5.13) forcent la somme des durées des tâches démarrant sur l'appareil *j* à un évènement *n* ou *supérieur à n* à être inférieure ou égale au temps restant. De la même manière, les contraintes (5.14) forcent la somme des durées des tâches réalisées sur l'appareil *j* à l'évènement *n* ou *inférieur à n* à être inférieure ou égale au temps écoulé.

$$tt^s_{j,n} \leq Ts_{i,n} + H[1-Ws_{i,n}] \qquad \forall j\in J, i\in I_j \quad \forall n\in N, \tag{5.15}$$

$$tt^s_{j,n} \geq Ts_{i,n} - H[1-Ws_{i,n}] \qquad \forall j\in J, i\in I_j \quad \forall n\in N, \tag{5.16}$$

$$tt^f_{j,n} \leq Tf_{i,n} + H[1-Wf_{i,n}] \qquad \forall j\in J, i\in I_j \quad \forall n\in N, \tag{5.17}$$

$$tt^f_{j,n} \geq Tf_{i,n} - H[1-Wf_{i,n}] \qquad \forall j\in J, i\in I_j \quad \forall n\in N, \tag{5.18}$$

4.6.2.1. Nomenclature complémentaire

Variables continues :

$Tst^s_{s,n}$: date à laquelle démarre la tâche fictive de stockage *s* au point d'évènement *n*,

$Tst^f_{s,n}$: date à laquelle se termine la tâche fictive de stockage *s* au point d'évènement *n*,

$Bst_{s,n}$: quantité de matière stockée par la tâche fictive *s* au point d'évènement *n*,

$tt^s_{j,n}$: date de début de la tâche active sur l'appareil *j* à l'évènement *n*,

$tt^f_{j,n}$: date de fin de la tâche active sur l'appareil *j* à l'évènement *n*,

4.6.2.2. Application à l'exemple

Paramètres :

- Nombre de points d'évènement : *N=7* points.
- Ordres de fabrication à réaliser : *1200 kg* de *P2*.
- Pas de délais de nettoyage des appareils.

- **Durées opératoires variables** et dépendantes de la taille du lot et de l'appareil utilisé comprenant une partie fixe et une partie variable :

-

n° Tâche	1	2	3
Ressource	BR1	BR2	Colonne
df (h)	7,9	7,9	9.56
dv (h/kg)	0,0535	0,0535	0,006

Tableau 4.16 Durées opératoires variables

Solution : obtenue en 1114,6 secondes

Tâche	t	Durée	Machine	Produit	Batch
T1	0.00	24.27	1	1	306.00
T1	24.27	24.27	1	1	306.00
T1	48.54	15.28	1	1	138.00
T1	0.00	24.27	2	1	306.00
T1	24.27	24.27	2	1	306.00
T1	48.54	15.28	2	1	138.00
T2	24.27	13.23	3	2	612.00
T2	48.54	13.23	3	2	612.00
T2	63.82	11.22	3	2	276.00

Tableau 4.17 Tableau de résultats de l'ordonnancement

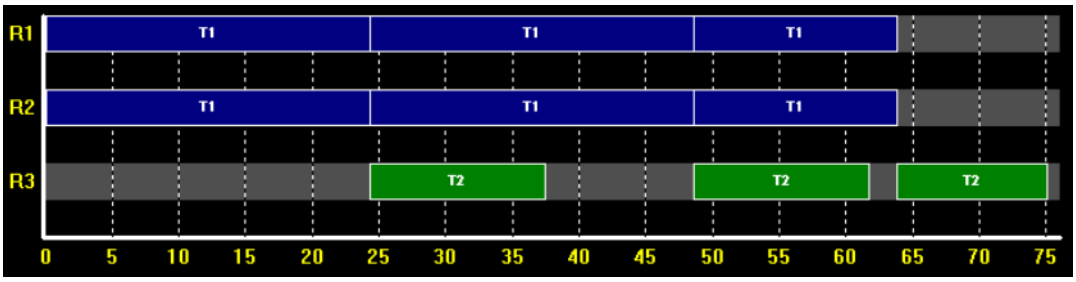


Figure 4.50 Diagramme de Gantt

Les dépassements ne sont plus observés aux dates $t = 24,27\ h$ et $t = 48,54\ h$. En effet, les dates de fin des tâches T1 et les dates de début des tâches T2 sont maintenant synchronisées.

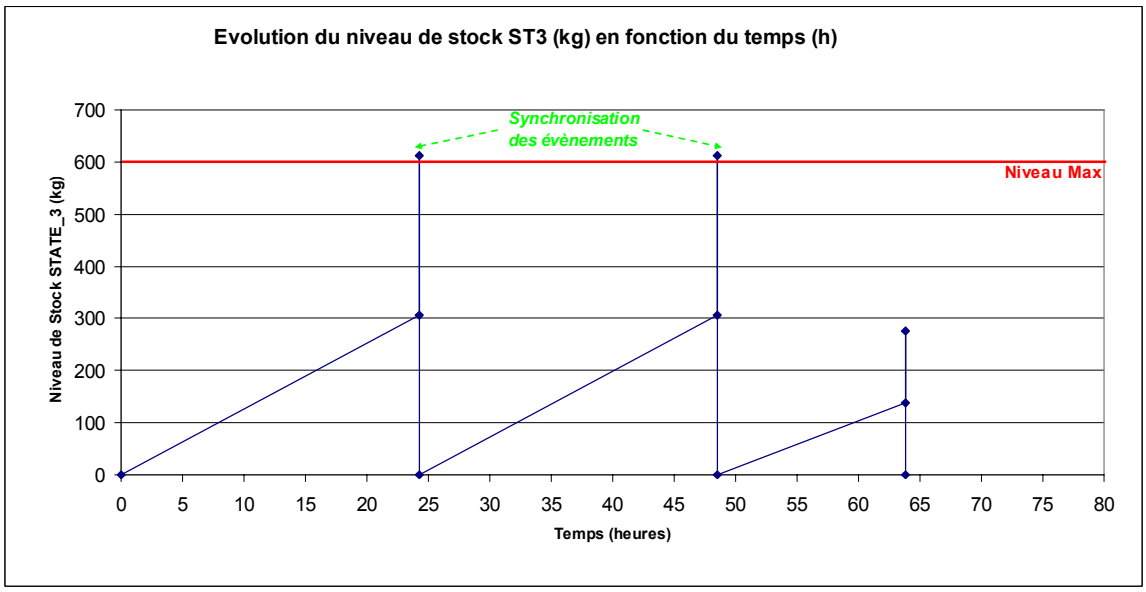
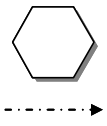


Figure 4.51 Evolution du niveau de stock de l'état 3 (en kg) en fonction du temps (en h)

4.7. EXTENSION DU MODÈLE À LA PRISE EN COMPTE D'OPÉRATIONS MULTI-MODALES

4.7.1. Extension des éléments sémantiques

L'extension du modèle présenté vise à prendre en compte les ressources multimodales. Cette extension permet de gérer la transition entre des tâches de démarrage ou d'arrêt et des tâches de production en fonctionnement nominal.



nœuds ***Etat ressource*** représentent l'état dans lequel se situe un appareil lorsque celui-ci est multi-modale

arc ***Transition Etat*** modélise le changement d'état d'une ressource

Figure 4.52 Arcs et nœuds définissant la Gestion de l'état d'un appareil en RTN

Règle 4.7 : Règle de modélisation associée aux Etats ressources

A chaque changement d'état d'une ressource correspond une tâche.

A un instant donné, une ressource ne peut être que dans un seul état.

Représente un changement d'état réversible (avec bouclage) ou irréversible (sans bouclage).

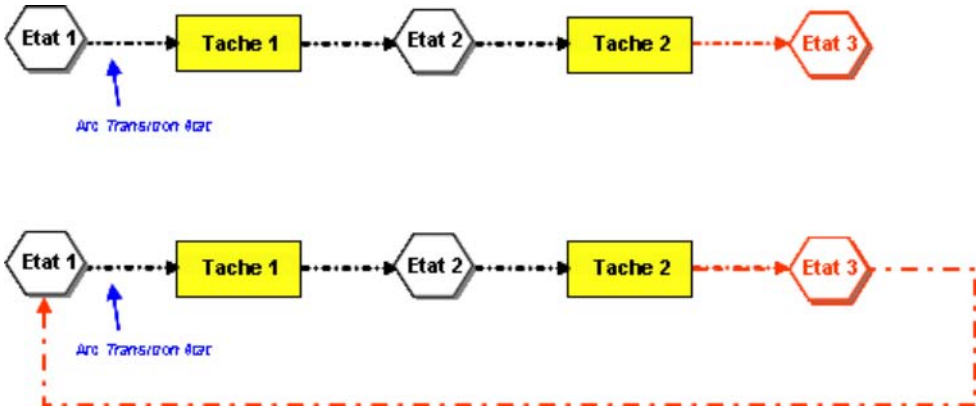


Figure 4.53 Exemple de changement d'état irréversible et réversible

4.7.2. Extension du modèle

- **« Consommation » des Etats Ressources par les tâches de production**

Des tailles de lots sont associées aux Bilans Etats Ressources, en effet, l'exécution d'une phase peut nécessiter deux occurrences de la phase qui la précède. Nous avons donc défini une taille variable de lot bien qu'elle soit fixée à 1 dans l'exemple.

$$Br_{i,n} = W_{i,n} \qquad \forall i \in I,\ \forall n \in N \tag{6.1}$$

- **Bilans « matière » standard sur les Etats Ressources**

$$Sr_{sr,n} = Sr_{sr,n-1} + \sum_{i \in I^p_{sr}} \rho r^p_{i,sr} Br_{i,n-1} - \sum_{i \in I^c_{sr}} \rho r^c_{i,sr} Br_{i,n}$$

$$\forall sr \in Sr,\ \forall n \in N \mid n>1 \tag{6.2}$$

Etat initial des Etats Ressources

$$Sr_{sr,1} = Sr0_{sr} + \sum_{i \in I^c_{sr}} \rho r^c_{i,sr} B^s{}_{i,1}$$

$$\forall sr \in Sr \tag{6.3}$$

Contraintes de capacité des Etats Ressources

$$Sr_{sr,n} \le Cr_{sr} \qquad \forall sr \in Sr,\ \forall n \in N \tag{6.4}$$

Extension formalisation graphique 1 : Contraintes liées au Bilan « matière » sur les Etats Ressources (6.2), (6.3) et (6.4)

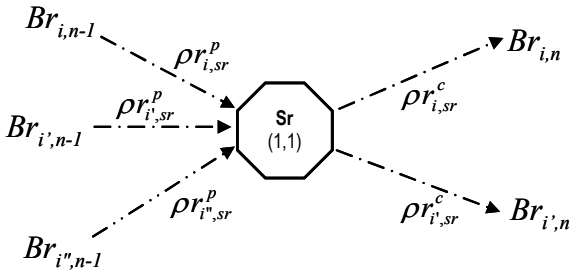


Figure 4.54 Etat initial et capacités de « stockage » des Etats Ressources en RTN

- **Contraintes de réinitialisation des Etats en fin de cycle de production**

Une contrainte identique à la Satisfaction des OF est utilisée afin de libérer les ressources en fin de campagne, en les laissant dans leur état initial.

$$SFr_s = Sr_{s,N} + \sum_{i \in I_s^p} \rho r_{i,s}^p Br_{i,N} \qquad \forall s \in S \tag{6.5}$$

$$SFr_s \geq Sr0_s \qquad \forall s \in S \tag{6.6}$$

- **Contraintes de durées : Tâches « fictives » de stockage des Etats Ressources**

$$Tstr^f{}_{sr,n} \geq Tstr^s{}_{sr,n} \qquad \forall sr \in Sr, \ \forall n \in N \tag{6.7}$$

Comme pour des tâches classiques, ces contraintes lient les dates de début des tâches à leurs dates de fin. Les dates de fin doivent toujours être supérieures ou égales aux dates de début des tâches.

- **Contraintes de séquence : Tâches fictives de « stockage »**

$$Ts_{i,n} \geq Tstr^f{}_{sr,n-1} \qquad \forall sr \in Srf, \forall i \in I^c_{sr}, \ n \in N \mid n>1 \tag{6.8}$$

$$Ts_{i,n} \leq Tstr^f{}_{sr,n-1} + H[1 - Ws_{i,n}] \qquad \forall sr \in Sr, \forall i \in I^c_{sr}, \ n \in N \mid n>1 \tag{6.9}$$

Les contraintes (6.8) et (6.9) conditionnent la date de début d'une *tâche de production i* à l'évènement *n* à date de fin de la *tâche fictive de stockage d'Etat Ressources sr* à l'évènement précédent *n-1*. La contrainte (6.9) est écrite pour tous les Etats Ressources car ils sont considérés de type *Stockage Intermédiaire Fini*.

En effet, quel que soit l'horizon de production les Ressources sont exécutées un nombre fini de fois. Si une tâche *i* démarre à l'évènement *n* et consomme un état avec un FIS, alors nous obtenons $Ts_{i,n} = Tstr^f{}_{s,n-1}$ pour l'état *sr*. Aussi, les dates de fin des tâches de stockage des états avec FIS et les dates de début des tâches qui les consomment sont reliées.

$$Tstr^{s}_{sr,n} \geq Tf_{i',n-1} - H[1 - Wf_{i',n-1}] \qquad \forall sr \in Sr, \forall i' \in I^{p}_{sr}, n \in N \mid n>1 \qquad (6.10)$$

$$Tstr^{s}_{sr,n} \leq Tf_{i',n-1} + H[1 - Wf_{i',n-1}] \qquad \forall sr \in Sr, \forall i' \in I^{p}_{sr}, n \in N \mid n>1 \qquad (6.11)$$

De la même façon que pour les contraintes (6.8) et (6.9), les contraintes (6.10) et (6.11) relient les dates de début des tâches de stockage des Etats Ressources aux dates de fin des tâches qui les produisent. Aussi, les dates de fin des tâches qui produisent des Etats Ressources et les dates de début des tâches de stockage de ces Etats Ressources sont reliées.

$$Tstr^{s}_{sr,n} = Tstr^{f}_{sr,n-1} \qquad \forall s \in S, n \in N \mid n>1 \qquad (6.12)$$

- **Contraintes de séquence : Pas de temps d'attente entre les tâches (Politique Zero-Wait)**

$$Ts_{i,n} \leq Tstr^{f}_{sr,n-1} + H[2 - Wf_{i',n-1} - Ws_{i,n}]$$

$$\forall sr \in Sr^{ZW}, \forall i \in I^{c}_{sr}, \forall i' \in I^{p}_{sr}, \forall j \in J_{i}, \forall j' \in J_{i'} \mid j \neq j', \forall n \in N \mid n>1 \qquad (6.13)$$

Les contraintes (4.27) sont écrites pour des tâches *i* et *i'* reliées par un même Etat Ressource et qui doivent s'exécuter consécutivement avec des conditions de type « Zero-Wait ». Combinées avec les contraintes (4.26), ces contraintes forcent la tâche *i* à s'exécuter immédiatement après la fin de la tâche *i'* à condition que les deux tâches soient actives.

Extension formalisation graphique 2 : Etat Ressource avec contraintes de types Zero-Wait

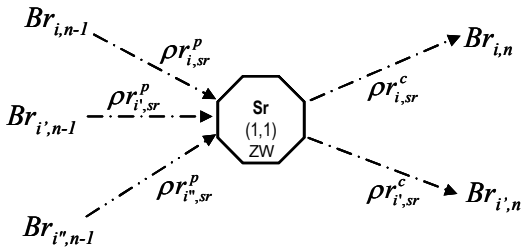


Figure 4.55 Etat Ressource avec contraintes de types Zero-Wait

- **Contraintes additionnelles pour borner les variables continues**

$$Tstr^{s}_{sr,1} \leq 0 \qquad \forall sr \in Sr, n \in N \mid n>1 \qquad (6.14)$$

$$Tstr^{s}_{sr,n} \leq H \qquad \forall sr \in Sr, n \in N \mid n>1 \qquad (6.15)$$

$$Tstr^{f}_{sr,n} \leq H \qquad \forall sr \in Sr, n \in N \mid n>1 \qquad (6.16)$$

4.7.3. Nomenclature complémentaire

Indices :

sr : indice d'Etat Ressource,

Ensembles :

I : ensemble des tâches de la recette,
Sr : ensemble des Etats Ressources,
$Sr^{\aleph}$: ensemble des Etats Ressources avec des contraintes de type Zero-Wait,
I_{sr}^{c} : ensemble des tâches consommant l'Etat Ressource *sr,*
I_{sr}^{p} : ensemble des tâches produisant l'Etat Ressource *sr,*

Variables continues :

$Br_{i,n}$: quantité de matière stockée par la tâche fictive *s* au point d'évènement *n*,
$Sr_{s,n}$: quantité de matière dans l'Etat Ressource *sr* au point d'évènement *n*,
$Tstr^{s}{}_{sr,n}$: date à laquelle démarre la tâche fictive de stockage *s* au point d'évènement *n*,
$Tstr^{f}{}_{sr,n}$: date à laquelle se termine la tâche fictive de stockage *s* au point d'évènement *n*,

Paramètres :

C_{sr} : « capacité de stockage » maximale de l'Etat Ressource *sr*,
$\rho r_{i,sr}^{c}$: proportion d'Etat Ressource *sr* consommée par la tâche *i*,
$\rho r_{i,sr}^{p}$: proportion d'Etat Ressource *sr* produite par la tâche *i*,
$Sr0_{s}$: stock initial l'Etat Ressource *sr*,

4.7.4. Exemple : La séparation discontinue

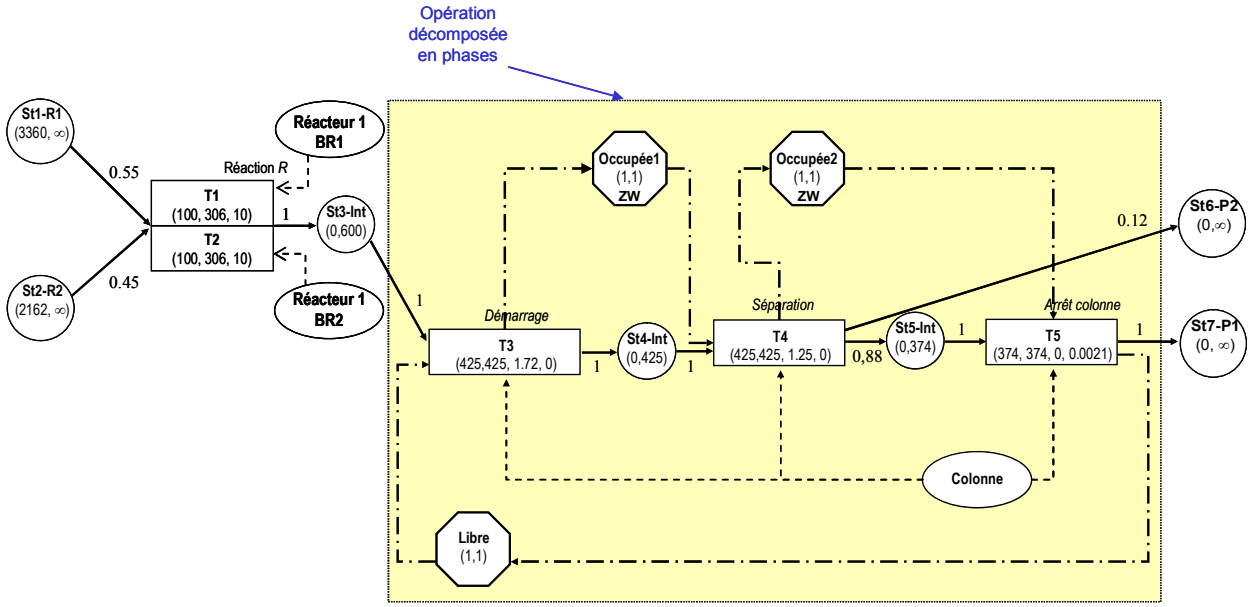


Figure 4.56 RTN exemple [Joglekar et al., 1985], avec hypothèse de séparation discontinue

Les durées utilisées pour initialiser le modèle d'ordonnancement sont tout d'abord estimées de manière approchée. Les paramètres sont les suivants :

Paramètres :

- Ordres de fabrication à réaliser : *600 kg* de *P2*.
- Pas de délais de nettoyage des appareils.
- **Durées opératoires variables** et dépendantes de la taille du lot et de l'appareil utilisé comprenant une partie fixe et une partie variable :

Les durées ont tout d'abord été estimées par la simulation d'un batch de taille moyenne, sans réaliser une analyse fine, puis une marge a été ajoutée pour la prise en compte de l'ensemble des tailles de lots admissibles.

n° Tâche	1	2	4	5	6
Ressource	BR1	BR2	Démarrage	Séparation	Vidange Colonne
df (h)	10	10	1.72	1.25	0
dv (h/kg)	0,06	0,06	0	0	0.0021

Tableau 4.18 Durées opératoires variables (température de réaction 365 K)

Solution :

Solution obtenue en 505,2 s (solution sous-optimale).

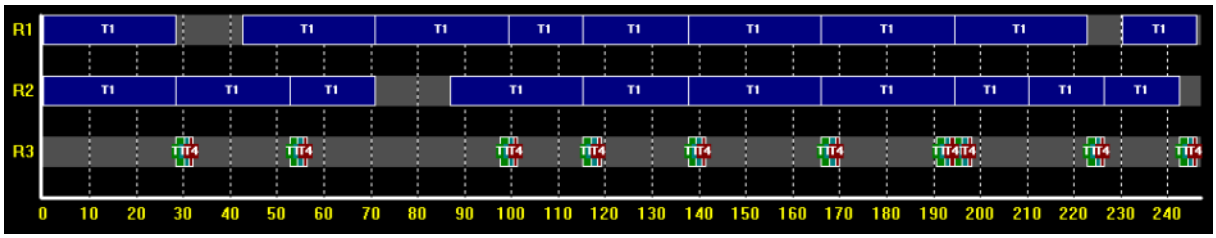


Figure 4.57 Diagramme de Gantt – Solution durées estimées

Tâche	t	Durée	Machine	Produit	Batch
T1	0.00	28.36	1	1	306.00
T1	42.56	28.36	1	1	306.00
T1	70.92	28.36	1	1	306.00
T1	99.28	16.00	1	1	100.00
T1	115.28	22.39	1	1	206.50
T1	137.67	28.36	1	1	306.00
T1	166.03	28.36	1	1	306.00
T1	194.39	28.36	1	1	306.00
T1	230.21	16.00	1	1	100.00
T1	0.00	28.36	2	1	306.00
T1	28.36	24.28	2	1	238.00
T1	52.64	18.28	2	1	138.00
T1	86.92	28.36	2	1	306.00
T1	115.28	22.39	2	1	206.50
T1	137.67	28.36	2	1	306.00
T1	166.03	28.36	2	1	306.00
T1	194.39	16.00	2	1	100.00
T1	210.39	16.00	2	1	100.00
T1	226.39	16.06	2	1	101.00
T2	28.36	1.72	3	2	425.00
T2	52.64	1.72	3	2	425.00
T2	97.56	1.72	3	2	425.00
T2	115.28	1.72	3	2	425.00
T2	137.67	1.72	3	2	425.00
T2	166.03	1.72	3	2	425.00
T2	190.63	1.72	3	2	425.00
T2	194.39	1.72	3	2	425.00
T2	222.75	1.72	3	2	425.00
T2	242.45	1.72	3	2	425.00
T3	30.08	1.25	3	3	425.00
T3	54.36	1.25	3	3	425.00

Tâche	t	Durée	Machine	Produit	Batch
T3	99.28	1.25	3	3	425.00
T3	117.00	1.25	3	3	425.00
T3	139.39	1.25	3	3	425.00
T3	167.75	1.25	3	3	425.00
T3	192.35	1.25	3	3	425.00
T3	196.11	1.25	3	3	425.00
T3	224.47	1.25	3	3	425.00
T3	244.17	1.25	3	3	425.00
T4	31.33	0.79	3	4	374.00
T4	55.61	0.79	3	4	374.00
T4	100.53	0.79	3	4	374.00
T4	118.25	0.79	3	4	374.00
T4	140.64	0.79	3	4	374.00
T4	169.00	0.79	3	4	374.00
T4	193.60	0.79	3	4	374.00
T4	197.36	0.79	3	4	374.00
T4	225.72	0.79	3	4	374.00
T4	245.42	0.79	3	4	374.00

Tableau 4.19 Tableau de résultats de l'ordonnancement – Solution durées estimées

4.8. EXTENSION DU MODÈLE À LA PRISE EN COMPTE DE TÂCHES DE PRODUCTION CONTINUES

4.8.1. Extension des éléments sémantiques

L'extension du modèle présentée vise à prendre en compte les tâches de production continues. Cette extension permet de gérer le lancement d'une tâche unitaire et son exécution cyclique pour simuler une tâche alimentée en continu.

nœuds ***Tâche Continue*** représentent les opérations de fabrication qui transforment la matière à partir d'un ou plusieurs états d'entrée et produisant un ou plusieurs états de sortie de manière continue

Figure 4.58 Tâche Continue en RTN

4.8.2. Extension du modèle

- **Contraintes de séquence des tâches continues :**

Enchaînement d'une même tâche sur le même appareil

$$Ts_{i,n} \leq Tf_{i,n-1} \qquad \forall i \in I^{C^\circ},\ \forall n \in N \mid n>1 \tag{6.17}$$

Couplées aux contraintes (3.10), les contraintes (6.17) imposent $Ts_{i,n} = Tf_{i,n-1}$ pour les tâches appartenant à l'ensemble des tâches continues I^{C°.

De plus, les tâches continues sont caractérisées par des durées et des tailles de lots constantes.

Extension formalisation graphique 3 : Tâches continues

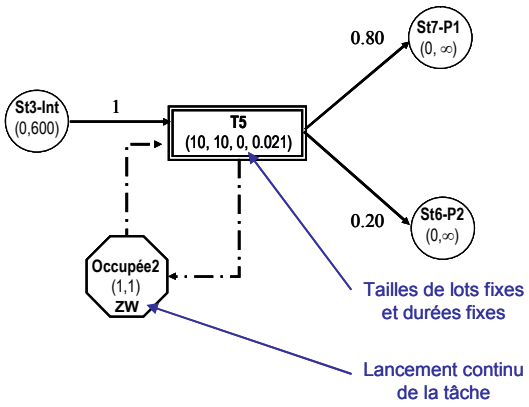


Figure 4.59 Tâches continues en RTN

4.8.3. Nomenclature complémentaire

Ensemble :

I^{C° : ensemble des tâches continues de la recette,

4.8.4. Exemple : La séparation continue

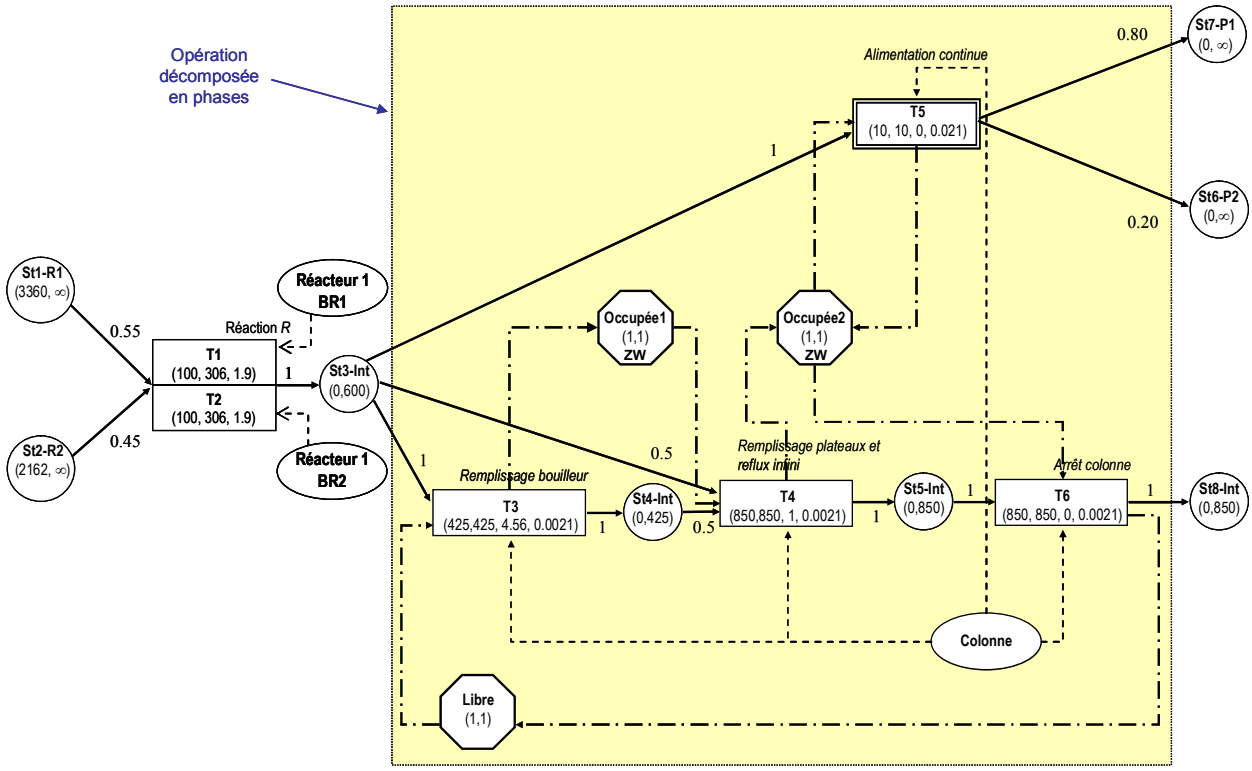


Figure 4.60 RTN exemple [Joglekar et al.*, 1985], avec prise en compte de la Gestion de l'état d'un appareil et des Tâches continues*

4.8.5. Simulation de l'exemple

4.8.5.1. Instance 1 : Gestion de l'état d'un appareil

Paramètres :

- Nombre de points d'évènement : $N = 79$ points.
- Ordres de fabrication à réaliser : *600 kg* de *P2*.
- Pas de délais de nettoyage des appareils.

- **Durées opératoires variables** et dépendantes de la taille du lot et de l'appareil utilisé comprenant une partie fixe et une partie variable :

n° Tâche	1	2	3	4	5	6
Ressource	**BR1**	**BR2**	**Remplissage bouilleur**	**Remplissage plateux + Reflux**	**Reflux fini**	**Vidange Colonne**
df (h)	1,9	1,9	4.56	1	0	0
dv (h/kg)	0,00535	0,00535	0,0021	0.0021	0.021	0.0021

Tableau 4.20 Durées opératoires variables

Solution : solution optimale obtenue en 42,3 secondes

Tâche	t	Durée	Machine	Produit	Batch	Tâche	t	Durée	Machine	Produit	Batch
T1	0.00	3.04	1	1	212.50	T4	16.31	0.21	3	4	10.00
T1	4.95	3.54	1	1	306.00	T4	16.52	0.21	3	4	10.00
T1	9.42	3.54	1	1	306.00	T4	16.73	0.21	3	4	10.00
T1	14.46	3.54	1	1	306.00	T4	16.94	0.21	3	4	10.00
T1	0.00	3.04	2	1	212.50	T4	17.15	0.21	3	4	10.00
T1	5.21	3.27	2	1	257.00	T4	17.36	0.21	3	4	10.00
T2	3.04	5.45	3	2	425.00	T4	17.57	0.21	3	4	10.00
T3	8.49	2.79	3	3	850.00	T4	17.78	0.21	3	4	10.00
T4	11.27	0.21	3	4	10.00	T4	17.99	0.21	3	4	10.00
T4	11.48	0.21	3	4	10.00	T4	18.20	0.21	3	4	10.00
T4	11.69	0.21	3	4	10.00	T4	18.41	0.21	3	4	10.00
T4	11.90	0.21	3	4	10.00	T4	18.62	0.21	3	4	10.00
T4	12.11	0.21	3	4	10.00	T4	18.83	0.21	3	4	10.00
T4	12.32	0.21	3	4	10.00	T4	19.04	0.21	3	4	10.00
T4	12.53	0.21	3	4	10.00	T4	19.25	0.21	3	4	10.00
T4	12.74	0.21	3	4	10.00	T4	19.46	0.21	3	4	10.00
T4	12.95	0.21	3	4	10.00	T4	19.67	0.21	3	4	10.00
T4	13.16	0.21	3	4	10.00	T4	19.88	0.21	3	4	10.00
T4	13.37	0.21	3	4	10.00	T4	20.09	0.21	3	4	10.00
T4	13.58	0.21	3	4	10.00	T4	20.30	0.21	3	4	10.00
T4	13.79	0.21	3	4	10.00	T4	20.51	0.21	3	4	10.00
T4	14.00	0.21	3	4	10.00	T4	20.72	0.21	3	4	10.00
T4	14.21	0.21	3	4	10.00	T4	20.93	0.21	3	4	10.00
T4	14.42	0.21	3	4	10.00	T4	21.14	0.21	3	4	10.00
T4	14.63	0.21	3	4	10.00	T4	21.35	0.21	3	4	10.00
T4	14.84	0.21	3	4	10.00	T4	21.56	0.21	3	4	10.00
T4	15.05	0.21	3	4	10.00	T4	21.77	0.21	3	4	10.00
T4	15.26	0.21	3	4	10.00	T4	21.98	0.21	3	4	10.00
T4	15.47	0.21	3	4	10.00	T4	22.19	0.21	3	4	10.00
T4	15.68	0.21	3	4	10.00	T4	22.40	0.21	3	4	10.00
T4	15.89	0.21	3	4	10.00	T4	22.61	0.21	3	4	10.00
T4	16.10	0.21	3	4	10.00	T4	22.82	0.21	3	4	10.00

Tâche	t	Durée	Machine	Produit	Batch
T4	23.03	0.21	3	4	10.00
T4	23.24	0.21	3	4	10.00
T4	23.45	0.21	3	4	10.00
T4	23.66	0.21	3	4	10.00
T4	23.87	0.21	3	4	10.00
T4	24.08	0.21	3	4	10.00
T4	24.29	0.21	3	4	10.00
T4	24.50	0.21	3	4	10.00
T4	24.71	0.21	3	4	10.00
T4	24.92	0.21	3	4	10.00
T4	25.13	0.21	3	4	10.00
T4	25.34	0.21	3	4	10.00
T4	25.55	0.21	3	4	10.00
T4	25.76	0.21	3	4	10.00
T4	25.97	0.21	3	4	10.00
T4	26.18	0.21	3	4	10.00
T4	26.39	0.21	3	4	10.00
T4	26.60	0.21	3	4	10.00
T4	26.81	0.21	3	4	10.00
T5	27.02	1.78	3	5	850.00

Tableau 4.21 Tableau de résultats de l'ordonnancement de l'instance 1

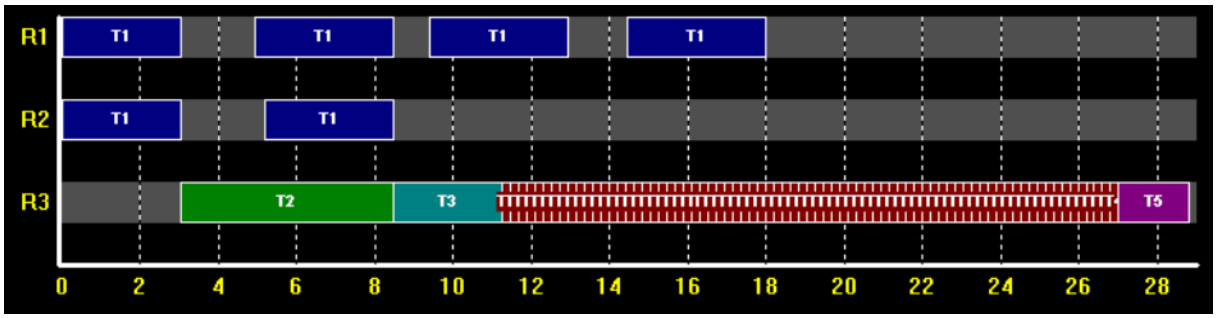


Figure 4.61 Diagramme de Gantt de l'instance 1 (durées en heures)

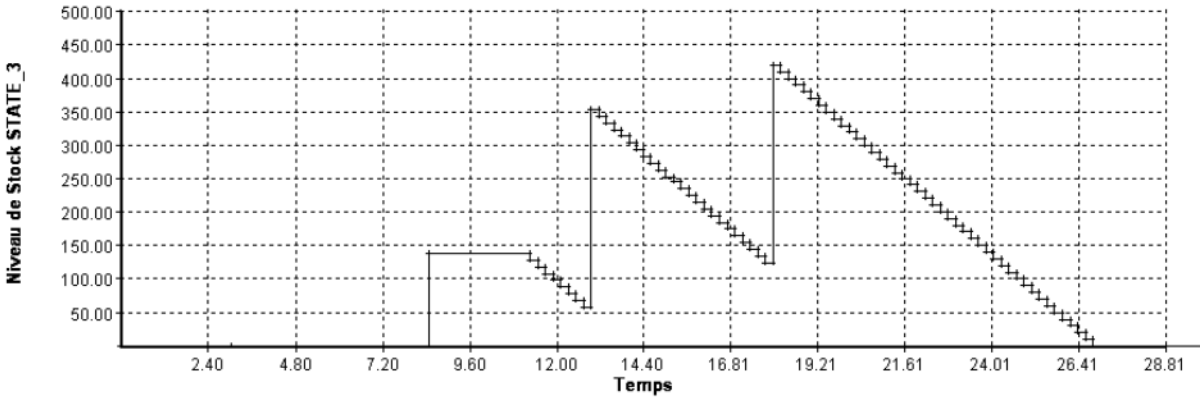


Figure 4.62 Evolution du niveau de stock de l'état 3 (en kg) en fonction du temps (en h) pour l'instance 1

4.9. EXEMPLE DU MODÈLE PRENANT EN COMPTE L'ENSEMBLE DES EXTENSIONS

Paramètres :

- Nombre de points d'évènement : *N = 79* points.
- Ordres de fabrication à réaliser : *600 kg* de *P2*.
- Pas de délais de nettoyage des appareils.

