Contributions à la validation d'ordonnancement temps réel en présence de transactions sous priorités fixes et EDF

Ahmed Rahni

Ecole Nationale Supérieure de Mécanique et d'Aérotechnique
Ecole Doctorale des Sciences Pour l'Ingénieur et Aéronautique ▪ Université de Poitiers

THESE

Pour l'obtention du grade de

DOCTEUR DE L'UNIVERSITÉ DE POITIERS

Spécialité : INFORMATIQUE

Présentée et soutenue publiquement par

Ahmed RAHNI

Le 05 décembre 2008

****

Contributions à la validation d'ordonnancement temps réel en présence de transactions sous priorités fixes et EDF

****

Directeurs de Thèse : ***Pascal RICHARD, Emmanuel GROLLEAU et Michaël RICHARD***

JURY

Rapporteurs :	Ye-Qiong SONG	Professeur, INPL/ENSEM, LORIA
	Maryline CHETTO	Professeur, IUT, IRCCyN
Examinateurs:	Yvon TRINQUET	Professeur, IUT, IRCCyN
	Francis COTTET	Professeur, ENSMA, LISI
	Joël GOOSSENS	Professeur associé, Université Libre de Bruxelles
	Pascal RICHARD	Professeur Université de Poitiers, LISI/ENSMA
	Emmanuel GROLLEAU	Maître de conférences, ENSMA, LISI
	Michaël RICHARD	Maître de conférences, ENSMA, LISI

LABORATOIRE D'INFORMATIQUE SCIENTIFIQUE ET INDUSTRIELLE
Ecole Nationale Supérieure de Mécanique et d'Aérotechnique
Téléport2- 1, avenue Clément Ader -BP 40109 - 86961 Chasseneuil-Futuroscope Cédex
Tél: 05.49.49.80.63 - Fax: 05.49.49.80.64

Remerciements

Je tiens tout d'abord, à exprimer mes vifs remerciements et ma grande reconnaissance à Emmanuel GROLLEAU pour l'encadrement dont il m'a fait bénéficier, sa disponibilité et son suivi tout au long de ma thèse. Ses très nombreux conseils, ainsi que les discussions que nous avons eues m'ont été d'une aide inestimable et ont été déterminants pour ma formation de chercheur.

Mes vifs remerciements vont également à Michaël RICHARD pour son encadrement, ses encouragements pour faire avancer les travaux durant toute la thèse, et plus particulièrement durant la première année. Pour cette première année, merci à Patrick GIRARD. Cette période à été d'une grande utilité pour ma formation dans la recherche.

Je tiens également, à remercier mon directeur de thèse, le Professeur Pascal RICHARD, de m'avoir accueilli au sein de son équipe de recherche, de son apport et surtout pour sa relecture très attentive de mon manuscrit qui m'a permis d'en améliorer sensiblement la qualité scientifique et pédagogique.

Je voudrais aussi remercier les Professeurs Guy PIERRA (ancien directeur du LISI) et Yamine AIT-AMEUR, l'actuel directeur du LISI, pour les moyens techniques et scientifiques qu'ils ont mis à ma disposition.

Merci aux Professeurs Maryline CHETTO et Ye-Qiong SONG qui ont accepté la lourde charge de rapporteurs, ainsi qu'aux Professeurs Yvon TRINQUET, Francis COTTET et Joël GOOSSENS pour l'honneur qu'ils m'ont fait en acceptant d'être examinateurs de mon jury et de l'importance qu'ils ont accordée à mon travail.

Un remerciement particulier à Ladjel et Mokhtar pour tout ce qu'ils ont fait pour m'aider pendant ces trois années.

Et puis, je remercie tous les membres du LISI plus particulièrement, Claudine, Fred, Laurent, Loé, Idir, Nabil, Stéphane, Zoom, Karim, Sybille, Dilek, Chimène, Kamel, Hondjack, François, Malou, Christian, Hieu,Valery, Youcef, Dago, Eric, Jean-Claude, Tex, pour tous les bons moments passés en leur compagnie. Merci à l'équipe de football du LISI.

Je souhaite également remercier ma famille, ainsi que tous mes amis.

Table des matières

Table des figures

Liste des tableaux

Introduction générale

L'informatique temps réel est largement utilisée et prend une importance de plus en plus grande dans la vie quotidienne. Plusieurs domaines d'application sont concernés, notamment, l'automobile, l'avionique, la téléphonie mobile, le multimédia, l'énergie, ...etc. Les systèmes temps réel concernent des applications informatiques ayant un rôle de suivi ou de contrôle de procédé dans un environnement qui évolue dynamiquement. L'application est réactive, en effet, elle doit réagir au changement d'état du système contrôlé dans des contraintes temporelles précises. D'où le bon fonctionnement d'un système temps réel ne dépend pas seulement de l'exactitude des résultats du calcul mais aussi des dates auxquelles ces résultats sont produits. Certains systèmes sont classés critiques lorsque le non respect des contraintes temporelles peut provoquer des conséquences catastrophiques (perte de vies humaines, destruction de matériel, ...). Les systèmes de contrôle de vol, systèmes de contrôle de centrale nucléaire en sont des exemples.

Généralement, un système temps réel est composé d'un ensemble de tâches exécutées en concurrence pour utiliser le (ou les) processeur et certaines ressources partagées. Le plus souvent les exigences temporelles sont reportées sur les échéances d'exécution des tâches. Par conséquent, une validation temporelle consiste de vérifier que toutes les tâches d'un système terminent leur exécution avant leurs échéances, durant toute la vie du système, et cela en examinant le pire scénario d'exécution de tâches, s'il existe un moyen de la garantir alors le système est dit ordonnançable. Le test d'ordonnançabilité est dépendant du modèle de tâches qui définit l'ensemble des restrictions auxquelles doivent se conformer les tâches et de l'algorithme d'ordonnancement utilisé pour déterminer l'ordre dans lequel doivent être exécutées les tâches. L'un des principaux problèmes du test d'ordonnançabilité est l'identification du pire scénario d'exécution des tâches et la complexité de l'analyse.

L'objectif de cette thèse est d'introduire de nouveaux tests d'ordonnançabilité et d'optimiser leur qualité de réponse et leur temps de calcul pour le modèle de tâches avec décalages d'activations (appelé transactions), dans les deux contextes d'ordonnancement : en priorités fixes et en priorités dynamiques EDF.

Dans le chapitre 1, nous exposons le problème de l'ordonnancement, après avoir présenté brièvement l'architecture logicielle et matérielle des systèmes temps réel et les différents facteurs définissant le contexte d'ordonnancement de tâches. Puis nous présentons les différentes techniques d'analyse d'ordonnançabilité. Les résultats d'analyse d'ordonnançabilité des tâches périodiques et sporadiques, basées sur le modèle de Liu et Layland, seront présentés. Ces résultats seront

classés par technique d'analyse (analyse de facteur d'utilisation, de temps de réponse ou de la demande processeur) et par algorithme d'ordonnancement (priorités fixes ou priorités dynamiques).

Les conditions d'ordonnançabilité présentées dans le chapitre 1 sont pessimistes pour certains types d'application car le modèle de tâches considéré ne modélise pas finement leur comportement. Plusieurs modèles de tâches sont spécifiques aux types d'applications où un décalage d'activation entre les tâches est prise en considération. Nous commençons, dans le chapitre 2, par présenter le modèle de tâches à offsets (transactions temps réel), puis nous présentons le test d'ordonnançabilité par l'analyse de temps de réponse. En priorités dynamiques, comme en priorités fixes, l'analyse exacte a une complexité exponentielle due au nombre de scénarios à étudier, et seulement des méthodes d'analyse approchées (et donc pessimistes) avec une complexité pseudo-polynomiale sont proposées. D'où la problématique du test d'ordonnançabilité des systèmes de transactions. Les mêmes problèmes sont recensés dans l'analyse des tâches multiframes qui est présentée à la fin du chapitre 2.

Dans la partie contribution, nous montrons, dans un premier temps, que les transactions sont le modèle le plus général dans tous les modèles présentés précédemment (en particulier les tâches multiframes) i.e. tout résultat valable pour les transactions est valable pour les tâches multiframes. Le chapitre 3 présente une nouvelle méthode d'analyse mixte de temps de réponse, avec une complexité pseudo-polynomiale, des tâches à offset avec priorités fixes. Notre idée est de combiner l'analyse exacte d'interférence pour certaines transactions avec l'analyse approchée pour les autres transactions. Cette méthode fournit un temps de réponse meilleur (mais pessimiste) que celui fourni par les méthodes d'analyse approchées. Comme nous montrons dans l'évaluation de performance, plus le nombre de transactions pour lesquelles une analyse exacte d'interférence est appliquée, meilleure est la qualité de la réponse, mais cet avantage est limité par le temps de calcul. Cependant, la méthode mixte avec une analyse exacte d'interférence pour deux transactions fournit le meilleur compromis entre qualité et temps de calcul.

Le chapitre 4 focalise sur le problème d'identification de pire scénario, en priorités fixes. Un instant critique (pire scénario) se produit lorsqu'une tâche de chaque transaction s'active simultanément. Malheureusement, nous ne savons pas quelle tâche candidate dans chaque transaction initie l'instant critique, d'où la nécessité d'étudier toutes les combinaisons possibles entre les tâches des transactions. Dans un premier temps nous étudions et montrons les limites des transactions monotoniques (sans gigue). Ce sont des transactions respectant une structure particulière, où la tâche initiant l'instant critique est connue. Ensuite, nous définissons une propriété d'accumulativité monotonique permettant d'identifier toutes les transactions (avec et sans gigues) pour lesquelles la tâche initiant l'instant critique pourra être connue. Cette propriété, comme nous le montrons dans l'évaluation de performances, minimise significativement le nombre de scénarios à étudier ainsi que le temps de calcul, et fournit des temps de réponse exacts lorsque l'accumulativité monotonique est satisfaite par toutes les transactions du système. Nous proposons, dans le cas des transactions non accumulativement monotoniques, une propriété, celle de dominance de tâche, permettant d'éviter l'étude des scénarios définis par des tâches dominées et ainsi améliorer le temps de calcul.

Le chapitre 5 est consacré à l'analyse d'ordonnançabilité des transactions ordonnancées par EDF. Seule un test exact avec une complexité exponentielle était connu. Les tests pseudo-polynomiaux étaient approchés. Ces résultats sont basés sur l'analyse de temps de réponse. Nous proposons dans ce chapitre un test d'ordonnançabilité exact avec une complexité pseudo-polynomiale, ce test est basé sur l'analyse de la demande processeur. Le principe est de prouver que pour tout intervalle de temps de longueur t, la demande processeur du système est inférieure ou égale à t.

Finalement, quelques conclusions autour des résultats majeurs présentés dans ce document et des perspectives sont données.

ÉTAT DE L'ART

Chapitre 1

Systèmes Temps Réel et Ordonnancement

Résumé : *Dans ce chapitre , nous présentons un état de l'art de l'ordonnancement temps réel. Après une brève présentation des systèmes temps réel, le problème d'ordonnancement est exposé. Nous présentons ensuite les concepts et les facteurs interférant dans le test d'ordonnançabilité. Nous focalisons sur l'ordonnancement en-ligne pour des systèmes monoprocesseurs. Les différents résultats d'analyse, en utilisant différentes techniques analytiques, des systèmes de tâches périodiques et sporadiques basés sur le modèle de Liu et Layland sont présentés.*

1.1 Introduction aux systèmes Temps-Réel

Les systèmes temps réels sont de plus en plus présents dans le quotidien, on les trouve dans l'aéronautique, transport ferroviaire, l'automobile, l'électroménager ou le multimédia. On désigne du temps réel, toute application mettant en œuvre un système informatique dont le comportement (fonctionnement) est conditionné par l'évolution dynamique de l'état d'un environnement (appelé procédé) qui lui est connecté et dont il doit contrôler le comportement [36]. Le rôle du système informatique est alors de suivre ou de piloter ce procédé en respectant des contraintes temporelles définies dans le cahier des charges de l'application. Également, dans [104], on qualifie du temps réel tout système informatique dont la correction du fonctionnement ne dépend pas seulement de l'exactitude logique des résultats qu'il fournit, mais surtout dépend de la date à laquelle ces résultats sont produits. Ainsi, le système ne traite plus uniquement des valeurs, mais des couples valeur et temps [33]. Cette définition implique que la seule rapidité moyenne d'exécution du logiciel ne conditionne pas la validité du système, mais des contraintes temporelles doivent être respectées (par exemples les échéances des traitements) sont relatives à un temps physique mesurable, et font partie de la spécification du système à implanter.

Dans les systèmes temps réel, le système informatique doit réagir en permanence aux variations du procédé et agir en conséquence sur celui-ci afin d'obtenir le comportement ou l'état souhaité, cette caractéristique définit la notion de réactivité du système informatique vis-à-vis du procédé (environnement auquel il est connecté) [44]. Une définition des systèmes réactifs, décrivant le fonctionnement d'un système temps réel (l'interaction entre le système informatique et l'environnement), est donnée dans [39] : *Un système réactif est un système qui réagit continuement avec son environnement à un rythme imposé par cet environnement. Il reçoit, par l'intermédiaire de capteurs, des entrées provenant de l'environnement, appelées stimuli, réagit à tous ces stimuli en effectuant un certain nombre d'opérations et produit, grâce à des actionneurs, des sorties utilisables par l'environnement, appelées réactions ou commandes.*

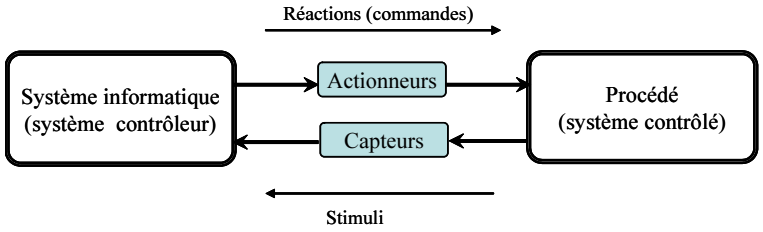

Fig. 1.1 – Système temps réel.

De cette définition, un système temps réel est alors composé principalement de deux éléments distincts : une ou plusieurs entités physiques constituent le procédé (le système contrôlé) et un système de contrôle (contrôleur) qui est chargé de surveiller de manière régulière, périodique, l'état du procédé en récupérant les valeurs en provenance des capteurs. En fonction de l'algorithme de contrôle et les valeurs décrivant l'état actuel du procédé, le contrôleur commande les changements d'état, via des actionneurs. Notons qu'un système temps réel est dite **embarqué**

lorsqu'il est situé à l'intérieur du l'environnement qu'il doit contrôler, comme un calculateur dans une voiture ou un avion.

1.2 Caractéristiques des systèmes temps réel

Dans le contexte des systèmes temps réel, les données ont une validité définie à la fois par le domaine de valeurs admises et par la durée de validité (temporelle) de celles-ci, qui dépend naturellement de l'échéance. Les données ont donc une durée d'existence limitée [117]. Autrement, les systèmes de contrôle doivent respecter deux exigences fonctionnelles et temporelles en parallèle. Selon les contraintes temporelles, on distingue deux grandes catégories des systèmes temps réel :

- ***Systèmes temps réel à contraintes strictes (temps réel dur) :*** lorsque toutes les contraintes temporelles doivent être impérativement respectées. Le non respect des contraintes peut provoquer des conséquences catastrophiques [116]. Les systèmes de contrôle de vol, systèmes de contrôle de station nucléaire, systèmes de contrôle de voies ferrées en sont des exemples.
- ***Systèmes temps réel à contraintes relatives (temps réel souple) :*** au contraire des systèmes durs, le non respect des contraintes temporelles est toléré (acceptable) par le système, et sans que cela ait des conséquences catastrophiques [24, 23]. Par exemple des applications multimédias.
- ***Systèmes temps réel à contraintes mixtes :*** sont composés des tâches à contraintes strictes et des tâches à contraintes relatives [41].

D'autre part, les systèmes temps réel sont considérablement influencés par l'architecture matérielle sur laquelle doit s'exécuter le système informatique. Cette architecture matérielle désigne les ressources physiques nécessaires au pilotage du procédé : les processeurs, les cartes d'entrées/sorties, les mémoires, les réseaux, etc. En fonction de nombre de processeurs et de l'utilisation éventuelle d'un réseau, on distingue trois grandes catégories d'architecture [88] :

- ***Architecture monoprocesseur :*** Dans ce type de systèmes, toutes les fonctions (tâches) de l'application sont exécutées par un unique processeur partagé.
- ***Architecture multiprocesseur :*** Les fonctions de l'application sont réparties sur n processeurs partageant une mémoire centrale commune.
- ***Architecture distribuée :*** Les fonctions de l'application sont réparties sur n processeurs, mais au contraire des architectures multiprocesseurs, il n'y a pas de mémoire centrale partagée. Les processeurs sont reliés les uns aux autres par l'intermédiaire d'un réseau. Réseaux de terrain comme CAN (Controller Area Network) [1, 2].

Dans la suite nous nous intéressons aux systèmes de contrôle à **contraintes strictes** implantés dans une **architecture monoprocesseur**. Nous nous focalisons principalement sur l'architecture logiciel du système informatique, et d'une façon plus détaillée sur son rôle d'ordonnançement des tâches (fonctions) de l'application de contrôle. Deux principaux composants constituent l'architecture logicielle d'un système temps réel, l'un de bas niveau représenté par l'exécutif temps réel ou OS temps réel (RTOS : Real Time Opérating System) et l'autre de plus haut niveau représenté par l'application informatique de contrôle.

- ***L'exécutif temps réel :*** désigne le système d'exploitation adapté au contexte particulier, par ses exigences temporelles, de l'informatique qualifiée de temps réel. Il possède un noyau qui en interaction directe avec la partie physique de la machine. Le rôle principal de l'exécutif temps réel est d'ordonnancer les exécutions des tâches de l'application de contrôle (**sujet d'étude de cette thèse**), et aussi celui de protéger l'accès aux ressources partagées. il joue donc un rôle centralisateur, un véritable rôle d'interface qui aiguille les événements reçus du procédé vers les tâches qui les attendent, déclenche le réveil des tâches en attente d'un délai ou d'une heure de démarrage, reçoit et transmet des signaux de synchronisation ou des données entre des tâches asynchrones [115, 20, 94]. Nous citons quelques exécutifs les plus connus : VxWorks, pSOS, Lynx-OS, POSIX Temps réel, et OSEK/VDX [4].
- ***L'application de contrôle du procédé :*** désigne le programme informatique permettant de réaliser les différentes fonctions nécessaires au contrôle du procédé. Ce programme est composé d'un ensemble d'entités de programmes appelées tâches temps réel. Chacune des tâches assure un ou plusieurs fonctions. Les tâches font appel, lors de leur exécution, aux routines proposées par l'exécutif temps réel afin d'accéder au capteurs et actionneurs, ou de communiquer entre elles.

1.3 Caractéristiques des tâches temps réel

Afin de répondre aux exigences temporelles du système temps réel, pour chaque tâche temps réel composant l'application de contrôle, des contraintes temporelles sont définies [55, 64]. Une tâche peut être exécutée une multitude fois durant la vie du système, comme par exemple une tâche qui doit lire régulièrement l'état des capteurs. Nous appelons alors une instance d'une tâche une exécution ou occurrence de celle-ci. Tant que le terme tâche désigne la partie de code informatique résultant de compilation d'un langage de haut niveau qui sera exécuté par le processeur. Parmi les paramètres temporels usuels définissant une tâche, nous citons :

- ***Date d'activation*** r_i ***:*** La date à laquelle la tâche τ_i commence son exécution. i.e. la première instance de τ_i est réveillée à la date r_i. Ce paramètre est appelé aussi date de réveil ou offset. Dans le cas où toutes les tâches d'un système sont initialement réveillées au même instant ($r_1 = r_2 = ... = r_n$), on dit que les tâches sont simultanées (ou elles sont synchrones).
- ***Durée d'exécution*** C_i ***(charge processeur) :*** Le temps processeur requis par chaque instance de τ_i. Généralement, ce paramètre est le pire (borne supérieure) temps d'exécution de cette tâche (Worst-Case Execution Time WCET) sur le processeur auquel elle est affectée. De nombreux travaux ont traité le problème du calcul de WCET de façon à ce qu'il ne soit pas trop surestimé. Notons que certains systèmes temps réel souple considèrent un temps d'exécution moyen, ou minimal. La problématique d'estimation de la valeur de durée d'exécution fait objet de nombreux travaux [99, 79, 72, 78, 46]
- ***Période d'activation*** T_i ***:*** La durée de temps fixe ou minimale entre deux activations de deux instances successives de τ_i. Suivant la nature de la tâche (périodique par exemple) on peut calculer facilement les dates d'activation de τ_i à travers le temps.
- ***Délai critique*** D_i ***:*** Le temps alloué à la tâche pour terminer complètement son exécution. Chaque instance de τ_i doit terminer son exécution avant D_i unités de temps après la date

de son activation. Le dépassement de cette date limite pour l'exécution produit une faute temporelle.

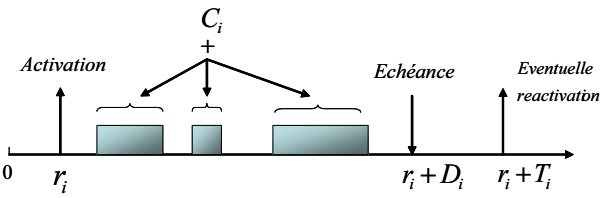

FIG. 1.2 – Modèle usuelle d'une tâche temps réel.

1.3.1 Modes d'exécution

Nous distinguons deux modes d'exécution des tâches temps réel, en fonction de contraintes de ressource et de précédence.

- ***Non-préemptif :*** L'ordonnanceur ne peut pas interrompre l'exécution d'une tâche (possédant le processeur) en faveur d'une autre tâche. Il doit attendre jusqu'à la terminaison de l'exécution de la tâche en cours, avant de débuter l'exécution de toute autre tâche. Si une tâche non-préemptive est interrompue, son exécution doit être reprise de nouveau depuis le début.
- ***Préemptif (interruptible) :*** L'exécution d'une tâche peut être interrompue par une autre tâche plus prioritaire (la priorité d'une tâche est définie, en fonction de l'algorithme d'ordonnancement sous-jacent qui seront présentés plus tard dans ce chapitre). Son exécution est reprise ultérieurement. Notons qu'une tâche demandant une ressource critique, occupée par une tâche moins prioritaire, doit attendre que cette dernière la libère (terminer l'utilisation de la ressource) afin de continuer son exécution.

1.3.2 États d'une tâche

Au cours de la vie de l'application temps réel, plusieurs instances d'une tâches peuvent être exécutées. A un instant donnée, une instance peut être dans l'un des états d'exécutions présentées dans la figure 1.3 qui montre aussi les passages possibles d'un état à un autre [41, 3].
Une tâche (supposée déjà créée) est initialement dans état endormi. Quand elle se réveille, elle passe en état prêt. Dans cette état l'instance de la tâche est activée mais en attente d'être sélectionnée par l'ordonnanceur (le noyau de l'exécutif temps réel) pour l'exécution (dans cet état la tâche requiert le processeur). Lorsque l'ordonnanceur le décide, suivant la politique d'ordonnancement choisie, la tâche se verra allouer le processeur afin d'être exécutée (état Exécution). Une tâche en cours d'exécution peut : (i) être interrompue par une autre tâche plus prioritaire et passe en état prêt ; (ii) se mettre en attente d'un message, d'une date, d'un évènement, ou bien de l'accès à une ressource ; (iii) être suspendue et passer en état endormi. Une tâche en état d'attente d'une condition (arrivée d'un message, d'une date, d'un évènement ou libération d'une ressource) devient prête l'exécution une fois la condition vérifiée.

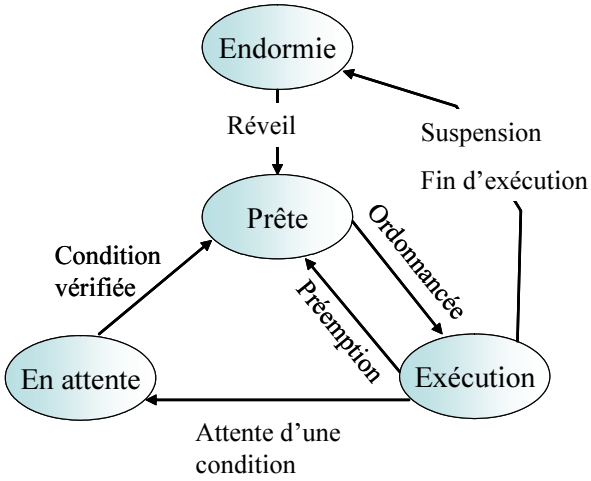

FIG. 1.3 – Différents états possibles d'une tâche temps réel.

1.3.3 Modèles de tâches

La loi d'arrivée d'une tâche définit sa nature. Cette loi s'agit des contraintes temporelles, imposées par le procédé piloté, définissant la répartition des dates d'activation des instances d'une tâche dans le temps. Un modèle de tâches porte le nom de la nature des tâches à laquelle il appartient. On distingue trois grandes classes de tâches :

1.3.3.1 Tâches périodiques

Lorsque la tâche s'active à des intervalles réguliers de temps T_i (période). Il s'agit généralement des tâches de suivi de contrôle du procédé, ou de génération de signaux réguliers. Par exemple la lecture régulière de l'état d'un capteur. La majorité des tâches constituant, généralement, une application temps réel sont des tâches périodiques. Cette classe est basée sur le modèle, qui est très largement utilisé, de Liu et Layland [55] (figure 1.4).

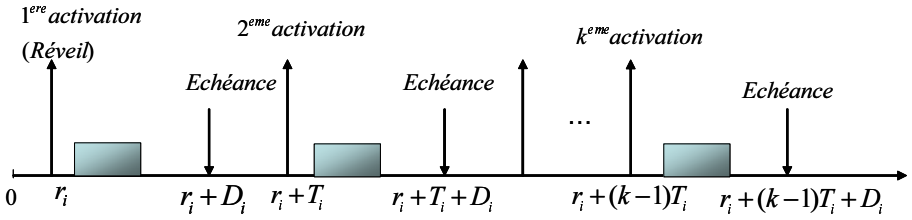

FIG. 1.4 – Modèle de tâches de Liu et Layland.

Une tâche τ_i est caractérisée par une durée d'exécution C_i, une période d'activation T_i, un délai critique D_i et éventuellement la date de la première activation r_i. Une tâche est dite à **échéance sur requête** si $D_i = T_i$. Les tâches d'un système sont à **échéances contraintes** lorsque les échéances sont inférieures ou égales aux périodes. Une tâche est **bien formée** si cette

condition $0 < C_i \leq D_i \leq T_i$ est satisfaite. Si la date d'activation de la premier instance r_i est connue, la tâche est dite **concrète**. Les tâches d'un système sont dites **synchrones** ou **à départ simultané** lorsqu'elles sont toutes concrètes et que leurs dates d'activation sont égales ($r_1 = r_2 = ... = r_n$). Dans le cas où pour chaque pair de tâches τ_i et τ_j du système, la période de l'une est un multiple de la période des autres ; les périodes sont dites **harmoniques**. Lorsque toutes les tâches du système sont synchrones, le motif d'activation des instances se répète à l'identique sur un intervalle de durée d'une hyperpériode (le plus petit commun multiple des périodes des tâches du système, i.e. la plus grande période dans le cas particulier d'un système harmonique). Une tâche est dite **réentrante** s'il est possible que plusieurs de ses instances soient en cours d'exécution simultanément.

1.3.3.2 Tâches sporadiques

Une tâche sporadique τ_i [48, 66] est caractérisée par un loi d'arrivée sporadique, où la période d'activation n'est pas connue a priori, par contre un intervalle minimal de temps T_i séparant deux activations successives est donné. Au contraire des tâches périodiques, les dates d'activation des différentes instances d'une tâche sporadique ne peuvent pas être déterminées a priori. Une instance peut s'activer à n'importe quelle date après la date T_i relative à la date d'activation de l'instance précédente. Une tâche périodique est alors un cas particulier de tâche sporadique. Finalement une tâche sporadique τ_i est caractérisée par trois paramètres temporelles, durée d'exécution C_i, délai critique D_i et un intervalle minimal T_i séparant deux occurrences de τ_i.

1.3.3.3 Tâches apériodiques

Les tâches apériodiques possèdent souvent des délais critiques, mais pas de période (ni intervalle minimal séparant deux instances), ni de date de réveil. Leurs exécutions (activations) sont déterminées par l'arrivée des évènements (message ou requête de l'opérateur) qui peuvent être produits à tout instant. Dans ce cas il n'est pas possible de valider temporellement, a priori, un système à contraintes strictes. Des travaux traitant ce type de tâches sont présentés dans [101, 28].

1.3.4 Techniques d'ordonnancement

L'ordonnancement est de définir ou choisir une politique pour l'affectation des tâches (en concurrence) aux processeurs durant la vie du système afin de respecter toutes les contraintes temporelles comme l'échéance ou afin d'optimiser d'autres critères temporels (comme temps de réponse ou minimiser la longueur de l'ordonnancement). On distingue deux approches :

1.3.4.1 Un ordonnancement hors-ligne (off-line scheduling)

Cette politique est basée sur une séquence d'ordonnancement valide (à répéter à l'infini) définissant l'ordre dans lequel les tâches doivent être exécutées (dates de début d'exécution des tâches, de préemption/reprise éventuelles). La séquence est générée (avant de la mettre en œuvre) par l'algorithme choisi et implantée dans une table qui sera consultée par un ordonnanceur simplifié que l'on appelle **séquenceur**. Cette approche est alors inflexible (statique) et ne s'adapte pas

aux changements de l'environnement ou à un environnement dont le comportement n'est pas complètement prévisible [18, 22].

1.3.4.2 Un ordonnancement en-ligne (on-line scheduling)

Une politique basée sur un algorithme implantée et exécutée par l'ordonnanceur à chaque instant (éventuellement à chaque requête d'exécution), au cours de fonctionnement du système, afin d'attribuer le processeur à une tâche. Ces algorithmes reposent sur les paramètres temporels des tâches, à l'instant de décision, pour choisir la tâche à ordonnancer parmi les tâches effectivement présentes dans la file d'ordonnancement (activées). Cette politique est flexible et capable de s'adapter facilement aux changements de l'environnement au cours de la vie du système. La validation d'un système, fondé sur un ordonnancement en-ligne, repose sur une analyse effectuée hors-ligne du système, appelée analyse d'ordonnançabilité ou de faisabilité.

Nous nous intéressons, dans la suite, à l'ordonnancement en-ligne en présentant les différents algorithmes d'ordonnancement (généralement d'affectations de priorités) et les méthodes de validation de l'ordonnancabilité.

1.4 Étude de l'ordonnancement en-ligne et problématique

L'ordonnanceur est en charge de choisir la tâche à exécuter par le processeur, à chaque instant de décision. Ce choix est fondé sur des algorithmes d'ordonnancement de complexité temporelle faible (le plus souvent linéaire en fonction du nombre de tâches). Ces algorithmes consistent à attribuer une priorité à chacune des tâches. L'ordonnanceur, lorsqu'il est invoqué, sélectionne alors la tâche la plus prioritaire, selon l'algorithme choisi. Deux types d'affectation de priorités sont distingués :

- ***Priorité fixe :*** Chaque tâche possède une priorité fixe, durant toute la vie du système. Les priorités sont attribuées aux tâches par des algorithmes dits à priorités fixes en fonction des paramètres temporels statiques comme la période ou le délai critique. L'attribution des priorités est effectuée initialement lors de la conception du système.
- ***Priorité dynamique :*** Les priorités des tâches varient au cours de l'exécution du système. Effectivement, les algorithmes d'ordonnancement dit dynamique attribuent les priorités aux tâches en fonction des paramètres temporels dynamiques comme l'échéance la plus proche. L'algorithme doit alors recalculer les priorités à chaque instant de décision.

1.4.1 Contexte d'ordonnancement

Le choix d'un algorithme d'ordonnancement dépend essentiellement du contexte dans lequel il est exécuté. Le contexte est défini par les différentes caractéristiques des systèmes temps réel que nous avons évoquées précédemment dans ce chapitre. Un système est défini par son architecture matérielle du support, nature des tâches, mode d'exécution des tâches, etc. Le tableau 1.1 résume les différentes caractéristiques d'un contexte d'ordonnancement.

Architecture matérielle	Nature des tâches	Mode d'exécution	Dates d'activation	délais critiques
Monoprocesseur	Périodique	Préemptif	Synchrone/ asynchrone	Requête sur échéance
Multiprocesseur	Sporadique			Bien formé
Distribuée	Apériodique	Non-préemptif	Concrète/ non concrète	$D_i \leq T_i$
				Quelconque

TAB. 1.1 – Contextes d'ordonnancement.

1.4.2 Propriétés

Nous donnons quelques propriétés et définitions utilisées dans l'analyse d'ordonnançabilité des systèmes. Soit S un système de tâches, composé de n tâches $\{\tau_1, \tau_2, ...\tau_n\}$, donné dans un contexte X.

- Une séquence d'ordonnancement des tâches d'un système est valide si, et seulement si, toutes les échéances sont respectées.
- Un système de tâches S est **ordonnancable** si, et seulement si, il existe une séquence d'ordonnancement de longueur infinie valide.
- Soit A un algorithme d'ordonnancement, et S un système de tâches temps réel. S est fiablement ordonnancé par A si, et seulement si, la séquence produite par A est valide.
- Un algorithme d'ordonnancement optimal [38] pour une classe de problèmes d'ordonnancement donné est tel que : si un système est ordonnançable par au moins un algorithme de la même classe, alors le système est ordonnançable par l'algorithme d'ordonnancement optimal. En conséquence, si un système n'est pas ordonnançable par un ordonnanceur optimal d'une classe donnée, alors il ne l'est par aucun ordonnanceur de la même classe.
- Un test d'ordonnançabilité est un algorithme qui, pour un système S et un algorithme d'ordonnancement A donnés, renvoie une réponse négative si l'algorithme A peut générer une séquence non valide. Le sujet de cette thèse traite ce problème d'analyse d'ordonnançabilité.

1.4.3 Problème d'ordonnancement en tant que problème de décision

Étant donné un système de tâches S, et un algorithme d'ordonnancement A (priorités fixes ou dynamiques). Un problème de décision est posé :

l'algorithme A Ordonnance-t-il fiablement le système S ?

La réponse est sous forme de tests a priori d'ordonnançabilité. Donc la preuve ou la validation de l'ordonnancabilité d'un système avant de le mettre en œuvre est indispensable. Cela nécessite une connaissance complète des contraintes temporelles du système, plus l'algorithme d'attribution de priorités choisi. La validation s'effectue hors-ligne soit par **simulation** du système de tâches durant une durée suffisante, appelée **durée de simulation** [30], sous l'hypothèse que

le système de tâches soit « robuste », c'est-à-dire que le pire comportement du système s'obtient lorsque les tâches s'exécutent avec leurs pires durées d'exécution (i.e. WCET), soit à l'aide de conditions d'ordonnancabilité analytiques basées sur les critères temporels des tâches. Ces conditions peuvent être des conditions suffisantes ou des conditions nécessaires et suffisantes. Il est important de noter qu'un critère analytique s'applique dans un contexte restreint et pour un algorithme d'ordonnancement particulier. On distingue trois principales techniques d'analyse [90, 89] :

- ***Analyse de l'utilisation processeur(Processor Utilization Analysis) :*** Le facteur d'utilisation est la fraction de temps que le processeur passe à exécuter des tâches. Il se définit par $U = \sum_{i=1}^{n} \frac{C_i}{T_i}$. Dans certains cas particuliers, le facteur d'utilisation du processeur permet de conclure si une configuration de tâches est ordonnançable ou non, et de calculer des conditions d'ordonnançabilité.
- ***Analyse de la demande processeur(Processor Demand Analysis) :*** Cette technique repose sur le calcul de la demande cumulée des exécutions des tâches réveillées et terminées dans un intervalle de temps (Demand Bound Function). Cette approche générale permet de construire des tests d'ordonnancabilité [52, 45, 11] en limitant l'étude d'ordonnançabilité à quelques intervalles d'une période d'activité du processeur.
- ***Analyse des temps de réponse(Response Time Analysis) :*** Cette technique consiste à calculer les pires temps de réponse des tâches et à les comparer avec leurs délais critiques. Une tâche est ordonnançable si, et seulement si, son pire temps de réponse est inférieur ou égal à son délai critique. Dans le cadre de l'ordonnancement à priorité fixe, cette méthode est souvent désignée sous le nom d'analyse RTA (Response Time Analysis)[47].

Notons que les deux dernières techniques reposent sur le même concept de période d'activité du processeur (l'intervalle de temps durant lequel le processeur n'est pas oisif). Ces deux techniques font la base du travail effectuée dans cette thèse. Et avant de les détailler nous présentons quelques résultats d'analyse d'ordonnancabilité basés sur l'étude du facteur d'utilisation, classiques dans la littérature, classés par le type d'algorithmes d'ordonnancement (à priorité fixe ou dynamique).

1.4.4 Algorithmes d'ordonnancement à priorités fixes

L'attribution, avant la mise en fonctionnement du système, des priorités est assurée par les algorithmes à priorités fixes, qui sont généralement basés sur des contraintes temporelles statiques des tâches comme la période ou le délai critique. Dans le cadre de ces algorithmes, on trouve, dans la littérature, plusieurs conditions d'ordonnancabilité (nécessaires, suffisantes, ou nécessaires et suffisantes). Notons que le problème de faisability étant co-NP-difficile lorsque des tâches sont asynchrones [16]. Ces conditions sont basées sur la notion d'instant critique.

Définition 1 *(Instant critique) Un instant critique d'une tâche est une date d'activation de l'occurrence (instance) ayant le pire temps de réponse parmi toutes ses occurrences.*

Dans le contexte de tâches indépendantes de Liu et Layland [55], un instant critique pour une tâche a lieu lorsqu'elle est réveillée en même temps que toutes les tâches plus prioritaires qu'elle. Les algorithmes à priorités fixes usuels, leurs contextes d'optimalité et les conditions d'ordon-

nancabilité, dans le contexte monoprocesseur sont présentées ci-après. Pour plus de détails, un état de l'art complet sur ce sujet est présenté dans [7].

