Approche énergétique pour l'ordonnancement de tâches sous contraintes de temps et de ressources

Pierre Lopez

Année 1991

THESE

présentée

au **Laboratoire d'Automatique et d'Analyse des Systèmes** du **C.N.R.S.**

en vue de l'obtention

du **DOCTORAT de l'Université Paul Sabatier de Toulouse (Sciences)**
Spécialité : Automatique

par

Pierre LOPEZ
Maître ès-Sciences

APPROCHE ENERGETIQUE POUR L'ORDONNANCEMENT DE TACHES SOUS CONTRAINTES DE TEMPS ET DE RESSOURCES

Soutenue le 23 Septembre 1991 devant le Jury :

MM.	M.	COURVOISIER	*Président*
	J.	CARLIER	*Rapporteurs*
	D.	DUBOIS	
	J.	FAVREL	
	J.	ERSCHLER	*Directeur de thèse*
	G.	BEL	*Examinateurs*
	R.	COMPANYS	

Rapport LAAS n° 91337

Cette thèse a été préparée au Laboratoire d'Automatique et d'Analyse des Systèmes du C.N.R.S.
7, avenue du Colonel Roche – 31077 TOULOUSE Cédex

Avant-propos

Le travail présenté dans ce mémoire a été effectué au Laboratoire d'Automatique et d'Analyse des Systèmes (L.A.A.S.) du C.N.R.S. Il a été réalisé en collaboration avec la région grâce au Groupement d'Intérêt Public "Productique Midi-Pyrénées" (PROMIP). Je tiens à remercier Monsieur le Professeur Alain COSTES, directeur du L.A.A.S. et Monsieur Georges GIRALT, alors directeur de PROMIP, de m'avoir accueilli dans leurs équipes de recherche.

Je tiens à remercier particulièrement mon directeur de recherche, Monsieur Jacques ERSCHLER, Professeur à l'I.N.S.A. de Toulouse. Ses conseils, ses critiques, sa compétence, ses encouragements et sa sympathie m'ont permis de mener cette thèse à son terme.

Ma reconnaissance envers Monsieur Marc COURVOISIER est double pour l'honneur qu'il me fait de présider le jury de cette thèse et pour m'avoir accueilli au sein du groupe "Systèmes de Production" dont il était alors responsable.

Je suis extrêmement reconnaissant a Messieurs Jacques CARLIER, Professeur à l'Université de Technologie de Compiègne, Didier DUBOIS, Directeur de Recherche au C.N.R.S. à l'Institut de Recherche en Informatique de Toulouse, et Joël FAVREL, Professeur à l'I.N.S.A. de Lyon, pour avoir accepté d'étudier mes travaux et d'en être les rapporteurs.

J'exprime également tous mes remerciements à Monsieur Gérard BEL, Ingénieur de Recherche au Centre d'Etudes et de Recherches de Toulouse, pour l'intérêt qu'il a bien voulu porter à ce travail et pour sa participation au jury.

Je remercie sincèrement Jean B. LASSERRE pour m'avoir souvent aidé en matière de mathématiques et de langue anglaise, Robert VALETTE pour ses conseils pratiques, ainsi que mes collègues de bureau, Catherine THURIOT et Gilbert de TERSSAC, qui ont permis d'améliorer certains points de mon travail.

Une mention spéciale est réservée à Patrick ESQUIROL pour avoir partagé avec moi sa précieuse connaissance, son temps... son humour aussi.

Je tiens à exprimer ma gratitude aux personnes qui ont consacré du temps à la lecture du manuscrit, et particulièrement Mademoiselle Anne JULIA pour s'être investie dans un domaine qui n'est pas le sien.

Merci à Madame Eliane DUFOUR pour son efficacité et sa gentillesse et aux personnes ayant permis la réalisation matérielle de ce document : Mesdames POWELL et RABARY, Messieurs CATALA, DAURAT, LAPEYRE-MESTRE, LORTAL et ZITTEL.

Je ne pourrais terminer sans parler des gens qui ont contribué à la bonne ambiance au L.A.A.S au cours de ces trois années. Je n'énumère pas la liste mais je suis sûr que les personnes concernées n'y verront pas offense et se reconnaîtront.

A mes parents, à LAURA : que ce mémoire soit un témoignage de mon affection.

Table des Matières

Introduction

L'automatisation intégrée de la production fait appel à des méthodes de **gestion de la production** destinée à dispenser des outils d'aide à la décision. En raison (1) de la complexité du problème due à la variété des produits et des processus de fabrication (fabrication continue, discrète, en petites, moyennes ou grandes séries...) et (2) de difficultés liées, par exemple, à l'interconnexion de décisions issues de l'organisation hiérarchique d'une entreprise, la gestion de production est généralement abordée à l'aide d'une structure de décisions multi-niveaux (long terme, moyen terme, court terme) [Buffa et Miller 79]. Le plus détaillé de ces niveaux s'occupe de l'**ordonnancement de tâches** et correspond à la gestion à court terme. C'est à ce niveau que se situe notre travail.

L'objectif de la fonction ordonnancement est d'organiser dans le temps la réalisation de tâches interdépendantes compte tenu de contraintes sur le temps et les ressources. De tels problèmes se rencontrent dans différents contextes tels que la gestion de grands projets, la conduite d'ateliers de fabrication, l'organisation d'activités de service, l'exploitation de systèmes informatiques...

Un problème d'ordonnancement présente deux aspects :

- un aspect **statique** lié à la génération d'un plan sur la base de données prévisionnelles ;
- un aspect **dynamique** lié à la prise de décisions en temps réel compte tenu de l'état effectif d'avancement des tâches.

En pratique, la résolution du problème consiste à concilier ces deux aspects de façon à atteindre des objectifs globaux liés au respect du plan, tout en étant capable de s'adapter, lors de la prise de décision, aux aléas qui surviennent au cours du temps. Ceci conduit à s'intéresser à l'autonomie de décision disponible, compte tenu d'un cadre lié au respect d'objectifs globaux.

Deux approches sont fréquemment utilisées pour aborder les problèmes d'ordonnancement :

1. la recherche de solutions optimales à l'aide de **méthodes d'optimisation**

combinatoire. Cette approche privilégie le point de vue statique. Pour des raisons de complexité, elle a des difficultés à traiter des problèmes réalistes ;

2. l'utilisation de **règles heuristiques** de décision locale. Cette approche peut être utilisée aussi bien dans une optique statique pour générer progressivement un plan, que dans une optique dynamique pour prendre des décisions en temps réel. Elle permet d'aborder des problèmes réalistes, mais la qualité des solutions obtenues vis-à-vis d'objectifs globaux n'est pas facile à maîtriser.

Par ailleurs, les problèmes d'ordonnancement ont pu être abordés grâce aux techniques de représentation de connaissances et de résolution de problèmes issues de l'**intelligence artificielle**. Cette approche, de nature heuristique, vise à aborder des problèmes réalistes tout en conservant une certaine rigueur vis-à-vis d'objectifs globaux. La connaissance est structurée en différents types ; on peut ainsi distinguer les connaissances théoriques, empiriques et pratiques.

L'approche retenue dans notre travail s'intéresse principalement à la connaissance **théorique**, relative à la gestion du temps et des ressources. Elle peut s'appuyer sur un module qui cherche à caractériser l'ensemble des décisions compatibles avec les décisions prises à un niveau plus agrégé qui correspondent à des contraintes pour le niveau considéré : cette démarche conduit à la caractérisation des ordonnancements **admissibles**. Un tel module explicite les degrés de liberté disponibles pour la prise de décision ; il peut ensuite coopérer avec d'autres modules utilisant d'autres types de connaissances pour générer une solution intéressante.

Les développements présentés constituent un prolongement et une généralisation de travaux antérieurs sur l'**analyse sous contraintes** de problèmes d'ordonnancement [Erschler 76][Esquirol 87].

Ce travail de recherche a été accompli dans le cadre du projet Gestion de Production du Groupement d'Intérêt Public "PROductique MIdi-Pyrénées" (G.I.P. – PROMIP).

Le mémoire est divisé en cinq chapitres.

Après avoir situé la fonction ordonnancement au sein d'un système de gestion structuré en plusieurs niveaux hiérarchiques, le premier chapitre présente les concepts de base en ordonnancement de tâches et en tire une typologie des problèmes d'ordonnancement. Les méthodes traditionnellement utilisées pour leur résolution sont ensuite exposées, avant de définir la problématique retenue et l'approche que nous proposons.

Le deuxième chapitre introduit les éléments de base utilisés dans notre approche. L'**intervalle temps-ressource** et la **consommation obligatoire d'énergie** sont ainsi définis. La notion d'intervalle temps-ressource amène à employer

un nouvel outil de modélisation : le **graphe potentiels-bornes.**

Les deux chapitres suivants proposent des règles d'inférence basées sur le concept d'énergie pour la caractérisation des ordonnancements admissibles.

Le troisième chapitre utilise la notion d'**ensemble critique de tâches en conflit** et étudie son impact sur le séquencement de tâches. L'intérêt du concept d'énergie pour la résolution des conflits est mis en évidence. Plusieurs règles sont combinées de façon à trouver des caractéristiques séquentielles actualisantes.

Un raisonnement et des règles associées, permettant d'**actualiser directement** des dates limites sont proposés dans le quatrième chapitre. Cette approche consiste à équilibrer l'énergie consommée dans le temps grâce à un resserrement des fenêtres temporelles associées aux tâches.

Le cinquième chapitre enfin, présente les modules réalisés en utilisant l'approche énergétique. Le fonctionnement de ces modules est illustré sur des exemples.

* *

*

Chapitre I

Les problèmes d'ordonnancement

Préambule — *Un grand nombre d'ouvrages ou d'articles traitent des problèmes d'ordonnancement de plus ou moins près. En dehors des références bibliographiques mentionnées dans ce chapitre, et de manière générale, nous nous sommes également rapportés à : [Faure et al. 76] et [Carlier et Chrétienne 88].*

I.1 Ordonnancement et gestion de production

I.1.1 Système de production – Gestion de production

La *production* est une opération de transformation d'un ensemble de matières premières ou de composants en produits finis. Un *système de production* est constitué de l'ensemble des éléments qui interviennent dans cette transformation et qui correspondent aux moyens ou ressources (humaines ou technologiques). La *gestion de production* a pour rôle de rechercher une organisation efficace dans le temps du système de production afin de fabriquer des produits dans les quantités et les délais demandés.

> "Elle s'appuie sur un ensemble d'outils d'analyse et de résolution de problèmes qui visent à limiter les ressources nécessaires à l'obtention d'une production dont les caractéristiques technico-commerciales sont connues" [Giard 88].

Le champ de la gestion de production est un domaine très vaste qui recouvre des aspects aussi divers que :

- la gestion des caractéristiques techniques : celles-ci concernent les nomenclatures (représentation hiérarchique décrivant les relations entre un produit et

les composants intervenant dans sa réalisation), les gammes opératoires (processus de fabrication presenté sous la forme d'un enchaînement d'opérations), les délais de production, etc ;

- la gestion des données commerciales : elle a en charge un carnet de commandes et établit les calendriers de livraisons souhaitées pour satisfaire les objectifs de vente ;
- la gestion des matières : elle assure l'approvisionnement en matières premières et composants, mais aussi le stockage des produits fabriqués (gestion des stocks) ;
- la gestion du travail : elle détermine de manière précise à quel moment et avec quelles ressources doivent être réalisées les tâches permettant de fabriquer les produits. Elle utilise des données commerciales, techniques et des informations issues du suivi de production (celui-ci renseigne sur l'état actuel du système de production : moyens disponibles, volume des stocks, des en-cours, ...). Cette gestion du temps et des ressources est assurée par la fonction *planification-ordonnancement*, qui assure la commande dynamique du système de production.

I.1.2 Approche hiérarchisée

La gestion de production, même réduite à la seule fonction planification-ordonnancement, demeure un problème vaste et complexe. Pour le résoudre, on peut utiliser une approche globale où l'on modélise l'intégralité du problème par un modèle unique, ou une *approche hiérarchisée* [Faure 86] qui consiste à décomposer le problème global en une succession de sous-problèmes. Cette dernière approche présente l'avantage de réduire le nombre de variables pour chaque niveau de décision et d'homogénéiser leurs caractéristiques. Ces décisions sont communément réparties dans une structure à trois niveaux (figure I.1) :

- les *décisions stratégiques* traduisent la politique à long terme de l'entreprise. Elles définissent les objectifs de l'entreprise et les moyens qu'elle entend mettre en œuvre pour y parvenir. Ces décisions, prises sur un horizon de deux à cinq ans, concernent par exemple : la politique budgétaire (investissements en moyens matériels, financiers ou humains), les nouvelles orientations suivant la demande, les campagnes publicitaires, etc ;
- les *décisions tactiques* assurent la gestion à moyen terme, comme la planification de la production et de l'utilisation des moyens, sur une période dont l'horizon est de plusieurs mois ;
- les *décisions opérationnelles* rentrent dans le cadre de la gestion à court terme. Elles déterminent, pour les jours ou les semaines à venir, l'ordonnancement des opérations de fabrication, l'affectation des différents moyens, ...

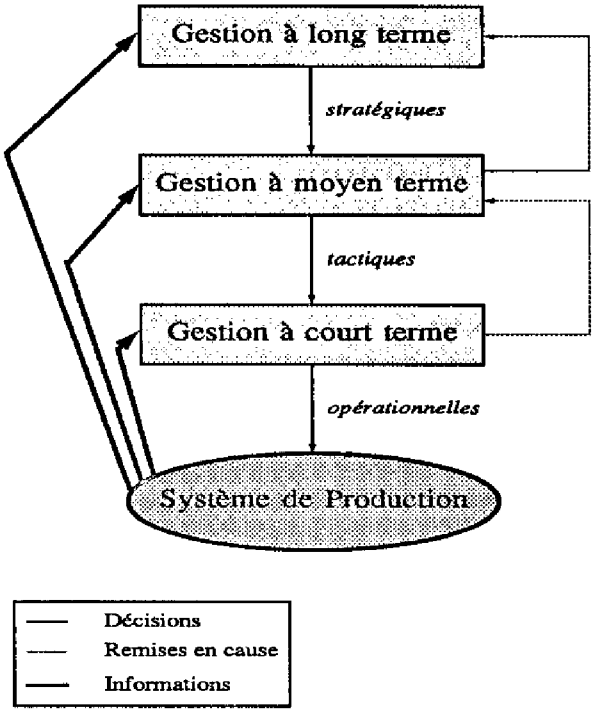


Figure I.1 : Structure de décisions à trois niveaux

Chaque niveau est caractérisé par l'horizon temporel sur lequel s'effectue la gestion, mais aussi par le niveau de détail des entités considérées (familles de produits pour des décisions tactiques ou stratégiques, composants élémentaires de produits pour les décisions opérationnelles). Elle est souvent associée à une organisation hiérarchique de l'entreprise, chaque type de décision étant du ressort d'une catégorie de personnel, selon sa compétence.

Des interactions existent nécessairement entre les niveaux : toute décision issue d'un certain niveau fixe les contraintes que doit respecter le niveau inférieur ; réciproquement, cette décision risque d'être remise en cause par le niveau inférieur si celui-ci ne peut suivre la consigne imposée. Ce problème de *cohérence* entre niveaux peut être abordé de façon analytique [Mercé 87] ou par itérations successives entre niveaux [Imbert 86].

Les problèmes d'ordonnancement, qui sont l'objet de notre travail, se situent dans le cadre de la gestion à court terme.

I.2 Concepts de base

I.2.1 Le problème d'ordonnancement

Un *problème d'ordonnancement* se pose lorsqu'il s'agit de programmer dans le temps, l'exécution de diverses tâches soumises à des contraintes, auxquelles sont attribuées des ressources, de manière à réaliser un objectif donné. Une solution de ce problème détermine l'ordre et le calendrier d'exécution des tâches en fixant leur date de début. On trouve dans [Le Pape 91] : "Le problème d'ordonnancement consiste à déterminer *qui* exécutera *quelles* actions et *quand*" (une action peut être vue comme une tâche à laquelle on affecte une ressource). Les problèmes d'ordonnancement se rencontrent dans des domaines aussi variés que nombreux. Citons par exemple : l'informatique (où les tâches représentent les programmes et les ressources sont les processeurs ou la mémoire), la gestion de grands projets ou l'industrie manufacturière (conduite d'atelier de fabrication), etc. Bien que notre secteur d'application privilégié soit plutôt l'industrie manufacturière, notre travail est suffisamment générique pour s'appliquer à d'autres domaines. Les caractéristiques du problème d'ordonnancement peuvent dépendre du contexte ; toutefois, les données de base portent essentiellement sur les tâches, les ressources et les contraintes, définies tour à tour ci-après.

I.2.2 Tâches – Ressources

I.2.2.1 Les tâches

Une *tâche* est, par définition, une activité élémentaire localisée dans le temps par une date de début ou de fin et dont la réalisation nécessite un certain intervalle de temps appelé *durée* ; par ailleurs, les tâches nécessitent l'utilisation de ressources. Les tâches sont, le plus souvent, liées entre elles par des conditions d'origines diverses. Si tel n'est pas le cas, elles sont dites *indépendantes*. Une tâche est *morcelable* ou *non morcelable*. Dans le premier cas, on dit que la *préemption* est autorisée (*problème préemptif*) ; l'exécution de la tâche peut s'effectuer par morceaux, sur une ou plusieurs machines, avec ou sans mémoire du traitement déjà effectué.

I.2.2.2 Les ressources

Une *ressource* est un moyen technique ou humain destiné à être utilisé pour la réalisation d'au moins une tâche et disponible en quantité limitée ou non. En utilisant la terminologie introduite dans [Słowiński 81] et [Węglarz 81], les ressources sont définies comme *renouvelables*, *consommables* ou *doublement contraintes*. Une ressource est renouvelable si, après avoir été allouée à une tâche, elle redevient disponible pour les autres (machine, processeur, convoyeur, ...). La quantité

de ressource utilisable à chaque instant peut être limitée mais la ressource reste disponible indéfiniment. En revanche, une ressource est consommable si elle n'est plus disponible, ou si la quantité disponible a diminué, après avoir été allouée à un certain nombre de tâches (matières premières, budget attribué, ...). La consommation globale au cours du temps d'une ressource consommable est limitée. Une ressource est doublement contrainte lorsque son utilisation instantanée et sa consommation globale sont toutes deux limitées (l'argent est peut-être le meilleur exemple de ressource doublement contrainte). On distingue de plus des ressources *disjonctives* utilisables par une seule tâche à la fois et des ressources *cumulatives* qui peuvent être utilisées simultanément par plusieurs tâches . On parlera selon les cas de *problème disjonctif* ou de *problème cumulatif.*

I.2.3 Les contraintes

La typologie des contraintes dans les problèmes d'ordonnancement peut se faire selon plusieurs critères de classification. Cette classification peut être spécifique à une application particulière ou générique. M.S. Fox [Fox 87] énumère exhaustivement et formalise les contraintes rencontrées dans une unité de production réelle (Westinghouse Turbine Component Plant). Il les classe en cinq catégories :

- les contraintes d'organisation, englobant le respect des dates d'échéance et des niveaux d'en-cours, les quantités de ressource à maintenir pour la bonne exécution des opérations, la réduction des coûts matériels et les objectifs de productivité (production maximale, ...) ;
- les contraintes physiques spécifient les caractéristiques qui limitent la fonctionnalité de l'atelier (capacité des diverses ressources, temps de préparation ou d'exécution d'une opération à réaliser, temps de transport jusqu'à la machine, ...) ;
- les contraintes causales regroupant les précédences imposées, les types et les quantités de ressource requises par les opérations (précisant ainsi les gammes de fabrication) ;
- les préférences (par exemple sur les machines) liées à des notions de coût ou de qualité ;
- les contraintes de disponibilité d'une ressource selon qu'elle est affectée à une opération ou pas, qu'elle est détériorée ou hors service.

La représentation des contraintes qu'il emploie, mentionne quelle contrainte est utilisée, quand et comment elle est utilisée. Les décisions s'appuient sur une hiérarchisation des contraintes, qui sont rangées par classes d'importance à l'aide d'une valeur, fournie par l'utilisateur, comprise entre 0 et 1 (0 signifie que la contrainte ne doit pas être considérée). Pour chaque hypothèse concernant le choix

d'une opération, d'une ressource ou d'une fenêtre temporelle pour une opération, on établit le "degré de satisfaction" de chacune des contraintes. Les coefficients d'importance sont combinés à ces degrés de satisfaction de manière à évaluer chaque hypothèse et conserver les meilleures.

C. Le Pape [Le Pape 88] s'aide de cette classification pour définir les contraintes absolues et les préférences, qui ont un sens plus étendu que dans [Fox 87]. Les contraintes absolues d'ordonnancement ou restrictions (il s'agit essentiellement de contraintes causales, physiques, de disponibilité de ressources) délimitent l'espace des ordonnancements *admissibles* c'est-à-dire effectivement réalisables dans l'atelier. Les préférences, appelées contraintes relaxables (dates d'échéance, regroupement d'opérations par ressource, ...) déterminent la qualité d'un ordonnancement. Ce sont des critères qu'il faut s'efforcer de respecter, dans la mesure du possible, en tenant compte des restrictions.

Dans [Roy 70], les contraintes sont formalisées grâce à la notion d'*inégalités de potentiels* (une définition approfondie et des exemples de contraintes mises sous cette forme sont présentées au paragraphe II.2.1). Par cette formulation, on peut représenter :

- les contraintes de localisation temporelle : la tâche i ne peut commencer avant telle date ou doit être achevée à telle autre,
- les contraintes de précédence : la tâche i ne peut commencer avant que la tâche j soit terminée.

D'autres contraintes, plus complexes, peuvent être prises en compte en considérant plusieurs inégalités de potentiels liées entre elles :

- par un ET, les contraintes *conjonctives*, lorsque les inégalités de potentiels doivent être vérifiées conjointement (cf. II.2.2) : la tâche i commence dès que la tâche j est terminée (cette contrainte est une conjonction de : ⟨ la tâche i ne peut commencer avant que la tâche j soit terminée ⟩ ET ⟨ la tâche i ne peut commencer après que la tâche j soit terminée ⟩). Un problème d'ordonnancement, uniquement soumis à des contraintes conjonctives, est de complexité polynomiale.
- par un OU, les contraintes *non conjonctives*, lorsqu'il suffit que l'une au moins des inégalités soit vérifiée. Ces contraintes, qui représentent par exemple des conflits pour l'utilisation de moyens, introduisent un aspect combinatoire de par la présence de choix pour la satisfaction de contraintes. On distingue :
  - les contraintes disjonctives : elles imposent la réalisation non simultanée d'une paire de tâches. Elles représentent un cas particulier des contraintes suivantes ;

- les contraintes cumulatives que l'on rencontre lorsqu'un moyen existe en une certaine quantité et que chaque tâche en utilise un certain nombre. Ces contraintes peuvent être représentées par des ensembles non conjonctifs d'inégalités de potentiels (cf. III.1.1.1).

  Remarque : le caractère disjonctif ou cumulatif des contraintes est généralement lié à la nature des ressources utilisées. Ainsi dans l'ordonnancement d'atelier (cf. I.3), une ressource est une machine et ne peut accueillir qu'une seule tâche à la fois.

P. Baptiste [Baptiste 85] cite en plus des contraintes précédentes, les contraintes ensemblistes qui sont des contraintes de regroupement de pièces lors de la prise en compte des temps de réglage des machines.

Si la fonction ordonnancement est située dans une structure de décision multi-niveaux, le niveau duquel sont vues les contraintes peut être un élément de classification [Erschler 76]. On parle de contraintes internes lorsqu'elles sont locales au niveau considéré et de contraintes externes lorsqu'elles sont l'expression des objectifs du niveau supérieur.

- Les contraintes internes sont principalement de deux types :
  - les contraintes de cohérence technologique qui décrivent les relations d'ordre entre les différentes tâches, la précédence par exemple ;
  - les contraintes d'utilisation des ressources qui expriment la nature et la quantité des ressources utilisées par les tâches, ainsi que les caractéristiques d'utilisation de ces ressources.
- Les contraintes externes, également de deux types :
  - les contraintes de temps alloué relatives aux dates limites des tâches (début au plus tôt, fin au plus tard) ou à la durée totale d'un projet (date échue) ;
  - les contraintes de disponibilité des ressources qui précisent la nature et la quantité des ressources disponibles pour l'exécution des diverses tâches ou encore leur durée de fonctionnement.

Les classifications présentées ci-dessus ne sont pas indépendantes. Une relation de précédence peut être vue comme une contrainte causale, une contrainte potentielle ou une contrainte interne. Ces classifications sont essentiellement dépendantes du niveau de représentation désiré pour un problème ou de la structuration du système global (processus de décisions multi-niveaux).

Le tableau I.1 énumère une liste de contraintes et leur catégorie selon les auteurs.

<table>
<tr><th>Contrainte</th><th>[Fox 87]</th><th>[Le Pape 88]</th><th>[Roy 70] - [Baptiste 85]</th><th>[Erschler 76]</th></tr>
<tr><td>dates d'échéance</td><td>organisation</td><td>préférence</td><td>potentielle</td><td>externe</td></tr>
<tr><td>temps de transport et préparation</td><td rowspan="2">physique</td><td>restriction -préférence</td><td colspan="2" rowspan="2">(⋆)</td></tr>
<tr><td>durée</td><td rowspan="5">restriction</td></tr>
<tr><td>précédences</td><td>causale</td><td>potentielle</td><td>interne</td></tr>
<tr><td>capacité des ressources</td><td>physique</td><td rowspan="2">de ressource</td><td>externe</td></tr>
<tr><td>ressources requises</td><td>causale</td><td>interne</td></tr>
<tr><td>en-cours</td><td>organisation</td><td>préférence</td><td></td><td></td></tr>
<tr><td>lotissements</td><td>physique</td><td></td><td>ensembliste</td><td></td></tr>
</table>

Tableau I.1

(⋆) Dans les deux dernières colonnes du tableau I.1 (ainsi que dans notre approche), la durée et les temps de transport et de préparation ne sont pas perçus comme des contraintes, mais comme des caractéristiques des tâches.

Pour certaines approches, il est intéressant d'établir une hiérarchie parmi les contraintes, en fonction de leur caractère plus ou moins strict. Il faut cependant noter que cette hiérarchie dépend fortement du contexte d'application. En effet, si l'on considère par exemple une contrainte de limitation de capacité des ressources, elle peut être considérée comme stricte s'il n'est pas possible d'utiliser des ressources complémentaires, ou relaxable si le niveau de ressource peut être modulé (heures supplémentaires, sous-traitance, etc).

L'approche retenue dans ce travail ne suppose aucune hiérarchie a priori sur les contraintes prises en compte. L'analyse est effectuée en considérant les contraintes comme si elles étaient strictes, de façon à en évaluer le réalisme. Si des contraintes doivent être relaxées, alors une hiérarchisation de ces contraintes peut être utilisée.

I.3 Typologie des problèmes d'ordonnancement

Le premier point de choix, en vue d'établir une classification des problèmes d'ordonnancement, peut se poser au niveau de la nature des tâches. On distingue ainsi (cf. I.2.2.1) : (1) les problèmes préemptifs [Słowiński 82], qui se séparent en deux classes selon que l'interruption des tâches s'effectue avec ou sans mémorisation du travail partiellement accompli, et (2) les problèmes non-préemptifs.

Dans le cadre des problèmes non-préemptifs, B. Roy propose une classification selon la nature des inconnues [Roy 70]. Une tâche est décrite par trois sortes de caractéristiques (T,D,W) qui représentent respectivement les dates de début des tâches, leur durée, les ressources utilisées (nature et quantité). Il dégage ainsi cinq types de problèmes en fonction de la famille de caractéristiques inconnues : les problèmes de type T, W, (T,D), (T,W) et (T,D,W). Les problèmes du type W, où les caractéristiques concernant les ressources sont inconnues, constituent les problèmes d'affectation. Dans le cas où la durée ne dépend que de la ressource utilisée, les problèmes du type (T,D,W) peuvent être rangés dans le type (T,W).[1] Les problèmes du type T où seules les dates de début des tâches sont inconnues constituent ce que l'on peut appeler les problèmes d'ordonnancement *purs*.

Ce type de classification, bien qu'étant assez général, ne fait pas apparaître tous les problèmes qu'inspire la liste des contraintes évoquées au paragraphe précédent. Ainsi, lors de lotissements, les temps de préparation peuvent être vus comme des tâches, dont leur durée dépend à la fois de tâches antérieures et de tâches suivantes. Il s'agit là d'une catégorie de problèmes ne rentrant pas dans le cadre de la classification (T,D,W).

Au sein des problèmes du type T, on peut établir une classification plus fine. On distingue :

[1] Sur la résolution de problèmes d'affectation, le lecteur peut consulter les travaux suivants où les durées des tâches sont variables :

- [Węglarz 81] propose un modèle préemptif dans lequel les tâches sont représentées par une fonction qui traduit la relation entre la vitesse d'exécution de la tâche et la quantité de ressource attribuée. Il propose des algorithmes utilisant ce modèle dans le cas de fonctions vitesse/quantité de ressource attribuée, strictement convexes, strictement concaves et linéaires.

- Dans le cas de fonctions linéaires, [Leachman et al. 90] développe une étude basée sur le taux de ressource appliqué à une tâche à un instant donné et déterminé par un indice de performance, appelé *intensité* de la tâche. Cette intensité varie de manière continue entre une borne inférieure et une borne supérieure.

- le cas où l'on ne prend pas en compte les ressources, appelé problème central de l'ordonnancement,
- le cas général où les ressources sont prises en compte.

Dans le cas général, deux classifications parallèles sont alors possibles suivant le type de ressources utilisées dans le problème (cf. I.2.2.2). On a d'une part :

- les problèmes à ressources renouvelables,
- les problèmes à ressources consommables

et d'autre part :

- les problèmes à ressources disjonctives ou problèmes disjonctifs,
- les problèmes à ressources cumulatives ou problèmes cumulatifs.

Les problèmes à ressources renouvelables et disjonctives sont souvent appelés *problèmes d'atelier*. Dans ce cas particulier, on parle indifféremment de tâche ou d'opération, de ressource ou de machine. Une opération nécessite une machine pour sa réalisation. Les opérations relatives à la fabrication d'un produit doivent généralement être réalisées en séquence et constituent un *travail* ou "job".

Nos travaux se rattachent essentiellement aux problèmes du type T. Quelques développements touchant aux problèmes (T,D), (T,W) et (T,D,W) seront néanmoins évoqués. Les ressources utilisées sont de type renouvelables et cumulatives (les problèmes d'atelier sont donc un cas particulier d'application de notre étude).

En se limitant aux problèmes d'atelier, diverses classifications ont été présentées au travers d'articles de synthèse [Graves 81], [Lawler et al. 89], [Gotha 91] et d'ouvrages [Muth et Thompson 63], [Baker 74], [Conway et al. 67]. Dans ce dernier, un problème d'ordonnancement est décrit par quatre types d'information :

- les travaux et les opérations à effectuer ;
- le nombre et le type de machines ;
- l'ordre d'utilisation des machines (lié à la *gamme de fabrication*) ;
- le critère à optimiser.

Les principaux critères retenus sont les minimisations de la durée totale d'un ordonnancement ("makespan"), du plus grand retard, du retard moyen pondéré ou des stocks de produits en cours de fabrication (*en-cours*). La nature des gammes est donnée par le type d'atelier considéré. Dans la littérature, on distingue les problèmes à ressource unique (une machine ou machines à exemplaires multiples) et des problèmes à ressources multiples dont les problèmes d'atelier font partie. Au sein de ce dernier type de problèmes, et en établissant une classification selon la nature des gammes de fabrication, on trouve, du plus au moins contraint :

- le *flow-shop* ou atelier à cheminement unique : la séquence des machines sur lesquelles doit passer un produit est fixe (chaîne de fabrication) ;

- le *job-shop* ou atelier à cheminements multiples : chaque produit ou famille de produits possède une gamme spécifique (le flow-shop correspond à un cas particulier de job-shop où les gammes sont identiques pour chaque produit) ;

- l'*open-shop* ou atelier à cheminements quelconques : l'ordre d'exécution des opérations d'un travail est libre (cellule flexible).

Les problèmes diffèrent également par la nature d'arrivée des produits dans l'atelier. On distingue le problème *statique* où les travaux arrivent simultanément (dates de début au plus tôt identiques) dans l'atelier qui est immédiatement disponible pour leur exécution, et le problème *dynamique* où les travaux arrivent de façon continue (dates de début au plus tôt différentes).

Remarque : une autre définition de problèmes statique et dynamique apparaît dans [Erschler et al. 91a]. Le problème statique consiste à générer un plan de réalisation de l'ensemble des travaux présents dans l'atelier, sur la base de données prévisionnelles. Ce plan peut ensuite servir de guide pour définir des procédures de décision en temps réel tout en tenant compte de l'état des ressources et des arrivées de nouveaux travaux (contexte dynamique). Pour le problème dynamique, on trouve souvent les appellations d'*ordonnancement temps-réel* ou de *pilotage* ; le problème statique est associé à *ordonnancement prévisionnel.*

Compte tenu de la typologie présentée ci-dessus, le tableau I.2 et la figure I.2 permettent de situer notre travail.

nature des tâches	préemption interdite
inconnues	problèmes du type T
critères	pas pris en compte
ressources	prises en compte

Tableau I.2

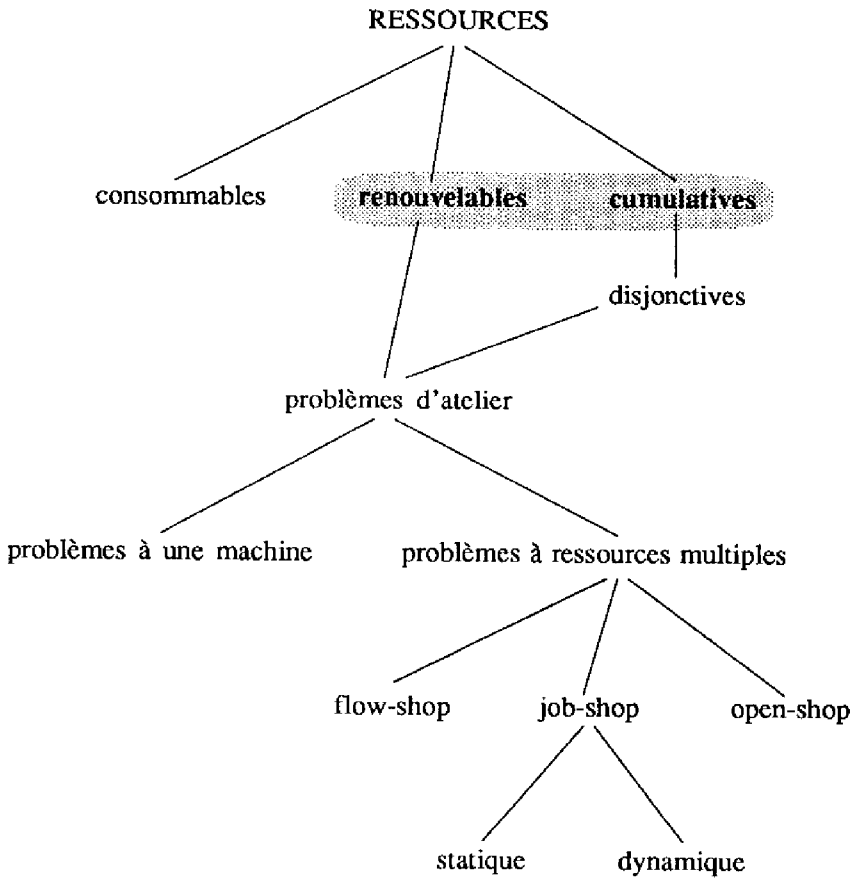


Figure I.2 : Situation de notre travail

I.4 Méthodes de résolution

I.4.1 Approches issues de la Recherche Opérationnelle (R.O.)

Ces méthodes de résolution visent à optimiser un certain critère d'un problème d'ordonnancement (minimisation de la durée totale, des retards, des en-cours, des coûts, maximisation du débit, etc). Elles sont communément divisées en deux champs : les *méthodes d'optimisation*, encore appelées méthodes exactes car elles garantissent l'obtention d'une solution optimale pour le critère et les *méthodes heuristiques*, qui visent à générer une solution satisfaisante vis-à-vis d'un ou plusieurs critères.

I.4.1.1 Les méthodes d'optimisation

Elles consistent à trouver la (ou les) meilleure(s) solution(s) du problème vis-à-vis d'un critère donné.

I.4.1.1.a Problème central

On rappelle que dans le problème central de l'ordonnancement, on ne prend pas en compte les contraintes de ressource : seules les contraintes de succession et de localisation temporelle sont considérées. Les tâches soumises à ces contraintes doivent être ordonnancées en une durée minimale. La résolution optimale de ce problème est obtenue grâce à des méthodes de chemin critique (M.P.M. ou C.P.M./P.E.R.T.)[2] [Kaufmann et Desbazeille 64][Battersby 67][Dibon 70]. Ce sont des méthodes polynomiales qui fournissent un ordonnancement au plus tôt et au plus tard par la recherche de plus longs chemins sur le graphe conjonctif associé au problème. On peut déduire des *marges* qui représentent les degrés de liberté dont on dispose pour fixer le début d'exécution des tâches, compte tenu de l'objectif de minimisation de la durée totale. On caractérise ainsi l'*ensemble* des solutions qui minimisent la durée totale de réalisation. Cette formulation est bien adaptée au problème de planification de projet que l'on rencontre lors de la construction ou la gestion d'unité importante (bâtiment, navire, mission spatiale, ...).

I.4.1.1.b Problème général

Dans le cadre plus général des problèmes à contraintes de ressources, certains problèmes sont résolus de manière polynomiale en raison de l'existence de résultats spécifiques associés à ces problèmes. Lorsque ces résultats n'existent pas, on doit faire appel à des méthodes générales pour la résolution des problèmes.

[2] M.P.M. : Méthode des Potentiels Metra – C.P.M. : Critical Path Method – P.E.R.T. : Program Evaluation and Review Technic *ou* Program Evaluation Research Task.

Méthodes basées sur des résultats spécifiques

Parmi la variété des problèmes d'ordonnancement (cf. paragraphe I.3), on peut construire des algorithmes spécifiques pour certains de ces problèmes. A titre d'exemple, nous présentons ci-dessous quelques problèmes pour lesquels des méthodes efficaces (polynomiales) ont été proposées (les procédures suivantes s'intéressent, sauf spécification contraire, au critère de minimisation de la durée totale). Cette présentation de quelques résultats n'est absolument pas exhaustive tant par les travaux cités pour ces problèmes particuliers que par la singularité des problèmes rencontrés :

- problème à une machine : les résultats les plus connus sont ceux proposés dans [Smith 56] pour minimiser les en-cours et dans [Jackson 55] pour minimiser le plus grand retard ;
- problème à m machines identiques : lorsque la préemption est autorisée, un algorithme très simple minimisant la durée totale est proposé dans [Mc Naughton 59]. Coffman et Graham ont donné un algorithme optimal pour le cas non-préemptif, deux machines et tâches de même durée [Coffman et Graham 72] ;
- flow-shop – 2 machines : le problème est résolu par la célèbre règle de Johnson [Johnson 54] ;
- job-shop – 2 machines : une extension de la règle précédente a été proposée par Jackson pour la résolution optimale de ce problème [Jackson 56] ;
- job-shop – 2 travaux : une résolution graphique du problème fut proposée dans [Akers 56] grâce à un plan xOy où un axe représente le temps pour chaque travail. Ses travaux furent repris par P. Brücker qui présenta un algorithme polynomial en permettant la résolution [Brücker 88]. Pour trouver l'ordonnancement optimal, il recherche le plus court chemin[3] dans le plan sur lequel sont disposés des rectangles infranchissables représentant l'utilisation des machines par les deux travaux dans l'ordre de la gamme.

Méthodes génériques

La programmation mathématique peut être utilisée comme un outil général d'optimisation du problème d'ordonnancement mis sous forme algébrique. On peut employer, par exemple, la programmation linéaire qui s'avère surtout intéressante lorsque les tâches sont morcelables et indépendantes.

Pour certains problèmes de complexité exponentielle (problèmes *NP-difficiles*), on appliquera la programmation dynamique, qui est une procédure très générale

[3] Ce chemin est constitué de segments horizontaux (exécution du $travail_1$), verticaux (exécution du $travail_2$) ou obliques à 45° (exécution simultanée des deux travaux).

d'optimisation, mais qui possède l'inconvénient d'être trop énumérative pour des problèmes de taille importante (minimisation de la somme pondérée des retards dans le problème à une machine [Schrage et Baker 78], minimisation de la durée totale dans le job-shop – 2 travaux [Szwarc 60]).