- **Durées opératoires variables** et dépendantes de la taille du lot et de l'appareil utilisé comprenant une partie fixe, une partie variable et une partie continue par morceaux

n° Tâche	1	2	3	4	5	6
Ressource	**BR1**	**BR2**	**Remplissage bouilleur**	**Remplissage plateux + Reflux**	**Reflux fini**	**Vidange Colonne**
df (h)	1,9	1,9	4.56	1	0	0
dv (h/kg)	0,00535	0,00535	0,0021	0.0021	0.021	0.0021

Tableau 4.22 Durées opératoires variables

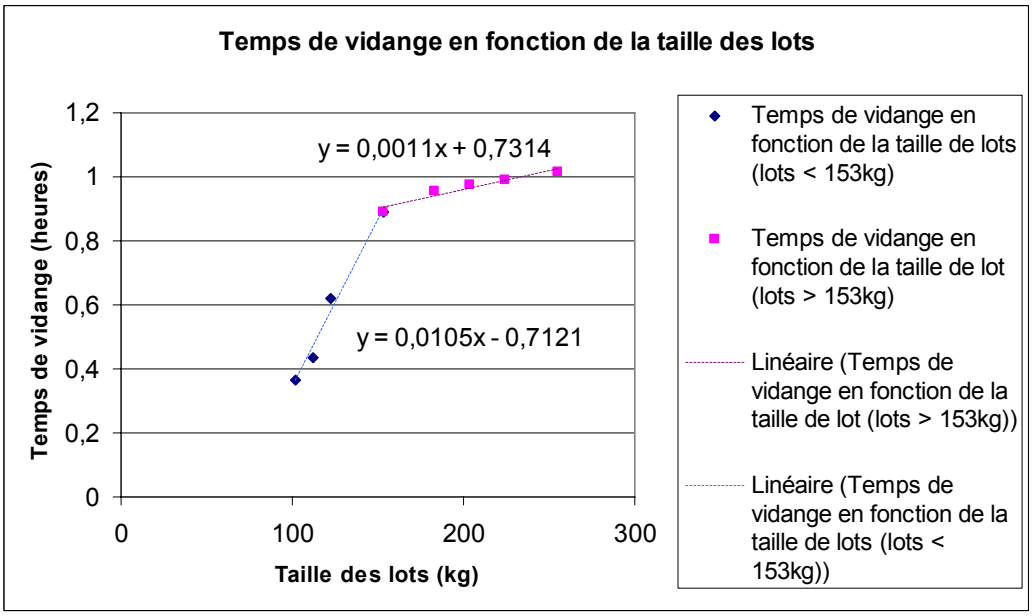


Figure 4.63 Durée vidange (en h) en fonction de la taille des lots (kg)

Solution : solution optimale obtenue en 39,6 secondes

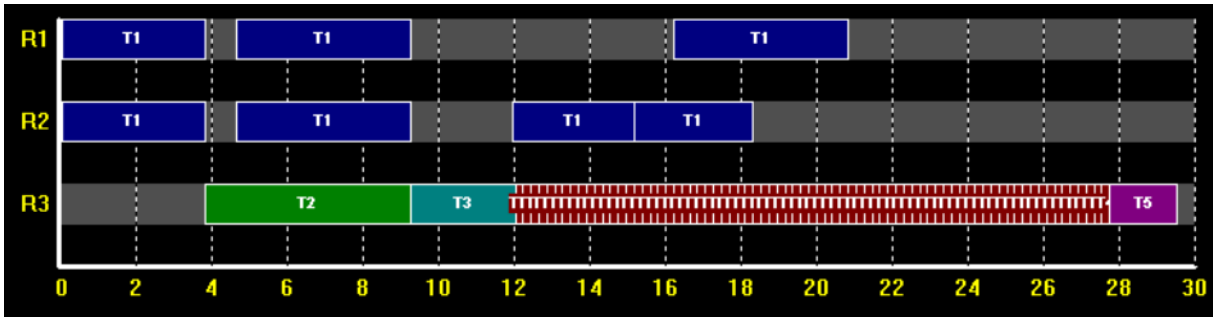


Figure 4.64 Diagramme de Gantt de l'instance 2 (durées en heures)

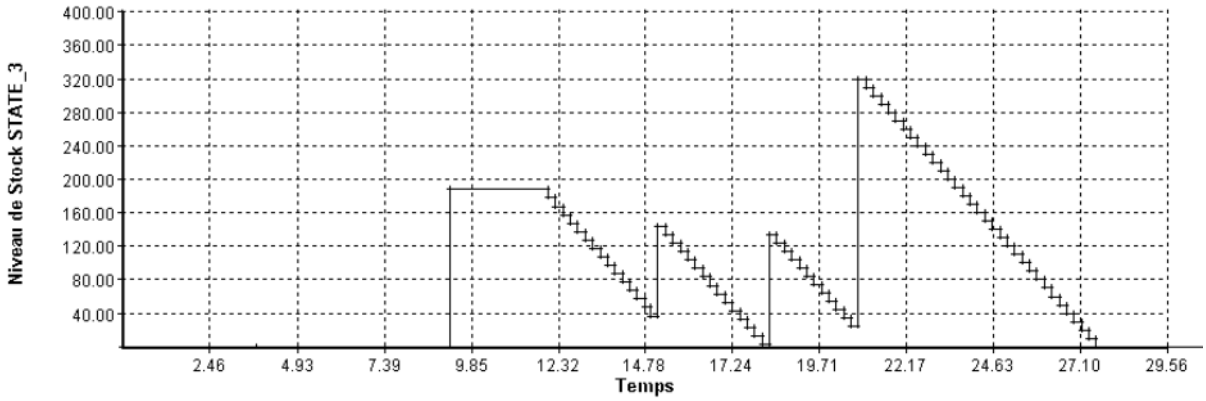


Figure 4.65 Evolution du niveau de stock de l'état 3 (en kg) en fonction du temps (en h) pour l'instance 2

Tâche	t	Durée	Machine	Produit	Batch	Tâche	t	Durée	Machine	Produit	Batch
T1	0.00	3.79	1	1	212.50	T4	17.06	0.21	3	4	10.00
T1	4.62	4.62	1	1	306.00	T4	17.27	0.21	3	4	10.00
T1	16.23	4.62	1	1	306.00	T4	17.48	0.21	3	4	10.00
T1	0.00	3.79	2	1	212.50	T4	17.69	0.21	3	4	10.00
T1	4.62	4.62	2	1	306.00	T4	17.90	0.21	3	4	10.00
T1	11.97	3.21	2	1	116.22	T4	18.11	0.21	3	4	10.00
T1	15.17	3.15	2	1	140.78	T4	18.32	0.21	3	4	10.00
T2	3.79	5.45	3	2	425.00	T4	18.53	0.21	3	4	10.00
T3	9.24	2.79	3	3	850.00	T4	18.74	0.21	3	4	10.00
T4	12.02	0.21	3	4	10.00	T4	18.95	0.21	3	4	10.00
T4	12.23	0.21	3	4	10.00	T4	19.16	0.21	3	4	10.00
T4	12.44	0.21	3	4	10.00	T4	19.37	0.21	3	4	10.00
T4	12.65	0.21	3	4	10.00	T4	19.58	0.21	3	4	10.00
T4	12.86	0.21	3	4	10.00	T4	19.79	0.21	3	4	10.00
T4	13.07	0.21	3	4	10.00	T4	20.00	0.21	3	4	10.00
T4	13.28	0.21	3	4	10.00	T4	20.21	0.21	3	4	10.00
T4	13.49	0.21	3	4	10.00	T4	20.42	0.21	3	4	10.00
T4	13.70	0.21	3	4	10.00	T4	20.63	0.21	3	4	10.00
T4	13.91	0.21	3	4	10.00	T4	20.84	0.21	3	4	10.00
T4	14.12	0.21	3	4	10.00	T4	21.05	0.21	3	4	10.00
T4	14.33	0.21	3	4	10.00	T4	21.26	0.21	3	4	10.00
T4	14.54	0.21	3	4	10.00	T4	21.47	0.21	3	4	10.00
T4	14.75	0.21	3	4	10.00	T4	21.68	0.21	3	4	10.00
T4	14.96	0.21	3	4	10.00	T4	21.89	0.21	3	4	10.00
T4	15.17	0.21	3	4	10.00	T4	22.10	0.21	3	4	10.00
T4	15.38	0.21	3	4	10.00	T4	22.31	0.21	3	4	10.00
T4	15.59	0.21	3	4	10.00	T4	22.52	0.21	3	4	10.00
T4	15.80	0.21	3	4	10.00	T4	22.73	0.21	3	4	10.00
T4	16.01	0.21	3	4	10.00	T4	22.94	0.21	3	4	10.00
T4	16.22	0.21	3	4	10.00	T4	23.15	0.21	3	4	10.00
T4	16.43	0.21	3	4	10.00	T4	23.36	0.21	3	4	10.00
T4	16.64	0.21	3	4	10.00	T4	23.57	0.21	3	4	10.00
T4	16.85	0.21	3	4	10.00	T4	23.78	0.21	3	4	10.00

Tâche	t	Durée	Machine	Produit	Batch
T4	23.78	0.21	3	4	10.00
T4	23.99	0.21	3	4	10.00
T4	24.20	0.21	3	4	10.00
T4	24.41	0.21	3	4	10.00
T4	24.62	0.21	3	4	10.00
T4	24.83	0.21	3	4	10.00
T4	25.04	0.21	3	4	10.00
T4	25.25	0.21	3	4	10.00
T4	25.46	0.21	3	4	10.00
T4	25.67	0.21	3	4	10.00
T4	25.88	0.21	3	4	10.00
T4	26.09	0.21	3	4	10.00
T4	26.30	0.21	3	4	10.00
T4	26.51	0.21	3	4	10.00
T4	26.72	0.21	3	4	10.00
T4	26.93	0.21	3	4	10.00
T4	27.14	0.21	3	4	10.00
T4	27.35	0.21	3	4	10.00
T4	27.56	0.21	3	4	10.00
T5	27.77	1.78	3	5	850.00

Tableau 4.23 Tableau de résultats de l'ordonnancement de l'instance 2

4.10. LE BILAN

Dans ce chapitre, les formalismes *STN* et *RTN* ont été présentés. Le modèle d'ordonnancement retenu pour le couplage ordonnancement / simulation a été présenté : *Unit-Specific Event – RTN Formulation*. Les limites de ce modèles concernant la gestion des stocks au niveau des états ont été illustrées et une formulation basée sur les travaux de [Janak *et al.,* 2004] a été choisie.

Des extensions ont été proposées pour la prise en compte de paramètres non linéaires, de tâches multimodales ou encore de tâches continues.

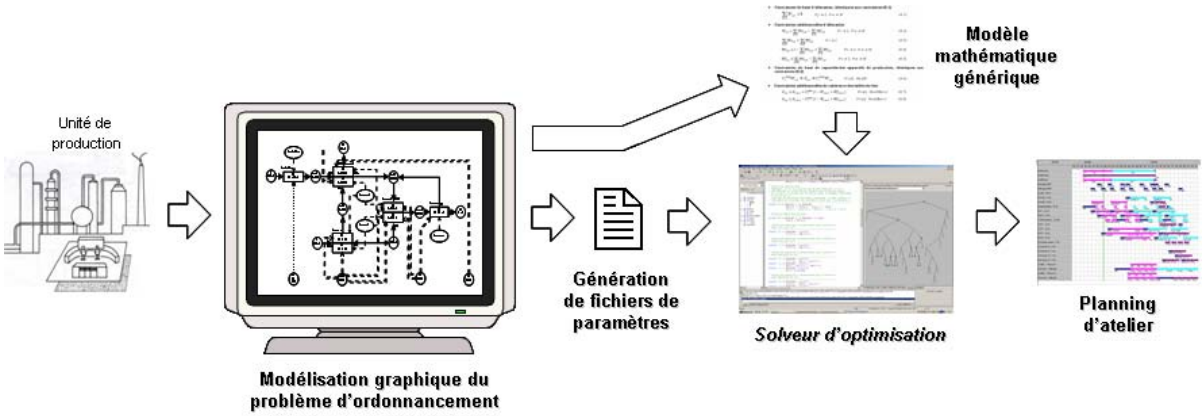


Figure 4.66 Intégration de la formalisation graphique et du modèle d'optimisation

Les points forts du modèle d'optimisation proposé s'inscrivent à plusieurs niveaux, tant sur le plan théorique que sur le plan implémentation logicielle :

- La résolution du modèle permet de connaître les quantités à produire, les dates de lancement des lots et les ressources utilisées, en tenant compte des limitations physiques des ressources.
- S'appuyant sur des méthodes de Programmation Linéaire en Variables Mixtes, le modèle d'optimisation propose une modélisation générique, tout en laissant la possibilité d'ajouter des contraintes particulières comme la prise en compte de recyclages, ou de l'aspect multimodal de certaines ressources (par exemple : lors des phases de démarrage de colonnes à distiller).
- Basé sur l'extension du formalisme RTN, l'ensemble des contraintes et des données du problème peut être saisi par l'intermédiaire d'une formalisation graphique adaptée.

CHAPITRE 5

CHAPITRE 5

INTÉGRATION DE LA CONDUITE DE LA SIMULATION SOUS *PRODHYS*

Résumé

Les concepts généraux de la plate-forme *PrODHyS* ont été présentés dans le chapitre 2. Les limites de la modélisation actuelle de la recette dans le simulateur ont été démontrées (gestion des dates de lancement et des volumes des lots impossible).

Dans ce cadre, le nouveau module *ProSched* et en particulier, le package *Scheduling* qui intègre les développements liés à la conduite de la simulation ont été développés. Ces derniers visent à formaliser la recette de *PrODHyS* afin de permettre l'identification d'un « centre de décision » au niveau du simulateur *PrODHyS*.

Ces développements fournissent un ensemble de classes permettant une *description hiérarchique rigoureuse de la recette* et d'autre part, l'intégration d'un *jeton de type Task* permettant de stocker et de propager au sein de la recette les informations relatives à la réalisation d'un lot (tâche, ressource nécessaire, date de lancement, paramètres opératoires, état de la tâche, etc.)

5.1. INTRODUCTION

Une nouvelle couche logicielle a été ajoutée à la structure de la plate-forme *PrODHyS*. Cette couche correspond à un ensemble de classes dédiées à la conduite des procédés. La couche conduite s'appuie sur les deux couches « modélisation » et « simulation » et fournit un ensemble d'entités permettant des études orientées sur la conduite des procédés. Cette couche comprend :

- le module *PrODHySAEM (**Pr**ocess **O**bject **Dy**namic **Hy**brid **S**imulator for **A**bnormal **E**vent **M**anagement)*, qui contient un ensemble de classes dédiées à l'étude des défaillances et des stratégies de surveillance des procédés (Olivier-Maget, 2007)
- le nouveau module *ProSched (**Pro**cess **Sched**uling)* dédié à la conduite de la simulation dynamique. Par ailleurs, un certain nombre d'outils sont inclus dans cette bibliothèque et seront détaillés dans le chapitre 6.

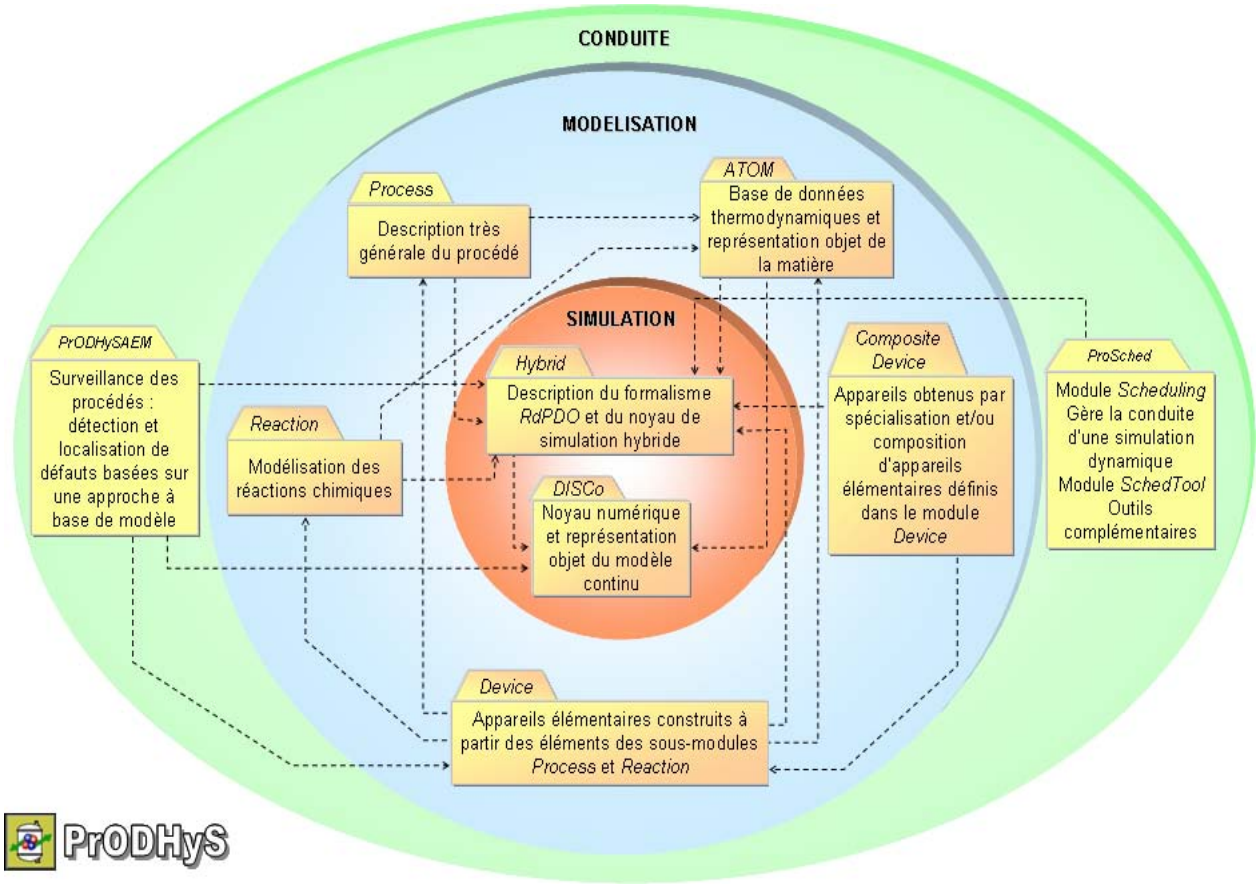


Figure 5.1 La structure de la plateforme PrODHyS

5.2. IMPLÉMENTATION DE LA BIBLIOTHÈQUE PROSCHED/SCHEDULING SOUS *PRODHYS*

Un nouveau package *ProSched* (Figure 5.2) a été ajouté à la plate-forme *PrODHyS* afin de modéliser de manière hiérarchisée la recette et de permettre la mise en place d'un « centre de décision » (cf. Figure 2.44) au niveau du simulateur.

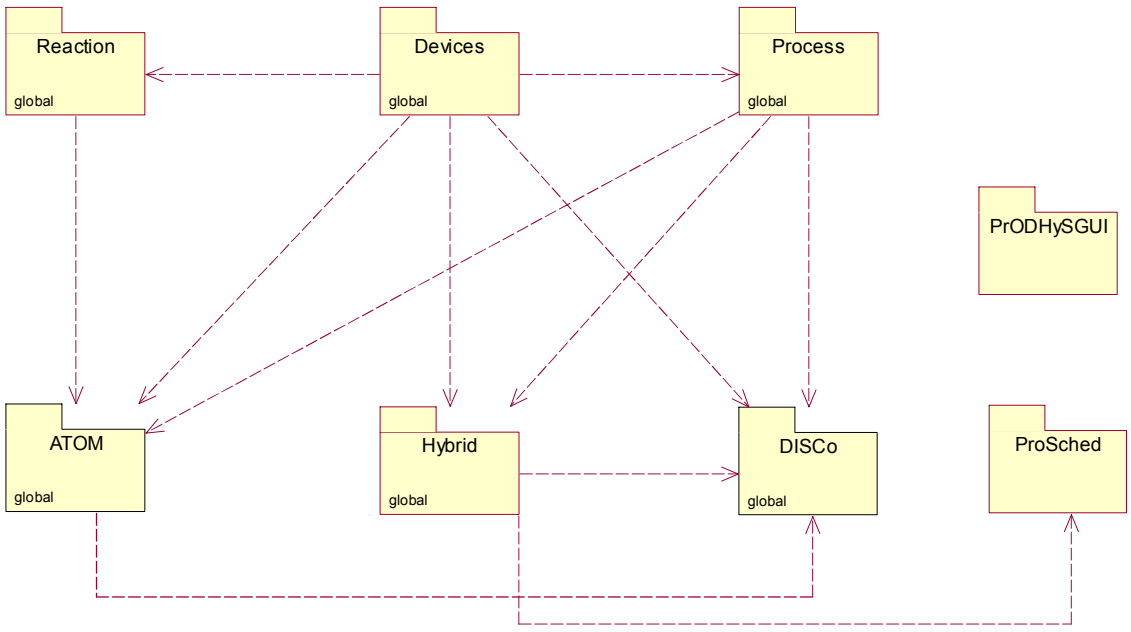


Figure 5.2 Intégration du package ProSched *sous* PrODHyS

Le package *ProSched* (Figure 5.3) se divise en deux parties :

- La première ***SchedTools*** contient le développement des outils permettant de faciliter la phase d'ordonnancement.
- La deuxième ***Scheduling*** intègre l'ajout des classes permettant une description hiérarchique rigoureuse de la recette et d'autre part, l'intégration de jetons de type *Task* au sein de la recette.

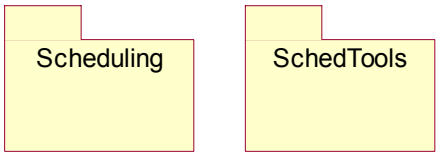


Figure 5.3 Description du package ProSched sous PrODHyS

5.3. HIÉRARCHISATION DE LA RECETTE SOUS *PRODHYS*

Afin d'établir une approche standard pour traiter la complexité du contrôle/commande des procédés batch, la norme *ISA/SP88* (www.isa.org) propose une modélisation hiérarchisée de la recette (cf. chapitre 1). Elle spécifie notamment la structuration du système de conduite ainsi que de la partie procédurale de la recette.

La partie procédurale est l'élément clé de la recette. La norme *ISA/SP88* préconise une hiérarchisation des informations selon cinq niveaux : *Instruction, Pas, Phase, Opération, Procédure* détaillés dans le chapitre 4.

L'objet de cette section est l'identification de cette hiérarchisation et sa mise en œuvre tant au niveau de la conception du simulateur qu'au niveau de la procédure de modélisation de la recette par l'utilisateur.

5.3.1. Identification des Opérations/Phases/Pas sous *PrODHyS*

Compte tenu de la complexité de la partie procédurale des recettes considérées, la décomposition hiérarchique proposée par la norme *ISA/SP88* a été adoptée afin de faciliter la modélisation de la procédure sous *PrODHyS* lors de la simulation d'un scénario de production.

La mise en œuvre de cette modélisation hiérarchisée de la recette est illustrée à travers un exemple de procédé inspiré de [Joglekar et al. 1985] et déjà décrit dans le chapitre 2. On rappelle ci-dessous les principales caractéristiques.

Pour rappel :

La recette de contrôle de cette opération est modélisée par la séquence opératoire montrée sur la Figure 5.5. Initialement, les cuves de stockages *ST1* et *ST2* sont remplies respectivement de produits *R1* et *R2*.

La séquence opératoire est composée de 5 étapes :

- La première étape consiste à ouvrir la vanne *V1*, jusqu'à atteindre une rétention *VLOT* dans *BR1*.
- Lorsque le transfert de *R1* est terminé, une phase de chauffe de *R1* est lancée dans *BR1*, cette phase s'arrête lorsque *R1* atteint 365K dans *BR1*.
- Une réaction contrôlée en température démarre alors dès l'ouverture de la vanne *V2*.
- L'obtention d'une concentration en produit *P* de 0.8 dans le réacteur déclenche la vidange de ce dernier par l'ouverture de la vanne *V6*.

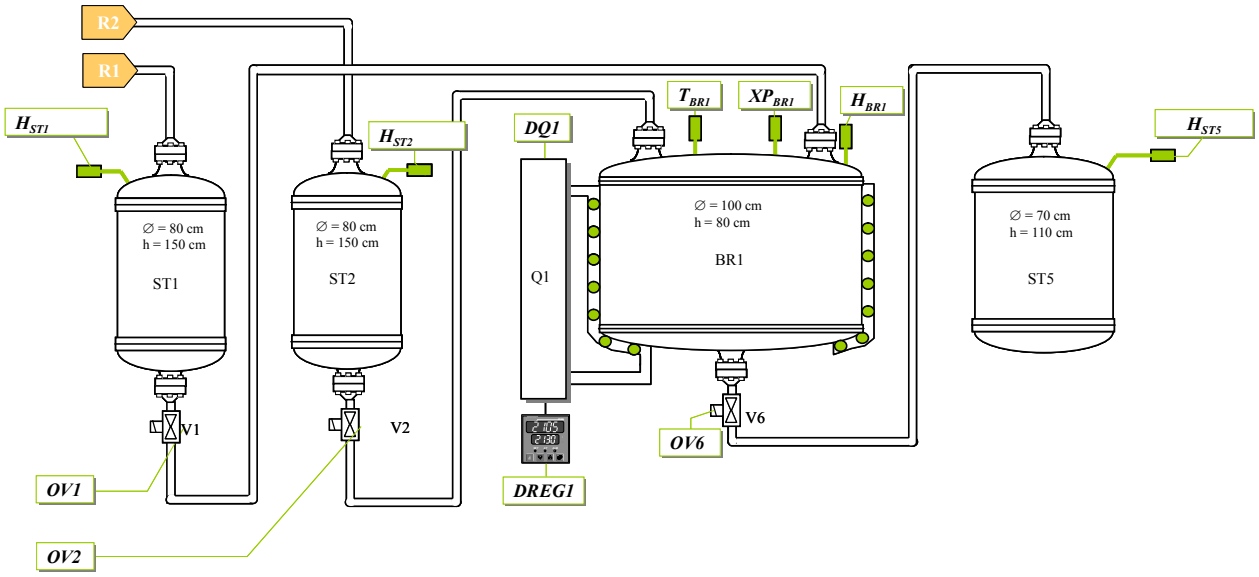


Figure 5.4 Opération de fabrication du produit P *dans* BR1

Une *opération* définit un ensemble de *phases* exécutées en séquence ou en parallèle dans une unité de production. Ce niveau de description de la recette rejoint la notion d'*opération unitaire* couramment rencontrée en génie des procédés. Il est alors possible d'identifier les différentes phases qui constituent la recette dans le réseau de Petri de la recette qui représente l'opération de réaction (Figure 5.5).

De même, une *phase* est une succession de *pas* permettant de réaliser une fonction élémentaire. Il est alors possible de les identifier au sein du *RdP*. Il s'agit des différentes places qui constituent une phase.

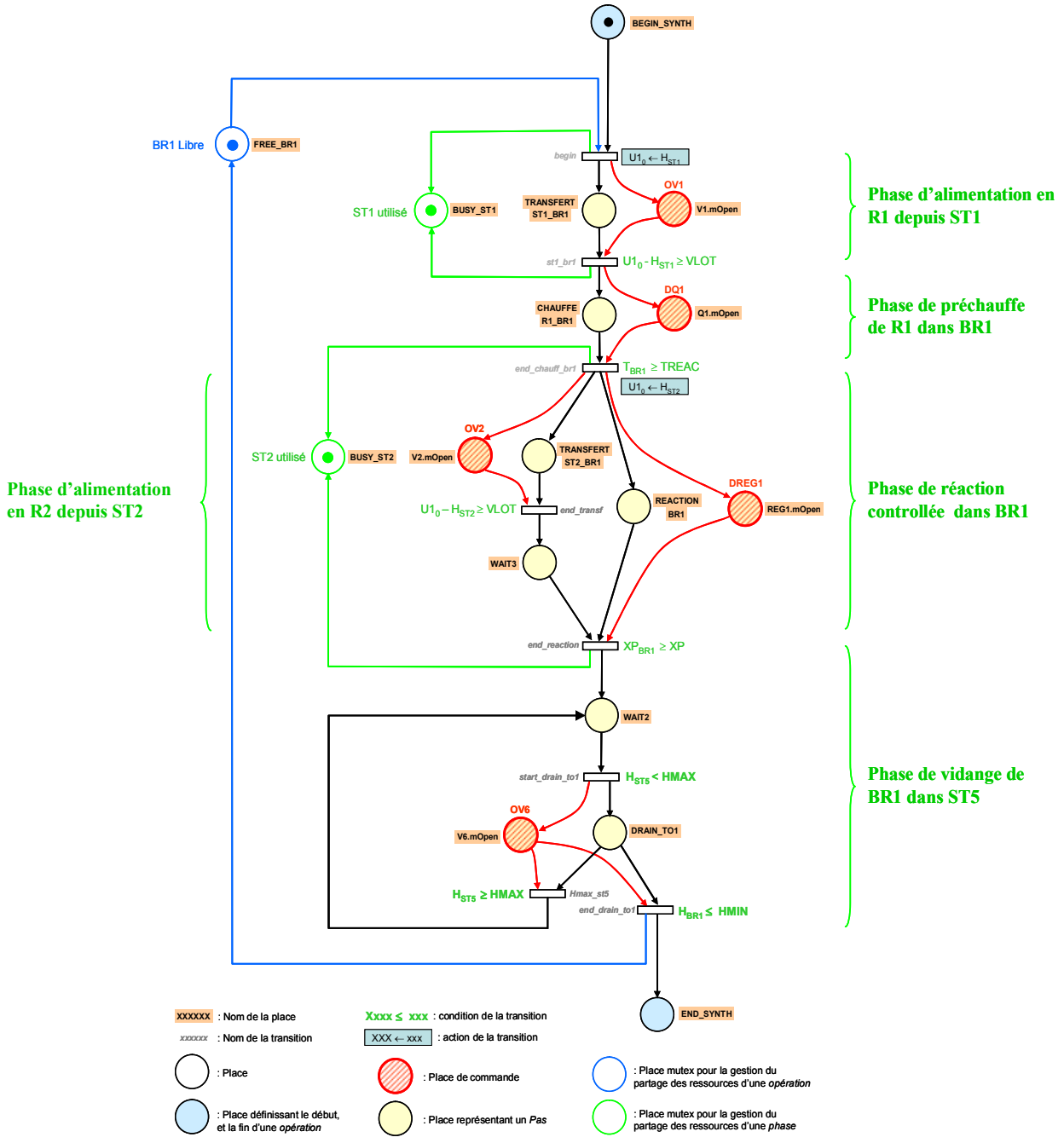


Figure 5.5 Identification des phases séquentielles et parallèles du réseau de Petri recette de l'exemple

La **Figure 5.6** présente les modèles *UML* associés à la procédure. On retrouve ainsi la classe `ProcedureRecipe` définie comme un ensemble d'opérations (classe `OperationRecipe`), elles-mêmes définies comme un ensemble de phases (classe `PhaseRecipe`).

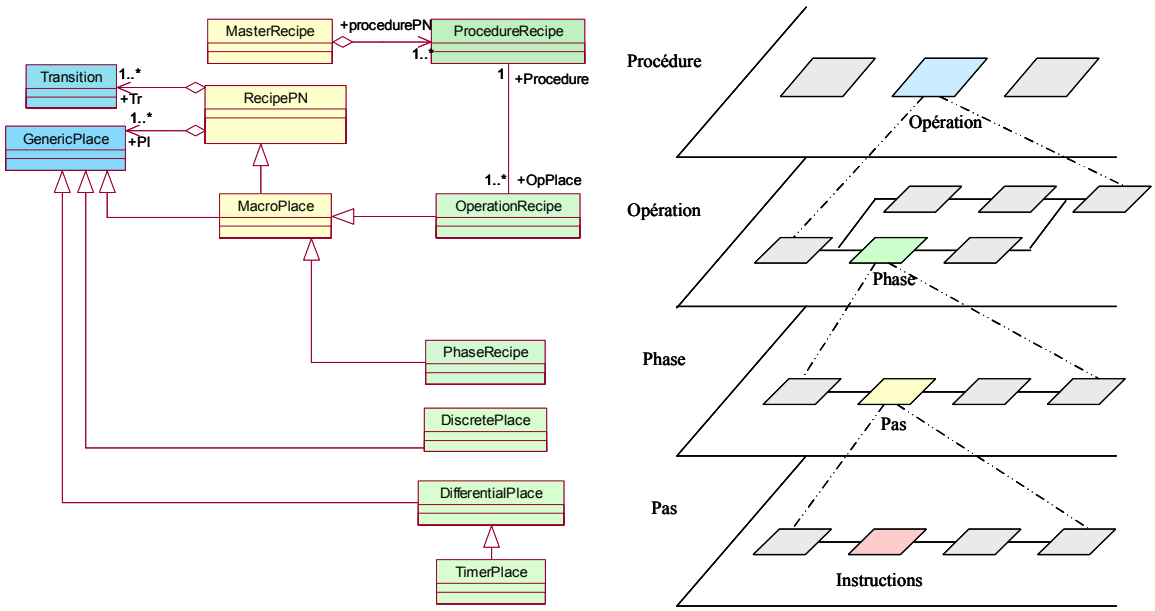


Figure 5.6 Hiérarchisation de la recette

5.4. NOTION DE MACRO-PLACE SOUS *PRODHYS*

5.4.1. Définition

La complexité de la partie procédurale de la recette de contrôle est mise en évidence sur la Figure 5.5 qui pourtant, est limitée à une opération unitaire et à un seul lot. Afin de simplifier cette modélisation, la notion de *macro-place* a été introduite. Elle permet de remplacer une séquence de places/transitions relatives à une opération ou à une phase. Cette hiérarchisation permet de maîtriser la complexité de la recette et facilite la mise en place du modèle de simulation par l'utilisation de séquences réutilisables.

Définition 5.1 : Macro-place

Une *macro-place* p_k est une place qui agrège une séquence de places/transitions.

La séquence agrégée dans une macro-place est constituée de :

- une place d'entrée EXX
- une place de sortie SXX
- une série de places/transitions intermédiaires, qui peuvent contenir elles-mêmes des macro-places, ce qui offre un niveau de décomposition infini

---

Représentée par un cercle et deux barres horizontales, une *macro-place* permet donc de remplacer une séquence de places/transitions relative à une opération ou une phase.

La traduction de cette décomposition appliquée à l'opération de fabrication du produit *P* dans *BR1* est présentée sur la figure 5.7.

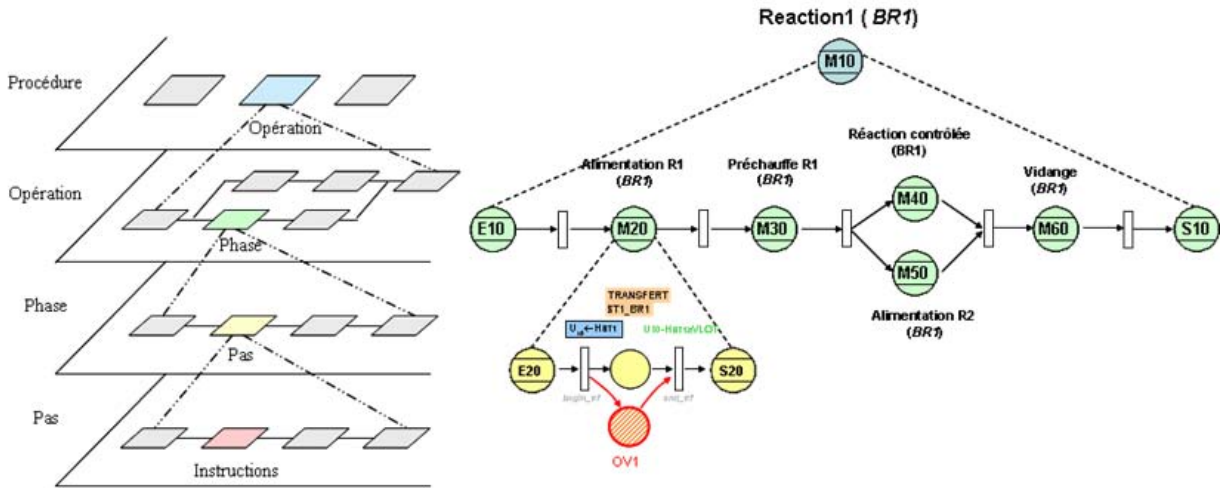


Figure 5.7 Décomposition du réseau de Petri recette du réacteur BR1 *en opération / phases / pas*

On peut décrire, sur le même principe, les autres phases de l'opération de réaction dans *BR1*.

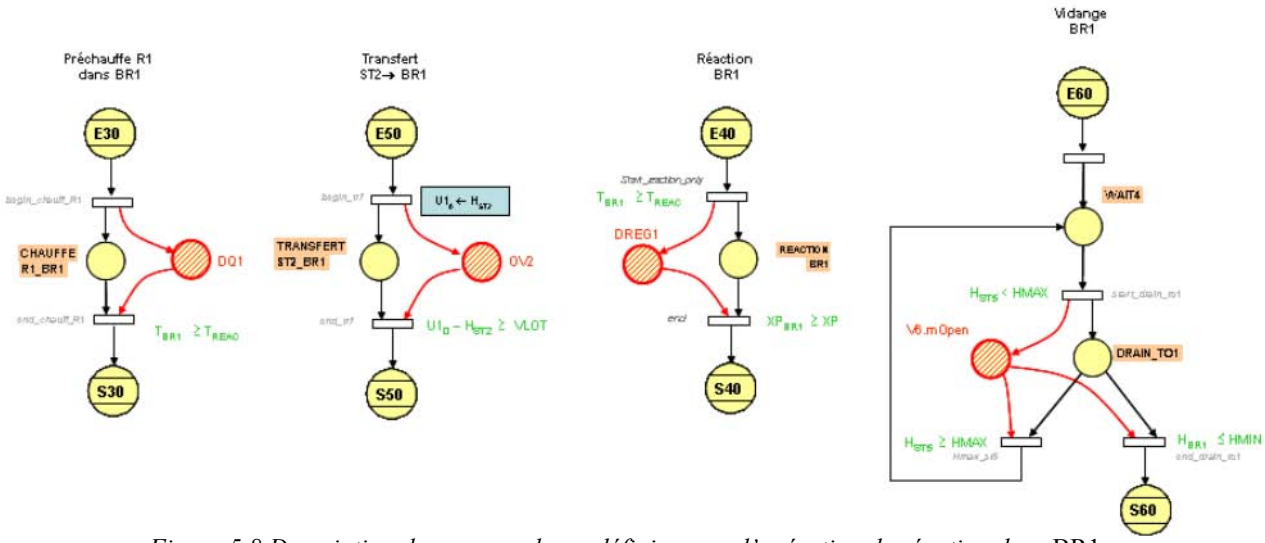


Figure 5.8 Description des macro-places définies pour l'opération de réaction dans BR1

Cette modélisation permet une modélisation simplifiée de la recette, par l'utilisation de composants réutilisables. Cependant, comme aucune information n'est portée par le jeton une macro-tâche ne peut être conçue de manière générique. En effet, les données opératoires et les conditions de franchissement des transitions doivent être codées directement dans les macro-tâches, ce qui limite leur réutilisation.