1.4.4.1 Rate Monotonic (RM)

L'algorithme RM [95, 55] affecte les priorités aux tâches de façon inversement proportionnelle à leurs périodes. Ainsi la tâche la plus prioritaire correspondra à la tâche avec une période la plus courte. En cas d'égalité de priorités des plusieurs tâches, le choix de la tâches à exécuter sera arbitraire. Théorème 1 montre dans quel contexte d'ordonnancement RM est optimal.

Théorème 1 *[55, 34] L'algorithme RM est optimal, dans la classe des algorithmes à priorités fixes, pour des systèmes de tâches indépendantes, synchrones, et périodiques à échéances sur requête.*

Une condition suffisante d'ordonnancabilité d'un système de tâches périodiques ordonnancées par RM est donnée dans [55, 35]

Théorème 2 *[55] Un système S composé de n tâches périodiques à échéance sur requête et indépendantes est ordonnancable par RM si :*

$$U = \sum_{i=1}^{n} \frac{C_i}{T_i} \leq n(2^{1/n} - 1) \tag{1.1}$$

Le deuxième membre de l'inégalité tend vers $\ln 2$ $(= 0,69)$ *lorsque n tend vers l'infini.*

1.4.4.2 Deadline Monotonic (DM)

L'algorithme DM a été proposée par Leung et Whithead [54] (appelé aussi inverse deadline), elle attribue les priorités aux tâches inversement proportionnelle à leurs délais critiques. La tâche la plus prioritaire est celle possédant le délai critique le plus court. En cas d'égalité de priorités, le choix de la tâche à exécuter est fait arbitrairement. Théorème 3 montre dans quel contexte d'ordonnancement DM est optimal.

Théorème 3 *[54] L'algorithme DM est optimal, dans la classe des algorithmes à priorités fixes, pour des systèmes de tâches indépendantes, synchrones, et à échéances inférieures aux périodes ($D_i \leq T_i$).*

Lorsque les tâches sont à échéances sur requêtes ($D_i = T_i$), on se ramène au même ordre de priorité produit par l'algorithme RM. Une condition suffisante d'ordonnançabilité basée sur la théorème 2 (pour RM [55]) est présentée dans [54].

Théorème 4 *[54] Un système S composé de n tâches périodiques à échéance sur requête et indépendantes est ordonnançable par DM si :*

$$U = \sum_{i=1}^{n} \frac{C_i}{T_i} \leq n(2^{1/n} - 1) \tag{1.2}$$

Le lecteur peut se référer à [8] pour une démarche générale de validation temporelle de systèmes de tâches par DM. D'autre algorithmes d'ordonnancement à priorités fixes ont été proposés, nous citons l'algorithme d'Audsley [6] qui est optimal pour les tâches indépendantes à départs différés.

1.4.5 Algorithmes d'ordonnancement à priorités dynamiques

Généralement, ils sont basés sur des contraintes temporelles dynamiques (leurs valeurs varient au cours du vie de système) comme l'échéance la plus proche. Ces algorithmes doivent être exécutés, dans les meilleurs cas, à chaque instant de réveil, afin de recalculer (actualiser) les priorités des différentes tâches. Nous focalisons sur les contextes d'optimalité et les conditions d'ordonnançabilités des algorithmes les plus connus.

1.4.5.1 Earliest Deadline First (EDF)

A chaque invocation d'ordonnancement, EDF [95, 55] assigne la priorité la plus forte à l'instance d'une tâche dont l'échéance arrive le plus tôt. En cas d'égalité de priorités de plusieurs instances, un choix aléatoire est effectué. Le théorème 5 précise le contexte d'optimalité d'EDF.

Théorème 5 *[31, 50] Earliest Deadline est optimal, dans la classe des algorithmes d'ordonnancement en-ligne, pour des systèmes de tâches indépendantes, à échéances inférieures ou égales aux périodes ($D_i \leq T_i$).*

Le théorème 6 démontre l'optimalité de l'algorithme EDF même pour l'ordonnancement d'un ensemble arbitraire (fini ou infini) d'instances.

Théorème 6 *[31, 14] Si un ensemble d'instances indépendante J est ordonnancé fiablement par un algorithme quelconque d'ordonnacement, alors J est ordonnancé fiablement par l'algorithme Earliest Deadline First (EDF).*

Une condition nécessaire et suffisante d'ordonnançabilité de tâches synchrones et à échéances sur requête, proposée par Liu et Layland [55], est donnée par le théorème 7. Ce théorème a été étendu, par Baruah [15], pour des tâches synchrones et à échéances non reliées au périodes (quelconques).

Théorème 7 *[55] Un système S composé de n tâches synchrones à échéances sur requêtes est ordonnançable par EDF si, et seulement si :*

$$U = \sum_{i=1}^{n} \frac{C_i}{T_i} \leq 1 \tag{1.3}$$

1.4.5.2 Least Laxity (LL)

Cet algorithme s'appuie sur un paramètre temporel dynamique appelé Laxité. A une date t, la laxité $l_i(t)$ d'une instance, activée à la date a d'une tâche τ_i est l'intervalle entre son échéance $a + D_i$ et sa fin d'exécution prévue $t + C_i(t)$, tel que $C_i(t)$ est la durée d'exécution restante pour l'instance de τ_i ($l_i(t) = a + D_i - t - C_i(t)$. Donc à chaque instant d'invocation t, LL [65, 32] attribue la priorité la plus forte à la tâche ayant la plus faible laxité. En cas d'égalité de priorités, aucune règle n'est précisée. Ceci engendre un grand nombre de changements de contexte ce qui génère une surcharge processeur très importante. Cet algorithme est optimal dans le même contexte que EDF.

Théorème 8 *[65, 32] Least Laxity est optimal, dans la classe des algorithmes d'ordonnancement en-ligne, pour des systèmes de tâches indépendantes et à échéances inférieures ou égales aux périodes ($D_i \leq T_i$).*

1.4.6 Ordonnancement des tâches dépendantes

L'étude d'ordonnancement qui été présentée dans la section précédente suppose un modèle de système monoprocesseur assez restrictif. Toutes les tâches sont indépendantes (sans aucune relation de précédence entre elles) et aucun partage de ressources n'est considéré. Nous présentons brièvement dans les sous sections qui suivent les extensions apportées à l'étude l'ordonnancement des modèles de tâches, en prenant en compte plus de contraintes temporelles.

1.4.6.1 Contraintes de précédence

La majorité des systèmes temps réel nécessitent des communications entre les tâches. Une tâche (réceptrice) en cours d'exécution, à un point de communication, est mise en attente d'une condition (message) jusqu'à la satisfaction de la condition (arrivée du message). Ainsi l'exécution la partie de (située après le point de communication) de la tâche réceptrice doit être précédée par l'exécution de la tâche émettrice, ce qui introduit des contraintes de précédence entre certaines parties des tâches. Afin d'utiliser les conditions d'ordonnançabilité des tâches indépendantes sans modification, les tâches communicantes sont découpées autour des points de communication (synchronisation) d'une manière à obtenir des tâches recevant leurs messages au début d'exécution et les envoyant à la fin. Les paramètres temporelles (en l'occurrence les r_i et les D_i), des tâches obtenues, sont modifiés en assurant une affectation des priorités par les algorithmes classiques assurant que la tâche précédée par une autre aura une priorité inférieure à la tâche qui la précède.

1.4.6.2 Ressources partagées

Il est indispensable d'intégrer le partage de ressources (accès protégé à une ressource) aux algorithmes d'ordonnancement car rares sont les applications temps réel qui n'utilisent pas de ressources critiques. On distingue deux types de ressources [26, 88] : les ressources matérielles (comme les organes périphériques, la mémoire, etc.) et les ressources logicielles (telles que les variables, les buffers de données, les fichiers, etc.). Chaque ressource possède une capacité d'entrée, cette capacité représente le nombre d'accès simultanés permis (acceptés) par la ressource. On dit que la ressource est critique lorsqu'elle n'accepte qu'un seul accès à la fois. [17] fournit une présentation détaillée des ressources critiques et de sémaphores. [65] a montré que le problème de l'ordonnancement en présence de ressources est $NP - difficile$ au sens fort, et qu'aucun algorithme non clairvoyant (i.e. ayant connaissance du future) n'était optimal dans ce contexte. De plus, deux phénomènes ou problèmes peuvent se produire lors de l'ordonnancement d'un système de tâches avec une partage de ressource :

- ***Interblocage :***
  Ce phénomène est connu non seulement pour les systèmes temps réel, elle se produit lorsque deux ou plusieurs tâches sont bloquées car chacune a demandé l'accès à une ressource en possession de l'autre. Supposons, par exemple, deux tâches τ_1 et τ_2 en partage de deux

ressources en accès exclusif R_1 et R_2. Un interblocage survient lorsque τ_1 possède R_1 et τ_2 possède R_2 et que τ_1 et τ_2 ont à leur tour besoin de R_2 et R_1 respectivement.

– ***Inversion de priorité non borné :***
Ce phénomène survient lorsque une tâche, plus prioritaire, est bloquée par l'exécution des tâches moins prioritaires et qui ne partage pas de ressource, avec elle. Effectivement une tâche τ_1, de priorité supérieure, est bloquée en attente d'une ressource en possession d'une tâche moins prioritaire τ_2, cette dernière (τ_2) peut être préemptée par toutes tâches τ_j, de priorité intermédiaire (entre celle de τ_1 et τ_2), et qui n'est pas en conflit, ni avec τ_1 ni avec τ_2. Cette préemption d'exécution de l'exécution de τ_2 produit un blocage non borné pour τ_1. Ce phénomène d'inversion de priorité doit être pris en considération lors de la conception afin de mener une analyse fiable d'ordonnançabilité. L'absence de gestion d'accès aux ressources peut engendrer des dysfonctionnements importants comme cela a été dans la mission Mars Pathfinder lancée par la NASA [27].

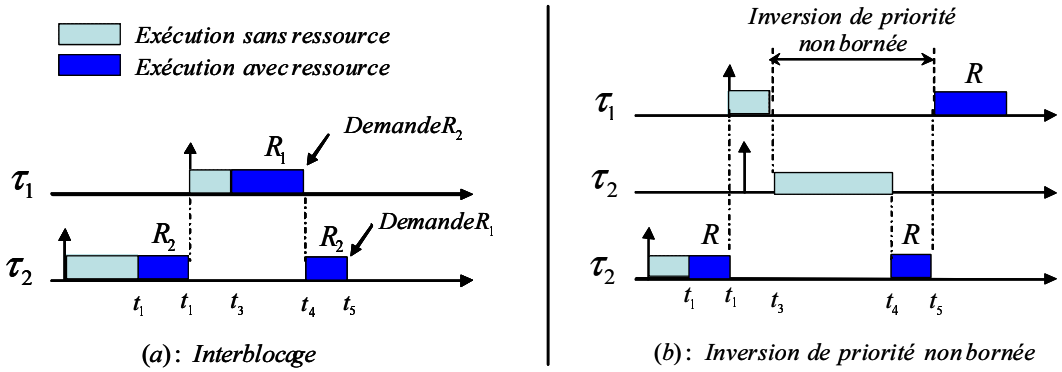

FIG. 1.5 – Interblocage et inversion de priorité non bornée.

Dans le but d'empêcher ou de borner l'inversion de priorité, plusieurs protocoles de gestion de ressources ont été proposés dans la littérature.

1.4.6.2.1 Protocole à priorités héritées (PPH) PPH (Priority Inheritance Protocol) [97, 98] a été proposé pour palier le problème d'inversion de priorité non bornée. L'idée est simple et consiste à interdire toute préemption, de la tâche possédant une ressource, demandée par des tâches plus prioritaires (tâches bloquées), par toute tâche de priorité intermédiaire. Pour cela on affecte, à la tâche bloquante, la plus grande priorité parmi les priorités de toutes les tâches qu'elle bloque (On dit qu'elle hérite d'une priorité). Lorsque la tâche bloquante libère une ressource, sa priorité est recalculée en fonction des ressources qui sont toujours en sa possession. Elle reprend sa priorité initiale lorsqu'elle libère toutes les ressources. Notons que l'héritage de priorité est transitif, tel que si une tâche τ_k bloque une tâche τ_j qui elle même bloque une autre tâche τ_i, alors τ_k hérite de la priorité de τ_i par l'intermédiaire de τ_j, on parle dans ce cas de blocage transitif. Une tâche ne peut être bloquée à l'entrée d'une section critique qu'une fois par ressource existante. Ainsi le temps de blocage est la somme des plus longues sections critiques sur chaque ressource (théorème 9). Des méthodes de calcul de temps de blocage sont données dans [98, 20, 56]

Théorème 9 *[97, 98] En utilisant le protocole à priorités héritées, une tâche τ_i d'un système S ne peut être bloquée que pendant un temps égal à*

$$\sum_{R\ ressource} \max_{j \neq i} \{durée\ \ de\ \ utilisation\ \ de\ \ R\ \ par\ \ \tau_j\}$$

Avec cette méthode, le temps de blocage est élevé si plusieurs tâches partagent une même ressource, et il ne peut être calculé que si les tâches possèdent des priorités fixes. Ainsi, ce protocole n'est utilisable qu'avec des algorithmes d'ordonnancement à priorités fixes. Notons enfin que l'utilisation de PPH ne résout pas le problème d'interblocage.

1.4.6.2.2 Protocole à priorité plafond (PPP) Le PPP (Priority Ceiling Protocol) a été présenté,dans les mêmes travaux de Sha et al [98], afin de résoudre les limitations de PPH. Le principe consiste à affecter à chaque ressource une priorité plafond (seuil), qui est la plus forte des tâches qui y accèdent. Le mécanisme [29] du protocole est le suivant :

- Si la ressource est libre :
  - une tâche accède à cette ressource si sa priorité est strictement superieure aux priorités plafonnées des ressources en cours d'utilisation au niveau de tout l'application ;
  - sinon la tâche est bloquée et la tâche possédant la ressource hérite de la priorité de la tâche bloquée.
- Si la ressource n'est pas libre : la tâche est bloquée et la tâche possédant la ressource hérite de la priorité de la tâche bloquée.

Notons que ce protocole est destiné aux algorithmes d'ordonnancement à priorités fixes. Cependant, [21] l'ont étendu pour l'utiliser avec l'algorithme EDF. Le protocole résultant est nommé protocole à priorité plafond dynamique ou PPPD. L'avantage de PPP est d'empêcher tout interblocage et d'améliorer la borne du temps de blocage (limitation du nombre de blocages). Une tâche τ_i ne peut être bloquée que pendant la durée d'une section critique d'une tâche de priorité inférieure utilisant une ressource de priorité plafond plus grande ou égale à celle de τ_i.

Théorème 10 *[98] En utilisant le protocole à priorité plafond, le temps de blocage maximal B_i d'une tâche τ_i est donnée par la durée de la plus grande section critique des autres tâches.*

Soit S_k une ressource critique, $C(S_k)$ est la priorité plafond de S_k (la plus forte priorité des tâches accédant à S_k), et $D_{j,k}$ la durée de la plus longue section critique appartenant à une tâche τ_j (de priorité π_j) et l'accès à une ressource S_k, alors le pire temps de blocage d'une tâche τ_i (de priorité π_i) est :

$$B_i = \max_{j,k} \{D_{j,k} / \pi_j > \pi_i, C(S_k) \leq \pi_i\} \tag{1.4}$$

Dans le contexte des tâches à échéance sur requête, ordonnancées par l'algorithme à priorités fixes RM avec PPP, une condition suffisante d'ordonnancabilité [98], d'un système composé de n tâches τ_1,...τ_n (par convention, les tâches sont ordonnées suivant les périodes croissantes) est donnée par :

$$\forall i \in \{1..n\} : \frac{B_i}{T_i} + \sum_{j=1}^{i} \frac{C_j}{T_j} \leq i.\left(2^{1/i} - 1\right) \tag{1.5}$$

Notons qu'une condition suffisante d'ordonnançabilité, a été proposée dans [21], pour l'algorithme à priorité dynamique EDF avec PPPD.

1.4.6.2.3 Protocole d'allocation de la pile Cette adaptation de PPP étant très coûteuse pour EDF (à cause des réévaluations des priorités plafonds), Baker [9] a proposé le protocole d'allocation de la pile PAP (SRP, pour Stack Resource Policy), qui est plus adapté à l'algorithme d'ordonnancement EDF. Ce protocole est un évolution du PPP, notamment vers le cas de ressources à plusieurs instances. Il consiste à attribuer un nouveau paramètre π_i, nommé niveau de préemption, à chaque tâche τ_i. Ce paramètre est fixé hors-ligne et indépendant du type l'algorithme d'ordonnancement utilisé (à priorité fixe ou dynamique) ; par exemple on peut attribuer les π_i suivant DM, ce qui réduit le nombre de changements de contexte avec l'algorithme d'ordonnancement EDF. Chaque ressource R possède une valeur plafond C_r, qui est la valeur maximale des niveaux de préemption des tâches actives ayant besoin de plus d'instances de la ressource R qu'il n'y en a de disponibles ; donc C_r est un paramètre dynamique.
Une tâche τ_j ne peut préempter une tâche τ_i que si les conditions suivantes sont vérifiées :

- τ_j est plus prioritaire que la tâche τ_i (suivant l'algorithme d'ordonnancement)
- le niveau de préemption de τ_j est supérieur à celui de τ_i ($\pi_j > \pi_i$)
- Le niveau de préemption de τ_j est supérieur au plafond système (la plus forte valeur plafond parmi celle des ressources)

En utilisant ce protocole, une tâche n'est pas autorisée à démarrer son exécution tant que toutes les ressources qui lui sont nécessaires ne sont pas disponibles, ce qui permet d'éviter l'interblocage. Notons que l'héritage de priorité est implicite, car une tâche bloquante, d'une tâche bloquée, ne peut être préemptée par une tâche de niveau de préemption intermédiaire, grâce au plafond système qui prend la valeur du niveau de préemption de la tâche bloquée. De plus PAP limite le nombre de sections critiques pouvant bloquer une tâches à un. Notons que [9] propose une condition suffisante d'ordonnançabilité pour un système de tâches quelconques lorsque le PAP est utilisée en association avec l'algorithme d'ordonnancement EDF.

Nous avons présenté ci-dessus des conditions suffisantes ou nécessaires et suffisantes (mais pour des ensembles de tâches avec des contraintes très restreintes) d'ordonnançabilité, basées sur le facteur d'utilisation. Dans le cas ou aucune décision ne peut être prise, nous pouvons utiliser des techniques de test d'ordonnançabilité plus précises, basées sur l'analyse de pires cas. Dans la suite nous détaillons deux techniques l'analyse de temps réponse et l'analyse de demande processeur.

1.5 Analyse de temps de réponse

Dans cette section, nous développons l'analyse de temps de réponse, pour tâches périodiques indépendantes, les notations suivantes sont utilisées pour caractériser une tâche τ_i.

- C_i : durée d'exécution.

- D_i : délai critique.
- T_i : période d'activation.
- P_i : priorité de la tâche τ_i.
- R_i : temps de réponse.

Étant donné un système de tâche ordonnancé en-ligne par un algorithme à priorités (fixes ou dynamiques). La validation temporelle du système consiste à vérifier que toutes les échéances de toutes les tâches seront respectées durant toute la vie du système. L'analyse de temps réponse consiste à valider l'ordonnançabilité d'un système tâche par tâche, en calculant leurs pires temps de réponse. Une tâche est ordonnancée (valide), si son pire temps de réponse est inférieur ou égal à son échéance. Évidement, un système est ordonnancé si, et seulement si, toutes ses tâches sont ordonnancées. Puisqu'un dépassement d'échéance ne survient jamais lorsque le processeur est libre, alors le test d'ordonnançabilité est limité dans des intervalles de temps, où le processeur exécute des tâches (sans temps creux), appelés périodes d'activité.

Définition 2 *Une période d'activité du processeur est un intervalle de temps durant lequel le processeur est pleinement utilisé.*

Dans le contexte des tâches périodiques, la plus longue période d'activité se produit lors d'une activation simultanée de toutes les tâches du système, cet instant est appelé instant critique [55]. Soit $t_0 = 0$ la date d'activation simultanée de toutes les tâches d'un système. La périodicité d'activation de tâches permet de calculer le nombre d'instances, d'une tâche τ_j, activées dans n'importe quel intervalle de temps de longueur t (ces instances ne sont pas nécessairement terminées dans t), ce nombre est égal à $\left\lceil \frac{t}{T_j} \right\rceil$. Ainsi la durée cumulée des exécutions (appelée interférence ou travail processeur), causée par τ_j dans t est :

$$W_j(t) = \left\lceil \frac{t}{T_j} \right\rceil .C_j \tag{1.6}$$

Évidement, La charge totale $W(t)$ du processeur dans un intervalle de temps t est la somme des charges causées par toutes les tâches.

$$W(t) = \sum_{j=1}^{n} W_j(t) = \sum_{j=1}^{n} \left\lceil \frac{t}{T_j} \right\rceil .C_j \tag{1.7}$$

La figure 1.6 représente graphiquement la fonction de la charge processeur $W(t)$, du système de tâches de la table 1.2, lorsque toutes les tâches s'activent simultanément. La fonction de la charge processeur $W(t)$ est en escalier, et la ligne $f(t) = t$ définit la capacité maximum de traitement du processeur. La date à laquelle la charge processeur égale la longueur de l'intervalle de temps considéré, signifie que toutes les instances réveillées avant t sont terminées à cette date. Dans l'exemple de figure 1.6 à la date $t = 14$, $W(t) = t = 14$. La première date vérifiant la condition $W(t) = (t)$, donne la longueur de la période d'activité L. L correspond à un point fixe de l'équation $W(L) = (L)$. La détermination de la longueur de la période d'activité revient alors à calculer itérativement le plus petit point fixe de l'équation $W(L) = (L)$. Puisque toutes les

τ_i	C_i	T_i
τ_1	1	4
τ_2	1	5
τ_3	2	8
τ_4	3	18

TAB. 1.2 – Exemple d'un système de tâches périodiques.

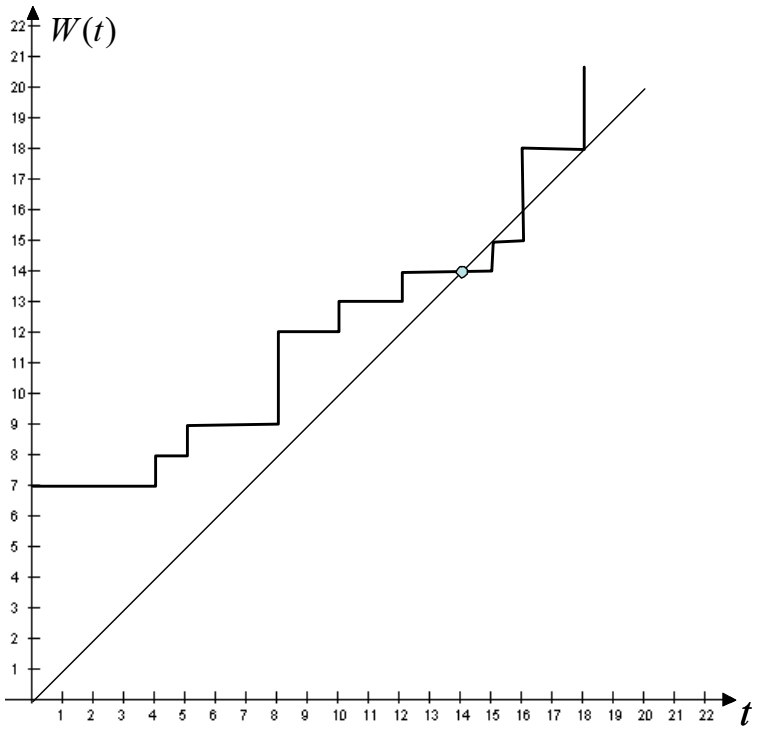

FIG. 1.6 – Fonction de charge processeur et période d'activité.

tâches sont réveillées à l'instant $t = 0$, alors L est initialisé par la somme des durées d'exécutions de toutes les tâches.

$$\begin{aligned} L^{(0)} &= \sum_{j=1}^{n} C_j \\ L^{(k+1)} &= W(L^{(k)}) \end{aligned} \tag{1.8}$$

Le calcul itératif est arrêté lorsque deux valeurs consécutives de L sont égales ($L = L^{(m)} = L^{(m+1)}$). L'équation 1.8 converge en un nombre fini d'itérations si le facteur d'utilisation du processeur est inférieur ou égal à 1 ($U = \sum_{j=1}^{n} \frac{C_j}{T_j} \leq 1$). Nous calculons la longueur de la période d'activité synchrone de notre exemple du système dont les paramètres sont donnés dans la table 1.2.

$$
\begin{aligned}
L^{(0)} &= \sum_{j=1}^{4} C_j = 7 \\
L^{(1)} &= W(L^{(0)}) = \lceil \tfrac{7}{4} \rceil 1 + \lceil \tfrac{7}{5} \rceil 1 + \lceil \tfrac{7}{8} \rceil 2 + \lceil \tfrac{7}{18} \rceil 3 = 9 \\
L^{(2)} &= W(L^{(1)}) = \lceil \tfrac{9}{4} \rceil 1 + \lceil \tfrac{9}{5} \rceil 1 + \lceil \tfrac{9}{8} \rceil 2 + \lceil \tfrac{9}{18} \rceil 3 = 12 \\
L^{(3)} &= W(L^{(2)}) = \lceil \tfrac{12}{4} \rceil 1 + \lceil \tfrac{12}{5} \rceil 1 + \lceil \tfrac{12}{8} \rceil 2 + \lceil \tfrac{12}{18} \rceil 3 = 13 \\
L^{(4)} &= W(L^{(3)}) = \lceil \tfrac{13}{4} \rceil 1 + \lceil \tfrac{13}{5} \rceil 1 + \lceil \tfrac{13}{8} \rceil 2 + \lceil \tfrac{13}{18} \rceil 3 = 14 \\
L^{(5)} &= W(L^{(4)}) = \lceil \tfrac{14}{4} \rceil 1 + \lceil \tfrac{14}{5} \rceil 1 + \lceil \tfrac{14}{8} \rceil 2 + \lceil \tfrac{14}{18} \rceil 3 = 14
\end{aligned}
$$

Un test d'ordonnançabilité, utilisant l'analyse de temps de réponse, nécessite alors identifier le pire scénario d'exécution d'une tâche. Pour cela une connaissance sur le contexte d'ordonnancement (surtout la loi d'arrivée des tâches) et l'algorithme d'ordonnancement utilisé est indispensable. Nous présentons d'abord l'analyse pour les tâches périodiques avec priorités fixes puis celle avec priorités dynamiques (EDF).

1.5.1 Analyse de temps de réponse avec priorités fixes

Soit S un système de n tâches. Sans perte de généralité, supposons que les indices des tâches indiquent leurs niveaux de priorité. Ainsi τ_1 a le niveau de priorité le plus élevé et τ_n a le plus faible. Supposons que les tâches sont affectées à des niveaux de priorité différents. Afin de calculer le pire temps de réponse, d'une tâche analysée τ_i, on doit caractériser le ou les scénarios d'arrivées des tâches qui conduisent à lui. τ_i ne peut être interrompue que par des tâches plus prioritaires, ce qui introduit la notion de période d'activité de niveau i.

Définition 3 *[51] En priorité fixe, une période d'activité du processeur de niveau i est un intervalle de temps durant lequel le processeur est pleinement utilisé, seulement par les tâches de priorité supérieure ou égale à i.*

Notons $W_i(t)$ la charge processeur causée seulement par les tâches de priorité supérieure ou égale à i.

$$W_i(t) = \sum_{j=1}^{i-1} \left\lceil \frac{t}{T_j} \right\rceil C_j \tag{1.9}$$

[47, 55, 65] ont prouvé que le pire temps de réponse d'une tâche τ_i d'un ensemble de tâches périodiques ou sporadiques, est obtenu dans le scénario dans lequel toutes les tâches sont activées à leur rythme maximal et de manière synchrone à un instant critique $t = 0$. Ce résultat à été étendu par [51]. Il a prouvé que le pire temps de réponse d'une tâche τ_i de priorité i, se produit dans une période d'activité de niveau i. Une période d'activité de niveau i est initiée par un instant critique, survenant lorsque τ_i s'active au même instant que toutes les tâches plus prioritaires.

Théorème 11 *[51] Le plus grand temps de réponse d'une tâche τ_i survient dans une période d'activité de niveau i où toutes les tâches débutent leurs exécutions par un instant critique.*

Dans une période d'activité de niveau i, plusieurs instances de la tâche analysée τ_i sont exécutées. Le pire temps de réponse est alors le maximum parmi les temps de réponse de toutes les instances activées dans la période d'activité. La longueur de la période d'activité de la même manière que l'équation 1.8 en se limitant aux tâches de priorité supérieure ou égale à i.

$$\begin{aligned} L^{(0)} &= \sum_{j=1}^{n} C_j \\ L^{(k+1)} &= W_i(L^{(k)}) \end{aligned} \tag{1.10}$$

La première instance ($k = 1$) de τ_i est activée à la date $t = 0$ (instant critique). Le calcul de temps de réponse de la K^{eme} instance de τ_i située dans la période d'activité synchrone est basée sur le calcul de la date de sa fin d'exécution qui correspond à la longueur de la période d'activité, en prenant en compte seulement les instances activées à ou avant $(K-1)T_i$ pour τ_i.

$$\begin{aligned} R_i^{(0)} &= K.C_i \\ R_i^{(n)} &= K.C_i + W_i(R_i^{(n-1)}) = K.C_i + \sum_{j \in hp_i(\tau_i)} \left\lceil \frac{R_i^{(n-1)}}{T_j} \right\rceil C_j \end{aligned} \tag{1.11}$$

Le temps de réponse est l'intervalle de temps entre la date d'activation et celle de fin d'exécution. Sachant que la K^{eme} activation de τ_i survient à la date $t = (K-1)T_i$. Ainsi le temps de réponse est donné par :

$$R_i^{(K)} = R_i - (K-1)T_i \tag{1.12}$$

Soit Q le nombre d'instances de τ_i activées dans la période d'activité synchrone. D'où le pire temps de réponse est donné par :

$$R_i = \max_{K=1,2,..Q} R_i^{(K)} \tag{1.13}$$

Exemple 1 *Nous calculons le temps de réponse de la tâche τ_i du système dont les paramètres sont donnés dans la table 1.2. Supposons que τ_i la moins prioritaire que toutes les tâches.*

Notons que pour les systèmes de tâches périodiques à échéance sur requête, Liu et Layland [55] a prouvé que le pire temps de réponse d'une tâche correspond au temps de réponse de sa première instance dans la période d'activité synchrone. Le même résultat a été prouvé, par Joseph et Pandya [47] pour les tâches périodiques avec délais critiques inférieurs ou égaux aux périodes ($D_i \leq T_i$). Il suffit alors, dans ce contexte, de tester uniquement l'ordonnançabilité de la première instance de chaque tâches (appliquer l'équation 1.11 uniquement pour $k = 1$).

Les tâches sont considérées indépendantes, dans les calculs présentés ci-dessus. Supposons l'existence de ressources partagées entre tâches et étendons les formules afin de prendre en compte la durée maximum de blocage B_i, d'une tâche analysée τ_i. L'inversion de priorité ne survient qu'une fois et seulement au début de la période d'activité. Il suffit alors d'ajouter le temps maximum de blocage B_i dans la formule du temps de réponse.

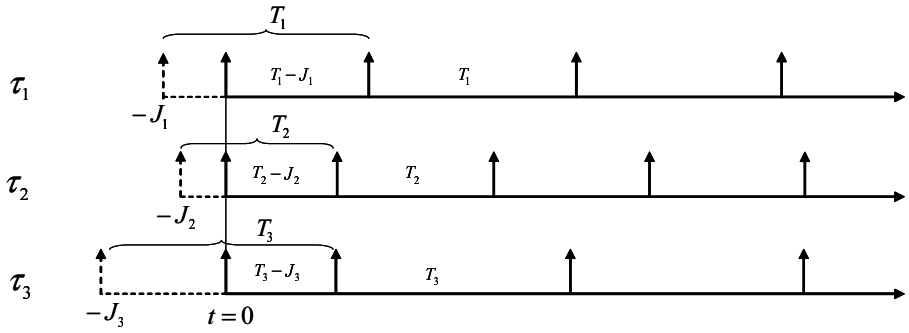

FIG. 1.7 – Gigues sur l'activation des tâches.

$$R_i^{(n)} = K.C_i + B_i + \sum_{j \in hp_i(\tau_i)} \left\lceil \frac{R_i^{(n-1)}}{T_j} \right\rceil C_j \tag{1.14}$$

Nous présentons l'extension de cette méthode lorsque les dates effectives d'activation des tâches peuvent être différées en attente d'un message par exemple. La date d'activation effective (le décalage d'activation) n'est pas connue a priori, mais elle est modélisée par un paramètre J_i appelé la gigue sur activation (valeur maximale d'un éventuel décalage). Soit $\tau_i =< r_i, C_i, D_i >$ une tâche avec un gigue J_i et r_i^k la date d'activation de la k^{eme} instance. Cette instance peut s'activer effectivement à n'importe quel instant entre r_i^k et $r_i^k + J_i$. La prise en compte de la gigue dans l'analyse de temps de réponse a été étudiée par Tindell [108]. Il a prouvé que le pire scénario d'une tâche analysée τ_i se produit lorsqu'elle est activée simultanément avec toutes les tâches plus prioritaires après avoir être retardées par une valeur maximale de gigue. Les tâches s'activent, après l'instant critique, à leur rythme maximal. Ainsi la date d'activation de la K^{eme} instance dans la période d'activité est donnée par :

$$\max\{(K-1)T_i - J_i, 0\} \tag{1.15}$$

Théorème 12 *[108] Dans un système* Γ *de tâches avec des gigues sur activation, l'instant critique d'une tâche analysée* τ_i *coïncide avec le réveil simultané de toutes les tâches de priorité supérieure avec une activation retardée d'une valeur correspondante à leur gigue maximale.*

La figure 1.7 montre un instant critique qui survient lorsque les trois tâches s'activent effectivement au même instant, après un décalage d'une valeur maximum de gigue. Donc l'interférence des tâches de niveau de priorité supérieur à un niveau i, dans un intervalle de temps t, est donnée par :

$$W_i(t) = \sum_{j=1}^{i-1} \left\lceil \frac{J_j + t}{T_j} \right\rceil C_j \tag{1.16}$$

La formule de calcul de la date de fin d'exécution d'une instance k d'une tâche τ_i devient :

$$R_i^{(n)} = K.C_i + B_i + \sum_{j \in hp_i(\tau_i)} \left\lceil \frac{J_j + R_i^{(n-1)}}{T_j} \right\rceil C_j \tag{1.17}$$

Ainsi le temps de réponse est :

$$R_i = R_i + J_i - (K-1)T_i \tag{1.18}$$

[100] a proposé une borne inférieure de temps de réponse, d'une instance, pour initialiser le calcul récurrent de l'équation 1.17, et ceci sans risque de rater le premier point fixe.

1.5.2 Analyse de temps de réponse avec priorités dynamique EDF

Nous présentons, dans cette section, les résultats d'analyse de temps de réponse des tâches sporadiques ordonnancées par EDF. Dans le cas des tâches avec priorités dynamiques, la définition d'instant critique n'est pas équivalent à celui des tâches avec priorités fixes (début de la période d'activité synchrone). [102] a montré que l'instant critique produisant le pire temps de réponse survient dans la plus longue période d'activité (Lemme 1), mais pas nécessairement à la première activation au sein de celle-ci.