Pour d'autres, on peut également avoir recours aux méthodes arborescentes qui fournissent plus d'informations sur l'environnement de la solution optimale que n'en donnent les méthodes de programmation mathématique ; elles sont donc d'utilisation plus souple et se rapprochent ainsi davantage des approches issues de l'intelligence artificielle (cf. I.4.2). Une procédure arborescente très souvent utilisée est la procédure d'exploration par séparations et évaluations progressives (P.S.E.P. ou "Branch and Bound") ; comme la programmation dynamique, la P.S.E.P. est une méthode d'énumération implicite car elle permet de trouver la solution optimale sans avoir à les énumérer toutes. A chaque P.S.E.P., correspond la définition d'un principe de séparation et d'une fonction d'évaluation permettant l'exploration de l'arborescence. Ces méthodes ont largement été appliquées dans le domaine de l'ordonnancement pour la résolution de problèmes NP-difficiles : la minimisation de la durée totale du flow-shop général [Ashour 70][Mc Mahon et Burton 67][Baker 75][Lageweg et al. 78] ou du job-shop général [Balas 69][Lageweg et al. 77][Schrage 70]. Remarquons qu'au contraire du flow-shop, le job-shop se prête mal à des caractérisations analytiques.[4] Aussi la plupart des méthodes arborescentes proposées pour sa résolution sont basées sur une modélisation par graphe disjonctif [Carlier et Pinson 89] : les auteurs fondent leur méthode sur des propositions amenant des sélections immédiates de disjonctions et limitant par conséquent la recherche arborescente. Quand deux conditions sont satisfaites, la première ou la dernière tâche d'un sous-ensemble d'opérations effectuées sur une machine peut être déterminée.

D'autres théories, comme celle des inégalités valides ou des représentations polyhédrales, trouvent des applications dans les problèmes d'ordonnancement (comme le célèbre problème du "voyageur de commerce" par exemple) (voir [Nemhauser et Wolsey 88] et [Queyranne 88]).

I.4.1.1.c Conclusion

L'intérêt de telles démarches est incontestable quant au résultat obtenu. Elles présentent néanmoins des inconvénients :

- pour des raisons de complexité, elles ne permettent pas toujours de prendre en compte toutes les données du problème : elles ont ainsi des difficultés à

[4] En effet, de par la présence d'une gamme unique dans un problème de flow-shop, un certain nombre de propriétés ont pu être déduites rendant ainsi plus aisée sa caractérisation. Le job-shop se différencie du flow-shop par le fait que lorsqu'une opération se termine sur une machine, toutes les opérations réalisables doivent être prises en compte pour déterminer la prochaine opération à exécuter [Bellman et al. 82].

traiter des situations réalistes. La plupart des problèmes sont en effet NP-difficiles et on ne peut les résoudre en un temps polynomial. C'est le cas, par exemple, de la minimisation de la somme pondérée des retards dans le problème à une machine [Rinnooy Kan 76] ;

- le problème d'optimisation est toujours engagé vis-à-vis d'un certain critère alors que généralement plusieurs critères, souvent antagonistes, sont à considérer simultanément ;

- le fait de trouver une solution unique peut être perçu comme un désavantage, essentiellement face à des perturbations ou des remises en cause de décisions (notion de flexibilité dans le choix d'alternatives. Cf. paragraphe I.5.1).

C'est principalement pour pallier le premier inconvénient que sont utilisées les heuristiques.

I.4.1.2 Les méthodes heuristiques

Elles consistent à obtenir une "bonne" solution du problème avec une durée de recherche acceptable.

I.4.1.2.a Méthodes dérivées des méthodes d'optimisation

Ce sont des algorithmes polynomiaux qui permettent d'approcher, parfois d'obtenir, la solution optimale. Ces méthodes sont largement employées en ordonnancement d'atelier ([Gupta 71][Proust et al. 87]... pour le flow-shop), pour le problème du voyageur de commerce ([Lin et Kernighan 71][Carlier 90]...), etc. Pour obtenir une solution approchée d'un problème de job-shop, on peut également utiliser des méthodes de décomposition [Yamamoto 77][Portmann 88] [Meguelati 88]. Le problème principal est décomposé en sous-problèmes que l'on résout séparément : la solution globale est obtenue à partir des solutions des sous-problèmes.[5]

I.4.1.2.b Approche par règles de priorité [Canals 86]

Il s'agit de construire un ordonnancement en simulant l'état d'avancement des opérations. Les opérations en attente de la libération d'une ressource sont stockées dans une file d'attente. Les conflits d'utilisation de la ressource, pouvant apparaître lorsque celle-ci se libère, sont arbitrés par diverses règles de priorité (FIFO, SPT, LPT, MWKR, ...) qui permettent d'imposer un ordre de passage des opérations en attente devant la ressource. Le résultat recherché peut être soit l'établissement d'un ordonnancement par la détermination des dates de début des tâches sur les

[5]Si les sous-problèmes obtenus après décomposition sont indépendants, l'application d'une méthode d'ordonnancement aux différents sous-problèmes donne la même valeur du critère global en un temps inférieur ; sinon, il est nécessaire de mettre en œuvre une procédure permettant de gérer les liens résiduels entre sous-problèmes.

machines (le progiciel TZAR II par exemple [Tzar-II 83]), soit l'étude des performances des règles de priorité (en mesurant les écarts-types, les temps de cycle, les niveaux moyen ou maximum des stocks dans les files d'attente) en vue de les utiliser comme procédure de choix en temps réel.

I.4.1.2.c Méthodes de placement

Elles consistent à classer les différents travaux à réaliser (fabrication d'un produit) et à les sélectionner par ordre de priorité. Les opérations de chaque travail sont alors ordonnancées en tenant compte des opérations déjà placées sur les machines (le progiciel SAVEPLAN par exemple [Saveplan 84]).

I.4.1.3 Conclusion

Les approches précédentes sont efficaces pour un problème statique[6], mais elles manquent de souplesse vis-à-vis de l'évolution d'un atelier (pannes-machines, ordres contradictoires avec les prévisions, ...). A ce niveau, l'intelligence artificielle (I.A.) propose des techniques de structuration des connaissances et de "réordonnancement" afin de prendre en compte les modifications au cours de l'exécution d'un plan.

I.4.2 Approches issues de l'Intelligence Artificielle

Un grand nombre de logiciels utilisant certaines méthodes heuristiques sont implantés en milieu industriel. S'ils proposent des ordonnancements, en revanche les performances en termes de réduction des en-cours ou des temps de cycle et de respect des délais sont parfois très insuffisantes. D'autre part, ils ne permettent pas de prendre en compte de façon souple les contraintes liées à l'exploitation du système de production. Le problème d'ordonnancement prévisionnel apparaît ainsi comme un domaine d'application possible des techniques d'I.A.. Le but principal est de créer des systèmes à bases de connaissance permettant de le résoudre tout en laissant une certaine flexibilité d'utilisation permettant de s'adapter à d'éventuelles modifications de production. D'après M.S. Fox [Fox 87], deux axes de recherche de l'I.A. ont un impact important sur les problèmes d'ordonnancement :

- la recherche en planification ;
- l'analyse des contraintes.

Les méthodes courantes utilisées sont heuristiques. Fox s'est intéressé à la construction de séquences d'opérations qui satisfont autant de contraintes que possibles, dans un problème de type job-shop. Le problème d'ordonnancement intéresse en effet l'I.A. sur plusieurs points :

[6] au sens de génération de plan sur la base de données prévisionnelles

- la représentation des connaissances pour la modélisation d'un système de production ;
- le développement d'une sémantique pour la représentation des contraintes ;
- l'intégration des contraintes dans le processus de recherche et leur utilisation pour borner la génération de solutions ;
- la relaxation des contraintes quand une incohérence apparaît ;
- l'analyse des interactions entre contraintes de façon à repérer les solutions pauvres.

I.4.2.1 Systèmes dédiés au problème d'ordonnancement

Dans [Steffen 86] et [Le Pape 88], différents systèmes sont présentés avec la méthodologie utilisée et les perspectives d'utilisation. On peut citer notamment :

- ISIS (Intelligent Scheduling and Information System) [Fox et Smith 84]. Dans ce système, toutes les contraintes sont identifiées et modélisées dans un langage de représentation[7]. Une décomposition par lots a été adoptée dans ISIS dans le but de casser la combinatoire du problème. L'ordonnancement de l'atelier est obtenu en générant un ordonnancement pour chacun de ces lots. Les décisions sont prises de façon progressive, sans retour arrière, en s'appuyant sur une hiérarchisation des contraintes (cf. paragraphe I.2.3) ;
- OPIS (OPportunistic Intelligent Scheduler) [Le Pape 86]. Devant la complexité d'ISIS, un nouveau système a été envisagé. Une première version OPIS0 construit l'ordonnancement des ressources critiques avant d'effectuer un ordonnancement "lot par lot" comme celui utilisé dans ISIS. La dernière version OPIS1 génère un ordonnancement de façon plus opportuniste en travaillant, soit par lot, soit par ressource, en fonction d'une analyse des conflits résultant de l'antagonisme entre contraintes. La stratégie de construction de l'ordonnancement est ainsi définie de façon dynamique ;
- SOJA (Système d'Ordonnancement Journalier d'un Atelier) [Le Pape et Sauve 85]. Il réalise l'ordonnancement prévisionnel d'une journée de travail en sélectionnant les opérations à effectuer, puis en les ordonnançant. L'ordonnancement est donc généré en s'appuyant sur :
  - un module de sélection qui détermine un "portefeuille" d'opérations à ordonnancer et précise pour chaque opération la machine qui lui est allouée,

[7] S.R.L. : Schema Representation Language.

- un module d'ordonnancement qui détermine des dates de début des opérations sélectionnées, de manière à satisfaire les contraintes du problème. Deux types de contraintes sont distinguées : des contraintes absolues et des préférences. La progression vers la solution est itérative ; à chaque étape, on choisit une contrainte à satisfaire exprimée sous forme d'inégalité temporelle. Lors de la détection d'une contradiction, les derniers faits affirmés sont annulés et les données sont restaurées (à la différence d'ISIS qui ne revient pas sur ses décisions).

- SONIA [Collinot et al. 88], comme SOJA, est basé sur une sélection des opérations, mais travaille de façon plus opportuniste et plus flexible. Cette sélection s'opère par l'intermédiaire de deux modules : le premier sélectionne des opérations en fonction de la capacité des différentes machines (variante du module utilisé dans OPIS) ; le second sélectionne une partie de la gamme de fabrication d'un lot qui lui est désigné. Après chaque sélection, une variante du module d'ordonnancement employé dans SOJA peut être utilisée. En cas de conflits, il est possible de relaxer des contraintes en permutant deux opérations qui utilisent la même machine ou en repoussant des dates de fin d'opérations ou la date d'échéance. Il peut ainsi être utilisé de façon dynamique en réalisant un ordonnancement "glissant" qui prend en compte l'état réel du système.

L'organisation de ces systèmes est représentée schématiquement sur la figure I.3.

I.4.2.2 Différents types de connaissances

Ces approches visent à aborder des problèmes réalistes tout en conservant une certaine rigueur vis-à-vis de la satisfaction d'objectifs globaux. Toutefois, ces systèmes sont gros consommateurs de temps C.P.U.. Une des raisons est l'intégration d'un seul type de connaissance au niveau conceptuel du système (ISIS utilise un langage de représentation des connaissances[8] basé sur des schémas composés d'attributs (objets, travaux, durées, opérateurs, ressources, ...) et de méta-information définissant la sémantique). La connaissance statique qui correspond aux données de l'atelier s'oppose à la connaissance dynamique qui fournit des méthodes dans le but de déduire des solutions intéressantes. Une des difficultés réside dans la diversité des connaissances dynamiques à prendre en compte pour la résolution du problème. Trois composantes sont par exemple mises en avant dans [Bel et al. 89] :

1. des connaissances *théoriques* relatives à la gestion du temps et des ressources ;

2. des connaissances *empiriques* issues de l'évaluation de procédures heuristiques par simulation (règles de priorité) ;

[8] S.R.L.

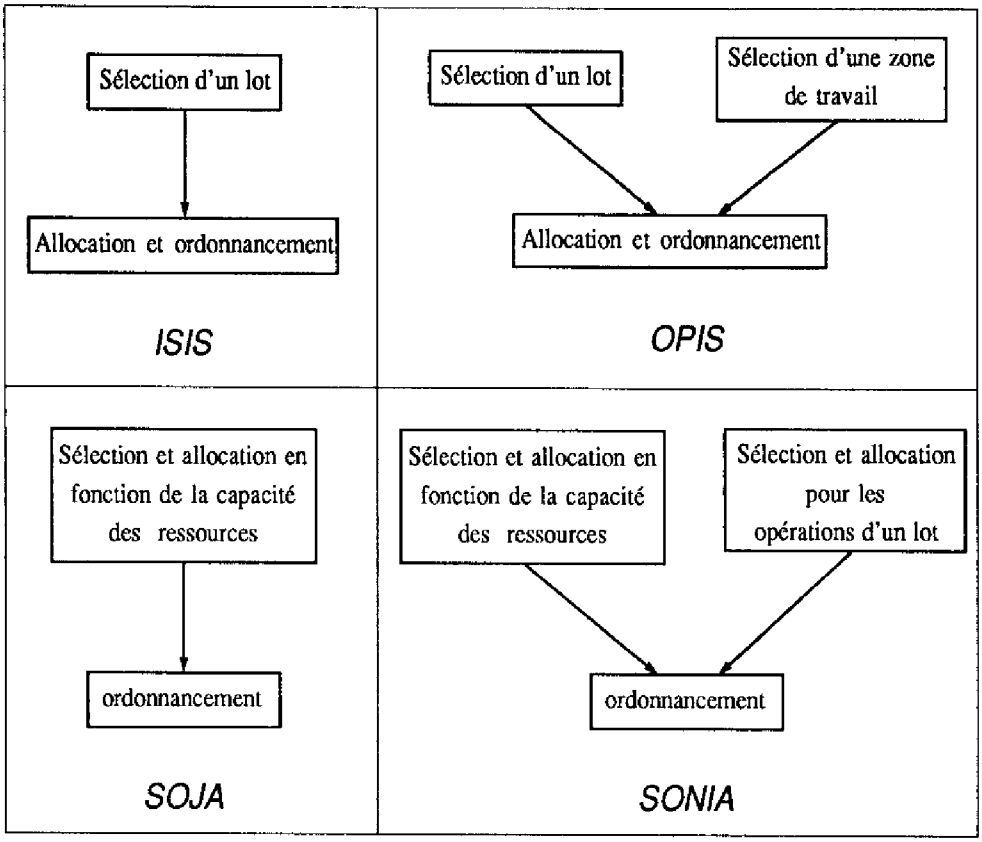


Figure I.3 : Sélection, allocation et ordonnancement dans quatre systèmes (d'après [Le Pape 88])

3. des connaissances *pratiques* spécifiques d'une application, fournies par les experts de cette application, s'il en existe. Ils renseignent par exemple, sur les contraintes technologiques ou d'origine humaine à respecter.

Le problème d'ordonnancement lorsqu'il est vu comme un problème d'optimisation n'a essentiellement besoin que du premier type de connaissance. Il n'est pas forcément bien adapté à des applications réalistes, les demandes d'un expert au sens d'un objectif de production étant, pour la majeure partie, multi-critères. Compte tenu de ces différents types de connaissances, le système OPAL (Ordonnancement Prévisionnel d'Ateliers) a été développé [Bel et al. 87][Bensana 87]. Il fait appel à plusieurs modes de représentation des connaissances.

- La description de l'atelier, des gammes et de l'objectif de production se fait grâce à des objets structurés reliés par des liens d'héritage.

- Le premier type de connaissance est mis en œuvre à travers un processus d'analyse et de *propagation des contraintes*[9] qui constituent le noyau de ce système ; cet aspect, étroitement lié au sujet de ce travail, sera détaillé ultérieurement (paragraphe I.5.2).

- Les deux derniers types de connaissances peuvent être représentés par des règles de type "SI... ALORS" symbolisant des conseils à respecter (*il faut..., il ne faut pas..., il vaut mieux...*). La théorie des ensembles flous est utilisée de manière à éviter de réaliser des distinctions tranchantes entre les informations. Elle permet en ce sens de maîtriser le caractère imprécis de ce type de connaissance et plus particulièrement de modéliser la partie "SI" des règles.

- La description de la solution en cours de construction s'effectue par un graphe de précédence (cf. II.2.2) et est visualisée à l'aide d'un diagramme à barres ou diagramme de GANTT [Gantt 19] (cf. II.2.3).

I.4.2.3 Raisonnement temporel

Nombre de travaux en I.A. portent sur le traitement des connaissances temporelles. Parmi les domaines que recouvre ce raisonnement, on rencontre la planification qui est définie comme l'élaboration d'une suite d'actions destinées à faire passer le monde de l'état initial à un état final. Les systèmes de représentation du temps se distinguent par les entités temporelles qu'ils utilisent [Bestougeff et Ligozat 89]. Il y a les systèmes basés sur une structure ponctuelle [van Benthem 83] et ceux basés sur une structure d'*intervalles* [Allen 81]. Les derniers rendent plus facilement compte de la notion de durée que les premiers.

Sur la base du raisonnement temporel à structure d'intervalles, une étude est présentée par J.F. Rit [Rit 88] sur la propagation de contraintes pour la planification. Pour lui, la manière d'exécuter une tâche doit concilier :

- des informations symboliques telles que les relations de précédence, la définition des ressources et des opérations ;

- des informations numériques telles que la durée des tâches.

Rit rassemble ces deux aspects en un seul formalisme. Il propose un modèle théorique pour la définition et la propagation de contraintes temporelles numériques. Des événements liés entre eux par des contraintes de Allen[10] sont chacun

[9] La propagation de contraintes qui sera abordée au paragraphe II.2, consiste à dériver de nouvelles contraintes à partir des contraintes existantes ; elle détecte également des incohérences entre différents types de contraintes [Davis 87][Le Pape 91].

[10] Outre une relation de précédence stricte, les intervalles peuvent avoir une partie commune (chevauchement ou inclusion). J. Allen a ainsi proposé une énumération exhaustive de relations symboliques possibles entre deux intervalles (treize primitives au total dont la relation identité) et une logique de manipulation associée (voir [Allen 81][Allen 83]).

caractérisés par une plage de temps permise, représentée par une aire appelée "Domaine des Occurrences Possibles" (DOP) ; une occurrence est définie comme un intervalle durant lequel un événement se passe. Les occurrences rendues impossibles par les relations symboliques sont éliminées, entraînant par conséquent la réduction de DOPs. Son but est donc de modifier ces DOPs pour qu'ils soient compatibles avec les contraintes symboliques.

I.4.2.4 Conclusion

L'approche I.A. permet d'aborder le problème d'ordonnancement avec une grande souplesse d'utilisation face au comportement évolutif d'un atelier et d'augmenter la puissance de description des connaissances. D'un point de vue pratique, les techniques de programmation utilisées couramment en I.A., comme la programmation logique, s'adaptent bien au processus de raisonnement automatique. En dehors de ces avantages réels, les travaux réalisés dans ce cadre n'ont pas apporté de progrès déterminants dans l'analyse, la modélisation et la résolution des problèmes d'ordonnancement.

I.4.3 Approche par caractérisation de solutions admissibles

Le point commun des approches précédentes (mis à part la résolution du problème central) est qu'elles visent à générer une solution unique du problème d'ordonnancement considéré. Une autre approche consiste à caractériser non pas une solution unique mais un ensemble de solutions. Cette démarche peut être rapprochée des méthodes à chemin critique, à travers la notion de marge, sans s'intéresser pour autant à la minimisation d'un critère. Pour cela, les méthodes et techniques développées s'inscrivent dans la problématique de l'*Analyse Sous Contraintes* (A.S.C.) des problèmes d'ordonnancement [Erschler 76]. Sur cette base ont été développés un module prototype, MASCOT, mettant en œuvre les résultats théoriques de l'A.S.C. [Erschler et Esquirol 86][Esquirol 87], et un progiciel d'ordonnancement et de pilotage en temps réel, ORDO, commercialisé par la société SYSECA Temps Réel [Thomas 80][Le Gall 89].

Cette approche constitue notre problématique de recherche et est donc développée en détail en I.5. Une telle démarche vise à caractériser une famille d'ordonnancements respectant les contraintes du problème indépendamment de tout critère : cela explicite les degrés de liberté qui peuvent ensuite être utilisés pour prendre en compte d'autres aspects du problème.

I.5 Approche proposée

I.5.1 Flexibilité dans les systèmes de production [Erschler et de Terssac 88]

L'évolution des systèmes de production doit beaucoup au développement du concept de *flexibilité*. La flexibilité d'un système est caractérisée par sa capacité d'adaptation à des variations de ses conditions de fonctionnement, grâce à sa capacité de changements d'états tout en restant intègre. Cette évolution s'explique par :

- "...une évolution du marché vers une demande de plus en plus diversifiée et incertaine...";
- "...une évolution technologique vers des dispositifs de plus en plus polyvalents et rapidement configurables grâce à l'introduction de la micro-informatique notamment..." [Erschler et de Terssac 88].

Par rapport à la capacité à s'adapter à des perturbations, on peut distinguer deux types de flexibilité entre lesquels doit être établi un compromis :

1. la flexibilité *interne* : aptitude à s'adapter aux aléas affectant le système physique de production (pannes de machines, absentéïsme, ...)

2. la flexibilité *externe* : aptitude à s'adapter à certaines contraintes provenant des interactions entre le système de production et son environnement extérieur (variation de la demande des clients, ...).

Le concept de la flexibilité externe permet de situer le problème d'ordonnancement dans son contexte réel. Il s'agit d'un problème non isolé dépendant de décisions prises en dehors de son cadre de traitement. Dans une structure multi-niveaux (cf. I.1.2), la cohérence entre deux niveaux successifs est assurée en considérant les objectifs des décisions prises au niveau supérieur comme des contraintes pour le niveau inférieur. Le concept de flexibilité est étroitement lié d'une part à l'aspect incertain du futur et d'autre part aux contraintes : l'analyse des solutions admissibles revient à expliciter la flexibilité disponible compte tenu d'objectifs pris en compte sous forme de contraintes.

I.5.2 Problématique retenue

Parmi les différents types de connaissances dynamiques présentées au paragraphe I.4.2.2, on retient plus particulièrement les connaissances théoriques qui constituent le noyau générique de tout système d'ordonnancement.

La problématique retenue consiste à analyser le problème d'ordonnancement défini exclusivement en termes de contraintes sur le temps et les ressources. Les contraintes considérées constituent un sous-ensemble de celles mises en jeu dans ISIS : les contraintes de précédence entre tâches, les contraintes de dates limites (début au plus tôt, fin au plus tard) associées à certaines tâches, les contraintes cumulatives associées aux ressources. Au sens de leur importance, toutes les contraintes présentes sont banalisées. Une telle analyse sous contraintes peut conduire à s'intéresser à trois problèmes :

1. l'*existence* de solutions admissibles : il s'agit d'étudier la consistance de l'ensemble des contraintes (contraintes internes propres au niveau considéré et contraintes externes imposées par le niveau supérieur) ;

2. la *caractérisation* de solutions admissibles : l'ensemble des contraintes étant consistant, il s'agit de caractériser un ensemble de solutions respectant ces contraintes sans pour autant les énumérer. Ces solutions admissibles sont caractérisées en recherchant leurs propriétés locales sous forme de conditions d'admissibilité. Celles-ci concernent la mobilité des tâches sur l'axe temporel (marges) et leur permutabilité en cas de conflit : ces caractéristiques expriment les degrés de liberté disponibles pour prendre en compte d'autres connaissances ;

3. la *génération* de solutions admissibles : à partir des caractéristiques issues de l'étape précédente, il s'agit de générer une ou plusieurs solutions en prenant en compte d'autres connaissances.

Comme on va le voir au paragraphe suivant, l'A.S.C. s'intéresse essentiellement à la caractérisation des solutions. Elle peut cependant être relayée par un module heuristique afin de générer une solution ou constater l'insolubilité d'un problème. Le système d'ordonnancement prévisionnel OPAL déjà présenté, a été développé conformément à cette idée. Il intègre et fait coopérer des connaissances expertes issues de l'expérience et un module (constituant son noyau) comprenant des règles simplifiées d'A.S.C..

Dans un contexte dynamique, l'approche par caractérisation de solutions admissibles permet de proposer au *décideur* un choix d'actions satisfaisant les contraintes de temps et de ressource. Il a ainsi la possibilité de générer une solution en fonction de contraintes non modélisées comme ses préférences. Il possède également les degrés de liberté pour réagir en temps réel face au contexte perturbé de l'atelier. L'A.S.C. apparaît alors comme un outil d'*aide à la décision*.

I.5.3 Caractérisation des solutions admissibles

Le problème central est celui de la caractérisation des ordonnancements admissibles. En effet, la consistance du système de contraintes, donc l'existence de

solutions, est dévoilée par la cohérence des conditions d'admissibilité. De même, la prise de décisions qui intervient dans l'étape de génération peut prendre en compte les connaissances empiriques et/ou pratiques mentionnées précédemment dans le cadre défini par les conditions d'admissibilité. Ainsi la gestion du temps et des ressources peut s'appuyer sur un module qui cherche à caractériser les solutions admissibles. Pour caractériser l'ensemble de tous les ordonnancements admissibles, il faut rechercher des *conditions nécessaires et suffisantes* de l'admissibilité. J. Erschler a montré que c'était théoriquement possible par l'A.S.C. [Erschler 76], mais sa mise en œuvre fait apparaître un problème de complexité. Dans le but de limiter cette complexité, la recherche de conditions nécessaires et suffisantes est décomposée en deux axes qui ne garantissent qu'une caractérisation partielle des ordonnancements admissibles. Dans cette optique, deux démarches ont été développées au LAAS :

- [Demmou 77], [Thomas 80] et [Le Gall 89] décrivent des travaux de recherche des *conditions suffisantes* (C.S.) d'admissibilité, telles que tout ordonnancement satisfaisant ces conditions est admissible. Ils caractérisent ainsi, non pas l'ensemble des ordonnancements admissibles, mais un sous-ensemble particulier ;

- dans [Erschler 76] et [Esquirol 87], la recherche est axée sur des *conditions nécessaires* (C.N.) d'admissibilité que tout ordonnancement solution du problème doit satisfaire. Cette démarche n'aboutit pas non plus à la caractérisation de l'ensemble des ordonnancements admissibles, mais à celle d'un ensemble d'ordonnancements contenant tous les ordonnancements admissibles. Cette procédure a été mise en œuvre sous la forme d'un processus d'inférence [Erschler et Esquirol 86] : il active une base de règles de séquencement et d'actualisation[11] de dates limites qui modifient une base de faits représentant les caractéristiques séquentielles et temporelles des ordonnancements admissibles. L'application de ces règles constitue une simple déduction logique à partir des contraintes initiales.

Notre travail se situe dans le dernier axe : la recherche de conditions nécessaires d'admissibilité.

[11]Une *actualisation* correspond à la modification d'une valeur, résultant de la prise en compte de faits séquentiels et temporels. Cette modification est régie par des règles d'actualisation qui seront décrites au cours des chapitres III et IV.

I.5.4 Tableau récapitulatif

Le tableau I.3 permet de visualiser le problème étudié ainsi que la façon dont il est abordé.

<table>
<tr><td>niveau de décision</td><td colspan="2">gestion à court terme</td></tr>
<tr><td>domaine d'application</td><td colspan="2">industrie manufacturière, ...</td></tr>
<tr><td>tâches</td><td colspan="2">non morcelables</td></tr>
<tr><td>ressources</td><td colspan="2">renouvelables</td></tr>
<tr><td>contraintes</td><td colspan="2">potentielles et cumulatives</td></tr>
<tr><td>problèmes</td><td colspan="2">type T [Roy 70]</td></tr>
<tr><td>connaissance dynamique</td><td colspan="2">théorique</td></tr>
<tr><td rowspan="2">méthode utilisée</td><td colspan="2">aide à la décision</td></tr>
<tr><td>caractérisation des solutions admissibles</td><td>recherche de C.N. d'admissibilité</td></tr>
</table>

Tableau I.3

* *

*

Chapitre II

Concepts de base

II.1 L'intervalle temps-ressource

II.1.1 Définition

Un *intervalle temps-ressource* I (figure II.1) peut être défini par :

- ses *extrémités* ou *bornes temporelles* C_I et F_I désignant respectivement le début et la fin de l'intervalle ;
- ses *fonctions d'intensité de ressource* $Q_I^k(t)$ définies $\forall t \in [C_I, F_I]$ et $\forall k \in K_I$ où K_I est l'ensemble de ressources associé à l'intervalle I.

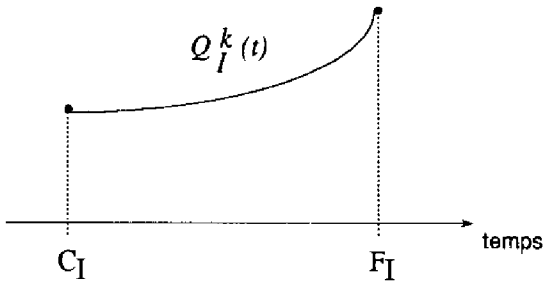


Figure II.1 : Intervalle temps-ressource

A partir de ces caractéristiques *instantanées*, des caractéristiques *intégrales* peuvent lui être également associées :

- sa *durée* : $D_I = \int_{C_I}^{F_I} dt = F_I - C_I$
- son *énergie de ressource* k : $W_I^k = \int_{C_I}^{F_I} Q_I^k(t).dt$

L'énergie W_I^k est exprimée en unités de [ressource $\times$ temps] comme des hommes-jour, kilowatts-heure, etc.

Remarque : il est possible d'envisager un intervalle temps-ressource I, non connexe, décrit par une suite de sous-intervalles I^1, I^2,... disjoints ($I^1 = [C_I^1, F_I^1]$, $I^2 = [C_I^2, F_I^2]$, ... avec $C_I^1 < F_I^1 < C_I^2 < F_I^2 < ...$) ; cette représentation peut, par exemple, s'avérer utile lors de la prise en compte d'horaires de travail dans un atelier de fabrication. Pour calculer les caractéristiques intégrales de l'intervalle temps-ressource, il faut appliquer l'intégration entre les bornes temporelles de chacun des sous-intervalles

$$(\int_{C_I}^{F_I} \cdot = \int_{C_I^1}^{F_I^1} \cdot + \int_{C_I^2}^{F_I^2} \cdot + ...).$$

II.1.2 Intervalles temps-ressource consommateurs et fournisseurs

La modélisation de problèmes d'ordonnancement peut s'appuyer sur deux types d'intervalles temps-ressource :

- les intervalles *consommateurs* : il s'agit d'intervalles dans lesquels le temps et les ressources sont requis. Ils sont associés à une tâche, définie par sa date de début, sa date de fin et les quantités de ressource qu'elle nécessite. En conséquence, on assimilera intervalle consommateur et tâche dans toute la suite de ce travail.

- les intervalles *fournisseurs* : il s'agit d'intervalles dans lesquels le temps et les ressources sont alloués. Sur de tels intervalles, on étudie l'effet de la consommation de temps ou de ressource sur l'ordonnancement d'un ou plusieurs intervalles consommateurs.

II.1.3 Caractéristiques connues et inconnues des intervalles temps-ressource

Un intervalle temps-ressource I est noté à l'aide de ses caractéristiques instantanées : $I = (C_I, F_I, \{Q_I^k(t)\})$. Toutefois, selon les problèmes abordés, il peut être défini à l'aide de ses caractéristiques instantanées et/ou intégrales. Ces caractéristiques peuvent être connues précisément (des constantes), inconnues ou bornées (entre deux valeurs extrêmes) ; elles peuvent de plus être indépendantes ou liées entre elles. Pour les tâches, on considèrera par exemple les cas suivants :

- les intensités de ressource requises et la durée de la tâche sont connues ;

- l'énergie de la tâche est connue et sa durée, bornée par une valeur minimum et une valeur maximum, dépend de l'intensité de ressource utilisée.

Le genre de données et le niveau d'information sur ces données sont essentiellement tributaires du type de l'intervalle étudié. La nature des ressources utilisées est une donnée parfaitement connue (cf. I.3). Leur intensité est parfaitement connue pour des intervalles fournisseurs, connue ou bornée dans le cas des intervalles consommateurs. Comme on l'a déjà vu, les inconnues de notre problème d'ordonnancement sont les dates de début (et de fin), c'est-à-dire les caractéristiques permettant de localiser dans le temps les tâches. Toutefois pour les intervalles consommateurs, ces données sont encadrées (cf. paragraphe II.2.5) par une valeur minimale (date au plus tôt) et une valeur maximale (date au plus tard) qui sont, elles, parfaitement connues à un stade donné de l'analyse. Pour les intervalles fournisseurs, les dates de début et de fin peuvent être instanciées par des valeurs associées aux dates de début au plus tôt et de fin au plus tard d'intervalles consommateurs. Cependant, on verra (cf. paragraphes IV.2 et IV.3) qu'il peut être préférable de ne pas se limiter à ces valeurs. Pour les intervalles consommateurs, la durée est connue ou bornée ; dans ce dernier cas, on raisonne à énergie (durée $\times$ intensité) constante. Lorsque la durée de la tâche dépend de la quantité de ressource utilisée, elle est donc liée à l'intensité que requiert la tâche, par une fonction de la forme : $D = \frac{cte}{Q}$. La borne maximale de la durée est obtenue pour une utilisation de ressources correspondant ainsi à la borne minimale de l'intensité de ressource et inversement.

Les définitions seront toujours données dans le cas le plus général. Néanmoins, pour simplifier calculs et notations, et toutefois sans perte de généralité, on considèrera souvent le cas particulier où les intensités de ressources sont constantes dans le temps. Les intervalles sont alors dits *uniformes*. Dans le cas d'un intervalle temps-ressource I ($Q_I^k(t) = Q_I^k$ = constante $\forall k$), l'énergie de ressource k s'écrit alors : $W_I^k = (F_I - C_I).Q_I^k = D_I.Q_I^k$.

II.1.4 Utilisation des intervalles temps-ressource

Les problèmes classiques d'ordonnancement consistent à positionner dans le temps des intervalles consommateurs définis par leur durée et leurs intensités de ressources compte tenu des relations de ces intervalles avec d'autres intervalles consommateurs et fournisseurs. Ces relations sont l'expression de contraintes de deux types :

- des *contraintes de ressources* liées à l'utilisation de ressources limitées à l'intérieur de certains intervalles : la somme des quantités de ressource k utilisées à l'instant t sur un intervalle I ne peut excéder la quantité $Q_I^k(t)$.

On peut de plus imposer une limitation minimale sur l'utilisation simultanée des ressources (cf. II.4) ;

- des *contraintes temporelles* sur des positions relatives entre intervalles de temps.

Notons que les notions d'intervalles consommateur et fournisseur sont relatives. En effet, si l'on considère que le raisonnement s'effectue par affinements successifs au sein d'une structure hiérarchisée à plusieurs niveaux, un intervalle consommateur à un niveau donné peut être considéré comme fournisseur à un niveau plus détaillé. La caractérisation des ordonnancements admissibles à un niveau donné va s'appuyer sur l'interaction entre intervalles consommateurs et intervalles fournisseurs. Si les intervalles consommateurs sont généralement bien définis (tâches), le choix des intervalles fournisseurs considérés est plus arbitraire et constitue un facteur déterminant pour l'efficacité du processus d'analyse sous contraintes (A.S.C.).

Remarquons par ailleurs que le concept d'intervalle temps-ressource permet de modéliser des problèmes à ressources , soit renouvelables (contraintes sur l'intensité), soit consommables (contraintes sur l'énergie), soit doublement contraintes [Słowiński 81][Węglarz 81].

II.2 Modélisation des contraintes temporelles

Les contraintes temporelles entre intervalles peuvent s'exprimer simplement par des *inégalités de potentiels* portant sur les bornes de ces intervalles.

Considérons un ensemble de n tâches ($i = 1, ..., n$) auquel on peut ajouter des tâches particulières appelées tâches fictives ; il peut être utile d'introduire des tâches qui traduisent le début (tâche *déb* ou 0) et la fin (tâche *fin*) d'un travail et qui permettent de modéliser les contraintes de début au plus tôt et de fin au plus tard de ce travail.

II.2.1 Inégalité de potentiels [Roy 70]

Une inégalité de potentiels est une inégalité de la forme :

$$C_j - C_i \geq a_{ij} \qquad \text{où } a_{ij} \text{ est une constante réelle.}$$

En prenant des valeurs positives, nulles et négatives pour les a_{ij}, il est possible de représenter différentes relations entre intervalles.

L'inégalité de potentiels $C_j - C_i \geq D_i$ exprime le fait que la tâche j ne peut débuter avant l'achèvement de la tâche i. Il s'agit d'une contrainte de *précédence simple*. $C_i - C_0 \geq a_{0i}$ représente une contrainte sur le début au plus tôt de la tâche i. De même, $C_0 - C_i \geq D_i - t_i$ (avec $t_i > D_i$) représente une contrainte sur la fin au plus tard de i.

II.2.2 Graphes conjonctifs

Dans le problème central de l'ordonnancement défini par B. Roy [Roy 70], toutes les contraintes peuvent être représentées par des inégalités de potentiels liées par la conjonction ET (contraintes conjonctives). Les ordonnancements admissibles apparaissent comme les solutions d'un système conjonctif d'inégalités de potentiels. Un tel système peut être modélisé par un *graphe conjonctif* dans lequel les inégalités de potentiels doivent être vérifiées conjointement pour tous les arcs du graphe. On trouve généralement deux types de graphe conjonctif :

- le graphe *potentiels-tâches* où chaque sommet représente une tâche et chaque arc une inégalité de potentiels, l'arc étant valué par le second membre de l'inégalité. La longueur d'un arc représente ainsi l'intervalle de temps minimum séparant le début des tâches associées aux extrémités de l'arc (voir par exemple figure II.2) ;

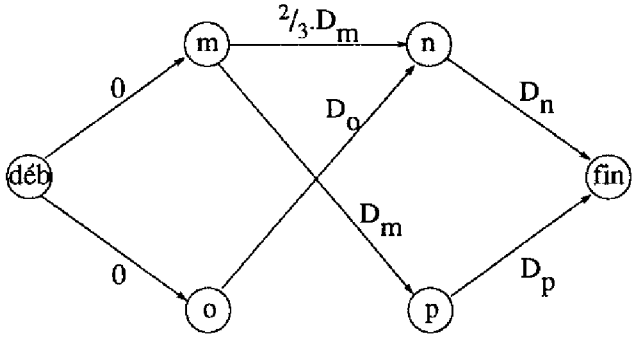


Figure II.2 : Graphe potentiels-tâches

- le *graphe potentiels-étapes* où chaque tâche donne naissance à un arc, éventuellement plusieurs, correspondant chacun à un fragment de la tâche ; de plus, certains arcs représentent des tâches fictives de durée nulle. Les sommets caractérisent des événements et servent à exprimer les contraintes (lorsque l'extrémité initiale d'un arc coïncide avec l'extrémité finale d'un autre, cela signifie que la tâche associée au premier de ces arcs ne peut pas commencer avant que celle associée au second soit terminée). Le graphe potentiels-étapes correspondant au graphe potentiels-tâches de la figure II.2 est représenté figure II.3.

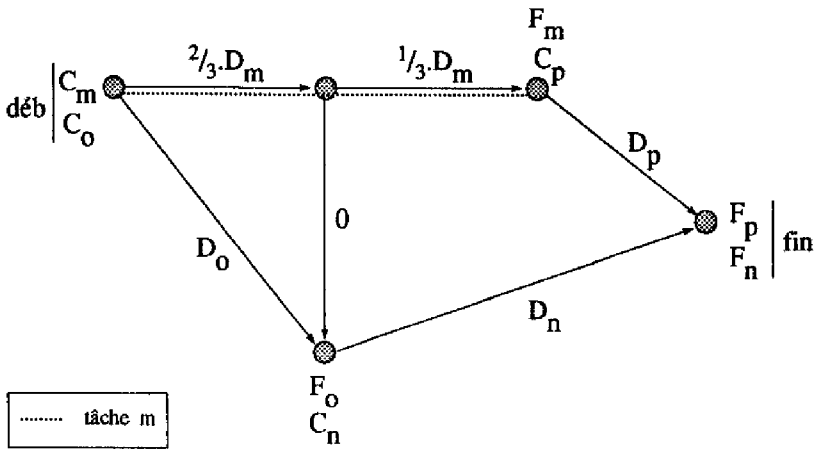


Figure II.3 : Graphe potentiels-étapes

II.2.3 Potentiels-tâches et potentiels-étapes

Lorsqu'on désire représenter uniquement des contraintes de précédence simples entre tâches, on peut préférer la modélisation par graphe potentiels-étapes. En revanche, lorsque le système de contraintes entraîne la prolifération de tâches fictives, cette formulation perd son intérêt et on préfèrera alors le modèle de graphe potentiels-tâches (où la définition de tâches fictives n'existe pas si ce n'est la création de tâches début et fin de durées nulles). En outre, la formulation potentiels-étapes est plus rigide : le graphe potentiels-tâches se prête mieux aux modifications de contraintes survenues par exemple à la suite d'aléas. Toutefois dans [Faure et al. 76] et [Roy 70], les auteurs assurent que selon les praticiens, le graphe potentiels-étapes est plus "lisible" par le personnel d'exécution que ne l'est le graphe potentiels-tâches, en raison de sa relation étroite avec le diagramme à barres (figure II.4).