5.4.2. Hiérarchisation de la recette et approche *RTN*

Afin de formaliser un « centre de décision » sous *PrODHyS*, nous avons complété la hiérarchisation de la recette pour agréger les macro-tâches non plus seulement en *Opérations* (*Task*), mais aussi en *Appareils* (*Units*) afin de pouvoir gérer l'allocation des opérations.

Définition 5.2 : Règle de modélisation de la recette sous *PrODHyS*

Le principe de modélisation retenu est le formalisme *RTN* car l'une de ses règles de base est qu'une **tâche ne peut être exécutée que par une ressource unique**. A partir de ce principe, toutes les tâches réalisables par une ressource seront identifiées par des macro-places différentes. L'exécution de ces dernières sera dépendante de la libération de la ressource partagée, modélisée classiquement par une place contenant un jeton binaire.

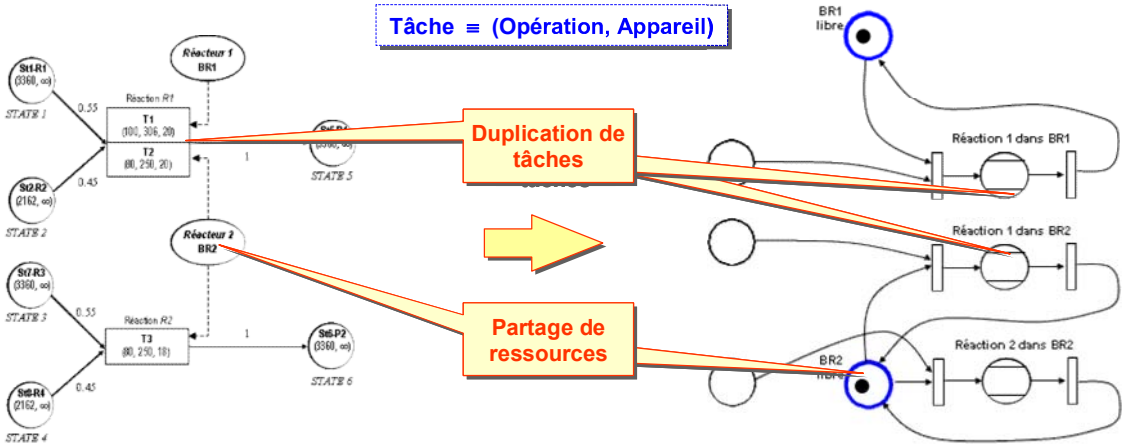


Figure 5.9 Règle de modélisation de la recette sous PrODHyS

Les *paramètres opératoires* sont actuellement portés par les *opérations* car aucune notion de lot n'est présente dans *PrODHyS* ; cette contrainte empêche le développement de macro-taches génériques qui simplifieraient encore la saisie de la recette.

Les jetons du réseau de Petri recette du simulateur sont des jetons binaires. Afin d'implémenter la gestion de la conduite, nous avons utilisé les avantages du formalisme *RdPDO* qui permet une intégration sur plusieurs niveaux des réseaux de Petri et des objets.

Dans ce contexte, pour gérer l'exécution d'une tâche (couple *opération/batch*), nous avons donc introduit un jeton objet de classe `TaskToken` (<T>) qui porte un objet de classe `Task`. Nous allons détailler ces deux objets par la suite.

5.4.3. Classe `TaskToken`

Il s'agit de l'intégration d'un nouveau type de jetons qui porte des objets de types Tâches (`Task`). Ces développements ont été réalisés au sein de la bibliothèque *ProSched* (Figure 5.10) qui gère l'ensemble des éléments relatifs à l'intégration de la conduite de la simulation sous *PrODHyS*.

Une classe `CreateTaskTokenAction` a aussi du être ajoutée afin de pouvoir gérer les cas de création de jetons lors de la simulation de la recette (Figure 5.10).

5.4.4. Classe `Task`

La classe `Task` (Figure 5.11), a été ajoutée par l'intermédiaire de la nouvelle bibliothèque *ProSched*. Un objet de type `task` est porteur des informations issues de l'optimisation qui permettent de gérer l'ordonnancement des tâches.

Ces informations sont :

- la date de début au plus tôt de la tâche,
- la taille du lot,
- le couple d'information *(task, unit),*
- et potentiellement, les données opératoires.

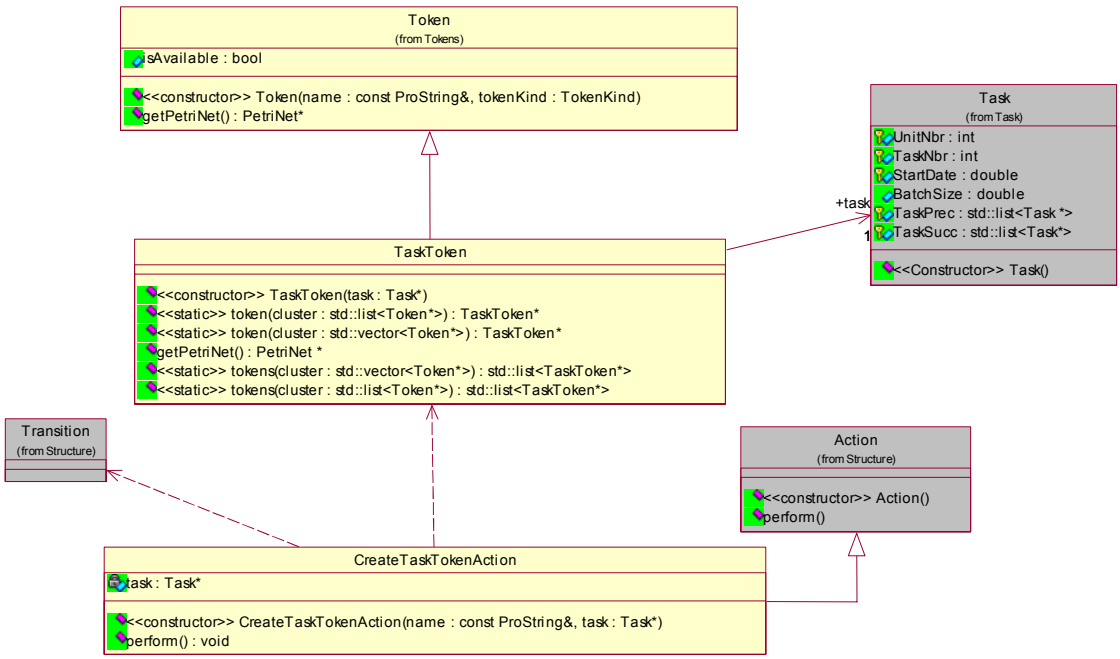


Figure 5.10 Diagramme de la classe `TaskToken` au sein de la bibliothèque ProSched/Scheduling

A ces informations ont été ajoutées, pour des développements futurs :

- la gestion d'un statut de la tâche au cours de la simulation
- la possibilité d'associer deux listes de tâches à une tâche identifiée : les prédécesseurs et les successeurs

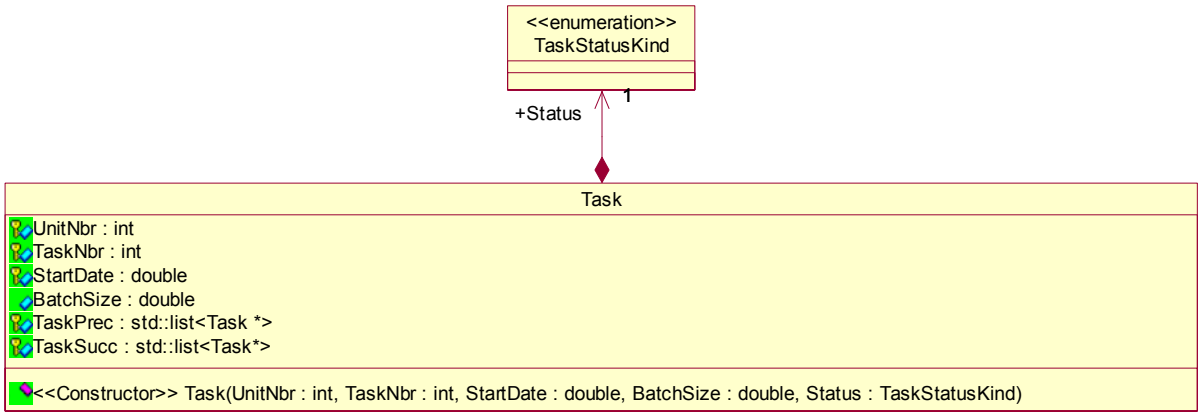


Figure 5.11 Diagramme de la classe Task au sein de la bibliothèque ProSched/Scheduling

5.5. ADAPTATION DE LA MÉTHODOLOGIE DE CONCEPTION SOUS *PRODHYS* : INTÉGRATION DU « CENTRE DE DÉCISION » À LA RECETTE

Afin d'intégrer la gestion des tâches sous *PrODHyS*, une adaptation de la méthodologie de conception d'un modèle de simulation (Figure 5.12) a été réalisée.

L'étape qui a été complétée correspond à la description de la *partie commande* sous *PrODHyS*. Une nouvelle structuration du réseau de Petri de la recette est proposée.

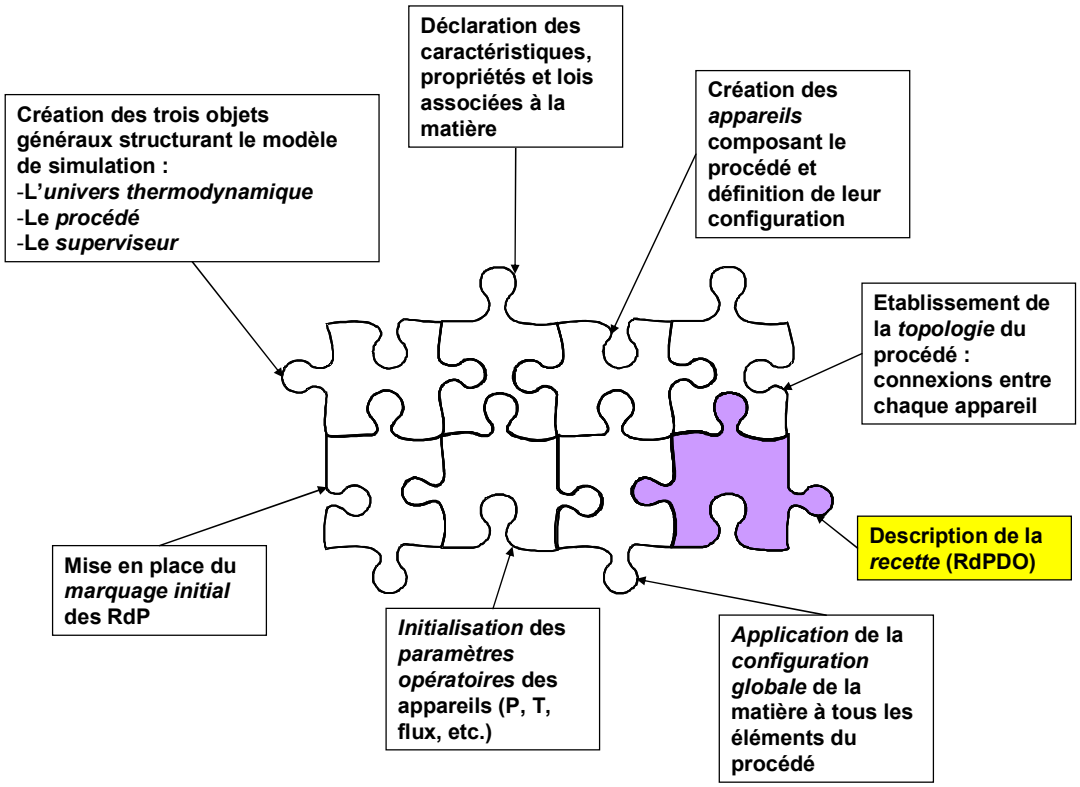


Figure 5.12 Procédure de conception d'un modèle de simulation sous PrODHyS

L'implémentation des objets présentés dans le paragraphe précédent (`TaskToken` et `Task`) nous a conduit à définir le « centre de décision d'un appareil » (cf. Figure 5.13).

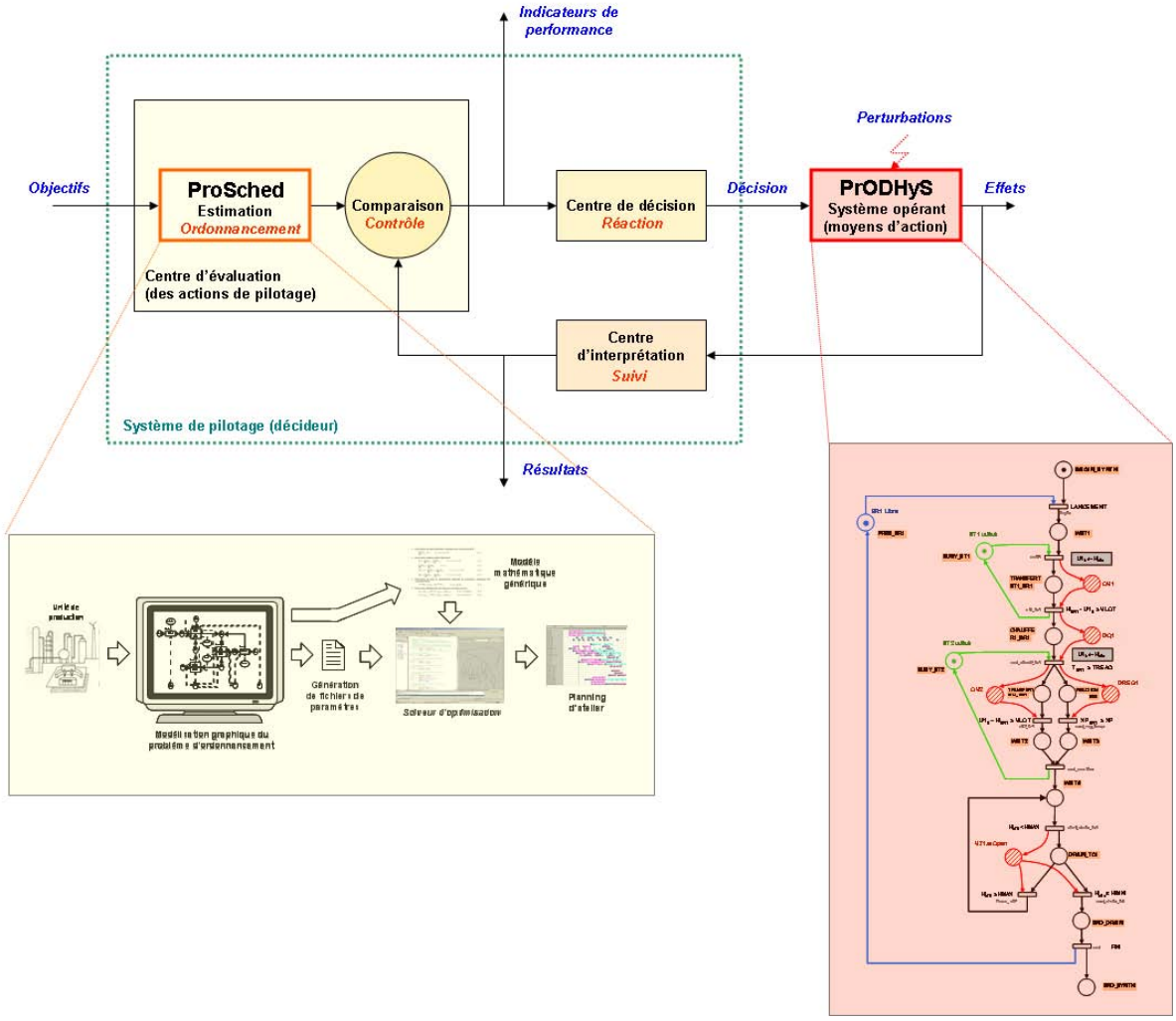


Figure 5.13 Système de pilotage décisionnel du simulateur

Définition 5.3 : « Centre de décision » d'un appareil

La structure associée à la gestion des tâches sous *PrODHyS*, représente le « centre de décision » de l'appareil.

Les éléments constitutifs du *« centre de décision »* d'un appareil sont de deux natures :

- **Temporelles** : par les gestions de *files d'attente* et de *lancement de lots*
- **Spatiales** : par les conditions de disponibilité de *matière* et de *ressources*

Corollaire **:** L'agrégation sur l'ensemble des appareils des « centres de décision » représente le centre de décision du modèle de simulation.

Les aspects temporels du *centre de décision* d'un appareil sont pris en compte pour chaque opération. La file d'attente est gérée par une place discrète tandis que le lancement des lots est modélisé par une séquence de tâches composée :

- d'une place discrète,
- d'une transition portant l'action de déclenchement de l'intégration dans la *place timer,*
- d'une place temporisée : chaque tâche est gérée par une *place timer* (`StartingDate` sur la Figure 5.14), ajoutée de manière à prendre en compte les dates de début (temps d'attente ← <T>.`StartingDate`).

Dans ce contexte, une *place discrète* est marquée avec un jeton <T> correspondant à la tâche qui doit être accomplie par cette opération, à la date <T>.`StartingDate`. Ce jeton est libéré lorsque la date de début est passée. Cette place *timer* est suivie d'une place dédiée à la gestion des files d'attente (queue sur la Figure 5.14) lorsque cela est nécessaire.

Les aspects spatiaux sont gérés par une place discrète pour modéliser la disponibilité de la ressource tandis que les conditions de disponibilité de la matière sont prises en compte au niveau des transitions.

Toutes les opérations qui ne partagent pas la même ressource peuvent potentiellement être réalisées en parallèle. A ce niveau, aucune relation de précédence n'apparaît explicitement dans le réseau de Petri recette.

Ainsi, une tâche ne peut être lancée qu'après s'être assuré de la présence d'un niveau suffisant de matière première (ou de produit intermédiaire). Pour cela, une condition (*cond* sur la Figure 5.14) est ajoutée sur la transition située avant la macro-place représentant l'opération, de manière à vérifier la présence de matière en quantité suffisante.

De plus, une place associée à la disponibilité de chaque ressource a été ajoutée, cela empêche le lancement d'une tâche alors que la ressource est occupée.

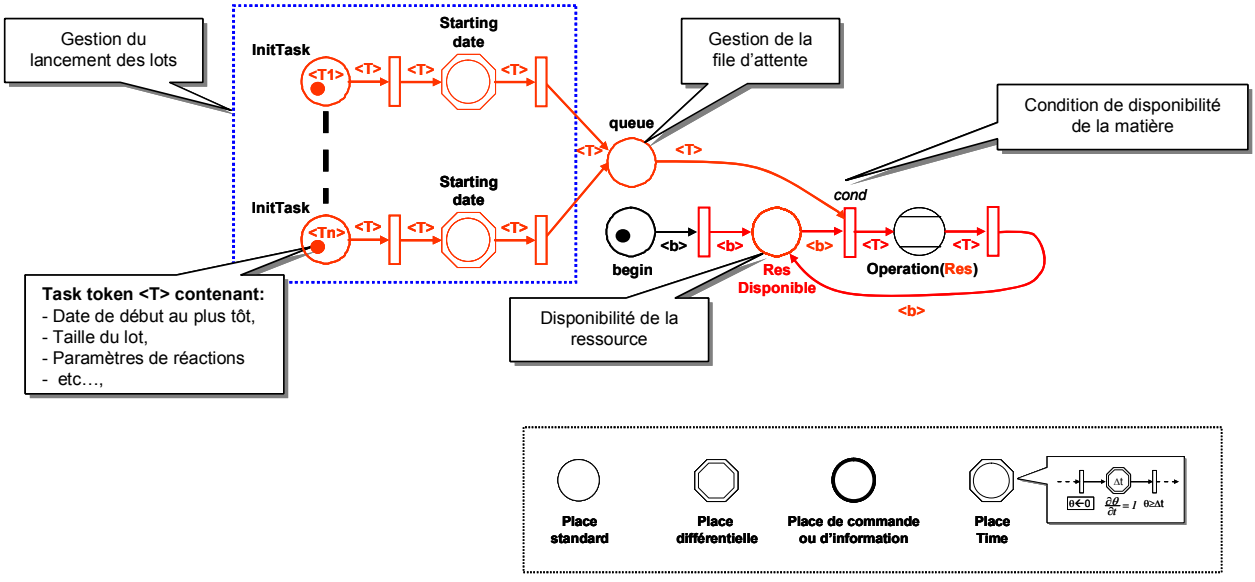


Figure 5.14 Structure générale associée à la gestion des tâches sous PrODHyS

Définition 5.4 : Places de pilotage

L'ensemble des places de la recette permettant la prise en compte des contraintes *Temporelles* et *Spatiales* sous *PrODHyS* sont appelées *Places de Pilotage*.

Définition 5.5 : Macro-places de pilotage

Une *macro-place de pilotage* p_k est une place qui agrège une séquence de places/transitions de pilotage.

Le concept de *macro-place de pilotage* peut être illustré au travers de la gestion temporelle d'une opération. Pour cela, il est important de noter que la séquence d'opérations relative au lancement des lots est toujours identique (cf. Figure 5.14). Dans ce contexte, une macro-place de pilotage de lancement des lots a été définie, elle regroupe l'ensemble des places et transitions nécessaires au lancement des jetons lots aux dates planifiées.

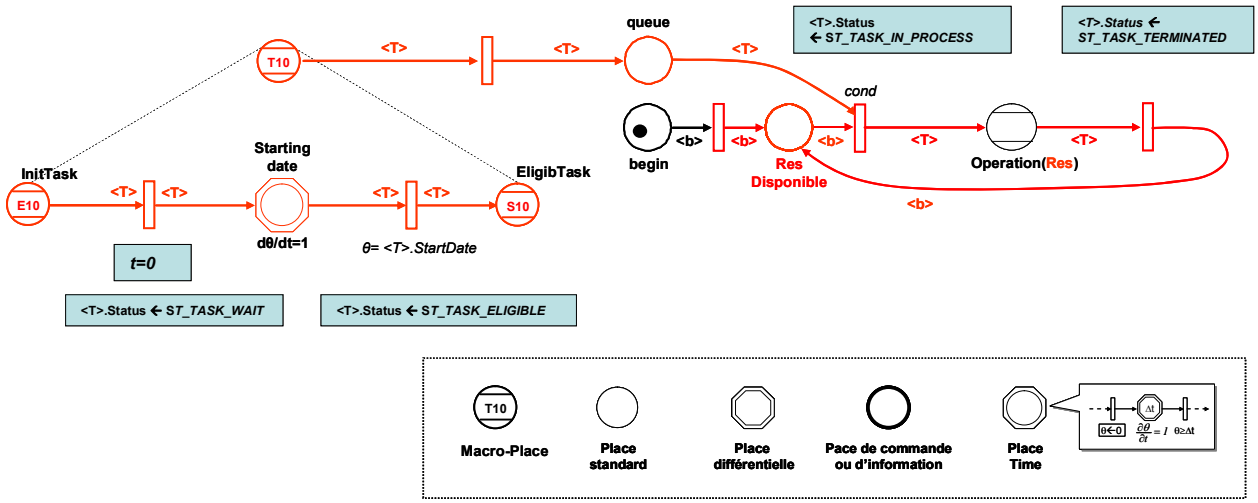


Figure 5.15 Structure détaillée associée à la gestion des tâches sous PrODHyS *et hiérarchisation de la gestion des lancements des lots* (macro-place de pilotage)

5.6. LES PREMIÈRES BRIQUES D'UN « *CENTRE D'INTERPRÉTATION* » SOUS *PRODHYS*

Les entrées du centre d'interprétation sont les effets du système opérant ou plutôt les données représentatives de ces effets (cf. Figure 2.32). Dans une procédure de diagnostic classique, on retrouve à ce niveau les phases d'acquisition (ou mesure) et de perception du signal qui consistent respectivement à fournir un ensemble d'observations tolérantes aux défauts du système de mesure et à fournir les indices ou paramètres indicateurs interprétables de la performance à l'origine des signaux observés. Mais la notion seule de mesure est réductrice car elle ne peut suffire à l'élaboration d'un diagnostic pertinent. Elle doit donc être complétée par une appréciation, une interprétation par rapport à une vision globale ou cadre de référence.

Dans ce contexte, les travaux sur la méthodologie de détection et localisation de défauts pour la *surveillance* des Systèmes Dynamiques Hybrides réalisés dans la thèse de Nelly Olivier-Maget [Olivier-Maget N., 2007] constituent une brique du *« Centre d'interprétation »* de *PrODHyS*. Cependant, ces

travaux centrés sur la supervision visent à détecter les dérives du procédé par rapport à un fonctionnement nominal des appareils. Dans un contexte de pilotage, il est alors nécessaire de pouvoir faire le lien avec la recette générale afin d'identifier quel lot sera en défaut et quels impacts cette dérive peut avoir par rapport au plan de production.

Une variable permettant de suivre l'évolution du statut de chaque *Tâche* a été intégrée à *PrODHyS* (Figure 5.15), afin de pallier à ce problème. Cette variable *statut* associée aux tâches portées par les jetons `TaskToken` dans le cadre du *centre de décision »* constitue l'élément principal du *« centre d'interprétation »* du module de pilotage de *PrODHyS*.

Définition 5.6 : Macro-places de pilotage

Le suivi du statut d'une tâche *(ST_TASK_IN_PROCESS, ST_TASK_TERMINATE, ST_TASK_ELIGIBLE, ST_TASK_WAIT)*, constitue l'élément principal du « Centre d'interprétation » du module de pilotage de *PrODHyS*.

5.7. GÉNÉRALISATION DU CONCEPT DE MACRO-PLACE

Nous nous intéressons maintenant à l'opération de réaction de l'exemple de procédé de [Kondili et al., 1985], en prenant en compte les ressources *BR1* et *BR2*. Si l'on ne tient pas compte du recyclage, l'*opération* de réaction est identique sur les deux réacteurs. Par conséquent, les macro-tâches (*phases*) décrites pour le *BR1* sont aussi valables pour *BR2*.

5.7.1. *RTN* de l'opération de réaction dans *BR1* et *BR2*

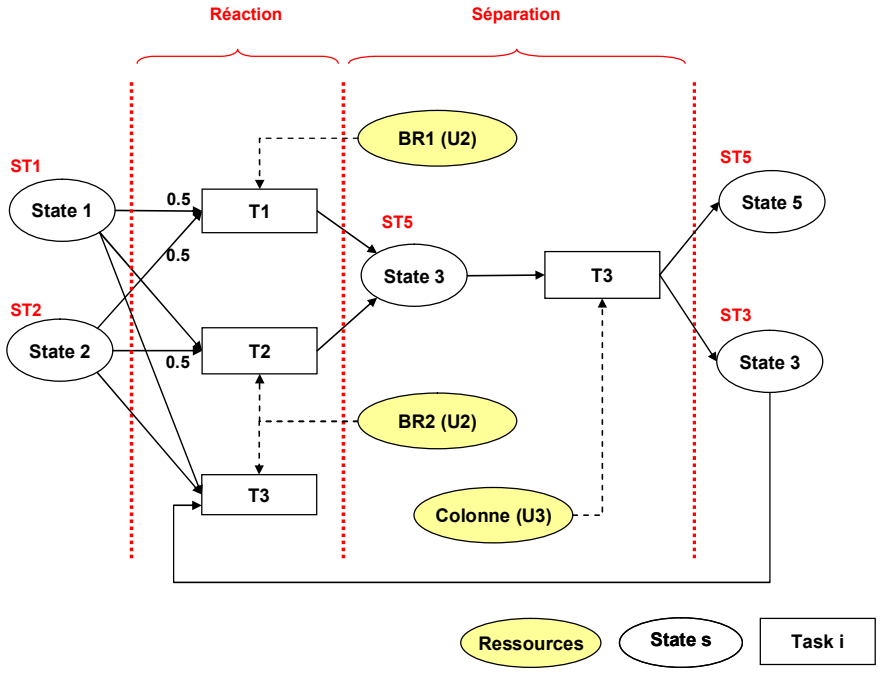


Figure 5.16 RTN de l'exemple inspiré de [Joglekar et al., 1985] avec phase de démarrage de la colonne

5.7.2. Macro-Places paramétrables

Le codage des phases des opérations peut être fastidieux et source d'erreur, surtout lorsque les enchaînements de phases sont identiques tout en faisant intervenir des paramètres opératoires distincts (ex : place commande des vannes distinctes).

La mise en œuvre de jetons de type `TaskToken` permet de :

- sortir les paramètres opératoires de la macro-place
- gérer la liste des actionneurs nécessaires à la réalisation d'une tâche

Deux types d'objets sont donc ajoutés à la tâche (voir Figure 5.17) :

- un objet de type `Operation` qui permet de stocker l'ensemble des actionneurs nécessaires à la réalisation de la tâche (vannes, pompes…) qui seront pilotés dans les macro-places via des *places de commandes*.
- un objet de type `ProcessingParameter` qui contient les différents paramètres nécessaires à la réalisation d'une tâche

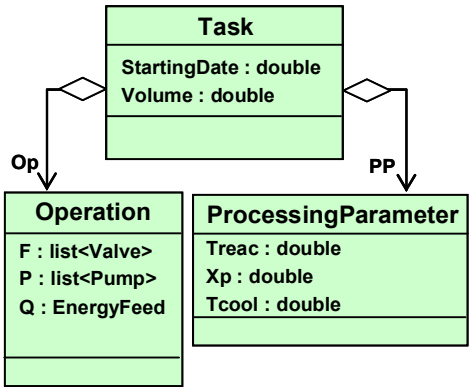


Figure 5.17 Gestion des paramètres opératoires et des actionneurs au sein d'une tâche

Dans ce contexte, il est alors possible de définir des *macro-places paramétrables* (Figure 5.18) dont les paramètres seront renseignés lors du passage d'un jeton de type `TaskToken` nécessaire à leur lancement.

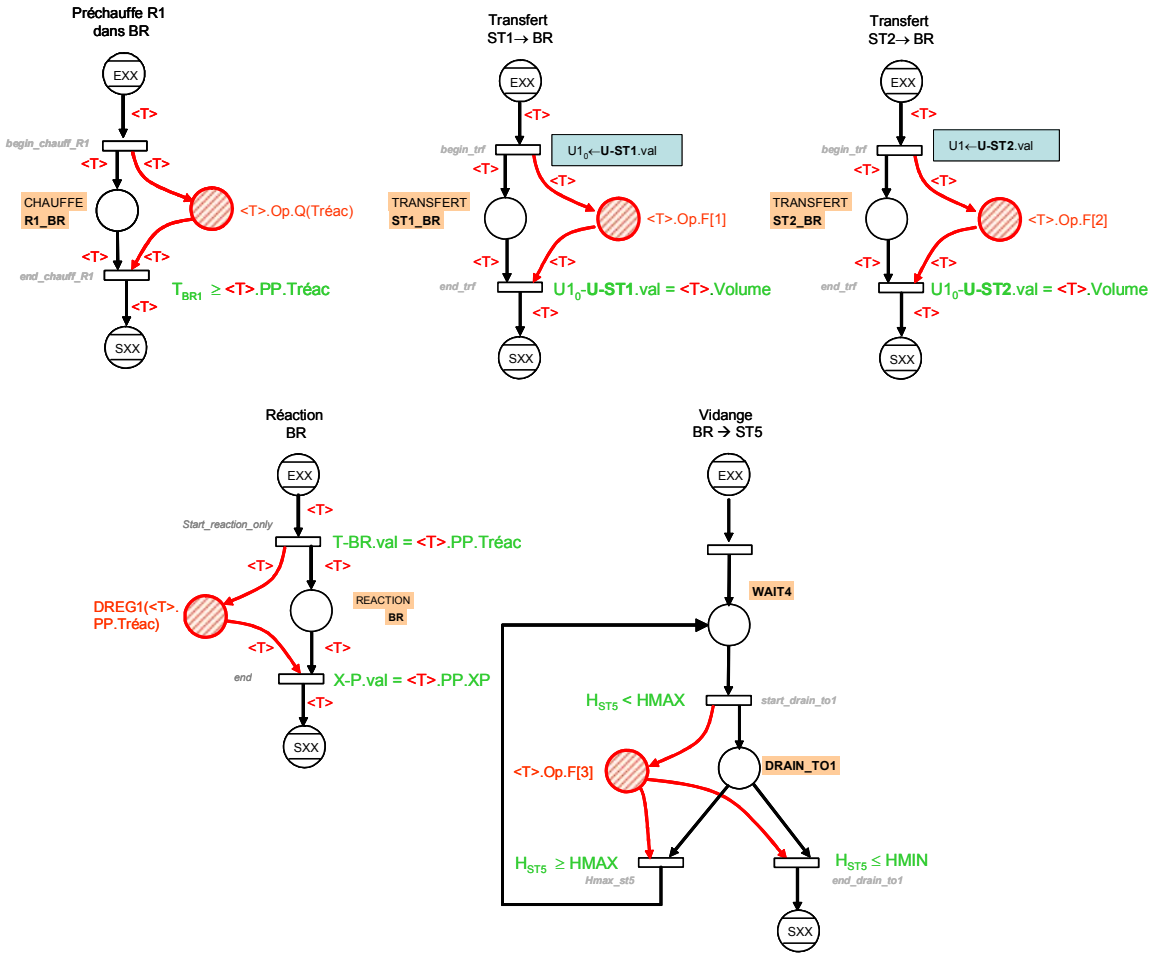


Figure 5.18 Description des macro-places paramétrables définies pour l'opération de réaction dans un réacteur BR

Définition 5.6 : Macro-places paramétrables

Une macro-place est dite paramétrable si l'ensemble de ses actions et paramètres opératoires est définis au moyen de paramètres formels. Ces paramètres sont transmis via un jeton de type `TaskToken`.

Quelques modifications ont du être réalisées dans les *RdP* des appareils afin de gérer leur évolution au travers d'un jeton de type `TaskToken` et non pas d'un jeton binaire. Une fois ces paramétrages réalisés, nous avons pu définir un ensemble de tâches paramétrables correspondant aux *phases* de l'*opération de réaction*.

5.8. CRÉATION DE LA RECETTE SOUS *PRODHYS* : LE NIVEAU PROCÉDURE

La macro-place associée à l'opération de fonctionnement de *BR2* en mode recyclage est représentée afin d'illustrer le concept de libération de ressources.

Conformément à la représentation *RTN*, les différentes tâches réalisées par une même ressource (*Unit*) sont représentées au sein du réseau de Petri recette.

Pour le réacteur *BR2*, le choix de fonctionnement en mode recyclage ou non est directement géré lors de la simulation par le suivi de quantité seuil au niveau des transitions qui précèdent les lancements des macro-places (Figure 5.19).

La structure du réseau de Petri de la *partie commande* est présenté Figure 5.19.

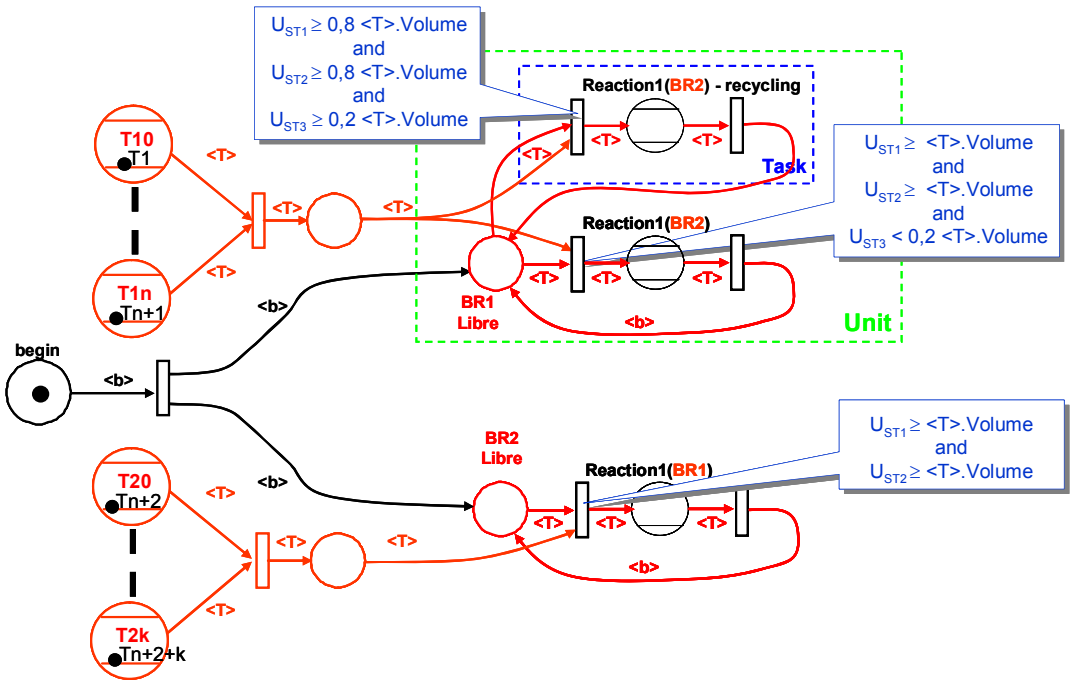


Figure 5.19 Réseau de Petri de la partie commande de l'exemple

La Figure 5.20, décrit un exemple succinct du diagramme de classe de l'objet `Task` et de l'instanciation de cette classe correspondant à la tâche qui démarre à l'instant t=0 sur le réacteur 1.