Lemme 1 *[102] Le pire temps de réponse d'une tâche τ_i est trouvée dans une période d'activité où toutes les tâches autre que τ_i sont réveillées de façon synchrone et à leur rythme maximum (cas des tâches sporadiques).*

Le pire temps de réponse d'une tâche τ_i sera déterminé dans une période d'activité où toutes les tâches d'échéances inférieures ou égales sont exécutées avant elle. L'analyse de temps de réponse d'une tâche τ_i nécessite alors d'étudier plusieurs scénarios qui correspond aux dates d'activations possibles (échéances possibles). Sans perte de généralité, supposons la date $t = 0$, la date à laquelle toutes les tâches autre que τ_i sont activées simultanément. Étant donnée une instance de τ_i a une échéance d. Sa date de réveil est la date $a = d - D_i$. La date $S_i(a)$ de sa première instance activée dans la période d'activité est déduite alors comme :

$$S_i(a) = a - \left\lfloor \frac{a}{T_i} \right\rfloor T_i \tag{1.19}$$

La date de fin d'exécution de l'instance activée à la date a (avec échéance d), correspond à la longueur de la période d'activité associée à d (deadline d-busy period [38]) qui est définie par les tâches dont les échéances sont inférieures ou égales à d. Afin de calculer l'interférence ou la charge processeur causée par une tâche τ_j ($j \neq i$) dans un intervalle de temps t, nous prenons en compte, parmi les instances activées dans t, seulement celles ayant des échéances inférieures à d (instances plus prioritaires). Plus précisément, $\left\lceil \frac{t}{T_j} \right\rceil$ instances sont activées dans t, mais au plus $1 + \left\lfloor \frac{a+D_i-D_j}{T_j} \right\rfloor$ ont une échéance inférieure ou égale à d. D'où l'interférence totale des tâches plus prioritaires que τ_i est donnée par :

$$K_i(a,t) = \sum_{j \neq i \ \wedge \ D_j \leq a+D_i} \min \left\{ \left\lceil \frac{t}{T_j} \right\rceil, 1 + \left\lfloor \frac{a + D_i - D_j}{T_j} \right\rfloor \right\} C_j \tag{1.20}$$

La charge processeur $I_i(a,t)$ associée aux instances de τ_i est calculée en fonction de la valeur de t par rapport à a et $S_i(a)$.

$$I_i(a,t) = \min \left\{ \left\lceil \frac{t - S_i(a)}{T_i} \right\rceil, 1 + \left\lfloor \frac{a}{T_i} \right\rfloor \right\} C_i \ si \ t > S_i(a), \ 0 \ sinon \tag{1.21}$$

La pire charge processeur totale est alors

$$W_i(a,t) = K_i(a,t) + I_i(a,t) \tag{1.22}$$

La longueur de la période d'activité $L_i(a)$ associée à d (l'échéance de l'instance, de τi, activée à la date a) peut être calculée itérativement, en utilisant la fonction de la charge processeur W_i. La suite sera initialisée par une instance de chaque tâche τ_j ($j \neq i$) ayant un délai critique inférieur ou égal à d, et une instance de τ_i si elle est activée au début de la période d'activité (i.e. $S_i(a) = 0$).

$$\begin{aligned} L_i^{(0)}(a) &= b \times C_i \sum_{j \neq i \ \wedge \ D_j \leq a+D_i} C_j \\ L_i^{(k+1)}(a) &= W_i(a, L_i^{(k)}(a)) \end{aligned} \tag{1.23}$$

L'équation 1.23 converge vers un point fixe si, et seulement si, la charge processeur est inférieure ou égale à 1 ($\sum \frac{C_i}{T_i} \leq 1$) [102]. $L_i(a)$ représente la date de fin d'exécution de l'instance, de τ_i, activée à la date a. Si $L_i(a) < a$ alors l'instance de τ_i considérée dans le scénario (d'échéance $a + D_i$) n'appartient pas à cette période d'activité. D'où le temps de réponse $r_i(a)$ de cet instance est donné par :

$$r_i(a) = \max \{C_i, L_i(a) - a\} \tag{1.24}$$

1.5.2.1 Détermination des scénarii

Afin de trouver le pire temps de réponse, on doit identifier les dates d'activation a possibles de τ_i qui peuvent engendrer le pire cas. Soit L la plus longue période d'activité calculée par l'équation 1.8. L représente alors une borne supérieure du test, c'est-à-dire les dates a, à considérer dans le test, doivent être dans l'intervalle $[0, L - C_i)$. [102] montre que $L_i(a)$ est maximum lorsque $d = a + D_i$. C'est-à-dire lorsque l'échéance de τ_i coïncide avec l'échéance d'une autre tâche dans la période d'activité ou bien à considérer les dates d'activation a de τ_i lorsqu'elle est activée au début de la période d'activité (i.e. $S_i(a) = 0$)

$$A = \left\{ k.T_j + D_j - D_i \ , \ j = 1, ..., n \ , \ k = 1, 2, ..., \left\lfloor \frac{L - C_i + D_j}{T_j} \right\rfloor \right\} \tag{1.25}$$

Cet ensemble limite alors les dates d'activation de τ_i possibles dans la période d'activité qui doivent être testées. Donc le pire temps de réponse R_i de τ_i est le maximum des temps de réponses obtenues dans ces scénarii

$$R_i = \max_{a \in A} r_i(a) \tag{1.26}$$

1.5.2.2 Prise en compte des gigues

En prenant en compte les gigues des tâches, la plus longue période se produit lorsque toutes les tâches s'activent à la même date après avoir retardées par une valeur maximum de gigue (figure 1.7), définissant l'instant critique. Et après l'instant critiques les tâches arrivent à leur rythme maximal (i.e. avec gigue nulle). Les conditions d'analyse présentées ci-dessus restent valides pour les tâches avec gigues [102]. Il suffit seulement une adaptation des formules pour déterminer les dates d'activation des tâches, en prenant en compte les gigues.
La première instance de chaque tâche τ_j s'active effectivement à la date $t = 0$ après un retard de J_j (i.e. elle est activé à la date $-J_j$). Et les instances suivantes s'activent aux dates $K.T_j - J_j$, avec k un entier naturel (figure 1.7). Donc L'équation 1.7 qui évalue la charge processeur devient :

$$W(t) = \sum_{j=1}^{n} \left\lceil \frac{t + J_j}{T_j} \right\rceil .C_j \tag{1.27}$$

Également, le nombre d'instances avec des échéances avant ou à une date t est donné par :

$$1 + \left\lfloor \frac{(t + J_j) - D_j}{T_j} \right\rfloor \tag{1.28}$$

Si une instance d'une tâche τ_i est activée a une date a (i.e. avec date d'échéance $d = a + D_i$) dans la période d'activité, sa première instance est alors activée à la date $S_i(a)$.

$$S_i(a) = a + J_i - \left\lfloor \frac{a + J_i}{T_i} \right\rfloor T_i \tag{1.29}$$

La charge processeur totale causée par toutes les tâches plus prioritaires que τ_i (instances ayant une échéance avant ou à la date t) est :

$$W_i(a,t) = \sum_{j \neq i \ \wedge \ D_j \leq a + D_i + J_j} \min \left\{ \left\lceil \frac{t + J_j}{T_j} \right\rceil , 1 + \left\lfloor \frac{a + D_i + J_j - D_j}{T_j} \right\rfloor \right\} C_j + I_i(a,t) \tag{1.30}$$

Où $I_i(a,t)$ la charge processeur causée par les instances de τ_i.

$$I_i(a,t) = \min\left\{\left\lceil \frac{t - S_i(a) + J_i}{T_i} \right\rceil, 1 + \left\lfloor \frac{a + J_i}{T_i} \right\rfloor\right\} C_i \ si\ t > S_i(a),\ 0\ sinon \tag{1.31}$$

La longueur de la période d'activité associée à une échéance $d = a + D_i$ est calculée par la recherche du plus petit point fixe en adaptant la suite suite récurrente 1.23. $b = 1$ si $S_i(a) = 0$ sinon $b = 0$.

$$\begin{aligned} L_i^{(0)}(a) &= b \times C_i \sum_{j \neq i \ \wedge \ D_j \leq a + D_i + J_j} C_j \\ L_i^{(k+1)}(a) &= W_i(a, L_i^{(k)}(a)) \end{aligned} \tag{1.32}$$

$L_i(a)$ représente la date de fin d'exécution de l'instance de τ_i activée à la date a. Nous pouvons alors déduire le temps de réponse $r_i(a)$ de τ_i.

$$r_i(a) = \max\left\{J_i + C_i,\ L_i(a) - a\right\} \tag{1.33}$$

Pour obtenir le pire temps de réponse, nous devons calculer le temps de réponse $r_i(a)$ pour toute date d'activation $a \in A$. Notons les dates a à considérer et qui doivent nécessairement appartenir à l'intervalle $[-J_i, L - J_i - C_i)$ (période d'activité). Ces dates d'activation sont choisies d'une manière que l'échéance de l'instance $d = a + D_i$ coïncide avec l'échéance d'une instance d'une autre tâche ou bien de considérer les dates d'activation de τ_i telles que la première activation arrive au début de la période d'activité (i.e. $S_i(a) = 0$). L'ensemble A est donc donné par :

$$A = \{a/a = k.T_j + D_j - J_j - D_i\ ,\ j = 1, ..., n\ ,\ k \in N, KT_j + D_j - J_j \leq L - C_i - J_i\} \tag{1.34}$$

Le pire temps de réponse est alors le maximum parmi tous les temps obtenus pour toutes les dates d'activation considérées a.

1.5.2.3 Prise en compte des ressources partagées

Lorsque les tâches partagent des ressources, nous devons prendre en compte un nouveau paramètre appelé facteur de blocage B_i. La durée maximum de blocage est calculée suivant le protocole gérant l'accès aux ressources. La charge processeur ne change pas par le partage de ressources et le phénomène d'inversion de priorité va modifier seulement l'ordonnancement EDF. La longueur de la plus longue période d'activité n'est pas sensible aux partages des ressources. Dans l'analyse de temps de réponse d'une tâche analysée τ_i, parmi les tâches les plus prioritaires, nous devons prendre en compte le facteur de blocage de la tâche τ_k dont la valeur $D_k - J_k$ est la plus grande [102]. L'indice k de la tâche à considérer, pour une date d donnée, est identifié par la fonction $K(d)$.

$$K(d) = \max\left\{k : D_k - J_k \leq d\right\} \tag{1.35}$$

Le facteur de blocage va être pris en compte dans le calcul de la longueur de la période d'activité associée à une échéance $d = a + D_i$. On obtient

$$L_i^{(n+1)}(a) = W_i\left(a, L_i^{(n)}(a)\right) + B_{k(a+D_i)} \tag{1.36}$$

$B_{k(a+D_i)}$ est le facteur de blocage causé par la tâche considérée τ_k (équation 1.35), pour la date $d = a + D_i$. De la même manière, il suffit d'intégrer le facteur de blocage dans le calcul, le temps de réponse pour une date d'activation a est donné par :

$$r_i(a) = \max\{J_i + C_i + B_i,\ L_i(a) - a\} \tag{1.37}$$

L'ensemble A des dates d'activation a à considérer est le même que celui défini par l'équation 1.34. Ainsi le pire temps de réponse est le maximum des temps obtenus pour chacune des dates d'activations $a \in A$.

$$R_i = \max_{a \in A} r_i(a) \tag{1.38}$$

1.6 Analyse de la demande processeur

Afin de valider temporellement un système de tâches, la technique d'analyse de la demande processeur, comme l'analyse de temps de réponse, repose sur la caractérisation de pire comportement de l'application temps réel. Cette technique donne une réponse sur l'ordonnançabilité globale du système. Au contraire de l'analyse de temps de réponse qui est appliquée sur chacune des tâches du système.

Définition 4 *[90] L'analyse de la demande processeur revient à tester pour tout intervalle de temps $[t_1, t_2]$ que la durée maximum cumulée (ou une borne supérieure) des exécutions des requêtes qui ont leur réveil et leur échéance dans l'intervalle est inférieure à $t_2 - t_1$ (c-à-d., n'excède pas la longueur de l'intervalle).*

1.6.1 Priorités fixes

[52] propose une condition nécessaire et suffisante dans le cas où les tâches sont simultanées (synchrones)

Théorème 13 *[52] Soit un système S composé de n tâches indépendantes, périodiques à échéance sur requêtes et synchrone, avec $T_1 \leq T_2 ... \leq T_n$. S est ordonnançable par RM si et seulement si :*

$$\forall i, 1 \leq i \leq n : \min_{t \in S_i} \sum_{j=1}^{i} \frac{C_j}{t} \left\lceil \frac{t}{T_i} \right\rceil \leq 1, \tag{1.39}$$

$$avec \quad S_i = \left\{ kT_j,\ 1 \leq j \leq i,\ k = 1, ..., \left\lfloor \frac{T_i}{T_j} \right\rfloor \right\}$$

Le théorème 14 donne une condition nécessaire et suffisante [53] d'ordonnançabilité, pour l'algorithme DM. Cette condition est l'extension du test de [52], aux cas de tâches à échéances inférieures aux périodes.

Théorème 14 *[53] Soit un système S composé de n tâches indépendantes, périodiques à échéances contraintes ($D_i \leq T_i$), avec $D_1 \leq D_2 ... \leq D_n$. S est ordonnançable par DM si et seulement si :*

$$\forall i, 1 \leq i \leq n : \min_{t \in S_i} \sum_{j=1}^{i} \frac{C_j}{t} \left\lceil \frac{t}{T_i} \right\rceil \leq 1, \tag{1.40}$$

$$avec \quad S_i = \{D_i\} \cup \left\{ kT_j,\ 1 \leq j \leq i,\ k = 1, ..., \left\lfloor \frac{D_i}{T_j} \right\rfloor \right\}$$

1.6.2 Priorités dynamiques (EDF)

L'analyse de la demande processeur est généralement utilisée pour l'analyse d'ordonnancement produit par EDF. [15] montre qu'un système de tâches sporadiques est ordonnançable par EDF si le système synchrone correspondant est ordonnançable. L'analyse est alors basée sur la fonction de la demande processeur causée par des tâches activées et devant être terminées dans l'intervalle $[t_1, t_2]$.
Notons $dbf(t_1, t_2)$ la demande processeur causée par des tâches activées et devant être terminées dans l'intervalle $[t_1, t_2]$; c-à-d les instances avec échéance avant ou à la date t [15, 10, 16, 20].

$$dbf(t_1, t_2) = \sum_{i=1}^{n} n_i(t_1, t_2) C_i \tag{1.41}$$

Où $n_i(t_1, t_2)$ le nombre d'instances de chaque tâche τ_i dont l'activation et l'échéance est dans l'intervalle $[t_1, t_2]$.
Une condition nécessaire d'ordonnançabilité est de vérifier que la demande processeur causée par les tâches dans tout intervalle de temps est toujours inférieure ou égale à la longueur de l'intervalle :

$$\forall 0 \leq t_1 < t_2 : dbf(t_1, t_2) \leq t_2 - t_1 \tag{1.42}$$

Baruah ([15]) définit un intervalle d'étude pour EDF débutant à l'instant critique 0, jusqu'à $H = ppcm(T_i)$ pour des tâches périodiques synchrone, ou jusqu'a $\Phi + 2H$ (avec $\Phi = \max\{r_i\}$) pour les tâches asynchrones. H est appelée hyper-période puisque l'ordonnancement est périodique de période H. Notons que cet intervalle peut être réduit [41, 25].

Lemme 2 *[16] Un système de tâches Γ est ordonnançable sur une architecture monoprocesseur si et seulement si :*

1. $U \leq 1$ *et*
2. $\forall 0 \leq t_1 < t_2 \leq \Phi + 2H : dbf(t_1, t_2) \leq t_2 - t_1$

L'analyse repose sur le constat qu'un dépassement d'échéance ne survient jamais lorsque le processeur est oisif [90]. Il est prouvé que le premier dépassement d'échéance s'il existe, se produit dans la plus longue période d'activité. L'analyse d'ordonnançabilité est alors restreint dans la plus longue période d'activité en vérifiant que toutes les échéances des tâches sont respectées dans cette période d'étude qui débute à l'instant $t = 0$. [16] a prouvé que la longueur de la période d'activité est limitée dans le cas $U < 1$ par

$$\frac{U}{1-U} \max_{i=1..n} (T_i - D_i) \tag{1.43}$$

Ce qui permet d'effectuer l'analyse suivant :

Théorème 15 *[16] Un système de tâches synchrones* Γ *est ordonnançable par EDF sur une architecture monoprocesseur si et seulement si :*

$$\forall L^* \leq L, dbf(0, L^*) \leq L^* \tag{1.44}$$

tel que L^* *est un échéance d'une instance d'une tâche et* L *la longueur de la plus longue période d'activité*

Afin de déterminer le nombre des instances $n_i(0,t)$ d'une tâche τ_i dont les échéances surviennent dans un intervalle $[0,t]$ dans la période d'activité, nous identifions la dernière instance de τ_i avec les deux inégalités suivantes :

$$\begin{aligned} r_i + (n_i - 1)T_i + D_i \leq t \quad &\Rightarrow \quad n_i \leq \frac{t - r_i - D_i}{T_i} + 1 \\ r_i + n_i T_i + D_i > t \quad &\Rightarrow \quad n_i > \frac{t - r_i - D_i}{T_i} \end{aligned}$$

Notons que $n_i \geq 0$. D'où $n_i = \max\left(0, \left\lfloor \frac{t-r_i-D_i}{T_i} \right\rfloor + 1\right)$. La fonction de la demande processeur est définie alors par :

$$dbf(0,t) = \sum_{i=1}^{n} \max\left(0, \left\lfloor \frac{t - r_i - D_i}{T_i} \right\rfloor + 1\right) \times C_i \tag{1.45}$$

La complexité de l'analyse est :

$$O\left(n \frac{U}{1-U} \max_{i=1..n} (T_i - D_i)\right) \tag{1.46}$$

Un test suffisant d'ordonnançabilité, moins pessimiste, avec une complexité pseudo-polynomiale, a été proposée par R. Pellizzoni et G. Lipari [74, 74, 73] pour les tâches périodiques asynchrones. Cette méthode d'analyse consiste à fixer une tâche, a priori, puis vérifier l'ordonnançabilité du système dans la période d'activité calculée en prenant en compte le décalage d'activation de toutes les tâches par rapport à la tâche fixée. D'où le test nécessite de vérifier l'ordonnançabilité du système pour toutes les scénarii possibles correspondant à chacune des tâches du système. Le pessimisme, de l'analyse, peut être réduit en fixant plus d'une tâche à la fois, mais cette procédure augmente la complexité de méthode.

1.7 Conclusion

Nous avons présenté, dans ce chapitre, une introduction aux systèmes temps réel, où notre présentation s'est focalisée sur la validation temporelle des systèmes. Plusieurs facteurs sont pris en compte pour valider un système :

- L'architecture matérielle (monoprocesseur, multiprocesseurs, ...).
- Modes d'exécution des tâches (préemptif, non-préemptif).
- Modèle de tâches (périodiques, sporadiques,...).
- Politique d'ordonnancement (en-ligne, hors-ligne).

Dans ce chapitre, nous nous sommes intéressés à l'ordonnancement en-ligne, des tâches indépendantes de L&L [55], dans une architecture monoprocesseur. L'ordonnancement en-ligne est basée sur des algorithmes à priorités (fixes ou dynamiques), chargés d'attribuer le processeur aux tâches, au cours de la vie du système, en fonction de leurs priorités. Le test d'ordonnancabilité consiste alors à donner une réponse concernant l'ordonnançabilité d'un système, par un algorithme donné. Plusieurs techniques analytiques sont utilisées pour ce but :

- Analyse du facteur d'utilisation : les résultats de base, obtenus pour les différents algorithmes d'ordonnancement, sur l'ordonnancement des tâches indépendantes sont présentées dans les sections 1.4.4 et 1.4.4.
- Analyse de temps de réponse : section 1.5 présente les différents résultats obtenus, dans les deux contextes priorités fixes et dynamiques. Ces résultats concernent les tâches indépendantes de L&L [55].
- Analyse de la demande processeur : section 1.6 présente les principaux résultats obtenus pour les tâches indépendantes sporadiques et périodiques.

Considérer le modèle de tâches indépendantes, pour certains types d'applications temps réel, rend l'application des conditions d'ordonnançabilité présentées dans ce chapitre pessimistes. Afin de prendre en compte toutes les contraintes qui modélisent finement le comportement d'une application temps réel, plusieurs modèles ont été proposés, nous citons le modèle de tâches à offset qui fait l'objet d'étude de cette thèse.

Chapitre 2

Analyse d'ordonnançabilité des tâches avec décalage d'activation (à offsets)

Résumé : *Ce chapitre présente les principaux modèles de tâches prenant en compte les relations de décalage d'activation entre les tâches, les transactions et les tâches multiframes. Après avoir présenté la motivation de proposer un tel modèle, nous détaillons les méthodes existantes d'analyse d'ordonnançabilité des transactions ordonnancées par priorités fixes et EDF. Ces méthodes sont basées sur l'analyse du pire scénario. Nous présentons ensuite les problèmes de complexité algorithmique et de pessimisme d'analyse pour les tests proposés, ainsi que la motivation du travail effectué dans cette thèse.*

2.1 Introduction

Le modèle de tâches périodiques proposé par Liu et Layland [55], suppose que les tâches sont indépendantes, c'est-à-dire qu'il n'existe aucune contrainte ou relation entre les paramètres temporels des tâches. D'après cette supposition, le pire temps de réponse d'une tâche se produit à un instant critique qui est défini par l'activation simultanée de la tâche analysée avec toutes les tâches du système (chapitre 1) . Cette condition d'ordonnançabilité est très pessimiste (dans le sens où elle peut conclure une réponse négative pour l'ordonnançabilité tandis que le système est ordonnançable en réalité) pour certaines types d'applications temps réel où l'exécution (l'activation) de certaines tâches sont liées, rendant irréaliste l'instant critique.

Dans une application de contrôle d'une drone miniature [110], par exemple, les tâches d'acquisition de données générées par des périphériques série (GPS, centrale inertielle) sont liées par une relation de décalage d'activation qu'on appelle Offset. Chaque ensemble de tâches (par exemple celles générées par un GPS) sont réveillées après un temps de décalage par rapport à l'arrivée de la première trame. Ce dernier est alors l'évènement déclencheur du processus d'acquisition où chaque tâche est chargée d'acquérir un bloc de données. Ces tâches sont activées l'une après l'autre. Par conséquent les tâches d'acquisition des différents blocs d'une trame ne pourront jamais être activées simultanément, ce qui rend la condition d'ordonnançabilité de $L\&L$ (de Liu et Layland) pessimiste pour ce genre de tâches. Afin de prendre en compte ces relations de décalage, un modèle de transactions est utilisé pour regrouper un ensemble de tâches déclenchées par un même évènement.

Une transaction est alors une collection de tâches liées par des relations d'offset. Ces tâches sont liées soit par une fonction effectuée collectivement, soit par une calendrier de temps partagé selon lequel il convient de regrouper ces tâches dans une seule entité [107]. Notons que l'analyse prenant en compte les relations d'offsets englobe l'analyse classique. i.e. s'il n'existe pas de relation d'offset entre les tâches, l'analyse est équivalente à celle des tâches indépendantes de L&L.

Nous présentons le modèle de tâches à offsets dans la section 2.2, ainsi que les différents techniques d'analyse d'ordonnaçabilité proposées dans le contexte des priorités fixes (section 2.3) comme en priorités dynamiques (section 2.4). Quelques modèles particuliers sont présentés dans la section 2.5 : transactions série, transactions monotoniques et les tâches à suspension. Dans la section 2.6 nous présentons le modèle de tâches multiframes et dans la section 2.7 une généralisation de ce modèle est donnée. Le chapitre est conclu par une présentation de différents problèmes et difficultés de l'analyse d'ordonnançabilités de tous les modèles exposés.

2.2 Modèle de tâches à offset (Transactions)

Dans le contexte des tâches à offsets, un système Γ de tâches est composé d'un ensemble de $|\Gamma|$ transactions Γ_i activées par des évènements externes, avec $1 \leq i \leq |\Gamma_i|$ (tel que $|\Gamma_i|$ est le nombre d'éléments dans l'ensemble Γ_i). Cet activation par évènement externe justifie le fait qu'une transaction est non concrète (date d'arrivée d'évènement inconnue) mais qu'à l'intérieur , chaque

tâche composant la transaction est bien connue (date de réveil relatif à la date de l'évènement déclencheur de la transaction).

Formellement un système de transactions est défini comme suit :

$$\begin{aligned} \Gamma &: \{\Gamma_1, \Gamma_2, .., \Gamma_{|\Gamma|}\} \\ \Gamma_i &: \{\tau_{i1}, \tau_{i2}, ..., \tau_{i|\Gamma_i|}, T_i\} \\ \tau_{ij} &: < C_{ij}, O_{ij}, D_{ij}, J_{ij}, B_{ij}, P_{ij} > \end{aligned}$$

Chaque transaction Γ_i (Figure 2.1) est constituée d'un ensemble de $|\Gamma_i|$ tâches τ_{ij} possédant la même période T_i , avec $1 < j \leq |\Gamma_i|$. T_i est alors le temps séparant deux activations successives de la transaction (par conséquent de deux activations successives de chaque tâche τ_{ij} de Γ_i). Remarquons que cette période est une « pire période », i.e. un délai minimal entre deux activations successives (modèle sporadique, voir section 1.3.3.2).

Une tâche τ_{ij} est définie par son pire temps d'exécution (WCET pour Worst Case Execution Time) C_{ij}, un intervalle de temps séparant la date d'activation de la transaction et la date d'activation de la tâche, appelé offset O_{ij}, un délai critique D_{ij}, un gigue sur activation J_{ij}, un facteur de blocage maximum B_{ij}, et une priorité P_{ij} (si une politique d'ordonnancement à priorités fixes est considérée). B_{ij} modélise le pire temps de blocage induit par un éventuel partage de ressources (suivant le protocole de gestion de ressources utilisé [97, 21, 9]) et J_{ij} permet de prendre en compte un retard sur l'activation de la tâche dans le cas par exemple d'application distribuée (retard lié à l'attente d'un message par exemple). Une tâche peut être donc activée à tout instant entre O_{ij} et $O_{ij} + J_{ij}$ après activation de la transaction et elle doit terminer son exécution avant la date $O_{ij} + D_{ij}$ après l'activation de la transaction. Notons que les échéances des tâches sont quelconques (non reliées aux périodes) et que les tâches sont non réentrantes. Afin de modéliser certaines applications dans l'aérospatial où les données sont rassemblées périodiquement et à chaque groupe de n périodes certains traitements supplémentaires sont effectués, [107] dote chaque tâche τ_{ij} d'un paramètre supplémentaire e_{ij} qui détermine le nombre d'activations de la transaction durant lesquels une exécution de τ_{ij} est permise (i.e. une activation de τ_{ij} pour e_{ij} activations de Γ_i), dans ce cas la période de la tâche τ_{ij} est $e_{ij} \times T_i$.
Dans tout le document, nous supposons cependant que $e_{ij} = 1$. Dans le cadre d'une étude d'ordonnançabilité nous étudions le pire cas, qui correspond dans le sporadiquement périodique à un réveil strictement périodique la plus petite période T_i. Supposons qu'une transaction Γ_i soit activée à la date $t = 0$, durant la vie du système plusieurs instances d'une tâche τ_{ij} sont exécutées. La k^{eme} instance (avec k un nombre naturel positif) est donc caractérisée par :

- *Date d'activation :* entre $O_{ij} + (k-1)T_i$ et $O_{ij} + J_{ij} + (k-1)T_i$.
- *Échéance :* à la date $O_{ij} + (k-1)T_i + D_{ij}$.

La figure 2.1 présente un exemple de transaction Γ_i contenant trois tâches τ_{i1}, τ_{i2} et τ_{i3} avec des WCET égale à $C_{i1} = 3$, $C_{i2} = 2$ et $C_{i3} = 4$ respectivement. Les trois tâches ont la même période $T_i = 16$ (période de la transaction). La tâche τ_{i1} (Resp τ_{i2} et τ_{i3}) s'active à l'instant $O_{i1} = 1$

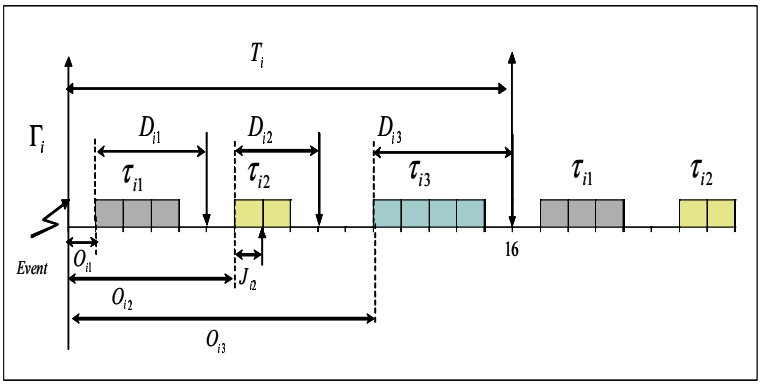

FIG. 2.1 – Exemple d'une transaction.

(Resp $O_{i2} = 6$, $O_{i3} = 11$) après la date d'activation de la transaction.

Nous nous intéressons dans la suite à l'analyse d'ordonnançabilité des transactions non concrètes exécutées sur une architecture monoprocesseur. D'après la définition d'une transaction, certaines tâches d'une même transaction ne peuvent jamais s'activer simultanément, ce qui rend la condition d'ordonnançabilité proposées par Liu et Layland pour des tâches indépendantes [55, 102] très pessimiste (dans le sens où elle fournit des réponses négatives sur l'ordonnançabilité pour des systèmes qui sont ordonnançables en réalité). Ainsi sur l'exemple de la figure 2.1, si l'on considère avec affectation des priorités DM, un système composé uniquement des tâches de Γ_i est déclaré non ordonnançable si l'on considère le modèle de L&L.

Plusieurs travaux ont été effectués sur l'analyse d'ordonnançabilité des transactions. Tous les travaux trouvés dans la littérature portent sur l'analyse de temps de réponse. Les premiers travaux ont été proposés par Tindell [106, 107, 109], dans lesquels il a défini le modèle de transactions et il a identifié le pire scénario d'exécution de la tâche analysée pour calculer son pire temps de réponse. Il a prouvé qu'un instant critique se produit lorsque une tâche plus prioritaire que la tâche analysée de chaque transaction s'active simultanément à la tâche analysée. D'après la définition de l'instant critique, plusieurs difficultés sont rencontrées afin de calculer le pire temps de réponse d'une tâche analysée τ_{ua}.

- Plusieurs instants critiques sont possibles. Parmi ces instants, lequel induit le pire temps de réponse de la tâche analysée ?
- Comment calculer l'interférence réelle (retard causé par) d'une transaction sur le temps de réponse de la tâche analysée ?
- Trouver une méthode d'analyse avec une complexité raisonnable sur des systèmes de taille réaliste.

Afin de bien comprendre les différentes méthodes d'analyse proposées, ainsi que leurs points forts et leurs lacunes, nous divisons notre étude en deux grandes sections suivant l'algorithme d'ordonnancement utilisé. La section 2.3 présente l'analyse de temps de réponse des tâches à offset ordonnancées par un algorithme à priorités fixes et la section 2.4 présente celui des tâches

ordonnancées par un algorithme à priorités dynamiques (EDF).

2.3 Analyse d'ordonnançabilité des tâches à offset avec priorités fixes

Soit Γ un système composé de n transactions (Γ_i avec $i = 1..n$) où transaction Γ_i contient $|\Gamma_i|$ tâches. Notons τ_{ua} la tâche à analyser. Supposons que chaque tâche possède une priorité fixe unique. Nous donnons ci-après les notations utilisées dans la suite de ce chapitre :

Notations :
- $hp_i(\tau_{ua})$: l'ensemble des indices des tâches les plus prioritaires que τ_{ua} appartenant à la transaction Γ_i.
- R_{ua} : Le temps de réponse de la tâche sous analyse τ_{ua} (temps entre la sa date d'activation et sa date de fin d'exécution).

Dans le contexte des transactions, calculer du pire temps de réponse d'une tâche τ_{ua} repose sur l'approche générale de calcul présentée pour les tâches indépendantes qui consiste à :

1. Identifier et examiner les périodes d'activités de niveau *ua* , ainsi que les instants critiques. Ensuite calculer le temps de réponse correspondant à chacun des instants critiques ; le maximum est le pire temps de réponse de τ_{ua}.
2. Pour une période d'activité de niveau *ua* donnée, déterminer la pire interférence causée par les tâches les plus prioritaires que τ_{ua} dans la période d'activité . Cette interférence permettra d'estimer le retard que la tâche τ_{ua} peut subir.
3. Déterminer la longueur de la période d'activité (où celle associée à une instance donnée) afin de déduire le temps de réponse d'une instance de τ_{ua}

2.3.1 Identification du Pire scénario

Nous savons que le pire temps de réponse d'une tâche τ_{ua} se produit toujours dans une pire période d'activité de niveau *ua*. Afin de pouvoir identifier le pire scénario d'exécution de τ_{ua}, il est indispensable de caractériser les pire périodes d'activité durant lesquelles le pire temps de réponse peut se produire, et ainsi limiter le nombre de périodes à considérer.

Théorème 16 *[70] La contribution pire cas d'une transaction Γ_i lors d'un instant critique d'une tâche analysée τ_{ua} est obtenue quand la première activation d'une tâche τ_{ic} de chaque transaction Γ_i plus prioritaire que τ_{ua} coïncide avec cet instant critique après avoir été retardée par sa gigue maximale J_{ic}.*

[107, 70] a montré qu'une pire (plus longue) période d'activité de niveau *ua* commence à l'instant où une tâche τ_{ic_i} plus prioritaire que τ_{ua} de chaque transaction Γ_i ($i = 1...n$ avec Γ_u inclue) sont activées simultanément après avoir été retardées par une valeur maximale de leurs gigues, et après le début de la période d'activité, les tâches arrivent à leur rythme maximale (i.e. avec gigue nulle), et période T_i.
Le théorème 17 détermine les instances d'une tâche qui peuvent contribuer à une période d'activité de niveau *ua*.

Théorème 17 *[107] Le temps d'exécution d'une instance d'une tâche τ_{ij} plus prioritaire que τ_{ua} activée avant le début de la période d'activité de niveau ua (ua étant le niveau de priorité de τ_{ua}) peut être ignoré pourvu qu'une période d'activité supplémentaire démarrant au réveil de la tâche τ_{ij} soit examinée.*

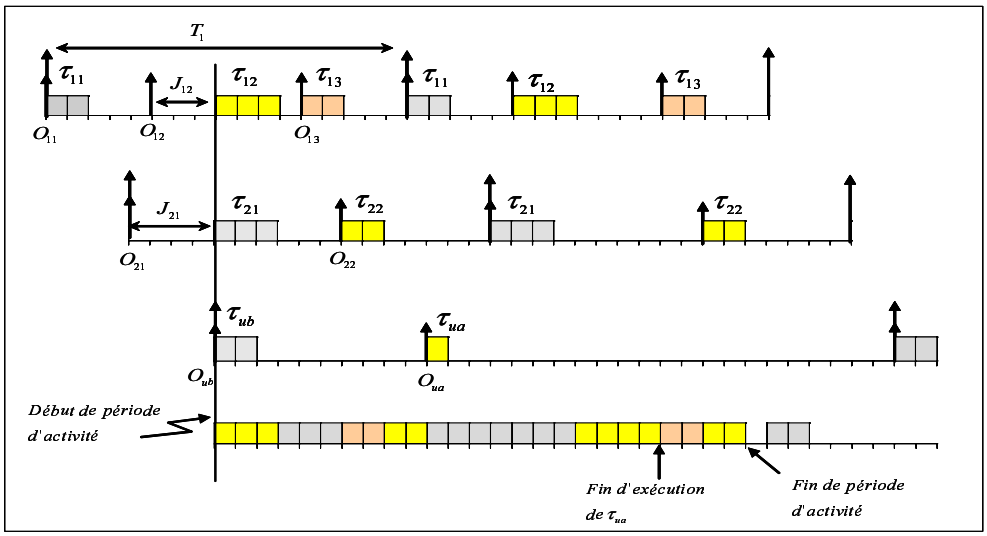

FIG. 2.2 – période d'activité de niveau *ua*.

Considérons un système Γ, de la figure 2.2, composé de trois transactions Γ_1, Γ_2 et Γ_u. Soit τ_{ua} la tâche à analyser avec la plus faible priorité. La figure 2.2 illustre une période d'activité de niveau *ua* lorsque les tâches τ_{12}, τ_{21} et τ_{ub} s'activent simultanément après avoir été retardées par une valeur maximale de gigue $J_{12} = 3$, $J_{21} = 4$ et $J_{ub} = 0$ respectivement. D'après le théorème 17, la première instance de τ_{11} ne contribue pas à la periode d'activité considérée car elle s'active avant même si elle se fait retarder par sa gigue maximale. Rappelons que les transactions sont non concrètes, donc d'autres périodes d'activité sont possibles changeant la combinaison des tâches activées simultanément au début de la période d'activité (tâches initiant l'instant critique). Pour notre exemple $|\Gamma_1| \times |\Gamma_2| \times |\Gamma_u| = 3 \times 2 \times 2 = 12$ périodes d'activité (combinaisons) sont possibles. Notons qu'au contraire des tâches indépendantes de Liu et Layland pour lesquelles l'instant critique coïncide avec l'activation de la tâche analysée, dans le contexte des tâches à offset l'instant critique d'une tâche τ_{ua} coïncide avec le début de la période d'activité (i.e. l'instant critique ne coïncide pas forcement avec l'activation de la tâche analysée τ_{ua}). [110] donne un exemple simple d'une tâche analysée d'une transaction Γ_u pour laquelle l'instant critique (période d'activité) produisant le pire temps de réponse coïncide avec l'activation d'une autre tâche de la même transaction Γ_u.

A ce stade d'étude, nous connaissons les pires scénarii (les périodes d'activité) à considérer, ainsi les instants critiques que nous devons étudier pour trouver le pire temps de réponse. Dans la section suivante nous présentons la procédure du calcul de temps de réponse d'une tâche τ_{ua} pour chacun des scénarii possibles.

2.3.2 Facteurs d'interférence

Supposons un instant critique initié par l'activation d'une tâche τ_{ic} de chaque transaction Γ_i. Le temps de réponse d'une instance de τ_{ua} est la durée entre sa date d'activation et sa date de fin d'exécution. La date de fin d'exécution correspond à la première date dans la période d'activité où la charge processeur est égale à la longueur de l'intervalle de temps considéré, on parle de point creux.

Différentes sources de charge processeur (interférence) peuvent participer à retarder l'exécution de τ_{ua}, nous les citons ci-après :

- Interférence causée par des tâches ayant des priorités supérieures à celle de τ_{ua} (pour chacune des transactions).
- Interférence causée par les instances de τ_{ua} activée avant l'instance analysée (dans le cas où le délai critique peut être supérieur à la période).
- interférence causée par des tâches moins prioritaires partageant des ressource critiques avec τ_{ua}. Cette interférence est bornée par une valeur B_{ua}, estimée selon le protocole d'accès aux ressources considéré.

2.3.3 Fonction d'interférence

Pour une transaction Γ_i, soit τ_{ic} la tâche dont l'activation coïncide avec l'instant critique (on parle de tâche candidate à l'instant critique pour Γ_i), supposons une tâche τ_{ij} appartenant à la transaction Γ_i, plus prioritaire que la tâche analysée τ_{ua}. Dans une période d'activité de longueur t, plusieurs instances de la tâche τ_{ij} peuvent être exécutées, d'où l'interférence de la tâche τ_{ij} pour cet instant critique est représentée par les différentes instances de τ_{ij}. Palencia et Harbour [70] distinguent ces instances en trois classes :

- Set0 : Les instances qui sont activées avant l'instant critique (début de la période d'activité) même avec un retard d'activation par une valeur maximale du gigue. Ces instances correspondent au résultat du théorème 17 et n'auront aucune contribution dans la période d'activité.
- Set1 : Les instances activées à l'instant critique après avoir été retardées par une certaine valeur de la gigue pour coïncider avec l'instant critique.
- Set2 : Les instances activées après l'instant critique (dans la période d'activité).