II.2.4 Graphes non conjonctifs [Roy 70]

Les tâches peuvent être soumises, en plus des contraintes conjonctives, à des contraintes non conjonctives représentables par des inégalités de potentiels reliées par un OU. C'est le cas, par exemple, lorsque l'on veut représenter les conflits résultant des contraintes de limitation de ressources. Ce problème se modélise grâce à un *graphe non conjonctif* constitué d'une partie conjonctive et d'un ensemble de groupes non conjonctifs. Toutes les inégalités de potentiels associées à la partie conjonctive doivent être vérifiées et au moins une de celles relatives à chaque groupe non conjonctif. Un cas particulier de graphe non conjonctif est le *graphe disjonctif* (figure II.5) où tout groupe non conjonctif est constitué de deux arcs

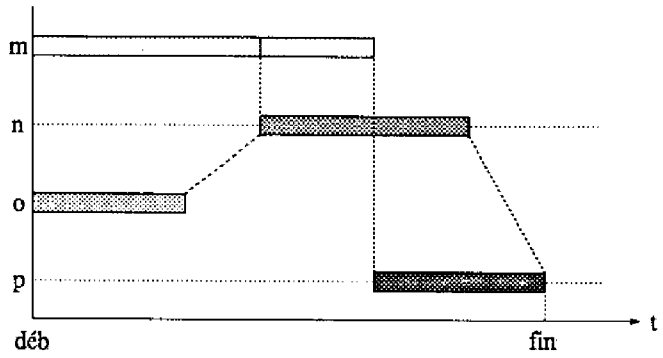
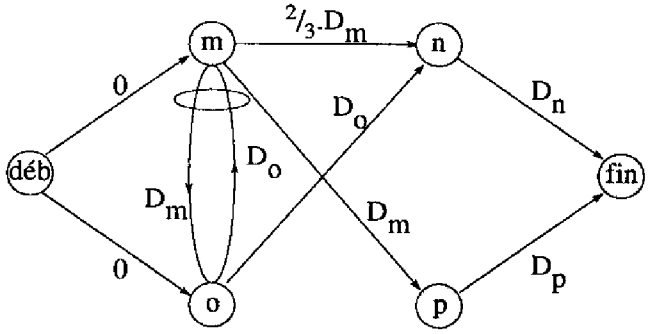


Figure II.4 : Diagramme à barres associé au graphe potentiels-étapes

appelés *paire de disjonction* et qui est de la forme :

$$C_j - C_i \geq a_{ij} \quad \text{OU} \quad C_i - C_j \geq a_{ji} \qquad \text{avec } a_{ij} + a_{ji} > 0.$$



Entre m et o, il existe une paire de disjonction qui exprime que les intervalles de réalisation de m et o doivent être disjoints. Elle peut se représenter par :

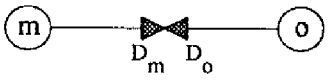


Figure II.5 : Graphe disjonctif

II.2.5 Graphe potentiels-bornes

Les graphes potentiels-tâches et potentiels-étapes possèdent l'inconvénient de ne pas pouvoir représenter les tâches à durées variables. Un autre type de graphe permettant de pallier ce problème est introduit ici.

La modélisation des contraintes temporelles retenue dans notre approche utilise la notion de *graphe potentiels-bornes.* Il s'agit d'un graphe dont les sommets représentent des bornes d'intervalles (début et fin) et qui est bien adapté au concept d'intervalle temps-ressource introduit en II.1. On crée donc deux sommets par intervalle. Lorsque la durée est connue, un seul sommet suffit pour représenter un intervalle, la durée intervenant dans la valuation des arcs, comme dans les graphes potentiels-étapes et potentiels-tâches présentés précédemment. Les arcs représentent des inégalités de potentiels portant sur les bornes associées aux extrémités des arcs. Par le biais de ce graphe, une relation entre deux intervalles peut s'écrire à l'aide de quatre relations entre les deux débuts et les deux fins de ces intervalles. On fait correspondre à chaque sommet B_I de ce graphe deux valeurs réelles $\underline{B_I}$ et $\overline{B_I}$ qui définissent les valeurs minimum et maximum prises par la borne d'intervalle considérée.[1] La création de deux sommets spécifiques pour chaque intervalle permet d'exprimer des contraintes numériques, en particulier sur les durées.

Le graphe potentiels-bornes défini ci-dessus est également mentionné dans [Rit 88] où il apparaît sous le nom de réseau d'arcs valués (figure II.6).

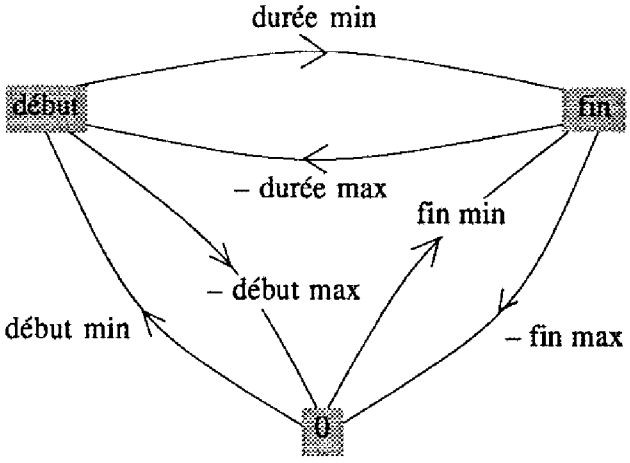


Figure II.6 : Réseau d'arcs valués

Ce graphe peut être vu comme une généralisation des graphes potentiels-tâches et potentiels-étapes utilisés couramment dans le domaine de l'ordonnancement. Au prix d'une augmentation limitée de la complexité, il offre un pouvoir de représentation bien supérieur, lié à l'association de sommets spécifiques à chaque début et à chaque fin de tâche ou d'intervalle. Ceci permet, par exemple, de représenter des tâches dont on ne connaît pas précisément la durée.

[1] Ce type de notation a également été utilisé dans [Leachman et al. 90].

II.2.6 Processus d'inférence par propagation sur un graphe potentiels-bornes

Le graphe potentiels-bornes ainsi défini constitue un réseau de contraintes particulier [Davis 87] sur lequel il est possible de propager les valeurs $\{\underline{B_I}, \overline{B_I}\}$ en activant une règle de déduction par affinement présentée ci-après.

Considérons les inégalités de potentiels :

$$\begin{cases} B_I - B_J \geq a_{JI} \\ B_K - B_I \geq a_{IK} \end{cases} \quad \text{avec} \quad \begin{cases} B_I \in [\underline{B_I}, \overline{B_I}] \\ B_J \in [\underline{B_J}, \overline{B_J}] \\ B_K \in [\underline{B_K}, \overline{B_K}] \end{cases}$$

B_I peut être affinée de la manière suivante :

$$B_I \in [max(\underline{B_I}, \underline{B_J} + a_{JI}), min(\overline{B_I}, \overline{B_K} - a_{IK})].$$

Cette propagation revient à calculer des chemins de longueurs maximales sur un graphe, ce qui peut être effectué en $O(n^3)$. Le processus d'inférence associé à cette propagation est sain et complet. Les valeurs obtenues représentent les valeurs "au plus tôt" et "au plus tard" des bornes d'intervalles. Lorsque les valeurs initiales des $\underline{B_I}$ et $\overline{B_I}$ ne sont pas connues, il suffit de prendre par défaut $\underline{B_I} = -\infty$ et $\overline{B_I} = +\infty$.

Pour illustrer, on présente un petit exemple de propagation de contraintes sur un graphe potentiels-bornes. On considère trois intervalles x, y, z dont on connaît certaines caractéristiques ($\underline{D_i}$ et $\overline{D_i}$ représentent les durées minimales et maximales d'un intervalle i) :

$$\underline{C_y} = 0,\ \overline{C_y} = 1,\ \underline{D_y} = 3,\ \overline{D_y} = 4\ ;\quad \underline{C_z} = 2,\ \overline{F_z} = 11,\ \underline{D_z} = 4,\ \overline{D_z} = 5.$$

Par ailleurs, les intervalles sont tels que y précède ou chevauche (au sens large) x et que z se situe à l'intérieur (au sens large) de x. De plus, y ne peut pas précéder (au sens strict) z. Ceci peut s'écrire :

$$\begin{cases} C_x - C_y \geq 0 \\ F_x - F_y \geq 0 \end{cases} \text{ et } \begin{cases} C_z - C_x \geq 0 \\ F_x - F_z \geq 0 \end{cases} \text{ et } F_y - C_z \geq 0.$$

Le graphe potentiels-bornes associé est représenté figure II.7.a (les arcs non étiquetés sont valués par 0). La propagation des valeurs $\{\underline{B_i}, \overline{B_i}\}$, par recherche des plus longs chemins sur ce graphe, permet d'obtenir les valeurs indiquées sur la figure II.7.b.

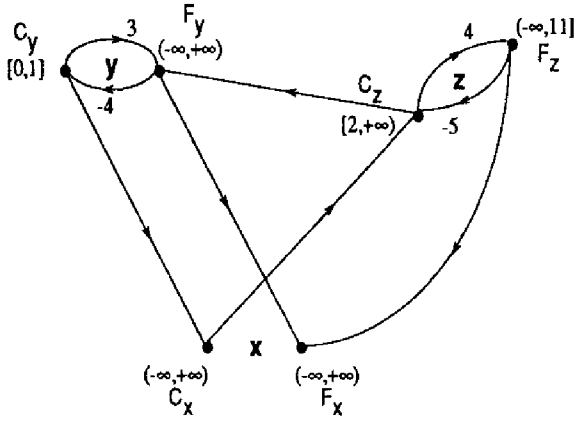


Figure II.7.a

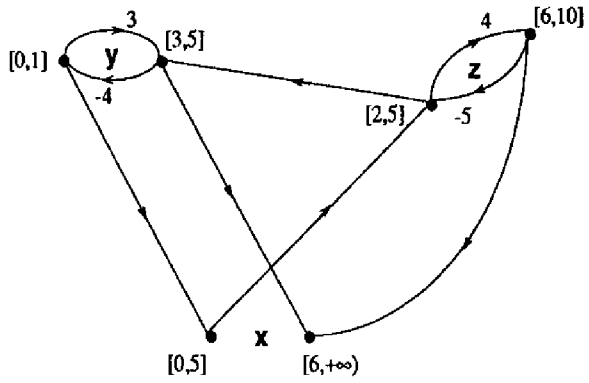


Figure II.7.b

II.2.7 Place du processus de propagation au sein du processus d'analyse sous contraintes

Le processus d'inférence par propagation sur les graphes potentiels-bornes peut être utilisé pour caractériser les ordonnancements admissibles en tenant compte uniquement de contraintes temporelles exprimables par une conjonction d'inégalités de potentiels ; cela ne présente pas de difficulté particulière. Les ordonnancements admissibles sont alors caractérisés par les dates de début au plus tôt et au plus tard des différentes tâches (caractéristiques temporelles). Par contre, dès qu'interviennent des contraintes nécessitant l'utilisation d'ensembles non conjonctifs d'inégalités de potentiels, la complexité du problème est bien plus grande, en raison du caractère combinatoire de ces contraintes. Or, ces contraintes résultent généralement de la prise en compte de contraintes de limitation de ressource dont le partage nécessite la *résolution de conflits* entre tâches.

L'approche retenue pour l'A.S.C. consiste à appliquer le processus d'inférence par propagation des contraintes temporelles en négligeant initialement les contraintes de ressources. A partir des valeurs obtenues, les contraintes de ressources sont prises en compte de façon à déduire de nouveaux faits (ajout d'une nouvelle contrainte temporelle, modification d'une valeur $\underline{B_I}$ ou $\overline{B_I}$) susceptibles de relancer le processus de propagation.

Dans les développements ci-après, on associe généralement des minuscules aux tâches et des majuscules aux intervalles fournisseurs. Ainsi pour une tâche i, C_i représente son début, F_i sa fin. De même, $\underline{C_i}, \overline{C_i}, \underline{F_i}, \overline{F_i}$ représentent respectivement les dates de début au plus tôt et au plus tard et les dates de fin au plus tôt et au plus tard.

II.3 Consommation obligatoire d'une tâche sur un intervalle fournisseur

II.3.1 La consommation obligatoire

II.3.1.1 Définition

Considérons un intervalle fournisseur $\Delta = (C_\Delta, F_\Delta, \{Q_\Delta^k(t)\})$ et une tâche $i = (C_i, F_i, \{q_i^k(t)\})$ de durée $D_i = F_i - C_i$. La *consommation obligatoire* de i sur Δ correspond à l'énergie obtenue lorsque l'on cherche à repousser au maximum i à l'extérieur de Δ. Le calcul de consommation obligatoire consiste donc à rechercher des positions de i telles que l'intersection de i avec Δ soit minimale voire nulle ; il revient ainsi à étudier la consommation d'énergie de i dans ses positions extrêmes, calée au plus tôt et au plus tard, ce qui amène à considérer quatre termes relatifs à la position et aux caractéristiques de la tâche.

La consommation obligatoire de la tâche i, associée à une ressource k, sur l'intervalle Δ est notée $w_i^{\Delta,k}$ et s'écrit :

$$w_i^{\Delta,k} = max\{0, min[\int_{C_\Delta}^{\underline{C_i}+D_i} q_i^k(t)dt, \int_{\overline{F_i}-D_i}^{F_\Delta} q_i^k(t)dt, \int_{C_\Delta}^{F_\Delta} q_i^k(t)dt, \int_{C_i}^{C_i+D_i} q_i^k(t)dt]\}$$

Il apparaît ainsi que la consommation obligatoire est une grandeur non-négative. Chacun des termes apparaissant sous l'opérateur *min* est représentatif d'une position de la tâche à l'intérieur de son intervalle alloué. Ainsi, on considère le premier terme lorsque la tâche est calée à gauche de son intervalle $[\underline{C_i}, \overline{F_i}]$, le second terme lorsqu'elle est calée à droite, le troisième lorsqu'elle recouvre complètement l'intervalle fournisseur et le quatrième lorsqu'elle est complètement à l'intérieur de l'intervalle fournisseur. Une illustration pour les deux premiers termes est donnée dans le paragraphe suivant.

Ce résultat, qui exprime la consommation obligatoire d'énergie d'une tâche non précisément localisée sur un intervalle fournisseur, est à la base des déductions présentées par la suite qui font intervenir la consommation d'énergie.

II.3.1.2 Consommation d'intervalles uniformes à durée constante

Dans cette hypothèse, les intensités de ressource restent égales à des constantes sur chaque intervalle d'étude et on suppose que la durée de la tâche est connue et

égale à D_i. La formule précédente s'écrit alors [2] :

$$\begin{aligned} w_i^{\Delta,k} &= max\{0, min[(\underline{C}_i + D_i - C_\Delta).q_i^k, \langle F_\Delta - (\overline{F}_i - D_i)\rangle.q_i^k, (F_\Delta - C_\Delta).q_i^k, D_i.q_i^k]\} \\ &\overset{q_i^k \geq 0}{=} max\{0, min[\underline{C}_i + D_i - C_\Delta, F_\Delta - \overline{F}_i + D_i, F_\Delta - C_\Delta, D_i]\}.q_i^k \\ &= \mathcal{T}_i^\Delta.q_i^k \quad \geq 0. \end{aligned}$$

Cette formule est illustrée sur la figure II.8 où, dans ce cas précis, la tâche i consomme le minimum d'énergie sur Δ lorsqu'elle est calée à droite (i.e. au plus tard). De cette façon, c'est le terme $[F_\Delta - (\overline{F}_i - D_i)].q_i^k$ qui fixe la consommation obligatoire.

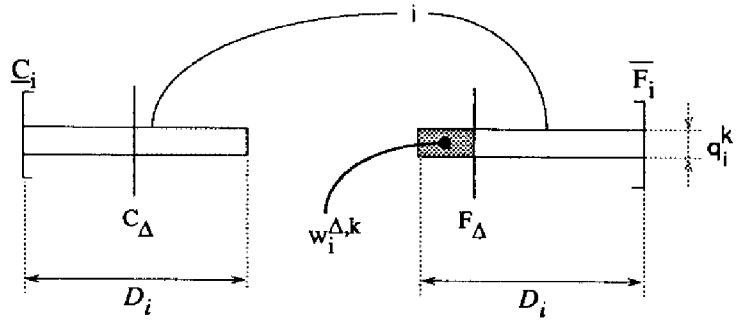


Figure II.8 : Consommation obligatoire

Sachant que $D_i.q_i^k = w_i^k$ (énergie de ressource k requise par i), on peut encore écrire :

$$w_i^{\Delta,k} = max\{0, min[\frac{\underline{C}_i - C_\Delta}{D_i} + 1, \frac{F_\Delta - \overline{F}_i}{D_i} + 1, \frac{F_\Delta - C_\Delta}{D_i}, 1]\}.w_i^k$$

Mise sous cette forme, il apparaît clairement que la consommation obligatoire d'une tâche sur un intervalle fournisseur est une fraction de l'énergie requise par la tâche pour son exécution.

Remarque : partie obligatoire et consommation obligatoire.
La notion de partie obligatoire est introduite dans [Lahrichi 82] comme suit.

Soit une tâche $i = (C_i, F_i, \{q_i^k(t)\})$ de durée $D_i = F_i - C_i$; si $\overline{F}_i - D_i < \underline{C}_i + D_i$, on dit que la tâche i a une *partie obligatoire* d'intervalle obligatoire $IO = [\overline{F}_i - D_i, \underline{C}_i + D_i]$ (cf. figure II.9). Dans cet intervalle IO, q_i^k unités de ressource sont utilisées par la tâche i.

[2] Ce résultat est voisin de la définition de la *charge obligatoire* énoncée dans [Lahrichi 82].

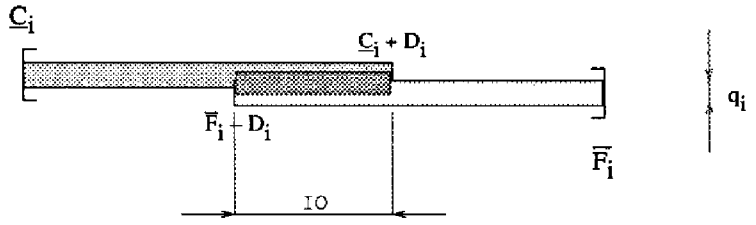


Figure II.9 : Partie obligatoire

La propriété suivante, qui établit un lien entre partie obligatoire et consommation obligatoire, peut être facilement démontrée :

$$\begin{cases} w_i^{\Delta,k} = (F_\Delta - C_\Delta).q_i^k \neq 0 \\ \text{ssi } i \text{ a une partie obligatoire et } \Delta \subseteq IO. \end{cases}$$

II.3.1.3 Consommation d'intervalles uniformes à énergie constante

On considère une tâche i, caractérisée par une énergie constante w_i^k, de durée D_i comprise entre $\underline{D}_i$ et $\overline{D}_i$.

$$\begin{cases} w_i^k = D_i.q_i^k = cte \\ \underline{D}_i \leq D_i \leq \overline{D}_i \end{cases} \Longrightarrow \underline{q}_i^k \leq q_i^k \leq \overline{q}_i^k$$

Pour une durée variable de la tâche, la consommation obligatoire est la borne inférieure de toutes les consommations minimales de i sur Δ. La détermination de cette borne passe par l'étude du signe des termes de $w_i^{\Delta,k}$.

$F_\Delta - C_\Delta > 0$
: C_Δ et F_Δ étant respectivement liés à des dates de début au plus tôt et de fin au plus tard, la condition $F_\Delta - C_\Delta \leq 0$ signifie la non-existence d'une tâche. Un minorant de $\frac{F_\Delta - C_\Delta}{D_i}$ est donc $\frac{F_\Delta - C_\Delta}{\overline{D}_i}$.

$\underline{C}_i - C_\Delta < 0$
: En effet, lorsque $\underline{C}_i - C_\Delta \geq 0$, le terme $\frac{\underline{C}_i - C_\Delta}{D_i} + 1$ n'est pas pris en compte dans la formule de la consommation obligatoire car il existe un autre terme ("1" par exemple) donnant un résultat inférieur ou égal. Un minorant de $\frac{\underline{C}_i - C_\Delta}{D_i}$ est $\frac{\underline{C}_i - C_\Delta}{\underline{D}_i}$.

$F_\Delta - \overline{F}_i < 0$

Le raisonnement est le même que pour $\underline{C}_i - C_\Delta$. Le terme minorant de $\frac{F_\Delta - \overline{F}_i}{D_i}$ est $\frac{F_\Delta - \overline{F}_i}{\underline{D}_i}$.

L'expression de la consommation obligatoire de i sur Δ pour une durée de i variable et une énergie constante peut alors s'écrire :

$$w_i^{\Delta,k} = max\{0, min[\frac{\underline{C}_i - C_\Delta}{\underline{D}_i} + 1, \frac{F_\Delta - \overline{F}_i}{\underline{D}_i} + 1, \frac{F_\Delta - C_\Delta}{\overline{D}_i}, 1]\}.w_i^k$$

II.3.1.4 Propriétés fondamentales de la consommation obligatoire

Nous allons présenter trois propriétés de la consommation obligatoire servant de support au raisonnement ou aidant à le clarifier.

Propriété 1 : Une tâche i consomme sur un intervalle fournisseur Δ ssi $w_i^{\Delta,k} \neq 0$ c'est-à-dire ssi $\begin{cases} \underline{C}_i + D_i > C_\Delta \\ \text{et} \\ \overline{F}_i - D_i < F_\Delta \end{cases}$

En effet, $F_\Delta - C_\Delta \neq 0$ et $D_i \neq 0 \Longrightarrow \begin{cases} \underline{C}_i + D_i - C_\Delta > 0 \\ \text{et} \\ F_\Delta - \overline{F}_i + D_i > 0 \end{cases}$

Propriété 2 : soit x et $y \in [\underline{C}_i, \overline{F}_i]$. Si $x + y = \underline{C}_i + \overline{F}_i$ alors $w_i^{[\underline{C}_i, x], k} = w_i^{[y, \overline{F}_i], k}$.

Démonstration : $\triangleright\triangleright$ on cherche $y \in [\underline{C}_i, \overline{F}_i]$ tel que la consommation de i sur $[\underline{C}_i, x]$ soit la même que celle de i sur $[y, \overline{F}_i]$:

$$w_i^{[\underline{C}_i, x], k} = w_i^{[y, \overline{F}_i], k}$$

$\Longrightarrow x - (\overline{F}_i - D_i) = \underline{C}_i + D_i - y$
$\Longleftrightarrow x + y = \underline{C}_i + \overline{F}_i$ $\triangleleft\triangleleft$

Cas particulier : l'instant z pour lequel la consommation obligatoire de i sur $[\underline{C}_i, z]$ est la même que celle de i sur $[z, \overline{F}_i]$ est le point médian du segment $[\underline{C}_i, \overline{F}_i]$:

$$w_i^{[\underline{C}_i, z], k} = w_i^{[z, \overline{F}_i], k} \text{ si } z = \frac{\underline{C}_i + \overline{F}_i}{2}$$

Propriété 3 : soient les intervalles fournisseurs $\Delta = (C, F, Q^k)$, $\Delta' = (F, F', Q'^k)$ [3] et $\Delta'' = (C, F', Q''^k)$ et la tâche $i = (C_i, F_i, q_i^k)$. On a la relation suivante :

$$w_i^{\Delta,k} + w_i^{\Delta',k} \leq w_i^{\Delta'',k}.$$

[3] $F' > F$

Autrement dit, la somme des consommations obligatoires d'une tâche sur deux intervalles adjacents est inférieure ou égale à la consommation obligatoire de la tâche sur l'intervalle obtenu par union des deux intervalles précédents.

Ce résultat exprime une propriété fondamentale de la consommation obligatoire que l'on exploitera dans les raisonnements ultérieurs. Il implique, par exemple, que la consommation obligatoire étant non-négative :

$$\text{si } w_i^{\Delta,k} = w_i^{\Delta'',k} \text{ alors } w_i^{\Delta',k} = 0.$$

Démonstration : ▷▷ sur chacun des trois intervalles fournisseurs , on écrit l'expression de la consommation obligatoire de i :

$$w_i^{\Delta,k} = max\{0, min[\frac{\underline{C}_i - C}{D_i} + 1, \frac{F - \overline{F}_i}{D_i} + 1, \frac{F - C}{D_i}, 1]\}.w_i^k$$

$$w_i^{\Delta',k} = max\{0, min[\frac{\underline{C}_i - F}{D_i} + 1, \frac{F' - \overline{F}_i}{D_i} + 1, \frac{F' - F}{D_i}, 1]\}.w_i^k$$

$$w_i^{\Delta'',k} = max\{0, min[\frac{\underline{C}_i - C}{D_i} + 1, \frac{F' - \overline{F}_i}{D_i} + 1, \frac{F' - C}{D_i}, 1]\}.w_i^k$$

Il s'agit à présent de comparer $w_i^{\Delta'',k}$ à $w_i^{\Delta,k} + w_i^{\Delta',k}$.

$$\begin{aligned} w_i^{\Delta,k} + w_i^{\Delta',k} = max\{0,\ & min[\tfrac{\underline{C}_i - C}{D_i} + 1, \tfrac{F - \overline{F}_i}{D_i} + 1, \tfrac{F - C}{D_i}, 1], \\ & min[\tfrac{\underline{C}_i - F}{D_i} + 1, \tfrac{F' - \overline{F}_i}{D_i} + 1, \tfrac{F' - F}{D_i}, 1], \\ & min[\ \tfrac{2\underline{C}_i - C - F}{D_i} + 2, \tfrac{\underline{C}_i - \overline{F}_i + F' - C}{D_i} + 2, \tfrac{\underline{C}_i + F' - F - C}{D_i} + 1, \tfrac{\underline{C}_i - C}{D_i} + 2, \\ & \quad \tfrac{\underline{C}_i - \overline{F}_i}{D_i} + 2, \tfrac{F + F' - 2\overline{F}_i}{D_i} + 2, \tfrac{F' - \overline{F}_i}{D_i} + 1, \tfrac{F - \overline{F}_i}{D_i} + 2, \\ & \quad \tfrac{\underline{C}_i - C}{D_i} + 1, \tfrac{F' + F - C - \overline{F}_i}{D_i} + 1, \tfrac{F' - C}{D_i}, \tfrac{F - C}{D_i} + 1, \\ & \quad \tfrac{\underline{C}_i - F}{D_i} + 2, \tfrac{F' - \overline{F}_i}{D_i} + 2, \tfrac{F' - F}{D_i} + 1, 2]\}.w_i^k \end{aligned}$$

Sachant que

$$\underline{C}_i - \overline{F}_i + F' - C > \underline{C}_i - \overline{F}_i$$

$$\underline{C}_i - \overline{F}_i < 0 \Rightarrow \tfrac{\underline{C}_i - \overline{F}_i}{D_i} + 2 < 2$$

$$\underline{C}_i - C < \underline{C}_i - C + F' - F$$

$$F' - \overline{F}_i + F - C > F' - \overline{F}_i$$

$$\begin{cases} \underline{C}_i - \overline{F}_i \leq -D_i \Rightarrow \tfrac{\underline{C}_i - \overline{F}_i}{D_i} + 2 \leq 1 \\ \tfrac{F - C}{D_i} + 1 > 1 \text{ et } \tfrac{F' - F}{D_i} + 1 > 1 \end{cases}$$

on obtient :

$$w_i^{\Delta,k} + w_i^{\Delta',k} = max\{0,\ min[\tfrac{\underline{C_i}-C}{D_i}+1, \tfrac{F-\overline{F}_i}{D_i}+1, \tfrac{F-C}{D_i}, 1],\ min[\tfrac{\underline{C_i}-F}{D_i}+1, \tfrac{F'-\overline{F}_i}{D_i}+1, \tfrac{F'-F}{D_i}, 1],\ min[\tfrac{2\underline{C_i}-C-F}{D_i}+2,\ \tfrac{\underline{C_i}-\overline{F}_i}{D_i}+2, \tfrac{F+F'-2\overline{F}_i}{D_i}+2, \tfrac{F'-\overline{F}_i}{D_i}+1, \tfrac{F-\overline{F}_i}{D_i}+2,\ \tfrac{\underline{C_i}-C}{D_i}+1, \tfrac{F'-C}{D_i},\ \tfrac{\underline{C_i}-F}{D_i}+2]\}.w_i^k$$

On compare $w_i^{\Delta'',k}$ à $w_i^{\Delta,k} + w_i^{\Delta',k}$. On a :

$$min(\frac{\underline{C_i}-C}{D_i}+1, \frac{F'-\overline{F}_i}{D_i}+1, \frac{F'-C}{D_i}, 1) \geq min(\frac{\underline{C_i}-C}{D_i}+1, \frac{F-\overline{F}_i}{D_i}+1, \frac{F-C}{D_i}, 1)$$

$$min(\frac{\underline{C_i}-C}{D_i}+1, \frac{F'-\overline{F}_i}{D_i}+1, \frac{F'-C}{D_i}, 1) \geq min(\frac{\underline{C_i}-F}{D_i}+1, \frac{F'-\overline{F}_i}{D_i}+1, \frac{F'-F}{D_i}, 1)$$

Quant à l'expression, sous le troisième opérateur min, les termes $\frac{\underline{C_i}-C}{D_i}+1$, $\frac{F'-\overline{F}_i}{D_i}+1$, $\frac{F'-C}{D_i}$ se retrouvent dans $w_i^{\Delta'',k}$ et dans $w_i^{\Delta,k} + w_i^{\Delta',k}$. Cherchons dans $w_i^{\Delta,k} + w_i^{\Delta',k}$ un terme majoré par 1 :
on a vu, dans une remarque précédente, que $\underline{C_i} - \overline{F}_i \leq -D_i$. D'où : $\frac{\underline{C_i}-\overline{F}_i}{D_i} + 2 \leq 1$ et $min(\frac{2\underline{C_i}-C-F}{D_i} + 2, ..., \frac{\underline{C_i}-F}{D_i} + 2) \leq min(\frac{\underline{C_i}-C}{D_i} + 1, \frac{F'-\overline{F}_i}{D_i} + 1, \frac{F'-C}{D_i}, 1)$.
On a trouvé, pour tous les termes de $w_i^{\Delta,k} + w_i^{\Delta',k}$, des majorants dans $w_i^{\Delta'',k}$. ◁◁

II.3.2 Exemple de calcul de consommation obligatoire

L'exemple suivant permet de détailler un calcul de consommations obligatoires sur un intervalle fournisseur.

Soit $E = \{m, n, o, p, x, y, z\}$ un ensemble de sept tâches de durées connues et $E' = \{i \ / \ i \in \{h, j, l\}\}$ un ensemble de trois tâches de durée $D_i \in [\underline{D_i}, \overline{D_i}]$ et telles que $w_i^k = cte$ pour $i \in E'$. Les caractéristiques pour une ressource k sont données dans les tableaux II.1.

i	D_i	$\underline{C_i}$	$\overline{F_i}$	q_i^k
m	6	0	7	4
n	5	5	12	3
o	10	1	13	2
p	2	4	8	1
x	6	0	11	1
y	4	8	12	3
z	1	8	12	2

i	$\underline{D_i}$	D_i	$\underline{C_i}$	$\overline{F_i}$	w_i^k
h	14	21	0	30	42
j	2	4	2	8	12
l	4	8	2	12	8

Tableau II.1

Sur l'intervalle fournisseur $I = (3, 10, Q^k)$, la formule de la consommation obligatoire donne :

$$\sum_{i \in E} w_i^{I,k} = 3 \times 4 + 3 \times 3 + 7 \times 2 + 2 \times 1 + 3 \times 1 + 2 \times 3 + 0 \times 2 = 46$$

$$\sum_{i \in E'} w_i^{I,k} = 0 \times 42 + (\frac{2-3}{2} + 1) \times 12 + (\frac{10-12}{4} + 1) \times 8 = 10$$

$$\sum_{i \in (E \cup E')} w_i^{I,k} = \sum_{i \in E} w_i^{I,k} + \sum_{i \in E'} w_i^{I,k} = 46 + 10 = 56.$$

II.4 La consommation maximale [Thuriot et al. 90]

L'énergie obligatoire définie précédemment représente la quantité minimale d'énergie qui doit être consommée sur l'intervalle fournisseur considéré. On peut définir la grandeur duale, c'est-à-dire la quantité maximale d'énergie pouvant être consommée sur l'intervalle fournisseur ; il s'agit de la *consommation maximale* d'énergie d'une tâche sur un intervalle fournisseur . Elle est notée $v_i^{\Delta,k}$ et s'écrit dans le cas général :

$$v_i^{\Delta,k} = max\{0, min[\int_{C_\Delta}^{\overline{F}_i} q_i^k(t)dt, \int_{\underline{C_i}}^{F_\Delta} q_i^k(t)dt, \int_{C_\Delta}^{F_\Delta} q_i^k(t)dt, \int_{C_i}^{C_i+D_i} q_i^k(t)dt]\}.$$

Soit, en considérant des intervalles temps-ressource uniformes :

$$\begin{aligned} v_i^{\Delta,k} &= max\{0, min[(\overline{F}_i - C_\Delta).q_i^k, (F_\Delta - \underline{C_i}).q_i^k, (F_\Delta - C_\Delta).q_i^k, D_i.q_i^k]\} \\ &= max\{0, min[\tfrac{\overline{F}_i - C_\Delta}{D_i}, \tfrac{F_\Delta - \underline{C_i}}{D_i}, \tfrac{F_\Delta - C_\Delta}{D_i}, 1]\}.w_i^k. \end{aligned}$$

Cette notion de consommation maximale est intéressante lorsqu'il existe une borne inférieure sur l'énergie minimale de ressource à consommer. En effet, de même

que l'on définit $Q_\Delta^k(t)$ comme la quantité de ressource k disponible, c'est-à-dire la quantité maximale pouvant être utilisée à l'instant t, on peut introduire la grandeur $R_\Delta^k(t)$ qui représente la quantité minimale de ressource k devant être utilisée à t. L'énergie minimale de ressource k à consommer s'écrit alors :

$$V_\Delta^k = \int_{C_\Delta}^{F_\Delta} R_\Delta^k(t)dt.$$

Ainsi, si pour un intervalle de référence Δ donné, la somme des consommations maximales $\sum_i v_i^{\Delta,k}$ est inférieure à l'énergie minimale de ressource V_Δ^k, alors, le problème n'est pas faisable ; si elle est supérieure à W_Δ^k (respectivement, lorsque $\sum_i w_i^{\Delta,k} < V_\Delta^k$), il faut modifier les fenêtres temporelles allouées aux tâches (plus exactement les comprimer), en actualisant certaines extrémités temporelles, de manière à diminuer (resp. augmenter) la consommation maximale (resp. obligatoire).

Ces concepts sont par exemple utiles lorsque l'on veut s'assurer de la rentabilité d'une ressource en garantissant une utilisation minimale. Les investigations futures offrent donc des possibilités de comparaisons entre W_Δ^k et $w_i^{\Delta,k}$ (cf. chapitres III et IV) d'une part et entre V_i^k, $v_i^{\Delta,k}$, W_Δ^k et $w_i^{\Delta,k}$ d'autre part [Thuriot et al. 90].

II.5 Hypothèses simplificatrices

Pour des intervalles non uniformes, un intervalle consommateur i possède une intensité de ressource $q_i^k(t)$ comprise entre $\underline{q_i^k}$ et $\overline{q_i^k}$. De même, une intensité de ressource $Q_I^k(t)$ comprise entre $\underline{Q_I^k}$ et $\overline{Q_I^k}$ est associée à un intervalle fournisseur I.

- Les règles de déduction présentées par la suite considèrent uniquement des intervalles uniformes,

mais elles peuvent être étendues à des intervalles quelconques. En effet, si l'on prend $q_i^k(t) = \underline{q_i^k}$ et $Q_I^k(t) = \overline{Q_I^k}$, les résultats restent valides.

De même, dans un souci de simplication des notations :

- on ne considèrera que le cas d'utilisation d'une seule ressource ; en conséquence, q_i^k devient q_i, w_i^k devient w_i, $w_i^{\Delta,k}$ devient w_i^Δ, etc.

L'analyse pour le cas multi-ressources peut être effectuée ressource par ressource.

- De plus, la quantité disponible de la ressource considérée étant supposée constante sur tout l'horizon d'étude, la quantité de ressource associée à un intervalle fournisseur sera notée Q quel que soit l'intervalle considéré.

II.6 Conclusion

Ce chapitre est consacré à la présentation des éléments de base nécessaires à la suite de notre travail.

L'intervalle temps-ressource permet à la fois de représenter les tâches ou intervalles consommateurs et les intervalles de temps alloués sur lesquels des ressources sont disponibles, appelés intervalles fournisseurs. L'outil de modélisation employé est le graphe potentiels-bornes, adéquat pour l'utilisation d'intervalles temps-ressource, sur lequel est décrit un processus de propagation de contraintes. Le couplage du temps et des ressources, induit par la notion d'intervalle temps-ressource, amène à utiliser une nouvelle grandeur : l'énergie. A ce propos, la consommation obligatoire est introduite pour étudier les interactions entre intervalles consommateurs et intervalles fournisseurs.

Il reste à analyser l'impact de ces interactions sur le problème d'ordonnancement en explicitant des règles de déduction mettant à profit les considérations énergétiques. L'élaboration de ces règles fait l'objet des deux chapitres suivants.

* *

*

Chapitre III

Déduction de contraintes temporelles à partir des contraintes de ressources

La prise en compte simultanée du temps et des ressources est rendue possible par le concept d'intervalle temps-ressource qui permet d'introduire un raisonnement basé sur des bilans énergétiques.

Ce chapitre propose des règles d'inférence permettant de générer de nouvelles contraintes par prise en compte des limitations sur les quantités instantanées des ressources et sur les consommations de temps et d'énergie. Ces contraintes expriment des conditions séquentielles entre tâches en conflit et sont susceptibles de relancer le processus de propagation sur le graphe potentiels-bornes.

III.1 Déduction basée sur la limitation d'intensité de ressource

III.1.1 Ensembles critiques de tâches

III.1.1.1 Définition

On considère un intervalle fournisseur $\Delta = (C_\Delta, F_\Delta, \{Q^k_\Delta\})$ et un ensemble E de tâches du type :

$$E = \{i \ / \ i = (C_i, F_i, \{q^k_i\})\}$$

[1] et tels que :

$$\forall i \in E, \quad C_i \geq C_\Delta, \quad F_i \leq F_\Delta.$$

[1] On rappelle que Q^k_Δ et q^k_i sont des constantes (cf. II.5).

L'ensemble $E' \subset E$ est appelé *ensemble de tâches non réalisables simultanément* si l'on a la condition :

$$\sum_{i \in E'} q_i^k > Q_\Delta^k.$$

Nous sommes alors confrontés à des conflits d'utilisation de la ressource k. Pour les modéliser, nous allons utiliser la notion d'*ensemble critique* de tâches [Nabeshima 73] [Erschler et al. 79] noté E_c^α et défini par :

$$\langle \exists\, k \text{ tel que } \sum_{i \in E_c^\alpha} q_i^k > Q_\Delta^k \rangle \text{ et } \langle \forall\, k' \text{ et } E' \subset E_c^\alpha, \text{ on a } : \sum_{i \in E'} q_i^{k'} \leq Q_\Delta^{k'} \rangle.$$

Un *ensemble critique* est un ensemble minimal de tâches dont la réalisation simultanée nécessite une quantité de ressource supérieure à la quantité disponible. Tout sous-ensemble de E_c^α respecte les contraintes cumulatives de ressource. Ainsi, il faut et il suffit d'ordonner deux tâches au sein de chaque ensemble critique pour résoudre tous les conflits. Les contraintes cumulatives sont donc satisfaites ssi :

$$\forall\, \alpha,\ \exists\, (i,j) \in E_c^\alpha \times E_c^\alpha,\ i \neq j \text{ telles que } (C_j - C_i \geq D_i) \text{ OU } (C_i - C_j \geq D_j).$$

Cela permet de ramener la prise en compte des contraintes de ressources à des contraintes de précédence modélisées par des groupes d'arcs constituant la partie non conjonctive d'un graphe non conjonctif (cf. Exemple).