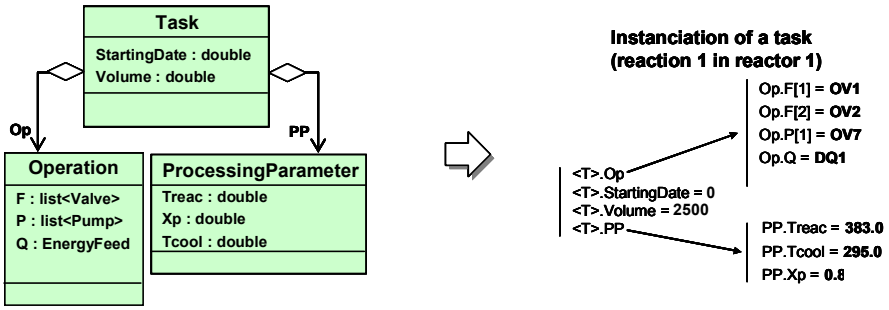


Figure 5.20 Exemple d'instanciation d'une tâche

5.9. BILAN

L'implémentation de la conduite de la simulation sous *PrODHyS* se situe à deux niveaux.

- Le premier est un *niveau structurel* car des modifications ont été apportées au simulateur, une bibliothèque *ProSched* a été ajoutée, avec la mise en place de jetons particuliers de types *tâches* en s'appuyant sur le formalisme *RdPDO* [Perret J., 2003].
- Le second est un *niveau conceptuel*, avec l'évolution de la *procédure de modélisation de la recette* sous *PrODHyS*. Dans ce cadre, un ensemble des places de pilotage a été ajouté à la recette, ainsi que l'identification des regroupements de macro-places en *tâches* et *appareils* (Figure 5.19).

Le *centre de décision* de *PrODHyS* a ainsi pu être défini et les premières briques d'un *Centre d'interprétation* posées. En appliquant les principes de modélisation de la partie commande détaillés dans le paragraphe précédent, les jetons `Tasks <T>` sont créés et sont en attente d'initialisation avec leurs dates de début, taille des lots, paramètres opératoires et listes d'actionneurs.

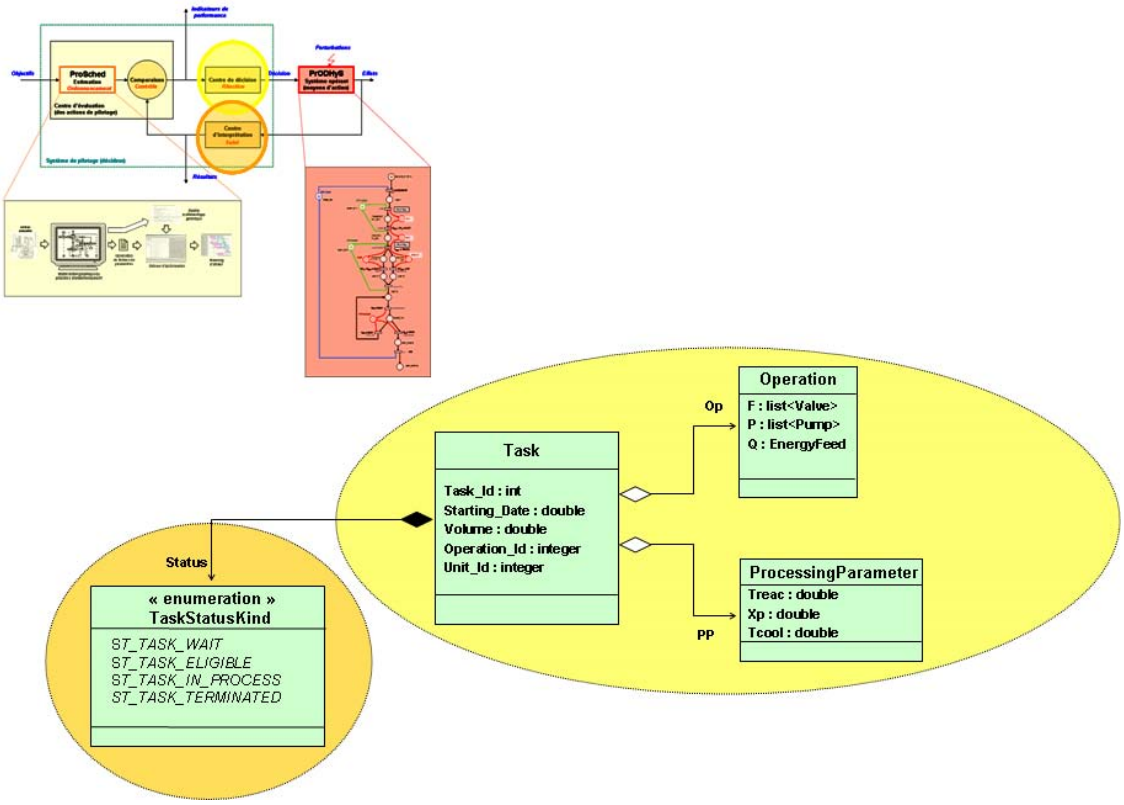


Figure 5.21 Intégration du système de pilotage de la simulation

A ce niveau, seul un module d'ordonnancement est capable de fournir les informations requises par le simulateur. Le modèle retenu basé sur la programmation linéaire en variables mixtes a été présenté au chapitre 4.

Dans le prochain chapitre nous décrivons la mise en œuvre d'une approche visant à coupler les développements liés à l'intégration de la conduite de la simulation et le module d'ordonnancement.

CHAPITRE 6

CHAPITRE 6

MISE EN ŒUVRE DU COUPLAGE OPTIMISATION / SIMULATION

Résumé

Dans ce chapitre, nous proposons les principes d'une approche basée sur le couplage *Optimisation/Simulation* pour le pilotage des procédés discontinus. Nous présentons ensuite un exemple complet de couplage optimisation / simulation. Par ailleurs, un certain nombre d'outils inclus dans la bibliothèque *ProSched* sont présentés dans ce chapitre afin de faciliter ce couplage.

6.1. STRATÉGIE D'EXPLOITATION DU COUPLAGE ENTRE OPTIMISATION ET SIMULATION

Le chapitre 4 a permis de décrire un modèle d'optimisation basé sur une méthode de programmation linéaire en variables mixtes.
Plusieurs extensions ont été proposées pour :

- la prise en compte de spécificités propres aux ateliers du génie des Procédés,
- la représentation de variables continues par morceaux appliquée au transfert par gravité,
- la gestion multi-modale de certains appareils comme les colonnes à distiller,
- la représentation de tâches continues au travers de la séparation continue.

Le chapitre 5 a présenté l'intégration de la conduite de la simulation sous *PrODHyS* tant au niveau structurel que conceptuel ; un *centre de décision* a ainsi pu être défini, basé sur une approche hiérarchique de la recette.

Dans le cadre du chapitre 6, les modèles d'optimisation mis en œuvre ne comportent pas de fonctions continues par morceaux, qui pénalisent fortement les temps de résolution par l'augmentation du nombre de variables. Cependant, le couplage avec le simulateur permet de réduire l'impact de telles simplifications sur la solution finale grâce au suivi des dérives potentielles en simulation dynamique. La stratégie de couplage adoptée, débute par un premier calcul d'optimisation avec une estimation des durées de traitement des différentes tâches. Suite à cette première optimisation, une simulation peut être lancée. Nous proposons d'expérimenter dans ce chapitre une approche itérative de couplage optimisation/simulation basée à la fois sur de la simulation globale et de l'analyse d'opérations unitaires.

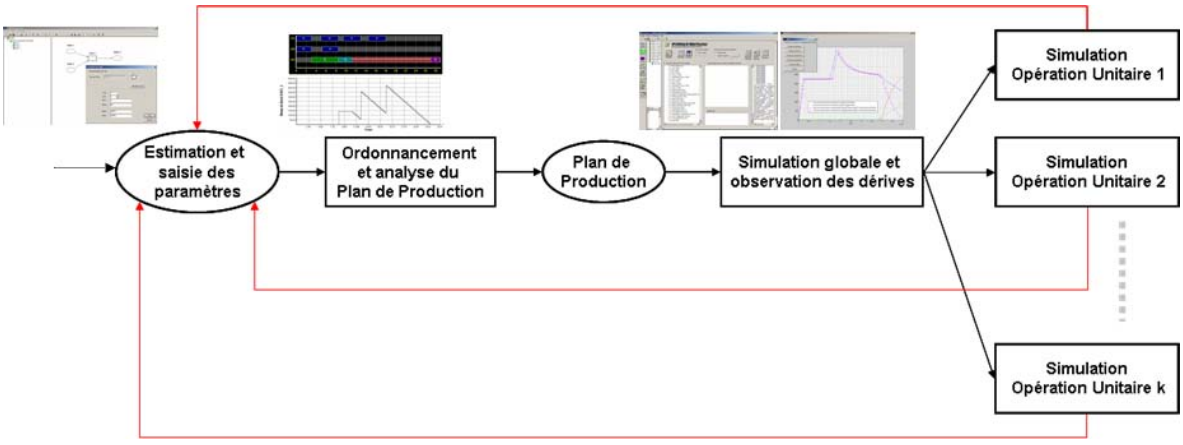


Figure 6.1 Stratégie d'exploitation du couplage optimisation/simulation

La mise en œuvre de cette stratégie d'exploitation est supportée par différents outils développés dans le cadre de ces travaux :

- La saisie du problème et l'adaptation des paramètres se fait via une interface graphique de type « drag and drop » au travers de l'outil : « RTN-Générator ».
- L'analyse des ordonnancements par diagrammes de Gantt et du suivi de l'évolution des stocks dans les cuves de stockage est réalisée avec l'outil « GanttChart ».
- L'observation des dérives en simulation est suivi via l'interface « PrODHyS Wintester ».

Les deux approches de couplage intégrées dans notre stratégie d'exploitation sont les suivantes :
le *couplage par opérations unitaires* qui vise à paramétrer le modèle d'optimisation par la simulation distincte de chaque opération unitaire seule (cf. Figure 6.2.a)
le *couplage par analyse globale* qui propose une approche itérative par simulation du procédé complet (cf. Figure 6.2.b)

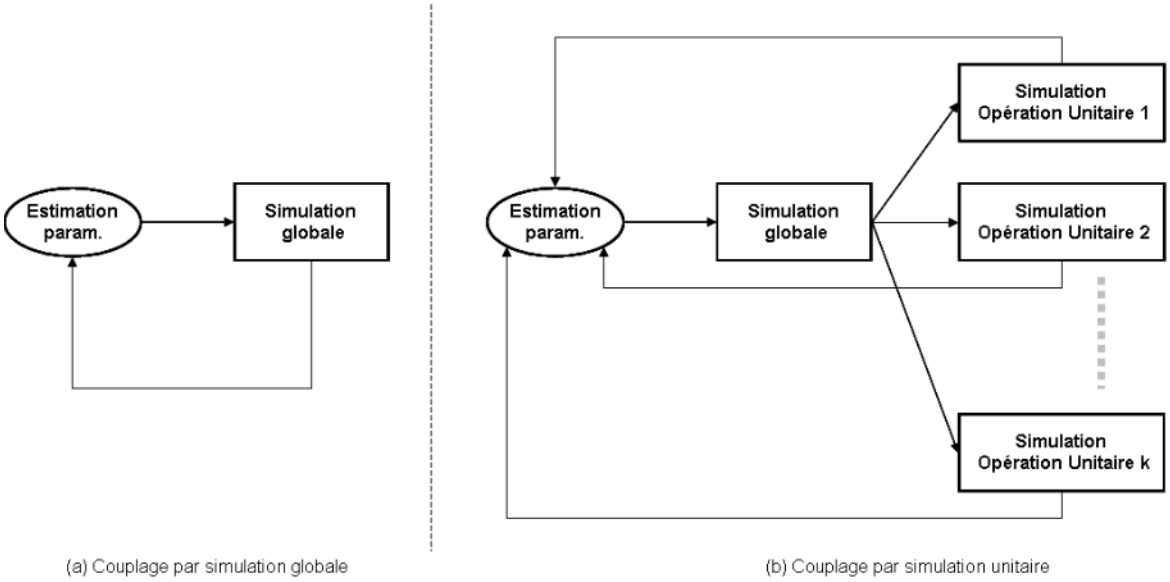


Figure 6.2 Différentes approches de couplage optimisation/simulation

6.2. OUTILS DÉVELOPPÉS : PROSCHED / SCHEDTOOLS

6.2.1. Saisie du problème d'ordonnancement : RTN-Generator

Afin de pallier les difficultés liées à la programmation du modèle de simulation, il nous paraît intéressant de proposer des *Générateurs de Code*. La description du modèle de simulation est réalisée au travers d'interfaces graphiques de type *« drag and drop »* qui semblent aujourd'hui la meilleure option du point de vue de l'accessibilité à un outil complexe. Dans ce cadre, deux générateurs sont actuellement en cours de développement : un pour spécifier la commande, l'autre pour construire le modèle de simulation du procédé. Ces générateurs de code sont des applications *Windows* s'appuyant sur un système de multifenêtrage développé à l'aide des *MFC*. Comme ces outils ont de nombreuses fonctionnalités communes, un certain nombre de packages ont été définis (gestion des éléments graphiques, fonction de génération) afin d'être réutilisés pour l'implémentation d'autres interfaces à venir.

Lors des chapitres précédents, nous avons vu que la modélisation *RTN* d'un problème est cruciale pour sa résolution. Il nous a donc semblé nécessaire d'offrir à l'utilisateur la possibilité de décrire son problème au format *RTN* et d'en sauvegarder les différentes études, pour qu'elles soient ensuite résolues par le solveur *Xpress-MP*. Dans ce contexte, le *RTN-Generator* a été développé afin de faciliter la démarche itérative lors du couplage Optimisation / Simulation. Celui-ci permet à l'utilisateur de créer plus facilement le fichier d'initialisation du modèle d'optimisation qui va être envoyé au modèle codé sous *Xpress-MP*. La saisie des hypothèses se fait par l'intermédiaire d'une interface graphique de type *« drag and drop »* : ajout et suppression d'unités, de tâches...

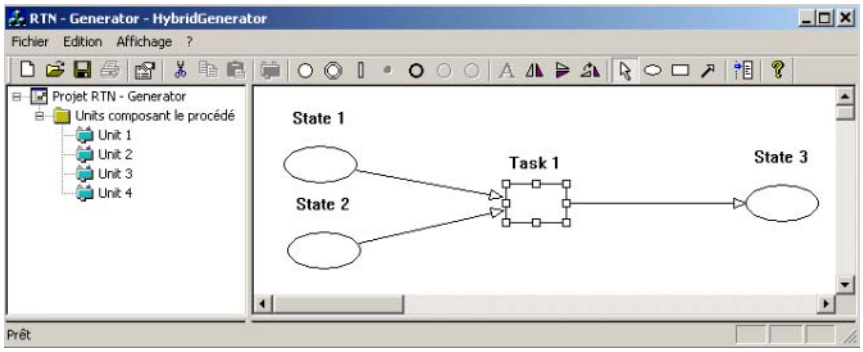


Figure 6.3 Interface de l'outil RTN-Generator

Cet outil permet donc de modifier facilement les paramètres du modèle *RTN* qui sont autant d'hypothèses faites en entrée du modèle d'optimisation et de sauvegarder les différentes versions de *RTN*. La modification de ces paramètres se fait par l'intermédiaire de différentes boîtes de dialogue (Figure 6.4)

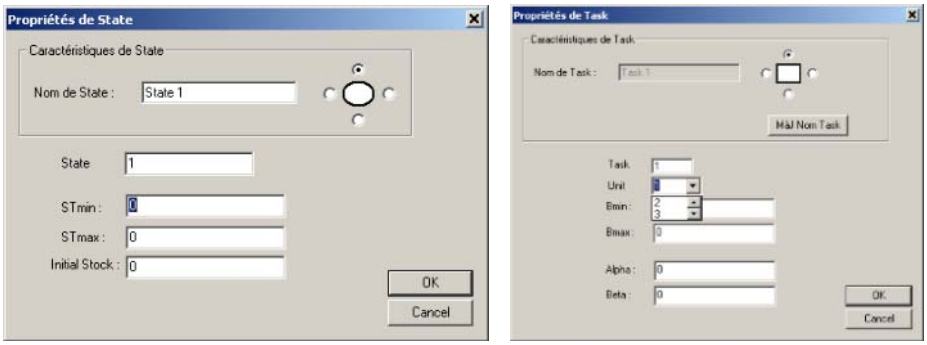


Figure 6.4 Boîtes de dialogue de l'outil RTN-Generator

Un *RTN* est un projet qui est constitué de données de base qui sont les *tâches* et les *opérations*. Les *tâches* correspondent à l'association au sein du graphe d'une *opération* et d'une *unité*. (Figure 6.5)

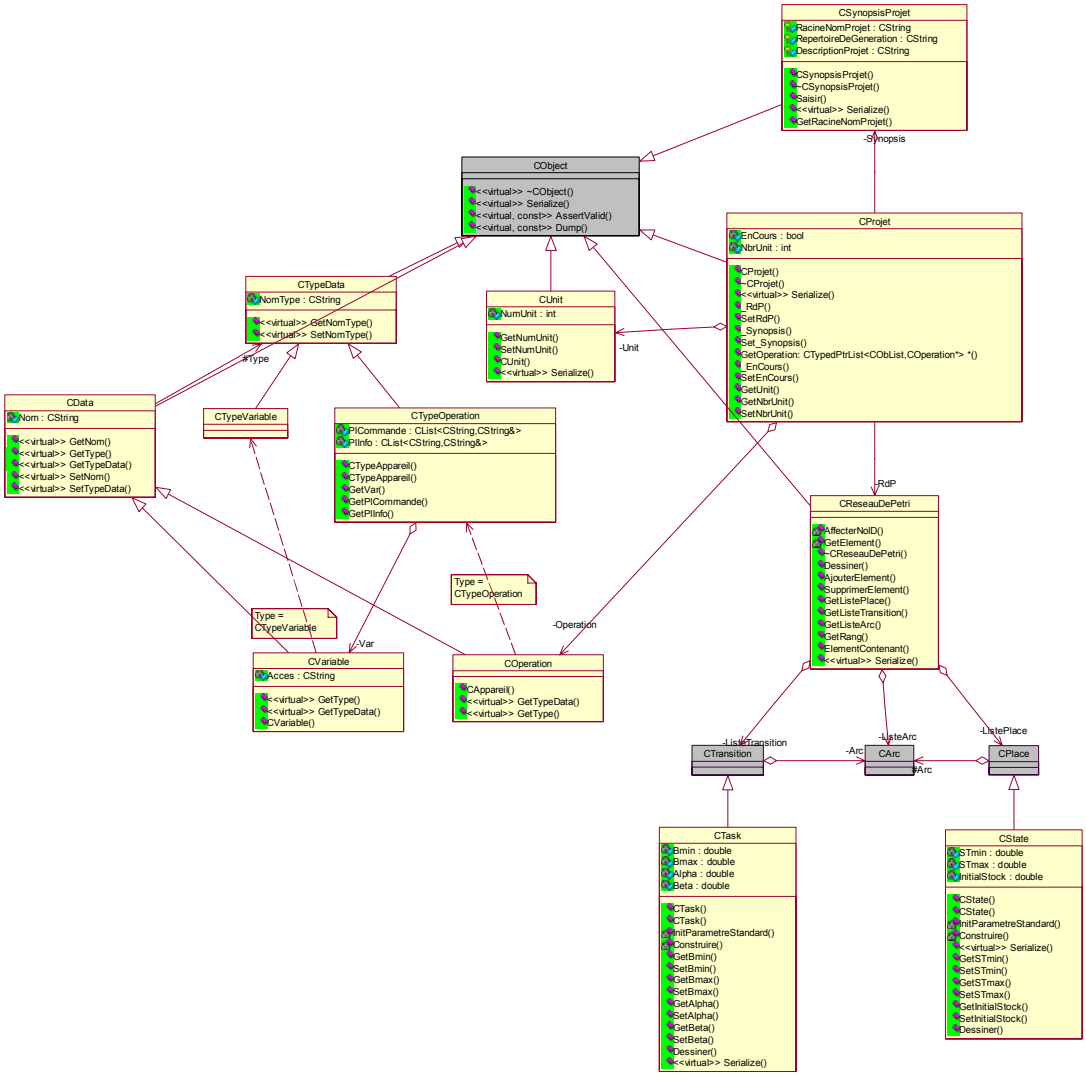


Figure 6.5 Diagramme de classe de l'outil RTN-Generator

6.2.1.1. Le projet dans le *RTN-Generator*

La classe `CProjet` contient l'ensemble des appareils *actifs* constituant le procédé ainsi que le graphe qui définit le RTN. Les objets *tâche* et *état* héritent des places et des transitions du réseau de Petri. En effet, cet outil a été conçu afin de servir de base à un futur générateur de recette qui permettrait de modéliser graphiquement l'ensemble de la *partie commande* du procédé à savoir :

la création du *RdP* recette du procédé incluant les aspects « conduite »

la création du *RTN* décrivant le problème d'ordonnancement de la partie commande.

6.2.1.2. Les états (classe *`CState`*)

La classe `CState` contient une partie des paramètres nécessaires au module d'ordonnancement soit :

les seuils minimaux et maximaux des états

les valeurs initiales de stockage

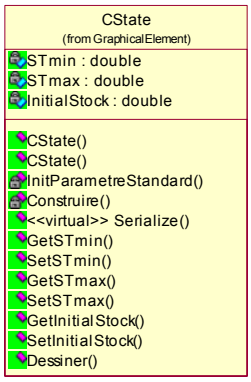


Figure 6.6 Le diagramme de la classe `Cstate` *au sein de l'outil* RTN-Generator

6.2.1.3. Les tâches (classe ***CTask***)

La classe `CTask` contient les paramètres complémentaires nécessaires au module d'ordonnancement soit :

les tailles minimales et maximales des lots

les durées de traitement dépendantes et indépendantes de la taille des lots

Comme nous l'avons noté précédemment, le lien entre la tâche et la ressource se fait au niveau du graphe.

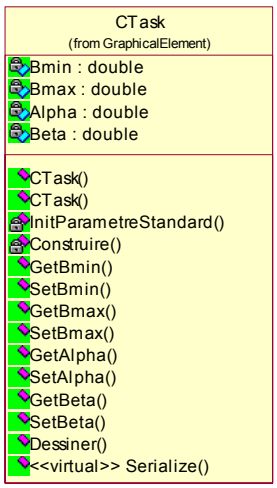


Figure 6.7 Le diagramme de la classe `CTask` *au sein de l'outil* RTN-Generator

6.2.1.4. La génération automatique du fichier d'initialisation

Lorsque l'utilisateur a terminé son *RTN* et qu'il a paramétré l'ensemble des tâches / états / connexions, il peut lancer la génération automatique du fichier d'initialisation.

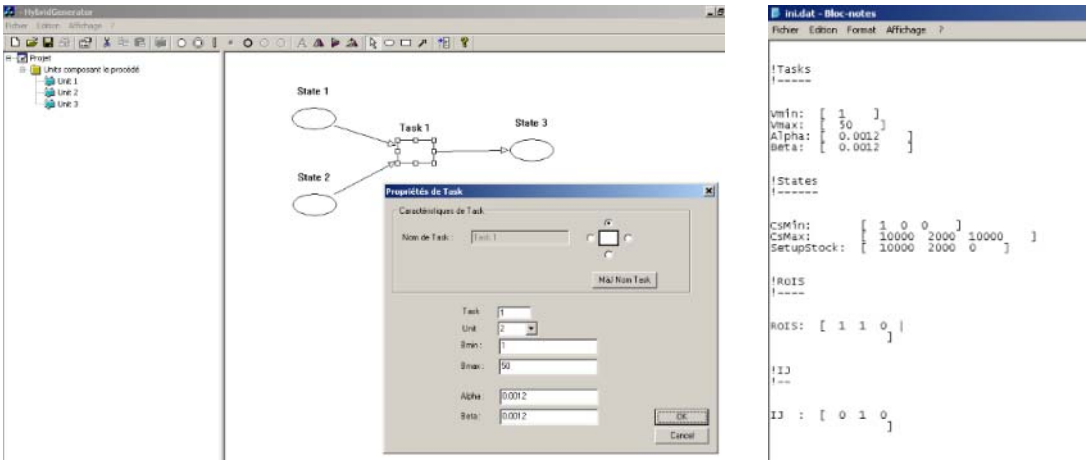


Figure 6.8 Interface de l'outil GANTTChart

6.2.2. L'analyse des résultats de la phase ordonnancement du problème : *GANTTChart*

Afin de simplifier les analyses, un outil a été développé permettant d'interpréter les données en sortie du modèle d'ordonnancement afin d'afficher :

le diagramme de Gantt de la solution obtenue

l'évolution des stocks pour les différents états

un tableau récapitulatif des données des batch (dates de début, dates de fin, taille du batch)

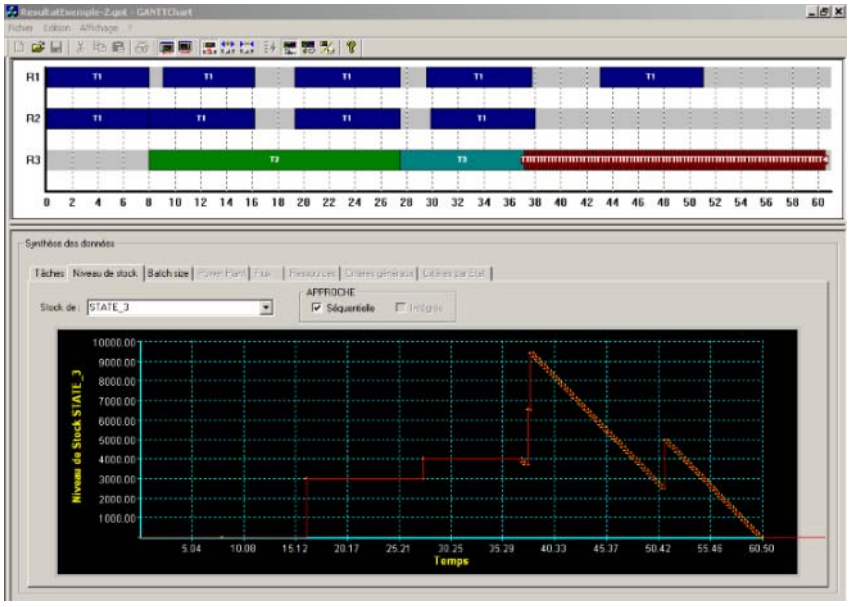


Figure 6.9 Interface de l'outil GANTTChart

6.3. STRATÉGIE D'EXPLOITATION DU COUPLAGE ENTRE OPTIMISATION ET SIMULATION : APPLICATION 1

Nous allons illustrer les concepts développés dans les chapitres précédents, ainsi que les différentes approches de couplage dans l'exemple suivant.

6.3.1. Le procédé mis en œuvre :

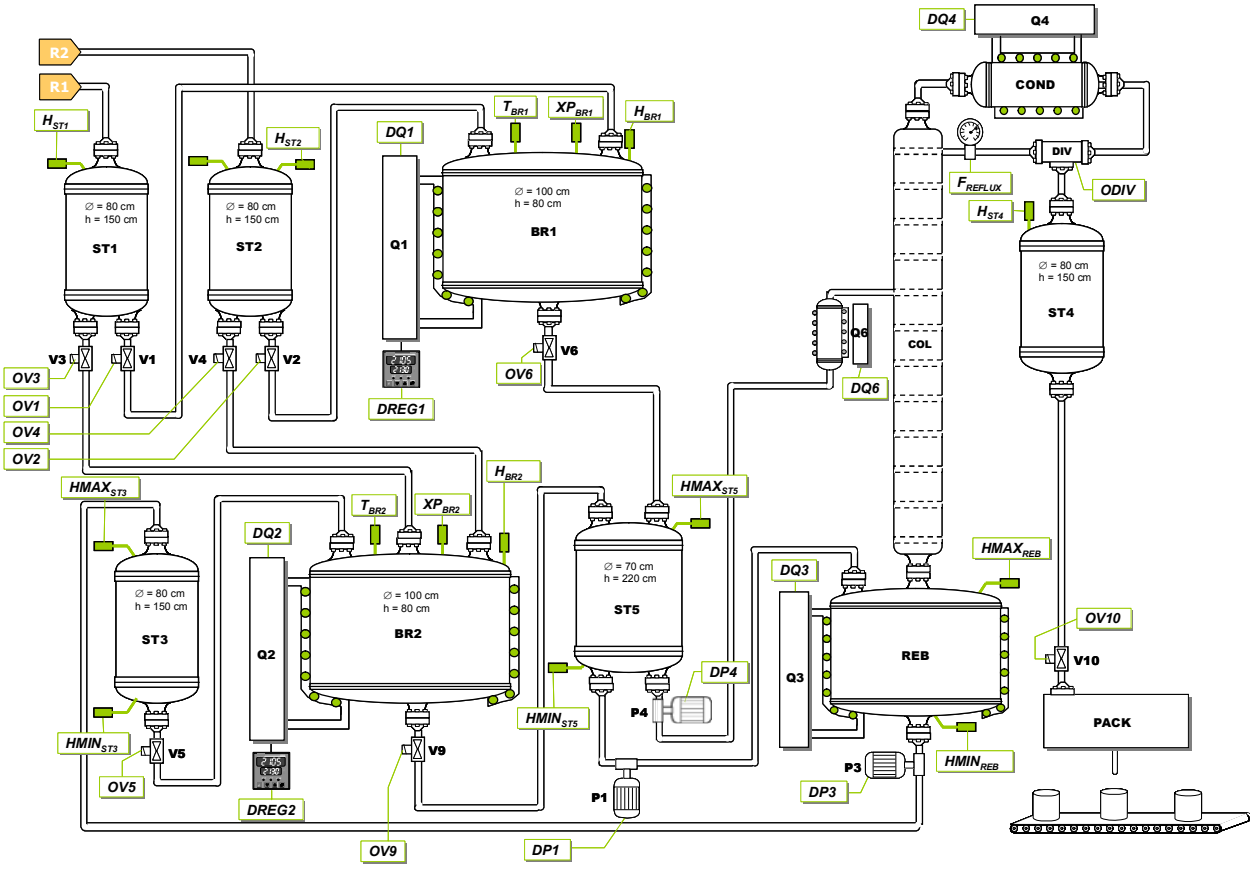


Figure 6.10 Exemple de procédé semi-continu dérivé de [Joglekar et al.1985]

Le procédé mis en œuvre est décrit sur la Figure 6.10. Le but de cette installation est de fabriquer et de conditionner un produit *P* dont la pureté doit être au moins égale à 98% molaire à partir de deux réactifs *R1* et *R2*.

6.3.2. Opération de réaction :

La réaction considérée est une réaction équilibrée endothermique impliquant deux réactifs *R1* et *R2* et un produit *P* :

$$R1 + R2 \longleftrightarrow P$$

Elle suit une loi cinétique d'ordre 2 de la forme :

$$r_R = k_1^{\circ}.e^{-\frac{Ea_1}{RT}}.x_{R1}.x_{R2} - k_{-1}^{\circ}.e^{-\frac{Ea_{-1}}{RT}}.x_P$$

Comme nous le montre la Figure 6.11, la fraction molaire de produit *P* ne peut excéder 0,83 et un temps de séjour infini serait nécessaire pour atteindre cette composition. Afin de maximiser le taux de conversion de la réaction sans pour autant pénaliser le temps de cycle du procédé, la réaction est stoppée dés que la composition en produit *P* atteint la valeur de 0,8. Par ailleurs, compte tenu de la loi de vitesse considérée, la réaction (*R*) est d'autant plus rapide que la température *T* est élevée. La température retenue pour la réaction doit garantir une réaction rapide tout en maintenant les constituants dans l'état liquide. Une température de 365 K permet de satisfaire ces deux contraintes.

La spécification de pureté imposée pour le produit *P* étant égale à 98%, une étape de distillation est donc nécessaire dans la recette générale du procédé complet.

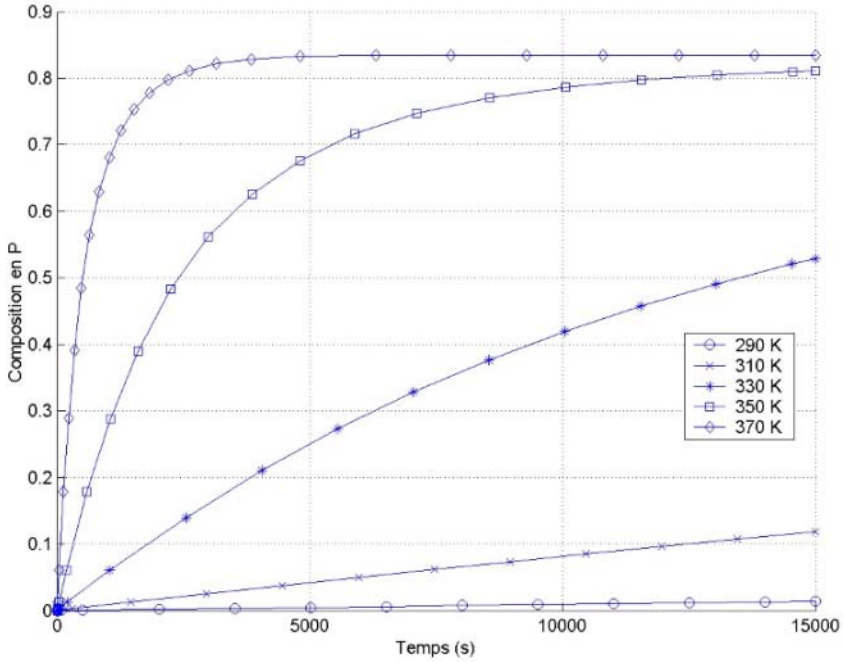


Figure 6.11 La vitesse de réaction en fonction de la température

6.3.3. La recette de fabrication

La recette générale du produit à fabriquer est présentée sur la Figure 6.12 sous la forme d'un *STN*.

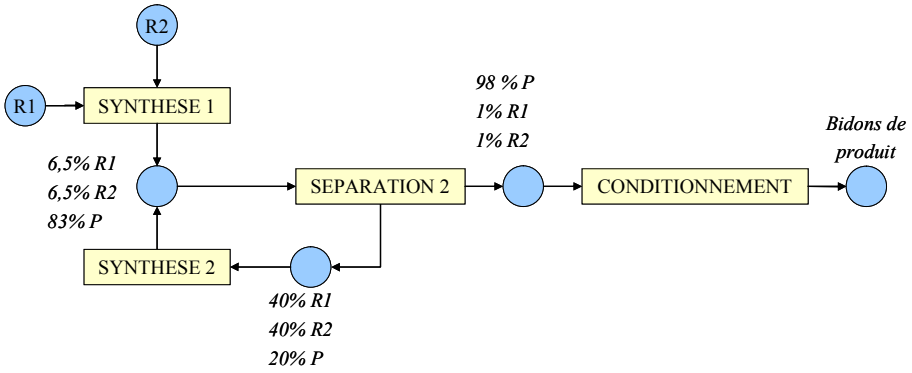


Figure 6.12 Recette de fabrication du produit P

Cette recette générale peut être détaillée en cinq opérations, dont les ressources sont présentées Figure 6.2 :

- La *fabrication* du produit *P* selon la réaction *R* qui peut être réalisée soit dans le réacteur *BR1*, soit dans le réacteur *BR2*. Cette opération comporte plusieurs phases dont le chargement, le préchauffage la réaction et la vidange.
- Le *stockage* du mélange après réaction dans la cuve *ST5*. Cette cuve permet de gérer le couplage entre les modes de production discontinus des réacteurs et le fonctionnement continu de la colonne.
- La *purification* du produit *P* avec le séparateur *SEP*, cette opération comprend les phases de chargement du rebouilleur, de remplissage et démarrage de la colonne, de mise en régime de la colonne, et finalement de séparation continue (distillation à reflux fini *r=1,2* par ouverture du diviseur de courant *OVDIV*) ou discontinue suivant les différentes hypothèses.
- Le *stockage* des réactifs n'ayant pas réagi se fait dans le réservoir *ST3*, ceux-ci sont recyclés dans le procédé et peuvent servir de matière première à la réaction mise en œuvre dans le réacteur *BR2*.
- Le *conditionnement* du produit *P*, stocké dans le réservoir *ST4* puis transféré dans des bidons de 10 litres.

6.3.4. La modélisation du procédé

Le procédé est modélisé selon l'architecture représentée sur la Figure 6.13. Le schéma donne, pour chaque élément, son type et la classe d'objet qui le décrit. Le procédé comporte :

- six cuves *STi* de *i*=1..5 (de type `StorageTank`),
- deux réacteurs (de type `HeatTank`),
- dix vannes Vi de i=1..10 (de type `TwoWayValve`), dans lesquelles les inversions de flux sont possibles,
- une colonne à distiller (de type `HeatDistColumn`),
- trois pompes (de type `PortActuator`),
- un diviseur de courant (de type `PortActuator`),
- un ensemble de capteurs de composition (de type `CompCaptor`) et de niveau (de type `LevelCaptor`) permettant de surveiller le système.