Nous nous intéressons donc à l'interférence causée par les deux ensembles Set1 et Set2. Dans l'exemple illustré par la figure 2.3 l'instant critique est initié par l'activation de l'instance $\tau_{i1}^{(3)}$ qui a été retardée par $J_{i1} = 4$ unités de temps. On remarque qu'il y a deux instances $\tau_{i2}^{(1)}$ et $\tau_{i2}^{(2)}$ qui sont retardées par une valeur de gigue ($\leq J_{12}$) pour que leurs activations coïncident avec l'instant critique. Ces deux instances appartiennent alors à l'ensemble Set1 de τ_{i2}. Les tâches activées après l'instant critique comme $\tau_{i2}^{(3)}$ et $\tau_{i2}^{(4)}$ forment l'ensemble Set2.

Signalons que l'interférence des instances de l'ensemble Set1 est indépendante de t (dépend de la valeur maximale de la gigue). Tandis que celle de l'ensemble Set2 dépend de la longueur de

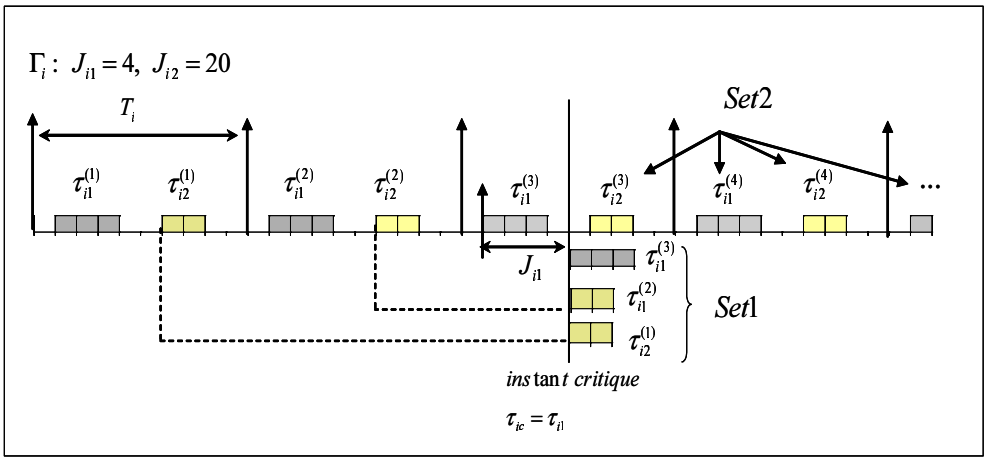

FIG. 2.3 – Ensembles des instances contribuant à une période d'activité.

la période d'activité qui débute à l'instant critique. La théorème 18 donne le mode d'activation pire cas des tâches plus prioritaires que τ_{ua}.

Théorème 18 *[70] Soit t_c l'instant critique d'une tâche τ_{ua} et ϕ le temps entre l'activation de la transaction Γ_i et l'instant critique t_c. La contribution pire cas de la tâche τ_{ij} apparaît quand les instances de l'ensemble Set1 ont une gigue telle qu'elles s'activent toutes à l'instant critique tandis que les activations dans l'ensemble Set2 ont une gigue nulle.*

2.3.4 Calcul de temps de réponse pour un instant critique

2.3.4.1 Interférence des autres transactions sur la tâche analysée

Nous connaissons le mode d'activation des instances d'une tâche τ_{ij} par rapport un instant critique ($t = 0$) initié par τ_{ic} ; cela ne suffit pas pour calculer le nombre d'instances de chaque ensemble contribuant au temps de réponse de τ_{ua}. Notons que par la suite, les indices ijc utilisés correspondent à la tâche τ_{ij} dans un scénario où τ_{ic} est la tâche de Γ_i dont l'activation correspond à l'instant critique étudié. La phase Φ_{ijc} est définie comme l'intervalle de temps entre l'instant critique et la première activation de τ_{ij} (avec gigue nulle) après l'instant critique (date d'activation de la première instance dans Set2). Φ_{ijc} sera utilisée comme repère pour calculer le nombre d'instances dans les ensembles Set1 et Set2.

$$\Phi_{ijc} = (O_{ij} - (O_{ic} + J_{ic})) \mod T_i \tag{2.1}$$

D'après la définition d'instant critique et d'après le théorème 18 l'instant critique initié par la tâche τ_{ic} survient à la date $(O_{ic} + J_{ic}) \mod T_i$ car τ_{ic} doit coïncider avec l'instant critique après avoir été retardé par la valeur maximale de la gigue. D'où la première instance de la tâche τ_{ij} de Set2 (activée après l'instant critique avec gigue nulle) est activée à la date Φ_{ijc} après l'instant critique, et les prochaines activations surviennent périodiquement chaque T_i.

2.3.4.2 Exemple :

La figure 2.4 illustre graphiquement les valeurs des phases des tâches de la transaction Γ_1 pour les différentes tâches candidates à initier l'instant critique. Dans la figure 2.4(b) le début de la tâche τ_{11} coïncide avec l'instant critique, d'où ce dernier (l'instant critique) survient à la date $O_{11}+J_{11}=2+1=3$. L'ensemble Set1 contient une seule instance de τ_{11}, la première instance de τ_{11} de Set2 s'active à la date $\Phi_{111}=15$, aucune instance de τ_{12} n'existe dans Set1, et la première instance dans l'ensemble Set2 s'active à la date $\Phi_{121}=6$ après l'instant critique.

$$\Phi_{111} = (O_{11} - (O_{11} + J_{11})) \mod T_1 = (2 - (2+1)) \mod 16 = 15$$
$$\Phi_{121} = (O_{12} - (O_{11} + J_{11})) \mod T_1 = (9 - (2+1)) \mod 16 = 6$$

Tandis que la figure 2.4(c) illustre les phases des deux tâches de la transaction Γ_1 quand l'activation de la tâche τ_{12} coïncide avec l'instant critique.

$$\Phi_{112} = (O_{11} - (O_{12} + J_{12}) \mod T_1 = (2 - (9+4)) \mod 16 = 5$$
$$\Phi_{122} = (O_{12} - (O_{12} + J_{12}) \mod T_1 = (9 - (9+4)) \mod 16 = 12$$

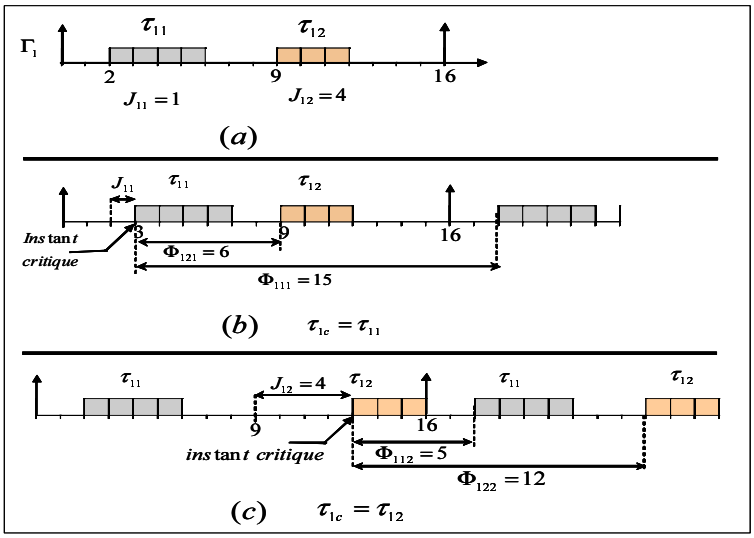

FIG. 2.4 – Phases d'activation.

Notons $W_{ijc}(t)$ l'interférence de la tâche τ_{ij} dans l'intervalle de temps t quand la tâche τ_{ic} initie l'instant critique. $W_{ijc}(t)$ est alors égale à la somme de l'interférence I_{ijc}^{Set1} des instances de l'ensemble Set1, plus l'interférence I_{ijc}^{Set2} des instances de Set2.

La partie d'interférence causée par les instances de Set1 est indépendante de la longueur de la période d'activité. Elle est principalement liée à la valeur maximale de la gigue. Cette partie d'interférence est donnée par l'équation suivante :

$$I_{ijc}^{Set1} = \left\lfloor \frac{J_{ij} + \Phi_{ijc}}{T_i} \right\rfloor C_{ij} \tag{2.2}$$

La deuxième partie est l'interférence causée par les instances de Set2 qui dépend de la longueur de la période d'activité t, son équation est donc donnée en fonction de la longueur de la période d'étude t.

$$I_{ijc}^{Set2}(t) = \left\lceil \frac{t - \Phi_{ijc}}{T_i} \right\rceil C_{ij} \tag{2.3}$$

D'où l'interférence totale $W_{ijc}(t)$ de la tâche τ_{ij} sur une tâche de plus faible priorité pendant un intervalle de temps t quand la tâche τ_{ic} initie l'instant critique est égale à la somme des interférences des deux ensembles Set1 et Set2 :

$$W_{ijc}(t) = I_{ijc}^{Set1} + I_{ijc}^{Set2}(t) \tag{2.4}$$

Par conséquent, l'interférence de la transaction Γ_i sur le temps de réponse de la tâche sous analyse τ_{ua}, quand la tâche τ_{ic} initie l'instant critique est donnée par l'addition des interférences de toutes les tâches de Γ_i ayant une priorité supérieure à celle de la tâche τ_{ua}.

$$W_{ic}(\tau_{ua}, t) = \sum_{\forall j \in hp_i(\tau_{ua})} (I_{ijc}^{Set1} + I_{ijc}^{Set2}(t)) \tag{2.5}$$

$$W_{ic}(\tau_{ua}, t) = \sum_{\forall j \in hp_i(\tau_{ua})} \left(\left\lfloor \frac{J_{ij} + \Phi_{ijc}}{T_i} \right\rfloor + \left\lceil \frac{t - \Phi_{ijc}}{T_i} \right\rceil \right) C_{ij} \tag{2.6}$$

Cette formule nous permet alors de déterminer la contribution de chacune des transactions du système dans une période d'activité, pour un instant critique donné initié par τ_{ic} pour la transaction $\Gamma_i \neq \Gamma_u$. Dans l'exemple de la figure 2.5 l'interférence de Γ_1 lorsque τ_{11} initie l'instant critique, dans un intervalle de temps de longueur $t = 18$ est obtenue par le calcul suivant :

$$\begin{aligned}
\Phi_{111} &= (O_{11} - (O_{11} + J_{11}) \mod T_1 = (2 - (2 + 0)) \mod 28 = 0 \\
\Phi_{121} &= (O_{12} - (O_{11} + J_{11}) \mod T_1 = (10 - (2 + 3)) \mod 28 = 5 \\
W_{i1}(\tau_{ua}, t) &= \left(\left\lfloor \frac{J_{11} + \Phi_{111}}{T_1} \right\rfloor + \left\lceil \frac{t - \Phi_{111}}{T_1} \right\rceil \right) C_{11} + \left(\left\lfloor \frac{J_{12} + \Phi_{121}}{T_1} \right\rfloor + \left\lceil \frac{t - \Phi_{121}}{T_1} \right\rceil \right) C_{12} \\
&= \left(\left\lfloor \frac{0 + 0}{28} \right\rfloor + \left\lceil \frac{18 - 0}{28} \right\rceil \right) 5 + \left(\left\lfloor \frac{3 + 5}{28} \right\rfloor + \left\lceil \frac{18 - 5}{28} \right\rceil \right) 6 = 11
\end{aligned}$$

Nous présentons la procédure de calcul de pire temps de réponse d'une tâche τ_{ua} pour un instant critique candidat donné. Supposons une période d'activité initiée par l'activation d'une tâche τ_{ic_i} de chaque transaction Γ_i du système.

2.3.4.3 Interférence de la transaction analysée sur la tâche analysée

Dans une période d'activité, plusieurs instances de la tâche τ_{ua} pouvant être activées à l'instar de [51] . Le maximum parmi les temps de réponse de ces instances donne le pire temps de réponse de τ_{ua}, pour le scénario d'instant critique considéré. Afin de calculer le temps de réponse de chacune de ces instances, on leur donne, en fonction de leurs dates d'activation (sans tenir en compte la valeur de la gigue), la numérotation suivante : on attribue à l'instance de la tâche τ_{ua} qui s'active dans l'intervalle $]0, T_u]$ la position (numéro) $p = 1$, et $p = 2$ à celle qui s'active à l'intervalle $]T_u, 2T_u]$ et ainsi de suite. Quant à l'instance qui s'active dans l'intervalle $]-T_u, 0]$ on attribue la position $p = 0$ et à celle de l'intervalle $]-2T_u, T_u]$ on attribue la position $p = -1$ et ainsi de suite. Il s'agit tout simplement de numéroter les instances de τ_{ua} appartenant aux deux ensembles Set1 et Set2.

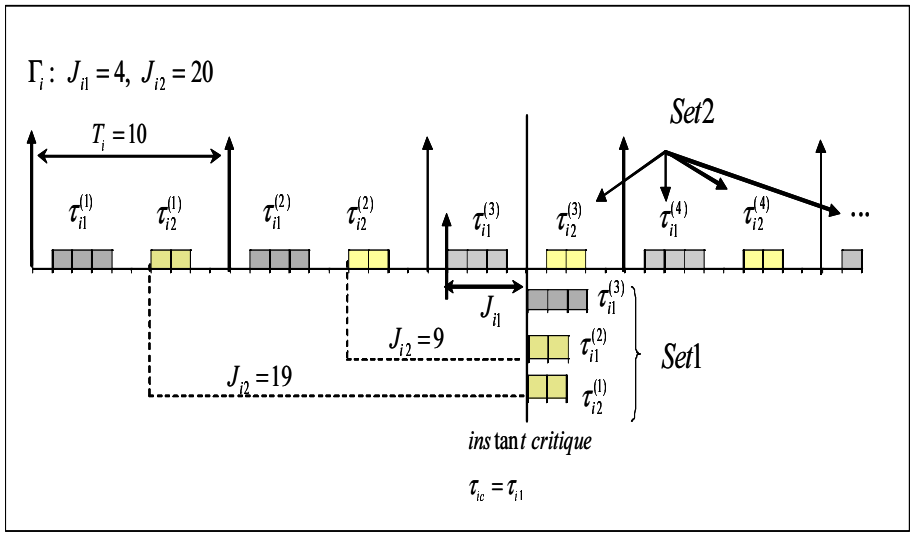

FIG. 2.5 – Instances d'une tâche analysée τ_{ua} avec numérotation.

La figure 2.5 présente la numérotation et les temps de réponse des instances de τ_{ua} activées dans la période d'activité initiée par l'activation simultanée de τ_{11} et τ_{ub}.

Donnons la position $p_{0,ua}$ à l'instance de τ_{ua} de numéro le plus faible présente dans la période d'activité.

$$p_{0,ua} = -\left\lfloor \frac{J_{ua} + \Phi_{uac_u}}{T_u} \right\rfloor + 1 \tag{2.7}$$

Dans notre exemple de la figure 2.5 nous avons

$$p_{0,ua} = -\left\lfloor \frac{9+4}{12} \right\rfloor + 1 = -1 + 1 = 0$$

Il est nécessaire de déterminer la longueur L de la période d'activité, afin de déterminer l'instance de la position la plus élevées $p_{L,ua}$ de τ_{ua}. Il est clair que plus la période d'activité est longue

plus $p_{L,ua}$ est élevée. La plus longue période d'activité est calculée itérativement en prenant en compte toutes les instances de toutes les tâches plus prioritaires y compris celles de τ_{ua}. La suite récurrente est initialisée par la durée d'exécution C_{ua} de la tâche analysée. Le calcul est arrêté lorsque le premier point fixe est trouvé.

$$L_{ua}^{(k)} = B_{ua} + \left(\left\lceil \frac{L_{ua}^{(k-1)} - \Phi_{uac_u}}{T_u} \right\rceil - p_{0,ua} + 1 \right) C_{ua} + \sum_{\forall i} W_{ic_i} \left(\tau_{ua}, L_{ua}^{(k-1)} \right) \tag{2.8}$$

Nous pouvons maintenant déterminer la valeur (de la position de la dernière instance de τ_{ua} présente dans la période d'activité) :

$$p_{L,ua} = \left\lfloor \frac{L_{ua} - \Phi_{uac_u}}{T_u} \right\rfloor \tag{2.9}$$

Pour la période d'activité présentée dans la figure 2.5, le calcul est arrêté en arrivant au premier point fixe à l'itération 4 ($L_{ua}^{(4)} = L_{ua}^{(3)}$).

$$\begin{aligned}
L_{ua}^{(0)} &= 2 \\
L_{ua}^{(1)} &= \left(\left\lceil \frac{L_{ua}^{(k-1)} - \Phi_{uac_u}}{T_u} \right\rceil - p_{0,ua} + 1 \right) C_{ua} + \sum_{\forall i} W_{ic_i} \left(\tau_{ua}, L_{ua}^{(0)} \right) \\
&= \left(\left\lceil \frac{0-4}{12} \right\rceil - 0 + 1 \right) 2 + W_{11} \left(\tau_{ua}, 2 \right) + W_{ub} \left(\tau_{ua}, 2 \right) \\
&= 2 + 5 + 3 = 10 \\
L_{ua}^{(2)} &= \left(\left\lceil \frac{10-4}{12} \right\rceil - 0 + 1 \right) 2 + W_{11} \left(\tau_{ua}, 10 \right) + W_{ub} \left(\tau_{ua}, 10 \right) \\
&= 4 + 11 + 3 = 18 \\
L_{ua}^{(3)} &= \left(\left\lceil \frac{18-4}{12} \right\rceil - 0 + 1 \right) 2 + W_{11} \left(\tau_{ua}, 18 \right) + W_{ub} \left(\tau_{ua}, 18 \right) \\
&= 6 + 11 + 6 = 23 \\
L_{ua}^{(4)} &= \left(\left\lceil \frac{23-4}{12} \right\rceil - 0 + 1 \right) 2 + W_{11} \left(\tau_{ua}, 23 \right) + W_{ub} \left(\tau_{ua}, 23 \right) \\
&= 6 + 11 + 6 = 23
\end{aligned}$$

Par conséquent, la position de la dernière instance de τ_{ua} est (figure 2.5) :

$$p_{L,ua} = \left\lfloor \frac{23-4}{12} \right\rfloor = 2$$

Après avoir déterminé toutes les instances de τ_{ua} présentes dans la période d'activité ($p = p_{0,ua}, ..., p_{L,ua}$), il suffit de calculer leurs temps de réponse et choisir le maximum comme le pire cas de τ_{ua} pour l'instant critique considéré. La date d'activation $a_{ua}(p)$ d'une instance de position p est donnée par :

$$a_{ua}(p) = (p-1)T_u + \Phi_{uac_u} \tag{2.10}$$

Afin de déterminer l'instant de fin d'exécution de l'instance p, et ainsi son temps de réponse, nous calculons la longueur de la période d'activité associée à l'instance p (en prenant en compte pour les instances de τ_{ua} seulement celles qui sont activées avant l'instance étudiée p). La longueur de cette période d'activité est calculée par la recherche du plus petit point fixe de la suite récurrente suivante :

$$L_{ua}^{(k)}(p) = B_{ua} + (p - p_{0,ua} + 1)\, C_{ua} + \sum_{\forall i} W_{ic_i}\left(\tau_{ua}, L_{ua}^{(k-1)}(p)\right) \tag{2.11}$$

Enfin, le temps de réponse $R_{ua}(p)$ de l'instance p est donné par la différence entre la date de fin d'exécution et celle de son activation :

$$R_{ua}(p) = L_{ua}(p) - a_{ua}(p) \tag{2.12}$$

Nous appliquons ces formules pour trouver le temps de réponse des instances de τ_{ua} présentes dans la période d'activité de la figure 2.5

– Instance p=0 :

$$\begin{aligned}
L_{ua}^{(0)}(0) &= 2 \\
L_{ua}^{(1)}(0) &= (p - p_{0,ua} + 1)\, C_{ua} + \sum_{\forall i} W_{ic_i}\left(\tau_{ua}, L_{ua}^{(0)}(p)\right) \\
&= (0 - 0 + 1)\, 2 + W_{11}\left(\tau_{ua}, 2\right) + W_{ub}\left(\tau_{ua}, 2\right) \\
&= 2 + 5 + 3 = 10 \\
L_{ua}^{(2)}(0) &= 2 + W_{11}\left(\tau_{ua}, 10\right) + W_{ub}\left(\tau_{ua}, 10\right) = 2 + 11 + 3 = 16 \\
L_{ua}^{(3)}(0) &= 2 + W_{11}\left(\tau_{ua}, 16\right) + W_{ub}\left(\tau_{ua}, 16\right) = 2 + 11 + 6 = 19 \\
L_{ua}^{(4)}(0) &= 2 + W_{11}\left(\tau_{ua}, 19\right) + W_{ub}\left(\tau_{ua}, 19\right) = 2 + 11 + 6 = 19
\end{aligned}$$

Le temps de réponse de l'instance de position $p = 0$ est :

$$R_{ua}(0) = L_{ua}(0) - (p-1)T_u - \Phi_{uac_u} = 19 - (-12) - 4 = 27$$

– Instance p=1 :

$$\begin{aligned}
L_{ua}^{(0)}(1) &= 2 \\
L_{ua}^{(1)}(1) &= (p - p_{0,ua} + 1)\, C_{ua} + \sum_{\forall i} W_{ic_i}\left(\tau_{ua}, L_{ua}^{(0)}(p)\right) \\
&= (1 - 0 + 1)\, 2 + W_{11}(\tau_{ua}, 2) + W_{ub}(\tau_{ua}, 2) \\
&= 4 + 5 + 3 = 12 \\
L_{ua}^{(2)}(1) &= 4 + W_{11}(\tau_{ua}, 12) + W_{ub}(\tau_{ua}, 12) = 4 + 11 + 3 = 18 \\
L_{ua}^{(3)}(1) &= 4 + W_{11}(\tau_{ua}, 18) + W_{ub}(\tau_{ua}, 18) = 4 + 11 + 6 = 21 \\
L_{ua}^{(4)}(1) &= 4 + W_{11}(\tau_{ua}, 21) + W_{ub}(\tau_{ua}, 21) = 2 + 11 + 6 = 21
\end{aligned}$$

Le temps de réponse de l'instance de position $p = 1$ est :

$$R_{ua}(0) = 21 - (1 - 1)12 - 4 = 17.$$

– Instance p=2 :

$$\begin{aligned}
L_{ua}^{(0)}(1) &= 2 \\
L_{ua}^{(1)}(1) &= (p - p_{0,ua} + 1)\, C_{ua} + \sum_{\forall i} W_{ic_i}\left(\tau_{ua}, L_{ua}^{(0)}(p)\right) \\
&= (2 - 0 + 1)\, 2 + W_{11}(\tau_{ua}, 2) + W_{ub}(\tau_{ua}, 2) \\
&= 6 + 5 + 3 = 14 \\
L_{ua}^{(2)}(1) &= 6 + W_{11}(\tau_{ua}, 14) + W_{ub}(\tau_{ua}, 14) = 6 + 11 + 3 = 20 \\
L_{ua}^{(3)}(1) &= 6 + W_{11}(\tau_{ua}, 20) + W_{ub}(\tau_{ua}, 20) = 6 + 11 + 6 = 23 \\
L_{ua}^{(4)}(1) &= 6 + W_{11}(\tau_{ua}, 23) + W_{ub}(\tau_{ua}, 23) = 6 + 11 + 6 = 23
\end{aligned}$$

Le temps de réponse de l'instance de position $p = 2$ est :

$$R_{ua}(0) = 23 - (2 - 1)12 - 4 = 7$$

D'où, lorsque les tâches τ_{11} et τ_{ub} initient l'instant critique le pire temps de réponse est :

$$R_{ua} = \max_{p=p0,ua \ldots pL,ua} R_{ua}(p) = \max\{27, 17, 7\} = 27$$

2.3.5 Analyse exacte de pire temps de réponse

Nous avons vu comment le pire temps de réponse d'une tâche peut être calculé pour un instant critique donné. Le problème est que les tâches candidates initiant l'instant critique produisant

Combinaison de tâches candidates	Instance p	$R_{ua}(p)$	$MaxR_{ua}(p)$	R_{ua}
τ_{11}, τ_{ub}	p=0	27	27	27
	p=1	17		
	p=2	7		
τ_{11}, τ_{ua}	p=0	7	15	
	p=1	15		
	p=2	5		
τ_{12}, τ_{ub}	p=0	17	17	
	p=1	12		
	p=2	2		
τ_{12}, τ_{ua}	p=0	8	8	
	p=1	7		

TAB. 2.1 – Calcul exacte du pire temps de réponse

le pire temps de réponse ne sont pas connues. Autrement dit, quel instant critique candidat provoque le pire temps de réponse de la tâche analysée ? car plusieurs pire scénarii ou instants critiques sont possibles.

Un analyse exacte [107, 70] consiste à étudier toutes les combinaisons de toutes les tâches plus prioritaires que τ_{ua} dans chaque transaction, ensuite de choisir le pire temps de réponse.
Pour un système de n transactions tel que chaque transaction Γ_i contient n_i tâches plus prioritaires que la tâche analysée τ_{ua}. Le nombre de combinaisons (instants critiques) à considérer est égal à :

$$\prod_{i=1}^{n} |\Gamma_i| = n_1 \times n_2 \times ... \times n_n \tag{2.13}$$

Le pire temps de réponse est alors donné par :

$$R_{ua} = \max_{\forall i, \forall c_i \in hp_i(\tau_{ua})} \{R_{ua}\}$$

Dans l'exemple de la figure 2.5 nous devons examiner $|\Gamma_1| \times |\Gamma_u| = 2 \times 2 = 4$ instants critiques. La table 2.3.5 donne tous les scénarii de calcul de la valeur exacte du pire temps de réponse de la tâche τ_{ua}

2.3.6 Bilan

Cette méthode d'analyse fournit une valeur exacte du pire temps de réponse d'une tâche sous analyse. Malheureusement elle a une complexité exponentielle qui la rend inapplicable pour des systèmes de taille réaliste. Afin d'éviter ce problème de complexité plusieurs travaux ont été réalisés. Tindell [107] a proposé une méthode approchée fournissant une borne supérieure du pire temps de réponse avec une complexité pseudo-polynomiale. Cette approche a été formalisée et

étendue pour des tâches avec offsets dynamiques (§ 2.3.9) par Palencia et Harbour [70, 60]. La fonction d'interférence approchée utilisée dans cette méthode a été améliorée (fonction d'interférence effective) par Turja et Nolin [61, 62, 63] cette nouvelle fonction a permis de réduire le pessimisme de la borne approchée du pire temps de réponse. Nous détaillons ces travaux dans les sections suivantes.

Notations : Pour chaque fonction f, nous notons f_M la valeur de f calculée par la méthode d'analyse M. Par exemple pour le pire temps de réponse nous notons :

- $\text{WCRT}_{ua,Exact}$ (ou $R_{ua,Exact}$) le pire temps de réponse exact de la tâche τ_{ua}.
- $\text{WCRT}_{ua,Palencia}$ (ou $R_{ua,Palencia}$) la borne du pire temps de réponse obtenue par la méthode approchée de Tindell-Palencia (section 2.3.7).
- $\text{WCRT}_{ua,Nolin}$ (ou $R_{ua,Nolin}$) la borne du pire temps de réponse obtenue par la méthode approchée de Turja et Nolin (section 2.3.8).

2.3.7 Méthode approchée de base (Tindell,Palencia)

Cette méthode d'analyse a été proposée par Tindell [107, 70] dans le but d'éviter le problème de la complexité exponentielle de l'analyse exacte. Elle permet d'obtenir une borne supérieure sûre (jamais inférieure à la valeur exacte) pour le pire temps de réponse, avec une complexité pseudo-polynomiale. Elle repose sur le même principe de calcul que la méthode exacte, mais au lieu de traiter tous les instants critiques possibles, on effectue un seul processus de calcul de temps de réponse en utilisant une fonction d'approximation d'interférence pour chaque transaction du système. Puisque la tâche qui initie l'instant critique n'est pas connue pour une transaction Γ_i, la fonction d'approximation prend tout simplement le maximum des interférences (borne supérieure) en considérant chacune des tâches candidates de Γ_i comme celle qui initie l'instant critique sans prendre en considération l'interaction avec les autres transactions. La borne supérieure de l'interférence d'une transaction Γ_i durant une période d'activité de longueur t est donc donnée par la formule suivante :

$$W_{i,Palencia}(\tau_u, t) = \max_{\forall c \in hp_i(\tau_{ua})} W_{ic,Palencia}(\tau_u, t) \tag{2.14}$$

Avec $W_{ic,Palencia}(t)$ (équation 2.6) l'interférence de l'ensemble des tâches de $hp_i(\tau_{ua})$ lorsque l'instant critique est initié par la tâche τ_{ic}.

L'interférence d'une tâche τ_{ij} est calculée comme dans l'approche exacte, c'est-à-dire pour une instance de τ_{ij} activée dans la période d'étude, toute la durée d'exécution (WCET) de τ_{ij} est prise comme contribution de cette instance dans la période d'étude. La fonction d'interférence d'une transaction peut être représentée graphiquement sous forme de courbes en escalier, chaque point d'escalier correspond à une activation d'une instance d'une tâche. Cette représentation permet d'illustrer l'évolution de l'interférence en fonction de la longueur de la période d'activité. La figure 2.7 montre les courbes d"interférences des transactions du système de la figure 2.6.

On voit bien que la valeur de l'interférence maximum peut correspondre à des tâches candidates (initiant l'instant critique) différentes pour des longueurs différentes de la période d'activité. Par

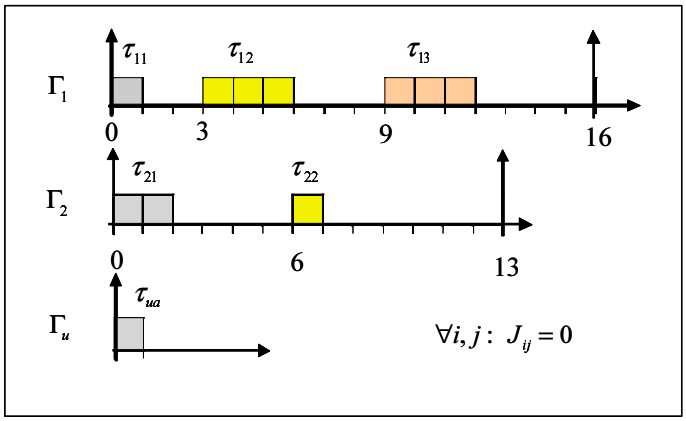

FIG. 2.6 – Exemple d'un système de transactions.

exemple, dans la figure 2.7(a), l'interférence maximum de Γ_1, dans une période d'activité de longueur 4 unités de temps correspond à son interférence lorsque la tâche τ_{11} initie l'instant critique, tandis qu 'elle correspond à son interférence lorsque τ_{12} initie l'instant critique lorsque la période considérée est de longueur égale à 7.

Pour cette analyse approchée, un seul processus du calcul itératif de temps de réponse est à effectuer. Dans chaque itération l'interférence maximum de chacune des transactions est prise en compte. Le fait de déterminer l'interférence maximum de chaque transaction indépendamment (sans interaction avec) des autres transactions permet de passer d'une complexité exponentielle à une complexité pseudo-polynomiale. De la même façon, nous calculons les temps de réponse de chacune des instances de τ_{ua} et le maximum est pris comme une borne supérieure du pire temps de réponse ; nous appliquons cela pour chaque tâche candidate τ_{uc} de Γ_u. La période d'activité associée à une instance p (date de fin d'exécution de l'instance p) est déterminée par :

$$\begin{aligned} L^{(k)}_{ua,Palencia}(p) &= B_{ua} + (p - p_{0,ua} + 1)\, C_{ua} + W_{uc,Palencia}\left(\tau_{ua}, L^{(k-1)}_{ua,Palencia}(p)\right) + \\ &\quad \sum_{\forall i \neq u} W_{i,Palencia}\left(\tau_{ua}, L^{(k-1)}_{ua,Palecia}(p)\right) \end{aligned} \tag{2.15}$$

De même, afin de trouver la position (numéro) de la dernière instance de τ_{ua} ; la plus longue période d'activité est donnée par la formule suivante :

$$\begin{aligned} L^{(k)}_{ua,Palencia} &= B_{ua} + \left(\left\lceil \frac{L^{(k-1)}_{ua,Palencia} - \Phi_{uac_u}}{T_u} \right\rceil - p_{0,ua} + 1 \right) C_{ua} + \\ &\quad W_{uc,Palencia}(\tau_{ua}, L^{(k-1)}_{ua,Palencia}(p)) + \sum_{\forall i} W_{ic_i,Palencia}(\tau_{ua}, L^{(k-1)}_{ua,Palencia}) \end{aligned} \tag{2.16}$$

Dans le but de simplifier et clarifier les calculs, dans toute la suite de document nous considérons que la transaction Γ_u ne contient qu'une seule tâche qui est la tâche analysée τ_{ua}, et qu'une seule

instance de τ_{ua} est exécutée dans la période d'activité. Cette supposition est supprimée par le calcul du temps de réponse de toutes les instances de τ_{ua} par une simple adaptation des formules à la [51] (Pour les formules complètes d'analyse, voir annexe A).
Avec cette supposition le temps de réponse de la seule instance de τ_{ua} est réduit à :

$$L^{(k)}_{ua,Palencia} = B_{ua} + C_{ua} + \sum_{\forall i \neq u} W_{i,Palencia}\left(\tau_{ua}, L^{(k-1)}_{ua,Palencia}\right) \tag{2.17}$$

Notons que lorsque les gigues sont nulles, la longueur de la période d'activité associée à l'instance de τ_{ua} représente son temps de réponse (car sa date d'activation est égale à la date de début de la période d'activité)

$$R^{(k)}_{ua,Palencia} = B_{ua} + C_{ua} + \sum_{\forall i \neq u} W_{i,Palencia}\left(\tau_{ua}, R^{(k-1)}_{ua,Palencia}\right) \tag{2.18}$$

2.3.7.1 Exemple :

Nous appliquons la méthode approchée pour trouver une borne du pire temps de réponse de la tâche τ_{ua} présentée sur la figure 2.6. Supposons que toutes les tâches soient plus prioritaires que τ_{ua} et aient des gigues nulles. La figure 2.7 (*a et c*) illustre l'évolution de la fonction d'interférence durant la première période de chacune des transactions du système en considérant une tâche de chaque transaction, celle qui coïncide avec l'instant critique. La fonction est alors représentée par une courbe en escalier.

Les courbes de la figure 2.7(b) et (d) représentent la fonction d'interférence approchée (maximum) pour les transactions Γ_1 et Γ_2 respectivement. Cette interférence sera utilisée dans le calcul de la borne supérieure du pire temps de réponse. Notons que toutes les formules utilisées dans ce calcul correspondent à la méthode de Palencia. Les étapes du calcul itératif de temps de réponse de la tâche τ_{ua} sont :

$$\begin{aligned}
\mathrm{R}^{(0)}_{ua} &= C_{ua} = 1 \\
\mathrm{R}^{(1)}_{ua} &= C_{ua} + \sum_{i=1..2} W_i\left(\tau_{ua}, R^{(0)}_{ua}\right) = C_{ua} + W_1\left(\tau_{ua}, R^{(0)}_{ua}\right) + W_2\left(\tau_{ua}, R^{(0)}_{ua}\right) \\
&= 1 + \max\left\{W_{11}(R^{(0)}_{ua}), W_{12}(R^{(0)}_{ua}), W_{12}(R^{(0)}_{ua})\right\} + \max\left\{W_{21}(R^{(0)}_{ua}), W_{22}(R^{(0)}_{ua})\right\} \\
&= 1 + \max\left\{\sum_{\forall j \in hp_1(\tau_{ua})}\left(\left\lfloor\frac{J_{1j} + \Phi_{1j1}}{T_1}\right\rfloor + \left\lceil\frac{R^{(0)}_{ua} - \Phi_{1j1}}{T_1}\right\rceil\right) C_{1j}, W_{12}(R^{(0)}_{ua}), W_{12}(R^{(0)}_{ua})\right\} + \\
&\quad \max\left\{W_{21}(R^{(0)}_{ua}), W_{22}(R^{(0)}_{ua})\right\} \\
&= 1 + 3 + 2 = 6
\end{aligned}$$

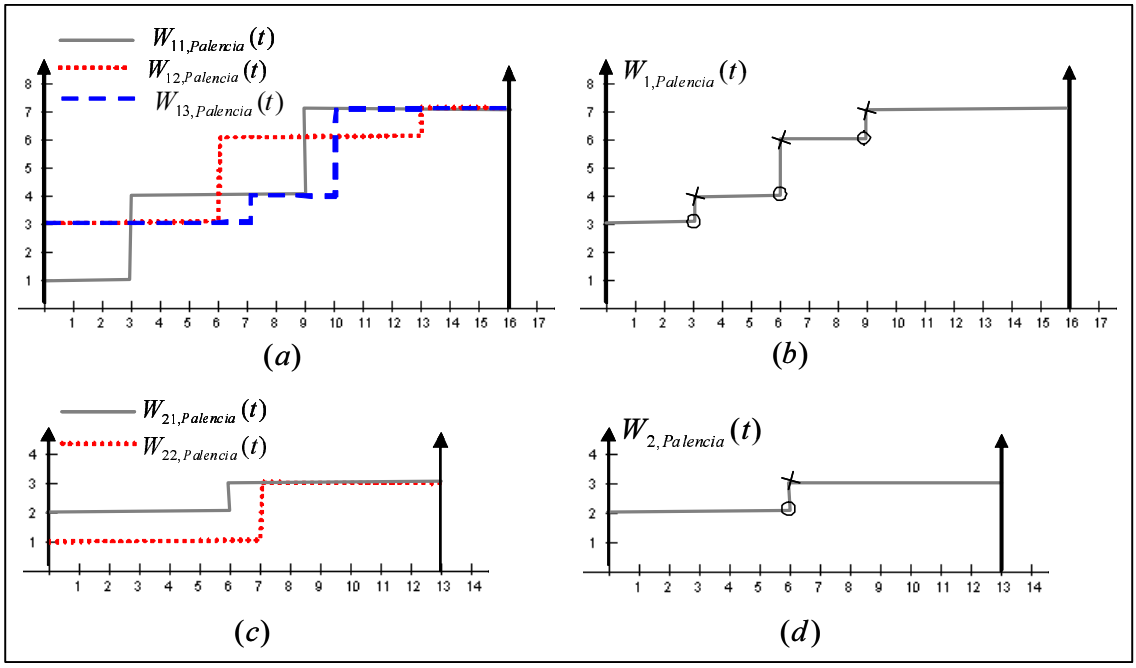

FIG. 2.7 – Interférences des transactions.

$$
\begin{aligned}
\mathrm{R}_{ua}^{(2)} &= C_{ua} + W_1(6) + W_2(6) = 1 + 4 + 2 = 7 \\
\mathrm{R}_{ua}^{(3)} &= C_{ua} + W_1(7) + W_2(7) = 1 + 6 + 3 = 10 \\
\mathrm{R}_{ua}^{(4)} &= C_{ua} + W_1(10) + W_2(10) = 1 + 7 + 3 = 11 \\
\mathrm{R}_{ua}^{(5)} &= C_{ua} + W_1(11) + W_2(11) = 1 + 7 + 3 = 11
\end{aligned}
$$

La borne de temps de réponse de τ_{ua} est égale à 11 unités de temps (plus petit point fixe du calcul itératif). Tandis que la valeur exacte de pire temps de réponse est égale à 8 unités de temps, ce qui correspond à l'instant critique initié par τ_{11} et τ_{21}. Cette méthode nécessite un temps d'exécution avec de complexité pseudo-polynomiale, en revanche elle fournit une borne très pessimiste (approchée) par rapport à la valeur exacte du pire temps de réponse.