III.1.1.2 Exemple

Quatre tâches m, n, o, p utilisent la ressource k disponible en cinq exemplaires ($Q^k = 5$) conformément au tableau III.1.

i	m	n	o	p
q_i^k	3	2	1	4

Tableau III.1

Les ensembles critiques sont :

$$\{m,p\}\ ;\ \{n,p\}\ ;\ \{m,n,o\}$$

et se représentent ainsi (figure III.1) :

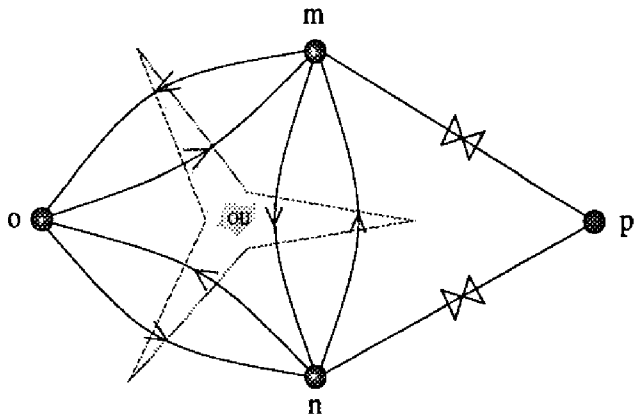


Figure III.1 : Modélisation des contraintes de ressource par graphe non conjonctif

Résoudre les conflits amène à respecter les conditions suivantes :

(m précède p) OU (p précède m)
ET
(n précède p) OU (p précède n)
ET
(m précède n) OU (n précède m) OU (m précède o) OU
(o précède m) OU (n précède o) OU (o précède n).

La résolution des conflits entraîne donc une caractérisation séquentielle des ordonnancements admissibles qui se superpose à la caractérisation temporelle vue en II.2.7.

III.1.1.3 Problème disjonctif et problème cumulatif

Le nombre $|E_c^\alpha| = card(E_c^\alpha)$ est appelé *ordre* de l'ensemble critique E_c^α. Un cas particulier important apparaît lorsque $|E_c^\alpha| = 2$. Ce cas est appelé "disjonctif" car les deux intervalles consommateurs doivent être disjoints. La modélisation des contraintes de ressource se fait alors grâce à un graphe disjonctif (paragraphe II.2.4). La résolution d'un conflit passe par le choix d'un arc dans une paire de disjonction. Si tel n'est pas le cas, c'est-à-dire $|E_c^\alpha| \geq 3$, on est en présence de problèmes "cumulatifs".

Dans le cas où l'on a à la fois des contraintes disjonctives et des contraintes cumulatives, il est intéressant d'arbitrer en priorité les paires de disjonction, plus structurantes pour l'ordonnancement. En effet, l'impossibilité de réaliser une opération avant une autre entraîne immédiatement la nécessité de les réaliser dans l'ordre inverse, ce qui n'est pas le cas en cumulatif.

III.1.2 Recherche des ensembles critiques

L'obtention des ensembles critiques passe par l'élaboration de deux algorithmes détaillés dans [Esquirol 87] :

Algorithme 1. Il est relatif à une seule ressource. On construit pour chaque ressource k, la liste L^k des tâches i utilisant k, ordonnées selon les valeurs décroissantes des q_i^k. L'algorithme élabore des sous-listes l en choisissant, ou en rejetant, un à un, les éléments x dans la liste L^k, dans l'ordre de classement de cette liste (méthode d'énumération arborescente) : si $q_x^k > q_y^k$, alors on ne peut choisir x si y a déjà été choisi. Deux paramètres guident la recherche : la somme s des intensités des éléments choisis parmi L^k et la somme r des intensités des éléments non rejetés ($\in \langle l \cup L^k \rangle$).

- Dès que $s > Q^k$, la liste l est close et un ensemble critique est mémorisé ;
- si x est choisi (x concaténé à l), s est actualisé à $s + q_x^k$;
- si x est rejeté, r est actualisé à $r - q_x^k$. Ce rejet n'est acceptable que si $r > Q^k$; on vérifie ainsi qu'il est encore possible de former un ensemble critique avec les éléments encore présents dans L^k.

Algorithme 2. Celui-ci assure la minimalité des ensembles critiques, en considérant l'ensemble des ressources. Il consiste en une suppression de certains ensembles qui en incluent d'autres. Pour chaque ensemble critique obtenu E_c^α :

- si $k \neq k'$ et $E_c^\alpha \subset E'$, alors supprimer E' mémorisé auparavant ;
- si $k \neq k'$ et si $\exists E_c^\alpha$ déjà mémorisé, tel que $E_c^\alpha \subset E'$, alors ne pas mémoriser E'.

Par ailleurs, seuls sont retenus les ensembles critiques d'intervalles effectivement en conflit, compte tenu des caractéristiques temporelles connues (dates limites) et de la partie conjonctive du graphe potentiels. En effet, les conflits possibles associés aux ensembles critiques ne considèrent que les contraintes de ressources. Certains de ces conflits peuvent être naturellement résolus par les contraintes temporelles [Erschler et al. 79].

Remarque : dans le cas général, l'algorithme 1 est exponentiel. Toutefois, dans certains cas particuliers (ressources à capacité unitaire, problème disjonctif général), la recherche des ensembles critiques est beaucoup plus directe. De plus, l'approche énergétique présentée au chapitre IV permet d'éluder ce problème de complexité dans le cas général.

III.1.3 Règle de déduction

On peut résumer ce qui a été dit précédemment ainsi :

Un ensemble critique E_c^α est un ensemble minimum de tâches qui ne peuvent se chevaucher compte tenu de la valeur Q_Δ^k de ressource k allouée : il en résulte qu'au moins deux des tâches de E_c^α doivent se succéder. Ceci fournit la règle d'inférence suivante :

$$(\Re 1) \quad \begin{cases} \text{si} & E_c^\alpha \text{ est un ensemble critique de tâches} \\ \text{alors} & \bigcup_{i \neq j \ ; \ (i,j) \in E_c^\alpha \times E_c^\alpha} (C_j - F_i \geq 0). \end{cases}$$

où le signe $\bigcup$ représente le OU logique. Dans l'exemple précédent (paragraphe III.1.1), les conditions sont :

$$(C_p - F_m \geq 0) \ \text{OU} \ (C_m - F_p \geq 0)$$

ET

$$(C_p - F_n \geq 0) \ \text{OU} \ (C_n - F_p \geq 0)$$

ET

$$(C_n - F_m \geq 0) \ \text{OU} \ (C_m - F_n \geq 0) \ \text{OU} \ (C_m - F_o \geq 0) \ \text{OU}$$
$$(C_o - F_m \geq 0) \ \text{OU} \ (C_o - F_n \geq 0) \ \text{OU} \ (C_n - F_o \geq 0).$$

Ainsi, la limitation de l'intensité de ressource fournie fait apparaître des conflits représentés sans redondance par les ensembles critiques. La résolution du conflit associé à un ensemble critique peut s'exprimer sous la forme d'un ensemble *non conjonctif* d'inégalités de potentiels. Cette première règle ne permet pas à elle seule de relancer le processus d'inférence par propagation.

III.2 Déduction basée sur la consommation temporelle

On se place désormais dans le cas d'utilisation d'une seule ressource (cf. II.5).

III.2.1 Introduction

On présente maintenant une règle de déduction très simple, voire triviale, qui prend en compte exclusivement la consommation temporelle, mais qui couplée à la règle ($\Re 1$), peut fournir des informations intéressantes. Dans les travaux antérieurs sur les problèmes disjonctifs [Erschler et al. 76][Erschler et al. 80], cette règle est d'ailleurs directement intégrée à la règle ($\Re 1$). Présentée sous cette forme, elle permet d'aborder tout type de problème.

III.2.2 Règle de déduction (précédence interdite)

Soient $i = (C_i, F_i, q_i)$ et $j = (C_j, F_j, q_j)$ deux tâches telles que :

$$F_i - C_i = D_i \ , \ F_j - C_j = D_j.$$

La règle de déduction suivante peut aisément être établie :

$$(\Re 2) \quad \begin{cases} \text{si} & \overline{F_j} \quad \underline{C_i} < D_i + D_j \\ \\ \text{alors} & C_j - F_i < 0. \end{cases}$$

Cette règle est illustrée par la figure III.2.

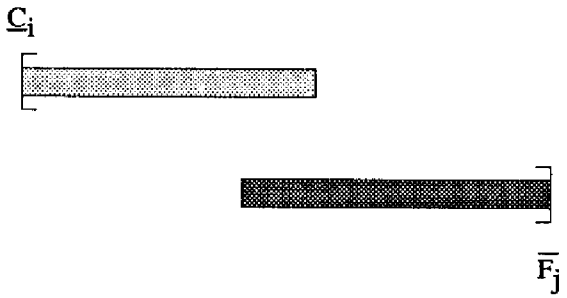


Figure III.2

Cette déduction interdit à l'intervalle i de précéder l'intervalle j compte tenu des valeurs $\underline{C_i}, \overline{F_i}$ et de la consommation temporelle D_i, D_j des valeurs i et j. Elle ne permet pas, à elle seule, de relancer le processus de propagation. Associée à une déduction issue de ($\Re 1$), elle peut cependant servir à réduire le nombre d'inégalités de potentiels intervenant dans l'ensemble non conjonctif associé à un ensemble critique de tâches. Si cette réduction permet de se ramener à une seule inégalité de potentiels, celle-ci peut être ajoutée au graphe des contraintes temporelles et permettre ainsi de relancer le processus de propagation. Si la réduction conduit à un ensemble non conjonctif d'inégalités de potentiels faisant toutes intervenir une des bornes d'un même intervalle, alors cet ensemble peut également être pris en compte dans le graphe des contraintes pour relancer le processus de propagation (cf. III.4 et III.5).

III.3 Déduction basée sur la consommation d'énergie

III.3.1 Introduction

Le raisonnement énergétique implique un traitement intégré du temps et des ressources. Dans les règles associées à ce raisonnement, le concept d'intervalle temps-ressource prend toute son importance. En effet, les tâches consomment

obligatoirement une certaine quantité d'énergie sur un intervalle fournisseur. L'ordonnancement des tâches à l'intérieur de cet intervalle fournisseur est ainsi conditionné par l'énergie qui reste disponible.

Dans ce contexte, on présente tout d'abord la règle ($\Re 3$) qui présente des analogies avec la règle ($\Re 2$) au niveau de la déduction. D'autres règles, basées elles aussi sur un raisonnement énergétique font l'objet d'un chapitre différent (chapitre IV) en raison de leur déduction plus avancée et des notions qu'elles engendrent.

III.3.2 Règle de déduction (précédence interdite)

Soient $i = (C_i, F_i, q_i)$ et $j = (C_j, F_j, q_j)$ deux tâches telles que :

$$F_i - C_i = D_i \ , \ F_j - C_j = D_j.$$

Sur un intervalle fournisseur $\Delta = (C_\Delta, F_\Delta, Q_\Delta)$, l'existence de solutions satisfaisant l'ensemble des contraintes dépend du résultat de la condition de faisabilité suivante :

$$\begin{cases} \text{si} & (F_\Delta - C_\Delta).Q_\Delta < \sum_{l \in E} w_l^\Delta \\ \\ \text{alors} & \text{le problème n'est pas faisable.} \end{cases}$$

En considérant l'intervalle fournisseur $IJ = (\underline{C_i}, \overline{F_j}, Q_{IJ})$, [2] la condition nécessaire d'admissibilité qui en découle s'énonce par la règle d'inférence suivante :

$$(\Re 3) \quad \begin{cases} \text{si} & (\overline{F_j} - \underline{C_i}).Q_{IJ} < \sum_{l \neq i,j} w_l^{IJ} + D_i.q_i + D_j.q_j \\ \\ \text{alors} & C_j - F_i < 0. \end{cases}$$

$(\overline{F_j} - \underline{C_i}).Q_{IJ}$ représente l'énergie de ressource pouvant être fournie par IJ,

$D_i.q_i + D_j.q_j$ représente l'énergie de ressource requise pour la réalisation de i et j,

$\sum_{l \neq i,j} w_l^{IJ}$ correspond à la consommation obligatoire (cf. II.3.1) sur IJ des tâches différentes de i et j.

Cette déduction analogue à celle obtenue en III.2.2, ne permet pas, à elle seule, de relancer le processus de propagation. Cependant, associée à la déduction issue des ensembles critiques de tâches, elle peut, par réduction, faire apparaître de nouvelles contraintes susceptibles de relancer le processus de propagation, comme cela a déjà été décrit en III.2.2.

[2] Pour les intervalles fournisseurs, on essaiera d'utiliser une notation mnémonique où les lettres majuscules correspondent aux lettres minuscules associées aux tâches pour lesquelles l'intervalle fournisseur est défini. Ainsi, pour l'intervalle IJ, I se rapporte à $\underline{C_i}$ et J se rapporte à $\overline{F_j}$.

Remarque : dans le cas très particulier où toutes les intensités de ressource sont identiques (*problème disjonctif pur*), (ℜ3) recouvre (ℜ2). Dans ce contexte précis, les résultats découlant de l'application de la règle temporelle (ℜ2) sont toujours moins forts ou, au mieux, équivalents à ceux déduits par la règle énergétique (ℜ3). Dans le cas général, les deux types de règles ne se recouvrent pas et présentent donc un intérêt propre comme le montrent les exemples présentés en III.5.

III.4 Actualisation de dates à partir de nouvelles contraintes temporelles

L'interaction entre les contraintes de temps et les conditions de succession issues de la résolution de conflits a donné lieu au développement d'un certain nombre de règles qui peuvent être consultées dans [Bel et al. 89]. On trouve des règles de séquencement et d'actualisation dans le cas disjonctif et des règles de séquencement dans le cas cumulatif à partir desquelles on peut appliquer les règles d'actualisation du cas précédent (présentées ci-dessous). Les règles de séquencement ont pour but de déduire des caractéristiques séquentielles d'admissibilité à partir des dates limites ; les règles d'actualisation sont destinées à rétrécir la fenêtre temporelle allouée à une tâche par augmentation de sa date de début au plus tôt et/ou diminution de sa date de fin au plus tard.

Lorsqu'on ne manipule que des contraintes conjonctives, une règle d'affinement du type de celle présentée en II.2.6 peut s'appliquer. Dès lors que des contraintes non conjonctives sont présentes, les règles d'actualisation ci-dessous peuvent être utilisées ($\underline{C}'_i$ et $\overline{F}'_i$ représentent les valeurs actualisées) :

$$\begin{cases} \text{si} & \bigcup_{j \in \Omega} j \text{ précède } i \\ \text{alors} & \underline{C}'_i = max[\underline{C}_i, \min_{j \in \Omega}(\underline{C}_j + D_j)] \end{cases}$$

et réciproquement :

$$\begin{cases} \text{si} & i \text{ précède } \bigcup_{j \in \Omega} j \\ \text{alors} & \overline{F}'_i = min[\overline{F}_i, \max_{j \in \Omega}(\overline{F}_j - D_j)]. \end{cases}$$

Par ailleurs, dans le cas particulier du problème disjonctif (où toute paire de tâches est un ensemble critique), la prise en compte simultanée de plusieurs faits séquentiels permet de déduire un résultat plus fort. Par exemple, pour l'actualisation de dates de début au plus tôt, on a la règle :

$$\begin{cases} \text{si} & (i,j),(i,l) \text{ et } (j,l) \text{ sont des ensembles critiques} \\ \text{et si} & j \text{ précède } i \text{ et } l \text{ précède } i \\ \text{alors} & \underline{C}'_i \geq max\{\underline{C}_i, min[max(\underline{C}_j + D_j, \underline{C}_l) + D_l, max(\underline{C}_l + D_l, \underline{C}_j) + D_j]\}. \end{cases}$$

Les figures III.3 et III.4 présentent des exemples d'actualisations illustrant les règles précédentes.

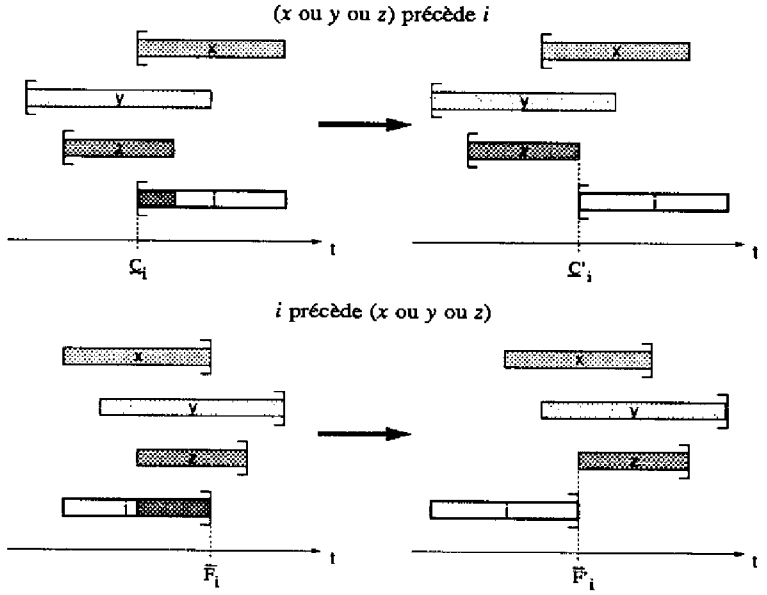


Figure III.3 : Actualisation à partir d'un fait séquentiel

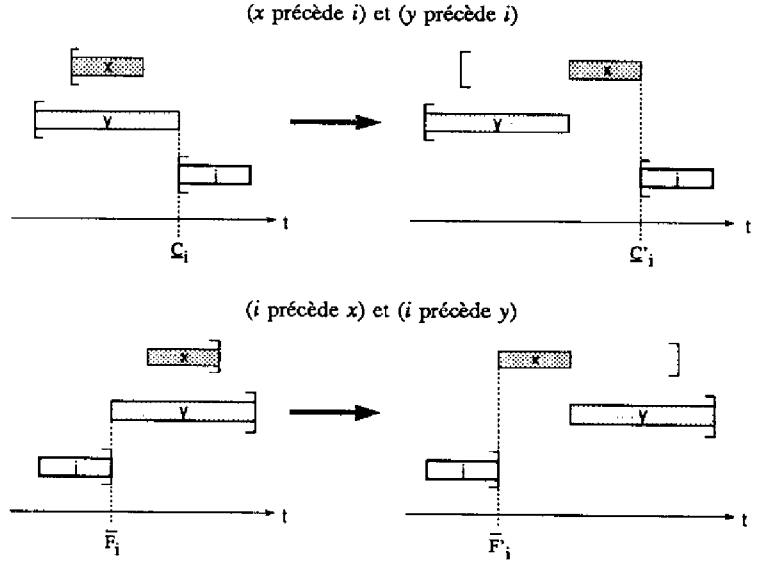


Figure III.4 : Actualisation à partir de plusieurs faits séquentiels (*cas disjonctif*)

III.5 Exemples

III.5.1 Exemple 1 (mise en évidence de l'intérêt de la règle ($\Re 2$) par rapport à la règle ($\Re 3$))

On considère trois tâches x, y, z constituant un ensemble critique (x, y, z utilisent, par exemple, la même machine disponible en deux exemplaires). Ces intervalles sont caractérisés par leur durée D_x, D_y, D_z. On connaît les valeurs $\underline{C}_x, \overline{F}_x, \underline{C}_y, \overline{F}_y$, $\underline{C}_z, \overline{F}_z$. Les différentes données sont indiquées dans le tableau III.2.

i	D_i	$\underline{C}_i$	$\overline{F}_i$
x	6	0	13
y	6	3	13
z	6	3	13

Tableau III.2

Si l'on cherche à appliquer la règle ($\Re 3$) à ce problème, aucune déduction n'est possible. En effet, les intervalles fournisseurs ont pour bornes temporelles $\{0, 13\}$ ou $\{3, 13\}$. L'intensité de ressource d'un intervalle fournisseur étant de 2, l'énergie pouvant être fournie par un tel intervalle admet un minimum à 20. D'autre part, la somme des énergies obligatoirement consommées par les tâches ou requises pour leur exécution, ne peut excéder 18 quel que soient la paire de tâches et l'intervalle fournisseur considérés. La règle ($\Re 3$) n'est ainsi jamais applicable.

Par contre, l'application de la règle ($\Re 2$) donne :

$$\overline{F}_x - \underline{C}_y < D_x + D_y \Longrightarrow C_x - F_y < 0$$

$$\overline{F}_x - \underline{C}_z < D_x + D_z \Longrightarrow C_x - F_z < 0$$

$$\overline{F}_y - \underline{C}_z < D_y + D_z \Longrightarrow C_y - F_z < 0$$

$$\overline{F}_z - \underline{C}_y < D_y + D_z \Longrightarrow C_z - F_y < 0.$$

Or, $\{x, y, z\}$ étant un ensemble critique, on a, d'après ($\Re 1$) :

$$(C_x - F_y \geq 0) \bigcup (C_y - F_z \geq 0) \bigcup (C_z - F_z \geq 0)$$

$$\bigcup (C_z - F_z \geq 0) \bigcup (C_y - F_z \geq 0) \bigcup (C_z - F_y \geq 0).$$

On peut donc en déduire après réduction :

$$(C_y - F_z \geq 0) \bigcup (C_z - F_z \geq 0).$$

Une telle contrainte peut être utilisée dans un processus de propagation, en appliquant une des règles d'affinement relatives aux contraintes non conjonctives (cf. III.4). La valeur de $\overline{F}_x$ est ainsi actualisée à 7.

III.5.2 Exemple 2 (mise en évidence de l'intérêt de la règle ($\Re 3$) par rapport à ($\Re 2$))

On a quatre tâches m, n, o, p dont chaque paire constitue un ensemble critique (m, n, o, p utilisent, par exemple, la même machine disponible en un seul exemplaire). Le tableau III.3 indique leurs caractéristiques.

i	D_i	$\underline{C}_i$	$\overline{F}_i$
m	3	3	10
n	3	0	10
o	4	3	14
p	1	9	14

Tableau III.3

Par application de ($\Re 2$), certaines déductions symboliques sont obtenues, mais aucune actualisation n'en découle. On trouve en effet les relations *n précède p* et *m précède p*, qui ne sont pas suffisantes pour déclencher le processus de propagation.

En revanche, l'application de ($\Re 3$) permet d'obtenir, en plus des déductions précédentes, la relation *n précède o*. Une actualisation est alors trouvée, comme le montre le raisonnement ci-dessous.

On calcule tout d'abord des valeurs de consommations obligatoires :

- intervalle fournisseur $PN = (\underline{C}_p, \overline{F}_n, 1) : w_m^{PN} + w_o^{PN} = 0 + 0$
- intervalle fournisseur $ON = (\underline{C}_o, \overline{F}_n, 1) : w_m^{ON} + w_p^{ON} = 3 + 0$
- intervalle fournisseur $PM = (\underline{C}_p, \overline{F}_m, 1) : w_n^{PM} + w_o^{PM} = 0 + 0.$

La règle ($\Re 3$) donne :

$$\overline{F}_n - \underline{C}_p < D_p + D_n + 0 \Longrightarrow C_n - F_p < 0$$

$$\overline{F}_n - \underline{C}_o < D_o + D_n + 3 \Longrightarrow C_n - F_o < 0$$

$$\overline{F}_m - \underline{C}_p < D_p + D_m + 0 \Longrightarrow C_m - F_p < 0.$$

$\{m, p\}, \{n, o\}, \{n, p\}$ étant des ensembles critiques, on a par ($\Re 1$) :

$$(C_m - F_p \geq 0) \bigcup (C_p - F_m \geq 0)$$

$$(C_n - F_o \geq 0) \bigcup (C_o - F_n \geq 0)$$

$$(C_n - F_p \geq 0) \bigcup (C_p - F_n \geq 0).$$

On en déduit ainsi :

$$(C_p - F_n \geq 0)$$

$$(C_o - F_n \geq 0)$$

$$(C_p - F_m \geq 0).$$

Les deux premières contraintes conjonctives permettent de relancer le processus de propagation en appliquant la règle d'actualisation valable dans le cas disjonctif :

$$\overline{F}_n \leq min\{\overline{F}_n, max[min(\overline{F}_o - D_o, \overline{F}_p) - D_p, min(\overline{F}_p - D_p, \overline{F}_o) - D_o]\}.$$

$\overline{F}_n$ est ainsi actualisée à 9.

III.6 Conclusion

En associant les règles basées sur la consommation temporelle ($\Re 2$) ou la consommation d'énergie ($\Re 3$) à la règle basée sur la limitation de l'intensité de ressource ($\Re 1$), il est donc possible de déduire de nouvelles contraintes temporelles susceptibles de relancer le processus de propagation décrit au paragraphe II.2.5.

Le schéma récapitulatif de la page suivante (figure III.5) montre la combinaison possible des différents types de déduction permettant de relancer le processus de propagation.

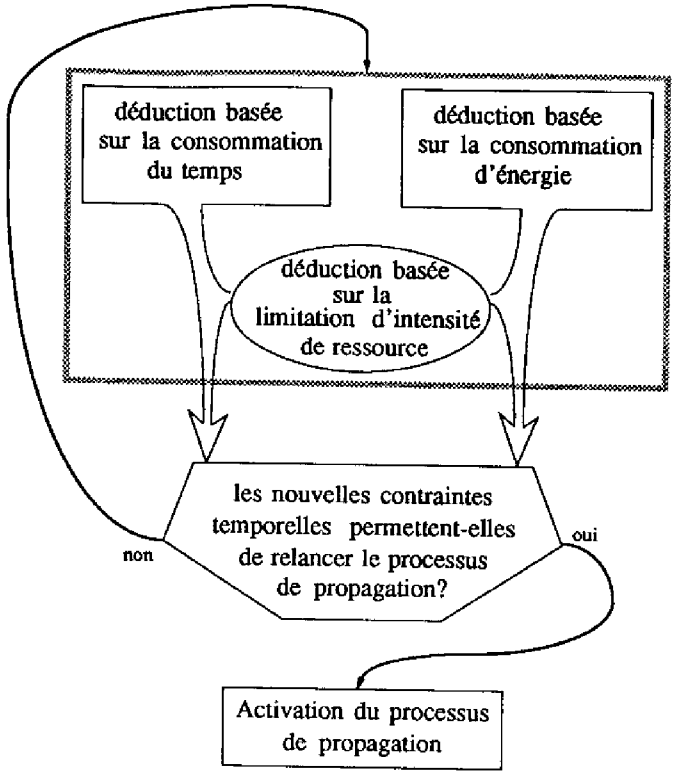


Figure III.5 : Combinaison de déductions

Dans le cas général, la règle ($\Re 2$) n'est pas redondante avec ($\Re 3$). Dans le cas particulier du problème disjonctif, il est possible d'intégrer chaque type de règle (($\Re 2$) ou ($\Re 3$)) et ($\Re 1$), dans une règle unique. Celle-ci permet d'obtenir des déductions directes de caractéristiques séquentielles potentiellement actualisantes [Erschler et al. 80].

* *

*

Chapitre IV

Actualisation directe des dates limites par raisonnement énergétique

Le chapitre III propose des règles basées sur le séquencement de tâches, en conflit pour l'utilisation de ressources limitées. L'application de ces règles présente un aspect fortement combinatoire : tout d'abord, par la résolution *explicite* des conflits liés à la notion d'ensemble critique de tâches ; ensuite, par la combinaison de règles afin de trouver des caractéristiques séquentielles actualisantes.

Dans ce chapitre, on étudie la conséquence directe de la limitation de consommation d'énergie sur la localisation dans le temps des tâches. Les règles utilisées conduisent à résoudre *implicitement* les conflits et déduisent des caractéristiques temporelles numériques, immédiatement actualisantes et donc utilisables pour relancer le processus de propagation sur le graphe potentiels-bornes.

L'application de ces règles, comme pour celles présentées au chapitre III, constitue un simple processus de déduction logique à partir des contraintes initiales.

On se place toujours dans les hypothèses fixées en II.5.

IV.1 Principe du raisonnement

Le raisonnement énergétique permet de déduire de nouvelles contraintes sur les extrémités temporelles d'une tâche et d'actualiser ainsi directement leurs valeurs limites. Ces nouvelles contraintes temporelles peuvent relancer le processus de propagation. Le principe de ce raisonnement est détaillé ci-après.

Considérons un intervalle fournisseur $\Delta = (C_\Delta, F_\Delta, Q)$ et une tâche $i = (C_i, F_i, q_i)$. On appelle W^Δ l'énergie fournie par Δ, $\underline{w_i^\Delta}$ (respectivement $\overline{w_i^\Delta}$) la consommation

de i calée à gauche (resp. à droite) sur Δ ($w_i^\Delta = min(\underline{w}_i^\Delta, \overline{w}_i^\Delta)$). [1] On a donc :

$$\begin{cases} \underline{w}_i^\Delta = max[0, min(\underline{C}_i + D_i - C_\Delta, F_\Delta - \underline{C}_i, F_\Delta - C_\Delta, D_i)].q_i \\ \text{et} \\ \overline{w}_i^\Delta = max\{0, min(\overline{F}_i - C_\Delta, F_\Delta - (\overline{F}_i - D_i), F_\Delta - C_\Delta, D_i]\}.q_i \end{cases}$$

La différence $W^\Delta - \sum_{j \neq i} w_j^\Delta$ correspond à l'énergie disponible pour l'exécution de i. Ainsi, la règle suivante (illustrée par la figure IV.1) est valide :

$$\begin{cases} W^\Delta - \sum_{j \neq i} w_j^\Delta < \underline{w}_i^\Delta \\ \Longrightarrow \text{ actualisation de } \underline{C}_i \text{ ou détection d'incohérence} \end{cases}$$

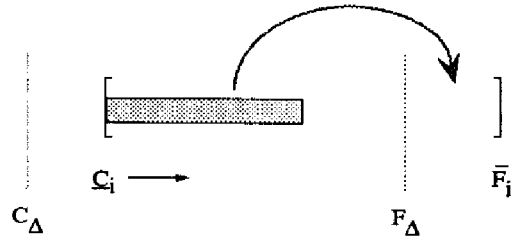


Figure IV.1

et la règle symétrique (figure IV.2) :

$$\begin{cases} W^\Delta - \sum_{j \neq i} w_j^\Delta < \overline{w}_i^\Delta \\ \Longrightarrow \text{ actualisation de } \overline{F}_i \text{ ou détection d'incohérence} \end{cases}$$

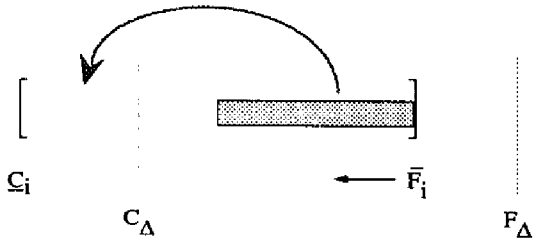


Figure IV.2

Ces deux premières règles permettent donc d'affiner les valeurs des dates limites $\underline{C}_i$ et $\overline{F}_i$ associées aux tâches, c'est-à-dire de caractériser plus précisément les ordonnancements admissibles. Leur efficacité est étroitement liée aux choix des bornes C_Δ et F_Δ de l'intervalle d'analyse associé aux règles.

Le paragraphe suivant s'intéresse particulièrement à un processus d'analyse centré autour d'une tâche où les bornes C_Δ et F_Δ dépendent directement des extrémités temporelles de la tâche.

[1] De manière plus générale, on peut définir la consommation de i commençant à C_i (resp. finissant à F_i), qui sera notée par convention $w_i^\Delta(C_i)$ (resp. $w_i^\Delta(F_i)$).
Notons bien que : $\underline{w}_i^\Delta = w_i^\Delta(\underline{C}_i)$ et $\overline{w}_i^\Delta = w_i^\Delta(\overline{F}_i)$.

IV.2 Analyse énergétique de l'intervalle associé à une tâche

IV.2.1 Introduction

L'idée retenue pour la définition des bornes C_Δ et F_Δ est de délimiter l'intervalle fournisseur par les extrémités temporelles de la tâche considérée. Lorsqu'on cherche à actualiser $\underline{C_i}$ (respectivement $\overline{F_i}$), on pose $C_\Delta = \underline{C_i}$ (resp. $F_\Delta = \overline{F_i}$), la borne F_Δ (resp. C_Δ), que nous appellerons désormais t_l, variant de $\underline{C_i}$ à $\overline{F_i}$ (resp. de $\overline{F_i}$ à $\underline{C_i}$).

1. $C_\Delta = \underline{C_i},\ F_\Delta = t_l \implies \underline{w_i^\Delta} = min(t_l - \underline{C_i}, D_i).q_i$
2. $F_\Delta = \overline{F_i},\ C_\Delta = t_l \implies \overline{w_i^\Delta} = min(\overline{F_i} - t_l, D_i).q_i$

IV.2.2 Règle de déduction

Soit $i = (C_i, F_i, q_i)$ une tâche telle que : $F_i - C_i = D_i$. Considérons l'intervalle fournisseur $IL = (\underline{C_i}, t_l, Q)$: [2]

$$\begin{cases} \text{si} & (t_l - \underline{C_i}).Q < \sum_{j \neq i} w_j^{IL} + min(t_l - \underline{C_i}, D_i).q_i \\ \\ \text{alors} & i \text{ ne peut commencer à la date } \underline{C_i}. \end{cases}$$

Il est donc nécessaire de repousser la tâche i vers la droite de manière à en rejeter une partie après t_l. Si l'on appelle S_{IL} l'énergie minimale consommée par i qu'il faut éjecter de l'intervalle IL, i doit finir à une date F_i telle que l'énergie requise par i après t_l soit au moins égale à S_{IL}, éventuellement additionnée (si $\underline{C_i} < t_l < \underline{C_i} + D_i$), de l'énergie requise par i au delà de t_l, avant éjection. Les conditions sur F_i dépendent donc de deux cas :

1. si $t_l \geq \underline{C_i} + D_i$ (figure IV.3)

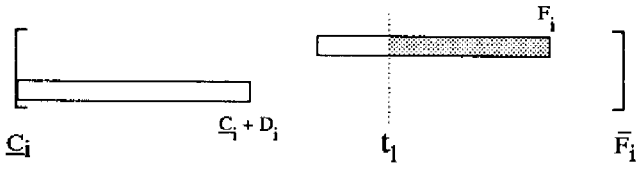


Figure IV.3

on a : $(F_i - t_l).q_i \geq S_{IL}$.

2. si $\underline{C_i} < t_l \leq \underline{C_i} + D_i$ (figure IV.4)

[2] On rappelle que dans IL, I se rapporte à $\underline{C_i}$ et L se rapporte à t_l (cf. III.3.2).

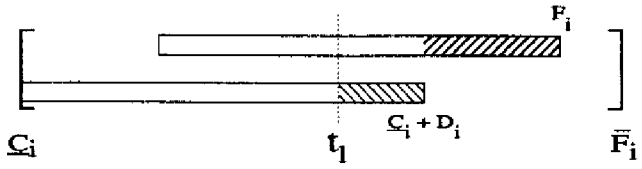


Figure IV.4

on a : $(F_i - t_l).q_i \geq S_{IL} + (\underline{C_i} + D_i - t_l).q_i$
$\iff [F_i - (\underline{C_i} + D_i)].q_i \geq S_{IL}$.

D'une manière générale, la règle d'inférence suivante est valide :

$$(\Re 4) \begin{cases} \text{si} & (t_l - \underline{C_i}).Q < \sum_{j \neq i} w_j^{IL} + min(t_l - \underline{C_i}, D_i).q_i \\ \text{alors} & [F_i - max(t_l, \underline{C_i} + D_i)].q_i \geq S_{IL} \quad \text{i.e.} \quad F_i \geq max(t_l, \underline{C_i} + D_i) + \dfrac{S_{IL}}{q_i} \end{cases}$$

où $$S_{IL} = \sum_{j \neq i} w_j^{IL} + min(t_l - \underline{C_i}, D_i).q_i - (t_l - \underline{C_i}).Q > 0.$$

Cette règle peut entraîner une actualisation directe de $\underline{F_i}$ (donc de $\underline{C_i}$) à partir de laquelle il est possible de relancer le processus de propagation. Elle peut également permettre de détecter une incohérence.

Une règle symétrique peut être établie en considérant un intervalle fournisseur $LI = (t_l, \overline{F}_i, Q)$:

$$(\Re' 4) \begin{cases} \text{si} & (\overline{F}_i - t_l).Q < \sum_{j \neq i} w_j^{LI} + min(\overline{F}_i - t_l, D_i).q_i \\ \text{alors} & [C_i - min(t_l, \overline{F}_i - D_i)].q_i \leq -S_{LI} \quad \text{i.e.} \quad C_i \leq min(t_l, \overline{F}_i - D_i) - \dfrac{S_{LI}}{q_i} \end{cases}$$

où $$S_{LI} = \sum_{j \neq i} w_j^{LI} + min(\overline{F}_i - t_l, D_i).q_i - (\overline{F}_i - t_l).Q > 0.$$

Cette règle peut entraîner une actualisation directe de $\overline{C_i}$ (donc de $\overline{F_i}$) susceptible de relancer le processus de propagation.

Pour des raisons de simplicité, il semble intéressant, dans un premier temps, de considérer des instants t_l qui correspondent à des valeurs directement liées aux bornes d'intervalles. On peut choisir ainsi : pour $(\Re 4)$, $t_l \equiv \overline{F}_j \ \forall j \neq i$ et $\overline{F}_j > \underline{C_i}$ et pour $(\Re' 4)$, $t_l \equiv \underline{C_j} \ \forall j \neq i$ et $\underline{C_j} < \overline{F}_i$. Une application est présentée dans l'exemple suivant.

IV.2.3 Exemple

On considère trois tâches x, y, z utilisant une seule ressource en quantité unitaire ($q_i = 1 \ \forall i = x, y, z$) et caractérisées par les valeurs définies dans le tableau IV.1.

i	D_i	$\underline{C}_i$	$\overline{F}_i$
x	2	1	4
y	1	0	4
z	1	1	4

Tableau IV.1

La ressource est supposée disponible en quantité unitaire sur tout l'intervalle $[0,4]$. On applique la règle $(\Re'4)$ pour l'analyse énergétique de l'intervalle associé à y, avec $t_l = \underline{C}_z$. Si l'on considère l'intervalle fournisseur $(\underline{C}_z, \overline{F}_y, Q) = (1,4,1)$, l'application de $(\Re'4)$ à la tâche $(C_y, F_y, 1)$ donne :

$$3 \times 1 < 2+1+1 \quad \Rightarrow \quad C_y \leq \underline{C}_z - \frac{1}{1} = 0$$

qui permet d'actualiser la valeur $\overline{C}_y$ de 3 à 0, et donc $\overline{F}_y$ de 4 à 1 (cf. figure IV.5).

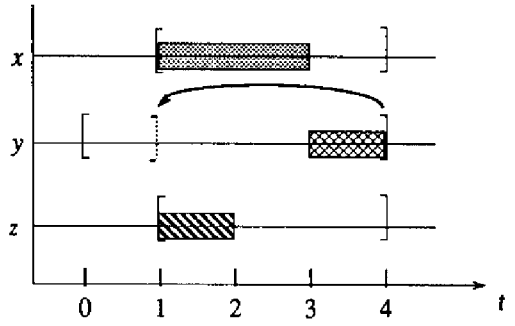


Figure IV.5 : Actualisation de $\overline{F}_y$

Le but du paragraphe suivant est d'étudier des valeurs de t_l moins triviales que les valeurs extrêmes associées aux bornes d'intervalles ($\overline{F}_j$ pour $(\Re 4)$ et $\underline{C}_j$ pour $(\Re'4)$) et qui présentent un intérêt vis-à-vis des déductions qu'elles permettent.

IV.2.4 Présentation du principe de l'analyse

Considérons une tâche $i = (C_i, F_i, q_i)$ de durée $D_i = F_i - C_i$ et un intervalle fournisseur $IL = (\underline{C}_i, t_l, Q)$. On définit, pour toute valeur de t_l, les grandeurs suivantes :

- $W^{IL} = (t_l - \underline{C}_i).Q$: énergie pouvant être fournie par IL,
- $\sum_{j \neq i} w_j^{IL}$: consommation obligatoire de toutes les tâches autres que i sur IL,
- $W_i^{IL} = W^{IL} - \sum_{j \neq i} w_j^{IL}$: énergie disponible sur IL pour l'exécution de i,

- $w_i^{IL}(C_i) = min(t_l - C_i, D_i).q_i$: énergie utilisée par i commençant à C_i sur IL.

La forme de l'expression définissant $w_i^{IL}(C_i)$ est semblable à celle de $\underline{w_i^{\Delta}}$, donnée au paragraphe IV.2.1. Elle est illustrée par la figure IV.6.