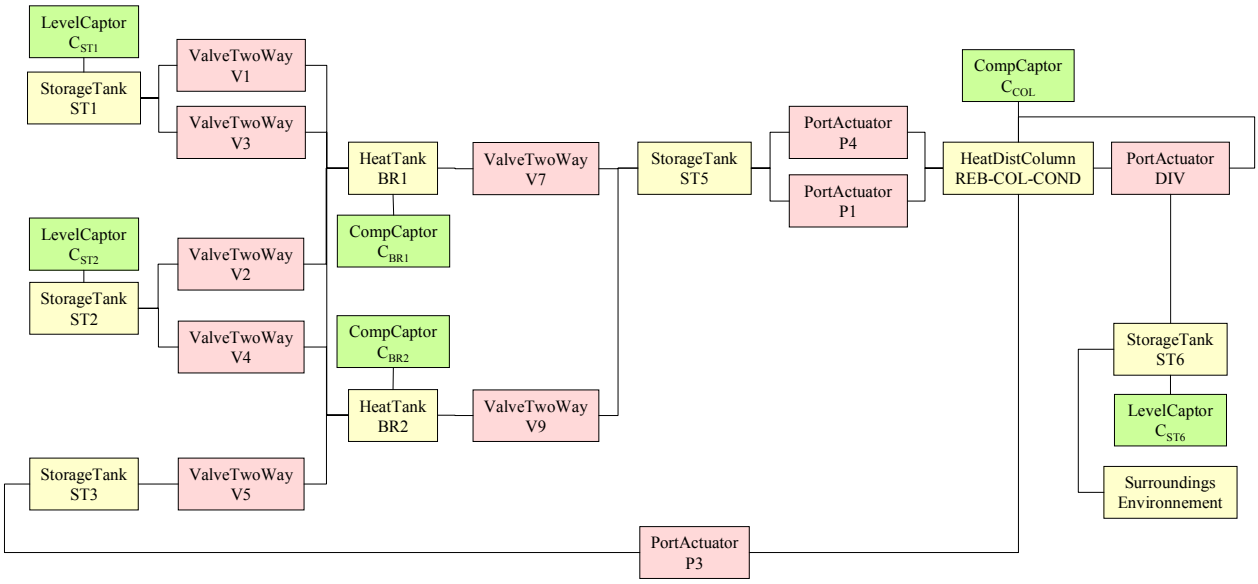


Figure 6.13 La modélisation du procédé [Joglekar et al., 1985]

6.3.5. Identification du « Centre de décision » de l'exemple

Les différents modes de fonctionnements de la colonne ont été identifiés et permettent de gérer la colonne comme une ressource multi-modale.

Suivant les hypothèses de fonctionnement retenues pour la colonne (continu ou discontinu), les étapes de remplissage, démarrage et arrêt demeurent. Seule la phase de séparation continue est ajoutée dans l'hypothèse correspondante.

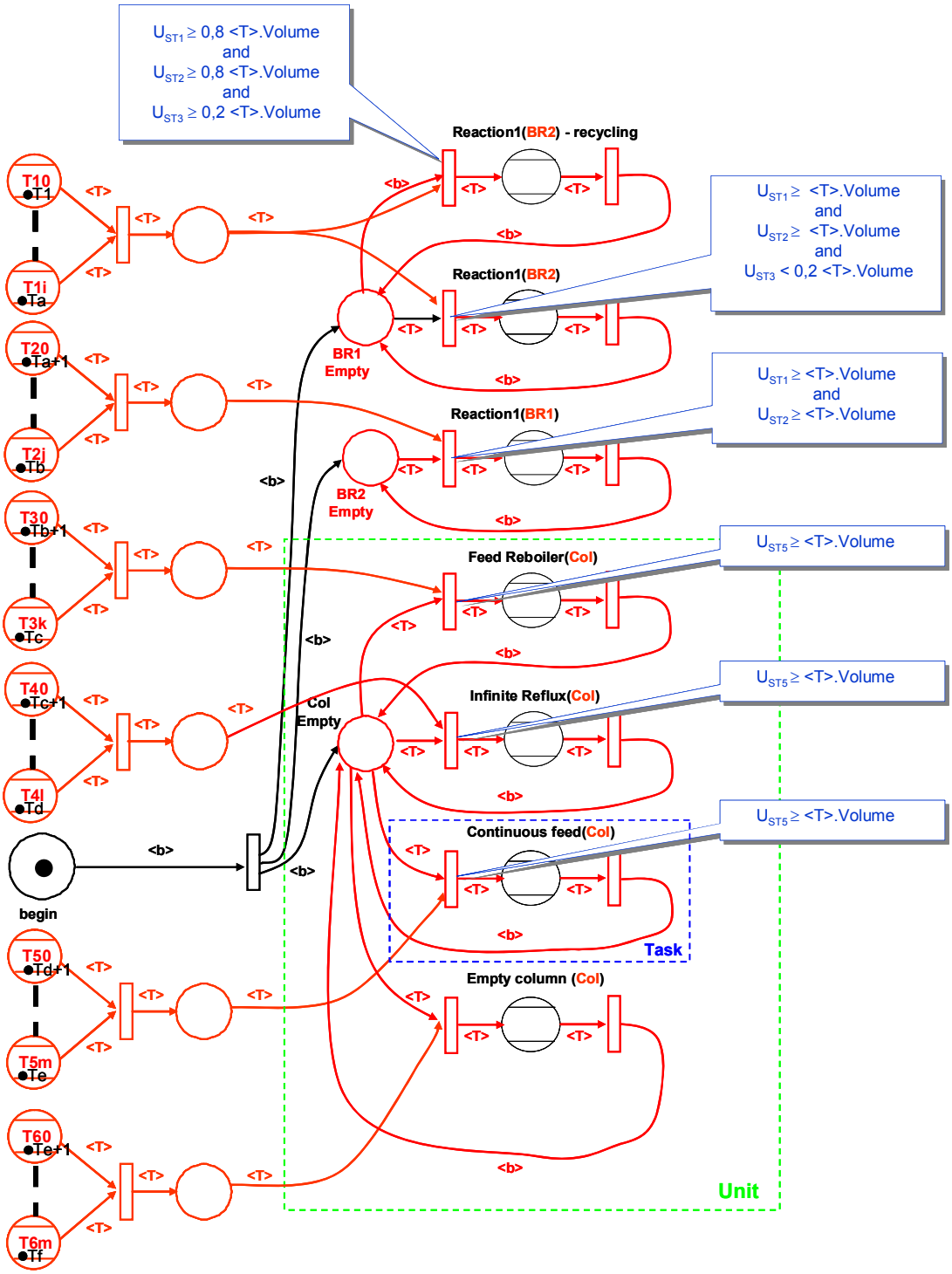


Figure 6.14 Recette et « centre de décision » *du modèle de simulation de l'exemple [Joglekar et al. 1985] avechypothèse de séparation contiune*

6.3.6. Modélisation RTN de l'exemple :

Deux hypothèses sont analysées, l'une considère une séparation continue pour atteindre la pureté souhaitée du produit *P*, tandis que l'autre hypothèse se base sur une séparation discontinue. Dans les deux cas, la colonne est une ressource à caractère multimodal.

6.3.6.1. Hypothèse de séparation continue

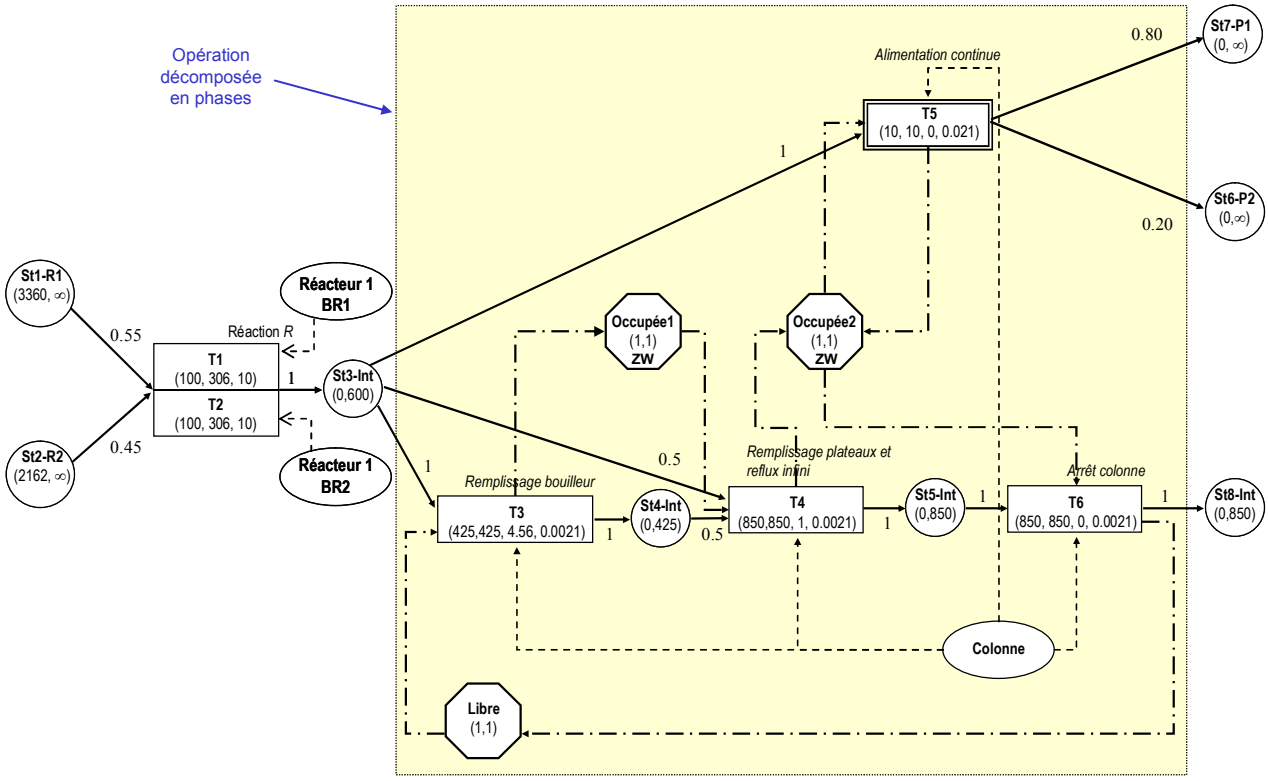


Figure 6.15 RTN exemple [Joglekar et al.*, 1985], avec hypothèse de séparation continue*

Paramètres :

Ordres de fabrication à réaliser : *600 kg* de *P2*.

Pas de délais de nettoyage des appareils.

Durées opératoires variables et dépendantes de la taille du lot et de l'appareil utilisé comprenant une partie fixe et une partie variable :

Les durées ont tout d'abord été estimées par la simulation d'un batch de taille moyenne, sans réaliser une analyse fine, puis une marge a été ajoutée pour la prise en compte de l'ensemble des tailles de lots admissibles.

n° Tâche	1	2	3	4	5	6
Ressource	**BR1**	**BR2**	**Remplissage bouilleur**	**Remplissage plateaux + Reflux**	**Reflux fini**	**Vidange Colonne**
df (h)	10	10	4.56	1	0	0
dv (h/kg)	0,06	0,06	0,0021	0.0021	0.021	0.0021

Tableau 6.1 Durées opératoires variables (température de réaction 365 K)

Bilan :

Plusieurs tentatives d'essais par simulation directe n'ont pas permis de stabiliser un mode de séparation continue. Après analyse, les durées et capacités des réacteurs ne permettent pas un couplage de ce type avec le dimensionnement et les paramètres opératoires imposés. Ce point a été confirmé par l'impossibilité de résoudre le modèle d'optimisation. L'orientation vers une séparation discontinue a alors été envisagée.

6.3.6.2. Hypothèse de séparation discontinue

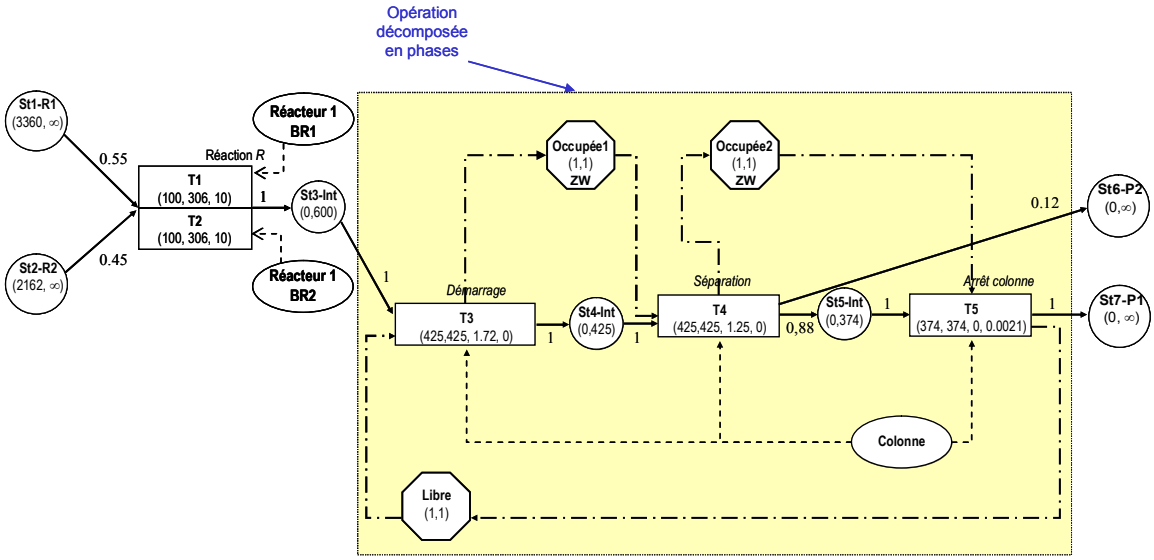


Figure 6.16 RTN exemple [Joglekar et al., 1985], avec hypothèse de séparation discontinue

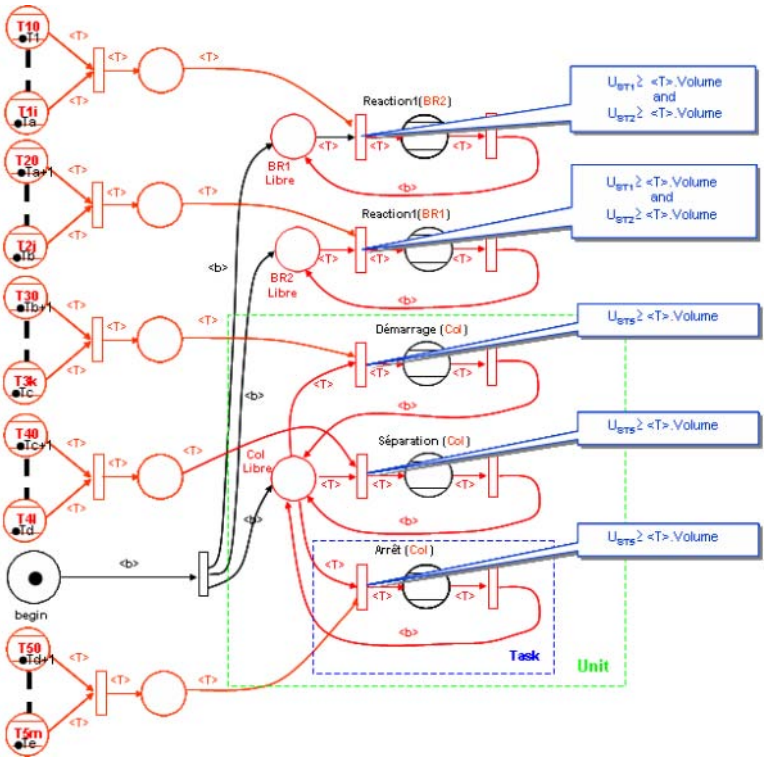


Figure 6.17 Création de la recette de contrôle sous PrODHyS avec hypothèse de séparation discontinue

Afin de pouvoir simuler le flowsheet complet, nous faisons l'hypothèse d'une séparation discontinue. Les durées utilisées pour initialiser le modèle d'ordonnancement sont tout d'abord estimées de manière approchée. Les paramètres sont les suivants :

Paramètres :
Ordres de fabrication à réaliser : *600 kg* de *P2*.

Pas de délais de nettoyage des appareils.

Durées opératoires variables et dépendantes de la taille du lot et de l'appareil utilisé comprenant une partie fixe et une partie variable :

Les durées ont tout d'abord été estimées par la simulation d'un batch de taille moyenne, sans réaliser une analyse fine, puis une marge a été ajoutée pour la prise en compte de l'ensemble des tailles de lots admissibles.

n° Tâche	1	2	4	5	6
Ressource	BR1	BR2	Démarrage	Séparation	Vidange Colonne
df (h)	10	10	1.72	1.25	0
dv (h/kg)	0,06	0,06	0	0	0.0021

Tableau 6.2 Durées opératoires variables (température de réaction 365 K)

Solution :
Solution obtenue en 505,2 s (solution sous-optimale).
(XPRESS-MP 1.6.3, processeur INTEL Pentium Centrino 1,76 GHz (1 Go RAM))

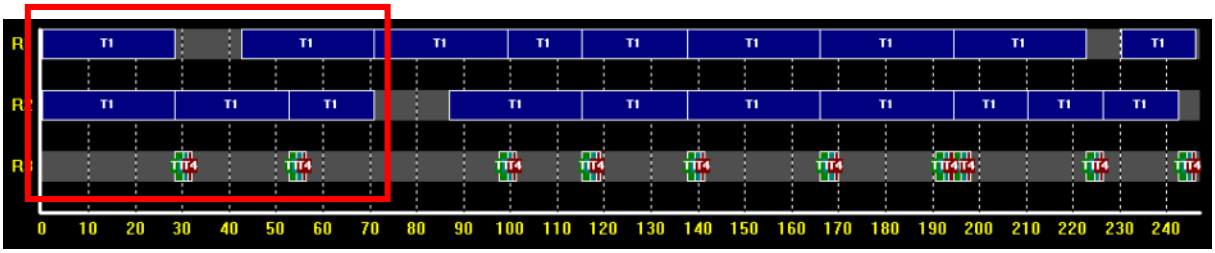


Figure 6.18 Diagramme de Gantt – Solution durées estimées

On s'intéressera plus particulièrement dans cet exemple à l'enchaînement des tâches de réaction sur les ressources *BR1* et *BR2*.

Le gain en terme de modélisation de la recette en simulation pour l'exemple suivant est de l'ordre d'un facteur neuf. En effet, la combinatoire liée aux différentes tailles de lots sur les différents appareils entraîne une complexité croissante de la recette de contrôle.

Tâche	t	Durée	Machine	Produit	Batch
T1	0.00	28.36	1	1	306.00
T1	42.56	28.36	1	1	306.00
T1	70.92	28.36	1	1	306.00
T1	99.28	16.00	1	1	100.00
T1	115.28	22.39	1	1	206.50
T1	137.67	28.36	1	1	306.00
T1	166.03	28.36	1	1	306.00
T1	194.39	28.36	1	1	306.00
T1	230.21	16.00	1	1	100.00
T1	0.00	28.36	2	1	306.00
T1	28.36	24.28	2	1	238.00
T1	52.64	18.28	2	1	138.00
T1	86.92	28.36	2	1	306.00
T1	115.28	22.39	2	1	206.50
T1	137.67	28.36	2	1	306.00
T1	166.03	28.36	2	1	306.00
T1	194.39	16.00	2	1	100.00
T1	210.39	16.00	2	1	100.00
T1	226.39	16.06	2	1	101.00
T2	28.36	1.72	3	2	425.00
T2	52.64	1.72	3	2	425.00
T2	97.56	1.72	3	2	425.00
T2	115.28	1.72	3	2	425.00
T2	137.67	1.72	3	2	425.00
T2	166.03	1.72	3	2	425.00
T2	190.63	1.72	3	2	425.00
T2	194.39	1.72	3	2	425.00
T2	222.75	1.72	3	2	425.00
T2	242.45	1.72	3	2	425.00
T3	30.08	1.25	3	3	425.00
T3	54.36	1.25	3	3	425.00

Tâche	t	Durée	Machine	Produit	Batch
T3	99.28	1.25	3	3	425.00
T3	117.00	1.25	3	3	425.00
T3	139.39	1.25	3	3	425.00
T3	167.75	1.25	3	3	425.00
T3	192.35	1.25	3	3	425.00
T3	196.11	1.25	3	3	425.00
T3	224.47	1.25	3	3	425.00
T3	244.17	1.25	3	3	425.00
T4	31.33	0.79	3	4	374.00
T4	55.61	0.79	3	4	374.00
T4	100.53	0.79	3	4	374.00
T4	118.25	0.79	3	4	374.00
T4	140.64	0.79	3	4	374.00
T4	169.00	0.79	3	4	374.00
T4	193.60	0.79	3	4	374.00
T4	197.36	0.79	3	4	374.00
T4	225.72	0.79	3	4	374.00
T4	245.42	0.79	3	4	374.00

Tableau 6.3 Tableau de résultats de l'ordonnancement – Solution durées estimées

6.3.7. Phase de couplage par analyse globale

L'analyse par simulation des opérations unitaires permet de mieux paramétrer le modèle d'ordonnancement. Cependant, certaines restrictions intrinsèques au procédé ne pourront pas être prises en compte dans le modèle d'ordonnancement.

Le contrôle du nombre de moles transférées ne peut être réalisé qu'au niveau de la cuve de stockage amont du réacteur. En effet, le remplissage d'un réacteur en produit *R2* s'accompagne du démarrage de la réaction, ce qui impose le positionnement des capteurs de contrôle du remplissage (capteur de niveau) en amont du réacteur. Par conséquent, le positionnement de ces capteurs ne permet en aucun cas un transfert simultané d'un même produit vers les deux réacteurs.

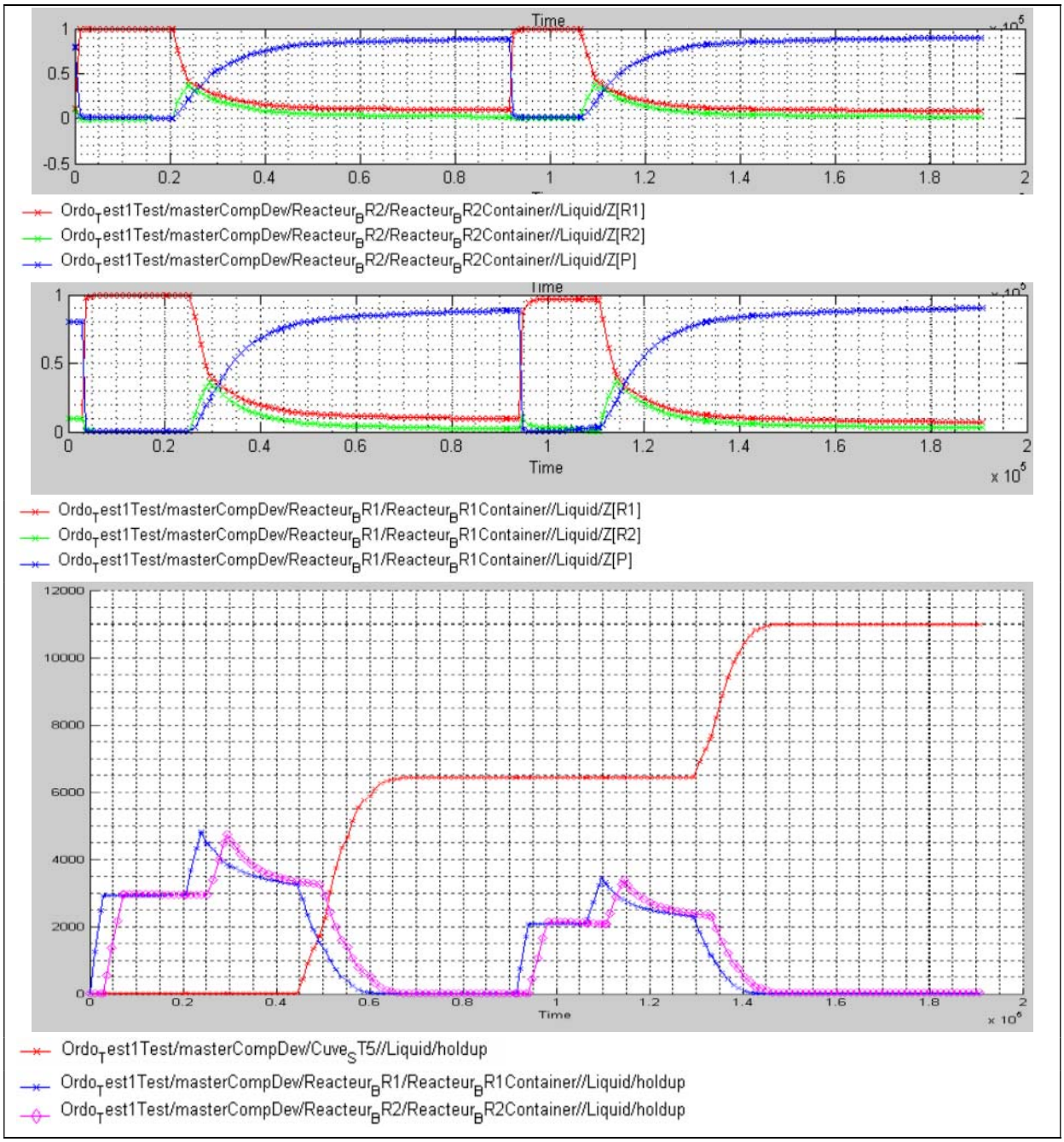


Figure 6.19 Mise en œuvre des places sémaphores et transferts simultanés

Afin d'éviter tout transfert parallèle, des places sémaphores (qui permettent de gérer la disponibilité des cuves) ont été ajoutées au niveau des macro-places génériques modélisant les phases de remplissage des opérations de réaction. Ces développements entraînent une incertitude sur le déroulement de l'ordonnancement des tâches de réaction liée à la disponibilité des cuves de matières premières. De telles incertitudes, prises en compte au niveau de l'ordonnancement par une légère surestimation des temps de traitement sont levées lors de la simulation (Figure 6.19). Dans le cas où ces incertitudes ne seraient pas prises en compte, une dérive peut être détectée par comparaison :

- des dates planifiées par l'ordonnancement et stockées dans les jetons lots,
- et des dates de changement de statut de ces mêmes jetons lors de la simulation de la recette.

6.3.8. Phase de couplage par opérations unitaires

6.3.8.1. Résultats de simulation des durées estimées

Réacteur - BR1

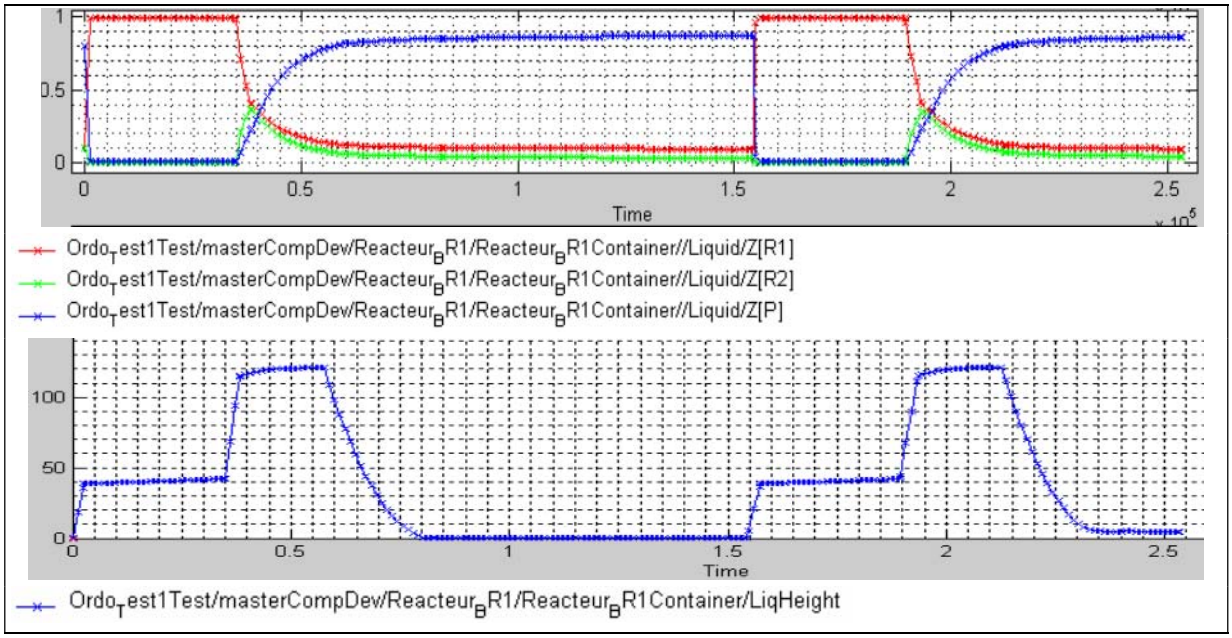


Figure 6.20 Evolution des concentrations et du niveau de liquide dans BR1 - Simulation « durées estimées »

Réacteur - BR2

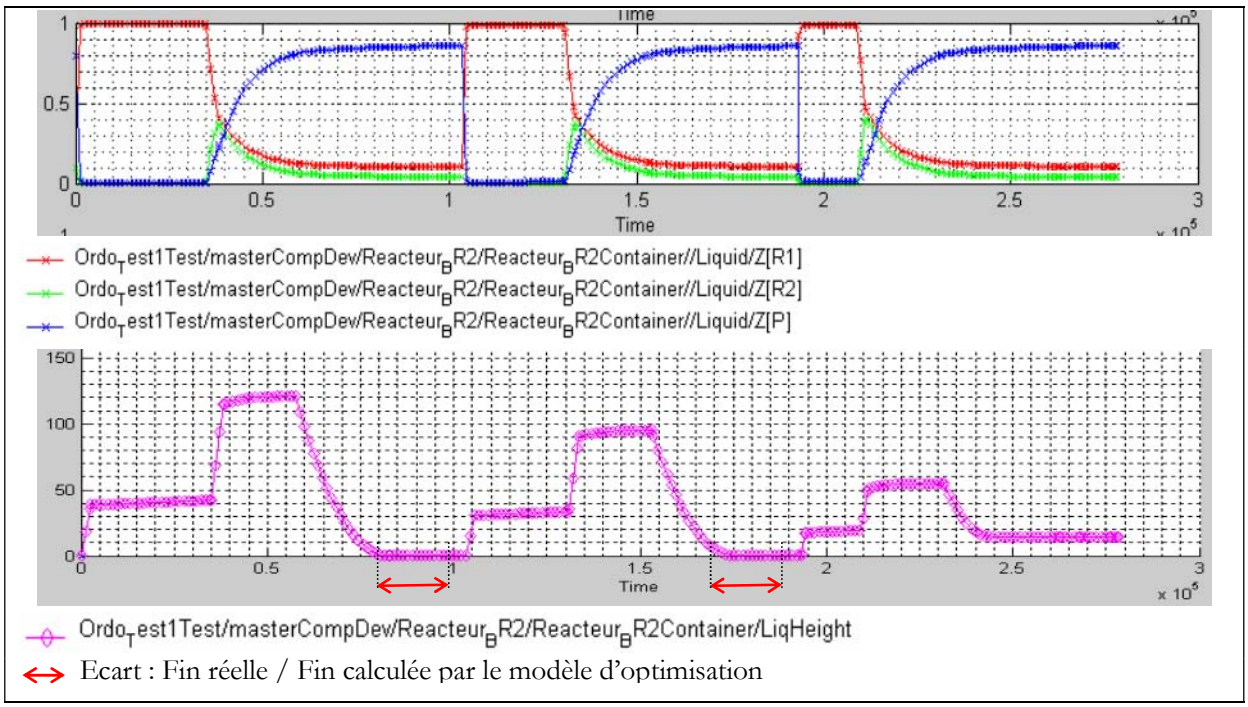


Figure 6.21 Evolution des concentrations et du niveau de liquide dans BR2 - Simulation « durées estimées »

Les durées estimées permettent de lancer une simulation, cependant lors du lancement de tâches supposées successives, la simulation fait apparaître des temps d'attente (Figure 6.21) liés à la surestimation des temps de traitement sur les réacteurs. Une analyse par simulation des opérations unitaires permet d'adapter ces temps de traitement.

6.3.8.2. Adaptation du modèle d'optimisation

Les durées opératoires des phases de réaction ont été évaluées par simulation, à l'aide du simulateur *PrODHyS*.

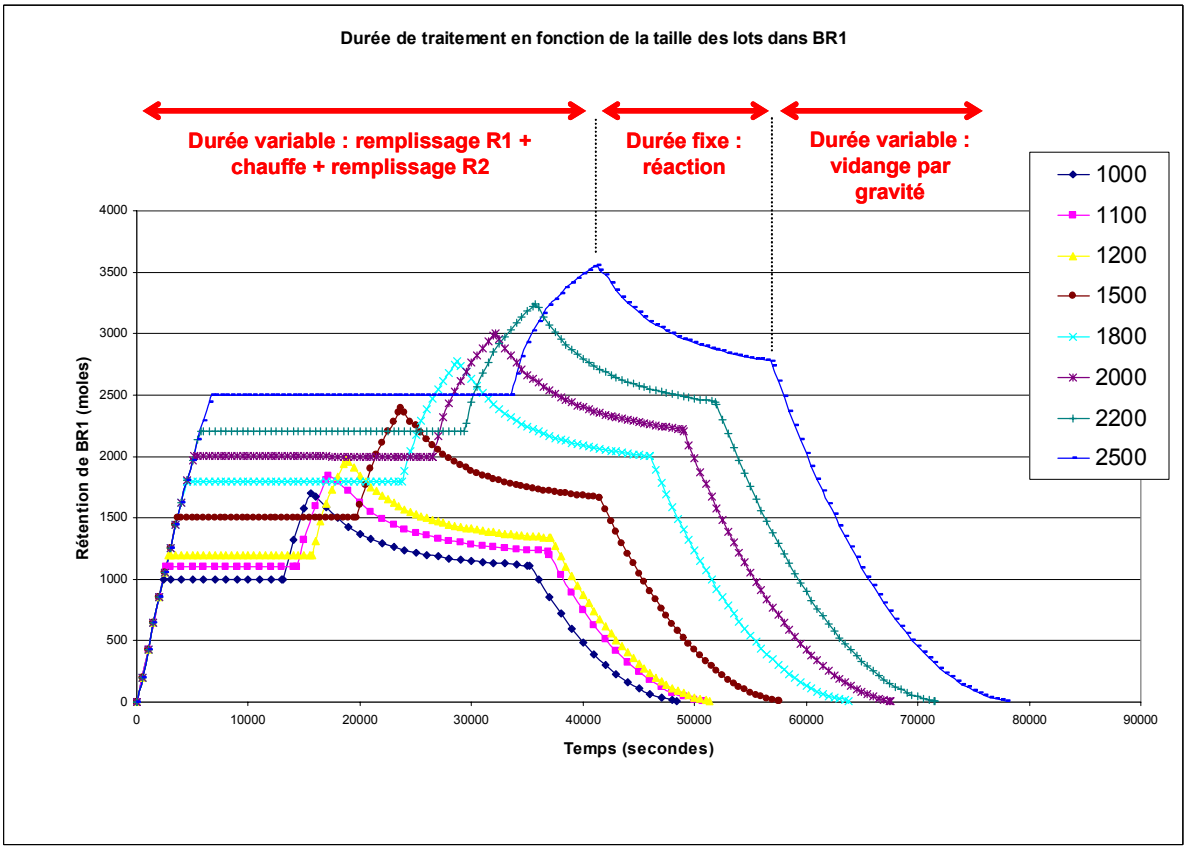


Figure 6.22 Identification des durées de traitement de l'opération de réaction dans BR1 et BR2

Ces durées opératoires peuvent être divisées en trois parties (Figure 6.22) :

- les durées opératoires fixes : temps de réaction lié à la température contrôlée,
- les durées de remplissage proportionnelles aux tailles de lots (débits fixes),
- les temps de chauffe et de vidange (fonctions non linéaires des tailles de lots).

En première approximation, nous avons identifié une partie fixe et une partie variable pour l'opération de réaction, en traçant la durée opératoire (en heures) en fonction de la taille des lots (en kg) sur la Figure 4.40, ce qui nous a permis de remplir le Tableau 4.11. En effet, les tailles de lot étant importantes (entre 100 et 300 kg), les aspects non linéaires peuvent être négligés.

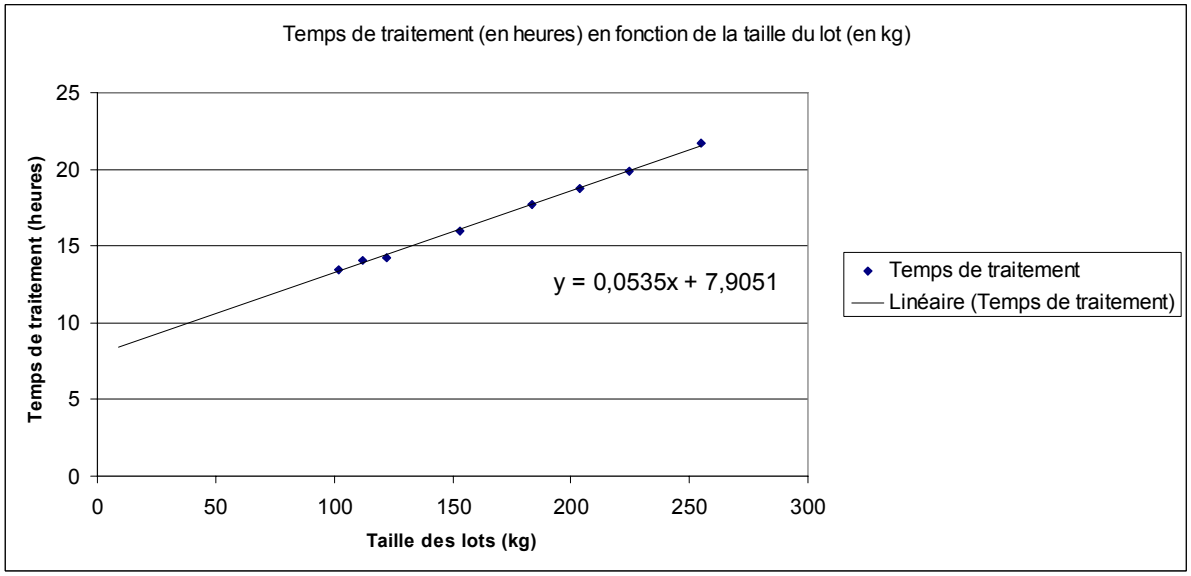


Figure 6.23 Durées de traitement de l'opération de réaction dans BR1 et BR2

Durées opératoires variables et dépendantes de la taille du lot et de l'appareil utilisé comprenant une partie fixe et une partie variable :

n° Tâche	1	2
Ressource	BR1	BR2
df (h)	7,9	7,9
dv (h/kg)	0,0535	0,0535

Tableau 6.4 Durées opératoires (température de réaction de 365 K)

6.3.8.3. Résultats de la simulation des durées affinées

Paramètres :

Ordres de fabrication à réaliser : *600 kg* de *P2*.