2.3.7.2 Implémentation efficace de la méthode approchée

Turja et Nolin [60] ont proposé une implémentation efficace permettant d'accélérer le temps d'exécution de l'algorithme du test approché d'ordonnançabilité. Cette technique d'implémentation exploite le motif périodique de la fonction d'interférence d'une transaction dans une période d'activité, elle sauvegarde de manière statique les informations de l'interférence de chaque transaction du système. Notons que cette technique ne réduit pas le pessimisme de la méthode d'approximation de base, mais elle permet tout simplement d'améliorer le temps de traitement.

Nous savons que l'interférence d'une transaction Γ_i est bien divisée en deux parties, pour une tâche candidate donnée τ_{ic} : la partie I_{ic}^{Set1} est statique et indépendante de longueur de la période

d'activité t, et la partie I_{ic}^{Set2} dépend de la longueur t de la période d'activité.

$$W_{ic,Palencia}(\tau_{ua},t) = \sum_{\forall j \in hp_i(\tau_{ua})} (I_{ijc}^{Set1} + I_{ijc}^{Set2}(t)) = \sum_{\forall j \in hp_i(\tau_{ua})} I_{ijc}^{Set1} + \sum_{\forall j \in hp_i(\tau_{ua})} I_{ijc}^{Set2}(t) \quad (2.19)$$

Donc, durant le calcul de temps de réponse c'est la partie d'interférence $I_i^{set2}(t)$ qui doit être réévaluée à chaque itération. De même, l'interférence maximum $W_i(t)$ peut alors être divisée en deux parties :

$$W_{i,Palencia}(\tau_{ua},t) = J_i^{ind} + T_i^{ind}(\tau_{ua},t) \quad (2.20)$$

– La partie statique induite par la gigue notée J_i^{ind}, cette partie est forcément le maximum des parties I_{ic}^{set1}. Elle correspond aux instances activées avant la période d'activité (ensemble set1).

$$J_i^{ind} = \max_{\forall c \in hp_i(\tau_{ua})} \sum_{\forall j \in hp_i(\tau_{ua})} I_{ijc}^{set1} \quad (2.21)$$

– La partie dynamique induite par le temps notée $T_i^{ind}(t)$, cette partie correspond aux instances de l'ensemble Set2, elle possède un motif périodique de période T_i

$$T_i^{ind}(t) = \max_{\forall c \in hp_i(\tau_{ua})} W_{ic}^{+}(\tau_{ua},t) \quad (2.22)$$

$$W_{ic}^{+}(\tau_{ua},t) = \sum_{\forall j \in hp_i(\tau_{ua})} \left(I_{ijc}^{set1} + I_{ijc}^{set2}(t)\right) - J_i^{ind} \quad (2.23)$$

L'évaluation de la formule de $T_i^{ind}(t)$ nécessite un temps d'exécution important, et ceci à chaque itération de calcul approchée du pire temps de réponse. Vue la périodicité (avec une période T_i) de $I_{ic}^{Set2}(t)$ il suffit de sauvegarder les points (dates, interférences) convexes de la fonction $I_{ic}^{Set2}(t)$ (figure 2.7(a,c)) pour chaque tâche candidate, et ensuite stocker ces points qui représentent l'interférence maximum durant une période dans une table (figure 2.7(b,d)). Cette table sera utilisée pour déduire l'interférence maximum de la transaction pour n'importe quelle période d'activité de longueur t, par une simple recherche dans la table.

2.3.7.3 Note

Nous appelons un point concave, un point (date, interférence) représentant l'interférence imposée à la date d'activation d'une tâche (début d'une demande processeur), et un point convexe le point représentant l'interférence à la date de sa fin d'exécution. Graphiquement, les points concaves sont les points minimaux des segments verticaux (ou obliques figure 2.8) de la fonction d'interférence, tandis que les points convexes sont les points maximaux de ces segments. Dans les figures 2.7-2.9 les points concaves sont marqués par cercles et les points convexes sont marqués par des croix.

2.3.8 Méthode approchée basée sur l'interférence effective (Turja-Nolin)

Pour réduire le pessimisme de l'analyse approchée, Turja et Nolin [61] ont proposé une nouvelle fonction d'interférence qui permet d'évaluer l'interférence causée effectivement par une tâche ou une transaction dans une période d'activité. L'utilisation de cette nouvelle fonction réduit le pessimisme causé par la surestimation de l'interférence approchée (maximum) d'une transaction. En effet, la fonction W_{ijc} présentée au dessus surestime l'interférence en prenant toute la durée entière d'exécution d'une tâche même si celle-ci ne se termine pas dans la période d'activité considérée. Par exemple sur la figure 2.7(a) l'interférence de Γ_1 lorsque τ_{11} initie l'instant critique est égale à 4 dans une période d'activité de longueur égale à 4. Si on analyse ce scénario d'exécution de tâches, à l'instant $t = 0$ la tâche τ_{11} s'active provoquant une charge de 1, puis la tâche τ_{12} s'active à l'instant $t = 3$ provoquant une charge de 3, mais une seule unité de temps de cette charge de τ_{12} ne pourra être exécutée dans la période d'activité étudiée (de longueur 4), donc l'interférence totale de Γ_1 dans cette période d'activité ne vaut que $1 + 1 = 2$ au lieu de $1 + 3 = 4$.

Il est important de signaler que la fonction classique d'interférence ne produit pas de pessimisme sur le temps de réponse d'une tâche pour un instant critique donné (analyse exacte) car on ne peut pas avoir un point fixe avant la fin d'exécution d'une tâche qui a déjà demandé le processeur. Par contre elle produit une surestimation de la demande processeur dans la fonction d'interférence approchée.

La méthode repose sur l'idée que l'interférence ne doit pas excéder la capacité du processeur. Autrement dit, l'interférence ne peut pas évoluer plus rapidement que la période d'activité considérée. Par conséquent la dérivée de la fonction d'interférence ne peut pas être supérieure à la dérivée de la fonction d'évolution de temps comme on va le montrer plus tard dans cette section.

$$\frac{\mathrm{d}W_{ijc(t)}}{\mathrm{d}t} \leq \frac{\mathrm{d}t}{\mathrm{d}t} \Rightarrow \frac{\mathrm{d}W_{ijc(t)}}{\mathrm{d}t} \leq 1 \tag{2.24}$$

D'où la fonction d'interférence effective peut être représentée graphiquement par des rampes (figure 2.8). Dans l'exemple présenté sur la figure 2.8, l'instant critique est initié par la tâche τ_{11}. L'interférence effective de la tâche τ_{13} de la transaction Γ_1 dans une période d'activité de longueur $t = 10$ est égale à 1 $((3 - 2) = 1)$ car cette instance ne pourra pas être exécutée complètement dans la période d'activité ($t = 10$), deux unités de temps ($x_{131} = 2$) sont hors de la période d'activité. Cette quantité sera retranchée de la WCET de τ_{13} contribuant à la période d'activité, tandis que la méthode classique donne une interférence de τ_{13} égale à 3. D'où l'interférence totale de la transaction Γ_1 pour $t = 10$ est égale à 5 $(1 + 3 + (3 - 2) = 5)$. La méthode classique donne une interférence égale à 7 $(1 + 3 + 3 = 7)$.

Théorème 19 *[59] [61] Soit une tâche τ_j de période T_j et de WCET C_j $(0 < C_j \leq T_j)$ activée à l'instant $t = 0$. Soit un intervalle de temps t positif, $t = k \times T_j + t'$ (où k est un entier et $0 \leq t' < T_j$. Alors $k \times C_j + \min(t', C_j)$ est une borne supérieure de l'interférence que la tâche τ_j peut imposer à n'importe quelle tâche de priorité inférieure durant un intervalle de temps t.*

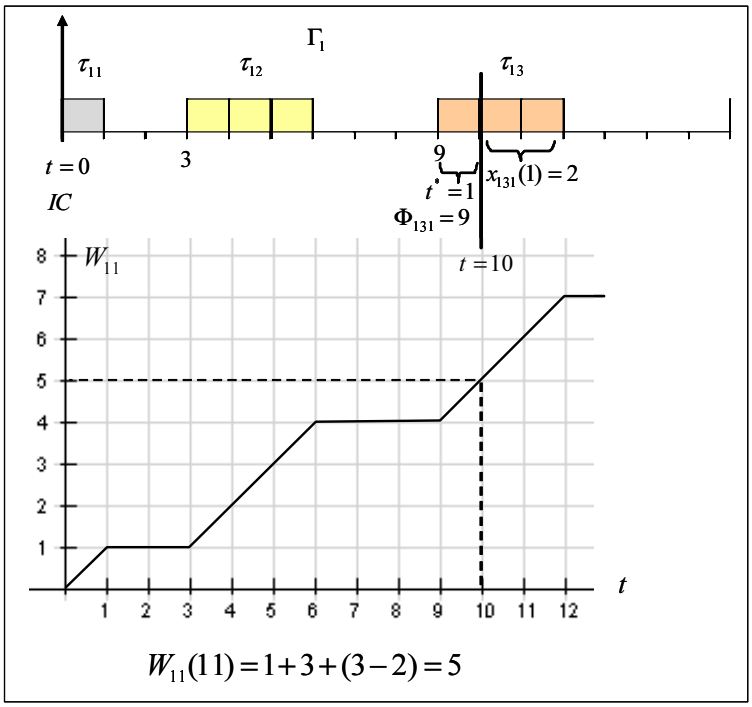

FIG. 2.8 – Interférence effective.

Le théorème 19 donne l'interférence exacte (effective) qu'une tâche peut imposer sur le temps de réponse d'une tâche moins prioritaire, dans une période d'activité de longueur t. Donc, pour une instance d'une tâche τ_{ij}, activée dans la période d'étude, les unités de temps qui pourront être exécutées dans la période d'étude qui seront prises en compte, tandis que la formule classique d'interférence (équation 2.4) prend toutes les unités de WCET de τ_{ij} en compte. Afin de trouver la formule de la fonction d'interférence effective, il suffit de supprimer la surestimation x_{ijc} causée par les unités de temps des instances de la tâche τ_{ij} ne pouvant pas être exécutées avant la date t. D'où la partie d'interférence dépendante de la longueur de l'intervalle d'étude $I_{ijc}^{Set2}(t)$ est obtenue par la nouvelle formule suivante :

$$
\begin{aligned}
I_{ijc}^{Set2}(t) &= \left\lceil \frac{t^*}{T_i} \right\rceil C_{ij} - x_{ijc}(t^*) \qquad (2.25)\\
t^* &= t - \Phi_{ijc}\\
x_{ijc}(t^*) &= \begin{cases} C_{ij} - (t^* \mod T_i) & \text{si } t^* > 0 \wedge (0 < t^* \bmod T_i) < C_{ij}) \\ 0 \text{ sinon} \end{cases}
\end{aligned}
$$

D'où :

$$
W_{ijc,Nolin}(t) = \left\lfloor \frac{J_{ij}+\Phi_{ijc}}{T_i} \right\rfloor C_{ij} + \left\lceil \frac{t-\Phi_{ijc}}{T_i} \right\rceil C_{ij} - x_{ijc}(t-\Phi_{ijc}) \qquad (2.26)
$$

Théorème 20 *[61] Pour une tâche à analyser τ_{ua} et une tâche candidate $\tau_{ic} \in \Gamma_i$, la valeur obtenue avec la nouvelle formule de I_{ijc}^{set2} (équation 2.25) n'est jamais plus élevée que celle obtenue avec l'ancienne formule 2.3.*

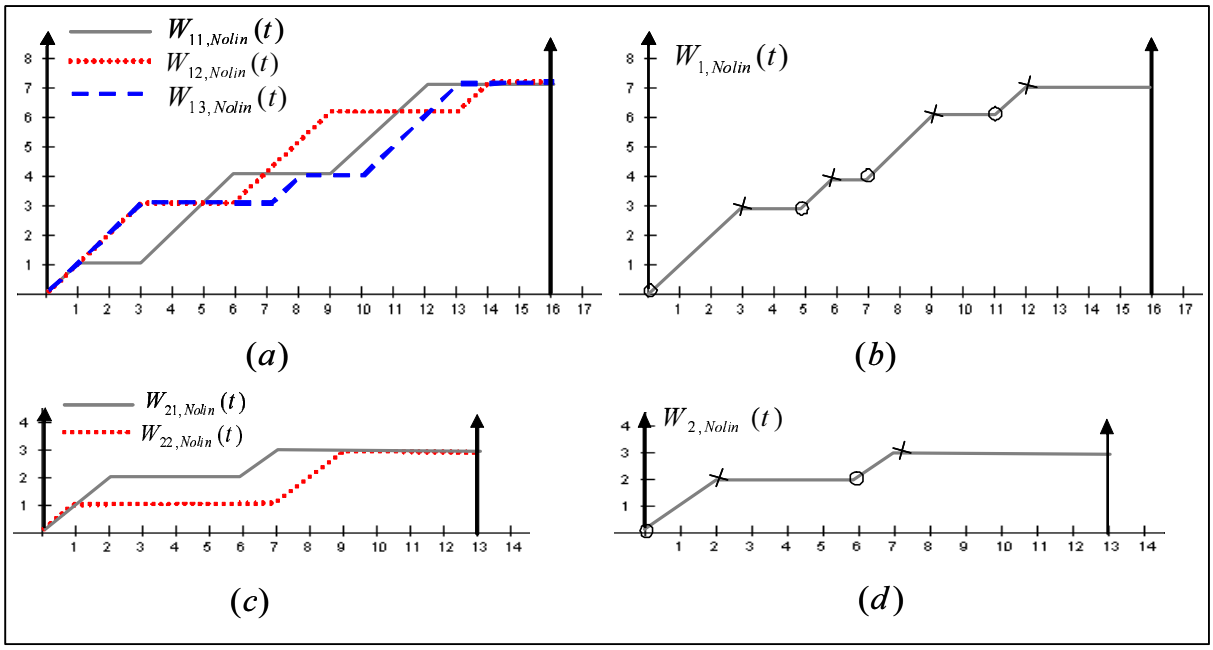

FIG. 2.9 – Courbes d'interférences effectives.

La borne du pire temps de réponse est calculée de la même manière par la recherche de point fixe en remplaçant I_{ijc}^{set2} de l'équation 2.3 par la nouvelle formule donnée par l'équation 2.25.

$$R_{ua,Nolin}^{(k)} = B_{ua} + C_{ua} + \sum_{\forall i \neq u} W_{i,Nolin}\left(\tau_{ua}, R_{ua,Nolin}^{(k-1)}\right) \tag{2.27}$$

$$W_{i,Nolin}(\tau_{ua}, t) = \max_{\forall c \in hp_i(\tau_{ua})} \sum_{\forall j \in hp_i(\tau_{ua})} \left(\left\lfloor \frac{J_{ij} + \Phi_{ijc}}{T_i} \right\rfloor C_{ij} + \left\lceil \frac{t - \Phi_{ijc}}{T_i} \right\rceil C_{ij} - x_{ijc}(t - \Phi_{ijc}) \right) \tag{2.28}$$

Par conséquent, cette nouvelle fonction d'interférence effective, élimine alors une surestimation de l'interférence maximum. Nous comparons les deux représentations graphiques des deux fonctions d'interférence approchée de la figure 2.7(b) et celle de la figure 2.9(b). Pour une période d'activité de longueur $t = 7$ par exemple, l'interférence maximum effective est égale à 4. Par contre celle calculée par la formule classique est égale à 6. En plus, en regardant la courbe de $W_1(t)$ de la figure 2.9(b), un point fixe de temps de réponse pourra exister à cette instant (car à cette instant on est sur un segment horizontal de la courbe). Tandis que l'interférence maximum est déjà surestimée et pourra amener à dépasser le premier point fixe.

$$R_{ua,Exact} \leq R_{ua,Nolin} \leq R_{ua,Palencia}$$

Notons que la borne du pire temps de réponse fournie par la méthode approchée en utilisant la fonction d'interférence effective est toujours comprise entre la valeur exacte et la borne fournie

par la méthode approchée classique, du pire temps de réponse. Donc l'analyse approchée reste toujours une condition suffisante d'ordonnançabilité. Pour une analyse complète i.e. la transaction contenant la tâche analysée τ_{ua} peut contenir d'autres tâches plus prioritaires que τ_{ua}, voir annexe A.1.

2.3.8.1 Exemple

Nous appliquons la méthode d'analyse approchée en utilisant l'interférence effective sur le même exemple de la figure 2.6. Les courbes illustrant l'évolution d'interférence effective de chacune des deux transactions Γ_1 et Γ_2 pour chaque tâche candidate sont représentées sur la figure 2.9. Notons que les formules utilisées dans ce calcul correspondent toutes à la méthode de Turja et Nolin

$$
\begin{aligned}
\mathrm{R}_{ua}^{(0)} &= C_{ua} = 1 \\
\mathrm{R}_{ua}^{(1)} &= C_{ua} + \sum_{i=1..2} W_i\left(\tau_{ua}, R_{ua}^{(0)}\right) = C_{ua} + W_1\left(\tau_{ua}, R_{ua}^{(0)}\right) + W_2\left(\tau_{ua}, R_{ua}^{(0)}\right) \\
&= 1 + \max\left\{W_{11}(R_{ua}^{(0)}), W_{12}(R_{ua}^{(0)}), W_{12}(R_{ua}^{(0)})\right\} + \max\left\{W_{21}(R_{ua}^{(0)}), W_{22}(R_{ua}^{(0)})\right\} \\
&= 1 + \max\Big(\sum_{\forall j \in hp_1(\tau_{ua})} \left(\left\lfloor \frac{J_{1j} + \Phi_{1j1}}{T_1} \right\rfloor C_{1j} + \left\lceil \frac{R_{ua}^{(0)} - \Phi_{1j1}}{T_1} \right\rceil C_{1j} - x_{ijc}\right), W_{12}(R_{ua}^{(0)}), \\
&\quad W_{12}(R_{ua}^{(0)})\Big) + \max\left\{W_{21}(R_{ua}^{(0)}), W_{22}(R_{ua}^{(0)})\right\} \\
&= \max\{1-0, 3-2, 3-2\} + \max\{2-1, 1-0\} = 1+1+1 = 3 \\
\mathrm{R}_{ua}^{(2)} &= C_{ua} + W_1(3) + W_2(3) = 1+3+2 = 6 \\
\mathrm{R}_{ua}^{(3)} &= C_{ua} + W_1(6) + W_2(6) = 1+4+2 = 7 \\
\mathrm{R}_{ua}^{(4)} &= C_{ua} + W_1(7) + W_2(7) = 1+4+3 = 8 \\
\mathrm{R}_{ua}^{(5)} &= C_{ua} + W_1(8) + W_2(8) = 1+5+3 = 9 \\
\mathrm{R}_{ua}^{(6)} &= C_{ua} + W_1(9) + W_2(9) = 1+6+3 = 10 \\
\mathrm{R}_{ua}^{(7)} &= C_{ua} + W_1(10) + W_2(10) = 1+6+3 = 10
\end{aligned}
$$

La borne du pire temps de réponse de la tâche τ_{ua} est égale à $R_{ua,Nolin} = 10$, et est moins pessimiste que celle calculée par la méthode classique ($R_{ua,Palencia} = 11$). Une extension de l'algorithme d'implémentation présentée dans la section 2.3.7.2 a été adaptée [62, 63] pour accélérer l'exécution de cette nouvelle méthode approchée. Dans la section suivante, nous détaillons cet algorithme que nous allons utiliser dans les chapitres suivants.

2.3.8.2 Algorithme efficace d'implémentation

Dans chaque itération de processus de calcul de la borne du pire temps de réponse, l'interférence maximum de chaque transaction est à calculer et ceci en examinant tous les couples de tâches : (tâche candidate, tâche plus prioritaire). Afin d'accélérer le calcul, Turja et Nolin [62, 63] déterminent le motif de la périodicité de l'interférence effective, puis l'utilisent dans l'algorithme

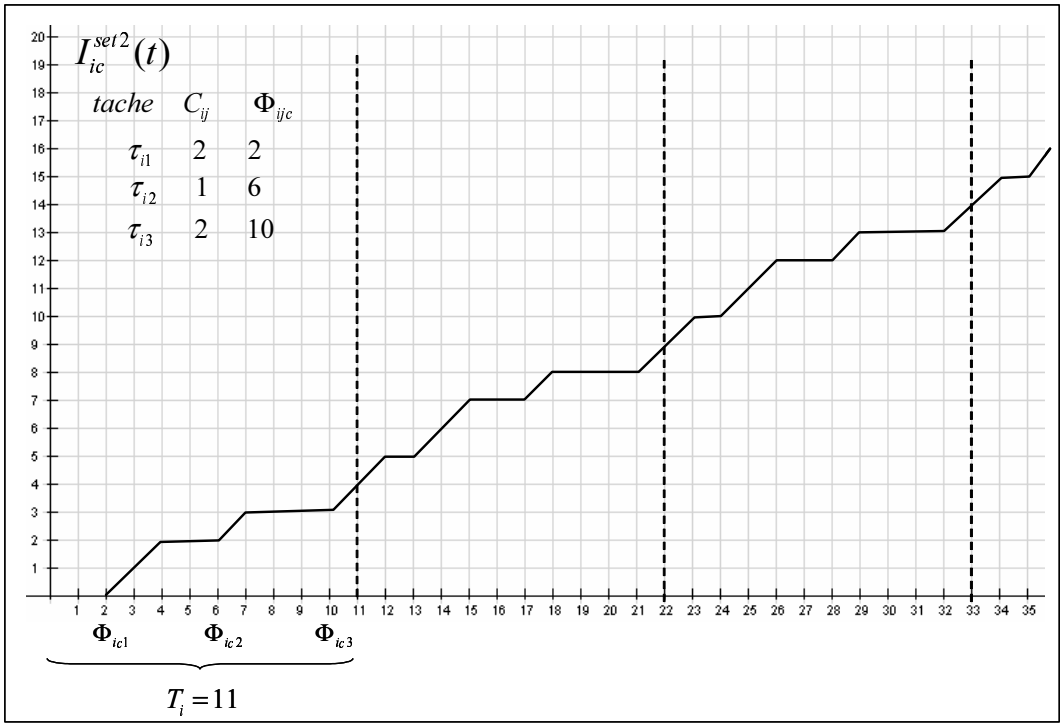

FIG. 2.10 – Périodicité de l'interférence effective.

d'implémentation présenté dans la section 2.3.7.2.

Le principe consiste alors à représenter statiquement l'interférence maximum de chaque transaction dans des tables. Dans le calcul de temps de réponse, ces tables sont utilisées par une simple recherche pour déduire l'interférence effective maximum causée par chacune des transactions. Cette représentation statique concerne la partie d'interférence dépendante de temps T_i^{ind}. Le motif de périodicité de l'interférence effective est différent de celui de l'interférence classique. La difficulté vient des instances activées avant la fin de la période lorsque leur exécution déborde sur la période suivante. Nous savons que l'arrivée des instances de l'ensemble Set2 est périodique lorsque les gigues sont nulles. La figure 2.10 illustre l'interférence I_{ic}^{set2} d'une transaction Γ_i lorsque une tâche τ_{ic} initie l'instant critique, l'instance de τ_{i3} est activée dans la première période et finit son exécution dans la période suivante. Nous constatons bien que l'interférence durant la première période est différente de celle produite dans la deuxième période qui est répétitive pour les périodes suivantes. L'interférence de l'instance débordante sera divisée en deux, celle causée dans la première période, et celle causée dans la deuxième période. Dans la suite nous détaillons l'algorithme de représentation statique de l'interférence maximum d'une transaction.

Soit une transaction Γ_i, sans perte de généralité supposons que toutes les tâches de Γ_i soient plus prioritaires que la tâche analysée τ_{ua}. Pour chaque tâche candidate τ_{ic} deux tables sont construites, P_{ic} pour sauvegarder l'interférence I_{ic}^{set2} (les points convexes de la courbe I_{ic}^{set2}, figure 2.10) causée durant la première période et une table P'_{ic} pour sauvegarder celle causée

dans la deuxième période. Pour pouvoir aborder ces calculs facilement, une étape préliminaire est indispensable.

2.3.8.2.1 Étape préliminaire. Trois opérations sont nécessaires pour restructurer les informations dans la transaction pour une tâche candidate τ_{ic} considérée. Cette restructuration n'a aucune influence sur la charge ou le comportement temporel de la transaction. On ne considère que les tâches de la transaction dont la priorité n'est pas inférieure à celle de τ_{ua}.

- **Opération d'ordre :** Consiste à classer (renuméroter) les tâches suivant leur date de première activation après l'instant critique. i.e. suivant un ordre croissant de la phase Φ_{ijc}.
- **Opération de fusion :** Cette opération consiste à fusionner deux à deux les tâches voisines pour lesquelles aucun temps creux n'est produit entre les exécutions. Pour $1 \leq j < |\Gamma_i|$, si $\Phi_{ijc} + C_{ij} \geq \Phi_{i(j+1)c}$ alors $\tau_{i(j+1)}$ est englobée dans τ_{ij}. Ces deux tâches forment alors une seule tâche avec WCET égale à $C_{ij} + C_{i(j+1)}$ et une date de première activation aprés l'instant critique (phase) égale à Φ_{ijc}. Les tâches de la transaction Γ_i sont renumérotées suivant l'ordre croissant de la phase Φ_{ijc} car $\tau_{i(j+1)}$ est supprimée. Cette opération est a répéter jusqu'à ce qu'aucun paire de tâches ne respecte la condition de fusion.

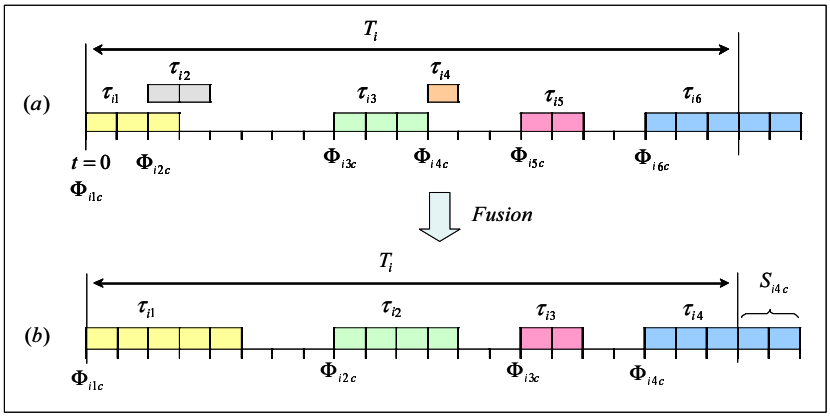

FIG. 2.11 – Opération de fusion.

Nous appliquons cette opération sur la transaction Γ_i, présentée dans la figure 2.11(a), lorsque une tâche τ_{ic} initie l'instant critique. Les tâches sont ordonnées (numérotées) suivant l'ordre croissant de leurs phases Φ_{ijc}. D'après les conditions de la fusion, la tâche τ_{i2} est intégrée dans τ_{i1} et la tâche τ_{i4} est fusionnée avec τ_{i3}. La figure 2.11 (b) présente la transaction obtenue après avoir renuméroté les nouvelles tâches.

- **Opération de séparation :** Pour une tâche τ_{ij}, telle que sa première activation après l'instant critique survienne dans la première période et que son exécution continue dans la deuxième période (une partie de son WCET est exécutée dans la première période et l'autre partie dans la deuxième période) une tâche S_{ijc} est séparée de τ_{ij} avec un WCET égal à la quantité de WCET de τ_{ij} qui a débordé dans la deuxième période. Cette quantité est calculée par :

$$S_{ijc} = \begin{cases} 0 \text{ si } \Phi_{ijc} + C_{ij} \leq T_i \\ \Phi_{ijc} + C_{ij} - T_i \text{ sinon} \end{cases}$$

D'une manière explicite, chaque tâche τ_{ij} débordant dans la deuxième période (quantité de WCET séparée positif, $S_{ijc} > 0$) est divisée en deux tâches $\tau_{ij'}$ et $\tau_{ij''}$. $\tau_{ij'}$ représente la quantité d'interférence de τ_{ij} produite dans la première période. La tâche $\tau_{ij''}$ est appelée tâche séparée et elle représente la quantité d'interférence de τ_{ij} produite au début de la deuxième période. Les valeurs de ces nouveaux paramètres sont données par la formule suivante :

$$C_{ij'} = C_{ij} - S_{ijc} \text{ et } C_{ij"} = S_{ijc}$$

$$\Phi_{ij'c} = \Phi_{ijc} \text{ et } \Phi_{ij"c} = 0$$

La tâche τ_{i4} de la transaction Γ_i, présentée dans la figure 2.11(b), déborde dans la deuxième période avec $S_{i4c} = 2$ unités de temps. Cette quantité ne contribue pas dans l'interférence de Γ_i durant la première période, tandis qu'elle est prise en compte pour les périodes suivantes (cela nécessite une opération supplémentaire de fusion).

Afin d'assurer une représentation statique de $W_i(\tau_{ua}, t)$, on doit stocker assez d'informations sur l'interférence maximum de Γ_i durant les deux premières périodes pour déduire correctement la valeur de $W_i(\tau_{ua}, t)$ dans des périodes d'activité arbitrairement longues. Puisque $T_i^{ind}(\tau_{ua}, t)$ est la seule partie d'interférence dépendant de t, il suffit seulement de représenter $T_i^{ind}(\tau_{ua}, t)$ statiquement.

Théorème 21 *[62, 63]*
Supposons que les tâches séparées soient prises en compte, et $t = k \times T_i + t'$ *avec* $k \in N$ *et* $0 \leq t' \leq T_i$ *alors*

$$T_i^{ind}(\tau_{ua}, t) = k \times T_i^{ind}(\tau_{ua}, T_i) + T_i^{ind}(\tau_{ua}, t')$$

2.3.8.2.2 Représentation statique. Puisque la tâche séparée (quantité débordante) n'est pas prise en compte dans l'interférence produite dans la première période, mais l'est pour la deuxième et les périodes suivantes, alors la représentation statique est divisée en deux cas, une pour calculer P_i correspond à la première période et l'autre cas pour calculerP_i' de la deuxième période et pour les périodes suivantes. Chaque point d'une table de stockage contient les cordonnées suivantes, la longueur de la période d'activité x, et l'interférence correspondante y. Un point est traduit comme suit : pour une période d'activité de longueur x, une interférence totale égale à y est imposée par Γ_i lorsque la tâche τ_{ic} initie l'instant critique.

– **La tâche séparée n'est pas prise en compte (première période)**

Pour chaque tâche candidate τ_{ic} de Γ_i, les trois opérations de restructuration Ordre, Fusion et Séparation présentées auparavant sont exécutées. Nous calculons les points de la table P_{ic} décrivant la partie de l'interférence induite par le temps $W_{ic}^{+}(\tau_{ua}, t)$ de Γ_i quand τ_{ic} initie l'instant critique. Chaque point de $P_{ic}[k]$ a la structure définie au-dessus. Les points contenus dans P_{ic} correspondent aux coins convexes de la courbe de la partie d'interférence $W_{ic}^{+}(t)$ (équation 2.23). Les formules suivantes définissent les points de la table P_{ic} :

$$\begin{aligned} P_{ic}[1].x &= 0 \\ P_{ic}[1].y &= \sum_{\forall j \in hp_i(\tau_{ua})} I_{ijc}^{Set1} - J_i^{ind}(\tau_{ua}) \\ &\dots \\ P_{ic}[k].x &= \Phi_{ikc} + C_{ik} \quad k \in 2..|\Gamma_i| \\ P_{ic}[k].y &= P_{ic}[k-1].y + C_{ik} \quad k \in 2..|\Gamma_i| \end{aligned} \tag{2.29}$$

$P_{ic}[1].y$ donne la relation initiale (i.e. la distance verticale à la date $t = 0$) entre les différents instants critiques candidats, elle est donnée par la différence des parties d'interférence induites par la gigue. En outre, l'interférence induite par le temps doit être nulle a la date $t = 0$, ce qui est obtenu par la soustraction du maximum des interférences induites par les gigues ($J_i^{ind}(\tau_{ua})$) pour initialiser la valeur de $P_{ic}[1].y$.

A cette étape du calcul, nous avons toutes les informations de la fonction d'interférence W_{ic}^{+} de Γ_i pour toutes les tâches τ_{ic} stockées dans l'ensemble des tables P_{ic}. Afin d'obtenir les points décrivant l'interférence maximum $T_i^{ind}(\tau_{ua}, t)$ nous devons trouver les points représentants le maximum de toutes les W_{ic}^{+}. Ces points seront stockés dans la table P_i.

Pour trouver les points de P_i (union de toutes les tables P_{ic}), au début P_i contient tous les points de toutes les tables P_{ic}, et à la fin, nous ne gardons dans P_i que les points représentant les points convexes maximum, en appliquant la relation « Envelopper » (subsume). Cette relation est définie comme suit :
Un point $P_i[a]$ enveloppe un autre point $P_i[b]$ (notons $P_i[a] \succ P_i[b]$), si la présence de $P_i[a]$ implique que $P_i[b]$ n'est pas un point convexe pour la courbe représentant le maximum des interférences.
Formellement, la relation « Envelopper » est définie par :

$$P_i[a] \succ P_i[b] \;\; SSI P_i[a].y \geq P_i[b].y \;\wedge\; (P_i[a].x - P_i[a].y \leq P_i[b].x - P_i[b].y) \tag{2.30}$$

D'où la table P_i finale contenant les points convexes de $T_i^{ind}(\tau_{ua}, t)$ est obtenue par la suppression de tous les points « Enveloppés » par d'autres points.

$$Supprimer\ P_i[b] \;\; de\ P_i \;\; SI\ \exists a \neq b \;:\; P_i[a] \succ P_i[b] \tag{2.31}$$

– **La tâche séparée est prise en compte (deuxième période et celles qui suivent)**

Le calcul de la table P'_i décrivant la fonction d'interférence $T_i^{ind}(\tau_{ua}, t)$ durant la deuxième période et les périodes suivantes est analogue au calcul de P_i sauf que la tâche séparée est prise en compte. Une opération d'ordre et de fusion sont nécessaires à cause de l'insertion de la tâche séparée.
Puisque P'_i représente la fonction d'interférence induite par le temps $T_i^{ind}(\tau_{ua}, t)$ pour $t \geq T_i$; $P'_{ic}[1].y$ doit refléter cette relation à la date de la fin de la première période. L'interférence de Γ_i lorsque la tâche τ_{ic} initie l'instant critique à la date T_i est représentée par $P_{ic}[|\Gamma_i|].y$. Par conséquent la relation initiale est définie par :

$$P'_{ic}[1].y = P_{ic}[|\Gamma_i|].y - \max_{x \in \Gamma_i} P_{ix}[|\Gamma_i|].y \tag{2.32}$$

La fonction d'interférence maximum $T_i^{ind}(\tau_{ua}, t)$ de la transaction Γ_i est représentée dans les deux tables P_i et P'_i, Afin de pouvoir déduire l'interférence de Γ_i pour n'importe quelle période d'activité assez longue, nous transformons les deux tables P_i et P'_i en tables V_i et V'_i respectivement. Ces deux dernières tables représentent les points concaves de la courbe d'interférence maximum, et elles seront facilement exploitées dans la déduction de l'interférence totale.

$$\begin{aligned} V_i[k].y &= P_i[k].y \\ V_i[k].x &= \begin{cases} P_i[k+1].x - (P_{ik}[k+1].y - P_{ik}[k].y) & \text{if } k < |\Gamma_i| \\ P_i[k].x \text{ if } k = |\Gamma_i| \end{cases} \\ k &\in 1..|\Gamma_i| \end{aligned} \tag{2.33}$$

Les points de V_i et V'_i sont utilisés pour calculer l'interférence maximum induite par le temps $T_i^{ind}(\tau_{ua}, t)$ dans n'importe quelle période d'activité. L'interprétation des tables V_i et V'_i est la suivante : pour $t \leq T_i$, $V_i[k].y$ représente l'interférence maximum induite par le temps $(T_i^{ind}(\tau_{ua}, t))$ que la transaction Γ_i peut produire dans une période d'activité de longueur $V_i[k].x$ $(k \in 1..|V_i|)$.