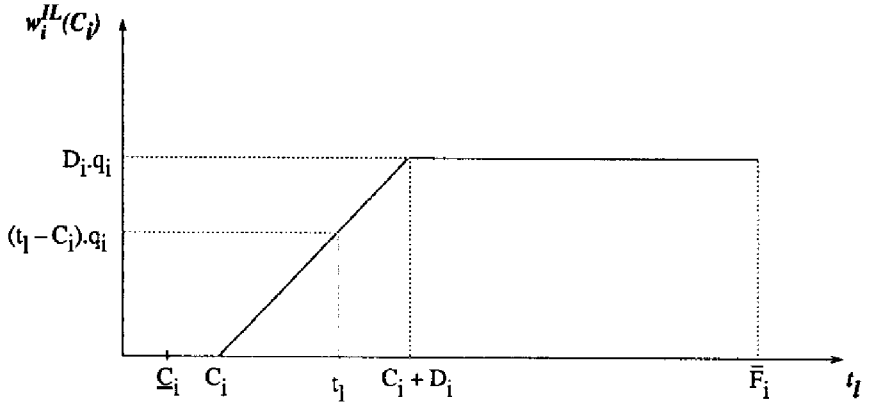


Figure IV.6

Un exemple d'évolution de ces différentes grandeurs pour t_l variant de $\underline{C_i}$ à $\overline{F_i}$ est représenté sur la figure IV.7.

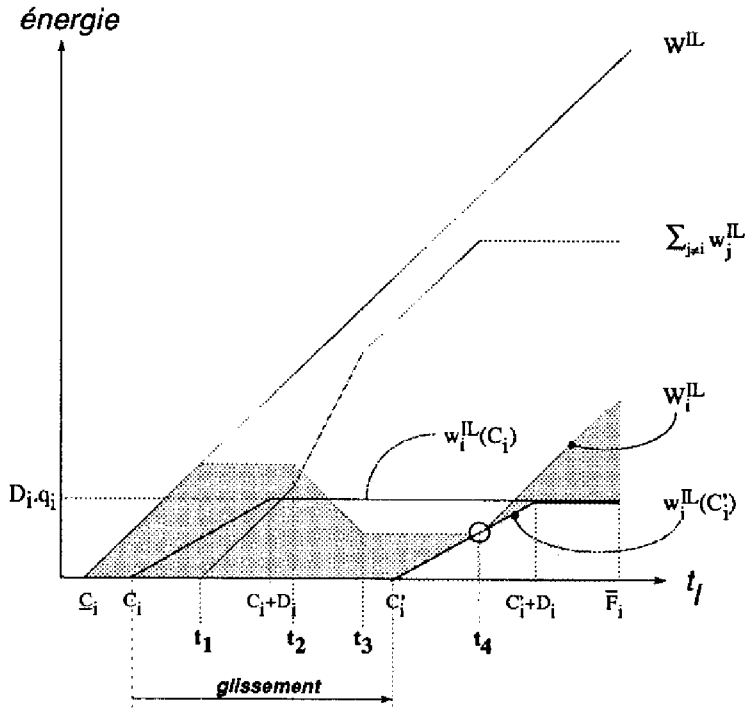


Figure IV.7

$t_1, ..., t_4$ représentent les instants correspondant aux points de cassure de la courbe donnant W_i^{IL} en fonction de t_l. [3] Ces instants sont appelés *instants remarquables* de l'intervalle fournisseur $I = (\underline{C_i}, \overline{F_i}, q_i)$. L'étude de la courbe W_i^{IL} donne une information sur la répartition de l'énergie disponible et sur son influence quant à la localisation de la tâche i (localisation au plus tôt dans le cas présent). En effet, pour que l'exécution de i commençant à la date C_i soit possible, il est nécessaire d'avoir $W_i^{IL} \geq w_i^{IL}(C_i)$ et ce à chaque instant t_l. Si cette inégalité n'est pas vérifiée, la fonction $w_i^{IL}(C_i)$ étant monotone et non décroissante, il faut la faire "glisser" vers la droite – comme indiqué sur la figure IV.7 –, la tâche ne pouvant plus commencer à la date C_i mais seulement à une date $C_i' > C_i$. Ceci entraîne donc une actualisation de la date de début au plus tôt de i. Cette démarche est appelée *analyse énergétique au plus tôt.*

Il est possible de mener une étude symétrique en considérant un intervalle fournisseur $LI = (t_l, \overline{F_i}, Q)$. Si la condition $W_i^{LI} \geq w_i^{LI}(F_i)$, t_l variant de $\overline{F_i}$ à $\underline{C_i}$ n'est pas vérifiée, il faut actualiser la date de fin au plus tard de la tâche i. On parle alors d'*analyse énergétique au plus tard.*

Dans ce chapitre seule l'analyse au plus tôt est détaillée. L'analyse au plus tard étant similaire, seuls les résultats propres à ce type d'analyse sont brièvement présentés au paragraphe IV.2.6 et appliqués lors du traitement des exemples (cf. paragraphe IV.2.7).

Compte tenu de l'objectif qui est d'actualiser au maximum $\underline{C_i}$, l'approche suivante est proposée [les caractères entre crochets sont des renvois à la figure IV.7, cités à titre d'exemple]:

1. rechercher tous les instants remarquables t_l de l'intervalle fournisseur I $[t_1, t_2, t_3, t_4]$,
2. rechercher t_m qui est le plus grand instant remarquable tel que, sur un intervalle fournisseur $IM = (\underline{C_i}, t_m, Q)$, $W_i^{IM} < w_i^{IM}(\underline{C_i})$, c'est-à-dire tel que l'exécution de i commençant à la date $\underline{C_i}$ n'est pas possible $[t_m \equiv t_4]$,
3. si t_m existe, actualiser $\underline{C_i}$ [en C_i'].

> Remarque 1 : on montre que, dans certains cas, cette procédure ne permet pas de trouver l'actualisation la plus forte. A ce sujet, on propose certaines extensions au paragraphe IV.3.

IV.2.5 Analyse énergétique au plus tôt

IV.2.5.1 Instants remarquables et conditions associées

Soit n le nombre d'instants remarquables d'un intervalle fournisseur $I = (\underline{C_i}, \overline{F_i}, Q)$. On note t_l les différentes valeurs des instants remarquables avec $\underline{C_i} < t_l \leq \overline{F_i}$ et

[3] Etant donné que l'on ne considère que des intervalles temps-ressource uniformes, la courbe W^{IL} est linéaire et la courbe $\sum_{j \neq i} w_j^{IL}$ l'est par morceaux. On peut remarquer que les points de cassure de W_i^{IL} sont les mêmes que ceux de $\sum_{j \neq i} w_j^{IL}$.

$t_l < t_{l'}$ ssi $l < l'$. Les intervalles à considérer pour l'analyse énergétique au plus tôt sont donc de la forme :

$$[\underline{C}_i, t_1], [\underline{C}_i, t_2], ..., [\underline{C}_i, t_l], ..., [\underline{C}_i, t_n].$$

Soit $O^I = \{j \ / \ w_j^I \neq 0\}$, l'ensemble des tâches qui ont une consommation obligatoire non nulle sur l'intervalle I. Considérons une tâche $j = (C_j, F_j, q_j)$, de durée D_j, telle que : $j \neq i$ et $j \in O^I$. Au sein de I, on peut associer à j les instants remarquables suivants :

- $t_l(j) = \overline{F}_j - D_j$: date de début au plus tard de j,
- $t_l(j) = \overline{F}_j + \underline{C}_j - \underline{C}_i$: instant t_l où la consommation obligatoire de j sur $(\underline{C}_i, t_l, Q)$ est la même que la tâche soit calée à droite ou à gauche : $[t_l - (\overline{F}_j - D_j)].q_j = (\underline{C}_j + D_j - \underline{C}_i).q_j$,
- $t_l(j) = \overline{F}_j$: date de fin au plus tard de j,
- $t_l(j) = \underline{C}_j + D_j$: date de fin au plus tôt de j.

Ces instants ne correspondent effectivement à des points de cassure de la courbe W_i^{IL} que si certaines conditions spécifiques sont satisfaites (en plus des faits qu'ils soient compris entre $\underline{C}_i$ et $\overline{F}_i$ et que $j \in O^I$). Ces conditions sont énoncées dans le tableau IV.2.

instant	condition	
$\overline{F}_j - D_j$	—	
$F_j + \underline{C}_j - \underline{C}_i$	$\underline{C}_j < \underline{C}_i < \overline{F}_j - D_j$	(γ_2)
$\overline{F}_j$	$\underline{C}_i \leq \underline{C}_j < \overline{F}_j < \overline{F}_i$	(γ_3)
$\underline{C}_j + D_j$	$\overline{F}_j - D_j < \underline{C}_i$	(γ_4)

Tableau IV.2

Démonstration :

(γ_2) 1. pour caler j à gauche, il est nécessaire d'avoir : $\underline{C}_j < \underline{C}_i$.

2. de plus, pour caler j à droite et pour retenir cette situation, il est nécessaire d'avoir $\overline{F}_j - D_j > \underline{C}_i$. En effet, si $\overline{F}_j - D_j \leq \underline{C}_i$, la consommation de j calée à droite est nécessairement supérieure ou égale à la consommation de j calée à gauche.

(γ_3) est évident.

(γ_4) si $\overline{F}_j - D_j \leq \underline{C}_i$ la consommation de j calée à gauche est nécessairement inférieure ou égale à la consommation de j calée à droite.

Les situations correspondant aux conditions $(\gamma_2, \gamma_3, \gamma_4)$ sont illustrées sur la figure IV.8.

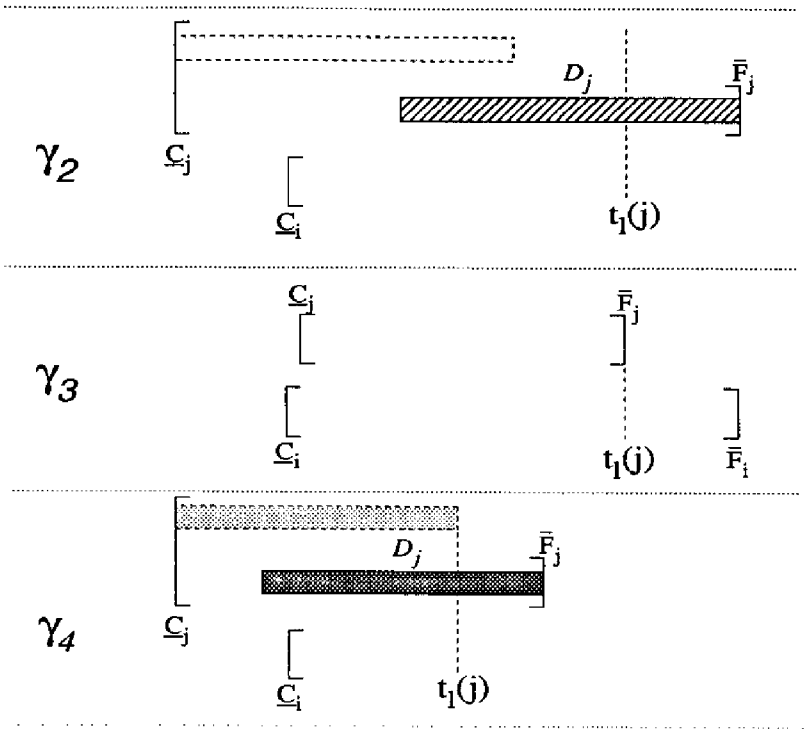


Figure IV.8

Remarque 2 : il existe un caractère exclusif entre certaines conditions associées aux instants : (γ_2) ou (γ_3), (γ_2) ou (γ_4), (γ_3) ou (γ_4)... De ce fait, il n'est possible, lors de l'étude de chaque tâche j, de retenir qu'au maximum *deux* instants remarquables $t_l(j)$ issus du tableau IV.2.

De plus, $\overline{F}_i$ est toujours un instant remarquable à considérer ; donc, $t_n = \overline{F}_i$.

IV.2.5.2 Choix de t_m

Soient la tâche $i = (C_i, F_i, q_i)$ et $IL = (\underline{C}_i, t_l, Q)$ un intervalle fournisseur ; on a vu (cf. paragraphe IV.2.4) que l'exécution de i n'est possible que pour $W_i^{IL} \geq w_i^{IL}(C_i)$ pour chaque instant. On va donc chercher, parmi les instants remarquables déjà trouvés, la valeur $t_m = \max_l t_l$ telle que l'on a :

$$W_i^{IM} < w_i^{IM}(\underline{C}_i) \qquad \text{(IV.1)}$$

c'est-à-dire :

$$(t_m - \underline{C}_i).Q - \sum_{j \neq i} w_j^{IM} < min(t_m - \underline{C}_i, D_i).q_i.$$

Remarque 3 : pour éviter de considérer tous les instants remarquables, il semble opportun de passer en revue les différentes valeurs de t_l dans l'ordre décroissant des l. Ainsi, le premier t_l qui vérifie la condition est le plus grand.

IV.2.5.3 Actualisation de $\underline{C_i}$

On considère la même tâche i et l'intervalle fournisseur $IM = (\underline{C_i}, t_m, Q)$ pour lequel la première règle vue en IV.1 s'applique. Il est donc nécessaire de repousser la tâche i vers la droite de manière à en rejeter une partie après t_m. En appliquant la règle ($\Re 4$), on obtient la condition :

$$[F_i - max(t_m, \underline{C_i} + D_i)].q_i \geq S_{IM} \quad \text{i.e.} \quad F_i \geq max(t_m, \underline{C_i} + D_i) + \frac{S_{IM}}{q_i}$$

où $$S_{IM} = w_i^{IM}(\underline{C_i}) - W_i^{IM}.$$

S_{IM} est l'énergie minimale consommée par l'intervalle i qu'il faut éjecter de l'intervalle IM.

Ceci permet une actualisation de $\underline{F_i}$, ou la détection d'une incohérence. Cette actualisation amène à une nouvelle définition des intervalles IM, et le processus peut être réitéré. L'évolution de la valeur actualisée de $\underline{F_i}$ est représentée par la relation récurrente définie par :

$$\underline{F_i}^{(0)} = \underline{F_i} \quad ; \quad \underline{F_i}^{(p+1)} = \frac{S_{IM^{(p)}}}{q_i} + max(t_m^{(p)}, \underline{C_i}^{(p)} + D_i)$$

avec $IM^{(p)} = (\underline{C_i}^{(p)}, t_m^{(p)}, Q)$. Comme $\underline{F_i} = \underline{C_i} + D_i$, on peut écrire :

$$\begin{aligned} \underline{C_i}^{(p+1)} &= \frac{S_{IM^{(p)}}}{q_i} + max(t_m^{(p)}, \underline{C_i}^{(p)} + D_i) - D_i \\ &= \frac{\sum_{j \neq i} w_j^{IM^{(p)}} + min(t_m^{(p)} - \underline{C_i}^{(p)}, D_i) - (t_m^{(p)} - \underline{C_i}^{(p)}).Q}{q_i} + max(t_m^{(p)}, \underline{C_i}^{(p)} + D_i) - D_i \end{aligned}$$

Rappel : $$\begin{cases} min(a,b) &= \dfrac{a+b}{2} - \dfrac{|a-b|}{2} \\ max(a,b) &= \dfrac{a+b}{2} + \dfrac{|a-b|}{2} \end{cases}$$

$$\Longrightarrow \underline{C_i}^{(p+1)} = \frac{\sum_{j \neq i} w_j^{IM^{(p)}} - (t_m^{(p)} - \underline{C_i}^{(p)}).Q}{q_i} + t_m^{(p)}$$

Soit en posant, $w_j^{IM^{(p)}} = \tau_j^{IM^{(p)}}.q_j$ (cf. paragraphe II.3.1.2)

et $$\begin{cases} \alpha_i = \dfrac{Q}{q_i} & \geq 1 \\ \beta_i^j = \dfrac{q_j}{q_i} & > 0 \end{cases}$$

on obtient :

$$\underline{C_i}^{(p+1)} = \underline{C_i}^{(p)}.\alpha_i + t_m^{(p)}.(1 - \alpha_i) + \sum_{j \neq i} \tau_j^{IM^{(p)}}.\beta_i^j \qquad \text{(IV.2)}$$

Etant donné que la valeur $\underline{C_i}^{(p)}$ ne peut diminuer et qu'elle est, de plus, bornée par la quantité $\overline{F_i} - D_i$, la suite $\underline{C_i}^{(p+1)}$ converge asymptotiquement vers une valeur

limite $\underline{C}_i^*$, en un nombre fini ou infini d'itérations[4] (dans ce dernier cas, on peut stopper le processus par une opération de troncature[5]) ; voir les exemples 2 et 3. D'après l'équation (IV.2), la valeur $\underline{C}_i^*$ est telle que :

$$\underline{C}_i^* = \underline{C}_i^*.\alpha_i + t_m^*.(1-\alpha_i) + \sum_{j \neq i} \mathcal{T}_j^{IM^*}.\beta_i^j$$

soit

$$\underline{C}_i^* = t_m^* + \frac{1}{(1-\alpha_i)} . \sum_{j \neq i} \mathcal{T}_j^{IM^*}.\beta_i^j$$

Compte tenu du nombre important de variables inconnues, cette expression ne permet pas une utilisation opérationnelle pour déterminer la valeur $\underline{C}_i^*$. Toutefois, dans le cas particulier du problème disjonctif pur, les coefficients α_i et β_i^j sont égaux à 1 et on peut écrire, d'après l'équation (IV.2) :

$$\underline{C}_i^{(p+1)} = \underline{C}_i^{(p)} + \sum_{j \neq i} \mathcal{T}_j^{IM^{(p)}}.$$

La valeur de $\underline{C}_i^*$ est alors obtenue lorsque $\sum_{j \neq i} \mathcal{T}_j^{IM^*} = 0$ (cf. Exemple 3).

IV.2.6 Analyse énergétique au plus tard

Ce paragraphe a pour unique but de présenter rapidement les principaux résultats de l'analyse énergétique au plus tard sans les expliquer, le raisonnement étant identique à celui de l'analyse énergétique au plus tôt.

IV.2.6.1 Instants remarquables et conditions associées

On rappelle que chaque instant est compris entre $\underline{C}_i$ et $\overline{F}_i$ et que $j \in \mathcal{O}^I$.

instant	condition
$\underline{C}_j + D_j$	—
$\underline{C}_j - \overline{F}_i + \overline{F}_j$	$\underline{C}_j + D_j < \overline{F}_i < \overline{F}_j$
$\underline{C}_j$	$\underline{C}_i < \underline{C}_j < \overline{F}_j \leq \overline{F}_i$
$\overline{F}_j - D_j$	$\overline{F}_i < \underline{C}_j + D_j$

Tableau IV.3

[4] En l'absence d'itérations, la valeur actualisée de $\underline{C}_i$ est notée $\underline{C}_i^1$.

[5] L'étude sur cette opération reste à être approfondie ; elle consiste à s'assurer qu'il n'existe pas d'instant remarquable dans l'intervalle d'analyse dont une seule borne varie ($[\underline{C}_i, t_m]$ par exemple). Il semble que cette absence d'instant remarquable soit liée à l'existence d'une partie obligatoire commune à toutes les tâches j différentes de i. $\underline{C}_i^*$ doit alors être actualisée à la borne supérieure de l'intervalle obligatoire associé à cette partie obligatoire.

IV.2.6.2 Choix de t_m

Soient la tâche $i = (C_i, F_i, q_i)$ et l'intervalle fournisseur $LI = (t_l, \overline{F}_i, Q)$. L'exécution de i n'est possible que si $W_i^{LI} \geq w_i^{LI}(F_i)$ à chaque instant. On cherche la valeur $t_m = \min_l t_l$ telle que l'on a $W_i^{MI} < w_i^{MI}(\overline{F}_i)$, c'est-à-dire :

$$(\overline{F}_i - t_m).Q - \sum_{j \neq i} w_j^{MI} < max(\overline{F}_i - t_m, D_i).q_i.$$

IV.2.6.3 Actualisation de $\overline{F}_i$

La règle d'inférence ($\Re'4$) permet d'obtenir la condition nécessaire suivante :

$$C_i \leq min(t_m, \overline{F}_i - D_i) - \frac{S_{MI}}{q_i}$$

où

$$S_{MI} = w_i^{MI}(\overline{F}_i) - W_i^{MI}.$$

Ce qui donne la formule d'actualisation de $\overline{F}_i$ suivante :

$$\overline{F}_i^{(p+1)} = \overline{F}_i^{(p)}.\alpha_i + t_m^{(p)}.(1 - \alpha_i) - \sum_{j \neq i} \tau_j^{MI^{(p)}}.\beta_i^j \qquad \text{(IV.3)}$$

qui devient dans le cas d'utilisation de machines à capacité unitaire :

$$\overline{F}_i^{(p+1)} = \overline{F}_i^{(p)} - \sum_{j \neq i} \tau_j^{IM^{(p)}}.$$

IV.2.7 Exemples

IV.2.7.1 Exemple 1 : problème disjonctif pur

On considère cinq tâches a, b, e, g, h utilisant une seule ressource en quantité unitaire et caractérisées par les valeurs définies dans le tableau IV.4.

i	a	b	e	g	h
$\underline{C}_i$	2	7	1	5	1
$\overline{F}_i$	8	14	9	13	14
D_i	2	1	5	2	2

Tableau IV.4

Analyse énergétique au plus tôt associée à h

Les intervalles fournisseurs considérés sont donc de la forme :

$$HL = (\underline{C}_h, t_l, q_h) \text{ avec } 1 < t_l < 14.$$

Pour illustration, l'évolution des différentes énergies pour la tâche h est représentée par la figure IV.9.

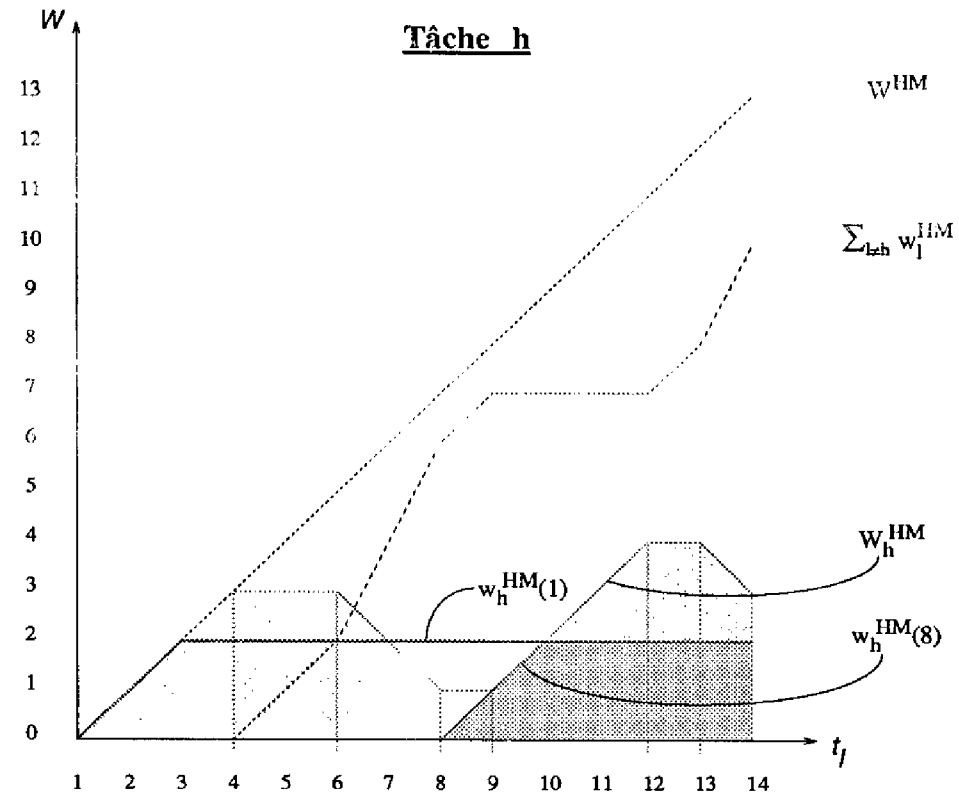


Figure IV.9

L'instant remarquable t_m est déterminé par la plus grande valeur de t_l telle que $W_i^{IM} < w_i^{IM}(\underline{C}_i)$. Conformément au paragraphe IV.2.5.1, on obtient le tableau IV.5.

t_l	14	13	12	9	8	6	4
W_h^{HL}	3	4	4	1	—	—	—
$w_h^{HL}(1)$	2	2	2	2	—	—	—
$W_h^{HL} < w_h^{HL}(1)$	NON	NON	NON	OUI	—	—	—

Tableau IV.5

On a ainsi : $t_m = 9$ et donc $HM = (\underline{C}_h, t_m, q_h) = (1, 9, 1)$.
La consommation obligatoire sur HM s'écrit : $\sum_{j \neq h} \mathcal{T}_j^{HM} = \mathcal{T}_a^{HM} + \mathcal{T}_e^{HM} = 7$.

D'où : $\boxed{\underline{C}'_h = 1 + 7 = 8}$

Analyse énergétique au plus tôt associée à g
Pour g, $5 < t_l \leq 13$ (cf. figure IV.10 et tableau IV.6).

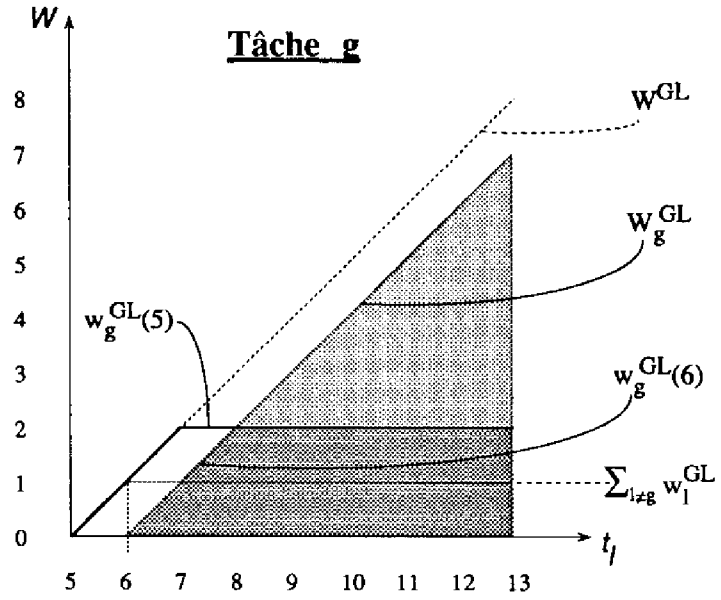


Figure IV.10

t_l	13	6
W_g^{GL}	7	0
$w_g^{GL}(5)$	2	1
$W_g^{GL} < w_g^{GL}(5)$	NON	OUI

Tableau IV.6

$t_m = 6 \Longrightarrow GM = (5,6,1)$.
$\sum_{j \neq g} \tau_j^{GM} = \tau_e^{GM} = 1$.

D'où : $\boxed{\underline{C}'_g = 5 + 1 = 6}$

Le tableau IV.7 montre les résultats finaux de l'étude énergétique au plus tôt et au plus tard, cette dernière n'entraînant pas d'actualisation dans cet exemple.

i	a	b	e	g	h
$\underline{C}_i$	2	7	1	6	8
$\overline{F}_i$	8	14	9	13	14
D_i	2	1	5	2	2

Tableau IV.7

IV.2.7.2 Exemple 2 : problème cumulatif (cas de deux itérations)

On considère désormais trois tâches utilisant une unité d'une ressource disponible en une quantité supérieure à un (tableau IV.8).

i	C_i	$\overline{F_i}$	D_i	q_i
a	0	5	4	1
b	0	3	2	1
e	0	5	3	1

Q
2

Tableau IV.8

Analyse énergétique au plus tôt associée à e

Itération 1

Les intervalles d'intérêt satisfont : $(0, t_l, 2)$ avec $0 < t_l \leq 5$ (cf. figure IV.11 et tableau IV.9).

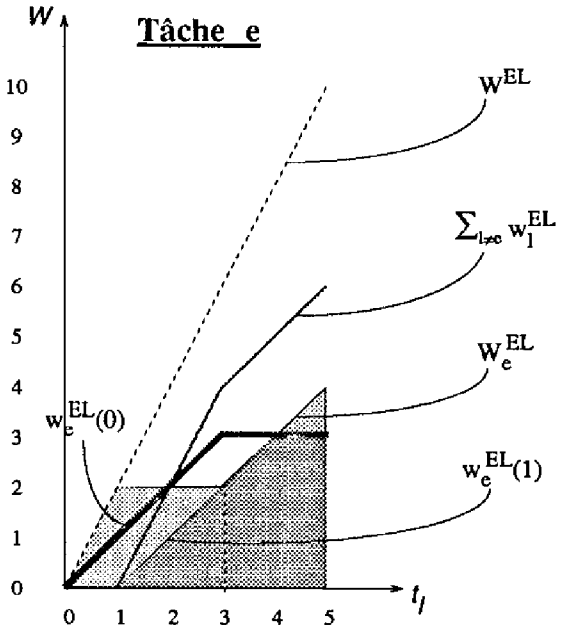


Figure IV.11

t_l	5	3	1
W_e^{EL}	4	2	—
$w_e^{EL}(0)$	3	3	—
$W_e^{EL} < w_e^{EL}(0)$	NON	OUI	—

Tableau IV.9

$t_m^{(1)} = 3 \Longrightarrow EM^{(1)} = (0,3,2)$.
$\sum_{j \neq e} \tau_j^{EM^{(1)}}.\beta_e^j = \tau_a^{EM^{(1)}}.\beta_e^a + \tau_b^{EM^{(1)}}.\beta_e^b = 2 \times 1 + 2 \times 1 = 4$ et $\alpha_e = 2$.

D'où : $\underline{C}_e^{(1)} = 0 + 3 \times (1-2) + 4 = 1$

A partir de cette nouvelle valeur, on poursuit le raisonnement sur e.

Itération 2

Désormais, on a : $EL = (1, t_l, 2)$ avec $1 < t_l \leq 5$ (figure IV.12 et tableau IV.10)

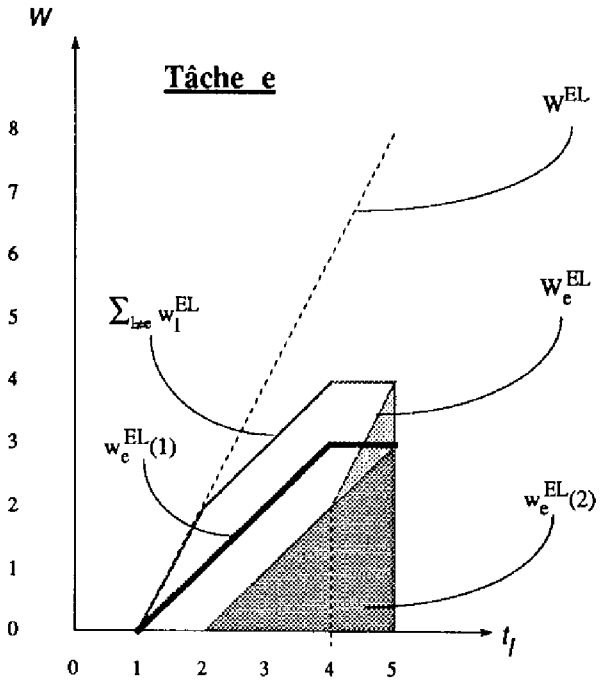


Figure IV.12

t_l	5	4	2
W_e^{EL}	4	2	—
$w_e^{EL}(1)$	3	3	—
$W_e^{EL} < w_e^{EL}(1)$	NON	OUI	—

Tableau IV.10

$t_m^{(2)} = 4 \Longrightarrow EM^{(2)} = (1,4,2)$.
$\sum_{j \neq e} \tau_j^{EM^{(2)}}.\beta_e^j = \tau_a^{EM^{(2)}}.\beta_e^a + \tau_b^{EM^{(2)}}.\beta_e^b = 3 \times 1 + 1 \times 1 = 4$ et $\alpha_e = 2$.

D'où : $\boxed{\underline{C}_e^{(2)} = 1 \times 2 + 4 \times (1-2) + 4 = 2}$

On peut montrer que $\underline{C}_e^{(3)} = \underline{C}_e^{(2)} = \underline{C}_e^*$.

Analyse énergétique au plus tard associée à b

Les nouvelles données du problème sont les suivantes :

i	C_i	$\overline{F}_i$	D_i	q_i
a	0	5	4	1
b	0	3	2	1
e	2	5	3	1

Q
2

Tableau IV.11

On a : $LB = (t_l, 3, 2)$ avec $0 \leq t_l < 3$ (figure IV.13 et tableau IV.12).

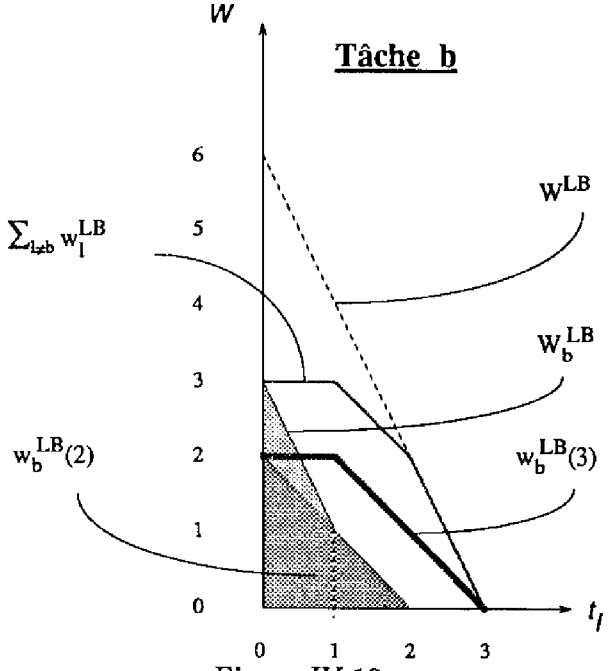


Figure IV.13

t_l	0	1	2
W_b^{LB}	3	1	—
$w_b^{LB}(3)$	3	3	—
$W_b^{LB} < w_b^{LB}(3)$	NON	OUI	—

Tableau IV.12

$t_m = 1 \Longrightarrow MB = (1, 3, 2)$.
$\sum_{j\neq b} \tau_j^{EM}.\beta_b^j = \tau_a^{EM}.\beta_b^a + \tau_e^{EM}.\beta_b^e = 2 \times 1 + 1 \times 1 = 3$ et $\alpha_b = 2$.

L'analyse s'effectue sans itération ; on obtient l'actualisation :

$$\boxed{\overline{F}_b' = 3 \times 2 + 1 \times (1 - 2) - 3 = 2}$$

Seules les analyses précédentes sont actualisantes. Les résultats finaux sont donc ceux inscrits dans le tableau IV.13.

i	$\underline{C_i}$	$\overline{F_i}$	D_i	q_i
a	0	5	4	1
b	0	2	2	1
e	2	5	3	1

Q
2

Tableau IV.13

IV.2.7.3 Exemple 3 : problème cumulatif (cas d'un nombre infini d'itérations)

Voici enfin un petit exemple (tableau IV.14), qui montre un processus d'actualisation de dates qui converge selon un nombre infini d'itérations.

i	$\underline{C_i}$	$\overline{F_i}$	D_i	q_i
a	0	1	1	1
b	0	3	2	2

Q
2

Tableau IV.14

Analyse énergétique au plus tôt associée à b

Itération 1

Pour b, on a : $0 < t_l \leq 3$. On obtient la figure IV.14 et le tableau IV.15.

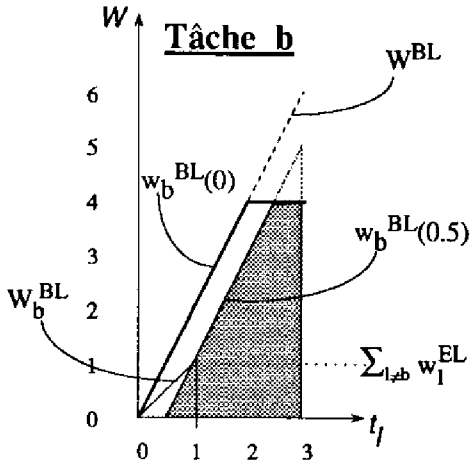


Figure IV.14

t_l	3	1
W_b^{BL}	5	1
$w_b^{BL}(0)$	4	2
$W_b^{BL} < w_b^{BL}(0)$	NON	OUI

Tableau IV.15

$t_m^{(1)} = 1 \Longrightarrow BM^{(1)} = (0,1,2)$.
$\sum_{j \neq b} \mathcal{T}_j^{BM^{(1)}}.\beta_b^j = \mathcal{T}_a^{BM^{(1)}}.\beta_b^a = 0.5$ et $\alpha_b = 1$.

D'où : $\underline{C}_b^{(1)} = 0 + 0 + 0.5 = 0.5$

Itération 2

On a le nouveau problème caractérisé par le tableau IV.16.

i	$\underline{C}_i$	F_i	D_i	q_i
a	0	1	1	1
b	0.5	3	2	2

Q
2

Tableau IV.16

L'évolution des énergies est pour : $0.5 < t_l \leq 3$ (cf. figure IV.15).

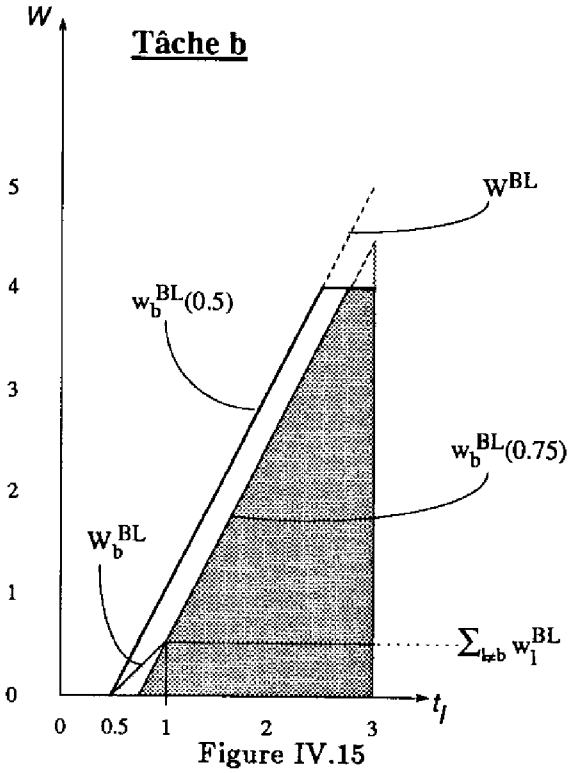


Figure IV.15

On a encore $t_m^{(2)} = 1$ et donc, $BM^{(2)} = (0.5, 1, 2)$.
$\sum_{j \neq b} \tau_j^{BM^{(2)}}.\beta_b^j = \tau_a^{BM^{(2)}}.\beta_b^a = 0.25$

D'où : $\underline{C}_b^{(2)} = 0.5 + 0 + 0.25 = 0.75$

$\vdots$

et la procédure peut être réitérée, les valeurs $\underline{C}_b^{(p)}$ tendant vers une valeur telle que : $\sum_{j \neq b} \tau_j^{BM^*}.\beta_b^j = 0$.

On obtient alors :

$$\boxed{\underline{C}_b^* = 1}$$

Le tableau IV.17 représentent les résultats en fin d'analyse.

i	$\underline{C}_i$	$\overline{F}_i$	D_i	q_i
a	0	1	1	1
b	1	3	2	2

Q
2

Tableau IV.17

IV.3 Affinements possibles du processus d'analyse

Ce paragraphe met en évidence, sur des exemples, certaines insuffisances du processus d'analyse décrit précédemment et propose des voies possibles pour y remédier.