Pas de délais de nettoyage des appareils.

Durées opératoires variables et dépendantes de la taille du lot et de l'appareil utilisé comprenant une partie fixe et une partie variable :

n° Tâche	1	2	4	5	6
Ressource	BR1	BR2	Démarrage	Séparation	Vidange Colonne
df (h)	7,9	7,9	1.72	1.25	0
dv (h/kg)	0,0535	0,0535	0	0	0.0021

Tableau 6.5 Durées opératoires variables (température de réaction 365 K)

Solution :
Solution obtenue en 183,7 s (solution sous-optimale).
(XPRESS-MP 1.6.3, processeur INTEL Pentium Centrino 1,76 GHz (1 Go RAM))

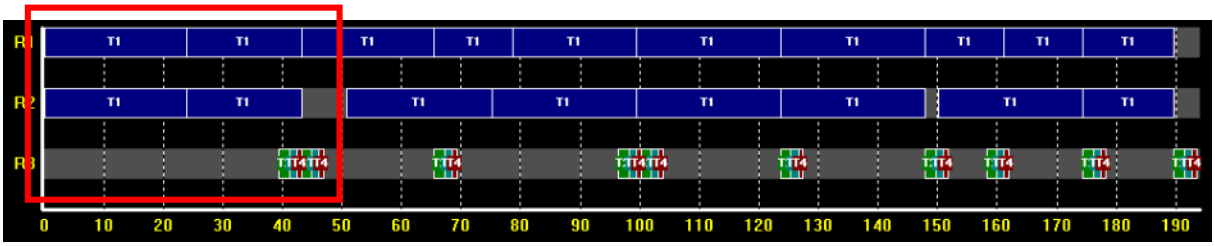


Figure 6.24 Diagramme de Gantt – Solution « durées affinées »

Tâche	t	Durée	Machine	Produit	Batch
T1	0.00	23.95	1	1	300.00
T1	23.95	19.27	1	1	212.50
T1	43.22	22.29	1	1	269.00
T1	65.51	13.25	1	1	100.00
T1	78.76	20.63	1	1	238.00
T1	99.39	24.27	1	1	306.00
T1	123.66	24.27	1	1	306.00
T1	147.94	13.25	1	1	100.00
T1	161.19	13.25	1	1	100.00
T1	174.44	15.28	1	1	138.00
T1	0.00	23.95	2	1	300.00
T1	23.95	19.27	2	1	212.50
T1	50.85	24.27	2	1	306.00
T1	75.12	24.27	2	1	306.00
T1	99.39	24.27	2	1	306.00
T1	123.66	24.27	2	1	306.00
T1	150.16	24.27	2	1	306.00
T1	174.44	15.28	2	1	138.00
T2	39.46	1.72	3	2	425.00
T2	43.22	1.72	3	2	425.00
T2	65.51	1.72	3	2	425.00
T2	96.42	1.72	3	2	425.00
T2	100.18	1.72	3	2	425.00
T2	123.66	1.72	3	2	425.00
T2	147.94	1.72	3	2	425.00
T2	158.22	1.72	3	2	425.00
T2	174.44	1.72	3	2	425.00
T2	189.72	1.72	3	2	425.00
T3	41.18	1.25	3	3	425.00
T3	44.94	1.25	3	3	425.00
T3	67.23	1.25	3	3	425.00

Tâche	t	Durée	Machine	Produit	Batch
T3	98.14	1.25	3	3	425.00
T3	101.90	1.25	3	3	425.00
T3	125.38	1.25	3	3	425.00
T3	149.66	1.25	3	3	425.00
T3	159.94	1.25	3	3	425.00
T3	176.16	1.25	3	3	425.00
T3	191.44	1.25	3	3	425.00
T4	42.43	0.79	3	4	374.00
T4	46.19	0.79	3	4	374.00
T4	68.48	0.79	3	4	374.00
T4	99.39	0.79	3	4	374.00
T4	103.15	0.79	3	4	374.00
T4	126.63	0.79	3	4	374.00
T4	150.91	0.79	3	4	374.00
T4	161.19	0.79	3	4	374.00
T4	177.41	0.79	3	4	374.00
T4	192.69	0.79	3	4	374.00

Tableau 6.6 Tableau de résultats de l'ordonnancement – Solution « durées affinées »

Réacteur - BR1

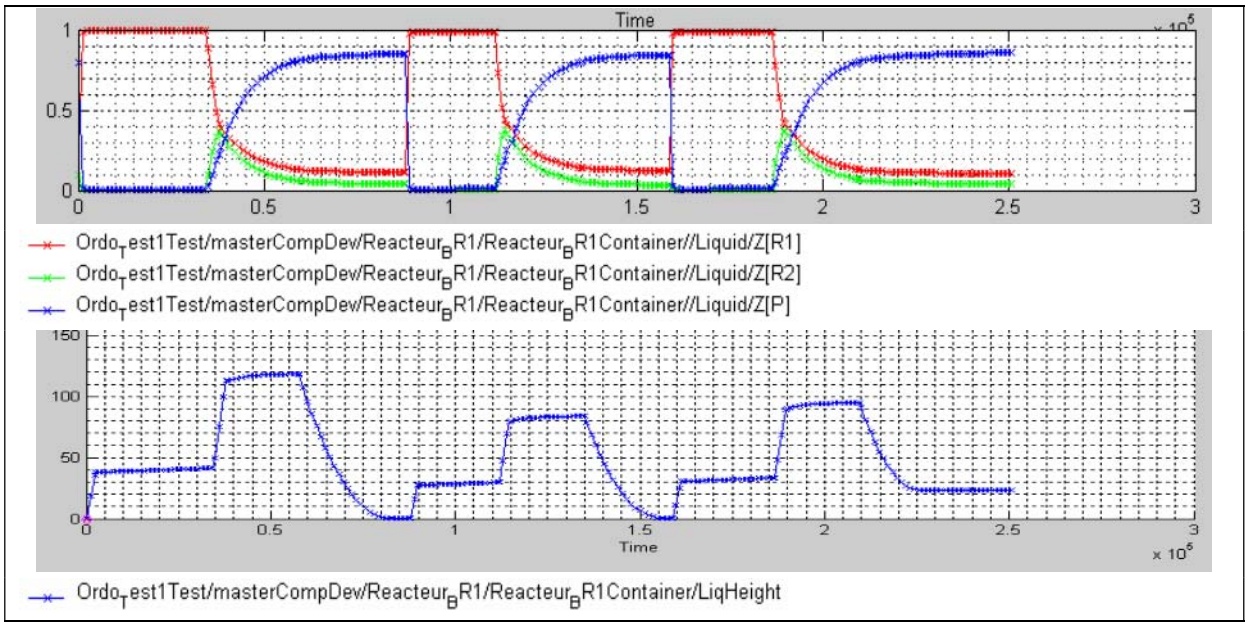


Figure 6.25 Evolution des concentrations et du niveau de liquide dans BR1 – Simulation « durées affinées »

L'adaptation des temps de traitement des réactions permettent de limiter les temps d'attente entre les différentes tâches et par conséquent d'optimiser l'utilisation des ressources (Figure 6.25 et Figure 6.26).

Réacteur - BR2

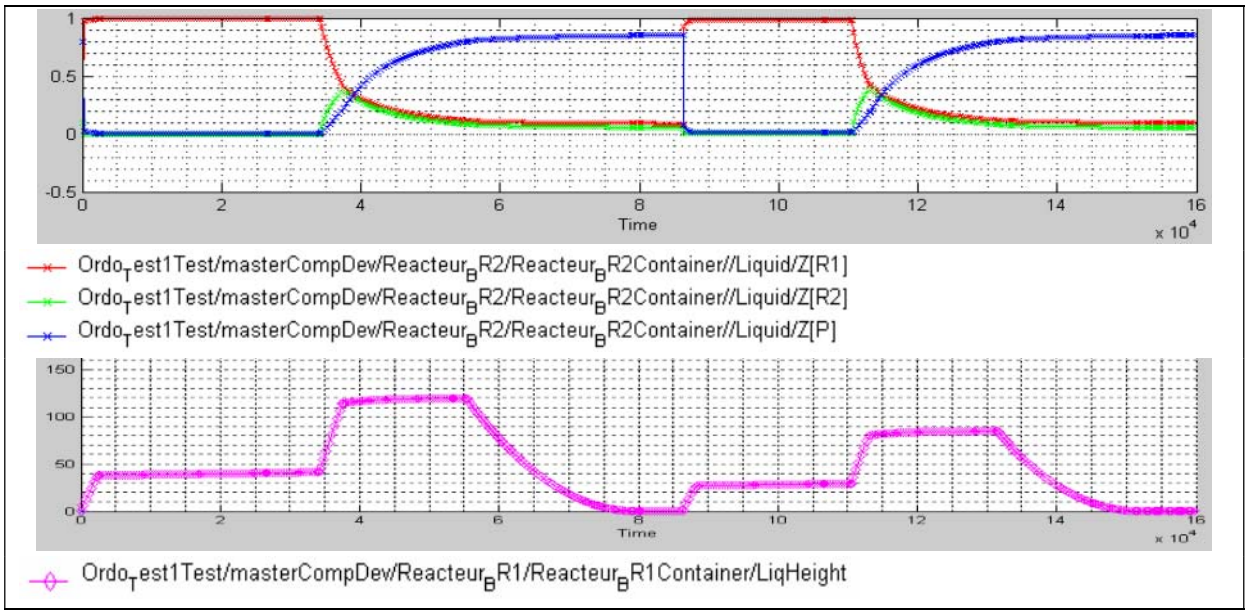


Figure 6.26 Evolution des concentrations et du niveau de liquide dans BR2 – Simulation durées affinées

De plus, l'implémentation du « Centre de décision » et des jetons lots permettent le lancement successifs de lots de tailles distincts (Figure 6.25 et Figure 6.26).

COLONNE

Figure 6.27 Evènement de démarrage de la colonne – Simulation durées affinées

Dans le cadre du couplage entre les opérations de réaction et la colonne à distiller, le pilotage des opérations multimodales est bien pris en compte par le simulateur (Figure 6.27).

6.3.9. Bilan

Une solution intéressante pour le paramétrage du modèle d'ordonnancement semble résider dans une approche mixte, couplant une analyse de sensibilité pour les opérations unitaires, tout en restant proche du scénario proposé par l'optimisation afin de garantir une simulation complète du procédé ; ce qui permettrait d'affiner rapidement les paramétrages des tâches par quelques simulations tout en respectant les limitations intrinsèques aux procédés et ainsi les intégrer au plus tôt dans la recherche d'un ordonnancement réalisable.

6.4. STRATÉGIE D'EXPLOITATION DU COUPLAGE ENTRE OPTIMISATION ET SIMULATION : APPLICATION 2 – MULTI-PRODUITS

6.4.1. Le procédé mis en œuvre :

Nous allons illustrer les concepts mis en œuvre dans les chapitres précédents, dans un exemple multi-produits.

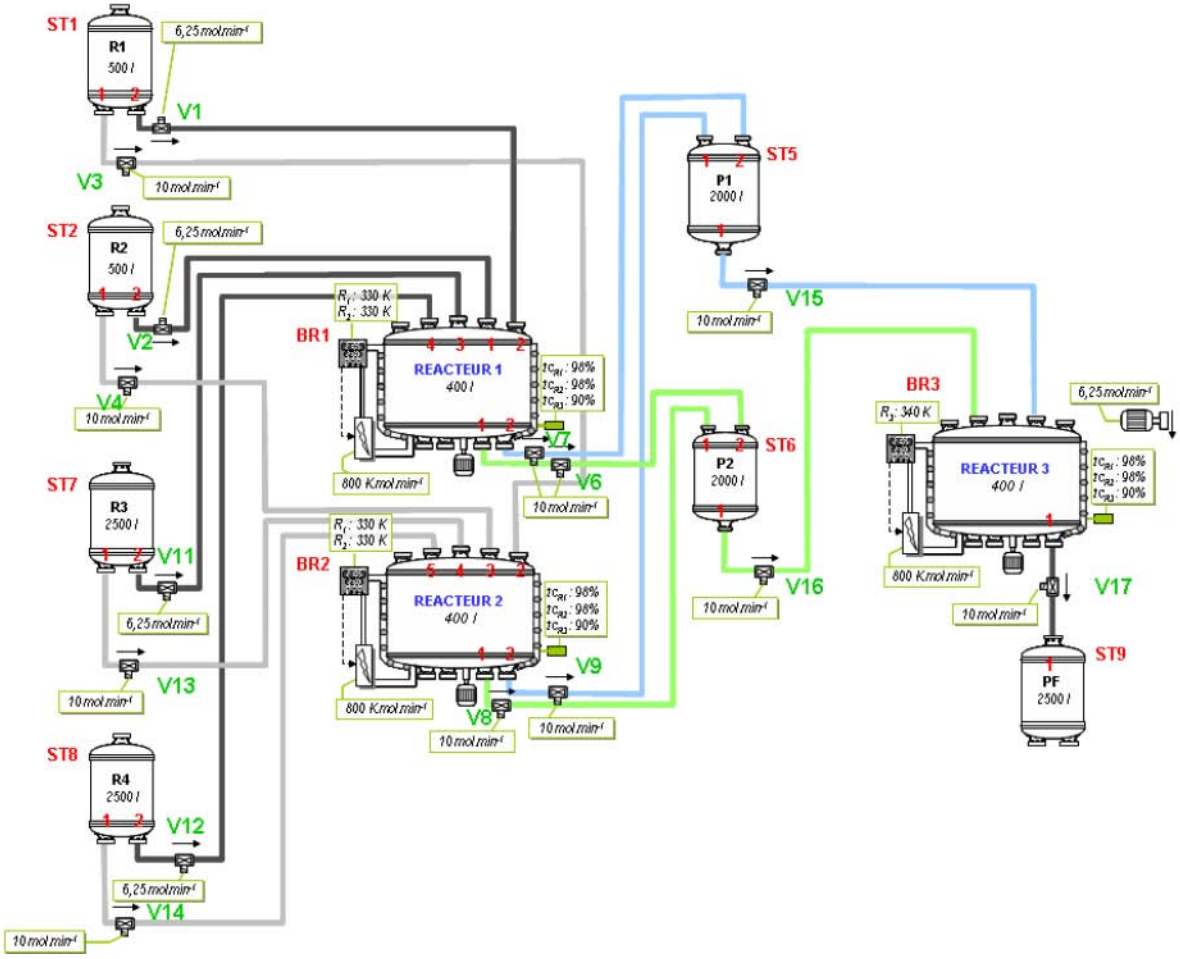


Figure 6.28 Exemple de procédé dérivé de [Kondili et al.1993]

Le procédé mis en œuvre est décrit sur la Figure 6.28. Le but de cette installation est de fabriquer 3 produits P1, P2 et PF. Le produit PF est obtenu à partir des deux produits P1 et P2.

6.4.2. Opération de réaction :

La recette générale des produits P1, P2 et PF à fabriquer est présentée sur la Figure 6.29 sous la forme d'un STN.

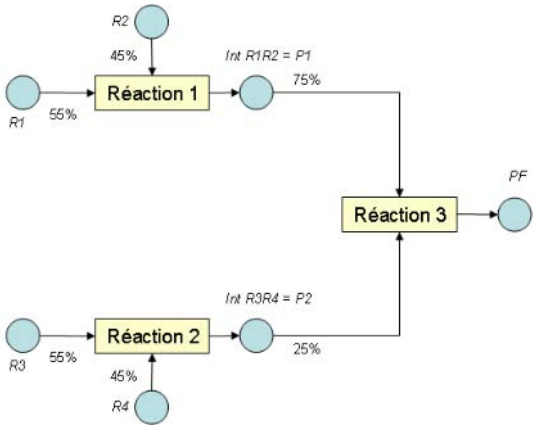


Figure 6.29 Recette de fabrication des produits P1, P2 et PF

6.4.3. Recette principale de l'exemple :

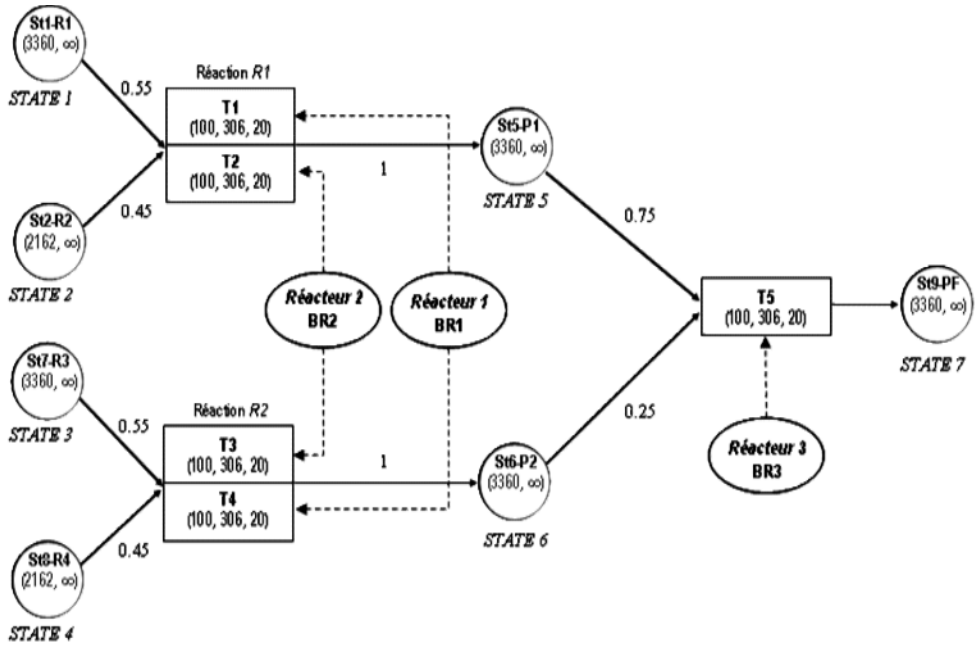


Figure 6.30 Recette principale de l'exemple d'application 2

La mise en œuvre d'un exemple multi-produits avec partage de ressources illustre bien la complexité croissante de la recette principale ; ainsi chaque réaction pouvant être réalisée par des ressources multiple doit être dupliquée.

6.4.4. Identification du « Centre de décision » de l'exemple :

La duplication des tâches réalisées par des ressources multiples se retrouve dans la traduction de la recette principale du procédé en recette de contrôle du simulateur.

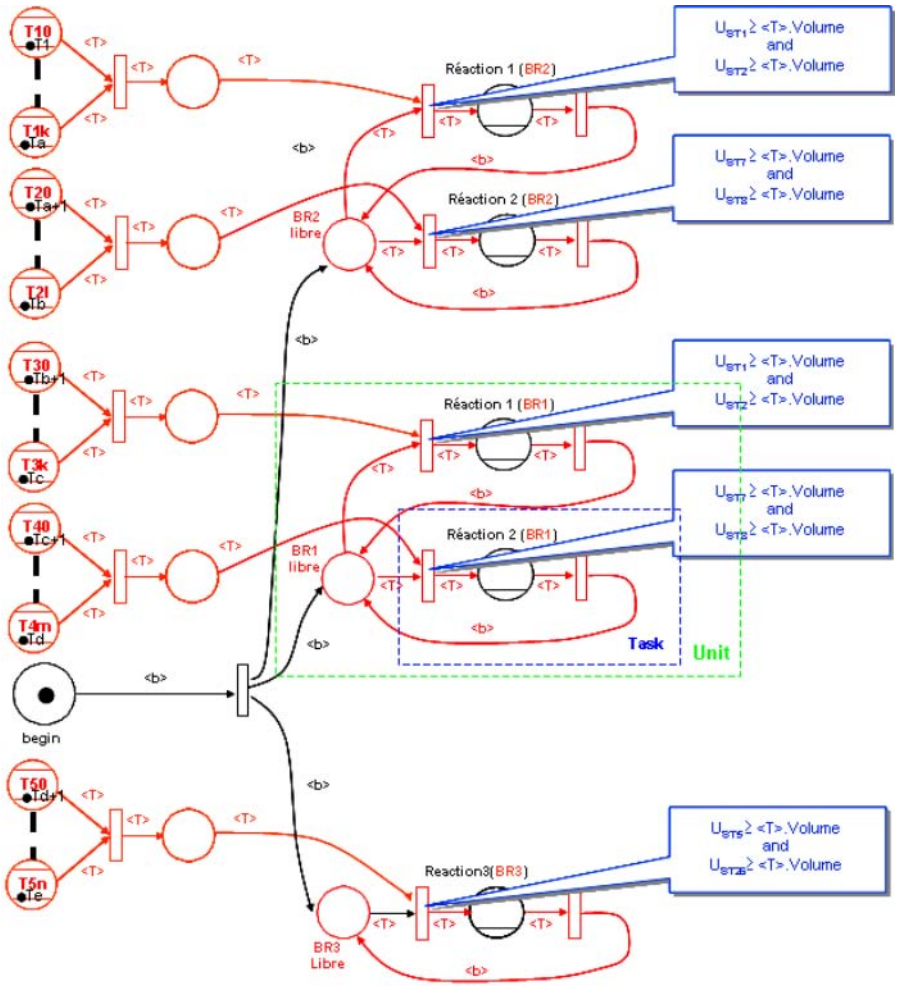


Figure 6.31 Recette et « Centre de décision » du modèle de simulation de l'exemple d'application 2 dérivé de [Kondili et al.1993]

6.4.5. Bilan :

Les résultats du modèle d'ordonnancement de l'exemple d'application 2 permettent de mettre en évidence la complexité croissante du couplage entre ordonnancement et simulation lors de l'analyse de procédés multi-produits. Ainsi, pour un exemple très simplifié d'ordonnacement, le gain de modélisation lié à la mise en œuvre de la gestion des tâches sous PrODHyS est d'un facteur cinq.

Par ailleurs, une analyse par opération unitaire sera nécessaire comme dans l'exemple précédent afin d'ajuster les paramètres du modèle d'ordonnancement.

Solution :
Solution obtenue en 12,3 s (solution sous-optimale).
(XPRESS-MP 1.6.3, processeur INTEL Pentium Centrino 1,76 GHz (1 Go RAM))

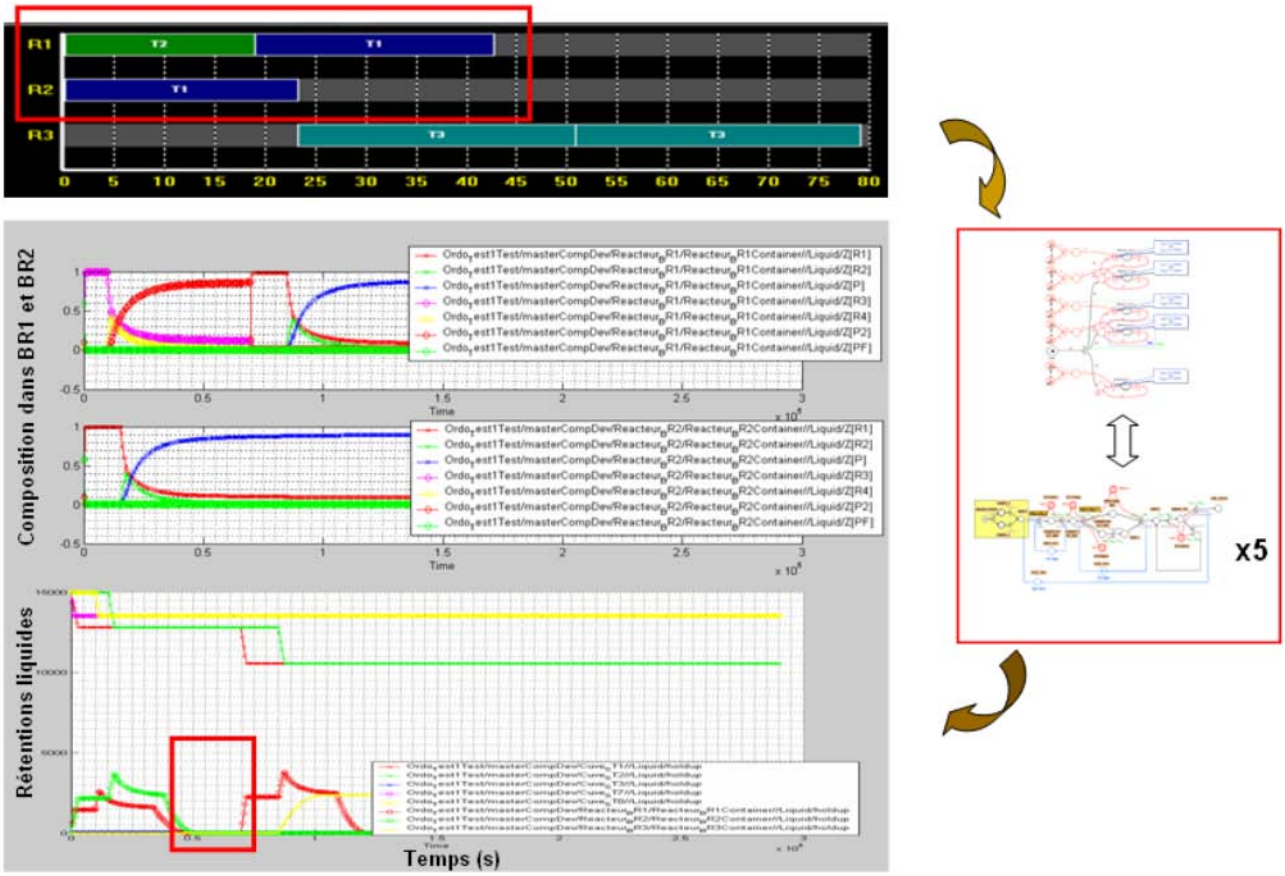


Figure 6.32 Résultats de l'exemple d'application 2

6.5. EXTENSION ET PERSPECTIVES DU COUPLAGE

En Génie des Procédés, les performances effectives de l'unité sont fortement liées à la fonction ordonnancement. Classées dans la catégorie des problèmes *NP-difficiles*, différentes approches ont été proposées pour résoudre le problème d'ordonnancement. Parmi celles-ci, on peut distinguer :

- celles reposant sur la programmation mathématique et l'écriture de modèles algébriques dans lesquels des hypothèses simplificatrices sont souvent introduites afin de pouvoir être résolues avec les méthodes classiques d'optimisation,
- celles reposant sur la simulation dynamique d'un modèle détaillé, fidèle au procédé mais ne permettant qu'une exploration limitée des solutions candidates.

Dans le cadre de nos travaux, nous avons intégré des notions de lots au simulateur dynamique et proposé un modèle d'ordonnancement adapté aux procédés. C'est pourquoi, combiner ces deux types d'approche est apparu comme une voie de recherche à explorer.

Les avancées réalisées dans les techniques de modélisation et les algorithmes de résolution des problèmes d'optimisation, associées à la puissance accrue des calculateurs permettent aujourd'hui de résoudre des problèmes de plus en plus complexes. Néanmoins, comme l'indique [Méndez *et al*, 2006], un

écart existe toujours entre théorie et pratique, soit à cause de la taille des systèmes considérés, soit à cause des hypothèses simplificatrices qu'il faut éventuellement introduire pour rendre le problème solvable (l'une et l'autre étant d'ailleurs souvent liées). Sur ce constat, cette section se propose de présenter les principes sur lesquels repose cette première étape de couplage.

6.5.1. Principe général de l'approche

Pour gérer la complexité due à la taille des systèmes, une voie de recherche consiste à développer des techniques visant à maintenir le nombre de variables de décision des modèles à un niveau raisonnable par l'utilisation de méthodes de décomposition ou d'agrégation [Basset et al, 1997], [Jackson and Grossmann, 2003], [Hétreux et al, 1996].

Pour le problème lié à l'adéquation de la solution avec le procédé réel, de nombreux facteurs induisent que la recherche d'un optimum mathématique sur des modèles trop simplifiés est souvent inutile, voire sans réel sens, dans la pratique. En effet, la mise en œuvre directe de l'ordonnancement obtenu est souvent limitée pour diverses raisons, dont notamment le fait que :

de nombreuses méthodes proposées dans la littérature s'appliquent sous l'hypothèse d'une durée constante et connue des tâches. Or, ceci constitue une restriction sévère face à la sensibilité de certaines opérations unitaires aux réglages des conditions opératoires. La colonne à distiller discontinue est un exemple où la durée opératoire dépend de plusieurs paramètres : la qualité de la charge initiale, la politique de chauffe au bouilleur, la politique de reflux en tête de colonne, les débits de soutirage latéraux, les pertes thermiques, etc. Ne pas inclure ces variables dans la représentation du système impose une approximation de la durée opératoire à laquelle est souvent ajoutée une marge de sécurité qui surestime sa valeur. Par ailleurs, la durée d'une tâche peut aussi dépendre de l'état du système à un instant donné. Par exemple, la durée d'un transfert par gravité est dépendante de la rétention dans la cuve source (effet de la pression hydraulique). De même, la durée de chauffe d'un produit dépend de la température initiale, elle-même pouvant dépendre de la durée d'attente du produit dans la cuve de stockage amont si des pertes thermiques existent.

l'optimalité d'une solution obtenue dans un contexte déterministe peut rapidement être perdue à cause de la nature dynamique du système industriel soumis en permanence aux aléas.

seulement un sous-ensemble des objectifs couramment recherchés par l'ordonnancement est pris en compte. Si le makespan est un des critères souvent évalués, les notions de robustesse du plan, de lissage de charge ou de consommation d'utilité pourraient être intégrées.

Lorsque la garantie d'optimalité n'est plus le critère central, certaines approches proposées consistent à établir une solution initiale a priori « sous-optimale » (par rapport à certains critères) que l'on tente ensuite d'améliorer progressivement et ponctuellement par différentes techniques [Méndez et Cerda, 2003], [Roslöf et al, 2001]. La solution initiale est obtenue dans un temps fixé ou pour un effort de calcul donné, soit par une méthode exacte, soit par une méta-heuristique. D'autres travaux sont fondés sur le couplage d'un algorithme génétique avec un simulateur à événements discrets [Baudet et al, 1995].

Pour notre part, l'outil d'ordonnancement proposé s'appuie sur ce principe d'amélioration à travers le couplage d'un module d'optimisation basé sur une modélisation de type MILP (Mixed Integer Linear Programming) avec un module de simulation dynamique hybride autorisant une modélisation plus fine du procédé. Le Tableau 6.7 récapitule les principaux points forts et points faibles de ces deux outils.

	Ordonnancement basé sur l'optimisation	Ordonnancement basé sur la simulation
Avantages	Exploration exhaustive des solutions candidates Prise en compte globale des contraintes Méthodes efficaces de résolution	Modélisation souvent plus fidèle du procédé réel Durées opératoires évaluées par des modèles phénoménologiques
Inconvénients	Modélisation souvent basée sur des hypothèses simplificatrices ne permettant plus d'exploiter toute la flexibilité de l'atelier Durées opératoires fixées et souvent surestimées	Evaluation d'une solution candidate ⇔ simulation du procédé pour un séquencement et une taille de lot donnés ↳ Exploration difficile et coûteuse en temps CPU Difficultés à prendre en compte les contraintes de date (no-wait, calendrier au conditionnement, politique de nettoyage) car vision partielle de l'horizon temporel

Tableau 6.7 Tableau comparatif des outils d'ordonnancement basés sur des modèles d'optimisation et de simulation

6.5.2. Processus de décision et structure de l'outil d'ordonnancement proposé

L'établissement d'un plan est basé sur un processus itératif entre un module d'optimisation nommé *ProSched* et le simulateur *PrODHyS* dans lequel l'opérateur est intégré (cf. Figure 6.33). Le module d'optimisation établit des plans de production sur la base d'une modélisation MILP (résolution avec le solveur *XPRESS-MP*). Ce premier modèle est construit à partir de durées moyennes estimées. Le séquencement et l'affectation des tâches sur les appareils ainsi que le volume des lots sont alors transmis au simulateur qui génère la recette correspondante et simule le fonctionnement dynamique du procédé. La faisabilité du plan est établie (ou pas) avec transmission en retour de durées opératoires affinées et d'indicateurs de performances (temps total d'exécution, taux d'occupation des ressources, etc.). Cette procédure itérative est effectuée jusqu'à ce que l'opérateur estime la solution satisfaisante. Par ailleurs, la reproductibilité du séquencement des lots sur les équipements entre deux phases d'optimisation peut constituer un indicateur de convergence de la méthode. Le principe fondamental de l'approche est donc de supposer que la solution « approchée » (en terme de comportement) établie par le modèle d'optimisation avec un effort de calcul réduit est compensée par la modélisation plus fine du procédé réalisée au niveau simulation, notamment en prenant en compte des modes de fonctionnement particuliers (chevauchement des transferts, par exemple). Par ailleurs, cette approche devrait globalement rendre plus robuste les plans de production établis et en faciliter l'analyse physico-chimique des phénomènes qui s'y déroulent.

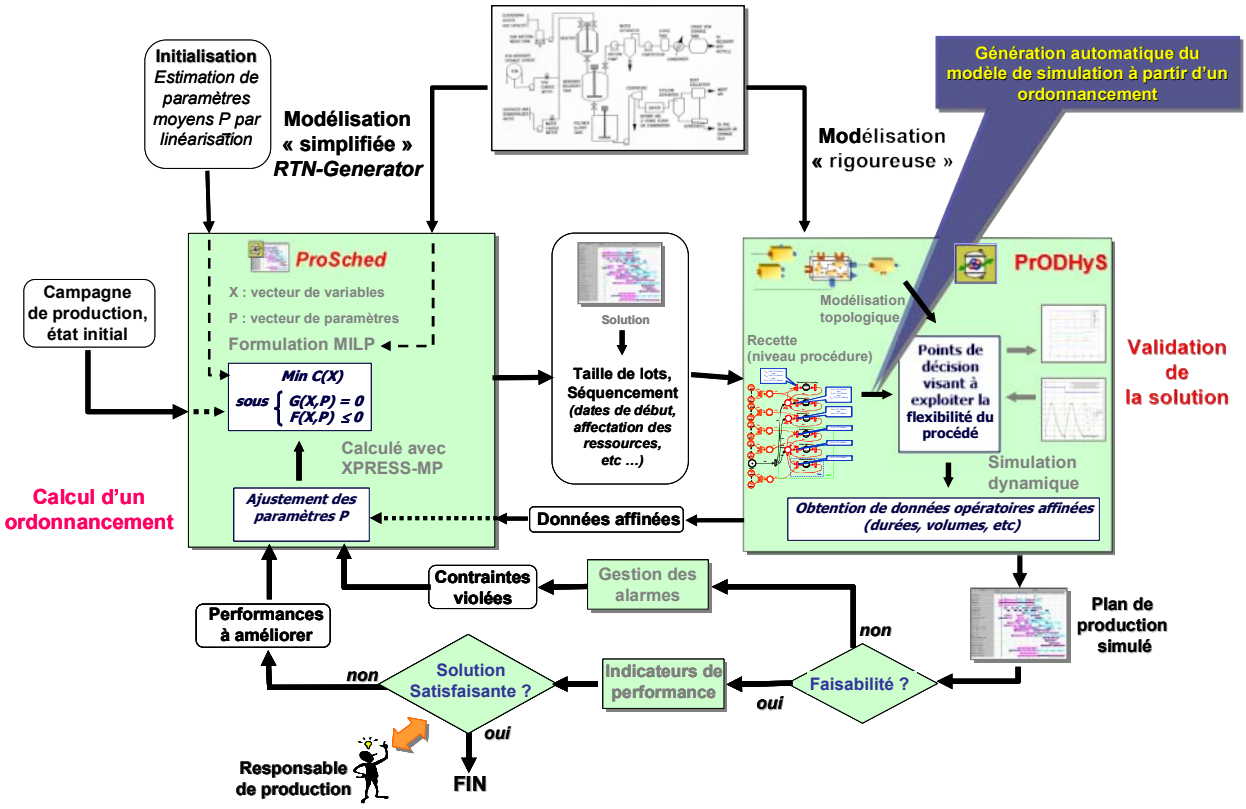


Figure 6.33 Processus de décision et structure de l'outil d'ordonnancement proposé

6.6. CONCLUSION

Ce chapitre a permis de poser les principes d'une approche basée sur le couplage ordonnancement/simulation pour la conduite des procédés batch. Cette approche est basée sur l'utilisation du module *ProSched*, et en particulier du « Centre de Décision » défini dans les chapitres précédents.

Quatre points essentiels du couplage ont été soulevés dans l'exemple traité :

- Les limites d'une approche uniquement basée sur la simulation d'opérations unitaires pour paramétrer le modèle d'ordonnancement.
- L'importance d'obtenir du module de conduite un ordonnancement réaliste plutôt qu'une solution optimale sans degré de liberté qui risquerait d'entraîner des conflits en simulation.
- La nécessité d'une approche itérative mixte couplant l'analyse des opérations unitaires à la simulation globale pour la prise en compte des incertitudes intrinsèques au procédé.
- L'importance de développer un « Centre d'Interprétation » qui ne se limiterait pas à la seule comparaison des dates de lancement prévues et réalisées au niveau du simulateur. Ainsi, la détection anticipée des dérives par rapport au plan de production fixé permettrait d'utiliser ce type d'approche dans le cadre d'ordonnancements réactifs.

Ce chapitre ouvre enfin la perspective d'un outil d'ordonnancement basé sur le couplage optimisation/simulation par la mise en place d'une boucle d'analyse incluant le suivi d'indicateurs de performances et une gestion des alarmes au niveau simulation.