2.3.8.2.3 Déduction de la valeur de l'interférence totale induite par le temps $T_i^{ind}(\tau_{ua}, t)$

L'interférence est représentée séparément pour les deux premières périodes dans les deux tables V_i et V'_i. Nous utilisons ces deux tables pour calculer l'interférence maximum de la transaction dans n'importe quelle période d'activité. Cette opération peut être réduite à une fonction de déduction. Pour $t \leq T_i$ nous n'utilisons que V_i, et pour $t > T_i$ nous utilisons les deux tables (V_i et V'_i) dans le calcul. La fonction de déduction est définie comme suit :

$$
\begin{aligned}
T_i^{ind}(\tau_{ua}, t) &= \begin{cases} V_i[n].y \text{ si } k < 1 \\ V \text{ si } k \geq 1 \end{cases} \\
V &= V_i[|V_i|].y + (k-1) \times V_i'[|V_i'|].y + V_i'[n'].y \\
k &= t \; Div \; T_i \\
t^* &= t \; rem \; T_i \\
n &= min\{m : t^* \leq V_i[m].x\} \\
n' &= min\{m : t^* \leq V_i'[m].x\}
\end{aligned} \tag{2.34}
$$

k représente le nombre de périodes complètes (T_i) dans t, et t^* est la partie de t étendue dans la dernière période. L'algorithme de cette implémentation est présenté dans l'annexe B.1.

2.3.8.3 Bilan

La méthode d'analyse présentée ci-dessus a une complexité pseudo-polynomiale. Mais malgré l'élimination de la surestimation de la fonction d'interférence maximum en utilisant la fonction d'interférence effective, cette méthode reste pessimiste.

2.3.9 Analyse des tâches à offset dynamique

Dans cette section, nous présentons comment adapter l'analyse précédente pour des tâches à offsets dynamiques. Pour ce type de modèle, l'offset d'une tâche n'est pas fixe, mais il peut varier d'une activation à une autre avec une valeur limitée par une borne minimale et maximale [70].

$$O_{ij} \in [O_{ij,min}, O_{ij,max}]$$

Dans l'analyse des tâches à offsets fixes, l'offset représente l'intervalle de temps minimal qui sépare l'activation de la tâche et l'arrivée de l'évènement externe (activation de la transaction). D'autre part, la gigue représente le délai maximum par lequel l'activation d'une tâche peut être retardée. D'où on peut modéliser facilement le cas des tâches à offsets dynamiques comme un cas particulier des tâches à offsets fixes, en redéfinissant un nouvel offset fixe et une gigue sur activation de chaque tâche τ_{ij} comme suit :

$$
\begin{aligned}
O_{ij}' &= O_{ij,min} \\
J_{ij}' &= J_{ij} + O_{ij,max} - O_{ij,min}
\end{aligned} \tag{2.35}
$$

Par conséquent, afin d'analyser l'ordonnançabilité des tâches à offset dynamique il suffit d'analyser les tâches à offsets fixes équivalentes. Notons que l'étude des offsets dynamiques est utile pour les tâches a suspension ou des systèmes distribués [69].

2.4 Analyse d'ordonnançabilité des tâches à offset avec priorités dynamiques (EDF)

Dans cette section, nous présentons l'extension de l'analyse de temps de réponse des tâches indépendantes ordonnancées par un algorithme d'ordonnancement dynamique *EDF* [102, 103] pour des tâches à offsets (transactions) avec priorités dynamiques [71]. La définition de l'instant critique pour une tâche analysée τ_{ua} dans le contexte des priorités dynamiques est différente de celle utilisée pour les priorités fixes (section 2.3) car l'ensemble des tâches plus prioritaires qu'une tâche τ_{ua} varie au cours du temps.

Le théorème 22 montre que le pire temps de réponse d'une tâche τ_{ua} peut être trouvé dans une période d'activité initiée par l'activation simultanée d'une tâche τ_{ic} de chaque transaction Γ_i ($i \neq u$) après avoir été retardée par sa gigue maximale.

Théorème 22 *[71] La pire contribution d'une transaction Γ_i sur le temps de réponse d'une tâche analysée τ_{ua} se produit lorsque la première activation d'une certaine tâche τ_{ic} coïncide avec le début de la période d'activité après avoir retardée par une valeur maximum de gigue.*

Pour trouver le pire temps de réponse, nous devons alors examiner toutes les périodes d'activité possibles en combinant toutes les tâches des transactions initiant la période d'activité. Pour une période d'activité initiée par une tâche τ_{ic} d'une transaction Γ_i, les instances d'une tâche τ_{ij} sont classées en trois ensembles Set0, Set1 et Set2. La définition de ces ensembles est la même que celle présentée pour les tâches avec priorités fixes.

La pire contribution d'une tâche τ_{ij} est produite lorsque les instances de l'ensemble Set1 sont retardées par une quantité de gigue telle que toutes s'activent au début de la période d'activité et les instances de l'ensemble Set2 s'activent à leurs rythmes maximum (i.e. avec gigues nulles, et à la période T_i) (théorème 3 dans [71]).

2.4.1 Fonction d'interférence

Puisque l'ordonnancement dynamique (EDF) d'une tâche est basé sur l'échéance de l'instance étudiée, pour calculer le temps de réponse d'une instance τ_{ua} avec une échéance survenant à la date D, il est nécessaire de déterminer la pire interférence $W_{ic}(t, D)$ de l'ensemble des tâches de chaque transaction Γ_i dans une période d'activité de longueur t avec échéance D. Pour cela on ne doit considérer que les instances activées dans $[0, t)$ mais avec un échéance survenant à ou avant D.

Le nombre d'instances (Set1 plus Set2) d'une tâche τ_{ij} activées dans $[0, t)$ lorsque une tâche τ_{ic} initie la période d'activité est donné par :

$$\left\lfloor \frac{J_{ij} + \Phi_{ijc}}{T_i} \right\rfloor + \left\lceil \frac{t - \Phi_{ijc}}{T_i} \right\rceil \tag{2.36}$$

Maintenant, nous déterminons le nombre d'instances ayant un échéance à ou avant D qui est donné par la formule suivante :

$$\left\lfloor \frac{J_{ij} + \Phi_{ijc}}{T_i} \right\rfloor + \left\lceil \frac{D - \Phi_{ijc} - d_{ij}}{T_i} \right\rceil + 1 \tag{2.37}$$

D'où l'interférence $W_{ijc}(t, D)$ d'une tâche τ_{ij} dans une période d'activité t avec un échéance D est :

$$W_{ijc,Palencia}(t, D) = \left(\left\lfloor \frac{J_{ij} + \Phi_{ijc}}{T_i} \right\rfloor + \min \left(\left\lceil \frac{t - \Phi_{ijc}}{T_i} \right\rceil, \left\lceil \frac{D - \Phi_{ijc} - d_{ij}}{T_i} \right\rceil + 1 \right) \right)_0 C_{ij} \tag{2.38}$$

Finalement, l'interférence totale d'une transaction Γ_i est donnée par la somme des interférences de toutes ses tâches.

$$W_{ic,Palencia}(t, D) = \sum_{\forall j \in \Gamma_i} W_{ijc,Palencia}(t, D) \tag{2.39}$$

Comme dans l'analyse à priorité fixe, nous devons étudier toutes les périodes d'activité afin de trouver le pire temps de réponse exacte d'une tâche analysée. Cette analyse exhaustive a une complexité exponentielle vu que le nombre de périodes d'activité à examiner est égal au produit des nombres de tâches des transactions. Une méthode avec une complexité pseudo-polynomiale qui donne une borne approchée de pire temps de réponse a été proposée dans [71]. Nous la présentons dans la section suivante.

2.4.2 Méthode d'analyse approchée

Cette approche est similaire à celle proposée pour les tâches à priorités fixes [107, 70]. Elle permet d'obtenir une borne supérieure du pire temps de réponse en utilisant une borne d'interférence $W_{i,gonzalez}(t, D)$ (fonction d'interférence maximum) pour chaque transaction Γ_i. Cela évite d'examiner toutes les combinaisons de tâches possibles.

$$W_{i,Palencia}(t, D) = \max_{\forall c \in \Gamma_i} W_{ic,Palencia}(t, D) \tag{2.40}$$

[102] a montré que l'instant critique produisant le pire temps de réponse d'une tâche analysée τ_{ua} survient au début de la période d'activité ou à un instant auquel l'échéance de τ_{ua} coïncide avec l'échéance d'autre tâche d'une autre transaction. Pour chaque instant critique, il est nécessaire de déterminer le temps de réponse de chacune des instances de τ_{ua} activées dans la période d'activité. La même numérotation des instances proposée dans la section précédente (équations 2.7-2.9) pour les tâches à priorités fixes est gardée dans ce cas.

L'ensemble Ψ des instants critiques à considérer correspond alors à l'ensemble des échéances de toutes les instances de toutes les tâches de toutes les transactions.

$$\Psi = \bigcup_{\forall \Gamma_i} \{\Phi_{ijc} + (p-1)T_i + d_{ij}\} \quad \forall p = p_{0,ijc} ... p_{L,ijc}, \forall c, j \in \Gamma_i \tag{2.41}$$

Pour un instant critique, soit A la date de la première activation de τ_{ua} après le début de la période d'activité, la date de fin d'exécution de l'instance p est donnée par l'équation itérative suivante, en ajoutant la contribution de toutes les transactions.

$$w^A_{uac}(p) = (p - p_{0,uac} + 1)C_{ua} + W^-_{uc}(w^A_{uac}(p), D^A_{uac}(p)) + \sum_{\forall i \neq u} W_i(w^A_{uac}(p), D^A_{uac}(p)) \tag{2.42}$$

Où $W^-_{uc}(w^A_{uac}(p), D^A_{uac}(p))$ est l'interférence de la transaction Γ_u sans prendre en compte l'interférence de la tâche analysée τ_{ua}, et $D^A_{uac}(p)$ est l'échéance de l'instance étudiée p lorsque la première instance est activée à la date A. Cette échéance est définie par :

$$D^A_{uac}(p) = A + \Phi_{uac} + (p-1)T_a + d_{ua} \tag{2.43}$$

Par conséquent le temps de réponse de l'instance est la différence entre sa date d'activation et sa date de fin d'exécution. Pour chaque p nous devons examiner seulement ceux qui ont la première activation A entre 0 et T_u, les instants critiques vérifiant cette condition doivent être analysés.

$$\Psi^* = \{\Psi_x \in \Psi \ / \ \Phi_{uac} + (p-1)T_a + d_{ua} \leq \Psi_x < \Phi_{uac} + pT_a + d_{ab}\} \tag{2.44}$$

Notons que la date de la première activation est déduite à partir de l'échéance Ψ_x d'une instance p d'une tâche τ_{ij} considérée.

$$A = \Psi_x - [\Phi_{ijc} + (p-1)T_u + d_{ua}] \tag{2.45}$$

2.4.3 Bilan

L'analyse exacte de pire temps de réponse des tâches à offsets avec priorités dynamiques est exponentielle. La seule méthode approchée proposée par Palencia et Harbour [71] fournit une borne pessimiste du pire temps de réponse avec un complexité pseudo-polynomiale. Cette complexité est plus élevée que celle de l'analyse approchée proposée pour les tâches à priorités fixes selon le critère du nombre d'instants critiques à considérer, qui correspond à toutes les échéances de toutes les instances survenant dans la période d'activité.

Tous ce que nous avons présenté jusque là concerne l'ordonnançabilité des transactions non concrètes avec priorités fixes ou dynamiques. De nombreux travaux ont été proposés pour analyser des transactions concrètes (supposons que les dates des premières activations des transactions sont connues). Ainsi les travaux de Pellizzoni et Lipari [75, 76, 73] ont proposé une méthode de

calcul de temps de réponse et d'analyse holistique pour des transactions concrètes asynchrones (chaque transaction possède un offset d'activation en plus des offset des tâches) avec un ordonnancement à priorités dynamiques. D'autres travaux ont étudié l'analyse de systèmes avec algorithmes d'ordonnancement hybride à priorités fixes et dynamiques [58, 43]. Pop et al [77] a étudié l'analyse de transactions dans le domaine des systèmes déclenchés par le temps et par évènement (time-triggered and event-triggered). Des systèmes avec ordonnancement hybride et avec un motif d'arrivée complexe pour les tâches ont été étudiés dans [87, 37].

2.5 Autres modèles prenant en compte des décalages d'activations

2.5.1 Transactions séries

Les transactions séries ont été proposées par Traoré [112, 110] pour modéliser certaines applications particulières, comme des tâches générées par des périphériques connectés sur les ports de type série (RS232 et CAN). Deux types de tâches sont générées :

- Tâches d'acquisition qui sont exécutées plusieurs fois par période de périphérique, elles sont chargées de récupérer les données transmises par le périphérique série.
- Tâches de traitement qui sont effectuées une fois par période du périphérique, elles sont chargées de traiter les données reçues par les tâches d'acquisition.

Une définition des transactions séries est donnée dans [110] précisant les différentes contraintes temporelles exigées par le mode de fonctionnement de ce type de tâches. Une transaction série est composée de $n-1$ tâches d'acquisition ayant la même priorité et même durée d'exécution, et une tâche de traitement exécutée à la fin de la transaction avec une durée d'exécution supérieure à celle des tâches d'acquisition et avec une priorité inférieure.

Définition 5 *Une transaction série Γ_i est une transaction avec les contraintes suivantes :*

- *gigues nulles :* $\forall j, 1 \leq j \leq |\Gamma_i| : J_{ij} = 0$
- *le motif d'arrivée des tâches dans la transactions est régulier et égal à p_i :* $\forall j, 1 \leq j \leq |\Gamma_i| : O_{ij} = (j-1) \times p_i$
- *il y a deux types de tâches dans la transaction*
  - *Les $L_i = |\Gamma_i| - 1$ tâches d'acquisition :* $\forall j, 1 \leq j \leq L_i : \tau_{ij} =< C_i, (j-1)p_i, p_i, 0, B_{ij}, P_i >$.
  - *La tâche de traitement* $\tau_{i|\Gamma_i|} =< C_{in}, L_i p_i, D_{in}, 0, B_{ij}, P_{in} >$.
- *avec $C_{in} > C_i$, $D_{in} > p_i$, $P_{in} < P_i$ et $(T_i - L_i \times p_i) - C_{in} >> p_i - C_i$. Cette dernière inéquation signifie que le temps séparant la fin d'une tâche et le début de la tâche suivante dans la transaction série est beaucoup plus important pour la dernière tâche de la transaction série (c'est-à-dire la tâche de traitement).*

La figure 2.12 présente une transaction série Γ_i composée de 4 tâches d'acquisition et une seule tâche de traitement.

Dans le contexte de l'ordonnancement des transactions séries avec priorités fixes, une tâche analysée τ_{ua} est dite intermédiaire pour une transaction série Γ_i si la priorité de τ_{ua} est plus faible

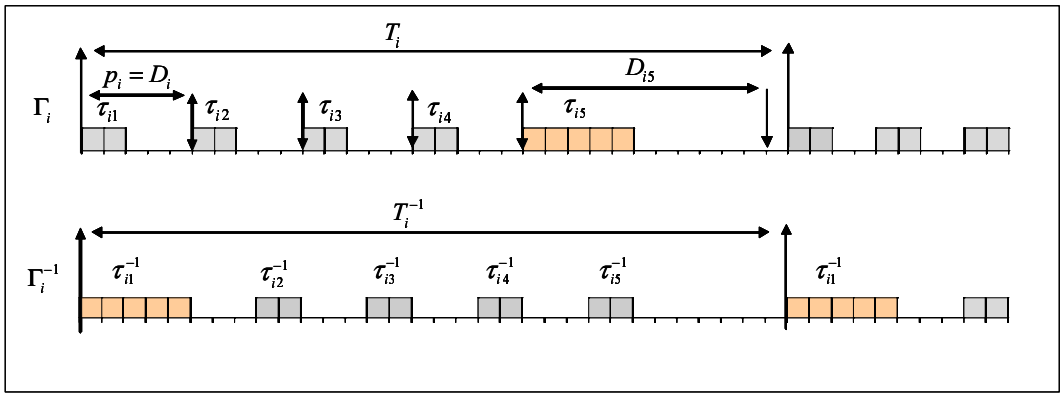

FIG. 2.12 – Exemple d'une transaction série et sa transaction inverse.

que celles des tâches d'acquisition de Γ_i et plus élevée que celle de la tâche de traitement de Γ_i. Lorsque la priorité de τ_{ua} est inférieure à celle de toutes les tâches de Γ_i, on dit que τ_{ua} est faible pour Γ_i. Si la tâche τ_{ua} est intermédiaire pour une transaction Γ_i, alors que les tâches d'acquisition qui contribuent à l'interférence de Γ_i, et l'instant critique est initié par la première tâche d'acquisition de Γ_i [112, 110]. Par conséquent si τ_{ua} est intermédiaire pour toutes les transactions du système, l'instant critique produisant le pire cas est alors initié par la première tâche de chacune des transactions.

Afin d'accélérer le processus de l'analyse approchée de temps de réponse d'une tâche τ_{ua} faible (dans contexte de priorités fixes), [113] définit une transaction inverse Γ_i^{-1} pour chaque transaction Γ_i et montre que l'interférence maximum de Γ_i obtenue par Nolin dans n'importe quelle période d'activité t est équivalente à celle de la transaction inverse correspondante Γ_i^{-1} lorsque la première tâche (tâche de traitement) de Γ_i^{-1} initie la période d'activité.

$$W_{i,Nolin}(\tau_{ua}, t) = W^{-1}_{i1,Nolin}(\tau_{ua}, t) \tag{2.46}$$

La figure 2.12 presente une transaction inverse Γ_i^{-1} de la transaction Γ_i. La transaction inverse Γ_i^{-1} d'une transaction série Γ_i est donnée par la transformation suivante

$$\begin{aligned} \tau_{i1}^{-1} &= < C_{in}, 0, X, 0, B_{in}, P_{in} > \\ \tau_{ij}^{-1} &= < C_i, C_{in} - C_i + (j-1)p_i, p_i, 0, B_{i(j-1)}, P_{i(j-1)} > .\ j = 2...n. \end{aligned} \tag{2.47}$$

Notons que cette méthode permet seulement d'accélérer l'analyse approchée dans le cas des transactions série, mais elle n'apporte aucune amélioration sur la borne de temps de réponse.

2.5.2 Transactions monotoniques

Le concept (ou la propriété) des transactions monotoniques a été proposé par Traoré [111, 114] pour caractériser certain type de transactions avec gigues nulles. Une transaction Γ_i est monotonique pour une tâche analysée τ_{ua} si et seulement si l'interférence maximum de Γ_i sur le temps

de réponse de τ_{ua} correspond toujours à son interférence lorsque une unique tâche candidate τ_{ik} initie l'instant critique. Cette propriété permet de déterminer quelle tâche candidate (τ_{ik}) de Γ_i initie l'instant critique pire cas pour τ_{ua}. Notons qu'une transaction Γ_i peut être monotonique pour une tâche et non pour d'autres tâches en fonction de l'ensemble de tâches plus prioritaires que la tâche analysée. Cette propriété fait l'objet de notre étude dans le chapitre 4.

2.5.3 Tâches à suspension

Les tâches à suspension [92, 93] qui suspend leurs exécution pour exécuter une opération externe. Une tâche τ_i est dite à suspension si durant son exécution elle peut demander l'exécution d'une opération externe et à la fin de l'exécution de l'opération externe elle reprend son exécution. L'exécution de τ_i est alors suspendue durant l'exécution de l'opération externe.

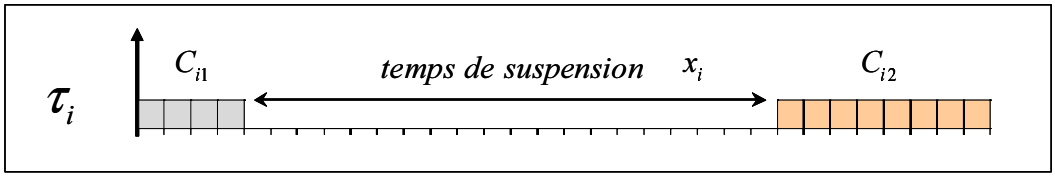

FIG. 2.13 – Tâche à suspension.

Une tâche à suspension τ_i (voir figure 2.13) peut être alors redéfinie par deux sous tâches τ_{i1} et τ_{i2} avec une durée d'exécution C_{i1} et C_{i2} respectivement. Ces deux sous tâches sont séparées par une durée maximale de suspension (durée d'exécution de l'évènement externe) entre la fin de la sous tâche τ_{i1} et le début de la sous tâche τ_{i2} [91]. Dans [91, 92] il a été prouvé que l'ordonnancement des tâches périodiques avec au plus une suspension par tâche et un délai critique implicite est $NP-complet$ au sens fort (théorème 2.2.8 de [110]). Nous étudions la possibilité de modéliser une tâche à suspension comme une transaction. Si nous considérons une tâche à suspension τ_i comme une transaction dont nous devons définir les paramètres (offset et gigues) définissant le motif d'activation des tâches τ_{i1} et τ_{i2} (sous tâches) dans la transaction.

Puisque τ_{i1} s'active avec l'activation de la tâche (transaction) τ_i, alors elle est définie par un offset nul $O_{i1} = 0$ et une gigue $J_{i1} = 0$ car aucun retard d'activation n'est probable. Afin de modéliser la loi d'activation de la tâche τ_{i2}, supposons que la durée de suspension X_i est comprise entre deux bornes $X_{i,min}$ et $X_{i,max}$ (valeurs connues) ; d'où τ_{i2} s'active après la fin d'exécution de τ_{i1} (i.e. le temps de réponse de τ_{i1}) et de l'opération externe X_i.

$O_{i2} = R_{i1} + X_i$

L'offset O_{i2} est dynamique (varie d'une instance à une autre) en fonction du temps de réponse R_{i1}, de l'instance précédente de τ_{i1}. Nous devons alors le transformer en offset fixe (section 2.3.9). Pour cela nous obtenons :

$O'_{i2} = O_{i2,min} = R_{i1,min} + X_{i,min}$

$J'_{i2} = O_{i2,max} - O_{i2,min} = R_{i1,max} + X_{i,max} - O_{i2,min}$

Notons que le temps de réponse de R_{i1} dépend de la gigue J_{i2}, d'où la détermination de modèle

fixe équivalent demande un calcul itératif. Un algorithme (WCDO : Worst-Case analysis for Dynamique Offset) a été proposé dans ce sens par [69].

2.6 Modèle de tâches Multiframes

Nous utilisons une notation similaire à celle utilisée pour les transactions afin de faciliter la présentation des differents concepts déja vus.
Le modèle des tâches multiframes (MF) a été proposé par Mok et Chen [67, 68] afin de réduire le pessimisme d'analyse d'ordonnançabilité induit par la considération du modèle de tâches périodiques proposé par Liu et Layland [55] pour certains types d'applications temps réel. Le modèle de Liu et Layland considère le pire temps d'exécution pour caractériser une tâche. Cette supposition rend pessimiste l'analyse d'ordonnançabilité lorsque le temps d'exécution moyen est très faible par rapport au pire temps d'exécution. Une surestimation sur la charge processeur est considérée, ce qui réduit le taux d'acceptation (de validation) de systèmes.

Soit un système Γ composé de deux tâches périodiques synchrones Γ_1 et Γ_2 avec une période 5 et 3 unités de temps respectivement. Γ_1 est de durée d'exécution $C_1 = 2$ unités de temps, tandis que le temps nécessaire pour exécuter Γ_2 prend alternativement deux valeurs différentes, $C_{21} = 2$ unités de temps pour les périodes impaires et $C_{22} = 1$ pour les périodes paires. Un exemple réel d'une tâche ayant le même fonctionnement que Γ_2 (donné dans [110]) est celui d'une tâche chargée d'envoyer des informations périodiquement au modem dans une application drone miniature [110], mais elle envoie alternativement, une fois les informations d'altitude (environs 10 octets) et une fois les informations GPS (environs 25 octets). Revenons sur notre exemple, lorsque le système est modélisé par L&L, alors le pire temps d'exécution est considéré pour chacune des tâches, par conséquent, en appliquant les conditions d'ordonnançabilité de L&L, le système est non ordonnançable car la charge processeur du système est supérieure à 1.

$$U = \sum \frac{C_i}{T_i} = \frac{2}{5} + \frac{2}{3} = 1,066 > 1$$

En réalité, le système est ordonnançable en prenant en compte pour chaque période la durée d'exécution correspondante (figure 2.14).

Le modèle de tâches multiframes permet donc de caractériser une tâche par plusieurs valeurs de durée d'exécution. Elle consiste à permettre à une tâche d'avoir un nombre fini de durées d'exécution telle que ses exécutions suivent un motif périodique connu a priori. La prise en compte de ces ensembles de temps d'exécution de chacune de tâches conduit à établir des conditions d'ordonnançabilité meilleures que celles établies par L&L [55]. Nous retrouvons ce genre de tâches dans les applications de transmission de trames vidéo encodées avec les normes MPEG [5, 40]. Dans le codage MPEG (Moving Picture Experts Group), il existe trois types de trames : les $I - frame$, $P - frame$ et $B - frame$ [57]. La taille de $I - frame$ est plus grande que celle de P et $B - frame$. La transmission des trames suit le motif suivant : « IBBPBBPBBIBBP... ».

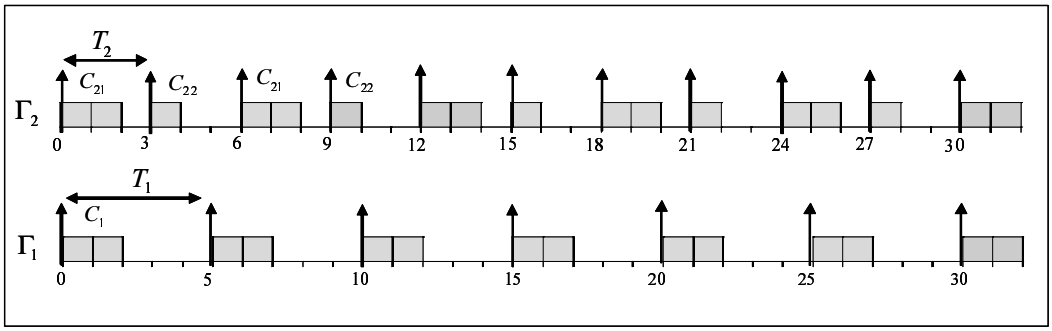

FIG. 2.14 – Exemple d'une tâche multiframes.

2.6.1 Présentation du modèle

Une tâche temps réel multiframe est définie par un couple (Γ_i, T_i), où Γ_i est un tableau de N_i ($N_i \geq 1$) trames $(\tau_{i1}, \tau_{i2}, ..., \tau_{iN_i})$ avec temps d'exécution $(C_{i1}, C_{i2}, .., C_{iN_i})$ respectivement, et T_i est le temps minimum de séparation entre deux trames successives. C'est-à-dire que deux activations successives doivent être séparées par un temps minimum T_i. Chaque trame τ_{ij} doit terminer son exécution avant un délai critique $D_{ij} = T_i$. Toutes les trames d'une même tâche ayant le même niveau de priorité. Supposons que chaque tâche possède une priorité unique. Notons que les tâches indépendantes de L&L peuvent être modélisées comme des tâches multiframes avec un tableau de temps d'exécution d'une seule valeur (le pire temps d'execution). Pour l'exemple de la figure 2.14 la tâche Γ_2 est composée de deux trames notées τ_{21} et τ_{22} avec des durées d'exécution C_{21} et C_{22} respectivement. Formellement cette tâche multiframe est présentée comme suit :

$$(\Gamma_2, T_2) = ((\tau_{21}, \tau_{22}), T_i) = ((2, 1), 3)$$

A chaque tâche multiframes on peut associer une tâche de L&L en prenant seulement le pire temps d'exécution parmi les temps d'exécution de ses trames. Nous trouvons dans la littérature plusieurs travaux qui ont étudié les conditions d'ordonnançabilité des tâches multiframes à priorité fixes [67, 68] comme à priorités dynamiques en utilisant en particulier deux techniques : analyse de facteur d'utilisation [67, 68, 57] et l'analyse de temps réponse [13, 118].

2.6.2 Analyse d'ordonnançabilité des tâches multiframes à priorités fixes

2.6.2.1 Analyse basée sur la densité

Pour une analyse moins pessimiste que celle proposée par L&L [55], Mok et Chen [67, 68] ont prouvé que l'algorithme d'ordonnancement à priorités fixe RM (Rate Monotonic) est optimal pour des tâches multiframes satisfaisant la propriété d'Accumulativité Monotonique (AM).

Définition 6 *(Tâche AM) Soit Γ_i une tâche MF composée de N_i trames, avec des temps d'exécution $(C_{i1}, C_{i2}, .., C_{iN_i})$ respectivement. On dit que Γ_i est AM si $\exists m \in (1 : N_i)$ tel que $\forall l, j \in (1 : N_i)$, $\sum_{k=m}^{m+l} C_{ik} \geq \sum_{k=j}^{j+l} C_{ik}$. On appel τ_{im} la trame de pointe de Γ_i.*

Notons que la trame de pointe τ_{im} est la trame ayant la plus grande durée d'exécution C_{im} dans Γ_i. Intuitivement, une tâche AM est une tâche pour laquelle la plus grande quantité de temps d'exécution totale de n'importe quelle séquence de $l \geq 1$ trames correspond à celle dans laquelle la première trame est la trame de pointe. Sans perte de généralité, nous supposons que la première trame de chaque tâche AM est la trame de pointe (i.e. $\tau_{im} = \tau_{i1}$). En effet si ce n'était pas le cas, cela le serait à une rotation près.

Supposons que toutes les tâches soient AM. Un instant critique produisant le pire scénario d'exécution d'une tâche survient lorsque sa trame de pointe s'active simultanément avec toutes les trames de pointe de toutes les tâches plus prioritaires. Une tâche MF est ordonnançable si elle est ordonnançable à son instant critique. Une borne d'utilisation est meilleure que celle de L&L est fournie par le théorème 23 [67, 68]. Pour une tâche MF $\Gamma_i = ((\tau_{i1}, \tau_{i2}, .., \tau_{iN_i}), T_i)$ nous définissons : $r_i = \frac{C_{i1}}{C_{i2}}$ si $N_i > 1$, sinon $r_i = 1$. Pour un système de n tâches nous définissons $r = \min_{i=1}^{n} r_i$.

Théorème 23 *[67] Pour un système* Γ *de n tâches MF,* Γ *est ordonnançable si le facteur d'utilisation (*$U = \sum_{i=1}^{n} \frac{\max_{j=1}^{N_i} C_{ij}}{T_i}$*) est inférieur à*

$$r.n.\left(\left(\frac{r+1}{r}\right)^{1/n} - 1\right) \tag{2.48}$$

Soit Γ un système de n tâches périodiques avec périodes T_i $(i = 1..n)$ ordonnées par un ordre croissant de périodes ($i \leq j$ alors $T_i \leq T_j$). Γ est dite harmonique si pour tout indice $i < j$, T_j est un multiple de T_i. Un système de n tâches harmoniques et AM [42] est ordonnançable par RM si pour tout $1 \leq k \leq n$:

$$W_k = \sum_{i=1}^{k} \sum_{j=1}^{\frac{T_k}{T_i}} C_{i(j \mod (N_i+1))} \leq T_k \tag{2.49}$$

Ces conditions d'ordonnançabilité ont été établies pour des tâches AM. Elles ne sont plus efficaces pour des systèmes contenant des tâches non AM. Un test suffisant d'ordonnançabilité pour des tâches MF arbitraires (non AM) a été proposé en transformant chaque tâche MF Γ_i du système Γ en une tâche AM Γ'_i. Si le système obtenu Γ' (des tâches AM) est ordonnançable alors le système d'origine, Γ l'est aussi. Une nouvelle tâche Γ'_i de Γ' modélise la charge maximale (périodique) de la tâche d'origine correspondante Γ_i en supposant que chacune de ses trames est celle qui débute chaque séquence de l trames. Cette méthode est similaire à la fonction d'interférence approchée proposée pour les tâches à offsets. Dans [49] Kuo et al, le test est amélioré en réduisant le nombre de tâches par fusion des tâches harmoniques durant le test. Lu et al [57] ont proposé une autre borne d'ordonnançabilité en utilisant des ratios de facteurs d'utilisation et de périodes.

2.6.2.2 Analyse de temps de réponse

Dans [13], Baruah a proposé un test suffisant d'ordonnançabilité par un analyse approchée de temps de réponse. L'idée est la même que celle proposée pour les transactions, elle consiste à uti-

liser une fonction qui détermine la demande d'exécution cumulative d'une tâche pour n'importe quelle séquence de k instances.

$$g(\Gamma_i, k) = \max_{j=1}^{N_i} \left\{ \sum_{l=j}^{j+k} C_{(i(l \mod N_i+1))} \right\} \tag{2.50}$$

Cette fonction fournit l'interférence maximum de k exécutions successives d'une tâche Γ_i. $g(\Gamma_i, k)$ est égal à la somme des temps d'exécutions de k trames de Γ_i. Afin de calculer l'interférence en fonction de la longueur de l'intervalle de temps t, nous devons déterminer le nombre d'instances pouvant être activées dans t.

$$G(\Gamma_i, t) = g(\Gamma_i, \left\lceil \frac{t}{T_i} \right\rceil) \tag{2.51}$$

Appelons la trame de pointe d'une tâche Γ_i (non AM) la trame possédant la plus grande durée d'exécution C_{ip} ($g(\Gamma_u, 1) = C_{ip}$). Une tâche Γ_u respectera toujours son échéance si sa trame de pointe respecte son échéance (théorème 1 dans [13]) car les trames d'une tâche ont les mêmes conditions d'ordonnançabilité et sont à échéance sur requêtes. Γ_u est alors ordonnançable si le pire temps de réponse de la trame de pointe est inférieur à son échéance. La borne de temps de réponse est obtenue par la méthode RTA de Joseph et Pandya [47].

$$\begin{aligned} R_u^0 &= g(\Gamma_u, 1) \\ R_u^k &= g(\Gamma_u, 1) + \sum_{\Gamma_i \in hp(\Gamma_u)} G(\Gamma_i, R_u^{k-1}) \end{aligned} \tag{2.52}$$

Un instant critique pour une tâche analysée Γ_u survient lorsque sa trame de pointe s'active simultanément avec une trame de chacune des tâches plus prioritaires [118]. Pour un analyse exacte du pire temps de réponse, on doit considérer tous les instants candidats à produire le pire temps de réponse en combinant toutes les trames de toutes les tâches plus prioritaires. La complexité de l'algorithme est exponentielle. Une réduction du nombre d'instants critiques qui doivent être examinés est possible [118], en détectant et éliminant tout instant qui ne pourra jamais produire le pire temps de réponse de la tâche analysée. Ces instants critiques coïncidant avec des l'activation des trames dominées par d'autres trames. La trame τ_{ix} est dominée par la trame τ_{iy} si et seulement si

$$W_{iy}(k) \geq W_{ix}(k).\ \forall k = 1, 2, .., N_i.$$

Où $W_{il}(k)$ est l'interférence de la tâche Γ_i pour k instances, en commençant par la trame τ_{il}.

2.7 Modèle des tâches Multiframes généralisées

Afin de relaxer les suppositions et les contraintes des tâches MF, Baruah et al [12] ont proposé le modèle de tâches multiframes généralisées (GMF). Les tâches MF généralisées sont, bien entendu, des tâches MF avec moins de restrictions sur les paramètres temporelles. Ce modèle GMF permet aux tâches d'avoir :

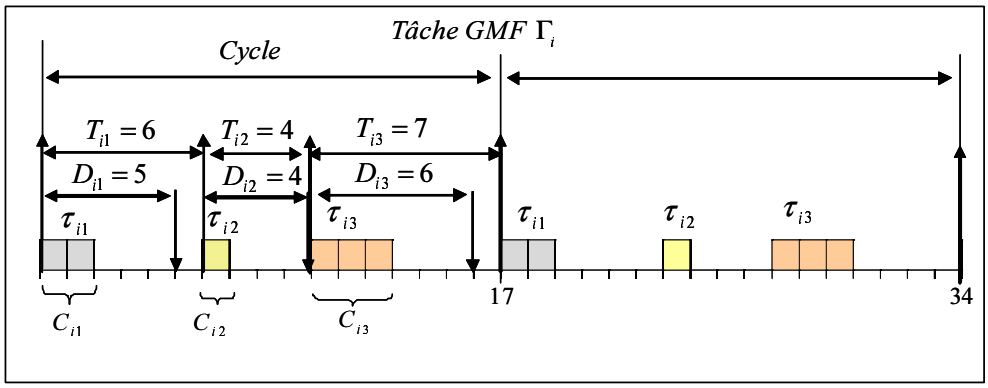

FIG. 2.15 – Exemple d'une tâche multiframes généralisée (GMF).

1. Des délais critiques différents de la période (temps minimum séparant deux activations successives), de plus les trames n'ont pas forcément le même délai critique.
2. Le temps minimum séparant l'activations de deux trames successives n'est pas forcément le même pour toutes les trames.

Formellement, une tâche GMF Γ_i est composée de N_i trames $(\tau_{i1}, \tau_{i2}, ..., \tau_{iN_i})$. Chaque trame τ_{ij} est caractirisée par un temps d'exécution C_{ij}, un délai critique D_{ij} et un temps minimum de séparation T_{ij} entre son activation et celle de la trame suivante.

La figure 2.15 présente une tâche GMF Γ_i contenant trois trames ($N_i = 3$) τ_{i1}, τ_{i2} et τ_{i3}. La première trame a une durée d'exécution $C_{i1} = 2$ et un délai critique $D_{i1} = 4$, l'activation de la trame suivant τ_{i2} survient $T_{i1} = 6$ unités de temps après l'activation de τ_{i1}. Les trois trames ont des délais critiques et des temps de séparation différents.

Avant d'aborder les différentes approches d'analyse des tâches GMF (avec priorités fixes et dynamiques), nous citons quelques propriétés qui ont été définies dans la littérature.