IV.3.1 Choix de l'instant remarquable pour l'actualisation

Considérons l'exemple suivant : nous sommes en présence de quatre tâches utilisant une ressource disponible en deux exemplaires et dont les caractéristiques sont indiquées dans le tableau IV.18.

i	$\underline{C}_i$	$\overline{F}_i$	D_i	q_i
m	0	7	4	2
n	0	6	2	2
o	0	10	1	2
p	0	13	7	1

Q
2

Tableau IV.18

Analyse énergétique au plus tôt associée à p (figure IV.16)

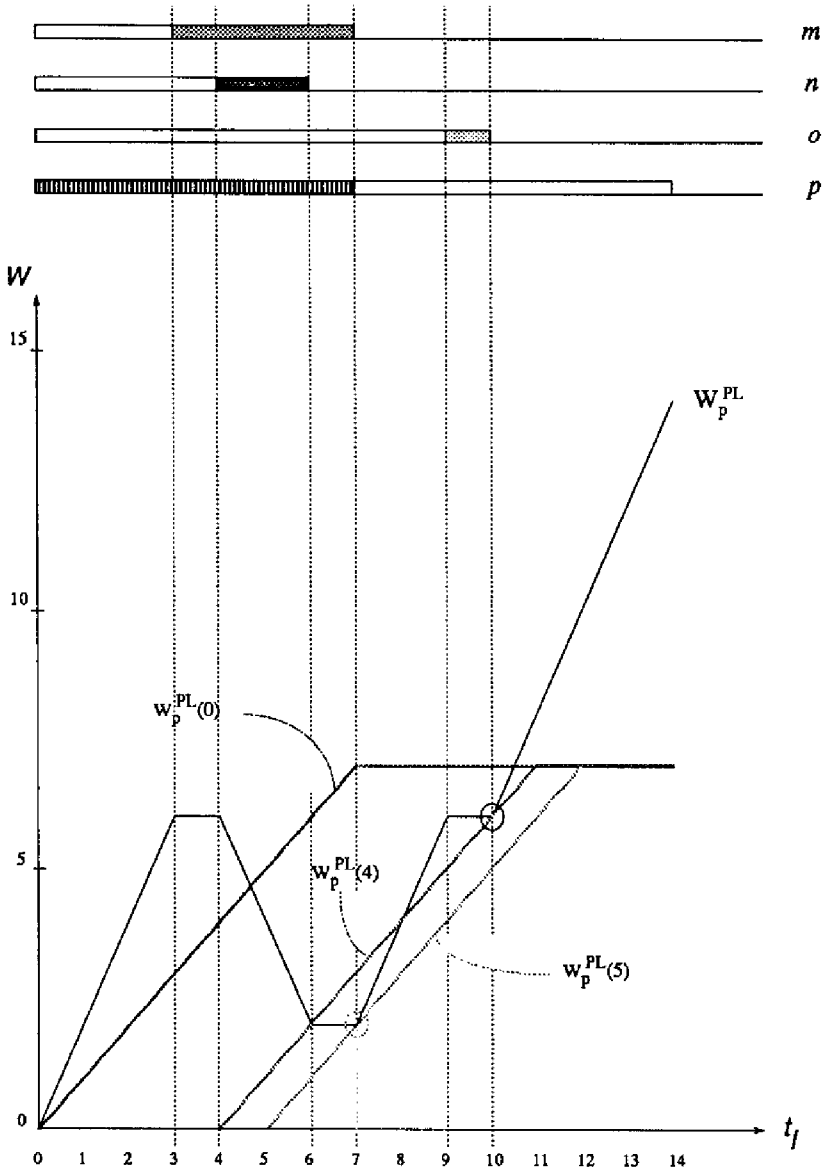


Figure IV.16

On s'aperçoit que la procédure mise en œuvre provoque une actualisation de $\underline{C_p}$ qui s'avère insuffisante. En effet, l'actualisation trouvée est $\underline{C_p'} = 4$, à partir d'une valeur $t_m = 10$; mais on remarque qu'à $t_l = 7$, $W_p^{PL} < w_p^{PL}(4)$. En revanche, pour une actualisation $\underline{C_p'} = 5$, qui peut être obtenue en choisissant $t_m = 7$, il apparaît sur le graphique de la figure IV.16 que l'actualisation est maximale. Il s'en suit que la procédure peut être améliorée et, pour cela, deux approches nous semblent possibles :

1. construction d'une stratégie autre que celle proposée en IV.2.4 page 75, pour le choix de t_m ;

2. énumération des instants remarquables qui entraînent un glissement de la date de début au plus tôt ou de fin au plus tard, calculer (toutes) les actualisations correspondantes, et retenir la plus forte.

On devine, sur la figure IV.16, que ce problème d'actualisation insuffisante provient de différences de pentes entre les courbes W_p^{PL} et $w_p^{PL}(C_i)$: on en conclut qu'il ne peut se poser que pour des problèmes où $q_i < Q$. Pour les problèmes disjonctifs purs, $q_i = Q$ (= 1 pour simplifier) $\forall i$, W_i^{IL} admet pour pentes +1,0,-1,-2,-3,... sans jamais excéder +1 (voir figure IV.17).

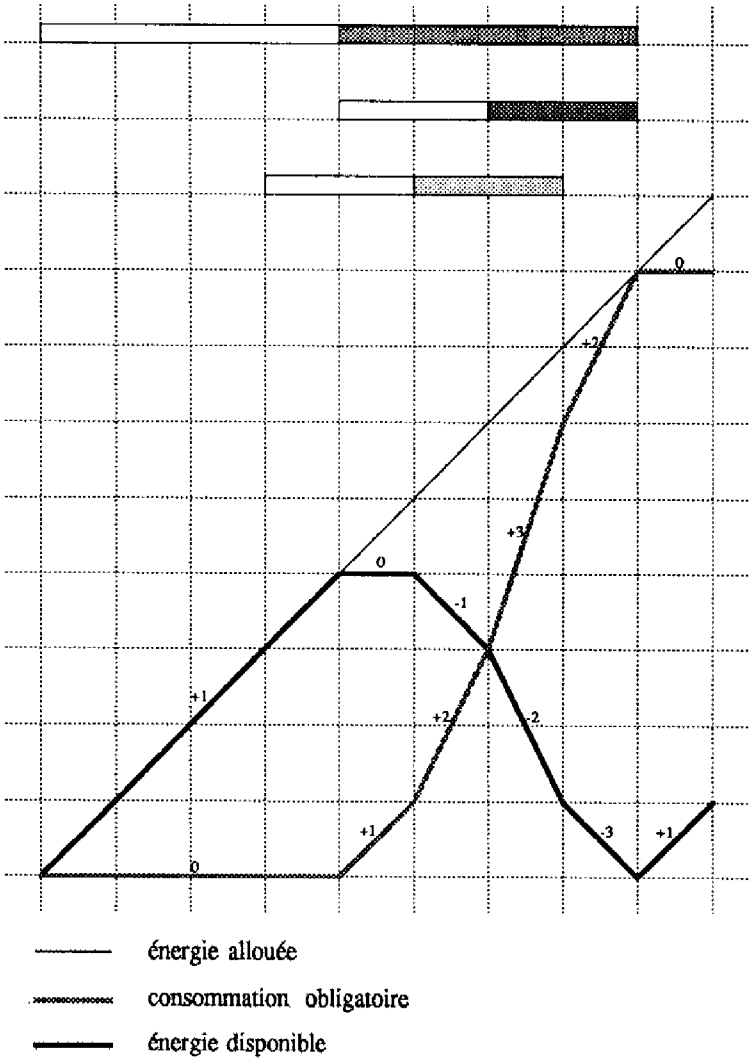


Figure IV.17

En raisonnant sur les pentes, et dans le cadre de la première approche, l'algorithme suivant peut être utilisé :

programme recherche_de_t_m

$L = \{t_n, t_{n-1}, ..., \underline{C_i}\}$
tant que $L \neq \{\}$ **répéter**
prendre le 1^{er} élément de L, soit t_n;
prendre le $2^{ème}$ élément de L, soit t_{n-1};
$L \leftarrow L - \{t_n\}$;
si $\langle t_n \neq \underline{C_i} \rangle$ **et** $\langle W_i^{IN} < w_i^{IN}(\underline{C_i}) \rangle$ **alors**
$t_m \leftarrow t_n$;
tant que $\langle L \neq \{t_n, \underline{C_i}\} \rangle$ **et** $\langle \text{pente}(t_{n-1}, t_n) \leq q_i \rangle$ **répéter**
prendre le 1^{er} élément de L, soit t_n;
prendre le $2^{ème}$ élément de L, soit t_{n-1};
$L \leftarrow L - \{t_{n-1}\}$;
si $\langle \text{pente}(t_{n-1}, t_n) > q_i \rangle$ **alors** $t_m \leftarrow t_{n-1}$;

retourner t_m

Algorithme 3

Le déroulement de cet algorithme, appliqué à l'exemple précédent, est décrit ci-après.

$L = \{14, 10, 9, 7, 6, 4, 3, 0\}$
$\rightarrow t_n = 14$; $t_{n-1} = 10$
$L = \{10, 9, 7, 6, 4, 3, 0\}$
$W_p^{PN} = 14$; $w_i^{PN}(0) = 7$
$\rightarrow t_n = 10$; $t_{n-1} = 9$
$L = \{9, 7, 6, 4, 3, 0\}$
$W_p^{PN} = 6$; $w_i^{PN}(0) = 7$
$t_m = 10$
$pente(9, 10) < q_i$
$t_n = 9$; $t_{n-1} = 7$
$L = \{9, 6, 4, 3, 0\}$
$pente(7, 9) > q_i$; $t_m = 7$
$t_m = 7$.

On trouve $t_m = 7$ qui permet d'actualiser $\underline{C_p'}$ à 5. On vérifie sur la figure IV.16 que c'est le résultat recherché.

Cette approche, comme la deuxième, présente l'inconvénient d'énumérer des valeurs (que ce soit des niveaux d'énergie ou des dates). Nous n'avons pas intégré l'algorithme 3 dans notre module.

Une autre approche, plus générale, semble intéressante à développer. Elle n'est pas basée sur l'amélioration du choix de l'instant remarquable, mais sur

une meilleure définition des bornes de l'intervalle fournisseur sur lequel va se faire l'analyse énergétique d'une tâche. Ceci permet également de remettre en cause l'autre borne retenue précédemment ($\underline{C}_i$ ou $\overline{F}_i$) et conduire ainsi, dans certains cas, à une déduction plus forte.

IV.3.2 Choix des bornes de l'intervalle d'analyse

Les processus décrits jusqu'à présent, sont basés sur la recherche d'un intervalle fournisseur (C_Δ, F_Δ, Q), en assimilant une borne à une extrémité temporelle de la tâche d'étude, l'autre borne variant. Ainsi, lors de l'analyse énergétique au plus tôt (respectivement au plus tard) d'une tâche i, $C_\Delta = \underline{C}_i$ (resp. $F_\Delta = \overline{F}_i$) et F_Δ (resp. C_Δ) est comprise entre $\underline{C}_i$ et $\overline{F}_i$ (resp. entre $\overline{F}_i$ et $\underline{C}_i$).

On montre qu'il existe des intervalles fournisseurs autres que ceux-ci, qui entraînent des actualisations plus fortes.

Reprenons à ce titre, l'exemple 1 (paragraphe IV.2.7.1) ; on rappelle que les résultats en fin d'analyse sont donnés par le tableau IV.7.

i	a	b	e	g	h
$\underline{C}_i$	2	7	1	6	8
$\overline{F}_i$	8	14	9	13	14
D_i	2	1	5	2	2

Tableau IV.7

Le diagramme de Gantt qui illustre ces résultats est le suivant :

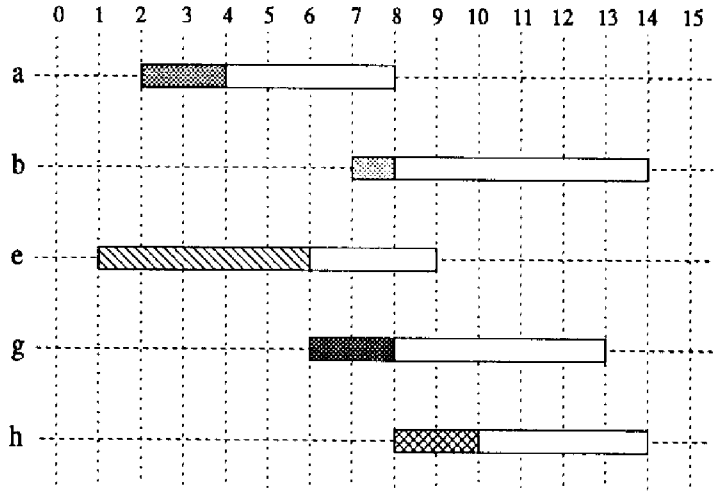


Figure IV.18

Comme il s'agit d'un problème disjonctif, il apparaît que l'on doit avoir nécessairement $C_b \geq 8$ et $C_g \geq 8$ ($\underline{C}'_b = 8$ et $\underline{C}'_g = 8$). Notons au passage que l'algorithme

3, décrit au paragraphe IV.3.1, ne peut pas nous aider à résoudre ce problème, l'insuffisance n'apparaissant pas sur les courbes d'énergie (problème disjonctif pur).

Notre problème est donc le suivant : étant donné un ensemble de tâches utilisant une ressource, on veut élaborer une stratégie déterminant les intervalles d'étude $[C_\Delta, F_\Delta]$ qui permettent d'inférer les meilleures actualisations de la date de début au plus tôt (fin au plus tard) d'une tâche i.

Comme auparavant, on ne considère que l'analyse énergétique au plus tôt. Pour l'étude d'une tâche $i = (C_i, F_i, q_i)$ (i appartient à un ensemble E), l'intervalle $\Delta = (C_\Delta, F_\Delta, Q)$ sera tel que l'on a la configuration représentée par la figure IV.19

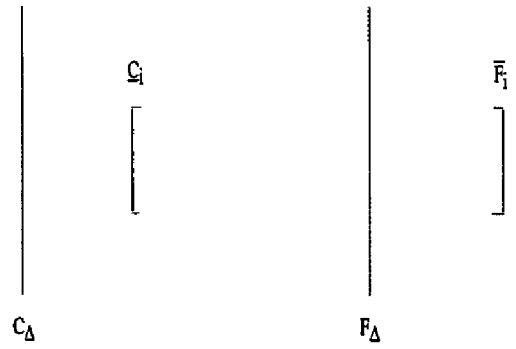


Figure IV.19

où les conditions auxquelles sont soumises les bornes de Δ sont :

$$\begin{cases} \min\limits_{j \in E} \underline{C_j} \leq C_\Delta < \underline{C_i} + D_i & (C_\Delta > \underline{C_i} + D_i \Longrightarrow w_i^\Delta(\underline{C_i}) = \underline{w_i^\Delta} = 0) \\ \underline{C_i} < F_\Delta < \overline{F_i} \\ C_\Delta < F_\Delta \end{cases}$$

La règle ($\Re 4$) peut ainsi être généralisée à :

$$\begin{cases} \text{si} & W_i^\Delta < \underline{w_i^\Delta} \\ \text{alors} & F_i \geq max(F_\Delta, \underline{C_i} + D_i) + \dfrac{S_\Delta}{q_i} \end{cases}$$

où

$$S_\Delta = \underline{w_i^\Delta} - W_i^\Delta > 0$$

et

$$\begin{aligned} \underline{w_i^\Delta} &= min(\underline{C_i} + D_i - C_\Delta, F_\Delta - \underline{C_i}, F_\Delta - C_\Delta, D_i).q_i \\ &= [min(\underline{C_i} + D_i, F_\Delta) - max(\underline{C_i}, C_\Delta)].q_i. \end{aligned}$$

D'où :

$$\underline{C_i'} = max(F_\Delta, \underline{C_i}+D_i)+min(\underline{C_i}+D_i, F_\Delta)-max(\underline{C_i}, C_\Delta)-D_i+\frac{\sum_{j \neq i} w_j^\Delta - (F_\Delta - C_\Delta).Q}{q_i}$$

$$\Longrightarrow \underline{C}'_i = C_\Delta.\alpha_i + F_\Delta.(1-\alpha_i) + \sum_{j\neq i} \mathcal{T}_j^\Delta.\beta_i^j + min(0, \underline{C}_i - C_\Delta).$$

Cas particulier : problème disjonctif pur.

$$\underline{C}'_i = C_\Delta + \sum_{j\neq i} \mathcal{T}_j^\Delta + min(0, \underline{C}_i - C_\Delta).$$

Trouver la plus forte actualisation revient à maximiser $\underline{C}'_i$, ou, plus simplement, à maximiser la quantité :

$$\Theta = max(F_\Delta, \underline{C}_i + D_i) + \frac{S_\Delta}{q_i} \quad \text{avec } S_\Delta > 0.$$

Exemple

Utilisons ce résultat sur l'exemple 1 (tableau IV.4) ; on s'intéresse à l'analyse énergétique au plus tôt associée à g. Sur l'intervalle [1,13], on trouve les instants remarquables[6] $1,2,3,4,6,11$ pour C_Δ et $13,12,9,8,6,4,3$ pour F_Δ (tous les intervalles possibles ne sont pas représentés). On obtient le tableau IV.19.

C_Δ	F_Δ	w_a^Δ	w_b^Δ	w_e^Δ	w_h^Δ	$\sum_{j\neq g} w_j^\Delta$	W^Δ	W_g^Δ	$\underline{w}_g^\Delta$	S_Δ	Θ
1	13	2	0	5	1	8	12	4	2	-2	$(S_\Delta < 0)$
1	12	2	0	5	0	7	11	4	2	-2	$(S_\Delta < 0)$
1	9	2	0	5	0	7	8	1	2	+1	10
2	9	2	0	4	0	6	7	1	2	+1	10
2	12	2	0	4	0	6	10	4	2	-2	$(S_\Delta < 0)$
2	8	2	0	4	0	6	6	0	2	+2	10
3	8	1	0	3	0	4	5	1	2	+1	9
4	8	0	0	2	0	2	4	2	2	0	$(S_\Delta = 0)$
6	8	0	0	0	0	0	2	2	1	-1	$(S_\Delta < 0)$

Tableau IV.19

Une analyse sur l'intervalle $[C_\Delta, F_\Delta] = [2,9]$ par exemple, permet de déduire l'actualisation $\underline{C}'_g = 8$, qui était la valeur attendue. De même, une analyse énergétique au plus tôt associée à la tâche b, sur un intervalle $[C_\Delta, F_\Delta] = [2,13]$, amène à actualiser $\underline{C}_b$ à 8.

IV.3.3 Bilan

Cette dernière approche semble intéressante car elle vise à généraliser l'analyse énergétique associée à une tâche, non nécessairement limitée à la fenêtre temporelle qui lui est allouée. Par ailleurs, un choix judicieux de l'intervalle d'analyse peut s'appuyer sur une procédure améliorée de recherche de l'instant remarquable pour l'actualisation de dates limites, comme celle proposée en IV.3.1.

[6] Les instants remarquables n'appartenant pas à l'intervalle [5,13] sont obtenus à partir des relations du paragraphe IV.2.5.1 en remplaçant $\underline{C}_i$ par $C_\Delta = 1$.

IV.4 Conclusion

Les résultats présentés ci-dessus utilisent le concept d'énergie pour analyser les interactions entre contraintes de temps et de ressources. Une procédure d'analyse énergétique (au plus tôt et au plus tard) a ainsi été développée ; elle permet d'actualiser directement les valeurs associées aux bornes d'intervalles. Ces actualisations correspondent à des caractéristiques nécessaires des ordonnancements admissibles. La qualité de ces caractéristiques est étroitement liée à la définition des bornes des intervalles fournisseurs sur lesquels s'opère l'analyse énergétique (instants remarquables). Cette approche est, pour l'instant, essentiellement centrée sur l'utilisation d'intervalles fournisseurs calés à gauche ($C_\Delta = \underline{C_i}$) ou à droite ($F_\Delta = \overline{F_i}$), dans la fenêtre allouée à la tâche considérée. Certaines extensions sont toutefois proposées en vue de généraliser l'analyse énergétique à tout intervalle fournisseur, ce qui permet d'affiner encore, dans certains cas, la caractérisation des ordonnancements admissibles.

Au chapitre III, on a décrit des règles permettant de déduire des relations séquentielles entre tâches. L'apport du concept d'énergie amène à comparer les règles qui manipulent des intervalles de temps simples ou des caractéristiques de ressource, avec des règles mettant en jeu des bilans énergétiques.

Dans le chapitre IV, les règles présentées aboutissent à des déductions numériques et ne peuvent ainsi être sujettes à comparaison avec les règles précédentes. La comparaison peut cependant être établie lors du traitement d'exemples en analysant les déductions obtenues de part et d'autre : cette étude est réalisée dans le chapitre V.

* *

*

Chapitre V

Modules de raisonnement temporel sous contraintes de ressources

V.1 Introduction

V.1.1 Préliminaire

La formalisation des travaux théoriques sur l'analyse sous contraintes (A.S.C.) [Erschler 76] a donné lieu à la réalisation de la première version d'un Module d'Analyse Sous Contraintes pour l'Ordonnancement de Tâches (**MASCOT1** [Erschler et Esquirol 86]) en Prolog-II [Giannesini et al. 85]. Les principes généraux de ce module, qui sont repris dans nos développements, sont rappelés ici ; une présentation détaillée peut être trouvée dans [Esquirol 87].

Les conclusions du chapitre III ont permis de développer une nouvelle version de ce module (**MASCOT2**) utilisant le concept d'énergie pour la résolution des conflits entre tâches (règle ($\Re 3$)). Une analyse comparative entre MASCOT1 et MASCOT2 est proposée.

Les résultats présentés au chapitre IV, conduisent à une remise en cause complète du processus d'analyse, qui n'est plus basé sur le séquencement de tâches en conflit, mais sur le lissage de l'énergie consommée par modification des fenêtres temporelles associées aux tâches. Un nouveau module d'analyse a ainsi été développé, toujours en Prolog-II, qui utilise un Raisonnement Energétique Pour l'ORdonnancement de Tâches (logiciel **REPORT**), basé sur l'application des règles ($\Re 4$) et ($\Re' 4$). Une comparaison des résultats obtenus à l'aide de REPORT et MASCOT2 est proposée.

Ces deux modules, différents par leur principe d'analyse, peuvent être intel-

ligemment combinés selon une stratégie d'utilisation visant à limiter l'exploration de solutions. L'application successive de MASCOT2 et REPORT sur un exemple général permet d'illustrer une stratégie possible pour l'A.S.C. d'un problème d'ordonnancement.

V.1.2 Présentation générale

Les deux modules sont constitués de :

1. une base de faits (BdF) incluant :
   - des faits invariants au cours du temps : les caractéristiques statiques des intervalles (utilisation ou disponibilité des ressources, durées, ...),
   - des faits susceptibles d'évoluer au cours du raisonnement, qui caractérisent les ordonnancements admissibles :
     - les valeurs extrêmes, associées aux bornes d'intervalles,
     - les contraintes temporelles entre intervalles ;
2. une base de règles (BdR) permettant d'ajouter ou d'actualiser les caractéristiques des ordonnancements admissibles. Lorsque les règles s'appliquent, ces nouvelles caractéristiques sont mémorisées dans la BdF. Compte tenu de l'interdépendance des caractéristiques, une règle, qui a échoué dans un contexte donné, peut éventuellement se déclencher lorsque le contexte évolue.

- La BdR de MASCOT2 comprend :
  - un processus d'inférence par propagation des valeurs extrêmes $\underline{B_i}$ et $\overline{B_i}$ sur le graphe potentiels-bornes ;
  - un bloc de déduction symbolique qui utilise les règles $(\Re 1)$, $(\Re 2)$ et $(\Re 3)$ pour générer de nouvelles contraintes temporelles entre intervalles ;
  - un bloc de contrôle qui guide dans leur travail le bloc de déduction et le processus de propagation ; il commande l'A.S.C. du problème.
- La BdR de REPORT, de composition semblable, est formée par :
  - le processus d'inférence par propagation ;
  - un bloc de déduction numérique qui utilise les règles $(\Re 4)$ et $(\Re' 4)$ pour actualiser les valeurs extrêmes des bornes ;
  - un bloc de contrôle.

V.2 Formalisation du problème

Nous rappelons ici les caractéristiques et les contraintes du problème, qui permettent de décrire le modèle central retenu dans notre approche. Celles-ci sont représentées, de manière très lisible et très naturelle, grâce au formalisme de Prolog-II, afin de pouvoir suivre, plus loin, le déroulement des exemples. Cette présentation est commune aux logiciels MASCOT et REPORT.

V.2.1 Caractéristiques des tâches et des ressources

- Nous disposons d'un ensemble de ressources K pour la réalisation d'un ensemble de tâches E.
- Chaque ressource est caractérisée par une intensité maximale Q^k constante.
- Chaque tâche i utilise une ou plusieurs ressources de K avec une certaine intensité q_i^k.
- Une durée est associée à la réalisation de chaque tâche pendant laquelle la ou les ressources requises par la tâche sont simultanément utilisées.

Ces données sont écrites en Prolog et intégrées dans la base de faits du module.

Exemple : Soient quatre tâches m, n, o, p et trois ressources séparées en : (1) deux *machines* "mach1" et "mach2" disponibles en exemplaire unique et (2) une ressource "Pers" de type *personnel* dont on possède deux exemplaires. m et n utilisent respectivement "mach1" et "mach2", o utilise "mach1" et un exemplaire de "Pers", p utilise "mach1" et deux exemplaires de "Pers". Les faits correspondants s'écrivent en Prolog-II :

```
DISPONIBILITE("Pers",2) ->;
DISPONIBILITE("mach2",1) ->;
DISPONIBILITE("mach1",1) ->;

UTILISE("m",<"mach1",1>.nil) ->;
UTILISE("n",<"mach2",1>.nil) ->;
UTILISE("o",<"mach1",1>.<"Pers",1>.nil) ->;
UTILISE("p",<"mach2",1>.<"Pers",2>.nil) ->;

DURE("m",7) ->;
DURE("n",4) ->;
DURE("o",5) ->;
DURE("p",5) ->;
```

V.2.2 Contraintes diverses

V.2.2.1 Fenêtre temporelle

Elle précise la marge temporelle disponible pour commencer (ou finir) une tâche. Elle est limitée par une date de début au plus tôt et une date de fin au plus tard : $\underline{C_i} \leq C_i \leq \overline{F_i} - D_i$. Certaines tâches peuvent n'être soumises qu'à une ou même à aucune de ces contraintes.

V.2.2.2 Caractéristiques séquentielles

On se limite à la représentation de contraintes de précédence simples : $(C_j - C_i \geq D_i)$. Toutefois, on peut représenter des contraintes du type : $C_j - C_i \geq a_{ij}$ avec $a_{ij} \neq D_i$, en utilisant une tâche fictive f de durée $D_f = a_{ij} - D_i$ et en vérifiant les relations : (i précède f) et (f précède j). On a alors :

$$\left.\begin{array}{l} C_j - C_f \geq D_f \\ C_f - C_i \geq D_i \\ D_f = a_{ij} - D_i \end{array}\right\} \Longrightarrow C_j - C_i \geq D_i + D_f = a_{ij}.$$

On peut représenter un choix parmi un ensemble de précédences simples entre une tâche et un groupe de tâches (figure V.1) par une relation de type "faisceau non conjonctif".

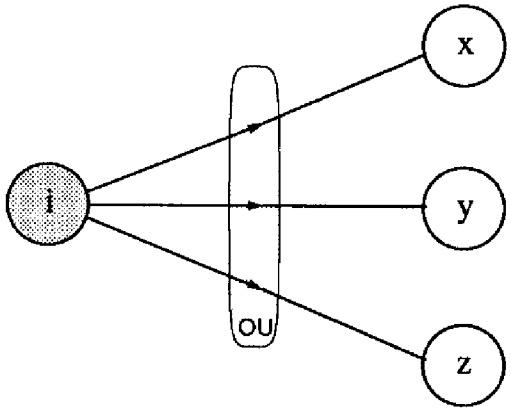


Figure V.1 : Faisceau non conjonctif

Le graphe potentiels-tâches non conjonctif de la figure V.1 se formalise en : (i précède $\langle x$ *ou* y *ou* $z\rangle$) .

D'un autre côté, une conjonction d'inégalités (figure V.2)

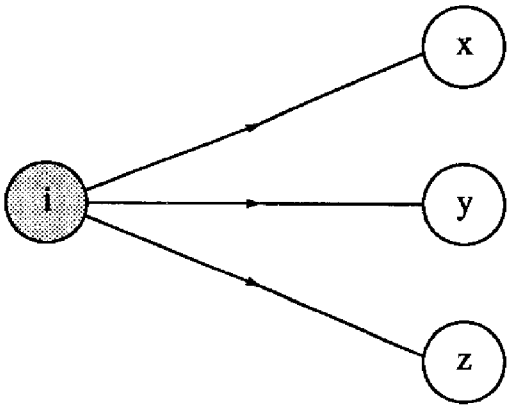


Figure V.2 : Faisceau conjonctif

se formalise en : (i précède x), (i précède y), (i précède z).

La réalisation d'une séquence de tâches constitue un travail.

V.2.2.3 Limitation d'intensité de ressource

Pour un ensemble de tâches $E' \subset E$ utilisant une ressource k disponible en quantité Q^k, on doit avoir à chaque instant :

$$\sum_{i \in E'} q_i^k \leq Q^k.$$

Contrairement aux deux premiers types de contraintes, celles sur la limitation d'intensité de ressource n'apparaissent pas dans la BdF du module, mais sont prises en compte dans le processus d'analyse, sous la forme de règles.

V.2.2.4 Exemple

On reprend l'exemple précédent et l'on pose : $\underline{C}_m = \underline{C}_p = 0$; $\overline{F}_n = \overline{F}_p = 14$; $\overline{F}_o = 15$. Le travail "T1" est formé par la succession des tâches m et n ; on a de plus la relation ($\langle n$ *ou* $p \rangle$) précède o).[1] Ces contraintes sont écrites en Prolog et incluses dans la BdF du module.

```
DEBUT-PLUS-TOT("m",0) ->;
DEBUT-PLUS-TOT("p",0) ->;

FIN-PLUS-TARD("n",14) ->;
```

[1] Remarquons qu'en pratique, une contrainte de type faisceau non conjonctif n'est jamais introduite dans la BdF initiale d'un problème ; elle apparaît en cours d'analyse où elle est ajoutée à la BdF.

```
FIN-PLUS-TARD("o",15) ->;
FIN-PLUS-TARD("p",14) ->;

PRECEDE("m".nil,"n".nil) ->;
PRECEDE("n"."p".nil,"o".nil) ->;
```

V.3 Contrôle de l'analyse

V.3.1 Stratégie générale

La stratégie générale des modules a pour objectif d'appliquer l'A.S.C., jusqu'à ce qu'on ne puisse plus rien déduire ou qu'une incohérence soit détectée. Dans le premier cas, la base de faits contient des informations sur les solutions admissibles (conditions nécessaires d'admissibilité) sous la forme de marges temporelles et de contraintes temporelles (conjonctives et non conjonctives) entre tâches. Dans le second cas, un processus de relaxation de contraintes est nécessaire. Une incohérence apparaît lorsque la fenêtre temporelle allouée pour la réalisation d'une tâche est plus petite que sa durée. Le déroulement du programme est alors arrêté et un message indique la tâche pour laquelle l'incohérence a été décelée. Ceci peut se produire dès la propagation des contraintes de la BdF initiale ou en cours d'analyse après actualisation(s).

V.3.2 Procédures de contrôle de l'analyse

V.3.2.1 Contrôle dans MASCOT

On a vu que les faits capables de changer portent sur les dates limites de tâches ou les contraintes de précédence entre tâches. La recherche des conséquences séquentielles d'une actualisation de date limite est beaucoup plus combinatoire que la recherche des conséquences temporelles numériques de l'ajout d'une relation de précédence. En conséquence, le contrôle va privilégier l'actualisation de dates limites par rapport à la recherche de conditions séquentielles entre tâches. Par ailleurs, la stratégie de contrôle s'appuie sur la valeur du cardinal des ensembles critiques considérés [Erschler et Esquirol 86][Esquirol 87]. Ainsi, les ensembles critiques sont traités dans l'ordre croissant de leur cardinal. L'analyse commence donc par les ensembles d'ordre 2 (cas disjonctif) et dès que le processus de propagation a été activé, on revient aux ensembles d'ordre 2. Ceci peut se justifier par le fait, d'une part, que les déductions obtenues sont d'autant plus fortes qu'elles sont issues d'ensembles critiques petits, et, d'autre part, qu'il est possible de définir des règles spécifiques et plus efficaces dans le cas disjonctif ($|E_c^\alpha| = 2$). L'algorithme correspondant est décrit ci-après (algorithme 4).[2]

[2]Etant donné le caractère *impératif* des algorithmes associés à une stratégie de contrôle (par opposition au caractère *déclaratif* des BdF ou des BdR), nous avons préféré les présenter sous la

programme contrôle_analyse_MASCOT

variables COHÉRENCE : BOOLEEN,
$|E_c^\alpha|$, n : ENTIERS;
INITIALISATION {recherche des ensembles critiques,
détermination du plus grand ordre n
des ensembles critiques,
déclarer possible l'activation
du processus de propagation,
initialiser COHÉRENCE=VRAI};
$|E_c^\alpha| \leftarrow 2$;

tant que COHÉRENCE=VRAI et $|E_c^\alpha| \leq n$ **répéter**
répéter
si propagation possible **alors** {étant donné les contraintes
temporelles actuelles}
activer le processus de propagation;
si COHÉRENCE=VRAI **alors** $|E_c^\alpha| \leftarrow 2$;
activer le bloc de déduction symbolique
jusqu'à "impossibilité de déduire une nouvelle contrainte temporelle";
$|E_c^\alpha| \leftarrow |E_c^\alpha| + 1$;
fin tantque

si COHÉRENCE=VRAI **alors**
retourner "Fin Analyse"
sinon
retourner "Incohérence détectée"

fin programme

Algorithme 4

V.3.2.2 Contrôle dans REPORT

Dans REPORT, les seuls faits susceptibles d'évoluer sont les valeurs extrêmes associées aux bornes d'intervalles. Dès qu'une actualisation de date est trouvée, l'analyse énergétique est arrêtée et le processus de propagation est déclenché. A la fin de la propagation, toutes les tâches sont de nouveau considérées. Si le problème est cohérent, la procédure est réitérée jusqu'à ce que plus aucune actualisation ne soit trouvée. Le bloc de déduction numérique envisage tout d'abord d'actualiser la date de début au plus tôt d'une tâche ; s'il n'y parvient pas, il tente l'actualisation de sa date de fin au plus tard. L'algorithme 5 présente, de manière générale, le contrôle dans REPORT.

forme classique d'un programme en pseudo-langage (inspiré de PASCAL).

```
programme contrôle_analyse_REPORT

    variables COHÉRENCE,ACTUALISATION_POSSIBLE : BOOLEENS;

    initialiser COHÉRENCE=VRAI;
    activer le processus de propagation;

    répéter
        initialiser ACTUALISATION_POSSIBLE=FAUX;
        R = ensemble des ressources;

        tant que R ≠ {} et ACTUALISATION_POSSIBLE=FAUX répéter
            r = 1ère ressource de R;
            T = ensemble des tâches utilisant r;

            tant que T ≠ {} et ACTUALISATION_POSSIBLE=FAUX répéter
                t = 1ère tache de T;
                activer le bloc de déduction numérique;
                si une actualisation est possible alors
                    ACTUALISATION_POSSIBLE=VRAI;
                    initialiser la propagation;
                sinon T ← T − t;
            fin tantque

            si ACTUALISATION_POSSIBLE=FAUX alors R ← R − r;
        fin tantque

        si ACTUALISATION_POSSIBLE=VRAI alors
            activer le processus de propagation;
    jusqu'à ACTUALISATION_POSSIBLE=FAUX ou COHÉRENCE=FAUX

    si COHÉRENCE=VRAI alors
        retourner "Fin Analyse"
    sinon
        retourner "Incohérence détectée"

fin programme
```

Algorithme 5

V.4 Processus de propagation

Le processus de propagation présenté est celui qui est utilisé dans MASCOT et qui est repris dans REPORT, sans modification. La propagation des dates limites s'appuie sur un graphe de précédence (chaque arc représente une contrainte de précédence simple entre deux tâches — éventuellement fictives). Il s'agit donc

d'une modélisation simplifiée vis-à-vis de la représentation par graphe potentiels-bornes. Celle-ci n'est toutefois pas sans intérêt pour permettre de traiter un éventail plus large de problèmes (cf. II.2.5). Un nouveau type de propagation, basé sur le graphe potentiels-bornes, reste donc à être intégré dans les modules.

Le but de ce processus est de parvenir à un jeu de dates tel qu'aucune règle d'actualisation ne puisse être activée. Le processus de propagation s'appuie sur les concepts de tâches "source" et "puits".

V.4.1 Définitions

Lors de la constitution de la BdF initiale, i est déclarée tâche *source* (respectivement *puits*) si $\nexists j$ telle que j précède i (resp. i précède j). En cours d'analyse, i devient une tâche source (resp. puits) si $\underline{C_i}$ (resp. $\overline{F_i}$) est actualisée ; elle cesse de l'être lorsque $\forall j \in S(i)$=ensemble des tâches suivantes[3] de i (resp. $P(i)$=ensemble des tâches précédentes[3] de i), l'actualisation de $\underline{C_j}$ (resp. $\overline{F_j}$) a été tentée.

V.4.2 Stratégie de propagation

D'une part il existe plusieurs règles d'actualisation, d'autre part une tâche peut être la suivante (respectivement la précédente) de plusieurs tâches sources (resp. puits) : ces deux raisons impliquent qu'une même tâche peut être actualisée plusieurs fois. Comme rien ne permet de présumer du résultat des règles avant leur application, il est nécessaire d'explorer toutes les possibilités pour s'assurer que l'actualisation a été réalisée [Esquirol 87].[4] La stratégie retenue, est une stratégie de recherche "en largeur d'abord" (par opposition à une recherche "en profondeur d'abord") qui consiste à étudier toutes les actualisations possibles compte tenu d'un ensemble de tâches sources. Ces deux stratégies sont illustrées sur la figure V.3 : dans cet exemple précis, la stratégie en largeur d'abord s'avère la plus performante.

- Une stratégie en profondeur d'abord consiste à emprunter les arcs dans l'ordre : $1,5,6,2,7,8,3,5,6,4,7,8$, soit 12 activations possibles des règles d'actualisation ;

- par la stratégie en largeur d'abord, les arcs sont parcourus dans l'ordre : $1,2,3,4,5,6,7,8$, soit 8 activations possibles des règles.

[3] tâches **suivantes** au sens strict ou au sens conditionnel, i.e. reliées par un faisceau non conjonctif

[4] L'auteur s'est inspiré des stratégies de déduction par *saturation*, caractéristique de certains systèmes à BdR fonctionnant en chaînage avant [Giannesini et al. 85].

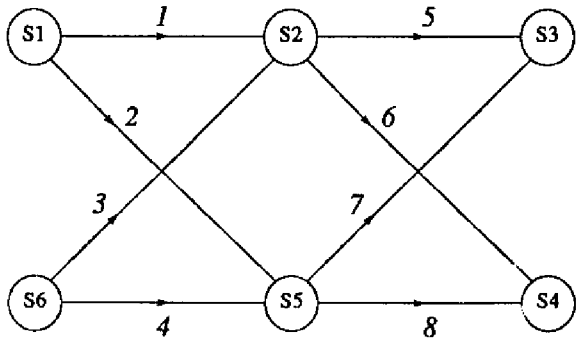


Figure V.3

V.5 Etude comparative des différents modules d'analyse

Ce paragraphe propose des comparaisons entre MASCOT1 et MASCOT2 d'une part et entre MASCOT2 et REPORT d'autre part.