CONCLUSION GENERALE

Conclusion générale

Parmi les outils d'IPAO, la simulation dynamique constitue un des moyens d'analyse privilégié par les ingénieurs de procédé dans leur travail quotidien. L'étude des ateliers discontinus nécessite généralement la simulation de tous les modes de fonctionnement possibles du système (régime permanent et transitoire) ainsi que la réalisation d'un plan de production. Dans ce cadre, une extension au simulateur dynamique hybride *PrODHyS* développé au *LGC* depuis une quinzaine d'années a été proposée. Celle-ci permet de construire automatiquement le scénario de simulation d'un procédé discontinu sur la base d'une recette et d'une liste d'ordres de fabrication. La mise en œuvre de cette fonctionnalité a conduit au couplage du simulateur avec un module d'ordonnancement d'atelier.

Dans ce cadre, trois points essentiels ont été développés dans ces travaux :

- d'abord, concevoir et développer des composants réutilisables qui modélisent le déroulement des opérations unitaires et permettent de décrire de manière systématique les recettes à réaliser par l'assemblage de ces composants,
- ensuite, définir un formalisme de représentation ainsi qu'un modèle mathématique générique d'ordonnancement qui permettent de modéliser les principales caractéristiques d'un procédé discontinu,
- enfin, définir l'interface existant entre le modèle d'optimisation et le modèle de simulation à travers la notion de « centre de décision ».

Les objectifs établis, un premier travail a consisté à caractériser le domaine d'application auquel s'est intéressée cette étude : les *procédés discontinus*. L'intérêt de la simulation dynamique hybride pour l'analyse de ces systèmes a alors été montré. Dans ce contexte, les concepts fondamentaux et les potentialités de la plate-forme *PrODHyS* ont été décrits. Sur cette base, les composants à mettre en place pour étendre ces fonctionnalités ont été identifiés.

Le premier aspect étudié et développé a été le modèle d'ordonnancement du module *ProSched*. Pour cela, un état de l'art des méthodes et outils d'ordonnancement généraux a été établi afin d'étayer le choix de l'approche retenue. Celui-ci a montré que les méthodes de résolution sont aussi variées que les problèmes d'ordonnancement. Néanmoins, trois classes de méthodes ont pu être distinguées :

- les *méthodes exactes* dont fait partie la programmation mathématique,
- les *méthodes approchées*, appelées aussi heuristiques permettant de trouver des solutions proches de l'optimal en un temps raisonnable,
- les méthodes rattachées au domaine de l'*intelligence artificielle*.

La méthode d'optimisation finalement retenue dans ces travaux est la *Programmation Linéaire en Variables Mixtes* (*PLM*). Les principales raisons de ce choix sont la généricité et la flexibilité de la modélisation, la possibilité de s'appuyer sur une représentation graphique clairement établie pour la construction des modèles, ainsi que la présence sur le marché de solveurs de *PLM* relativement efficaces (tels que *XPRESS-MP*, outil utilisé dans le cadre de cette thèse). Toutefois, il existe différentes formulations pour résoudre les problèmes d'ordonnancement d'ateliers discontinus. Ainsi, les principales caractéristiques, les atouts et les limites de ces formulations souvent rencontrées dans la communauté *génie des procédés* ont été étudiés sur la base de *review* récentes trouvées dans la littérature. Les formalismes graphiques de modélisation de ces problèmes ont aussi été examinés. Sur la base des résultats de cette analyse, notre choix s'est porté sur un modèle en temps continu nommé *Unit Spécific Event*. La formulation de base a alors été détaillée et plusieurs extensions ont été proposées compte tenu de nos besoins. Les

principaux points marquants de cette partie du travail s'inscrivent à plusieurs niveaux, tant sur le plan théorique que sur le plan implémentation logicielle :

- tout d'abord, le modèle de base mis en place permet de calculer simultanément le nombre et la taille de chaque lot, leur date de lancement et les ressources utilisées en tenant compte des limitations physiques des appareils et des cuves de stockage,
- l'aspect générique de la modélisation de ces problèmes a permis de s'appuyer sur une représentation graphique bien établie nommée *Resource Task Network* (*RTN*) et ainsi, faciliter la description des problèmes. Chaque instance ne dépendant plus alors que d'un fichier de paramètres, un prototype a été développé (*RTN Generator*) afin de simplifier et d'automatiser la construction de celui-ci. Cet outil permet la saisie des paramètres d'un problème via une interface graphique de type *Drag and Drop* basée sur une représentation *RTN* des recettes. De même, l'outil *GanttChart* a été développé afin d'analyser rapidement les résultats en proposant l'affichage de l'ordonnancement sous forme de diagrammes de Gantt et le suivi des niveaux de stocks.
- des contraintes particulières ont aussi été considérées : l'introduction de paramètres non linéaires (illustrée dans ces travaux par la prise en compte des transferts par gravité dans l'évaluation des durées opératoires dépendantes de la taille des lots) et l'aspect multimodal de certains appareils nécessitant des phases de *set-up* ou d'arrêt spécifiques (tels que les colonnes par exemple). Ceci a conduit à proposer des éléments sémantiques supplémentaires au formalisme *RTN* afin de pouvoir les prendre en compte explicitement.

Le deuxième aspect important de ces travaux a été la conception et le développement du package *Scheduling* du module *ProSched* qui intègre les objets nécessaires à la conduite de la simulation dynamique sous *PrODHyS*. L'apport de cette partie des travaux se situe essentiellement à un niveau structurel :

- l'utilisation des concepts *objet* et le formalisme *Réseau de Petri Différentiel Objet* (*RdPDO*) permettent une intégration sur plusieurs niveaux des réseaux de Petri et des objets. Ceci permet de réduire la taille du modèle, de réduire la complexité du système par une description de haut niveau, d'organiser l'information (autour des jetons notamment), d'améliorer la lisibilité des modèles en favorisant leur modularité (entité génériques et extensibles) et par voie de conséquence, d'en faciliter la réutilisation. Dans ce cadre, l'introduction de jetons objet de type *Task* a permis de découpler les paramètres opératoires statiques et spécifiques à la recette de contrôle (tels que pression et température de réaction, composition à atteindre, etc) et les paramètres dynamiques associés aux tâches à réaliser (tels que nombre et taille des lots, ressource à utiliser, etc).
- de même, l'introduction de la notion de *macro-place paramétrable* dans les *RdPDO* a permis de hiérarchiser la description de l'aspect procédural de la recette, conformément à la structuration proposée par la norme *ISA/SP88*. Cet élément permet de construire la *recette de contrôle* utile à la conduite de la simulation dynamique en restant à une description de niveau *procédure*, la description interne des opérations et des phases étant prédéfinis dans les macro-places.
- ensuite, l'introduction de places dites de *pilotage* dans la recette pour gérer le lancement des lots sur la base des données transmises par le module d'ordonnancement a permis l'identification de la notion de *centre de décision* au niveau du simulateur. Ces derniers peuvent évidemment être exploités dans d'autres situations qui nécessiteraient une prise de décision au niveau de la simulation (capacité à détecter les dérives par rapport au plan de production, analyse d'un nouveau plan, intégration rapide de nouvelles contraintes comme l'indisponibilité d'une ressource, etc.)
- enfin, la dernière partie propose une stratégie d'exploitation du couplage *Optimisation/Simulation* pour la conduite de la simulation dynamique. Néanmoins, l'extension de cette dernière dans un processus itératif pourrait conduire à la mise en place d'un outil de pilotage des procédés discontinus, notamment à un niveau ordonnancement. Les principes de cette approche ont été décrits.

Afin d'illustrer les potentialités de l'outil et de l'approche mise en œuvre, deux exemples complets de procédés ont été traités.

Ces travaux de recherche ouvrent désormais de nombreuses perspectives :

Sur un plan théorique :

- un travail important serait de continuer les développements des interfaces graphiques utilisateurs. En effet, elles faciliteront la saisie des données relatives au procédé étudié (constituants, thermodynamique, topologie, configuration) et la description de la recette à réaliser ; notamment en réutilisant les développements du *RTN-Generator* pour y inclure la partie *RdPDO* de la recette de contrôle ;
- l'idée directrice de ces travaux était d'abord de montrer la faisabilité et les différentes stratégies de couplage d'un simulateur dynamique avec un outil d'ordonnancement. La programmation linéaire a été exploitée ici, mais toute autre approche d'ordonnancement pourrait être candidate. Notamment, la *programmation par contraintes* semble donner des résultats prometteurs et il pourrait être intéressant d'étudier les avantages que pourrait apporter de telles méthodes dans notre procédure de simulation ;
- enfin, un dernier champ d'intérêt important repose sur la manière avec laquelle le système a été utilisé jusqu'à présent. En effet, la plateforme a été utilisée pour concevoir et analyser des opérations unitaires. Les développements réalisés permettent d'envisager l'analyse dynamique de systèmes de production complets et donc la réalisation d'analyses de performances sur l'ensemble d'une unité de fabrication.

Sur un plan pratique :

- une utilisation comme outil d'analyse hors-ligne de la plateforme développée peut être envisagée. Cela nécessiterait de concevoir des règles pour identifier / analyser / corriger de manière systématique les dérives entre ordonnancement et simulation. Ces règles pourraient être définies lors du franchissement d'un jeton tâche ;
- enfin, une utilisation en-ligne comme outil d'aide à la décision serait envisageable., en couplant les travaux développés à la détection de faute pour anticiper des situations de blocage et mettre en œuvre les tests des différents scénarios de reprises. Cela nécessiterait de compléter le centre de décision/centre interprétation au niveau du simulateur pour supporter les opérateurs dans les prises de décisions lors de la conduite des procédés

ANNEXES

Annexe A : DÉFINITION FORMELLE DES RÉSEAU DE PETRI DIFFÉRENTIEL OBJET

Formellement, un *réseau de Petri différentiel à objets* initialement marqué est un n-uplet :

$$N = \langle C, V, P, T, A, A_{AC}, Pre, Post, X, F, A_T, M_0 \rangle \text{ où :}$$

$C = \{C_1, C_2, \ldots, C_{n_c}\}$ *est un ensemble fini de classes d'objets (types) ;* $card(C) = n_C$. *De plus, chaque élément* C_k *de l'ensemble* C *est un doublet* $\langle Att_{C_k}, Meth_{C_k} \rangle$ *où :*

- Att_{C_k} est un ensemble fini d'attributs de la classe C_k ;
- $Meth_{C_k}$ est un ensemble fini de méthodes de la classe C_k.

$V = \{V_1, V_2, \ldots, V_{n_V}\}$ *est un ensemble fini de variables formelles typées par des éléments de* C *;* $card(V) = n_V$.

$P = \{P_1, P_2, \ldots, P_{n_P}\}$ *est un ensemble fini de places ;* $card(P) = n_P$. *L'ensemble* P *peut être décomposé en deux sous-ensembles disjoints* P_H *et* P_D *où* P_H *est un ensemble fini de places différentielles et* P_D, *un ensemble fini de places discrètes.*

$T = \{T_1, T_2, \ldots, T_{n_T}\}$ *est un ensemble fini de transitions ;* $card(T) = n_T$.

$A = \{A_1, A_2, \ldots, A_{n_A}\}$ *est un ensemble fini d'arcs ;* $card(A) = n_A$.

$A_{AC} : A \rightarrow V^{n_V}$ *est une application qui associe à chaque arc, une somme formelle de n-uplets d'éléments de* V.

$Pre : P \times T \rightarrow V^{n_V}$ *est l'application d'incidence avant qui définit les arcs entrants des transitions.*

$Post : P \times T \rightarrow V^{n_V}$ *est l'application d'incidence arrière qui définit les arcs sortants des transitions.*

$X = \{X_1, X_2, \ldots, X_{n_X}\}$ *est un ensemble fini de variables ;* $card(X) = n_X$. *L'ensemble* X *peut être décomposé en sous-ensembles non nécessairement disjoints* X_{P_k} *de variables associées aux places différentielles* P_k. *De plus, chaque ensemble* X_{P_k} *peut être décomposé en sous-ensembles non nécessairement disjoints* $X_{P_k}^G$, $X_{P_k}^L$, $X_{P_k}^F$ *tels que :*

$$X_{P_k} = X_{P_k}^G \cup X_{P_k}^L \cup X_{P_k}^F \text{ où :}$$

- $X_{P_k}^G$ est l'ensemble des variables globales au réseau de Petri tel que :

  $\forall\ i = 1, card(P),\ \forall\ j = 1, card(P)\ :\ X_{P_i}^G = X_{P_j}^G = X^G$

- $X^{L}_{P_k}$ est l'ensemble des variables locales à la place P_k ;
- $X^{F}_{P_k}$ est l'ensemble des variables d'état définies comme attributs des variables formelles V_{P_k} , où $V_{P_k} = Post\ (P_k, \bullet)$.

$F = \{F_1, F_2, \ldots, F_{n_F}\}$ *est un ensemble fini de fonctions ;* $\text{card}\ (F) = n_F$. *L'ensemble* F *peut être décomposé en sous-ensemble disjoints* F_{P_k} *de fonctions associées aux places différentielles* P_k *:*

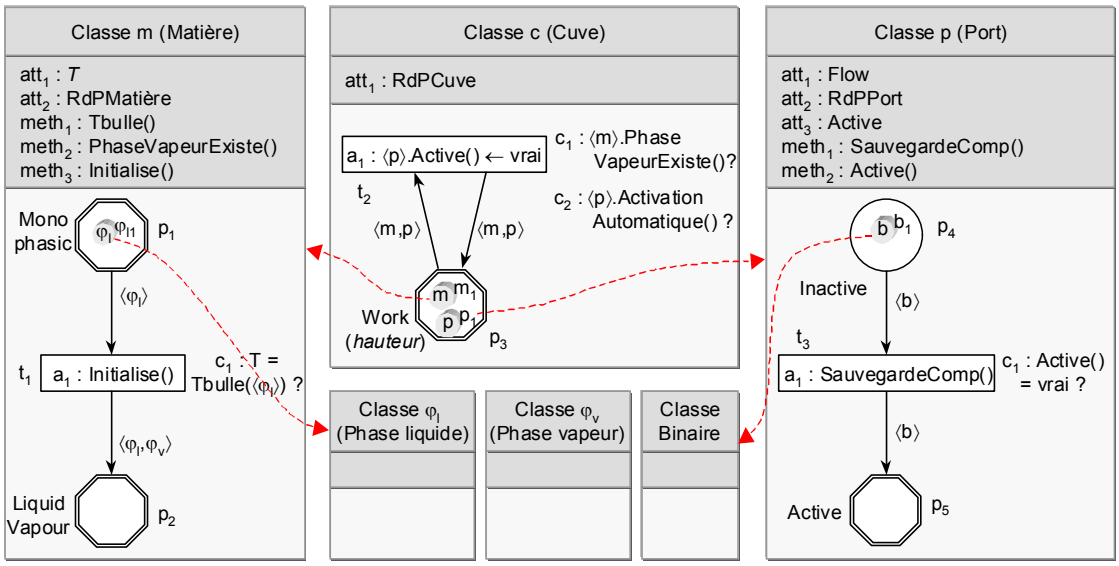
$$\left| \begin{array}{l} F(\dot{X}, X, p, \theta) = \begin{pmatrix} F_{P_1}(\dot{X}_{P_1}, X_{P_1}, p_{P_1}, \theta) \\ \vdots \\ F_{P_{n_{P_H}}}(\dot{X}_{P_{n_{P_H}}}, X_{P_{n_{P_H}}}, p_{P_{n_{P_H}}}, \theta) \end{pmatrix} \text{ avec} \\ F_{P_k}(\dot{X}_{P_k}, X_{P_k}, p_{P_k}, \theta) = \begin{bmatrix} f_{P_k 1}(\dot{X}_{P_k}, X_{P_k}, p_{P_k}, \theta) \\ \vdots \\ f_{P_k n_f}(\dot{X}_{P_k}, X_{P_k}, p_{P_k}, \theta) \end{bmatrix} \end{array} \right.$$

A_T *est l'annotation de* $N : A_T = \langle Meth^G , Att^G , Meth^{T_j}_{in} , Att^{T_j}_{in} , A_{TE}, A_{TA} \rangle$ *où :*

- $Meth^G$ est un ensemble fini de méthodes globales au réseau de Petri.
- Att^G est un ensemble fini d'attributs globaux au réseau de Petri.
- $Meth^{T_j}_{in}$ est l'union des méthodes des classes associées aux variables formelles de l'ensemble $V^{T_j}_{in}$ où $V^{T_j}_{in} = Pre\ (\bullet, T_j)$.
- $Att^{T_j}_{in}$ est l'union des attributs des classes associées aux variables formelles de l'ensemble $V^{T_j}_{in}$.
- $A_{TE} : T \rightarrow Att^{T_j}_{in} \times Meth^{T_j}_{in} \times Att^G \times Meth^G$ est une application qui associe à une transition, un événement sous la forme d'une conjonction de conditions en utilisant les éléments des ensembles $Meth^{T_j}_{in}$, $Meth_G$, $Att^{T_j}_{in}$ et Att^G ;
- $A_{TA} : T \rightarrow Att^{T_j}_{in} \times Meth^{T_j}_{in} \times Att^G \times Meth^G$ est une application qui associe à une transition, une action sous la forme d'une suite de traitements réalisés par les méthodes des ensembles $Meth^{T_j}_{in}$ et $Meth^G$ et appliqués sur les éléments des ensembles $Att^{T_j}_{in}$ et Att^G .

M_0 *est le marquage initial.*

Afin d'illustrer cette définition formelle, les ensembles définis ci-dessus sont identifiés au travers d'un exemple. Notons simplement qu'il comporte trois classes, chacune possédant un réseau de Petri spécifique.



Exemple illustratif

$C = \{C_1, C_2, \ldots, C_{n_c}\}$ *est un ensemble fini de classes d'objets (types) ;* $card(C) = n_C$. *De plus, chaque élément* C_k *de l'ensemble* C *est un doublet* $\langle Att_{C_k}, Meth_{C_k} \rangle$ *où :*

- Att_{C_k} est un ensemble fini d'attributs de la classe C_k ;

> *L'ensemble* C *associé au RdP de la cuve est le suivant :* $C = \{m, p, c\}$
> *L'ensemble des attributs* Att_{C_k} *de la classe* m *est le suivant :* $Att_{C_k} = \{T, RdPMatière\}$;

- $Meth_{C_k}$ est un ensemble fini de méthodes de la classe C_k.

> *L'ensemble des méthodes* $Meth_{C_k}$ *de la classe* m *est le suivant :*
> $Meth_{C_k} = \{Tbulle(), PhaseVapeurExiste(), Initialise()\}$.

$V = \{V_1, V_2, \ldots, V_{n_V}\}$ *est un ensemble fini de variables formelles typées par des éléments de* C *;* $card(V) = n_V$.

> *L'ensemble* V *associé au RdP de la cuve est le suivant :*
> $V = \{\langle m \rangle, \langle p \rangle\}$;
> $\langle m \rangle$ *est de type* m ;
> $\langle p \rangle$ *est de type* p.

$P = \{P_1, P_2, \ldots, P_{n_P}\}$ *est un ensemble fini de places ;* $card(P) = n_P$. *L'ensemble* P *peut être décomposé en deux sous-ensembles disjoints* P_H *et* P_D *où* P_H *est un ensemble fini de places différentielles et* P_D, *un ensemble fini de places discrètes.*

> *Les ensembles* P, P_D *et* P_H *associés au réseau de Petri du port sont les suivants :*
> $P = \{p_4, p_5\}$;
> $P_D = \{p_4\}$;
> $P_H = \{p_5\}$.

$T = \{T_1, T_2, \ldots, T_{n_T}\}$ *est un ensemble fini de transitions ; card(T) =* n_T.

> *L'ensemble* T *associé au réseau de Petri du port est le suivant :* $T = \{t_3\}$.

$A = \{A_1, A_2, \ldots, A_{n_A}\}$ *est un ensemble fini d'arcs ;* $\text{card}(A) = n_A$.

> *L'ensemble* A *associé au réseau de Petri du port est le suivant :*
> $A = \{a(p_4,t_3), a(t_3,p_5)\}$.

$A_{AC} : A \rightarrow V^{n_V}$ *est une application qui associe à chaque arc, une somme formelle de n-uplets d'éléments de* V.

> *Les n-uplets de variables formelles associés aux arcs des réseaux de Petri de l'exemple sont les suivants :*
> $a(p_1, t_1) \rightarrow \langle \varphi_l \rangle$;
> $a(t_1, p_2) \rightarrow \langle \varphi_l, \varphi_v \rangle$;
> $a(p_3, t_2) \rightarrow \langle m,p \rangle$;
> $a(t_2, p_3) \rightarrow \langle m,p \rangle$;
> $a(p_4, t_3) \rightarrow \langle b \rangle$;
> $a(t_3, p_5) \rightarrow \langle b \rangle$.

$Pre : P \times T \rightarrow V^{n_V}$ *est l'application d'incidence avant qui définit les arcs entrants des transitions.*

> *La matrice* Pre *associée au réseau de Petri de la matière est la suivante :*
>
> $$Pre : \begin{array}{c} \\ p_1 \\ p_2 \end{array} \begin{array}{c} t_1 \\ \begin{pmatrix} \langle \varphi_l \rangle \\ 0 \end{pmatrix} \end{array}$$

$Post : P \times T \rightarrow V^{n_V}$ *est l'application d'incidence arrière qui définit les arcs sortants des transitions.*

> *La matrice* $Post$ *associée au réseau de Petri de la matière est la suivante :*
>
> $$Post : \begin{array}{c} \\ p_1 \\ p_2 \end{array} \begin{array}{c} t_1 \\ \begin{pmatrix} 0 \\ \langle \varphi_l, \varphi_v \rangle \end{pmatrix} \end{array}$$

$X = \{X_1, X_2, \ldots, X_{n_X}\}$ *est un ensemble fini de variables ;* $\text{card}(X) = n_X$. *L'ensemble* X *peut être décomposé en sous-ensembles non nécessairement disjoints* X_{P_k} *de variables associées aux places différentielles* P_k. *De plus, chaque ensemble* X_{P_k} *peut être décomposé en sous-ensembles non nécessairement disjoints* $X_{P_k}^G$, $X_{P_k}^L$, $X_{P_k}^F$ *tels que :*

> $X_{P_k} = X_{P_k}^G \cup X_{P_k}^L \cup X_{P_k}^F$ où :

- $X_{P_k}^G$ est l'ensemble des variables globales au réseau de Petri tel que :
  $\forall\ i = 1, card(P),\ \forall\ j = 1, card(P)\ :\ X_{P_i}^G = X_{P_j}^G = X^G$
- $X_{P_k}^L$ est l'ensemble des variables locales à la place P_k ;
- $X_{P_k}^F$ est l'ensemble des variables d'état définies comme attributs des variables formelles V_{P_k}, où $V_{P_k} = Post\ (P_k, \bullet)$.

> *L'ensembles des variables* X *associé au RdP de la cuve est le suivant :*

X = {Flow,T,hauteur} ;

Les ensembles des variables x_{p_3}, $x^G_{p_3}$, $x^L_{p_3}$ *et* $x^F_{p_3}$ *associés à la place* p_3 *sont les suivants :*

x_{p_3} = {Flow,T,hauteur} ; $x^G_{p_3} = \varnothing$; $x^L_{p_3}$ = {hauteur} ; $x^F_{p_3}$ = {Flow,T}.

$F = \{F_1, F_2, \ldots, F_{n_F}\}$ *est un ensemble fini de fonctions ;* card (F) = n_F. *L'ensemble* F *peut être décomposé en sous-ensemble disjoints* F_{P_k} *de fonctions associées aux places différentielles* P_k *:*

$$F(\dot{X},X,p,\theta) = \begin{pmatrix} F_{P_1}(\dot{X}_{P_1},X_{P_1},p_{P_1},\theta) \\ \vdots \\ F_{P_{n_{P_H}}}(\dot{X}_{P_{n_{P_H}}},X_{P_{n_{P_H}}},p_{P_{n_{P_H}}},\theta) \end{pmatrix} \text{ avec}$$

$$F_{P_k}(\dot{X}_{P_k},X_{P_k},p_{P_k},\theta) = \begin{bmatrix} f_{P_k1}(\dot{X}_{P_k},X_{P_k},p_{P_k},\theta) \\ \vdots \\ f_{P_kn_f}(\dot{X}_{P_k},X_{P_k},p_{P_k},\theta) \end{bmatrix}$$

Les ensembles de fonctions définis au niveau des places p_1 *et* p_2 *du réseau de Petri de la matière sont les suivants :*

$$F_{P_1}(\dot{X}_{p_1},X_{p_1},p_{p_1},\theta) = \varnothing$$

$$F_{p_2}(\dot{X}_{p_2},X_{p_2},p_{p_2},\theta) = \begin{pmatrix} y_i - K_i.x_i = 0 \\ \vdots \\ y_{n_c} - K_{n_c}.x_{n_c} = 0 \\ \sum_{i=1}^{n_c}(y_i - x_i) = 0 \\ K_i - mK_i.(T,P,x,y) = 0 \\ \vdots \\ K_{n_c} - mK_{n_c}.(T,P,x,y) = 0 \end{pmatrix}$$

A_T *est l'annotation de* N : $A_T = \langle Meth^G , Att^G , Meth^{T_j}_{in} , Att^{T_j}_{in} , A_{TE}, A_{TA}\rangle$ *où :*

- $Meth^G$ est un ensemble fini de méthodes globales au réseau de Petri.

  L'ensemble des méthodes globales $Meth_G$ *associé au réseau de Petri de la matière est le suivant :*

  $Meth_G$ = {Tbulle() ; PhaseVapeurExiste() ; Initialise()}.

- Att^G est un ensemble fini d'attributs globaux au réseau de Petri.

  L'ensemble des attributs globaux Att_G *associé au réseau de Petri de la matière est le suivant :*

  Att_G = {T ; RdPMatière }.

- $Meth^{T_j}_{in}$ est l'union des méthodes des classes associées aux variables formelles de l'ensemble $V^{T_j}_{in}$ où $V^{T_j}_{in} = Pre(\bullet, T_j)$.

  L'ensemble $Meth^{t_2}_{in}$ *associé au RdP de la cuve est le suivant :*

> $Meth_{in}^{t_2}$ = *{Tbulle() ; PhaseVapeurExiste() ; Initialise() ; SauvegardeComp() ; Active()}*.

- $Att_{in}^{T_j}$ est l'union des attributs des classes associées aux variables formelles de l'ensemble $V_{in}^{T_j}$.

> *L'ensemble* $Att_{in}^{t_2}$ *associé au RdP de la cuve est le suivant :*
> $Att_{in}^{t_2}$ = *{T ; RdPMatière ; Flow ; RdPPort ; Active}*.

- $A_{TE} : T \rightarrow Att_{in}^{T_j} \times Meth_{in}^{T_j} \times Att^G \times Meth^G$ est une application qui associe à une transition, un événement sous la forme d'une conjonction de conditions en utilisant les éléments des ensembles $Meth_{in}^{T_j}$, $Meth_G$, $Att_{in}^{T_j}$ et Att^G ;

> *La condition* c_1 *associée à la transition* t_2 *de la cuve utilise, par exemple, la méthode PhaseVapeurExiste() de l'ensemble* $Meth_{in}^{t_2}$ *associé au RdP de la cuve.*
>
> *De même, la condition* c_1 *associée à la transition* t_1 *du système de phases utilise, par exemple, la méthode Tbulle() de l'ensemble* $Meth_G$ *associé au RdP de la matière.*

- $A_{TA} : T \rightarrow Att_{in}^{T_j} \times Meth_{in}^{T_j} \times Att^G \times Meth^G$ est une application qui associe à une transition, une action sous la forme d'une suite de traitements réalisés par les méthodes des ensembles $Meth_{in}^{T_j}$ et $Meth^G$ et appliqués sur les éléments des ensembles $Att_{in}^{T_j}$ et Att^G.

> *L'action* a_1 *associée à la transition* t_2 *de la cuve utilise, par exemple, la méthode Active() de l'ensemble* $Meth_{in}^{t_2}$ *associé au RdP de la cuve.*
>
> *De même, l'action* a_1 *associée à la transition* t_1 *du système de phases utilise, par exemple, la méthode Initialise() de l'ensemble* $Meth^G$ *associé au RdP de la matière.*

M_0 *est le marquage initial.*

> *Les marquages initiaux des RdP sont les suivants :*
> *Réseau de Petri de la matière :* $M_0 = \begin{pmatrix} \varphi_{l1} \\ 0 \end{pmatrix}$;
>
> *Réseau de Petri du port :* $M_0 = \begin{pmatrix} b_1 \\ 0 \end{pmatrix}$;
>
> *Réseau de Petri de la cuve :* $M_0 = (p_1 + m_1)$.

Annexe B : MODÈLES MATHÉMATIQUES DE BASES POUR L'ORDONNANCEMENT DES PROCÉDÉS BATCH

Exemple de formulation en temps discret : « Global time intervals » STN-formulation

Contraintes d'allocation :

$$\sum_{i \in I_j} \sum_{\substack{t'=t-pt_{ij}+1 \\ t>0}}^{t} W_{i,j,t'} \leq 1 \qquad \forall j \in J, \forall t \in 1,..,T \qquad (1.1)$$

A un instant *t*, une ressource *j* peut lancer au plus une opération. De plus si une opération (tâche *i*) est lancée en période *t* (*Wi,j,t=1*) alors la ressource *j* ne sera pas disponible (*Wi,j,t'=0*) entre les périodes *t'*= *t* et *t'*= *t* + *pi –1* (durée de la tâche). Cette contrainte est définie dans l'équation (1.1) :

Contraintes de capacité

$$W_{i,j,t} V_{ij}^{\min} \leq B_{i,j,t} \leq W_{i,j,t} V_{ij}^{\max} \qquad \forall i \in I, \forall j \in J, \forall t \in 1,..,T \qquad (1.2)$$

$$C_s^{\min} \leq S_{s,t} \leq C_s^{\max} \qquad \forall s \in S, \forall t \in 1,..,T \qquad (1.3)$$

Si la ressource n'est pas disponible en période *t* alors la valeur de son batch est nulle, si par contre le batch est lancé en période *t* alors la taille du lot est contrainte par les bornes V_{ij}^{min} et V_{ij}^{max}. (équation 1.2).

Les stockages intermédiaires sont aussi bornés (taille des cuves) (équation 1.3).

Bilan matière sur un état

$$S_{s,t} = S_{s,t-1} + \sum_{i \in I_s^p} \rho_{i,s}^{prod} \sum_{j \in J_i} B_{i,j,t-pt_{ij}} - \sum_{i \in I_s^c} \rho_{i,s}^{cons} \sum_{j \in J_i} B_{i,j,t} + \Pi_{st} - D_{st}$$

$$\forall s \in S_{mat}, \forall t \in 1,..,T \qquad (1.4)$$

Le bilan matière à un instant *t*, comprend :

- $S_{s,t}$ (resp. $S_{s,t-1}$) la quantité de l'*état s* à en période *t* (resp. *t-1*)

- $\sum_{i \in I_s^c} \rho_{i,s}^{cons} \sum_{j \in J_i} B_{i,j,t}$ les quantités de *S* consommées en début de période *t* par les tâches consommatrices,
- $\sum_{i \in I_s^p} \rho_{i,s}^{prod} \sum_{j \in J_i} B_{i,j,t-pt_{ij}}$ les quantités d'état *S* libérées en fin de période *t-1* par les tâches productrices
- Π_{st} la quantité d'*état s* reçue de l'environnement en période *t*
- D_{st} la quantité d'*état s* fournie à l'environnement en période *t* (exemple : demande en *état s* à satisfaire tout au long de l'horizon temporel et non pas seulement en fin d'horizon)

Fonction objectif

La fonction objectif peut être la minimisation de la durée de campagne, on peut ajouter à cela la minimisation des en-cours en définissant des coûts de stockage.

$$MS \geq W_{i,j,t}(t + pt_{ij} - 1) \qquad \forall i \in I, \forall j \in J, \forall t \in 1,..,T \qquad (1.5)$$

$$Min(MS) \qquad (1.6)$$

Avec :

$B_{i,j,t}$ la taille du lot de la *tâche i* sur la *ressource j* lancé en période *t,* la taille du lot est contrainte par les bornes V_{ij}^{min} et V_{ij}^{max}

$\rho_{i,s}^{cons}$ *et* $\rho_{i,s}^{prod}$ les proportions de l'*état s* consommées et produites par l'ensemble des tâches productrices / consommatrices de *s*

$S_{s,t}$ la quantité de l'*état s* en période *t* contrainte par les bornes C_s^{min} et C_s^{max}

Wi,j,t la variable binaire modélisant le lancement d'un lot (voir les contraintes d'allocation)

MS la variable définie pour le calcul du makespan

pt_{ij} représente la durée de la tâche *i* sur la ressource *j*

Exemple de formulation en temps continu : « Global time points » RTN-formulation

Cette formulation a été développée ces dernières années, elle est basée sur la définition d'une grille commune qui est variable et valable pour l'ensemble des ressources partagées. Cette définition entraîne la notion de *time points n* dont la date d'occurrence est une valeur *Tn* (n=[1, N] où N représente l'ensemble des *time points*) non fixée a priori. Le modèle impose que l'ensemble des tâches démarrant à l'évènement *n* soit synchronisé à la date *Tn*.

Contraintes d'allocation :

Les contraintes (2.1) et (2.2) imposent qu'au plus une tâche *i* puisse être lancée ($Ws_{i,n}$ =1) ou arrêtée ($Wf_{i,n}$ =1) sur l'équipement *j* à l'évènement *n* correspondant. La contrainte (2.3) garantit l'arrêt des tâches lancées. La contrainte (2.4) impose qu'au plus une tâche puisse être lancée sur la ressource *j* pour tout évènement *n*.

$$\sum_{i \in I_J} Ws_{i,n} \leq 1 \qquad \forall i \in I, \forall n \in N \qquad (2.1)$$

$$\sum_{i \in I_J} Wf_{i,n} \leq 1 \qquad \forall i \in I, \forall n \in N \qquad (2.2)$$

$$\sum_{n} Ws_{i,n} = \sum_{n} Wf_{i,n} \qquad \forall i \in I \qquad (2.3)$$

$$\sum_{i \in I_j} \sum_{n'<n} (Ws_{i,n'} - Wf_{i,n'}) \leq 1 \qquad \forall i \in I, \forall n, n' \in N \qquad (2.4)$$

Contraintes de capacité

Des tailles minimales (V_i^{min}) et maximales (V_i^{max}) de lots sont imposées en début et en fin de tâches par les contraintes (2.5) et (2.6). De plus, la taille du lot est aussi imposée pour chaque tâche active par la contrainte (2.7). La contrainte (2.7) impose une taille de lot constante durant la réalisation d'une tâche. Les stockages intermédiaires sont aussi bornés (taille des cuves) (contrainte 2.9)

$$Ws_{i,n} V_i^{\min} \leq Bs_{i,n} \leq Ws_{i,n} V_i^{\max} \qquad \forall i \in I, \forall n \in N \qquad (2.5)$$

$$Wf_{i,n} V_i^{\min} \leq Bf_{i,n} \leq Wf_{i,n} V_i^{\max} \qquad \forall i \in I, \forall n \in N \qquad (2.6)$$

$$V_i^{\min} \left(\sum_{n'<n} Ws_{i,n'} - \sum_{n' \leq n} Wf_{i,n'}\right) \leq Bp_{i,n} \leq V_i^{\max} \left(\sum_{n'<n} Ws_{i,n'} - \sum_{n' \leq n} Wf_{i,n'}\right)$$

$$\forall i \in I, \forall n \in N \qquad (2.7)$$

$$Bs_{i,n-1} + Bp_{i,n-1} = Bp_{i,n} + Bf_{i,n} \qquad \forall i \in I, \forall n \in N \qquad (2.8)$$

$$C_s^{\min} \leq S_{s,n} \leq C_s^{\max} \qquad \forall s \in S, \forall n \in N \qquad (2.9)$$

Bilan matière sur un état

Le bilan est équivalent au bilan en temps discret, à la différence qu'il est réalisé à la date *Tn* correspondant au *time point n*.

$$S_{s,n} = S_{s,n-1} + \sum_{i \in I_s^p} \rho_{i,s}^{prod} Bs_{i,n} - \sum_{i \in I_s^c} \rho_{i,s}^{cons} Bs_{i,n} + \Pi_{sn} - D_{sn}$$

$$\forall s \in S_{mat}, \forall n \in N \qquad (2.10)$$

L'annulation des valeurs $S_{s,n}$ permet d'annuler un stockage intermédiaire.

Contraintes de temps et de séquence

Le premier *time point* correspond au démarrage soit à $T_1=0$ et le dernier à $Tn=H$ qui correspond à la durée de l'horizon de temps choisi ; tandis que l'ordre croissant des time points est imposé par la contrainte (2.11). De plus, la date de fin d'une tâche *i* qui a été lancée au *time point n* est imposée par les contraintes (2.12) et (2.13). On peut noter ici que les temps de traitement sont dépendants des tailles de lots.

$$T_{n+1} \geq T_n \qquad \forall n \in N \qquad (2.11)$$

$$Tf_{i,n+1} \leq T_n + \alpha_i Ws_{i,n} + \beta_i Bs_{i,n} + H(1 - Ws_{i,n}) \qquad \forall i \in I, \forall n \in N \qquad (2.12)$$

$$Tf_{i,n+1} \geq T_n + \alpha_i Ws_{i,n} + \beta_i Bs_{i,n} - H(1 - Ws_{i,n}) \qquad \forall i \in I, \forall n \in N \qquad (2.13)$$

La contrainte (2.14) impose qu'une tâche *i* conserve sa date de fin depuis son lancement et jusqu'à sa prochaine occurrence ($Ws_{i,n}=1$). La contrainte (2.17) impose que la date de fin d'une tâche *i* finissant au *time point n* doit être inférieure ou égale à la date à laquelle le *time point n* est positionné.

$$Tf_{i,n} - Tf_{i,n-1} \leq H\, Ws_{i,n} \qquad \forall i \in I, \forall n \in N \,|\, n > 1 \qquad (2.14)$$

$$Tf_{i,n} \leq T_n + H(1 - Wf_{i,n}) \qquad \forall i \in I, \forall n \in N \,|\, n > 1 \qquad (2.15)$$

Fonction objectif

La fonction objectif peut être la minimisation de la durée de campagne, on peut ajouter à cela la minimisation des en-cours en définissant des coûts de stockage.

$$MS \geq Tf_{i,n} \qquad \forall i \in I, \forall n \in N \qquad (2.16)$$

$$Min(MS) \qquad (2.17)$$

Avec :

$Bs_{i,n}, Bf_{i,n}$ la taille du lot de la *tâche i* sur la *ressource j* lancé à l'évènement *n*. La taille du lot est contrainte par les bornes V_i^{min} et V_i^{max}.