– Une tâche GMF Γ_i satisfait la propriété $L - MAD$ (localized Monotonic Absolute Deadlines) si elle satisfait la condition suivante : L'échéance de chaque trame τ_{ij} survient avant l'échéance de la trame suivante $\tau_{i(j+1)}$ [12]. C'est-à-dire pour tout j $(1 \leq j \leq N_i - 1)$

$$D_{ij} \leq T_{ij} + D_{i((j+1) \bmod N_i+1)}$$

– Une tâche GMF satisfait la propriété FS lorsque le délai critique de chaque trame τ_{ij} survient avant le réveil de la trame suivante $\tau_{i(j+1)}$ [12]. Pour tout j $(1 \leq j \leq N_i - 1)$:

$$D_{ij} \leq T_{ij}$$

Il est clair que la propriété FS est plus restrictive que la propriété $L - MAD$. Toute tâche satisfaisant la propriété FS satisfait forcément la propriété $L - MAD$.

2.7.1 Analyse de temps de réponse des tâches GMF à priorités fixes

Nous présentons dans cette section un test suffisant d'ordonnançabilité des tâches GMF satisfaisant la propriété FS, en utilisant la technique d'analyse de temps de réponse [7, 96]. Soit $W_{ik}(t)$

la fonction qui détermine la charge totale causée par une tâche Γ_i dans un intervalle de temps t lorsque elle débute par la k^{eme} trame [12].

$$W_{ik}(t) = \sum_{h=k}^{k+j-1} C_{i(h \mod N_i+1)}. \quad o\breve{u}, \quad j = \min\left\{ j : \sum_{h=k}^{k+j-1} T_{i(h \mod N_i+1)} \geq t \right\} \tag{2.53}$$

Une adaptation de la définition de la propriété d'accumulativement monotonique pour les tâches GMF est comme suit : s'il existe un entier m tel que pour tout $t > 0$, la charge produite par Γ_i lorsque la trame τ_{im} débute l'intervalle t est toujours plus grande que lorsque n'importe quelle trame τ_{ik} $(k \neq m)$ débute l'intervalle t.

$$\forall t > 0, \forall k, 1 \leq k \leq N_i, W_{im}(t) \geq W_{ik}(t) \tag{2.54}$$

Afin d'étudier l'ordonnançabilité d'une tâche GMF Γ_u dans un système contenant des tâches non AM, il est nécessaire de tester l'ordonnançabilité de toutes ses trames τ_{ua} $(1 \leq a \leq N_u)$, pour tous les instants critiques candidats. Un instant critique candidat pour une trame τ_{ua} se produit lorsque il s'active simultanément avec une trame de toute tâche plus prioritaire que Γ_u. Comme pour les transactions et les tâches MF, l'analyse exacte (condition nécessaire et suffisante) de temps de réponse a une complexité exponentielle car on doit examiner toutes les combinaisons possibles entre trames. Une analyse avec une complexité pseudo-polynomiale est possible mais elle fournit une condition suffisante d'ordonnançabilité en calculant un temps de réponse approché pour chaque trame. Cette analyse consiste à étudier l'interférence maximum qu'une tâche GMF peut imposer sur le temps de réponse d'une trame moins prioritaire. La fonction d'interférence (MIF) [105] $M_i(t)$ représente l'interférence maximum de Γ_i dans l'intervalle de temps t, en considérant que chaque trame débute l'intervalle t.

$$M_i(t) = \max_{1 \leq k \leq N_i} W_{ik}(t) \tag{2.55}$$

La borne de temps de réponse d'une trame a d'une tâche GMF analysée Γ_u est donnée par la suite récurrente suivante :

$$\begin{aligned} R_{ua}^{(0)} &= C_{ua} \\ R_{ua}^{(k)} &= C_{ua} + \sum_{i \in hp(\Gamma_u)} M_i(R_{ua}^{(k-1)}) \end{aligned} \tag{2.56}$$

Cette analyse a été améliorée en remplaçant la fonction d'interférence W_{ik}, par une nouvelle fonction W'_{ik} qui détermine l'interférence imposée effectivement par chacune des tâches plus prioritaires. Cette fonction est équivalente à celle d'interférence effective proposée pour les transactions (équation 2.25).

$$W'_{ik}(t) = \sum_{h=k}^{k+j} C_{i(h \mod N_i+1)} + \min\left(C_{i((k+j) \mod N_i+1)}, t - \sum_{h=k}^{k+j-1} P_{i(h \mod N_i+1)} \right) \tag{2.57}$$

Rappelons que cette analyse est faite sous l'hypothèse que les tâches GMF satisfont la propriété FS et que la même priorité est attribuée à toutes les trames d'une même tâche.

2.7.2 Analyse de la demande processeur des tâches GMF à priorités dynamiques EDF

Considérons que toutes les tâches GMF satisfont la propriété $L-MAD$ et sont ordonnancées par l'algorithme à priorités dynamiques EDF. Un test pour analyser la demande processeur consiste à vérifier pour tout intervalle de temps que la charge processeur produite est toujours inférieure ou égale à la longueur de l'intervalle. Ce test est limité par la plus longue période d'activité [16] et plus précisément par les dates d'échéances survenant dans la plus longue période d'activité.

Soit $dbf(\Gamma_i, t)$ (demand-bound function) la fonction déterminant le temps processeur maximum requis par les instances de la tâche Γ_i activées dans l'intervalle de temps $[0, t)$ et dont les échéances sont elles aussi dans cet intervalle (en considérant chaque trame comme initiant l'instant critique). [12] donne un algorithme « Build'list » calculant la demande processeur maximum durant la première période de la tâche (somme des temps de séparation des N_i trames) de chaque tâche GMF Γ_i du système Γ. Cet algorithme enregistre la demande processeur sous forme de couples (date d'échéance, demande processeur). La liste de couples sera utilisée facilement pour trouver la demande processeur maximum dans n'importe quel intervalle de temps t (algorithme Compute-dbf(t) [12]). Cette technique est similaire à celle de la représentation statique de l'interférence des transactions présentée précédemment dans ce chapitre.

Exploitant la périodicité de la fonction de la demande maximum $dbf(\Gamma_i, t)$, une transformation de système à un ensemble de tâches sporadiques Γ' modélisant cette fonction est possible. Γ est ordonnançable dans une architecture monoprocesseur si, et seulement si, le système Γ' des tâches sporadiques correspondant est ordonnançable (théorème 2 de [12]).

Finalement, cette technique réduit alors le problème d'ordonnançabilité des tâches GMF avec priorités dynamiques EDF en problème d'ordonnancement de tâches sporadiques [15, 10]. Ces résultats peuvent être etendus pour des tâches GMF non $L-MAD$ par une simple adaptation en cherchant le motif de périodicité de la fonction $dbf(t)$.

2.8 Conclusion et problématique

Dans ce chapitre, nous avons présenté deux modèles de tâches : le modèle de tâches à offsets et le modèle de tâches multiframes (ainsi que les tâches multiframes généralisées). Ces deux modèles ont été proposées en lieu et place du modèle de tâches indépendantes de L&L [55] afin d'éviter le pessimisme de l'analyse d'ordonnançabilité correspondante lorsque certains types d'applications temps réel sont considérés. Nous n'avons présenté que les transactions et tâches MF (GMF) non concrètes.

Le modèle de L&L considère que toutes les tâches d'un système sont indépendantes et n'ont aucune contrainte concernant leurs activations. Le modèle de tâches à offset a été proposé afin de prendre en considération les relations de précédence entre les tâches. Toutes les tâches ayant une date d'activation liée au même évènement, et avec une même période sont regroupées dans une transaction. Ce modèle a été présenté dans la section 2.2. De nouvelles conditions d'ordon-

nançabilités ont été définies pour ce modèle. Le pire temps de réponse d'une tâche se produit dans une période d'activité initiée par l'activation simultanée d'une tâche de chaque transaction (une tâche plus prioritaire dans le contexte à priorités fixes). L'analyse exacte de temps de réponse (sections 2.3.4 et 2.4) a une complexité exponentielle vu le nombre de périodes d'activités devant être examinée pour trouver le pire cas. Une analyse approchée avec une complexité pseudo-polynomiale est proposée pour les tâches à priorités fixes (sections 2.3.7, 2.3.8) comme à priorités dynamiques (section 2.4.2), mais ces méthodes approximatives fournissent une borne pessimiste de pire temps de réponse.

Le modèle de tâches multiframes et modèle de tâches multiframes généralisées (sections 2.6 et 2.7) ont été proposée afin de permettre à une tâche d'avoir une liste de durées d'exécution possibles contrairement au modèle de L&L [55] qui considère seulement une pire durée d'exécution pour chaque tâche. Chaque tâche est composée d'un ensemble de trames de durées d'exécutions différentes. L'analyse d'ordonnançabilité d'une tâche nécessite d'analyser celle de toutes ses trames. Dans le contexte des priorités fixes, pour ce type de tâches, le pire temps de réponse d'une trame se produit lorsqu'elle s'active simultanément avec une trame de chacune des tâches de priorité supérieure. Comme en tâches à offset, l'analyse exacte nécessite une étude de tous les scénarii possibles en combinant toutes les trames de toutes les tâches. Dans cette situation l'analyse exacte est handicapée par la complexité exponentielle vu le nombre de combinaisons devant être considérées. Une analyse approchée (sections 2.6.2.2, 2.7.1) avec une complexité pseudo-polynomiale est possible mais avec un taux de pessimisme important sur la qualité de tests par rapport à l'analyse exacte. Dans le contexte de priorités dynamiques, un test d'ordonnançabilité par analyse de la demande processeur est présenté dans la section 2.7.2. Ce test consiste à réduire le problème d'ordonnancabilité des tâches GMF en problème d'ordonnancement de tâches sporadiques.

En résumé, dans les cas d'utilisation de la technique d'analyse de temps de réponse pour valider un système, il n'y a que l'analyse avec une complexité exponentielle qui fournit une décision exacte dans les deux contextes d'ordonnancement, à priorités fixes et à priorités dynamiques ; tandis que l'analyse avec une complexité pseudo-polynomiale est nécessairement pessimiste. Plusieurs problèmes sont alors à considérer dans l'analyse de ce type de modèles de tâches :

- *Problème de qualité du test :* Ce problème est lié à la caractérisation du pire scénario d'exécution pour une tâche données. C'est-à-dire parmi toutes les périodes d'activités (instants critiques) possibles, identifier laquelle provoque le pire temps de réponse pour la tâche analysée.
- *Problème de complexité et temps d'exécution :* Les algorithmes identifiant le pire scénario et le temps de réponse doivent avoir une complexité acceptable pour qu'ils puissent être applicables pour valider des systèmes de taille réaliste.
- *Problème d'optimisation :* Dans le cas où il n'existe pas une méthode d'analyse exacte avec une complexité acceptable le problème devient alors un problème d'optimisation. Les principaux critères d'optimisation considérés sont : amélioration de la qualité de décision de l'analyse approchée (réduction de pessimisme) et la minimisation du temps de traitement (complexité) nécessaire pour exécuter les algorithmes d'analyse.

CONTRIBUTIONS

Chapitre 3

Analyse mixte du pire temps de réponse des tâches à offsets en priorités fixes

Résumé : *Ce chapitre présente une méthode mixte d'analyse de temps de réponse, elle combine l'analyse exacte et l'analyse approchée dans un but de minimiser le pessimisme de la méthode approchée de Nolin tout en conservant une complexité pseudo-polynomiale. Nous montrons que cette méthode fournit un temps de réponse meilleur que celui fourni par les méthodes d'analyse approchée. Elle utilise comme paramètre le nombre de transactions pour lesquelles une analyse exacte d'interférence est appliquée. Plus ce paramètre est grand, plus la qualité de la solution est meilleure. Une évaluation de performances montre les limites d'application de la méthode en fonction du nombre de transactions choisi par rapport à une analyse exacte.*

3.1 Comparaison des différents modèles de tâches

Nous avons vu, dans le chapitre 2, que le modèle de tâches indépendantes proposé par Liu et Layland [55] ne s'adapte pas à tous les types d'applications. Ainsi, pour des motivations différentes, plusieurs modèles de tâches ont été proposés dans la littérature afin de réduire ce pessimisme : principalement les modèles multi-frames et transactions présentés auparavant. Dans la suite, nous essayons de comparer les deux modèles les plus connus : le modèle de tâches à offsets (transactions) et le modèle de tâches Multiframes (ainsi les tâches GMF), dans le but de trouver un rapprochement des résultats d'analyse d'ordonnançabilité.

Les tâches à offsets ont été proposées afin de prendre en compte les relations de décalage (précédence) d'activation entre les tâches, tandis que les tâches MF (ou GMF) ont été proposées afin de prendre en compte des différentes durées d'exécution pour une tâche, suivant un motif connu a priori. Les dates d'activations des trames successives d'une même tâche MF ou GMF sont séparées par un intervalle de temps appelé période d'activation. Cette loi d'activation des trames dans la tâche peut être considérée comme la relation d'offset (ou de précédence) définie pour les tâches d'une transaction. Le même motif d'exécution des trames se répète après un intervalle de temps (hyper période) correspondant à la somme des périodes de toutes les trames de la tâche. Cet intervalle peut être considéré alors comme la période d'une transaction. Par conséquent, une tâche multiframe peut être modélisée comme une transaction avec des gigues nulles. Soit Γ_i une tâche GMF composée de N_i trames $(\tau_{i1}, \tau_{i2}, ..., \tau_{iN_i})$, chaque trame τ_{ij} $((1 \leq j \leq N_i)$ a une durée d'exécution C_{ij}, un délai critique D_{ij} et une période T_{ij}. La tâche Γ_i est modélisée comme une transaction $\Gamma_i :< \tau_{i1}, \tau_{i2}, ..., \tau_{iN_i}, T_i >$ de N_i tâches avec les caractéristiques suivantes :

- Chaque trame de la tâche GMF est transformée en tâche à offset en gardant la même durée d'exécution et délai critique $\tau_{ij} :< C_{ij}, O_{ij}, D_{ij}, 0, 0, P_i >$ $((1 \leq j \leq N_i)$.
- La première tâche de la transaction τ_{i1} à un offset nul $O_{i1} = 0$.
- L'offset d'une tâche τ_{ij} $(2 \leq j \leq N_i)$ est donné par : $O_{ij} = \sum_{k=1}^{j-1} T_{ij}$.
- Les gigues des tâches sont nulles $J_{ij} = 0$ $(1 \leq j \leq N_i)$
- La période de la transaction est : $T_i = \sum_{j=1}^{N_i} T_{ij}$.

La figure 3.1 présente graphiquement un exemple de transformation d'une tâche GMF en une transaction.

Concernant l'analyse d'ordonnançabilité, les deux modèles présentant les mêmes conditions d'ordonnançabilité. Par exemple, dans le contexte d'ordonnacement à priorités fixes, le pire temps de réponse d'une tâche (resp. une trame) survient lorsque une tâche plus prioritaire de chaque transaction (resp. chaque trame de chaque tâche GMF plus prioritaire) s'activent simultanément. Par conséquent, les deux modèles présentant un problème de complexité exponentielle pour l'analyse exacte de pire temps de réponse car cela nécessite l'examen de toutes les combinaisons possibles entre les tâches de toutes les transactions (resp. toutes les combinaisons entre les trames de toutes les tâches GMF). De plus, le même principe d'approximation de calcul de pire temps de réponse a été proposé indépendamment pour les deux modèles. Les fonctions d'interférences sont les même, avec une simplification des formule pour les tâches MF (GMF) due à l'absence de gigue et aux restrictions d'exécution des MF (GMF).

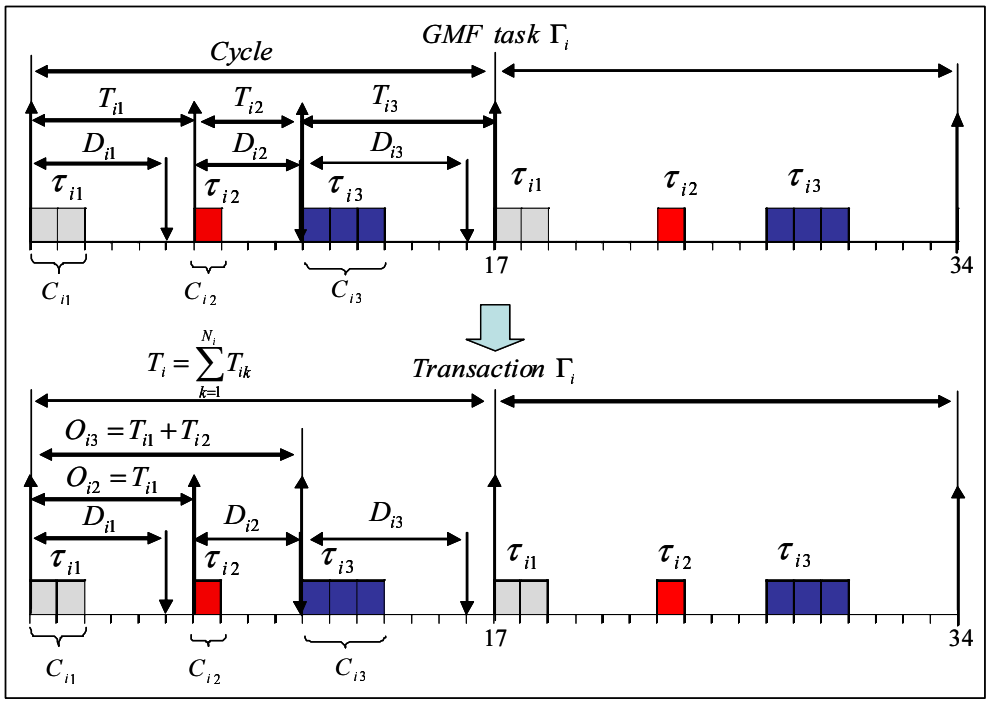

FIG. 3.1 – Transformation d'une tâche GMF en transaction.

En conclusion le modèle des transactions inclut donc le modèle GMF, qui inclut le modèle MF. Le modèle de tâches MF est un cas particulier de modèle de tâches GMF. Ces deux modèles sont des cas particuliers de modèle de transactions. Ainsi, n'importe quelle méthode d'analyse d'ordonnançabilité sur les transactions est applicable sur les tâches MF et GMF avec une simple adaptation (simplification) des formules en éliminant l'effet de la gigue et les contraintes propres à ce type de tâches. Afin que notre étude sur l'analyse d'ordonnançabilité couvre tous les modèles présentés précédemment, nous nous focalisons alors sur le modèle de tâches à offsets qui est le modèle le plus général. L'étude d'ordonnançabilité effectuée dans cette thèse considère, comme support d'exécution de tâches, une architecture monoprocesseur.
Dans ce chapitre, nous expliquons pourquoi l'analyse de Turja-Nolin est pessimiste, puis nous proposons une méthode d'analyse mixte de temps de réponse, avec un temps de calcul pseudo-polynomial. Cette méthode est moins pessimiste que la meilleure méthode d'analyse approchée de Turja-Nolin [62, 63]. L'analyse mixte est paramétrée par la quantité exacte du calcul d'interférence permise dans l'analyse d'ordonnançabilité d'une tâche, la complexité de l'analyse et la qualité de la réponse évoluent en fonction de ce paramètre.

3.2 Pessimisme de l'analyse approchée de Turja-Nolin

Malgré l'élimination de la surestimation causée par la fonction d'interférence maximum (grâce à l'utilisation de la fonction d'interférence effective), la méthode d'analyse proposée par Turja et Nolin reste pessimiste. Afin de réduire encore plus le pessimisme de l'analyse approchée, nous essayons d'identifier, dans cette section, à travers un exemple, la source du pessimisme. Nous

montrons que l'application de la fonction d'interférence approchée sur certaines transactions provoque le pessimisme, alors qu'un calcul exacte d'interférence pour ces quelques transactions donne des résultats moins pessimistes de ceux donnés par la méthode approchée de Turja et Nolin. Autrement dit, la combinaison du calcul exact et du calcul approché peut former une nouvelle méthode moins pessimiste que les méthodes approchées existantes avec une complexité acceptable, et des temps de calcul comparables.

3.2.1 Exemple

Nous appliquons la méthode approchée de Turja-Nolin pour analyser le pire temps de réponse de la tâche τ_{ua} pour l'exemple présenté sur la figure 2.6. Les courbes illustrant l'évolution d'interférence effective de chacune des deux transactions Γ_1 et Γ_2 pour chaque tâche candidate sont représentées dans la figure 3.2 (reportée sur la figure 2.9 pour plus de lisibilité). Nous calculons la borne du pire temps de réponse de la tâche τ_{ua} en exploitant les points convexes des courbes de la fonction approximative d'interférence ($W_i(t)$) de Γ_1 et Γ_2 (par conséquent les étapes de calcul itératif sont différentes du calcul présenté dans la section 2.3.8.1) :

$$\begin{aligned}
\mathrm{R}_{ua}^{(0)} &= C_{ua} = 1 \\
\mathrm{R}_{ua}^{(1)} &= C_{ua} + W_1(1) + W_2(1) = 1 + 3 + 2 = 6 \\
\mathrm{R}_{ua}^{(2)} &= C_{ua} + W_1(6) + W_2(6) = 1 + 4 + 2 = 7 \\
\mathrm{R}_{ua}^{(3)} &= C_{ua} + W_1(7) + W_2(7) = 1 + 4 + 3 = 8 \\
\mathrm{R}_{ua}^{(4)} &= C_{ua} + W_1(8) + W_2(8) = 1 + 6 + 3 = 10 \\
\mathrm{R}_{ua}^{(5)} &= C_{ua} + W_1(10) + W_2(10) = 1 + 6 + 3 = 10
\end{aligned}$$

La borne du pire temps de réponse de la tâche τ_{ua} égale à $R_{ua} = 10$, tandis que la valeur exacte du pire temps de réponse est égale à 8.

3.2.2 Discussion

Le pessimisme de l'approche de Turja-Nolin est produit par la fonction d'approximation $W_i(t)$ qui prend le maximum des interférences de la transaction Γ_i en considérant chacune de ses tâches comme celle qui initie l'instant critique. Cette fonction d'approximation donne une valeur exacte de la borne d'interférence effective, mais elle a un effet négatif, sur la qualité de temps de réponse, dû au changement de la tâche candidate initiant l'instant critique d'une itération à une autre au cours du calcul itératif de temps de réponse. Pour une tâche sous analyse donnée, le pessimisme peut se produire par l'application de la fonction W_i seulement sur une ou plusieurs transactions.

La table suivante montre le calcul détaillé du temps de réponse approché de la tâche τ_{ua} (avec WCET $C_{ua} = 1$) de l'exemple précédent. A chaque itération l'interférence de chacune des deux transactions est calculée pour chaque tâche candidate initiant l'instant critique. Pour la transaction Γ_2 l'instant critique est toujours initié par la tâche τ_{11} durant tout le processus de recherche

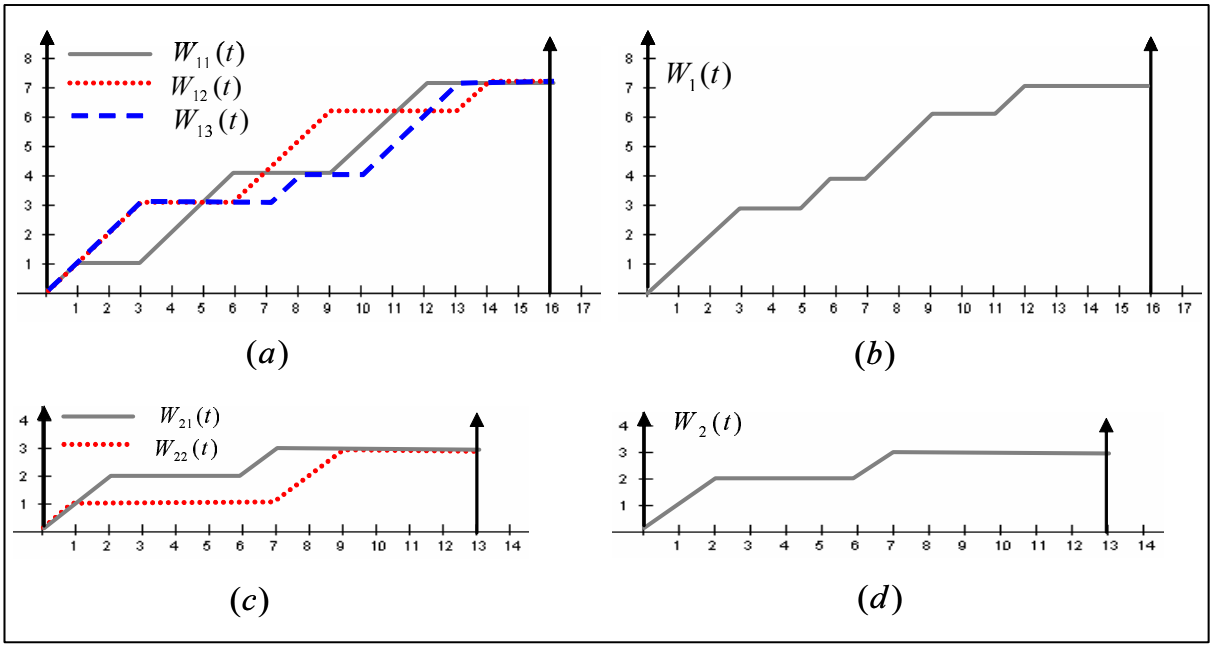

FIG. 3.2 – Courbes d'interférence effective.

du point fixe, tandis que la transaction Γ_1 a changé plusieurs fois la tâche initiant l'instant critique. Le temps de réponse obtenu est supérieur à la valeur exacte de pire temps de réponse, c'est à dire que du pessimisme a été produit sur le temps de réponse par l'application de la fonction d'interférence approchée sur la transaction Γ_1 (la figure 3.2(a) montre qu'à l'instant $t = 3$ la tâche candidate τ_{12} crée le pire cas, par contre à l'instant $t = 6$ c'est la tâche candidate τ_{11} qui crée le pire cas). En fait, on a atteint une pire durée de période d'activité égale à 8 en considérant τ_{11} comme candidate, ce qui nous a fait considérer τ_{12} comme candidate, mais si τ_{12} était candidate, la durée de la période d'activité n'aurait pas dépasser 6. Par conséquent, on atteint un cas irréaliste dû à l'approximation W_1.

Pour certaines tâches à analyser, la méthode approchée donne une borne exacte du temps de réponse, malgré l'existence d'un changement de tâche initiant l'instant critique, par exemple pour une tâche τ_{ua} de WCET $C_{ua} = 3$ la borne fournie par la méthode approchée est égale au temps de réponse obtenue par la méthode exacte, autrement pour cette tâche $C_{ua} = 3$ aucun pessimisme n'est apporté par la fonction d'approximation. Ainsi la production du pessimisme dépend des instants auxquels le changement de tâches initiant l'instant critique est produit.

Nous remarquons que la tâche τ_{ua} (avec un WCET $C_{ua} = 1$), dans le pire cas, termine son exécution à la date à $t = 8$ (pire temps de réponse exact). Cette date correspond à la date où l'interférence de la transaction Γ_1 lorsque l'instant critique est initié par la tâche τ_{12} rattrape et dépasse son interférence lorsque la tâche τ_{11} initie l'instant critique i.e. à l'instant $t = 7$ (voir figure 3.2) il y a un changement de la tâche initiant l'instant critique. Graphiquement ce chan-

t	W_{11}	W_{12}	W_{13}	W_{21}	W_{22}	W_1	W_2	R_{ua}
								1
1	1	1	1	**1**	1	1	1	3
3	1	**3**	3	**2**	1	3	2	6
6	**4**	3	3	**2**	1	4	2	7
7	**4**	4	3	**3**	1	4	3	8
8	4	**5**	4	**3**	2	5	3	9
9	4	**6**	4	**3**	3	6	3	10
10	4	**6**	4	**3**	3	6	3	10

TAB. 3.1 – Calcul approché du temps de réponse

gement correspond au point d'intersection des courbes d'interférences de la transaction Γ_1. Ce changement de tâche initiant l'instant critique à cette date précise a produit le pessimisme sur le temps de réponse de τ_{ua}.

Afin d'éviter ce pessimisme produit par l'application de la fonction d'interférence approchée $W_i(t)$ sur une transaction Γ_i qui est considérée comme une source du pessimisme, nous calculons le pire temps de réponse pour tous les instants critiques possibles correspondant à ses tâches candidates (la fonction d'interférence approchée est appliquée pour toutes les transactions exceptée Γ_i) et ensuite nous choisissons le maximum des temps de réponses obtenus.

Dans notre exemple, le pessimisme est bien produit par la fonction d'approximation appliquée sur la transaction Γ_1, une solution de ce problème est de ne pas appliquer cette fonction sur la transaction Γ_1, ce qui nous conduit à calculer le temps de réponse pour chaque instant critique initié par une tâche candidate de Γ_1. Nous calculons le temps de réponse de τ_{ua} correspondant à chaque tâche candidate ensuite nous prenons le maximum. Signalons que la fonction d'interférence approchée est appliquée sur Γ_2

La tâche candidate τ_{11} :

$$\begin{aligned}
\mathrm{R}^{(0)}_{ua11} &= C_{ua} = 1 \\
\mathrm{R}^{(1)}_{ua11} &= C_{ua} + W_{11}(1) + W_2(1) = 1 + 1 + 2 = 4 \\
\mathrm{R}^{(2)}_{ua11} &= C_{ua} + W_{11}(4) + W_2(4) = 1 + 4 + 2 = 7 \\
\mathrm{R}^{(3)}_{ua11} &= C_{ua} + W_{11}(7) + W_2(7) = 1 + 4 + 3 = 8 \\
\mathrm{R}^{(4)}_{ua11} &= C_{ua} + W_{11}(8) + W_2(8) = 1 + 4 + 3 = 8
\end{aligned}$$

La tâche candidate τ_{12} :

$$\begin{aligned}
\mathrm{R}^{(0)}_{ua12} &= C_{ua} = 1 \\
\mathrm{R}^{(1)}_{ua12} &= C_{ua} + W_{12}(1) + W_2(1) = 1 + 3 + 2 = 6 \\
\mathrm{R}^{(2)}_{ua12} &= C_{ua} + W_{12}(6) + W_2(6) = 1 + 3 + 2 = 6
\end{aligned}$$

La tâche candidate τ_{13} :

$$\begin{aligned} \mathrm{R}^{(0)}_{ua13} &= C_{ua} = 1 \\ \mathrm{R}^{(1)}_{ua13} &= C_{ua} + W_{13}(1) + W_2(1) = 1 + 3 + 2 = 6 \\ \mathrm{R}^{(2)}_{ua13} &= C_{ua} + W_{13}(6) + W_2(6) = 1 + 3 + 2 = 6 \end{aligned}$$

Le maximum des temps de réponse obtenus est égal au temps de réponse exact $R_{ua} = 8$.

En se basant sur ce constat, nous proposons dans la section suivante une nouvelle méthode d'analyse de temps de réponse qui fournit une borne de pire temps de réponse sûre et meilleure que celle fournie par les méthodes approchées présentées dans le chapitre 2.

3.3 Méthode d'analyse mixte

Notre nouvelle méthode d'évaluation de pire temps de réponse améliore la borne supérieure du temps de réponse fournie par l'analyse approchée de Turja-Nolin [61, 62, 63] présentée dans le chapitre précédent. L'idée mise en œuvre dans cette nouvelle approche est de combiner le principe du calcul exact et le principe d'approximation d'interférence afin de diminuer le pessimisme de l'analyse pire cas, permettant ainsi d'améliorer la borne supérieure du temps de réponse fournie tout en conservant une complexité pseudo-polynomiale.

L'utilisation de la fonction d'interférence approchée (équation 2.14) génère un pessimisme sur le temps de réponse de la tâches sous analyse. Le pessimisme est produit à cause du changement des tâches candidates initiant l'instant critique en fonction de la longueur de la période d'activité (d'une itération à une autre). Les transactions candidates qui sont source de pessimisme pour le temps de réponse d'une tâche τ_{ua} sont celles qui peuvent avoir un changement d'instant critique à la date qui représente le pire temps de réponse exacte de la tâche τ_{ua}. Dans l'exemple précédent à la date $t = 8$ (temps de réponse exact) la transaction Γ_1 change la tâche initiant l'instant critique (de τ_{11} à τ_{12}) ce qui introduit une surestimation dans l'interférence.

Soit Γ_i une transaction supposée comme une source de pessimisme (un scénario) de l'analyse approchée de temps de réponse d'une tâche analysée τ_{ua}. Afin d'éviter ce pessimisme, le calcul approché de son interférence (W_i) est remplacé par un calcul exact d'interférence, pour cela plusieurs instants critiques correspondant à ces tâches candidates doivent être examinés. C'est-à-dire que chaque tâche candidate de Γ_i est traitée séparément. Nous avons alors $|hp_i(\tau_{ua})|$ instants critiques à considérer, nous calculons le temps de réponse $R_{ua,ic}$ de τ_{ua} lorsque la tâche candidate τ_{ic} initie l'instant critique. Et puis nous choisissons le maximum des $R_{ua,ic}$ obtenus comme une borne du pire temps de réponse.

Le temps de réponse de τ_{ua}, correspondant à la tâche candidate τ_{ic} de Γ_i, est obtenu par la formule suivante :

$$R_{ua,ic}^{(k)} = C_{ua} + W_{ic}(R_{ua,ic}^{(k-1)}) + \sum_{k\neq i,k\neq u}(W_k(R_{ua,ic}^{(k-1)})) \tag{3.1}$$

$$W_k(t) = \max_{c\in hp_k(\tau_{ua})} W_{kc}(t)$$

Notons que $W_k(t)$ est l'approximation Turja-Nolin des interférences effectives de la transaction Γ_k. Les $W_{kc}(t)$ utilisées pour trouver l'interférence maximum d'une transaction Γ_k sont calculées par la fonction d'interférence effective de Turja-Nolin (équation 2.25 de chapitre 2 afin d'éviter le pessimisme induit par l'approximation d'interférence). Tandis que pour l'interférence exacte $W_{ic}(t)$ de la transaction Γ_i (choisie pour un analyse exacte d'interférence) lorsqu'une tâche candidate τ_{ic} initie la période d'activité, les deux formules d'interférence classique (équation 2.6) et effective (équation 2.25) aboutissent au même résultat.

Par conséquent, dans un scénario correspondant à une transaction Γ_i, $|hp_i(\tau_{ua})|$ instants critiques (tâches candidates) sont possibles. Nous déterminons le temps de réponse $R_{ua,ic}$ de τ_{ua} correspondant à chaque tâche candidate τ_{ic}. $R_{ua,i}$ est le maximum des temps de réponses obtenus dans tous les instants critiques est le pire temps de réponse de τ_{ua} correspondant à la transaction Γ_i.

$$R_{ua,i} = \max_{c\in hp_i(\tau_{ua})} R_{ua,ic} \tag{3.2}$$

$R_{ua,ic}$ est le temps de réponse de la tâche τ_{ua}, lorsque la tâche τ_{ic}, de la transaction Γ_i, initie l'instant critique. i.e. durant le calcul du temps de réponse une interférence exacte est utilisée de Γ_i (celle lorsque τ_{ic} initie l'instant critique) tandis qu'une interférence approchée (maximum) est utilisée pour les autres transactions.