V.5.1 Comparaisons MASCOT1 – MASCOT2

On a montré au chapitre III, l'intérêt propre de chacune des règles ($\Re 2$) et ($\Re 3$). Vis-à-vis du pouvoir de déduction (complétude), MASCOT2, qui intègre ces deux types de règles et la règle ($\Re 1$) relative aux ensembles critiques, est donc forcément meilleur que MASCOT1, qui ne combine que les règles ($\Re 2$) et ($\Re 1$). Cette supériorité est illustrée sur l'exemple simple suivant, voisin de celui proposé au paragraphe III.5.2, et défini par la BdF suivante :

```
"Disponibilite des ressources"
DISPONIBILITE("M1",1) ->;

"Quantites de ressources utilisees"
UTILISE("1",<"M1",1>.nil) ->;
UTILISE("2",<"M1",1>.nil) ->;
UTILISE("3",<"M1",1>.nil) ->;
UTILISE("4",<"M1",1>.nil) ->;
UTILISE("5",<"M1",1>.nil) ->;

"Durees des taches"
DURE("1",5) ->;
DURE("2",3) ->;
DURE("3",3) ->;
DURE("4",4) ->;
```

```
DURE("5",1) ->;

"Bornes temporelles"

DEBUT-PLUS-TOT("1",0) ->;
DEBUT-PLUS-TOT("2",5) ->;
DEBUT-PLUS-TOT("3",0) ->;
DEBUT-PLUS-TOT("4",5) ->;
DEBUT-PLUS-TOT("5",11) ->;

FIN-PLUS-TARD("1",8) ->;
FIN-PLUS-TARD("2",12) ->;
FIN-PLUS-TARD("3",12) ->;
FIN-PLUS-TARD("4",16) ->;
FIN-PLUS-TARD("5",16) ->;

;End world:  Faits
```

Les résultats fournis par l'analyse de MASCOT2 sont décrits ci-dessous.

```
Recherche des ensembles critiques
        ENS-CRITIQUE("M1","1"."2".nil)
        ENS-CRITIQUE("M1","1"."3".nil)
        ENS-CRITIQUE("M1","1"."4".nil)
        ENS-CRITIQUE("M1","1"."5".nil)
        ENS-CRITIQUE("M1","2"."3".nil)
        ENS-CRITIQUE("M1","2"."4".nil)
        ENS-CRITIQUE("M1","2"."5".nil)
        ENS-CRITIQUE("M1","3"."4".nil)
        ENS-CRITIQUE("M1","3"."5".nil)
        ENS-CRITIQUE("M1","4"."5".nil)

-------------------------------------------------------------------------------
 T | Duree | Dates limites [ ]      Diagramme de GANTT : RESSOURCE "M1"
-------------------------------------------------------------------------------
          +0.0 [||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||]+16.0

"1"  +5.0  +0.0 [*****************          ]+8.0
"2"  +3.0                    +5.0 [*********              ]+12.0
"3"  +3.0  +0.0 [*********                                ]+12.0
"4"  +4.0                    +5.0 [**************                        ]+16.0
"5"  +1.0                                      +11.0[***                 ]+16.0

Combinaison des regles (R1) d'une part et (R2) et (R3) d'autre part
-------------------------------------------------------------------

Recherche de contraintes temporelles, dans le cas disjonctif :
        PRECEDE("3".nil,"5".nil)
        PRECEDE("3".nil,"4".nil)
Actualisation de bornes temporelles :
        FIN-PLUS-TARD("3",11)
```

```
Recherche de contraintes temporelles, dans le cas disjonctif :
        PRECEDE("2".nil,"5".nil)
        PRECEDE("1".nil,"5".nil)
        PRECEDE("1".nil,"4".nil)
Actualisation de bornes temporelles :
        DEBUT-PLUS-TOT("4",8)
Recherche de contraintes temporelles, dans le cas disjonctif :
        PRECEDE("2".nil,"4".nil)
Actualisation de bornes temporelles :
        DEBUT-PLUS-TOT("4",11)
        FIN-PLUS-TARD("2",11)
Recherche de contraintes temporelles, dans le cas disjonctif :
        PRECEDE("1".nil,"2".nil)
Recherche de contraintes temporelles, dans le cas disjonctif :
                                      dans le cas cumulatif :

                                Fin Analyse
                                ^^^^^^^^^^^

-------------------------------------------------------------------------------
 T | Duree | Dates limites [ ]      Diagramme de GANTT : RESSOURCE "M1"
-------------------------------------------------------------------------------
           +0.0 [||||||||||||||||||||||||||||||||||||||||||||||||||||||||||]+16.0

"1"  +5.0  +0.0 [*****************          ]+8.0
"2"  +3.0                    +5.0 [*********          ]+11.0
"3"  +3.0  +0.0 [*********                            ]+11.0
"4"  +4.0                                        +11.0[**************    ]+16.0
"5"  +1.0                                        +11.0[***               ]+16.0
```

Lorsque ce problème est soumis à l'analyse de MASCOT1, certaines déductions symboliques sont trouvées, sans pour autant provoquer d'actualisations. L'analyse de MASCOT2 permet de déduire trois actualisations de bornes temporelles $(\overline{F}_3, \underline{C}_4, \overline{F}_2)$. Celles-ci sont rendues possibles par les relations *(2 précède 4)* et *(3 précède 4)* que ne déduit pas MASCOT1. On peut remarquer, pour cet exemple, que parmi les déductions séquentielles de MASCOT1, on trouve, entre autres choses :

(3 précède ⟨2 ou 4⟩), (⟨2 ou 3⟩ précède 4), (2 précède ⟨3 ou 4⟩).

Par traitement logique

- des deux premières, on déduit *(3 précède 4)* ;
- des deux dernières, – — *(2 précède 4)*.

Bien que certaines règles de traitement logique aient été intégrées dans MASCOT1, la trop grande particularité de ce cas de figure n'a pas donné lieu à l'écriture d'une règle spécifique, dont les conditions d'application exigent une analyse jugée trop coûteuse en raison de son caractère combinatoire : il faut, en effet, rechercher

une configuration faisant intervenir trois tâches reliées par des relations d'ordre 2, formant deux faisceaux, l'un convergent, l'autre divergent ⟨cf. figure V.4⟩). MASCOT2, lui non plus, ne prend pas en compte cette règle, mais parvient pourtant directement, grâce à la règle énergétique (ℜ3), au résultat escompté.

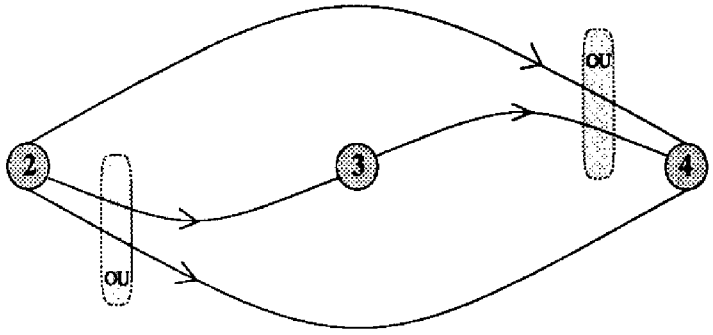


Figure V.4 : Modélisation de *(2 précède ⟨3 ou 4⟩)* et *(⟨2 ou 3⟩ précède 4)*

V.5.2 Comparaisons MASCOT2 – REPORT

On s'intéresse ici aux relations entre MASCOT2 et REPORT.

On considère un problème cumulatif de six tâches utilisant une unité d'une ressource dont on possède deux exemplaires. La BdF s'écrit en Prolog-II :

```
"Disponibilite des ressources"
DISPONIBILITE("R1",2)->;

"Quantites de ressources utilisees"
UTILISE("1",<"R1",1>.nil)->;
UTILISE("2",<"R1",1>.nil)->;
UTILISE("3",<"R1",1>.nil)->;
UTILISE("4",<"R1",1>.nil)->;
UTILISE("5",<"R1",1>.nil)->;
UTILISE("6",<"R1",1>.nil)->;

"Durees des taches"
DURE("1",6)->;
DURE("2",6)->;
DURE("3",4)->;
DURE("4",5)->;
DURE("5",3)->;
DURE("6",6)->;

"Bornes temporelles"
```

```
DEBUT-PLUS-TOT("1",2)->;
DEBUT-PLUS-TOT("2",3)->;
DEBUT-PLUS-TOT("3",5)->;
DEBUT-PLUS-TOT("4",1)->;
DEBUT-PLUS-TOT("5",3)->;
DEBUT-PLUS-TOT("6",0)->;

FIN-PLUS-TARD("1",15)->;
FIN-PLUS-TARD("2",16)->;
FIN-PLUS-TARD("3",15)->;
FIN-PLUS-TARD("4",14)->;
FIN-PLUS-TARD("5",15)->;
FIN-PLUS-TARD("6",15)->;

;End world:  Faits
```

L'analyse de REPORT est retranscrite ci-après.

```
-------------------------------------------------------------------------------
 T | Duree | Dates limites [ ]      Diagramme de GANTT : RESSOURCE "R1"
-------------------------------------------------------------------------------
          +0.0 [||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||]+16.0

"1"  +6.0         +2.0 [*******************                              ]+15.0
"2"  +6.0            +3.0 [*********************                          ]+16.0
"3"  +4.0                    +5.0 [*************                          ]+15.0
"4"  +5.0     +1.0 [*****************                                   ]+14.0
"5"  +3.0            +3.0 [*********                                      ]+15.0
"6"  +6.0  +0.0 [*******************                                     ]+15.0

Utilisation des regles (R4) et (R'4)
------------------------------------
Actualisation de bornes temporelles
        FIN-PLUS-TARD("6",6)   (Intervalle d'etude : [1,15])

                      Fin Analyse
                      -----------


-------------------------------------------------------------------------------
 T | Duree | Dates limites [ ]      Diagramme de GANTT : RESSOURCE "R1"
-------------------------------------------------------------------------------
          +0.0 [||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||]+16.0

"1"  +6.0         +2.0 [*******************                              ]+15.0
"2"  +6.0            +3.0 [*********************                          ]+16.0
"3"  +4.0                    +5.0 [*************                          ]+15.0
"4"  +5.0     +1.0 [*****************                                   ]+14.0
"5"  +3.0            +3.0 [*********                                      ]+15.0
"6"  +6.0  +0.0 [*******************]+6.0
```

Une seule actualisation est déduite de l'analyse : $\overline{F}_6$ devient 6. Il s'agit néanmoins d'un élément intéressant pour une comparaison avec les résultats obtenus par MASCOT2. Ce dernier en effet, déduit des déductions symboliques (du type "précédence interdite"), qui n'amènent aucune actualisation sur cet exemple. Une partie de son analyse est reproduite ci-dessous.

```
Recherche de contraintes temporelles, dans le cas disjonctif :
                                      dans le cas cumulatif :
        PRECEDE-INTERDIT("5","4")
        PRECEDE-INTERDIT("5","6")
        PRECEDE-INTERDIT("4","6")
        PRECEDE-INTERDIT("2","4")
        PRECEDE-INTERDIT("2","6")
        PRECEDE-INTERDIT("1","6")
        PRECEDE-INTERDIT("2","1")
```

Considérons à présent le même exemple, mais avec désormais $D_5 = 4$. Le traitement de REPORT est le suivant :

```
-------------------------------------------------------------------------------
 T | Duree | Dates limites [ ]      Diagramme de GANTT : RESSOURCE "R1"
-------------------------------------------------------------------------------
          +0.0 [||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||]+16.0

"1"  +6.0          +2.0 [********************                              ]+15.0
"2"  +6.0             +3.0 [********************                           ]+16.0
"3"  +4.0                    +5.0 [*************                           ]+15.0
"4"  +5.0     +1.0 [*****************                                    ]+14.0
"5"  +4.0             +3.0 [*************                                 ]+15.0
"6"  +6.0  +0.0 [********************                                    ]+15.0

Utilisation des regles (R4) et (R'4)
------------------------------------
Actualisation de bornes temporelles
        DEBUT-PLUS-TOT("2",10)   (Intervalle d'etude : [3,15])
Actualisation de bornes temporelles
        DEBUT-PLUS-TOT("1",10)   (Intervalle d'etude : [2,15])

                      FIN ANTICIPEE...
                      contraintes trop fortes sur "1"

-------------------------------------------------------------------------------
 T | Duree | Dates limites [ ]      Diagramme de GANTT : RESSOURCE "R1"
-------------------------------------------------------------------------------
          +0.0 [||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||]+16.0

"1"  +6.0                                      +10.0[****************]***
"2"  +6.0                                      +10.0[********************]+16.0
"3"  +4.0                    +5.0 [*************                           ]+15.0
"4"  +5.0     +1.0 [*****************                                    ]+14.0
"5"  +4.0             +3.0 [*************                                 ]+15.0
"6"  +6.0  +0.0 [********************                                    ]+15.0
```

L'actualisation de la tâche "1" provoque une incohérence temporelle, détectée par REPORT. Ce problème est donc impossible à résoudre, compte tenu de ces contraintes. Cependant, l'impossibilité n'apparaît pas par le traitement de MASCOT2. Ses déductions sont les suivantes :

```
Recherche de contraintes temporelles, dans le cas disjonctif :
                                      dans le cas cumulatif :
        PRECEDE-INTERDIT("5","4")
        PRECEDE-INTERDIT("5","6")
        PRECEDE-INTERDIT("4","6")
        PRECEDE-INTERDIT("3","4")
        PRECEDE-INTERDIT("2","6")
        PRECEDE-INTERDIT("2","5")
        PRECEDE-INTERDIT("2","4")
        PRECEDE-INTERDIT("2","3")
        PRECEDE-INTERDIT("1","6")
        PRECEDE-INTERDIT("1","5")
        PRECEDE-INTERDIT("5","1")
        PRECEDE("6".nil,"1"."5".nil)
Actualisation de bornes temporelles :
        FIN-PLUS-TARD("6",11)
Recherche de contraintes temporelles, dans le cas disjonctif :
                                      dans le cas cumulatif :
        PRECEDE-INTERDIT("3","6")
        PRECEDE-INTERDIT("1","4")
        PRECEDE-INTERDIT("4","1")
        PRECEDE-INTERDIT("4","5")
        PRECEDE-INTERDIT("1","3")
        PRECEDE-INTERDIT("2","1")

                    Fin Analyse
                    ~~~~~~~~~~~
```

```
-------------------------------------------------------------------------------
 T | Duree | Dates limites [ ]      Diagramme de GANTT : RESSOURCE "M1"
-------------------------------------------------------------------------------
          +0.0 [||||||||||||||||||||||||||||||||||||||||||||||||||||||||||]+16.0

"1"  +6.0         +2.0 [********************                          ]+15.0
"2"  +6.0            +3.0 [********************                        ]+16.0
"3"  +4.0                    +5.0 [*************                        ]+15.0
"4"  +5.0     +1.0 [*****************                                ]+14.0
"5"  +4.0            +3.0 [*************                               ]+15.0
"6"  +6.0  +0.0 [********************                ]+11.0
```

A l'inverse, REPORT ne permet pas toujours de conclure aussi puissamment que ne le fait MASCOT2. Sur l'exemple du paragraphe V.5.1, REPORT ne fournit aucune conclusion.

V.5.3 Bilan – Propositions

Les exemples précédents montrent que les deux types d'analyse ne se recouvrent pas ; une étude, menant de front les deux approches, présente un intérêt certain.

L'efficacité du processus global dépend essentiellement de la stratégie utilisée dans le bloc de contrôle. Une stratégie générale semble s'imposer.

1. Comme cela a déjà été dit, le processus d'inférence par propagation doit être activé chaque fois que cela est possible, compte tenu de sa faible complexité.

2. Le processus de déduction numérique, qui n'entraîne pas un gros effort combinatoire, doit être utilisé en premier pour relancer le processus de propagation.

3. Quand aucune déduction n'est plus possible d'après 2., le processus de déduction symbolique peut être utilisé pour relancer le processus de propagation.

Cette stratégie est mise en œuvre sur un exemple qui fait l'objet du paragraphe suivant.

V.6 Exemple illustratif

On considère un ensemble $E = \{M, N, O, P\}$ de pièces à réaliser. La réalisation de la pièce X ($X \in E$) est constituée de deux tâches ($x1$ et $x2$), respectivement exécutées sur deux machines "mach1" et "mach2", et toujours dans cet ordre (la réalisation de X consiste à exécuter $x1$ sur "mach1" puis $x2$ sur "mach2"). Pour chaque pièce X, on a donc $x1$ précède $x2$. Les caractéristiques connues sont les suivantes :

k	mach1				mach2			
i	m1	n1	o1	p1	m2	n2	o2	p2
D_i	7	6	5	6	3	2	4	3
$\underline{C_i}$	2	3	1	0	-	-	-	-
$\overline{F_i}$	-	-	-	-	16	18	16	12
q_i^k	1	1	1	1	1	1	1	1

Tableau V.1

De plus, la disponibilité des ressources est définie par :

$$\begin{cases} Q^{mach1} = 2 \\ Q^{mach2} = 1 \end{cases}$$

Les contraintes de ressource sont donc de type disjonctif sur "mach2" et de type cumulatif sur "mach1".

La BdF correspondant à toutes ces données est la suivante.

```
"Travaux a realiser"
TRAVAIL("PIECE M","m1"."m2".nil)->;
TRAVAIL("PIECE N","n1"."n2".nil)->;
TRAVAIL("PIECE O","o1"."o2".nil)->;
TRAVAIL("PIECE P","p1"."p2".nil)->;

"Disponibilite des ressources"
DISPONIBILITE("mach1",2)->;
DISPONIBILITE("mach2",1)->;

"Quantites de ressources utilisees"
UTILISE("m1",<"mach1",1>.nil)->;
UTILISE("m2",<"mach2",1>.nil)->;
UTILISE("n1",<"mach1",1>.nil)->;
UTILISE("n2",<"mach2",1>.nil)->;
UTILISE("o1",<"mach1",1>.nil)->;
UTILISE("o2",<"mach2",1>.nil)->;
UTILISE("p1",<"mach1",1>.nil)->;
UTILISE("p2",<"mach2",1>.nil)->;

"Durees des taches"
DURE("m1",7)->;
DURE("m2",3)->;
DURE("n1",6)->;
DURE("n2",2)->;
```

```
DURE("o1",5)->;
DURE("o2",4)->;
DURE("p1",6)->;
DURE("p2",3)->;

"Bornes temporelles"
DEBUT-PLUS-TOT("m1",2)->;
DEBUT-PLUS-TOT("n1",3)->;
DEBUT-PLUS-TOT("o1",1)->;
DEBUT-PLUS-TOT("p1",0)->;
FIN-PLUS-TARD("m2",16)->;
FIN-PLUS-TARD("n2",18)->;
FIN-PLUS-TARD("o2",16)->;
FIN-PLUS-TARD("p2",12)->;

"Contraintes de precedence simple"
PRECEDE("m1".nil,"m2".nil)->;
PRECEDE("n1".nil,"n2".nil)->;
PRECEDE("o1".nil,"o2".nil)->;
PRECEDE("p1".nil,"p2".nil)->;

;End world:  Faits
```

Les raisonnements temporels et énergétiques décrits dans les chapitres III et IV, sont appliqués à ce problème, en utilisant successivement les logiciels REPORT et MASCOT. Les résultats d'analyse sont rapportés ci-dessous ; ils sont suivis d'explications complémentaires qui expliquent le déroulement des processus.

```
                    ========================================
                                    REPORT
                    ========================================
Activation du processus de propagation
        DEBUT-PLUS-TOT("p2",6)
        DEBUT-PLUS-TOT("o2",6)
        DEBUT-PLUS-TOT("n2",9)
        DEBUT-PLUS-TOT("m2",9)
        FIN-PLUS-TARD("p1",9)
        FIN-PLUS-TARD("o1",12)
        FIN-PLUS-TARD("n1",16)
        FIN-PLUS-TARD("m1",13)
-------------------------------------------------------------------------------
 T | Duree | Dates limites [ ]      Diagramme de GANTT : RESSOURCE "mach1"
-------------------------------------------------------------------------------
          +0.0 [||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||]+18.0

"m1" +7.0        +2.0 [**********************           ]+13.0
"n1" +6.0           +3.0 [******************                    ]+16.0
"o1" +5.0     +1.0 [***************                 ]+12.0
"p1" +6.0  +0.0 [*****************         ]+9.0
```

```
-------------------------------------------------------------------------------
 T | Duree | Dates limites [ ]      Diagramme de GANTT : RESSOURCE "mach2"
-------------------------------------------------------------------------------
          +0.0 [||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||]+18.0

"m2" +3.0                              +9.0 [********             ]+16.0
"n2" +2.0                              +9.0 [*****                      ]+18.0
"o2" +4.0                     +6.0 [***********                   ]+16.0
"p2" +3.0                     +6.0 [********        ]+12.0

Utilisation des regles (R4) et (R'4)
------------------------------------
Actualisation de bornes temporelles
       DEBUT-PLUS-TOT("n1",6)   (Intervalle d'etude : [3,10])
Activation du processus de propagation
       DEBUT-PLUS-TOT("n2",12)
Actualisation de bornes temporelles
       DEBUT-PLUS-TOT("o2",9)   (Intervalle d'etude : [6,12])
Actualisation de bornes temporelles
       DEBUT-PLUS-TOT("n2",13)  (Intervalle d'etude : [9,14])

                              Fin Analyse
                              -----------

-------------------------------------------------------------------------------
 T | Duree | Dates limites [ ]      Diagramme de GANTT : RESSOURCE "mach1"
-------------------------------------------------------------------------------
          +0.0 [||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||]+18.0

"m1" +7.0        +2.0 [**********************          ]+13.0
"n1" +6.0                     +6.0 [*****************            ]+16.0
"o1" +5.0     +1.0 [***************                 ]+12.0
"p1" +6.0  +0.0 [******************         ]+9.0

-------------------------------------------------------------------------------
 T | Duree | Dates limites [ ]      Diagramme de GANTT : RESSOURCE "mach2"
-------------------------------------------------------------------------------
          +0.0 [||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||]+18.0

"m2" +3.0                              +9.0 [********             ]+16.0
"n2" +2.0                                        +13.0[*****           ]+18.0
"o2" +4.0                              +9.0 [***********          ]+16.0
"p2" +3.0                     +6.0 [********        ]+12.0

-------------------------------------------------------------------------------
```

```
                    MASCOT

Recherche des ensembles critiques
        ENS-CRITIQUE("mach1","m1"."n1"."o1".nil)
        ENS-CRITIQUE("mach1","m1"."n1"."p1".nil)
        ENS-CRITIQUE("mach1","m1"."o1"."p1".nil)
        ENS-CRITIQUE("mach1","n1"."o1"."p1".nil)
        ENS-CRITIQUE("mach2","m2"."n2".nil)
        ENS-CRITIQUE("mach2","m2"."o2".nil)
        ENS-CRITIQUE("mach2","m2"."p2".nil)
        ENS-CRITIQUE("mach2","n2"."o2".nil)
        ENS-CRITIQUE("mach2","n2"."p2".nil)
        ENS-CRITIQUE("mach2","o2"."p2".nil)


Combinaison des regles (R1) d'une part et (R2) et (R3) d'autre part

Recherche de contraintes temporelles, dans le cas disjonctif :
        PRECEDE("m2".nil,"n2".nil)
        PRECEDE("p2".nil,"o2".nil)
        PRECEDE("p2".nil,"n2".nil)
        PRECEDE("o2".nil,"n2".nil)
Actualisation de bornes temporelles :
        DEBUT-PLUS-TOT("n2",16)
Recherche de contraintes temporelles, dans le cas disjonctif :
        PRECEDE("p2".nil,"m2".nil)
Actualisation de bornes temporelles :
        FIN-PLUS-TARD("p2",9)
Activation du processus de propagation
        FIN-PLUS-TARD("p1",6)
Recherche de contraintes temporelles, dans le cas disjonctif :
                                      dans le cas cumulatif :
        PRECEDE-INTERDIT("n1","o1")
        PRECEDE-INTERDIT("n1","p1")
        PRECEDE-INTERDIT("o1","p1")
        PRECEDE-INTERDIT("m1","o1")
        PRECEDE-INTERDIT("m1","p1")
        PRECEDE-INTERDIT("n1","m1")

                              Fin Analyse
```

```
-------------------------------------------------------------------------------
 T | Duree | Dates limites [ ]      Diagramme de GANTT : RESSOURCE "mach1"
-------------------------------------------------------------------------------
          +0.0 [||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||]+18.0

"m1" +7.0        +2.0 [**********************           ]+13.0
"n1" +6.0                      +6.0 [******************            ]+16.0
"o1" +5.0     +1.0 [***************                  ]+12.0
"p1" +6.0  +0.0 [******************]+6.0

-------------------------------------------------------------------------------
 T | Duree | Dates limites [ ]      Diagramme de GANTT : RESSOURCE "mach2"
-------------------------------------------------------------------------------
          +0.0 [||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||]+18.0

"m2" +3.0                              +9.0 [********            ]+16.0
"n2" +2.0                                                 +16.0 [******]+18.0
"o2" +4.0                              +9.0 [***********         ]+16.0
"p2" +3.0                       +6.0 [********]+9.0
```

On utilise tout d'abord REPORT. Les contraintes de précédence permettent, dans un premier temps, de lancer le processus de propagation et de déterminer ainsi les valeurs extrêmes non spécifiées des bornes temporelles de tâches ($\underline{C}_{x2}$ et $\overline{F}_{x1}$). On peut alors représenter par un diagramme à barres les ordonnancements admissibles avant toute analyse prenant en compte les contraintes de ressource. Dans un deuxième temps, la règle ($\Re 4$) permet de déduire trois actualisations (e.g. la date de début au plus tôt de $n1$ devient 6 après analyse énergétique sur un intervalle $[\underline{C}_{n1}, B_t] = [3, 10]$). [5]

Lorsque l'analyse de REPORT est achevée, le résultat est représenté sur un nouveau diagramme à barres. MASCOT prend alors le relais, en recherchant en premier lieu les ensembles critiques du problème. Par combinaison des règles ($\Re 1$) et ($\Re 2$) ou ($\Re 1$) et ($\Re 3$), on déduit ensuite des contraintes de précédence d'où peuvent découler des actualisations des valeurs extrêmes des bornes temporelles. Le résultat final est visualisé sur le dernier diagramme à barres. Sur "mach2", il est possible d'avoir $m2$ avant $o2$ ou $o2$ avant $m2$. Sur "mach1", une marge est laissée aux tâches $m1, n1, o1$ pour leur localisation dans le temps. On peut remarquer que les dates limites obtenues sur "mach1" sont effectivement atteignables. Il y a néanmoins un fort couplage entre les tâches $m1, n1, o1$. On peut noter en effet que $m1$ et $o1$ sont en disjonction sur l'intervalle $[2, 6]$. Il en résulte que :

- si $C_{m1} < 6$, alors $C_{o1} \geq 6$ et $C_{n1} \geq C_{m1} + D_{m1} \geq 9$,
- si $C_{m1} \geq 6$, alors $C_{o1} \leq 5$ et $C_{n1} \geq C_{o1} + D_{o1} \geq 6$.

[5] N.B. : dès qu'une nouvelle contrainte temporelle le permet, le processus de propagation est relancé.

Ainsi l'on voit, par exemple, que la tâche *o*1 n'utilise pas de façon continue sa marge : elle ne peut commencer sur l'intervalle]5,6[.

V.7 Conclusion

L'objet de ce dernier chapitre est la présentation des modules réalisés pour mettre en œuvre l'approche énergétique des problèmes d'ordonnancement : MASCOT2 et REPORT. Les principes généraux, la description des données, le contrôle de l'analyse ou les mécanismes de propagation de ces modules, sont décrits. En outre, le formalisme de la programmation logique, bien adaptée au raisonnement automatique, est introduit.

Une étude comparative est menée entre les différents modules, sur la complétude des conditions d'admissibilité obtenues. Elle met en évidence l'intérêt de combiner les processus d'analyse par résolution de conflits et ceux qui exploitent l'analyse énergétique pour modifier directement des dates limites.

A cet effet, une stratégie est proposée. Elle ne permet cependant pas d'éviter les redondances qu'entraînent la coexistence des traitements des deux modules. Un effort doit donc être poursuivi sur l'amélioration des mécanismes de contrôle de l'analyse dans chacune des approches, de manière à mieux les intégrer.

* *

*

Conclusion générale

Ce travail présente une nouvelle approche pour l'analyse sous contraintes de problèmes d'ordonnancement. Dans les problèmes considérés, l'exécution des tâches est soumise à des contraintes de temps (fenêtre temporelle allouée pour l'exécution d'une tâche) et des contraintes de ressources (utilisation de moyens disponibles en quantité limitée).

La prise en compte simultanée du temps et des ressources permet d'introduire un raisonnement basé sur des bilans énergétiques permettant de traiter de manière homogène, les ressources disjonctives et cumulatives. Ce couplage du temps et des ressources est réalisé à l'aide du concept d'intervalle temps-ressource ; celui-ci permet de représenter les tâches ou intervalles consommateurs et les intervalles de temps alloués sur lesquels des ressources sont disponibles, appelés intervalles fournisseurs. L'originalité de cette approche réside dans l'utilisation du raisonnement énergétique pour la caractérisation de l'ensemble des décisions compatibles avec les contraintes ou ensemble des solutions admissibles.

Différentes règles élémentaires d'inférence ont ainsi été présentées. Elles ont donné lieu à la réalisation de deux modules d'analyse sous contraintes implémentés en Prolog. Le premier, MASCOT2, basé sur des conditions de séquencement issues de la résolution de conflits, constitue un prolongement de MASCOT1, déjà réalisé il y a quelques années. Le second, REPORT, permet d'affiner les valeurs limitant l'intervalle de temps initialement alloué à une tâche en considérant la consommation obligatoire d'énergie des autres tâches sur cet intervalle. La combinaison de ces règles a également été proposée suivant une stratégie permettant d'exploiter les avantages respectifs de chaque approche.

Le problème de l'intégration harmonieuse de ces deux approches reste toutefois ouvert : il faudrait essentiellement arriver à les combiner sans redondance de l'analyse. On peut ainsi envisager l'affinement des mécanismes de contrôle de ces processus en utilisant par exemple des notions de criticité pour guider l'application des règles.

De plus, des extensions du modèle retenu pourraient être étudiées en s'appuyant sur les concepts d'intervalles temps-ressource et d'énergie. Il serait par exemple

intéressant de prendre en compte des tâches interruptibles et des tâches dont la durée dépend de la quantité de ressource allouée. L'étude des intervalles discontinus qui a été ébauchée pourrait, de même, contribuer à la modélisation des horaires de travail ou des ressources à disponibilité intermittente.

Pour les types de problèmes traités dans notre travail, nous avons pu conserver le processus de propagation utilisé dans MASCOT1. Les extensions précédentes nécessitent néanmoins l'utilisation d'une propagation basée sur un graphe potentiels-bornes.

En outre, les études menées sur le raisonnement énergétique montrent que le choix de l'intervalle fournisseur est un facteur déterminant pour la qualité des déductions obtenues. Même si certaines extensions sont proposées, la définition des intervalles fournisseurs mérite d'être approfondie et peut donc être envisagée comme une continuation de ce travail.

Par ailleurs, le raisonnement énergétique offre de nouvelles perspectives pour des raisonnements plus agrégés sur le problème. A ce sujet, il serait utile de développer un module permettant d'évaluer le taux de contraintes d'un problème avant de le soumettre à l'analyse de MASCOT ou REPORT de manière à limiter le risque d'explosion combinatoire. Ce taux pourrait en effet être utilisé pour guider le choix de la méthode de résolution et concilier ainsi différentes approches. De plus, le module de relaxation de contraintes, destiné, en cas d'incohérence, à aider à la remise en cause de décisions prises à un autre niveau, n'a pas encore été étudié. Cette voie de recherche pourrait définir une "méta-analyse" d'un problème d'ordonnancement, pouvant jouer un rôle dans la coordination au sein d'un réseau de centres de décision.

* *

*

Bibliographie

[Akers 56] **S.B. AKERS**
A graphical approach to production scheduling problems
Operations Research, 4, 1956.

[Allen 81] **J.F. ALLEN**
An interval based representation of temporal knowledge
7^{th} IJCAI, pp. 221–226, Vancouver, Canada, 1981.

[Allen 83] **J.F. ALLEN**
Maintening knowledge about temporal intervals
Communications of the ACM, 26, pp. 832–843, 1983.

[Ashour 70] **S. ASHOUR**
A branch and bound algorithm for flow-shop scheduling problems
AIIE Transactions, 2, pp. 172–176, 1970.

[Baker 74] **K. R. BAKER**
Introduction to sequencing and scheduling
John Wiley & Sons, New-York, 1974.

[Baker 75] **K. R. BAKER**
A comparative study of flow-shop algorithms
Operations Research, 23, pp. 62–73, 1975.

[Balas 69] **E. BALAS**
Machine sequencing via disjunctive graphs : an implicit enumeration algorithm
Operations Research, 17, pp. 941–957, 1969.

[Baptiste 85] **P. BAPTISTE**
Contribution à la conception d'un atelier flexible : définition de la base de données techniques, ordonnancement de tâches à temps de réglage variables
Thèse de doctorat de l'INSA de Lyon, 1985.

[Battersby 67] **A. BATTERSBY**
Méthodes modernes d'ordonnancement
Dunod, Paris, 1967.

[Bel et al. 87] **G. BEL, E. BENSANA, D. DUBOIS**
Construction d'ordonnancements prévisonnels : un compromis entre approches classiques et systèmes experts
$2^{ème}$ Conférence Internationale Systèmes de Production, pp. 709–726, Paris, Avril 1987.

[Bel et al. 89] **G. BEL, E. BENSANA, D. DUBOIS, J. ERSCHLER, P. ESQUIROL**
A knowledge based approach to industrial job-shop scheduling
In "Knowledge-based systems in manufacturing" edited by A. Kusiak, Taylor & Francis, pp. 207–246, 1989.

[Bellman et al. 82] **R. BELLMAN, A.O. ESOGBUE, I. NABESHIMA**
Mathematical aspects of scheduling and applications
Pergamon Press, Oxford, 1982.

[Bensana 87] **E. BENSANA**
Utilisation de techniques d'intelligence artificielle pour l'ordonnancement d'atelier
Thèse de doctorat de l'ENSAE, Toulouse, 1987.

[van Benthem 83] **J.F.A.K. VAN BENTHEM**
The logic of time
Reidel, Dordrecht, 1983.

[Bestougeff et Ligozat 89] **H. BESTOUGEFF, G. LIGOZAT**
Outils logiques pour le traitement du temps
Masson, Paris, 1989.

[Brücker 88] **P. BRUCKER**
An efficient algorithm for the job-shop problem with two jobs
Computing, 40, pp. 353–359, 1988.

[Buffa et Miller 79] **E.S. BUFFA, J.G. MILLER**
Production inventory systems : planning and control
Irwin Inc., 1979.

[Canals 86] **D. CANALS**
Ordonnancement d'atelier par simulation : étude des règles de priorité et aide au lancement
Thèse de docteur-ingénieur de l'ENSAE, Toulouse, 1986.

[Carlier et Chrétienne 88] **J. CARLIER, P. CHRETIENNE**
Problèmes d'ordonnancement : modélisation / complexité / algorithmes
Masson, Paris, 1988.

[Carlier et Pinson 89] **J. CARLIER, E. PINSON**
A branch and bound method for solving the job-shop problem
Management Science, 35, pp. 164–176, 1989.

[Carlier 90] **J. CARLIER, P. VILLON**
A new heuristic for the travelling salesman problem
RAIRO, Recherche Opérationnelle, 24, pp. 245–253, 1990.

[Coffman et Graham 72] **E.G. COFFMAN, R.L. GRAHAM**
Optimal scheduling for two processor systems
Acta informatica, 1, pp. 200–213, 1972.

[Collinot et al. 88] **A. COLLINOT, C. LE PAPE, G. PINOTEAU**
SONIA : a knowledge-based scheduling system
International Journal for Artificial Intelligence in Engineering, 3(2), pp. 86–94, 1988.

[Conway et al. 67] **R.W.CONWAY, W.L. MAXWELL, L.W. MILLER**
Theory of scheduling
Addison Wesley, Reading Mass, 1967.

[Davis 87] **E. DAVIS**
Constraint propagation with interval labels
Artificial Intelligence, 32, pp. 281–331, 1987.

[Demmou 77] **R. DEMMOU**
Etude de familles remarquables d'ordonnancements en vue d'une aide à la décision
Thèse de docteur-ingénieur, Université Paul Sabatier, Toulouse, 1977.

[Dibon 70] **M. DIBON**
Ordonnancement et potentiels / Méthode M.P.M.
Hermann, Paris, 1970.

[Erschler 76] **J. ERSCHLER**
Analyse sous contraintes et aide à la décision pour certains problèmes d'ordonnancement
Thèse de doctorat d'état, Université Paul Sabatier, Toulouse, 1976.

[Erschler et al. 76] **J. ERSCHLER, F. ROUBELLAT, J.P. VERNHES**
Finding some essential characteristics of the feasible solutions for a scheduling problem
Operations Research, 24, pp. 774–783, 1976.

[Erschler et al. 79] **J. ERSCHLER, G. FONTAN, F. ROUBELLAT**
Potentiels sur un graphe non conjonctif et analyse d'un problème d'ordonnancement à moyens limités
RAIRO, Recherche Opérationnelle, 13, pp. 363–378, 1979.

[Erschler et al. 80] **J. ERSCHLER, F. ROUBELLAT, J.P. VERNHES**
Characterizing the set of feasible sequences for n jobs to be carried out on a single machine
European Journal of Operational Research, 4, pp. 189–194, 1980.

[Erschler et Esquirol 86] **J. ERSCHLER, P. ESQUIROL**
Decision-aid in job shop scheduling : a knowledge based approach
IEEE International Conference on Robotics and Automation, pp. 1651–1656, San Francisco, California, 1986.

[Erschler et de Terssac 88] **J. ERSCHLER, G. DE TERSSAC**
Flexibilité et rôle de l'opérateur humain dans l'automatisation intégrée de production
Rapport LAAS n° 88137, Mars 1988.

[Erschler et al. 91a] **J. ERSCHLER, G. FONTAN, F. ROUBELLAT**
Ordonnancement en ateliers spécialisés
Encyclopédie des sciences de gestion, édition Vuibert, 1991 (sous presse).

[Erschler et al. 91b] **J. ERSCHLER, P. LOPEZ, C. THURIOT**
Raisonnement temporel sous contraintes de ressources et problèmes d'ordonnancement
Rapport LAAS n° 89237, 1989. A paraître dans la "Revue d'Intelligence Artificielle".

[Esquirol 87] **P. ESQUIROL**
Règles et processus d'inférence pour l'aide à l'ordonnancement de tâches en présence de contraintes
Thèse de doctorat de l'Université Paul Sabatier, Toulouse, 1987.

[Faure 86] **G. FAURE**
Planification hiérarchisée en avenir incertain : le concept de robustesse
Thèse de docteur de $3^{ème}$ cycle, Université Paul Sabatier, Toulouse, 1986.

[Faure et al. 76] **R. FAURE, C. ROUCAIROL, P. TOLLA**
Chemins, flots et ordonnancements
Gauthiers-Villars, Paris, 1976.

[Fox et Smith 84] **M.S. FOX, S.F. SMITH**
ISIS : a knowledge-based system for factory scheduling
Expert Systems, **1**(1), pp. 25–49, 1984.

[Fox 87] **M.S. FOX**
Constraint-directed search : a case study of job-shop scheduling
Pitman, London, 1987.

[Gantt 19] **H. L. GANTT**
Organizing for work
Harcourt, Brace and Howe, New-York, 1919.

[Giannesini et al. 85] **F. GIANNESINI, H. KANOUI, R. PASERO, M. VAN CANEGHEM**
Prolog
Inter Editions, Paris, 1985.

[Giard 88] **V. GIARD**
Gestion de la production
2ème édition, Economica, Paris, 1988.

[Gotha 91] **GOTHA (groupe d'ordonnancement théorique et appliqué)**
J.Carlier,P.Chrétienne,J.Erschler,C.Hanen,P.Lopez,E.Pinson,M.C.Portmann,C.Prins,C.Proust,P.Villon
Les problèmes d'ordonnancement
Soumis à RAIRO, Recherche Opérationnelle.

[Graves 81] **S.C. GRAVES**
A review of production scheduling
Operations Research, 29, pp. 646–675, 1981.

[Gupta 71] **J.N.D. GUPTA**
An improved combinatorial algorithm for the flow-shop scheduling problem
Operations Research, 19, pp. 1753–1758, 1971.

[Imbert 86] **S. IMBERT**
Interaction entre deux niveaux de décisions en planification de la production
Thèse de docteur de 3ème cycle, Université Paul Sabatier, 1986.

[Jackson 55] **J.R. JACKSON**
Scheduling a production line to minimize maximum tardiness
Research report 43, Management sciences research project, UCLA, 1955.

[Jackson 56] **J.R. JACKSON**
An extension of Johnson's results on job lot scheduling
Naval Research Logistic Quartely, 3, pp. 201–203, 1956.

[Kaufmann et Desbazeille 64] **A. KAUFMANN, G. DESBAZEILLE**
La méthode du chemin critique
Dunod, Paris, 1964.

[Johnson 54] **S.M. JOHNSON**
Optimal two- and three-stage production schedules with setup times included
Naval Research Logistic Quartely, 1, pp. 61–68, 1954.

[Lageweg et al. 77] **B.J. LAGEWEG, J.K. LENSTRA, A.H.G. RINNOOY KAN**
Job-shop scheduling by implicit enumeration
Management Science, 24, pp. 441–450, 1977.

[Lageweg et al. 78] **B.J. LAGEWEG, J.K. LENSTRA, A.H.G. RINNOOY KAN**
A general bounding scheme for the permutation flow-shop problem
Operations Research, 26, pp. 53–67, 1978.

[Lahrichi 82] **A. LAHRICHI**
Ordonnancements : les notions de bosse, de partie obligatoire et leurs applications aux problèmes cumulatifs
C.R. Acad. Sc. Paris, t. 294, série I–16, pp. 209–211, 1982.

[Lawler et al. 89] **E.L. LAWLER, J.K. LENSTRA, A.H.G. RINNOOY KAN, D.B. SHMOYS**
Sequencing and scheduling : algorithms and complexity
Report BS-R8909, Centre for Mathematics and Computer Science, Amsterdam, 1989.

[Leachman et al. 90] **R.C. LEACHMAN, A. DINCERLER, S. KIM**
Resource-constrained scheduling of projects with variable-intensity activities
IIE Transactions, 22, pp. 31–40, 1990.