$\rho_{i,s}^{cons}$ *et* $\rho_{i,s}^{prod}$ les proportions de l'*état s* consommées et produites par l'ensemble des tâches productrices / consommatrices de *s*

$S_{s,n}$ la quantité de l'*état s* à l'évènement *n* contrainte par les bornes C_s^{min} et C_s^{max}

Wsi,n,(resp. *Wfi,n*) la variable binaire modélisant le lancement (resp. la fin) d'un lot (voir les contraintes d'allocation)

MS la variable définie pour le calcul du makespan

NOMENCLATURE

Nomenclature

Lettres romaines

a	poids associé à la durée du plan dans le critère
$B_{i,j,n}$	taille du lot de la tâche i sur la ressource j lancé à l'évènement n
$B_{i,j,t}$	taille du lot de la tâche i sur la ressource j lancé en période t
$B_{i,n}$	quantité de matière traitée par la tâche i au point d'événement n
$B^s_{i,n}$	quantité de matière consommée par la tâche i au point d'événement n
$B^f_{i,n}$	quantité de matière libérée par la tâche i au point d'événement n
$Bst_{s,n}$	quantité de matière stockée par la tâche fictive s au point d'événement n
$Br_{i,n}$	quantité de matière stockée par la tâche fictive s au point d'événement n
Cs	capacité de la cuve de stockage de l'état s
Csr	« capacité de stockage » maximale de l'Etat Ressource sr
Ds	Ordre de Fabrication de matière dans l'état s à satisfaire sur la campagne de production
D_{sn}	quantité d'état s fournie à l'environnement à l'instant n (exemple : demande en état s à satisfaire tout au long de l'horizon temporel et non pas seulement en fin d'horizon)
D_{st}	quantité d'état s fournie à l'environnement en période t (exemple : demande en état s à satisfaire tout au long de l'horizon temporel et non pas seulement en fin d'horizon)
f	modèle non linéaire du système
F	modèle du système
F^a_{in}	flux molaire d'entrée de l'appareil a mol.s^{-1}
F^a_{out}	flux molaire de sortie de l'appareil a mol.s^{-1}
ft_i	partie fixe de la durée opératoire de la tâche i
g	constante de gravitém.s^{-2}
H	Horizon de temps
h	hauteur
I	ensemble des tâches de la recette
I_j	ensemble des tâches pouvant être réalisées sur l'appareil j
I^c_s	ensemble des tâches consommant l'état de la matière s
I^p_s	ensemble des tâches produisant l'état de la matière s
I^c_{sr}	ensemble des tâches consommant l'Etat Ressource sr
I^p_{sr}	ensemble des tâches produisant l'Etat Ressource sr
J	ensemble des appareils du procédé
J_i	ensemble des appareils pouvant réaliser la tâche i
h_l	hauteur de liquide m
h_L	enthalpie molaire liquide J.mol^{-1}
hs	coût de stockage de l'état s
mh	modèle d'enthalpie molaire liquide J.mol^{-1}
mH	modèle d'enthalpie molaire vapeur J.mol^{-1}
mP	modèle de pression

MS variable définie pour le calcul du makespan

N ensemble des points d'événement

p paramètres physiques

p_{in} port d'entrée

P pression Pa

pt_{ij} représente la durée de la tâche i sur la ressource j

$pt_{i,n}$ durée opératoire de la tâche i au point d'événement n

$S_{t,c}$ surface (ou section) m^2

$S_{s,n}$ quantité de l'état s à l'instant n (date ou évènement suivant la formulation)

$S_{s,t}$ quantité de l'état s en période t contrainte par les bornes Csmin et Csmax

SF_s quantité finale de matière dans l'état s en fin d'horizon

S ensemble des états de la matière

S^f ensemble des états de la recette avec une politique de stockage en quantité finie (FIS)

S^z ensemble des états de la recette avec des contraintes de type Zero-Wait

S^n ensemble des états de la recette sans stockage intermédiaire

S0s stock initial de matière dans l'état s

Sr ensemble des Etats Ressources

Sr^z ensemble des Etats Ressources avec des contraintes de type Zero-Wait

$Sr_{s,n}$ quantité de matière dans l'Etat Ressource sr au point d'événement n

Sr0s stock initial l'Etat Ressource sr

t temps s

T température K

$Ts_{i,n}$ date à laquelle démarre la tâche i au point d'événement n

$Tf_{i,n}$ date à laquelle se finit la tâche i au point d'événement n

$tt^s_{j,n}$ date de début de la tâche active sur l'appareil j à l'évènement n

$tt^f_{j,n}$ date de fin de la tâche active sur l'appareil j à l'évènement n

$Tst^s{}_{s,n}$ date à laquelle démarre la tâche fictive de stockage s au point d'événement n

$Tst^f{}_{s,n}$ date à laquelle se termine la tâche fictive de stockage s au point d'événement n

$Tstr^s{}_{sr,n}$ date à laquelle démarre la tâche fictive de stockage s au point d'événement n

$Tstr^f{}_{sr,n}$ date à laquelle se termine la tâche fictive de stockage s au point d'événement n

U_l rétention molaire mol

V vecteur propre

V_{ml} volume molaire $m^3.mol^{-1}$

$V_i^{\min}, V_i^{\max}$ volume minimum et maximum de matière pour la tâche i

vt_i partie variable de la durée opératoire de la tâche i

$W_{i,j,t}$ variable binaire modélisant le lancement d'un lot en période t

$W_{i,n}$ variable binaire modélisant le lancement d'un lot au point d'événement n

$Ws_{i,n}$ variable binaire modélisant le lancement au point d'événement n

$Wf_{i,n}$ variable binaire modélisant l'arrêt au point d'événement n

Lettres grecques

Γ_p	générateur d'évènements
θ	paramètre du modèle
Π_{sn}	quantité d'état s reçue de l'environnement à l'instant n
Π_{st}	quantité d'état s reçue de l'environnement en période t
$\rho^c_{i,s}$	proportion de matière dans l'état s consommée par la tâche i
$\rho^p_{i,s}$	proportion de matière dans l'état s produite par la tâche i
$\rho r^c_{i,sr}$	proportion d'Etat Ressource sr consommée par la tâche i
$\rho r^p_{i,sr}$	proportion d'Etat Ressource sr produite par la tâche i
$\tau_{i',i}$	délai de nettoyage sur l'appareil j si on enchaîne une tâche i après une tache i'

Indices

in	entrée
i	indice de tâche dans la représentation R.T.N.
j	indice d'appareil dans la représentation R.T.N.
min	minimum
max	maximum
n	point d'évènement
out	sortie
s	indice d'état de la matière dans la représentation R.T.N.
sr	indice d'Etat Ressource

Abréviations

EANL	Équation Algébrique Non Linéaire
EDA	Équation Différentielle Algébrique
OO	Orienté Objet
RdPO	Réseau de Petri Différentiel objet
RdP	Réseau de Petri
RTN	Ressource-Task Network
SDH	Système Dynamique Hybride
SED	Système à Événements Discrets
STN	State-Task Network

Sigles

DISCo	**D**o **I**ntegrate by **S**oftware **Co**mponent
PrODHyS	**Pr**ocess **O**bject **D**ynamic **Hy**brid **S**imulator
ProSched	**Pr**ocess **Sched**uling

BIBLIOGRAPHIE

Bibliographie

A

[Al Kazzaz Y., 1989]
Al Kazzaz Y., – Sur l'ordonnancement d'atelier de fabrication : approche hiérarchisée et fonctionnement en boucle de pilotage. Thèse de l'Institut Polytechnique de Grenoble,1989.

[Alla H. et al., 1992]
Alla H., Cavaille J.B., Le Bail J., Bel G. (1992). Les systèmes de production par lot : une approche discret-continu utilisant les réseaux de Petri hybrides, Automation of Mixed Processes (ADPM 92), Janvier, Paris (France)

[Allan B.A., 1997]
A More Reusable Modeling System, PhD Thesis, Carnegie Mellon University, PittsBurg (USA)

[Alur R. et al., 1994]
Alur R., Dill D.L. (1994). A Theory of Timed Automata, Theoretical Computer Science, Vol.126, N°2, p.183-225

[Alur R. et al., 1995]
Alur R., Courcoubetis C., Halbwachs N., Henzinger T.A., Ho P.H., Nicollin X., A. Olivero, Sifakis J. and Yovine S. (1995). The algorithmic analysis of hybrid systems. Theoretical Computer Science, 138, p.3-34.

[Andersson M., 1994]
Andersson M., 1994. Object-Oriented Modelling and Simulation of Hybrid Systems, Ph.D thesis, Lund Institute of Technology, Sweden

[Andreu D., 1996]
Andreu D. (1996). Commande et Supervision des Procédés Discontinus : Une Approche Hybride. Thèse de doctorat, Novembre, Université Paul Sabatier, Toulouse (France).

[Archimède B. et al., 2001]
Archimède B. et Coudert T. – Reactive scheduling using a multi-agent system: the SCEP framework. International Journal Engineering Applications of Artificial Intelligence, vol. 14, Issue 5, 2001.

B

[Baker K.R.,1974]
Baker K.R., Introduction to sequencing and scheduling, John Wiley, New York, 1974

[Barton P.I. et al., 1994]
Barton P.I. and Pantelides C.C. (1994). The Modelling of Combined Discrete/Continuous Processes. AIChE Journal, 40, p.966-979

[Baudet et al, 1995]
Baudet P., Azzaro-Pantel C., Domenech S. et Pibouleau L., 1995, A discrete-event simulation approach for scheduling of batch processes, Computers and Chemical Engineering, 19, S633-S638

[Beedveld P. et al., 1991].
Beedveld P., Rosenberg R., Zhou T. (1991). Bibliography of bond graph theory and applications, Journal of Franklin Institute, Vol. 328, p.1067-1109.

[Belaud J.P. et al., 2002]
Belaud J.P., Pons M. (2002). Open Software Architecture For Process Simulation: The Current Status Of Cape-Open Standard, European Symposium of Computer Aided Process Engineering (ESCAPE 12), 26-29 mai, The Hague (The Neverthelands).

[Bellmann R. et al., 1974]
Bellmann R et Dreyfus S.E., Applied dynamic programming, Princeton University Press, 1974

[Bellmann R. et al., 1982]
Bellmann R., Esogbue A.O. et Nabeshima I., Mathematical Aspects of Scheduling and Applications, Pergamon Press, 1982

[Berard Ch., 1983]
BÉRARD (Ch.). – Contribution à la conception de structures logicielles pour le pilotage d'atelier. Thèse de Docteur-Ingénieur, Université de Bordeaux I, 1983.

[Bourseau P. et al., 1993]
Bourseau P., Modélisation des connaissances de l'ingénieur de procédé : application de l'intelligence artificielle, Doctorat d'Etat, Université Paris IV, 1993

[Branicky M.S., 1995]
Branicky M.S. (1995). Studies in hybrid systems : Modeling, Analysis and Control, PhD thesis, MIT, Massachusetts (USA)

[Buisson J. et al., 1998]
Buisson J., Cormerais H. (1998). Descriptor Systems for the Knowledge Modelling and Simulation of Hybrid Physical Systems, Journal Européen des systèmes automatisés, Vol.32, N°9-10, p.1047-1072

[Burkard R.E. et al., 2005]
Burkard R.E. et Hatzl J., 2005, Review, extensions and computational comparison of MILP formulations for scheduling of batch processes, Computers and Chemical Engineering, 29, 1752-1769

C

[Carlier J. et al., 1988]
Carlier J. et Chretienne P., Problèmes d'ordonnancement : modélisation / complexité / algorithmes, Masson, Paris, 1988

[Cellier F.E.,1991]
Cellier F.E. (1991). Continuous Systems Modelling, Berlin, Springer Verlag, 1991.
[Doumeingts G., 1990]
Doumeingts (G.). – Méthodes pour concevoir et spécifier les systèmes de production. LAP/GRAI université de Bordeaux, CIM-90 (1990).

[Champagnat et al., 1998]
Champagnat R., Esteban P., Pingaud H. and Valette R. (1998). Modeling and simulation of a hybrid system through Pr-Tr PN-DAE model. ADPM'98, p.131-137. Reims

[Chombart A. et al., 1996]
Chombart A., Flaus J.M., Valentin-Roubinet C. (1996). Hybrid Systems Modelling : a comparison of three methods applied to an example, Congrès IFAC'96, 30 Juin-5 Juillet, San Francisco (USA)

[Conway R.W. et al., 1967]
Conway R.W., Maxwell W.L. et Miller L.W., Theory of scheduling, Addison Wesley, 1967

[Combacau M. et al., 1990]
Combacau M., Courvoisier M. A hierarchical and modular structure for F.M.S. control and monitoring. 1rst conference on Artificial Intelligence, Simulation and Planning in High Autonomy Systems, Tuscon, Arizona, USA, Mars 1990

D

[Daubas B., 1994]
Daubas B. (1994). Modélisation et simulation des procédés continus et discontinus, Thèse de doctorat, INP, Toulouse (France).

[Deshpande A. et al., 1998]
Deshpande A., Gollu A. and Semenzato L. (1998). The shift programming language for dynamic networks of hybrid systems. IEEE Trans. Automatic Control special issue on Hybrid Systems

[Dindeleux E., 1992]
Dindeleux (E.). – Proposition d'un modèle et d'un système interactif d'aide à la décision ,pour la conduite d'atelier. Thèse de doctorat, université de Valenciennes (1992).

[Dorigo M. et al. 1999]
Dorigo (M.) et Di Caro (G.). – The Ant Colony Optimization Meta-Heuristic. In Corne (D.), Dorigo (M.) and Glover (F.) ed., New Ideas in Optimization, McGraw-Hill, 1999

[Doumeingts G. et al., 1983]
Doumeingts G. et al, (1983), La gestion de production assistée par ordinateur, Hermes publishing.

[Doumeingts G. , 1990]
Doumeingts G., Modelling techniques for CIM - CEC workshop on computer integrated manufacturing open system architecture. Brussels

[Dubois D. et al., 1990]
Dubois D., Gentil S. Intelligence Artificielle : un outil pour l'automatique. Journées annuelles du GR automatique, Octobre 1990

E

[Elmqvist et al., 1999]
Elmqvist H., Cellier F.E., Otter M. (1999). Modelica – A Unified Object-Oriented Language for Physical Systems Modeling, version 1.3.

[Engell S., 1997]
Engell S. (1997). Modelling and analysis of hybrid systems, 2nd IMACS MATHMOD Conference, p.17-31, Vienne (Autriche)

[Erschler J., 1976]
Erschler J., Analyse sous contraintes et aide à la décision pour certains problèmes d'ordonnancement, Thèse de doctorat d'Etat, Université Paul Sabatier, Toulouse, 1976

[Esquirol P. et al., 1999]
Esquirol P. et Lopez P., L'ordonnancement, Economica, Paris, 1999

F

[Fábián G. et al., 1998]
Fábián G., van Beek D.A. and Rooda J.E. (1998). Integration of the Discrete and the Continuous Behaviour in the Hybrid Chi Simulator, European Simulation Multiconference, Manchester

[Flaus J.M., 1998]
Flaus J.M. (1998). Modeling and Analysis of Hybrid Dynamical Systems : an Introduction, Journal Européen des Systèmes Automatisés (JESA), Vol.32, N°7-8, p.797-830

[Florin G. et al., 1991]
Florin G., Fraize C., Natkin S. (1991) Stochastic Petri Nets : Properties, Applications and Tools, Microelectronics and Reliability, Vol.31, N°4, p.669-697

[Floudas C.A. et al., 2004]
Floudas C.A. et Lin X., 2004, Continuous-time versus discrete-time approaches for scheduling of chemical processes : a review, Computers and Chemical Engineering, 28, 2109-2129

[French S., 1982]
French S., Sequencing and scheduling, Ellis-Horwood, 1982

[Fuch B. et al., 1997]
Fuchs B., Lieber J., Mille A. and Napoli A., Towards a Unified Theory of Adaptation in Case-Based Reasoning, Case-Based Reasoning Research and Development, Volume 1650/1999, Springer Berlin / Heidelberg

G

[Gani R. et al., 2002]
Gani R., Braunschweig B.L. (2002). Software architectures and tools for computer aided process Engineering, Elsevier, ISBN :0-444-50827-9.

[Gear C.W. et al., 1971]
Gear C.W. (1971). The Simultaneous Numerical Solution of Differential-Algebraic Equations, IEEE Transaction on Corcuit Theory, CT 18 (1), Ed. Academic Press

[Georgiadis M.C. and Macchietto S., 1997]
Georgiadis, MC, Macchietto, S, Layout of process plants: A novel approach, Joint 6th International Symposium on Process Systems Engineering/30th European Symposium on Computer Aided Process Engineering (PSE 97-ESCAPE-7), 1997, Pages: S337 - S342, ISBN: 0098-1354

[Giard V., 1988]
Giard V., Gestion de la production, 2ème édition, Economica 1988

[Glover F., 1986]
Glover F., Future paths for integer programming and links to artificial intelligence. Computer & Operations Research, 13, p533-549, 1986

[Glover F., 1989]
Glover F., Tabu search : Part i. ORSA Journal on Computing, 1, 1989

[Glover F., 1990]
Glover F., Tabu search : Part ii. ORSA Journal on Computing, 2, 1990

[GOTHA, 1993]
GOTHA, « Les problèmes d'ordonnancement », Recherche Opérationnelle, vol. 27, n°1, p.77-150, 1993

[Grabot B., 2006]
Grabot B., Ordonnancement d'ateliers manufacturiers, Techniques de l'Ingénieur, 2006

[Guéguen et al., 2001]
Guéguen, H. and Lefebvre, M. A. (2001). A comparison of mixed specification formalisms, Journal Européen des Systèmes Automatises (APII JESA), Vol.35, N°4, p.381-394

[Guéret C. et al., 2000]
Guéret C, Prins C. et Sevaux M., Programmation Linéaire, Eyrolles edition 2000

H

[Haït A. et al., 2007]
Haït A., Artigues C., Baptiste P., Trépanier M. (2007) Ordonnancement sous contraintes d'énergie et de ressources humaines. In: 11e congrès de la Société Française de Génie des Procédés, 9-11 Oct 2007, Saint-Etienne, France.

[Hétreux G. et al., 2003]
Hétreux G., Perret J., LeLann J.M. Bibliothèque orientée objet pour la conception de simulateurs dynamiques hybrides, Congrès Français de Génie des Procédés (CFGP'2003), 9-11 Septembre, Saint-Nazaire, 2003

[Holland J.H., 1975]
Holland J.H., Adaptation in Natural and Artificial Systems, MIT Press, Cambridge, Mass., 1975

I

[IEC 61512-1, 1997]
IEC 61512-1 Batch Control - Part 1: Models and Terminology, 1997

[Ierapetritou M.G. et Floudas C.A. , 1998]
Ierapetritou M.G., Floudas C.A., 1998, Effective continuous-time formulation for short-term scheduling: 1. Multipurpouse batch processes, Industrial and Engineering Chemistry Research, 37, 4341

J

[Janak S. et al., 2004]
Janak S., Lin X., Floudas C. A., 2004, Enhanced Continuous-Time Unit-Specific Event-Based Formulation for Short-Term Scheduling of Multipurpose Batch Processes : Resource Constraints and Mixed Storage Policies, Industrial Engineering and Chemical Research, 43 (10), 2516-2533

[Joglekar G.S. et al., 1985]
Joglekar G.S., Reklaitis G.V. (1985), A simulator for batch and semi-continuous processes, Computers and Chemical Engineering, Vol. 8, No 6, pp. 315-327

[Jourda L., 1996]
Jourda L., (1996). Composants Logiciels Orientés Objets pour la Modélisation et la Simulation des Procédés Chimiques, Thèse de doctorat, INP, Toulouse(France)

[Jourda L. et al., 1996]
Jourda L. Joulia X. and Koehret B. (1996). Introducing ATOM, the Applied Thermodynamic Object-Oriented Model. In: Computer & Chemical Engineering, 20A, S157-S164

K

[Kacem I., 2003]
Kacem I., Ordonnancement multicritère des jobs-shops flexibles : formulation, bornes inférieures, et approche évolutionniste coopérative, PhD thesis, l'Ecole Centrale de Lille et l'Université de Lille 1, 2003

[Kallrath J., 2002]
Kallrath J., 2002, Planning and Scheduling in the process industry, OR Spectrum, 24, 219-250

[Kaufmann A., 1992]
Kaufmann (A.). – Introduction à la logique floue. Techniques de l'Ingénieur, A 120, 1992
Mathématiques pour l'ingénieur.

[Kesten Y. et al., 1992]
Kesten Y., Pnueli A. (1992), Timed and hybrid statecharts and their textual representation, Lecture Notes in Computer Science (LNCS), Vol. 571

[Kirkpatrick S. et al., 1983]
Kirkpatrick S., Gelattand Jr., and M. P. Vecchi Optimization by simulated annealing. Science, 220 (4598): 671-680, Mai 1983.

[Kondili E. et al., 1993]
Kondili E., Pantelides C.C. et Sargent R.W.H., 1993, A general algorithm for short-term scheduling of batch operations – I MILP formulation, Computers and Chemical Engineering, 17-2, 211-227

L

[Le Bail J. et al., 1991]
Le Bail J., Alla H. and David R. (1991). Hybrid Petri Nets, Proceedings of the European Control Conference, p.1472-1477, Grenoble

[Le Lann J.M., 1999]
Le Lann J.M. (1999). Des mathématiques à la simulation dynamique robuste des procédés : le traitement algébro-différentiel des équations EDA. Habilitation à Diriger les Recherches, INP, Toulouse, France

[Lopez P. et al., 2001]
Lopez P., Roubellat F. (2001), Ordonnancement de la production. Hermès Sciences publications

M

[Maravelias C.T. et al., 2003]
Maravelias C.T., Grossmann I.E., 2003, Minimization of the makespan with a discrete-time State-Task network formulation, Industrial and Engineering Chemistry Research, 42, 6252-6257

[Mattsson S.E. et al.,1993]
Mattsson S.E., Andersson M., Åström K.J. (1993). Object-Oriented Modelling and Simulation, In Linkens, Ed., CAD for Control Systems , chapitre 2, p.31-69. Marcel Dekker Inc, New York.

[Méndez et Cerda, 2003]
Méndez C.A. et Cerdá J., 2003, Dynamic scheduling in multiproduct batch plants, Computers and Chemical Engineering, 27, 1247-1259

[Méndez C.A. et al., 2006]
Méndez C.A., Cerdá J., Grossmann I.E., Harjunkoski I. et Fahl M., 2006, State-of-the-art review of optimization methods for short-term scheduling of batch processes, Computers and Chemical Engineering, vol. 30, Issues 6-7, Pages 913-946

[Merlin P.M. et al., 1976]
Merlin P.M., Segall A. (1976). Recoverability of Communication Protocols-Implications of a Theoretical Study, IEEE Transactions on Communications

[Metropolis N. et al., 1953]
Metropolis N., Rosenbluth A., Rosenbluth M., Teller A., Teller E., Equation of state calculations by fast computing machines, Journal f Chemical Physics, 21, 1953

[Moyse, A., 2000]
Moyse, A., Odysseo : plate-forme orientée-objet pour la simulation dynamique des procédés. Thèse de Doctorat, INP de Toulouse, France, 2000

N

[Nawaz M. et al., 1983]
Nawaz M., Enscore Jr. E. E. et Ham I., "A heuristic algorithm for the m-macine, n-job flow shop sequencing problem", Omega, vol.11, 1983

[Napoli A., 1999]
Napoli A. (Éd.). – Raisonnement à partir de cas numéro spécial de la revue « Intelligence artificielle », vol. 13, no 1, Hermès, 1999.
[Nilsson B., 1993]
Nilsson B., Dynamic modeling of chemical processes using Omola, IchEmE Symposium Series, Vol. 133, pp. 103-110, 1993

O

[Olivier N. et al., 2005]
Olivier N., Hétreux G., LeLann J.M., Formal modelling and simulation for control of batch processes, Conference on Conceptual Modelling and Simulation CMS'05; Marseille, France, 2005

[Olivier N. et al., 2006]
Olivier N., Hétreux G., LeLann J.M., LeLann M.V., Use of an Object Oriented Dynamic Hybrid Simulator for the Monitoring of Industrial Processes, proceedings of ADHS'06, Alghero, Italia, 2006 235-240

[Olivier N. et al., 2007]
Olivier-Maget N., Hétreux G., LeLann J.M., LeLann M.V., Fault detection using a hybrid dynamic simulator: Application to a hydraulic system, International Modeling and Simulation Multiconference CMS'07 ; Buenos Aires (Argentina) ; 8-10 Février 2007

P

[Pantelides C.C., 1994]
Pantelides C. C. (1994). Unified frameworks for optimal process planning and scheduling. In Foundations of computer-aided process operations. New York: Cache publications, pp. 253–274.

[Pekny J.F., 1998].
Pekny J.F., Reklaitis G.V., Towards the convergence of theory and practice : a technology guide for scheduling / planning methodology; Proc. Of FOCA-PO (editors Pekny and Blau), CACHE, AIChE Symposium series, Vol. 94, p.91, 1998

[Pekny J.F. et al., 1998]
Pekny, J. F., & Reklaitis, G. V. (1998). Towards the convergence of theory and practice: A technology guide for scheduling/planning methodology. In Proceedings of the third international conference on foundations of computer-aided process operations (pp. 91–111).

[Perret J., 2003]
Perret J. Intégration des Réseaux de Petri Différen-tiels à Objets dans une plate-forme de simulation dynamique hybride : application aux procédés indus-triels, Thèse de Doctorat, INP, Toulouse (France)., 2003

[Perret J. et al., 2004]
Perret J., Hétreux G., LeLann J.M., Integration of an object formalism within a hybrid dynamic simulation environment, Control Engineering Practice (Elsevier), 12 10 (2004) 1211-1223

[Piela P.C., 1989]
Piela P.C., ASCEND : an object-oriented computer environment for modelling and analysis, PhD The-sis, Carnegie-Mellon University, USA, 1989

[Pinedo M., 1995]
Pinedo M., Scheduling : theory, algorithms and systems, Prentice Hall, 1995

[Pinto J.M. et al., 1998]
Pinto J. M., & Grossmann, I. E. (1998). Assignments and sequencing models of the scheduling of process systems. Annals of Operations Research, 81, 433–466.

R

[Rippin D.W.T., 1983]
Rippin D.W.T., Design and operation of multiproduct and multipurpose batch chemical plants – an analysis of problem structure, Computers and Chemical Engineering, Vol. 7, pp. 463-481, 1983.

[Roslöf et al, 2001]
Roslöf J., Harjunkoski I, björkqvist J., Karlsson S. and Westerlund T, 2001, An MILP-based reordering algorithm for complex industrial scheduling and rescheduling, , Computers and Chemical Engineering, 25, 821-828

[Roy B. et al., 1964]
Roy B. et Sussmann B. Les problèmes d'ordonnancement avec contraintes disjonctive. Note D.S. 9 bis, SEMA, Paris, France 1964

S

[Sargousse A., 1999]
Sargousse, A., Noyau numérique orientée-objet dédié à la simulation des systèmes dynamiques hybri-des. Thèse de Doctorat, INPT, France, 1999

[Scott Morton M., 1971]
Scott Morton M. (1971) : Management decision systems, computer based support for decision making. Harvard University, Boston, MA, USA.

[Semet F., 1995]
F. Semet, "A two-phase algorithm for the partial accessibility constrained vehicle routing problem", Annals of Operations Research 61, pp. 45-65, 1995.

[Sénéchal O., 2004]
Sénéchal O., 2004, Pilotage des systèmes de production vers la performance globale, HDR Université de Valencienne et du Hainot Cambresis

[Shah N., 1998]
Shah, N. (1998). Single and multisite planning and scheduling: Current status and future challenges. In Proceedings of the third international conference on foundations of computer-aided process operations (pp. 75–90).

[Shah N. et al., 1995]
Shah N., Kuriyan K., Liberis L., Pantelides C. C., Papageorgiou L. G. andRiminucci P., User interfaces for mathematical programming based multipurpose plant optimisation systems, Computers & Chemical Engineering, Vol. 19, Sup. 1, 1995, Pages 765-772
European Symposium on Computer Aided Process Engineering 3-5

[Shaik M.A. et al., 2005]
Shaik M.A., Janak S.L. et Floudas C.A., 2005, Continuous-Time Models for Short-Term Scheduling in Multipurpose Batch Plants: a Comparative Study, Chemical Engineering, Princeton University, Princeton, NJ 08540

[Shen W. , 1999]
Shen W. et Norrie D.H. – Agent-based systems for intelligent manufacturing: a state
of the art survey. Knowledge and Information Systems, vol. 1, no 2, 1999.

[Sifakis J., 1977]
Sifakis, J. (1977). Use of Petri Nets for Performance Evaluation, Measuring, Modelling and Evaluating Computer Systems, pp. 75-93. Amsterdam (North Holland)

[Stephanopoulos G. et al., 1990]
Stephanopoulos G., Henning G., Leone H.. MODEL.LA : a modelling language for process engineering. I- the formal framework, Computers and Chemical Engineering, 14,8, pp. 813-846, 1990

[Sundaramoorthy A. and Karimi I.A., 2005],
Sundaramoorthy A., & Karimi I. A. (2005). A simpler better slotbased continuous-time formulation for short-term scheduling in multiproduct batch plants. Chemical and Engineering Science, 60, 2679– 2702.

T

[Talbi D. et al., 2004]
Talbi (D.), Geneste (L.), Grabot (B.), Prévitali (R.) et Hostachy (P.). – Application of Optimization Techniques to Parameter Set-up of Industrial Scheduling Software. Computers In Industry, vol. 55, no 2, 2004.

[Thévenon L., 2000]
Thévenon L. (2000). Représentation des Systèmes Hybrides Complexes par Flux de Données : Développement d'un Outil de Modélisation et de Simulation des Procédés Batch, Thèse de doctorat, INP, Grenoble (France)

[Tittus M., 1995]
Tittus, M. (1995), Control Synthesis for Batch Processes. Thèse de doctorat, Chalmers Univ. of Technology, Sweden.

[Trentesaux D.et al., 2002]
Trentesaux D., Sénéchal O., Conduite des systèmes de production manufacturière, Techniques de l'Ingénieur ,TI-s7998, 2002

V

[Van Hulle M.M., 1991]
VAN HULLE (M.M.). – A goal programming network for mixed integer linear programming: a case study for the job-shop scheduling problem. International Journal of Neural Networks, vol. 2, no 3, 1991.

W

[Waldner J.-B., 1990]
Waldner J.-B. (1990), CIM, les nouvelles perspectives de la production, DUNOD-BORDAS

[Widmer M. et al., 2001]
Widmer M., Hertz A., and Costa D. Les métaheuristiques, chapter 3, pages 55-93. hermes edition, 2001.

[Wöllhaf K. et al., 1996]
Wöllhaf K., Fritz M., Schulz C. and Engell S. (1996). BaSiP – Batch Process Simulation With Dynamically Reconfigured Process Dynamics. Supplement to Computers and Chemical Engineering, 20(972), p.1281-1286

Z

[Zaraté P., 2005]
Zaraté P. Des Systèmes Interactifs d'Aide à la Décision aux Systèmes Coopératifs d'Aide à la Décision : Contributions conceptuelles et fonctionnelles. Habilitation à diriger des recherches, Institut National Polytechnique de Toulouse, décembre 2005.

[Zaytoon J., 2001]
Zaytoon J., Systèmes dynamiques hybrides. Hermès Sciences publications, 2001

LISTE DES PUBLICATIONS ET COMMUNICATIONS

Le travail réalisé au cours de ces quatre années a donné lieu à plusieurs communications dans des congrès nationaux et internationaux ainsi qu'à une publication parue et à une publication en cours de relecture.

- **SOUMISSION REVUE INTERNATIONALE (AVEC COMITE DE LECTURE)**

[1] FABRE F., HETREUX G., ZARATE P., LE LANN J.M.
Optimization for scheduling of batch and semi-continuous process
Computer and Chemical Engineering, 2008 (en cours de relecture)

- **REVUE NATIONALE (AVEC COMITE DE LECTURE)**

[2] FABRE F., HETREUX G., LE LANN J.M., THERY R.
Approche combinée Optimisation/Simulation pour l'ordonnancement des procédés discontinus
Récents Progrès en Génie des Procédés – 96 – 2007, ISBN 2-910239-70-5, Ed. SFGP, Paris, France

- **CONFERENCES / WORKSHOPS INTERNATIONAUX AVEC ACTES AVEC COMITE DE SELECTION ET ACCEPTATION SUR TEXTE COMPLET**

[3] FABRE F., HETREUX G., ZARATE P., LE LANN J.M.
Supporting Process Engineering: From Simulators to Simulation Environment.
International Conference on Creativity and Innovation in Decision Making and Decision Support (CIDMDS 2006), Londres, 28-JUN-06-01-JUL-06, Vol. 2, Frédéric Adam, Patrick Brézillon, Sven Carlsson, Patrick Humphreys (Eds.), Ludic Publishing Ltd, p. 929-945, juin 2006

[4] FABRE F., HETREUX G., ZARATE P., LE LANN J.M.
Supporting Modelling Phase in Process Engineering.
Workshop of the Euro Working Group on Decision Support Systems, Londres, 28-JUN-06, Pascale Zaraté (Eds.).

- **CONFERENCE NATIONALE AVEC ACTES AVEC COMITE DE SELECTION ET ACCEPTATION SUR TEXTE COMPLET**

[5] FABRE F., HETREUX G., ZARATE P., LE LANN J.M.
Système d'Aide à la modélisation pour la simulation dynamique hybride des procédés. Systèmes d'Information, Modélisation, Optimisation SIMO'06 ; Toulouse (France), 11-12 Octobre 2006

- **DIVERS**

[6] ZARATE P., FABRE F., HETREUX G.
From Simulators to Simulation Environment for Supporting Process Engineering. Dans : EURO XXI, Reykjavik, Iceland, 02-JUL-06-05-JUL-06.

[7] FABRE F
Optimisation pour l'ordonnancement des procédés semi-continus
8ème congrès des doctorants EDSYS ; Albi (France) ; 10 Mai 2007

RESUME

CONDUITE ORIENTÉE ORDONNANCEMENT D'UN SIMULATEUR DYNAMIQUE HYBRIDE : APPLICATION AUX PROCÉDÉS DISCONTINUS

Mots Clés :

Environnement de modélisation et simulation de procédés, Pilotage d'une simulation dynamique hybride, Ordonnancement court terme, Programmation Linéaire en Variables Mixtes, Réseau de Petri Différentiel Objet, Couplage Optimisation/simulation.

Résumé :

Ce manuscrit présente des travaux visant à intégrer un module d'ordonnancement (*ProSched*) à l'environnement de modélisation et simulation dynamique hybride *PrODHyS* dans le but d'automatiser la génération de scénarii de simulation de procédés discontinus sur la base d'une recette et d'une liste d'ordres de fabrication (*OF*). La méthodologie développée repose sur une approche mixte optimisation/simulation. Dans ce cadre, trois points essentiels ont été développés dans ces travaux :

- tout d'abord, concevoir et développer des composants réutilisables (*classes de recette*) permettant de modéliser de manière hiérarchisée et systématique le déroulement des opérations unitaires. Pour cela, les notions de jeton *Task* et de macro-place paramétrable ont été introduites dans les *RdPDO* et permettent de décrire les recettes à réaliser par assemblage de ces composants prédéfinis.
- ensuite, définir un modèle mathématique générique d'ordonnancement basé sur un formalisme de représentation bien établi (le *R.T.N.*) qui permet de modéliser les principales caractéristiques d'un procédé discontinu et de fournir l'ensemble des données d'entrée nécessaires au modèle de simulation. Pour cela, un modèle PLNE basé sur la formulation *Unit Specific Event* a été mis en œuvre.
- enfin, définir l'interface existant entre le modèle d'optimisation et le modèle de simulation, à travers la notion de *place de pilotage* et de *centre de décision* au niveau du simulateur. Dans ce cadre, différentes stratégies de couplage sont proposées.

Les potentialités de cette approche sont illustrées par la simulation de deux exemples de procédés complets.

CONTROL ORIENTED SCHEDULING OF A DYNAMIC HYBRID SIMULATOR: APPLICATION TO BATCH PROCESSES

Keywords :

Environment for modelling and simulation of processes, Control of dynamic hybrid simulation, Short-term scheduling, Mixed Integer Linear Programming, Object Differential Petri nets, Optimization / simulation approach

Abstract :

This thesis presents works which aim to incorporate a scheduling module (*ProSched*) to an environment for modeling and dynamic hybrid simulation *PrODHyS* in order to automate the generation of scenarios for simulation of batch processes based on a recipe and a list of production orders (OF). The methodology developed is based on a mixed optimization / simulation approach. In this context, three key points have been developed in this work:

- First, design and develop reusable components (recipe classes) for the hierarchical and systematic modeling of the sequencing of unit operations. For this, the notions of *Task token* and *macro-place* have been introduced in the *RdPDO* formalism and allow the modeling of recipes by assembling these predefined components.
- Secondly, define a generic mathematical model of scheduling based on a well defined graphical formalism (*RTN*) that models the main characteristics of batch processes and provide all input data necessary to the simulation model. For this, a *MILP* model based on the *Unit Specific Event* formulation has been implemented.
- Finally, define the interface between the optimization model and the simulation model through the concept of *control place* and *decision-making center* at the simulator level. In this context, various strategies of mixing optimization and simulation are proposed.

The potential of this approach is illustrated by the simulation of two complete manufacturing processes.