[Le Gall 89] **A. LE GALL**
Un système interactif d'aide à la décision pour l'ordonnancement et le pilotage en temps réel d'atelier
Thèse de doctorat de l'Université Paul Sabatier, Toulouse, 1989.

[Le Pape et Sauve 85] **C. LE PAPE, B. SAUVE**
SOJA : un Système d'Ordonnancement Journalier d'Atelier
$5^{èmes}$ journées internationales sur les systèmes experts et leurs applications, Avignon, 1985.

[Le Pape 86] **C. LE PAPE**
OPIS 1 : un système d'ordonnancement opportuniste
Rapport final de bourse INRIA, Carnegie-Mellon University, 1986.

[Le Pape 88] **C. LE PAPE**
Des systèmes d'ordonnancement flexibles et opportunistes
Thèse de doctorat en sciences, Orsay, Paris, 1988.

[Le Pape 91] **C. LE PAPE**
Constraint propagation in planning and scheduling
CIFE technical report, Stanford University, Janvier 1991.

[Lin et Kernighan 71] **S. LIN, B.W. KERNIGHAN**
An effective heuristic algorithm for the travelling salesman problem
Operations Research, 21, pp. 498–511, 1971.

[Mc Mahon et Burton 67] **G.B. MAC MAHON, P.G. BURTON**
Flow-shop scheduling with the branch and bound method
Operations Research, 15, pp. 473–481, 1967.

[Mc Naughton 59] **R. MAC NAUGHTON**
Scheduling with deadlines and loss functions
Management Science, 6, pp. 1–12, 1959.

[Meguelati 88] **S. MEGUELATI**
Méthodes de classification pour la constitution d'ilôts de fabrication et l'ordonnancement
Thèse de doctorat de l'INSA de Toulouse, 1988.

[Mercé 87] **C. MERCE**
Cohérence des décisions en planification hiérarchisée
Thèse de doctorat d'état, Université Paul Sabatier, Toulouse, 1987.

[Muth et Thompson 63] **J.F. MUTH, G.L. THOMPSON**
Industrial scheduling
Prentice Hall, Englewood Cliffs, New-Jersey, 1963.

[Nabeshima 73] **I. NABESHIMA**
Algorithms and reliable heuristics programs for multi-projects scheduling with resource constraints and related parallel scheduling
University of Electrocommunication, Chofu, Tokyo, 1973.

[Nemhauser et Wolsey 88] **G.L. NEMHAUSER, L.A. WOLSEY**
Integer and combinatorial optimization
John Wiley & Sons, New-York, 1988.

[Portmann 88] **M.C. PORTMANN**
Méthodes de décomposition spatiale et temporelle en ordonnancement de la production
RAIRO, APII, 22, pp. 439–451, 1988.

[Proust et al. 87] **C. PROUST, M. DROGOU, J.M. FOUCHER, E. FOUCHEYRAND**
Une heuristique pour le problème d'ordonnancement statique de type n/m/F avec prise en compte des temps de montage et démontage d'outils
$2^{ème}$ Conférence Internationale Systèmes de Production, pp. 125–141, Paris, Avril 1987.

[Queyranne 88] **M. QUEYRANNE**
Structure of a simple scheduling polyhedron
Rapport LAAS n° 88014, Janvier 1988. A paraître dans Mathematical Programming.

[Rinnooy Kan 76] **A.H.G. RINNOOY KAN**
Machine sequencing problems : classification, complexity and computation
Nijhoff, The Hague, 1976.

[Rit 86] **J.F. RIT**
Propagating temporal constraints for scheduling
5^{th} National Conference on Artificial Intelligence (A.A.A.I.-86), pp. 383–388, Philadelphia, Pennsylvania, 1986.

[Rit 88] **J.F. RIT**
Modélisation et propagation de contraintes temporelles pour la planification
Thèse de doctorat de l'INPG, Grenoble, 1988.

[Roy 70] **B. ROY**
Algèbre moderne et théorie des graphes, tome II
Dunod, Paris, 1970.

[Saveplan 84]
Saveplan : version 4.0
Note technique, CMG, Les Ullis, 1984.

[Schrage 70] **L. SCHRAGE**
Solving resource-constrained network problems by implicit enumeration / Nonpreemptive case
Operations Research, 18, pp. 263–278, 1970.

[Schrage et Baker 78] **L. SCHRAGE, K.R. BAKER**
Dynamic programming solution of sequencing problems with precedence constraints
Operations Research, 26, pp. 444–449, 1978.

[Słowiński 81] **R. SŁOWIŃSKI**
Multiobjective network scheduling with efficient use of renewable and non-renewable resources
European Journal of Operational Research, 7, pp. 265–273, 1981.

[Słowiński 82] **R. SŁOWIŃSKI**
Preemptive scheduling of independent jobs on parallel machines subject to financial constraints
European Journal of Operational Research, 1982.

[Smith 56] **W.E. SMITH**
Various optimizers for single-state production
Naval Research Logistic Quartely, 3, pp. 59–66, 1956.

[Steffen 86] **M.S. STEFFEN**
A survey of AI based scheduling systems
Fall Industrial Engineering Conference, Boston, Massachusetts, 1986.

[Szwarc 60] **W. SZWARC**
Solution for the Akers-Friedman scheduling problem
Operations Research, 8, pp. 782–788, 1960.

[Thomas 80] **V. THOMAS**
Aide à la décision pour l'ordonnancement d'atelier en temps réel
Thèse de docteur de $3^{ème}$ cycle, Université Paul Sabatier, Toulouse, 1980.

[Thuriot et al. 90] **C. THURIOT. J. ERSCHLER, P. LOPEZ**
The context of a temporal boundary : structuration for decisions of load temporal distribution in scheduling
In "Advances in support systems research" edited by G.E. Lasker & R.R. Hough, I.I.A.S., pp. 189–194, 1990.

[Tzar-II 83]
TZAR II : système de gestion de production
Production Systèmes, 1983.

[Węglarz 81] **J. WĘGLARZ**
Project scheduling with continuously-divisible, doubly constrained resources
Management Science, 27, pp. 1040–1053, 1981.

[Yamamoto 77] **M. YAMAMOTO**
An approximate solution of machine scheduling problems by decomposition methods
International Journal of Production Research, 15, pp. 599–608, 1977.

ANNEXE : REPORT

```
"~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~"
"           BOUCLE GENERALE DE REPORT            "
"~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~"

inter ->
   conversion
   impasse;
inter ->
   chercher-racines
   impasse;
inter ->
   page
   toutes-les-actualisations
   impasse;
inter ->
   init
   reset-cpu-time
   block(e,exec)
   cpu-time(x)
   gantt("r")
   line
   line
   outm("Temps CPU ecoule = ")
   out(x)
   outm(" ms")
   impasse;

"Initialisation pour l'affichage"
"Affichage du diagramme a barres par ressource"

init ->
   si(rule(norme-affichage(m,M,N),nil),suppress(1))
   affichage
   page
   gantt("r")
   line
   line;

"Boucles d'analyse et de propagation de dates"

exec ->
   debut
   outml("Actualisation de bornes temporelles")
   impasse;
exec ->
   toutes-les-actualisations
   impasse;
exec -> exec;

"Lancement de l'analyse : "
"1. au plus tot"
"2. au plus tard"
"3. jusqu'a ce qu'on ne puisse plus rien deduire"

debut ->
   DISPONIBILITE(R,Q)
   liste-des(o,utilise(R,o,q),T)
   enleve(x,T,T')
   utilise(R,x,q-i)
   Bornes-Sup(x,T',L)
   calcul(som-list,conso-energ-F(x,L),T',T'')
   search-tmax(x,L,T'',Q,q-i,t.nil)
   actua-C(x,t,Q,q-i,T',C-act)
```

```
    si(PRECEDE(x.nil,Y),precede-actualisant(x,"avant",Y))
    /;
debut ->
    DISPONIBILITE(R,Q)
    liste-des(o,utilise(R,o,q),T)
    enleve(x,T,T')
    utilise(R,x,q-i)
    Bornes-Inf(x,T',L)
    calcul(som-list,conso-energ-C(x,L),T',T'')
    search-tmin(x,L,T'',Q,q-i,t.nil)
    actua-F(x,t,Q,q-i,T',F-act)
    si(PRECEDE(Y,x.nil),precede-actualisant(x,"apres",Y))
    /;
debut ->
    line
    outm("                          FIN ANALYSE")
    line
    block-exit(fin-analyse);


"~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~"
"           OPERATIONS DE CONVERSION                         "
"~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~"
"Conversion de nombres entiers en nombres reels"

conversion ->
    rule(DEBUT-PLUS-TOT(x,i),nil)
    integer(i)
    val(float(i),r)
    suppress(1)
    assert'(DEBUT-PLUS-TOT(x,r),nil);
conversion ->
    rule(FIN-PLUS-TARD(x,i),nil)
    integer(i)
    val(float(i),r)
    suppress(1)
    assert'(FIN-PLUS-TARD(x,r),nil);
conversion ->
    rule(DURE(x,i),nil)
    integer(i)
    val(float(i),r)
    suppress(1)
    assert'(DURE(x,r),nil);
conversion ->
    rule(DISPONIBILITE(w,i),nil)
    integer(i)
    val(float(i),r)
    suppress(1)
    assert'(DISPONIBILITE(w,r),nil);
conversion ->
    rule(UTILISE(x,I),nil)
    conversion-liste(I,R)
    suppress(1)
    assert'(UTILISE(x,R),nil);

conversion-liste(nil,nil) ->;
conversion-liste(<x,i>.I,<x,r>.R) ->
    integer(i)
    val(float(i),r)
    conversion-liste(I,R);

"Conversion de nombres reels en nombres entiers"

reel-entier ->
```

```
    rule(DEBUT-PLUS-TOT(x,i),nil)
    real(i)
    val(trunc(i),r)
    suppress(1)
    assert'(DEBUT-PLUS-TOT(x,r),nil);
reel-entier ->
    rule(FIN-PLUS-TARD(x,i),nil)
    real(i)
    val(trunc(i),r)
    suppress(1)
    assert'(FIN-PLUS-TARD(x,r),nil);
reel-entier ->
    rule(DURE(x,i),nil)
    real(i)
    val(trunc(i),r)
    suppress(1)
    assert'(DURE(x,r),nil);
reel-entier ->
    rule(DISPONIBILITE(w,i),nil)
    real(i)
    val(trunc(i),r)
    suppress(1)
    assert'(DISPONIBILITE(w,r),nil);
reel-entier ->
    rule(UTILISE(x,I),nil)
    reel-entier-liste(I,R)
    suppress(1)
    assert'(UTILISE(x,R),nil);

reel-entier-liste(nil,nil) ->;
reel-entier-liste(<x,i>.I,<x,r>.R) ->
    real(i)
    val(trunc(i),r)
    reel-entier-liste(I,R);


"~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~"
"       ANALYSE ENERGETIQUE AU PLUS TOT                     "
"~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~"
"Calcul des instants remarquables sur [Ci,t]"

Bornes-Sup(i,nil,nil) ->;
Bornes-Sup(i,j.T,L) ->
    CONSO(i,j,v)
    si-alors-sinon(dif(v,+0.0),bornes-sup(i,j,L1),eq(L1,nil))
    Bornes-Sup(i,T,L2)
    conca(L1,L2,l)
    sans-repet(l,L')
    ordonner(L',nil,L);

bornes-sup(i,j,F-t.F.b3) ->
    DEBUT-PLUS-TOT(i,C)
    FIN-PLUS-TARD(i,F)
    FIN-PLUS-TOT(i,F-t)
    DEBUT-PLUS-TOT(j,c)
    FIN-PLUS-TARD(j,f)
    DURE(i,D)
    DURE(j,d)
    val(sub(f,d),x)
    somme(f,c,z)
    somme(c,d,w)
    val(sub(z,C),y)
    si-alors-sinon(val(inf(C,x),1).val(inf(x,F),1),eq(b1,x.nil),eq(b1,nil))
    si-alors-sinon(val(inf(C,w),1).val(inf(w,F),1).ou(val(inf(x,C),1),eq(x,C)),eq
```

```
    (b4,w.b1),eq(b4,b1))
    si-alors-sinon(val(inf(C,y),1).val(inf(y,F),1).val(inf(c,C),1).val(inf(C,x),1
    ),eq(b2,y.b4),eq(b2,b4))
    si-alors-sinon(ou(val(inf(C,c),1),eq(C,c)).val(inf(c,f),1).val(inf(f,F),1),eq
    (b3,f.b2),eq(b3,b2));

"Consommation obligatoire temporelle sur [Ci,t]"

CONSO-F(i,j,t,c-onso-tF) ->
   DEBUT-PLUS-TOT(i,C)
   DEBUT-PLUS-TARD(j,c)
   FIN-PLUS-TOT(j,f)
   DURE(j,D)
   val(sub(f,C),m1)
   val(sub(t,c),m2)
   val(sub(t,C),m3)
   min(m1,m2,m')
   min(m',m3,m'')
   min(m'',D,m''')
   max(+0.0,m''',c-onso-tF);

"Consommation obligatoire sur [Ci,t]"

conso-energ-F(i,nil,j,nil) -> /;
conso-energ-F(i,t.L,j,a.l) ->
   utilise(R,j,q-j)
   CONSO-F(i,j,t,b)
   produit(q-j,b,a)
   conso-energ-F(i,L,j,l);

"Recherche de l'instant tm"

search-tmax(i,nil,nil,Q,q,nil) ->;
search-tmax(i,t.T,c.l,Q,q,L) ->
   DEBUT-PLUS-TOT(i,C)
   DURE(i,D)
   val(sub(t,C),f1)
   produit(f1,Q,f2)
   val(sub(f2,c),f)
   min(f1,D,e-coule)
   produit(e-coule,q,r-equis)
   si-alors-sinon(val(inf(f,r-equis),1),eq(L,t.nil),search-tmax(i,T,l,Q,q,L));

"Actualisation de la date de debut au plus tot"

actua-C(i,t,Q,q,l,C-new) ->
   DEBUT-PLUS-TOT(i,C)
   calcul(som-list,conso-energ-F(i,t.nil),l,c-onso.nil)
   val(div(Q,q),a-lpha)
   val(sub(+1.0,a-lpha),u)
   produit(t,u,t-erme1)
   produit(C,a-lpha,t-erme2)
   somme(t-erme1,t-erme2,v)
   val(div(c-onso,q),b-eta)
   somme(v,b-eta,C'-new)
   si(rule(DEBUT-PLUS-TOT(i,C),nil),val(inf(C,C'-new),1).suppress(1))
   val(trunc(C'-new),C-int)
   reel(C-int,C-real)
   val(sub(C'-new,C-real),r-est)
   si-alors-sinon(eq(r-est,+0.0),eq(C-real,C-new),somme(C-real,+1.0,C-new))
   assert'(DEBUT-PLUS-TOT(i,C-new),nil)
   outl(DEBUT-PLUS-TOT(i,C-new))
   test-coherence(i);
```

```
"~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~"
"       ANALYSE ENERGETIQUE AU PLUS TARD          "
"~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~"
"Calcul des instants remarquables sur [t,Fi]"

Bornes-Inf(i,nil,nil) ->;
Bornes-Inf(i,j.T,L) ->
   CONSO(i,j,v)
   si-alors-sinon(dif(v,+0.0),bornes-inf(i,j,L1),eq(L1,nil))
   Bornes-Inf(i,T,L2)
   conca(L1,L2,l)
   sans-repet(l,L')
   ordonner2(L',nil,L);

bornes-inf(i,j,C-t.C.b3) ->
   DEBUT-PLUS-TOT(i,C)
   FIN-PLUS-TARD(i,F)
   DEBUT-PLUS-TARD(i,C-t)
   DEBUT-PLUS-TOT(j,c)
   FIN-PLUS-TARD(j,f)
   DURE(i,D)
   DURE(j,d)
   somme(c,d,x)
   val(sub(f,d),w)
   somme(f,c,z)
   val(sub(z,F),y)
   si-alors-sinon(val(inf(C,x),1).val(inf(x,F),1),eq(b1,x.nil),eq(b1,nil))
   si-alors-sinon(val(inf(C,w),1).val(inf(w,F),1).ou(val(inf(F,x),1),eq(F,x)),eq
   (b4,w.b1),eq(b4,b1))
   si-alors-sinon(val(inf(C,y),1).val(inf(y,F),1).val(inf(x,F),1).val(inf(F,f),1
   ),eq(b2,y.b4),eq(b2,b4))
   si-alors-sinon(val(inf(C,c),1).val(inf(c,f),1).ou(val(inf(f,F),1),eq(f,F)),eq
   (b3,c.b2),eq(b3,b2));

"Consommation obligatoire temporelle sur [t,Fi]"

CONSO-C(i,j,t,c-onso-tC) ->
   FIN-PLUS-TARD(i,F)
   DEBUT-PLUS-TARD(j,c)
   FIN-PLUS-TOT(j,f)
   DURE(j,D)
   val(sub(f,t),m1)
   val(sub(F,c),m2)
   val(sub(F,t),m3)
   min(m1,m2,m')
   min(m',m3,m'')
   min(m'',D,m''')
   max(+0.0,m''',c-onso-tC);

"Consommation obligatoire sur [t,Fi]"

conso-energ-C(i,nil,j,nil) -> /;
conso-energ-C(i,t.L,j,a.l) ->
   utilise(R,j,q-j)
   CONSO-C(i,j,t,b)
   produit(q-j,b,a)
   conso-energ-C(i,L,j,l);

"Recherche de tm"

search-tmin(i,nil,nil,Q,q,nil) ->;
search-tmin(i,t.T,c.l,Q,q,L) ->
   FIN-PLUS-TARD(i,F)
```

```
   DURE(i,D)
   val(sub(F,t),f1)
   produit(f1,Q,f2)
   val(sub(f2,c),f)
   min(f1,D,e-coule)
   produit(e-coule,q,r-equis)
   si-alors-sinon(val(inf(f,r-equis),1),eq(L,t.nil),search-tmin(i,T,1,Q,q,L));

"Actualisation de la date de fin au plus tard"

actua-F(i,t,Q,q,1,F-new) ->
   FIN-PLUS-TARD(i,F)
   calcul(som-list,conso-energ-C(i,t.nil),1,c-onso.nil)
   val(div(Q,q),a-lpha)
   val(sub(+1.0,a-lpha),u)
   produit(t,u,t-erme1)
   produit(F,a-lpha,t-erme2)
   somme(t-erme1,t-erme2,v)
   val(div(c-onso,q),b-eta)
   val(sub(v,b-eta),F'-new)
   si(rule(FIN-PLUS-TARD(i,F),nil),val(inf(F'-new,F),1).suppress(1))
   val(trunc(F'-new),F''-new)
   reel(F''-new,F-new)
   assert'(FIN-PLUS-TARD(i,F-new),nil)
   outl(FIN-PLUS-TARD(i,F-new))
   test-coherence(i);


"~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~"
"   PROPAGATION DES ACTUALISATIONS DE DATES   "
"~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~"
"Definition des sources et puits initiaux"

chercher-racines ->
   DURE(x,P)
   si(tache-debut(x),creer(SOURCE(x),nil))
   si(tache-fin(x),creer(PUITS(x),nil));

tache-debut(x) ->
   DEBUT-PLUS-TOT(x,C)
   non(PRECEDE(y.nil,x.nil));

tache-fin(x) ->
   FIN-PLUS-TARD(x,F)
   non(PRECEDE(x.nil,y.nil));

"Controle des actualisations"

toutes-les-actualisations ->
   propager(SOURCE);
toutes-les-actualisations ->
   propager(PUITS);

"La propagation s'arrete lorsqu'il n'y a plus de SOURCES (resp PUITS)."
"Une liste des taches SOURCES (resp. PUITS), est reconstituee lorsque"
"toutes les actualisations possibles ont ete realisees a partir"
"des taches de la liste precedente"

propager(R) -> non(<R,x>) /;
propager(R) -> liste-des(x,<R,x>,E) controle-actualisation(R,E);
propager(R) -> propager(R);

controle-actualisation(R,nil) ->;
controle-actualisation(SOURCE,x.S) ->
```

```
      PRECEDE(X,y.nil)
      dans'(x,X)
      type-actual-S(X,y,A)
      A
      creer(SOURCE(y),nil)
      impasse;
   controle-actualisation(PUITS,y.P) ->
      PRECEDE(x.nil,Y)
      dans'(y,Y)
      type-actual-P(x,Y,A)
      A
      creer(PUITS(x),nil)
      impasse;
   controle-actualisation(R,x.E) ->
      rule(<R,x>,nil)
      suppress(1)
      controle-actualisation(R,E);

   "Differents types d'actualisation"

   type-actual-S(x.nil,y,actual-deb(bloc,Z,y)) ->
      ou(paire-critique(R,x,y),paire-critique(R,y,x))
      former-bloc-avant(R,y,Z);
   type-actual-S(Z,y,actual-deb(faisceau,Z,y)) ->;

   type-actual-P(x,y.nil,actual-fin(bloc,x,Z)) ->
      ou(paire-critique(R,x,y),paire-critique(R,y,x))
      former-bloc-apres(R,Z,x);
   type-actual-P(x,Z,actual-fin(faisceau,x,Z)) ->;

   paire-critique(R,x,y) ->
      utilise(R,x,q1)
      utilise(R,y,q2)
      somme(q1,q2,S)
      DISPONIBILITE(R,Q)
      val(inf(Q,S),1)
      creer(paire-critique(R,x,y),nil);

   former-bloc-avant(R,y,Z) ->
      liste-triee-des(z,PRECEDE(z.nil,y.nil).ou(paire-critique(R,z,y),
      paire-critique(R,y,z)),Z);

   former-bloc-apres(R,Z,x) ->
      liste-triee-des(z,PRECEDE(x.nil,z.nil).ou(paire-critique(R,z,x),
      paire-critique(R,x,z)),Z);

   "Regles d'actualisation de dates limites"

   actual-deb(bloc,X,y) ->
      FIN-PLUS-TOT-BLOC(X,f)
      si(rule(DEBUT-PLUS-TOT(y,C),nil),val(inf(C,f),1).suppress(1))
      assert(DEBUT-PLUS-TOT(y,f),nil)
      outl(DEBUT-PLUS-TOT(y,f));
   actual-deb(faisceau,X,y) ->
      calcul(min,<FIN-PLUS-TOT>,X,f)
      si(rule(DEBUT-PLUS-TOT(y,C),nil),val(inf(C,f),1).suppress(1))
      assert(DEBUT-PLUS-TOT(y,f),nil)
      outl(DEBUT-PLUS-TOT(y,f));

   actual-fin(bloc,x,Y) ->
      DEBUT-PLUS-TARD-BLOC(Y,c)
      si(rule(FIN-PLUS-TARD(x,F),nil),val(inf(c,F),1).suppress(1))
      assert(FIN-PLUS-TARD(x,c),nil)
      outl(FIN-PLUS-TARD(x,c));
```

```
actual-fin(faisceau,x,Y) ->
   calcul(max,<DEBUT-PLUS-TARD>,Y,c)
   si(rule(FIN-PLUS-TARD(x,F),nil),val(inf(c,F),1).suppress(1))
   assert(FIN-PLUS-TARD(x,c),nil)
   outl(FIN-PLUS-TARD(x,c));

DEBUT-PLUS-TARD-BLOC(x.nil,c) -> DEBUT-PLUS-TARD(x,c) /;
DEBUT-PLUS-TARD-BLOC(L,c) ->
   ordon(max,<FIN-PLUS-TARD>,L,S)
   calcul-BLOC(min.sub.DEBUT-PLUS-TARD,S,c);

FIN-PLUS-TOT-BLOC(x.nil,f) -> FIN-PLUS-TOT(x,f) /;
FIN-PLUS-TOT-BLOC(L,f) ->
   ordon(min,<DEBUT-PLUS-TOT>,L,S)
   calcul-BLOC(max.add.FIN-PLUS-TOT,S,f);


"~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~"
" REPRISE DE L'ACTUALISATION DE DATES LIMITES "
"~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~"

precede-actualisant(x,o,y.nil) ->
   assign(reponse,0)
   impasse;
precede-actualisant(x,"avant",y.nil) ->
   type-actual-S(x.nil,y,A)
   A
   creer(SOURCE(y),nil)
   assign(reponse,1)
   impasse;
precede-actualisant(x,"avant",y.nil) ->
   type-actual-P(x,y.nil,A)
   A
   creer(PUITS(x),nil)
   assign(reponse,1)
   impasse;
precede-actualisant(x,"avant",y.nil) ->
   /
   val(reponse,1);
precede-actualisant(x,"apres",y.nil) ->
   /
   precede-actualisant(y,"avant",x.nil);
precede-actualisant(x,"avant",L) ->
   actual-fin(faisceau,x,L)
   creer(PUITS(x),nil);
precede-actualisant(x,"apres",L) ->
   actual-deb(faisceau,L,x)
   creer(SOURCE(x),nil);


"~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~"
"              FONCTIONS DIVERSES             "
"~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~"
"Calcul de dates limites"

DEBUT-PLUS-TARD(x,c) ->
   FIN-PLUS-TARD(x,F)
   DURE(x,D)
   val(sub(F,D),c);

FIN-PLUS-TOT(x,f) ->
   DEBUT-PLUS-TOT(x,C)
   DURE(x,D)
   somme(C,D,f);
```

```
"Utilisation detaillee des ressources"

utilise(R,x,q) -> UTILISE(x,L) dans(<R,q>,L);

"Consommation obligatoire temporelle sur [Ci,Fi]"

CONSO(i,j,c-onso) ->
   DEBUT-PLUS-TOT(i,C)
   FIN-PLUS-TARD(i,F)
   DEBUT-PLUS-TARD(j,c)
   FIN-PLUS-TOT(j,f)
   DURE(j,D)
   val(sub(f,C),m1)
   val(sub(F,c),m2)
   val(sub(F,C),m3)
   min(m1,m2,m')
   min(m',m3,m'')
   min(m'',D,m''')
   max(+0.0,m''',c-onso);

"Test de coherence : l'intervalle [Deb,Fin] contient la duree"

test-coherence(x) ->
   DEBUT-PLUS-TOT(x,C)
   FIN-PLUS-TARD(x,F)
   DURE(x,D)
   val(inf(sub(F,C),D),1)
   /
   line
   outml("                        FIN ANTICIPEE...")
   outm("                         contraintes trop fortes sur ")
   out(x)
   line
   block-exit(echec);
test-coherence(x) ->;

"Fonctions de calcul"

calcul(f,<p>,nil,t) -> /;
calcul(f,<p>,x.nil,t) -> <p,x,t> /;
calcul(f,<p>,x.L,t) -> <p,x,t1> calcul(f,<p>,L,t2) <f,t1,t2,t>;
calcul(f,<p,a,b>,nil,nil) -> /;
calcul(f,<p,a,b>,x.L,t) ->
   <p,a,b,x,t1>
   calcul(f,<p,a,b>,L,t2)
   <f,t1,t2,t>;
calcul(f,<p,r>,x.nil,t) -> <p,r,x,t> /;
calcul(f,<p,r>,x.L,t) -> <p,r,x,t1> calcul(f,<p,r>,L,t2) <f,t1,t2,t>;

calcul-BLOC(f.o.p,x.nil,t) -> <p,x,t> /;
calcul-BLOC(f.o.p,x.L,t) ->
   DURE(x,P)
   calcul-BLOC(f.o.p,L,t1)
   val(<o,t1,P>,t2)
   <p,x,t3>
   <f,t2,t3,t>;


"~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~"
"            REGLES DE CLASSEMENT                     "
"~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~"

ordon(f,o,x.L,S) ->
```

```
   ordon(f,o,L,L1)
   ranger(f,o,x,L1,S);
ordon(f,o,nil,nil) ->;

ranger(f,<p>,x,y.L,x.y.L) -> <p,x,a> <p,y,b> <f,a,b,b> /;
ranger(f,<p,R>,x,y.L,x.y.L) -> <p,R,x,a> <p,R,y,b> <f,a,b,b> /;
ranger(f,o,x,y.l,y.L) -> ranger(f,o,x,l,L);
ranger(f,o,x,nil,x.nil) ->;

creer(T,Q) -> rule(T,Q) /;
creer(T,Q) -> assert(T,Q);


"~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~"
"            RESULTATS DE REPORT                            "
"~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~"
"Sortie de la nouvelle base de faits"

sortie ->
   si(rule(norme-affichage(m,M,N),nil),suppress(1))
   reel-entier
   impasse;
sortie ->
   save;

"Regle d'affichage"

affichage ->
   liste-des(t1,DEBUT-PLUS-TOT(t1,C),T1)
   liste-des(t2,FIN-PLUS-TARD(t2,F),T2)
   calcul(min,<DEBUT-PLUS-TOT>,T1,c)
   calcul(max,<FIN-PLUS-TARD>,T2,f)
   reel(c,m)
   reel(f,M)
   val(sub(M,m),i)
   val(div(+58.0,i),N)
   assert(norme-affichage(m,M,N),nil);

"Cartouche du diagramme a barres"

banniere(B) ->
   norme-affichage(m,M,N)
   line
   set-line-cursor(0)
   outm("~",79)
   line
   set-line-cursor(0)
   outm(" T | Duree | Dates limites [ ]")
   intitule(B)
   line
   set-line-cursor(0)
   outm("~",79)
   line
   set-line-cursor(11)
   out(m)
   set-line-cursor(16)
   outm("[")
   outm("|",56)
   outm("]")
   out(M)
   line
   outm(" ");

intitule(R) ->
```

```
   DISPONIBILITE(R,Q)
   set-line-cursor(36)
   outm("Diagramme de GANTT : RESSOURCE ")
   out(R);
intitule(T) ->
   TRAVAIL(T,L)
   set-line-cursor(36)
   outm("Diagramme de GANTT : TRAVAIL ")
   out(T);
intitule(t) ->
   DURE(t,P)
   set-line-cursor(36)
   outm("Diagramme de GANTT : TACHE ")
   out(t);

"Appel du diagramme a barres"

GANTT ->
   outml("       Diagramme de Gantt par ressource(r) ou par travail(t) ?")
   outm("       votre reponse (r/t) : ")
   in-char(c)
   line
   si-alors-sinon(ou(eq(c,"r"),eq(c,"t")),outml("       OK"),outml("    !!??"))
   gantt(c);

"Sauvegarde du diagramme sur un fichier"

GANTT-SUR(f-ichier) ->
   string(f-ichier)
   outml("      Diagramme de Gantt par ressource(r) ou par travail(t) ?")
   outm("       votre reponse (r/t) : ")
   in-char(c)
   line
   si-alors-sinon(ou(eq(c,"r"),eq(c,"t")),outml("       OK"),outml("    !!??"))
   gantt-sur(c,"console",f-ichier);

"Diagramme : "
"1. par ressource"
"2. par travail"

gantt("r") ->
   DISPONIBILITE(R,Q)
   banniere(R)
   utilise(R,t,q)
   affich(t)
   impasse;
gantt("t") ->
   TRAVAIL(T,L)
   banniere(T)
   dans(t,L)
   affich(t)
   impasse;
gantt(x) ->;

gantt-sur(x,o,f-ichier) -> output(f-ichier) gantt(x);
gantt-sur(x,o,f-ichier) -> output(o) close-output;

affich(x) ->
   line
   set-line-cursor(0)
   out(x)
   DURE(x,P)
   reel(P,P1)
   set-line-cursor(5)
```

```
   out(P1)
   norme-affichage(m,M,N)
   affich-debut(x,m,N)
   val(trunc(mul(P1,N)),P2)
   val(if(inf(P2,1),0,sub(P2,1)),P3)
   outm("*",P3)
   affich-fin(x,m,N);

affich-debut(x,m,N) ->
   DEBUT-PLUS-TOT(x,C)
   /
   reel(C,c)
   val(sub(c,m),c1)
   val(mul(c1,N),c2)
   entier(c2,e1)
   val(add(16,e1),e2)
   val(sub(e2,5),e3)
   set-line-cursor(e3)
   out(c)
   set-line-cursor(e2)
   outm("[");
affich-debut(x,m,N) ->
   set-line-cursor(11)
   outm("  ?  ");

affich-fin(x,m,N) ->
   FIN-PLUS-TARD(x,F)
   /
   reel(F,f)
   val(sub(f,m),f1)
   val(mul(f1,N),f2)
   entier(f2,e1)
   val(add(e1,15),e2)
   set-line-cursor(e2)
   outm("]")
   out(f);
affich-fin(x,m,N) ->
   outm("  ?");


"~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~"
"                    OUTILS                    "
"~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~"

" CALCULS NUMERIQUES "

min(x,y,z) -> val(if(inf(x,y),x,y),z);

max(x,y,z) -> val(if(inf(y,x),x,y),z);

somme(x,y,z) -> val(add(x,y),z);

produit(x,y,z) -> val(mul(x,y),z);

"Conversions entiers-reels, reels-entiers."

reel(e,e) -> real(e) /;
reel(e,r) -> val(float(e),r);

entier(e,e) -> integer(e) /;
entier(r,e) -> val(if(inf(r,+0.0),mul(r,-1.0),r),a) val(trunc(a),e);

"Choix d'un element x dans une liste L"
```

```
dans(x,x.L) ->;
dans(x,y.L) -> dans(x,L);

"Presence d'un element x dans une liste L"

dans'(x,x.L) -> /;
dans'(x,y.L) -> dans'(x,L);

"Choix d'un element x dans L, R etant le reste a droite de x"

elem(x,x.R,R) ->;
elem(x,y.L,R) -> elem(x,L,R);

"Choix d'un element x dans L, R etant le complement de x dans L"

elem'(x,x.L,L) -> /;
elem'(x,y.L,y.l) -> elem'(x,L,l);

"Verifier qu'une liste l est une sous-liste d'une liste L"

sous-liste(nil,L) ->;
sous-liste(x.L1,L2) -> elem'(x,L2,l2) / sous-liste(L1,l2);

"Concatenation de deux listes"

conca(nil,L,L) ->;
conca(x.L1,L2,x.L3) -> conca(L1,L2,L3);

"Combinaison de n elements choisis dans une liste"

nuple(0,L,nil) -> /;
nuple(n,L,x.X) ->
   arg(0,L,N)
   val(inf(N,n),0)
   val(sub(n,1),r)
   elem(x,L,l)
   nuple(r,l,X);

"Negation definie comme l'echec de la demonstration"

non(P) -> P / impasse;
non(P) ->;

"Sans commentaires ..."

et(p,q) -> p q;

ou(p,q) -> p;
ou(p,q) -> q;

si-alors-sinon(H,C1,C2) -> H / C1;
si-alors-sinon(H,C1,C2) -> C2;

si(p,q) -> p / q;
si(p,q) ->;

vaut(p,vrai) -> p /;
vaut(p,faux) ->;

vrai ->;

faux -> impasse;

"Liste L (triee ou non )des valeurs possibles prises par  "
```

```
"une variable x lors de l'effacement d'une suite de buts P"
"La suite des buts doit etre ecrite en notation 'nuplet', "
"exemple : P = <dans,x,L>.<dif,x,y> "

liste-triee-des(x,P,L) -> liste-des(x,P,L1) sans-repet(L1,L);

liste-des(x,P,L) -> output-is(o) new-buffer(liste-des'(x,P,L,o));

liste-des'(x,P,L,o) ->
   P
   output("buffer")
   outm("(")
   out(x)
   outm(").")
   impasse;
liste-des'(x,P,L,o) ->
   output("buffer")
   outml("nil;")
   output(o)
   input-is(i)
   input("buffer")
   in(L)
   input(i);

sans-repet(nil,nil) ->;
sans-repet(x.X,x.L) -> Purge(X,x,L1) sans-repet(L1,L);

Purge(nil,x,nil) ->;
Purge(x.L,x,l) -> Purge(L,x,l);
Purge(y.L,x,y.l) -> dif(y,x) Purge(L,x,l);

som-list(l,nil,l) -> /;
som-list(nil,l,l) -> /;
som-list(x.L1,y.L2,z.l) ->
   somme(x,y,z)
   som-list(L1,L2,l);

retranche(l,nil,l) -> /;
retranche(nil,l,l) -> /;
retranche(x.l1,l2,l) ->
   dans'(x,l2)
   elem'(x,l2,l3)
   retranche(l1,l3,l);
retranche(x.l1,l2,x.l) ->
   non(dans'(x,l2))
   retranche(l1,l2,l);

le-plus-grand(nil,+0.0) -> /;
le-plus-grand(u.nil,u) -> /;
le-plus-grand(u.v.L,x) ->
   max(u,v,z)
   le-plus-grand(z.L,x);

le-plus-petit(nil,+0.0) -> /;
le-plus-petit(u.nil,u) -> /;
le-plus-petit(u.v.L,x) ->
   min(u,v,z)
   le-plus-petit(z.L,x);

ordonner(nil,i,i) ->;
ordonner(x.l,i,o) ->
   decouper(x,l,u,v)
   ordonner(u,i,r)
   ordonner(v,x.r,o);
```

```
ordonner2(nil,i,i) ->;
ordonner2(x.l,i,o) ->
   decouper(x,l,u,v)
   ordonner2(v,i,r)
   ordonner2(u,x.r,o);

decouper(x,nil,nil,nil) -> /;
decouper(x,y.l,y.u,v) ->
   inferieur(y,x)
   /
   decouper(x,l,u,v);
decouper(x,y.l,u,y.v) ->
   decouper(x,l,u,v);

inferieur(x,y) ->
   val(inf(x,y),1);

superieur-ou-egal(x,x) ->;
superieur-ou-egal(x,y) ->
   val(inf(x,y),0);

;End world: Regles
```

Thèse de Pierre LOPEZ

"Approche énergétique pour l'ordonnancement de tâches sous contraintes de temps et de ressources"

RESUME. Ce travail concerne l'ordonnancement de tâches interdépendantes sous contraintes de temps et de ressources. Les méthodes et techniques développées s'inscrivent dans la problématique de "l'Analyse Sous Contraintes" des problèmes d'ordonnancement. Celle-ci vise à caractériser les ordonnancements admissibles de manière à proposer au décideur un choix d'actions cohérentes vis-à-vis des contraintes. L'analyse est décrite comme un processus d'inférence mettant en interaction une base de règles et une base de faits temporels et séquentiels représentant les caractéristiques des ordonnancements admissibles. Des travaux antérieurs ont ainsi permis la réalisation du logiciel Mascot écrit en Prolog.
Ils sont ici enrichis par une nouvelle approche dont l'originalité réside essentiellement dans l'utilisation du concept d'énergie issu du couplage du temps et des ressources. Le concept d'intervalle temps-ressource est introduit ; il permet de représenter simultanément des caractéristiques temporelles et de ressource. On distingue les intervalles consommateurs (ou tâches) et les intervalles fournisseurs.
Le type de déduction mis en jeu dans Mascot a été amélioré par la prise en compte des interactions entre intervalles consommateurs et fournisseurs. De nouvelles règles de déduction ont été écrites et intégrées dans Mascot, donnant lieu au logiciel Mascot2.
D'autre part, un processus de déduction, basé sur un raisonnement purement énergétique, a été élaboré et implémenté en Prolog (logiciel Report). Il met en jeu des instants remarquables, points de cassure des courbes d'énergie associées aux tâches.
L'outil de modélisation utilisé est le graphe potentiels-bornes ; il permet de représenter des contraintes numériques et des contraintes symboliques entre intervalles. Il sert de support à un processus d'inférence par propagation numérique des contraintes.

MOTS-CLES. Ordonnancement de tâches, analyse sous contraintes, ordonnancements admissibles, raisonnement temporel, raisonnement énergétique.

"Energy-based approach for task scheduling under time and resource constraints"

ABSTRACT. This work concerns scheduling independent tasks under time and resource constraints. The methods and technics that have been developped refer to the so-called "Constraints-Based Analysis" in scheduling problems. This analysis aims to characterize admissible schedules in order to provide the decision maker with a set of coherent actions in relation to constraints. The analysis is described as an inference process between a basis of rules and a basis of temporal and sequential facts representing the characteristics of admissible schedules. Previous works have so forth allowed the realization of the Mascot software written in Prolog.
Here a new approach is presented and a concept of energy is introduced by combining time and resources. Also a time-resource interval is introduced so that simultaneous representation of temporal and resource characteristics is possible. We distinguish the consumer intervals (tasks) and the supplier intervals.
The deduction used in Mascot has been improved by taking into account the interactions between consumer and supplier intervals. New rules of deduction have been written and integrated in Mascot yielding a new Mascot2 software.
Also a deduction process based only on the energy concept, has been elaborated and implemented in Prolog (Report software). It involves remarkable times, break-points of energy curves associated with the tasks.
The bound-potential graph is used as a model : it allows to represent constraints between intervals. It constitutes a support to an inference process by propagation of numerical constraints.

KEY-WORDS. Task scheduling, constraints-based analysis, admissible schedules, temporal reasoning, energy-based reasoning.