Théorème 24 *Le temps de réponse $R_{ua,i}$ obtenu dans un scénario (correspond à une transaction Γ_i) est toujours compris entre le temps de réponse exact et la borne de temps réponse calculée par la méthode approximative de Turja et Nolin.*

$$R_{ua}^{Exact} \leq R_{ua}^{Mixte(i)} \leq R_{ua}^{Nolin} \tag{3.3}$$

Preuve

Afin de faciliter la compréhension de la preuve, nous notons $R_{ua}(f(t))$ le temps de réponse de la tâche τ_{ua}, calculé par la recherche du premier point fixe (calcul itératif), en utilisant la fonction d'interférence du système $f(t)$. Nous supposons que toutes les tâches de toutes les transactions sont plus prioritaires que la tâche analysée τ_{ua}. En nous basant sur cette notation, les temps de réponse d'une tâche τ_{ua}, obtenus par les différentes méthode sont notés comme suit :

– Le temps de réponse obtenue par la méthode approchée de Nolin

$$\begin{aligned} R_{ua}^{Nolin} &= R_{ua}\left(f^{Nolin}(t)\right) \\ &= R_{ua}\left(\max_{\forall c_1\in\Gamma_1} W_{1c_1}(t) + ... + \max_{\forall c_k\in\Gamma_k} W_{kc_k}(t) + ... + \max_{\forall c_n\in\Gamma_n} W_{nc_n}(t)\right) \end{aligned}$$

– Le temps de réponse obtenue par la méthode exacte

$$\begin{aligned} R_{ua}^{exact} &= \max_{\forall c_1 \in \Gamma_1, ..., c_k \in \Gamma_k, ..., c_n \in \Gamma_n} R_{ua}\left(f^{Exacte}(t)\right) \\ &= \max_{\forall c_1 \in \Gamma_1, ..., c_k \in \Gamma_k, ..., c_n \in \Gamma_n} R_{ua}\left(W_{1c_1}(t) + ... + W_{kc_k}(t) + ... + W_{nc_n}(t)\right) \end{aligned}$$

– Le temps de réponse obtenu par la méthode mixte, en choisissant une transaction Γ_e pour une analyse exacte d'interférence

$$R_{ua}^{Mixte(e)} = \max_{\forall c_e \in \Gamma_e} R_{ua}\left(f^{Mixte(e)}(t)\right)$$

$$= \max_{\forall c_e \in \Gamma_e} R_{ua}\left(\max_{\forall c_1 \in \Gamma_1} W_{1c_1}(t) + ... + W_{ec_e}(t) + ... + \max_{\forall c_k \in \Gamma_k} W_{kc_k}(t) + ... + \max_{\forall c_n \in \Gamma_n} W_{nc_n}(t)\right)$$

Notons que les fonctions d'interférences définies ci-dessus sont monotones. Pour un calcul itératif du premier point fixe, basé sur deux fonctions monotones différentes $f(t)$ et $g(t)$, nous avons la propriété suivante :

$$Si \;\; \forall t : \;\; f(t) \leq g(t) \;\; alors \;\; R_{ua}(f(t)) \leq R_{ua}(g(t)) \tag{3.4}$$

Afin de prouver les deux inégalités du théorème, il suffit alors de comparer les fonctions d'interférences du système utilisées dans les équations de calcul 3.4, 3.4 et 3.4. La preuve se fait en deux parties :

$\underline{R_{ua}^{Mixte(e)} \leq R_{ua}^{Nolin}}$ **:**
Pour cela nous devons prouver que :

$$\forall t : \;\; \max_{\forall c_e \in \Gamma_e} f^{Mixte(e)}(t) \;\; \leq \;\; f^{Nolin}(t) \tag{3.5}$$

Notons $R_{ua}^{Mixte(ec_e)}$ le temps de réponse de τ_{ua} en utilisant la méthode mixte avec une interférence exacte de Γ_e lorsque τ_{ec_e} initie l'instant critique ;

$$R_{ua}^{Mixte(ec_e)} \;\; = \;\; R_{ua}\left(\max_{\forall c_1 \in \Gamma_1} W_{1c_1}(t) + ... + W_{ec_e}(t) + ... + \max_{\forall c_k \in \Gamma_k} W_{kc_k}(t) + ... + \max_{\forall c_n \in \Gamma_n} W_{nc_n}(t)\right) \tag{3.6}$$

Donc $R_{ua}^{Mixte(e)}$ peut être récrit comme suit

$$R_{ua}^{Mixte(e)} = \max_{\forall c_e \in \Gamma_e} R_{ua}^{Mixte(ec_e)} \tag{3.7}$$

Rappelons que l'interférence d'une transaction est complètement indépendante des autres transactions du système. Pour importe quelle tâche candidate τ_{ec_e} dans une transaction Γ_e nous avons l'inégalité suivante :

$$\forall c_e \in \Gamma_e : \quad \max_{\forall c_1 \in \Gamma_1} W_{1c_1}(t) + ... + W_{ec_e}(t) + ... + \max_{\forall c_k \in \Gamma_k} W_{kc_k}(t) + ... + \max_{\forall c_n \in \Gamma_n} W_{nc_n}(t) \leq \max_{\forall c_1 \in \Gamma_1} W_{1c_1}(t) + ... + \max_{\forall c_e \in \Gamma_e} W_{ec_e}(t) + ... + \max_{\forall c_k \in \Gamma_k} W_{kc_k}(t) + ... + \max_{\forall c_n \in \Gamma_n} W_{nc_n}(t) \quad (3.8)$$

D'où :

$$\forall c_e \in \Gamma_e : \; R_{ua}(f^{Mixte(ec_e)}(t)) \leq R_{ua}(f^{Nolin}(t)) \quad \Rightarrow$$

$$\max_{\forall c_e \in \Gamma_e} R_{ua}(f^{Mixte(ec_e)}(t)) \leq R_{ua}(f^{Nolin}(t))$$

D'où la première inégalité du théorème est vérifiée (d'après l'équation 3.7)

$$R_{ua}^{Mixte(e)} \leq R_{ua}^{Nolin}$$

$\underline{R_{ua}^{Exact} \leq R_{ua}^{Mixte(e)}$:

Notons $R_{ua}^{Exact(ec_e)}$ le temps de réponse maximum lorsque τ_{ec_e} initie l'instant critique dans Γ_e, pour toute tâche candidate de $\Gamma_i \neq \Gamma_e$.

$$R_{ua}^{Exact(ec_e)} = \max_{\forall c_i \in \Gamma_i \; (i \neq e)} \left(R_{ua}\left(W_{1c_1}(t) + ... + W_{ec_e}(t) + ... + W_{kc_k}(t) + ... + W_{nc_n}(t)\right)\right) \quad (3.9)$$

En utilisant la notation $R_{ua}^{Exact(ec_e)}$ (équation 3.9), on peut écrire le temps de réponse exact de la façon suivante :

$$R_{ua}^{Exact} = \max_{\forall c_e \in \Gamma_e} R_{ua}^{Exact(ec_e)} \quad (3.10)$$

Pour une tâche candidate τ_{ec_e}, d'une transaction Γ_e nous avons l'inégalité suivante :

$$\forall c_i \in \Gamma_i \; (i \neq e) : \quad \max_{\forall c_1 \in \Gamma_1} W_{1c_1}(t) + ... + W_{ec_e}(t) + ... + \max_{\forall c_k \in \Gamma_k} W_{kc_k}(t) + ... + \max_{\forall c_n \in \Gamma_n} W_{nc_n}(t) \geq W_{1c_1}(t) + ... + W_{ec_e}(t) + ... + W_{kc_k}(t) + ... + W_{nc_n}(t)$$

Par conséquent (propriété de l'équation 3.4) :

$$R_{ua}^{Mixte(ec_e)} \geq \; R_{ua}^{Mixte(ec_e)}$$

D'où pour toutes les tâches candidates de Γ_e on obtient l'inégalité suivante :

$$\forall c_{ec_e} \in \Gamma_e : R_{ua}^{Mixte(ec_e)} \geq \; R_{ua}^{Mixte(ec_e)} \Rightarrow$$

$$\max_{\forall c_{ec_e} \in \Gamma_e} R_{ua}^{Mixte(ec_e)} \geq \max_{\forall c_{ec_e} \in \Gamma_e} R_{ua}^{Mixte(ec_e)}$$

Par définition du temps de réponse exact (équation 3.10) et du temps de réponse obtenu par la méthode mixte (équation 3.7) on obtient

$$R_{ua}^{Exact} \leq R_{ua}^{Mixte(e)}$$

Donc la deuxième inégalité du théorème est vérifiée.
□

Afin de montrer les étapes de l'analyse complète de temps de réponse, dans le cas général, nous supprimons la supposition (chapitre 2) que la tâche analysée est l'unique tâche de Γ_u et qu'une seule instance de τ_{ua} est activée dans la période d'activité. Rappelons que cette supposition a été proposée (dans chapitre 2) juste pour simplifier les formules présentées. Nous devons en fait prendre en compte, pour chaque tâche candidate τ_{uc}, l'interférence de toutes les tâches plus prioritaires que τ_{ua} dans Γ_u, dont les instances de τ_{ua}. De plus, il est nécessaire d'analyser le temps de réponse de toutes les instances de τ_{ua} activées dans la période d'activité considérée. Soit une tâche τ_{uc} de Γ_u initiant l'instant critique et une transaction Γ_i du système choisie pour une analyse exacte d'interférence. Afin de calculer le pire temps de réponse $R_{ua,ic}$ de τ_{ua} lorsque la tâche candidate τ_{ic} initie l'instant critique nous devons déterminer tous les instances de τ_{ua} activées dans la période d'activité. Les instances de τ_{ua} sont numérotées de la même façon que celle présentée dans le chapitre 2. $p_{0,ua}$ est la position la plus faible des instances de τ_{ua} présentes dans la période d'activité.

$$p_{0,ua} = -\left\lfloor \frac{J_{ua} + \Phi_{uac}}{T_u} \right\rfloor + 1 \tag{3.11}$$

Afin de déterminer la position $p_{L,ua}$ (liée à la longueur de la période d'activité), la plus élevée de ces instances, il est indespensable de trouver la longueur de la période d'activité. La longueur est calculée itérativement par la suite récurrente suivante.

$$\begin{aligned} L_{ua}^{(k)} &= B_{ua} + \left(\left\lceil \frac{L_{ua}^{(k-1)} - \Phi_{uac}}{T_u} \right\rceil - p_{0,ua} + 1 \right) C_{ua} + W_{uc}\left(L_{ua,ic}^{(k-1)}\right) + W_{ic}\left(L_{ua}^{(k-1)}\right) \\ &\quad + \sum_{\forall k \neq i,u} W_{kc_k}\left(L_{ua}^{(k-1)}\right) \end{aligned} \tag{3.12}$$

Une fois déterminée la longueur de la période d'activité, la valeur de la position de la dernière instance de τ_{ua} présente dans la période d'activité est :

$$p_{L,ua} = \left\lfloor \frac{L_{ua} - \Phi_{uac}}{T_u} \right\rfloor \tag{3.13}$$

Il suffit maintenant de calculer le temps de réponse de toutes les instances de τ_{ua} de position p ($p = p_{0,ua}, ..., p_{L,ua}$), puis de choisir le maximum comme le pire temps de réponse pour l'instant critique considéré. La date d'activation $a_{ua}(p)$ d'une instance de position p est donnée par :

$$a_{ua}(p) = (p-1)T_u + \Phi_{uac} \tag{3.14}$$

La date de fin d'exécution de l'instance p correspond à la longueur de la période d'activité associée à l'instance p. La longueur de cette période d'activité est calculée par la recherche itérative du plus petit point fixe.

$$\begin{aligned} L_{ua}^{(k)}(p) &= B_{ua} + (p - p_{0,ua} + 1)\, C_{ua} + W_{uc}\left(L_{ua}^{(k-1)}(p)\right) + W_{ic}\left(L_{ua}^{(k-1)}(p)\right) \\ &\quad + \sum_{\forall k \neq u,i} W_{kc_k}\left(L_{ua}^{(k-1)}(p)\right) \end{aligned} \tag{3.15}$$

Par définition, la différence entre la date de fin d'exécution et celle de son activation donne le temps de réponse $R_{ua}(p)$ de l'instance p. Ce temps de réponse est donné par :

$$R_{ua}(p) = L_{ua}(p) - a_{ua}(p) \tag{3.16}$$

Le pire temps de réponse de τ_i pour l'instant critique initié par la tâche candidate τ_{ic} est :

$$R_{ua,ic} = \max_{p=p_{0,ua}..p_{L,ua}} R_{ua}(p) \tag{3.17}$$

3.3.1 Analyse complète

L'application d'un calcul exact sur une transaction quelconque dans un système avec une approximation pour le reste des transaction peut améliorer la qualité de la borne de pire temps de réponse. Dans un système de transactions donné, la ou les transactions pour lesquelles l'application de la fonction d'interférence approchée produit un pessimisme sur le pire temps de réponse d'une tâche analysée ne sont pas connues. Afin d'avoir le moins de pessimisme possible, nous effectuons un examen pour toutes les transactions une à une. D'où le nombre de scénarii possible est égal au nombre de transactions ($|\Gamma|$) du système.

Le théorème 24 démontre que le temps de réponse obtenu par la méthode mixte pour un scénario (transaction) est une borne supérieure sûre de pire temps de réponse, d'où la meilleure borne de pire cas correspond à la plus petite borne parmi celles obtenues dans tous les scénarii.

$$R_{ua} = \min_{i \in 1..|\Gamma|, i \neq u} R_{ua,i} \tag{3.18}$$

D'après le théorème 24, la borne du pire temps de réponse R_{ua} obtenue par l'équation 3.18 est une borne sûre de pire temps de réponse. Le calcul de cette borne supérieure de temps réponse par la méthode mixte, nécessite alors un traitement de $|hp_1(\tau_{ua})| + |hp_2(\tau_{ua})| + ... + \left|hp_{|\Gamma|}(\tau_{ua})\right|$ processus de recherche de temps de réponse, chaque processus correspond à une tâche candidate

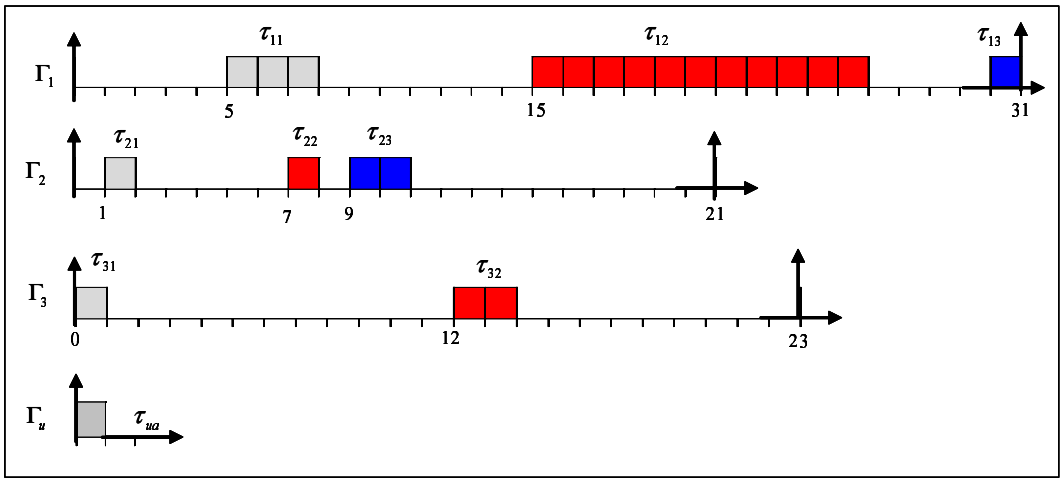

FIG. 3.3 – Système de transactions.

du système. D'où une complexité pseudo-polynomiale.

Pour un système composé de deux transactions, la méthode d'analyse mixte de temps de réponse de toutes les tâches du système est équivalente à la méthode exacte. Car pour une tâche analysée, de l'une des deux transactions, un calcul exact est appliqué sur l'autre transaction.

La méthode d'analyse mixte et la méthode approchée de Turja-Nolin ont toutes les deux une complexité pseudo-polynomiale sauf que la méthode mixte utilise $|\Gamma_1|+|\Gamma_2|+...+\left|\Gamma_{|\Gamma|}\right|$ opérations de calcul de temps de réponse tandis que la méthode de Turja-Nolin ne fait qu'une.

3.3.2 Exemple

Nous appliquons la méthode mixte sur l'exemple de la figure 3.3. La valeur exacte de pire temps de réponse est égale à 20 unités de temps. La borne de temps de réponse obtenue par la méthode approchée de Turja-Nolin est égale à 28. La table 3.2 résume les différentes étapes de l'analyse mixte de la borne de temps de réponse, en choisissant dans chaque scénario une transaction Γ_i ($i = 1..3$) pour laquelle une analyse exacte est appliquée. La borne obtenue pour le pire temps de réponse de τ_{ua} est égale à 20 unités de temps, cette borne est alors avec un pessimisme nul , contrairement à celle fournie par l'analyse Turja-Nolin.

pour la tâche candidate τ_{11} :

$$
\begin{aligned}
\mathrm{R}^{(0)}_{ua,11} &= C_{ua} = 1 \\
\mathrm{R}^{(1)}_{ua,11} &= C_{ua} + W_{11}(1) + W_2(1) + W_3(1) &&= 1+3+2+2 = 8 \\
\mathrm{R}^{(2)}_{ua,11} &= C_{ua} + W_{11}(8) + W_2(8) + W_3(8) &&= 1+12+4+3 = 20 \\
\mathrm{R}^{(3)}_{ua,11} &= C_{ua} + W_{11}(20) + W_2(20) + W_3(20) &&= 1+12+4+3 = 20
\end{aligned}
$$

Γ_i	tâche candidate τ_{ic}	$R_{ua,ic}$	$R_{ua,i}$	R_{ua}
Γ_1	τ_{11}	20	20	20
	τ_{12}	11		
	τ_{13}	9		
Γ_2	τ_{21}	19	23	
	τ_{22}	23		
	τ_{23}	23		
Γ_3	τ_{31}	28	28	
	τ_{32}	27		

TAB. 3.2 – Analyse mixte de temps de réponse.

3.4 Méthode mixte paramétrée

Le pessimisme sur le temps de réponse peut être généré par l'application de la fonction d'interférence approchée sur plusieurs transactions ensemble. Par conséquent, l'application de la méthode mixte pour une seule transaction ne suffira pas pour éliminer tout le pessimisme. En utilisant le principe d'analyse mixte, nous pouvons améliorer la qualité de la borne de temps de réponse fournie, en augmentant le nombre de transactions E pour lesquelles on applique une analyse exacte.

Soit ξ un ensemble de E transactions parmi toutes les transactions d'un système Γ. La formule suivante donne le temps de réponse $R_{ua,\xi}$, d'une tâche τ_{ua} lorsque l'instant critique est initié par une tâche τ_{ic_i} de chaque transaction Γ_i appartenant à ξ. Rappelons que pour les transactions n'appartenant pas à ξ une analyse approchée est appliquée.

$$R_{ua}^{(n)} = C_{ua} + \sum_{i \in \xi} W_{ic_i}(R_{ua,\xi}^{(n-1)}) + \sum_{i \in \Gamma - \xi} (W_i(R_{ua,\xi}^{(n-1)})) \tag{3.19}$$

Une borne de pire temps de réponse est obtenue par l'examen de tous les instants critiques possibles en combinant les tâches plus prioritaires que la tâche analysée des transactions appartenant à ξ.

$$R_{ua,\xi} = \max_{\forall \Gamma_i \in \xi, \forall c \in hp_i(\tau_{ua})} R_{ua} \tag{3.20}$$

Cette borne est toujours comprise entre la valeur exacte et la valeur de la borne obtenue par l'analyse approchée de Turja et Nolin. L'algorithme d'analyse nécessite le traitement de $|hp_i(\tau_{ua})|^{|\xi|} = |hp_i(\tau_{ua})|^E$ instants critiques. Une meilleure borne pourra encore être obtenue, en traitant toutes les combinaisons (ensembles) de E transactions du système, et à la fin le minimum est choisi comme le pire temps de réponse. Soit Ω l'ensemble de combinaisons possibles de E transactions parmi toutes les transactions du système. La meilleure borne est alors :

$$R_{ua}(E) = \min_{\xi \in \Omega} R_{ua,\xi} \tag{3.21}$$

Dans ce cas le nombre de scénarii (ensembles ξ) qui doivent être traités est égal au nombre de combinaisons possible de E éléments parmi $|\Gamma|$ transactions.

$$K = \frac{|\Gamma_i|!}{E!\ (|\Gamma_i| - E)!} \tag{3.22}$$

Notons que lorsque la valeur du paramètre E augmente, la méthode mixte converge vers l'analyse exacte. C'est-à-dire plus la valeur de E croit plus la qualité de la borne de temps de réponse est bonne, mais cela est accompagné par une croissance dans la complexité de calcul. Notons $R_{ua}(e)$ le temps de réponse fourni par la méthode mixte avec $E = e$, nous avons alors.

$$R_{ua}^{exact} = R_{ua}^{mixte(|\Gamma|-1)} \leq ... \leq R_{ua}(e) \leq ... \leq R_{ua}(1) \leq R_{ua}^{Nolin}$$

La méthode mixte présentée précédemment (section 3.3) correspond à la méthode paramétrée lorsque $E = 1$ (voir l'annexe A.2 pour les formules complètes et l'annexe B.2 pour l'algorithme d'implémentation).

3.5 Étude de complexité

Pour l'implémentation de la méthode mixte, nous reposons sur l'algorithme de représentation statique d'interférence proposé pour la méthode approchée de Turja et Nolin [62, 63]. Par conséquent, la seule différence en complexité de calcul entre la méthode mixte et la méthode approchée de Turja-Nolin se localise au niveau de nombre de scénarii à considérer. Notons que la complexité de la méthode approchée de base [70, 61] est $O(X\ |\Gamma_i|^3)$, où X est le nombre d'itérations étudiées du processus de la recherche de point fixe.

La complexité de la représentation statique de l'interférence dans des tables est $O\left(|\Gamma_i|^3\right)$. La recherche d'un point dans une table d'interférence est de complexité $O\left(log\,|\Gamma_i|^2\right)$, d'où l'analyse approchée de Turja et Nolin s'effectue avec une complexité $O(|\Gamma_i|^3 + X\ log\,|\Gamma_i|^2)$.

Pour l'analyse mixte une seule opération de représentation statique d'interférence est effectuée pour tous les instants critiques considérés. Le nombre d'instants critiques dépend de la valeur E, il est déterminé par le nombre de combinaisons de E transactions multiplié par le nombre de combinaisons de tâches de E transactions.

$$K.\,|\Gamma_i|^E$$

K est le nombre de combinaison de E transactions parmi toutes les transactions du système (équation 3.22). La complexité totale de la méthode mixte est alors :

$$O\left(|\Gamma_i|^3 + K.\,|\Gamma_i|^E\,.X.log\,|\Gamma_i|^2\right) \tag{3.23}$$

La complexité de la méthode avec $E = 1$ est la suivante :

$$O\left(|\Gamma_i|^3 + |\Gamma_i|^2 .X.log\,|\Gamma_i|^2\right)$$

Si la valeur de E vaut 2 alors la complexité devient

$$O\left(|\Gamma_i|^3 + |\Gamma_i|^4 .X.log\,|\Gamma_i|^2\right)$$

Il est clair que la complexité augmente avec le nombre de transactions fixées pour l'analyse exacte. Afin d'évaluer les apports et les limites d'application de la méthode mixte sur des systèmes de taille réaliste, nous effectuons dans la section suivante une étude expérimentale.

3.6 Expérimentation

Afin de montrer les améliorations apportées par la méthode d'analyse mixte de temps de réponse (avec des valeurs différentes pour E) par rapport aux méthodes existantes, nous avons implémenté les méthodes d'analyse suivantes :

- Nolin : la méthode approchée basée sur l'interférence effective de Turja et Nolin [62, 63].
- NM1 : la méthode mixte, en fixant le nombre de transactions pour lesquelles une analyse exacte est appliquée à 1 ($E = 1$).
- NM2 : la méthode mixte avec $E = 2$.
- NM3 : la méthode mixte avec $E = 3$.
- Exact : la méthode d'analyse exact [107, 70].

L'implémentation de ces méthodes a été réalisée avec les équations complètes de calcul de temps de réponse. C'est-à-dire qu'elle calcule le temps de réponse de chaque instance de la tâche analysée pour chaque instant critique considéré, et prend en compte l'interférence de la transaction contenant la tâche analysée. L'algorithme de la représentation statique [60, 62] de l'interférence est utilisée pour toutes les méthodes.

Dans les graphes de résultat, chaque point est obtenu pour une moyenne de 100 systèmes de transactions générés aléatoirement. Les tests proposés ci-dessous ont été réalisés sur un ordinateur de type PC, équipé d'un processeur Pentium IV et disposant de 1Go de mémoire vive.

3.6.1 Description du générateur aléatoire

Le générateur des systèmes de transactions prend en entrée plusieurs paramètres réglables : charge totale du système (en pourcentage %), le nombre de transactions par système et le nombre de tâches par transaction. En fonction de ces paramètres les autres propriétés des tâches sont générées :

- La charge totale de système est uniformément distribuée sur les transactions suivant l'algorithme UUniFast (voir l'annexe B.5) présenté dans [19].
- Les périodes T_i des transactions sont générées aléatoirement entre 1000 et 1000000 (distribution uniforme)
- La charge d'une transaction Γ_i est distribuée sur les tâches suivant l'algorithme UUnifast, les C_{ij} sont calculés en fonction de T_i et de la charge.

- Les offsets O_{ij} des tâches d'une transaction sont distribués aléatoirement dans la période (distribution uniforme)
- Les delais (D_{ij}) critique sont générés aléatoirement entre C_i et T_i par une distribution uniforme.
- Les priorités des tâches sont affectées suivant la politique d'ordonnancement DM.
- Le facteur de blocage B_{ij} et la gigue sur activation J_{ij} sont nuls.

3.6.2 Critères de comparaison

L'évaluation de performance des méthodes implémentées par rapport à la méthode d'analyse exacte est basée sur les critères présentés ci-après. Deux types de tests sont effectués, un en fonction de la variation du nombre de tâches par transaction, et l'autre en fonction de la variation du nombre de transactions par système.

- *Taux de pessimisme (%) :* Cette métrique mesure le pessimisme de temps de réponse fourni par une méthode M par rapport à la valeur exacte de temps de réponse. Elle est déterminée par la formule suivante.

$$(R_{ua}^{M} - R_{ua}^{Exacte})/R_{ua}^{Exacte} \tag{3.24}$$

  Notons que, plus le taux de pessimisme est faible, meilleure est la méthode M.

- *Taux de nombre de tâches avec pessimisme (%) :* Le nombre de tâches dans un système pour lesquelles une méthode M fournit un temps de réponse pessimiste (quel soit le taux de pessimisme). Soit un système de N tâches, parmi elles P tâches ayant un temps de réponse pessimiste. Le taux est donné alors par le rapport suivant :

$$\frac{P}{N} \tag{3.25}$$

- *Temps de calcul moyen (secondes ou millisecondes) :* Le temps moyen que chaque méthode met pour analyser complètement un système.

3.6.3 Performance de la méthode mixte

Les figures 3.4 et 3.5 regroupent les taux de pessimisme moyen de la méthode approchée de Turja-Nolin et la méthode mixte pour $E = 1..3$ (NM1, NM2, NM3). Les systèmes générés pour ces tests ont une charge processeur totale égale à 80%. La figure 3.4 montre l'évolution du pessimisme des méthodes en fonction du nombre de transactions (entre 2 et 12) en fixant le nombre de tâches par transaction à 5. On peut remarquer aussi que le pessimisme des méthodes croît avec le nombre de transactions mais avec des taux de croissance differents. Cette croissance est justifiée par le fait que plus il y a de transactions plus il y a une probabilité qu'une transaction soit une source de pessimisme. Notons que toutes les transactions ont la même probabilité d'être une source de pessimisme car elles contiennent le même nombre de tâches. Le taux de pessimisme de NM3 et NM2 sont très faibles par rapport à celui de Nolin ce qui signifie que le pessimime est rarement généré par plus de 2 transactions au même temps. Par exemple pour un système de

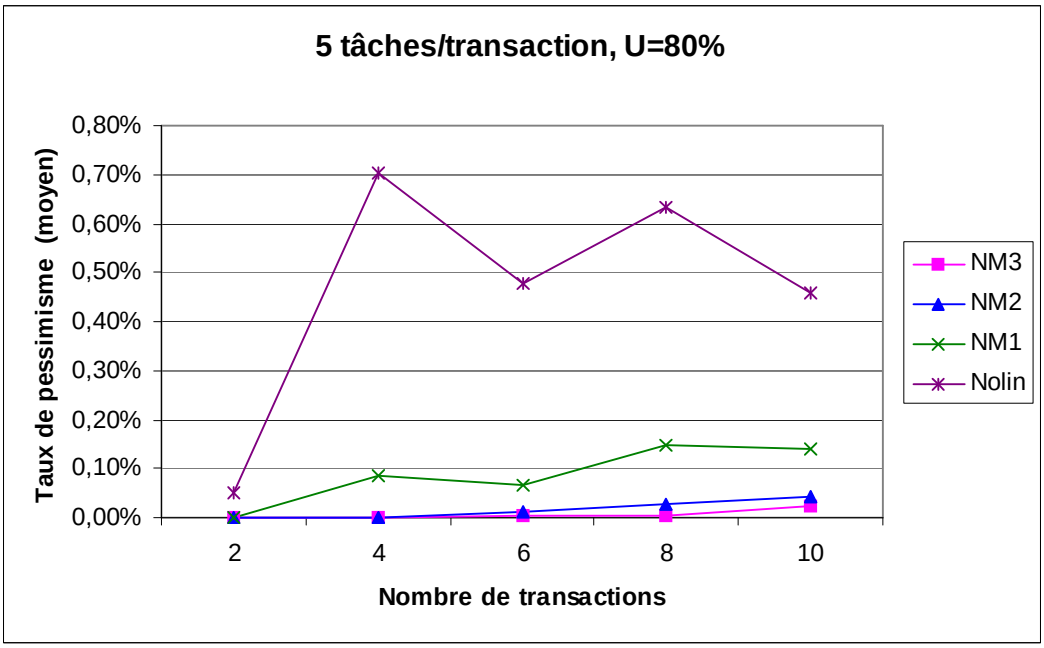

FIG. 3.4 – Influence du nombre de transactions sur le pessimisme.

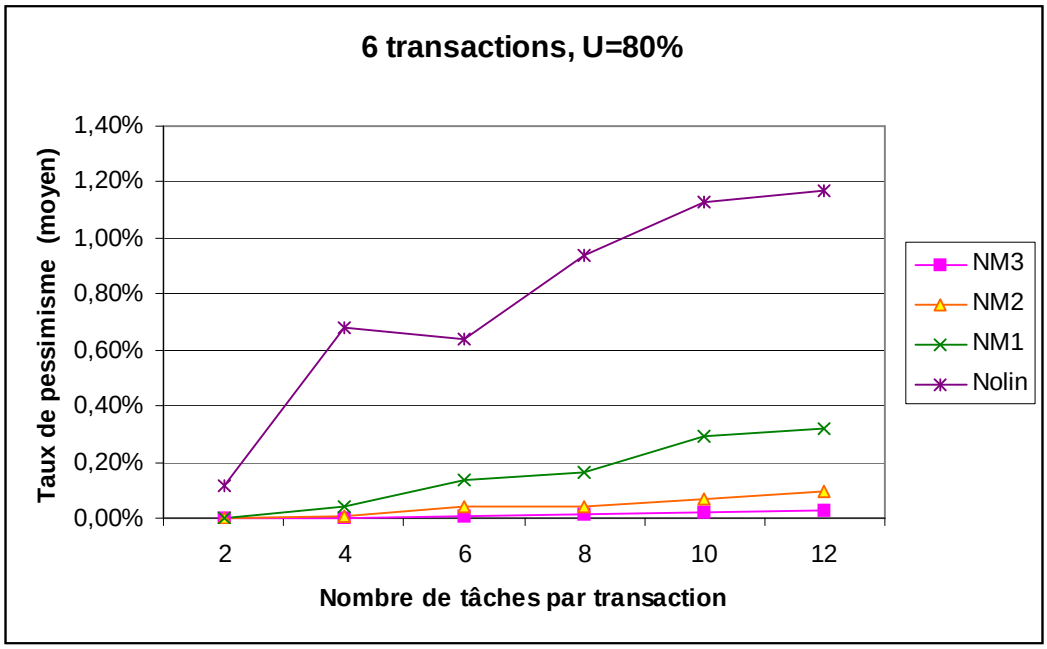

FIG. 3.5 – Influence du nombre de tâches sur le pessimisme.

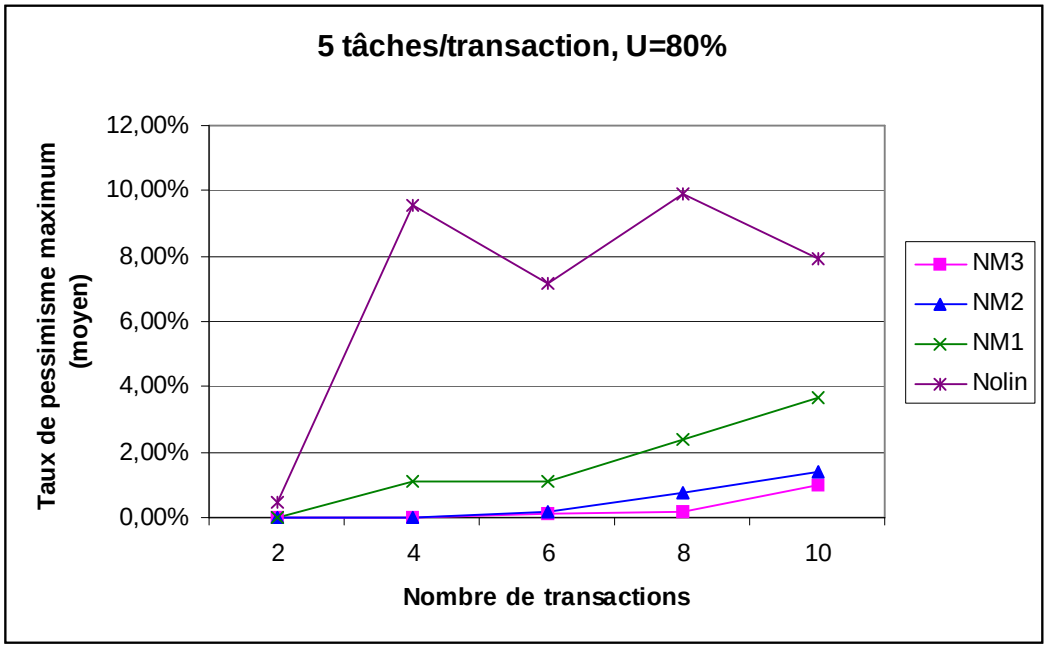

FIG. 3.6 – Influence du nombre de transactions sur le pessimisme maximum.

10 transactions de 5 tâches NM2 et NM3 fournissent un temps de réponse avec un pessimisme moyen de 0.04%, NM1 a un pessimisme moyen de $0,14\%$, tandis que la méthode de Nolin fournit des bornes avec un pessimsime moyen de $0,46\%$.

Nous retrouvons les mêmes constats sur l'influence du nombre de tâches par transaction sur le pessimisme des méthodes implémentées. Ces résultats sont illustrés par la figure 3.5. Dans ce cas le pessimisme est provoqué par l'augmentation du nombre de changements d'instants critiques, en augmentant le nombre de tâches par transaction, ainsi la probabilité de croisement de temps de réponse avec un changement d'instant critique s'élève.

Puisque la validité temporelle d'un système dépend de l'ordonnançabilité de chacune des tâches, alors il est très significatif d'évaluer le taux de pessimisme maximum dans chaque système. Les figures 3.6 et 3.7 présentent l'influence du nombre de transactions et du nombre de tâches respectivement, sur le taux moyen de pessimisme maximum dans un système. Nous remarquons que, pour un système de 6 transactions de 12 tâches la méthode approchée de Nolin fournit une borne de temps de réponse avec un taux de pessimisme de 21%, tandis que le WCRT fourni par la méthode mixte NM1 est pessimiste à $7,5\%$, le WCRT de NM2 est pessimiste de 4% seulement, et le WCRT de NM3 a un pessimsime de 1%, ce qui signifie que le WCRT calculé est exact pour toutes les tâches dans plus d'un système sur 2. Le taux de pessimisme maximum, comme le taux moyen, croît avec le nombre de tâches car le le taux de changement de tâche candidate, ainsi l'instant critique, augumente avec le nombre de tâches dans une transaction.

Les figures 3.8 et 3.9 présentent laux de tâches dans un système pour lesquelles les méthodes

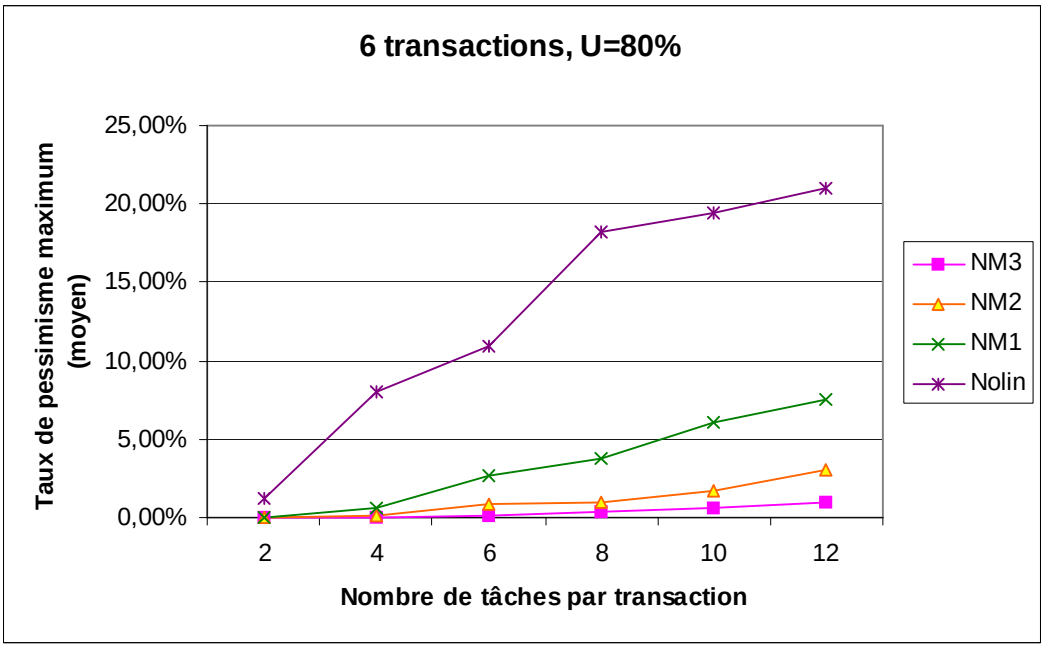

FIG. 3.7 – Influence du nombre de tâches sur le pessimisme maximum.

d'analyse fournissent un WCRT pessimiste (sans prendre en compte le taux de pessimisme du WCRT). On voit sur la figure 3.9 que pour un système de 12 transactions de 5 tâches la méthode approchée de Nolin est pessimiste pour 14% des tâches, NM1 est pessimiste pour 11% de tâches, tandis que NM2 n'est pessimiste que pour 5% et NM3 pour 3% de tâches. Dans la figure 3.8 NM2 et NM3 sont très proches et le taux de croissance des courbes est faible a cause de la stabilité du nombre de tâches dans les transactions. Tandis que dans la figure 3.9 le taux de croissance des courbes est sensiblement superieur à celui de la figure 3.8, cette croissance est due à l'augementation du risque de changer de tâche candidate (à cause de l'augmentation du nombre de tâches dans les transactions), au cours de l'analyse.

Nous avons vu que la méthode mixte est toujours moins pessimiste que la méthode approchée de Nolin, et que le pessimisme réduit en augmentant la valeur E. Les méthode NM1 et NM2 apportent une amélioration très importante par rapport à la méthode approchée de Nolin, tandis que NM3 apporte une amélioration très faible en la comparant avec celle apportée par NM2 (et peu intéressante vu le coût en temps d'exécution (figure 3.8,3.9)). Ceci peut être justifié par une probabilité plus élevée d'avoir 1 ou 2 transactions comme source de pessimisme que celle d'avoir 3 ou plus transactions étant source de pessimisme.

Les figures 3.10 et 3.11 présentent l'influence du nombre de transactions et du nombre de tâches sur le temps de traitement moyen mis par les différentes méthodes pour analyser le temps de réponse d'une tâche. Le temps mis par la méthode mixte NM1 est toujours similaire à celui mis par la méthode approchée de Nolin. NM2 nécessite un temps faiblement supérieur à celui de Nolin et NM1. Le temps mis par NM2 croit faiblement avec le nombre de tâches et le nombre

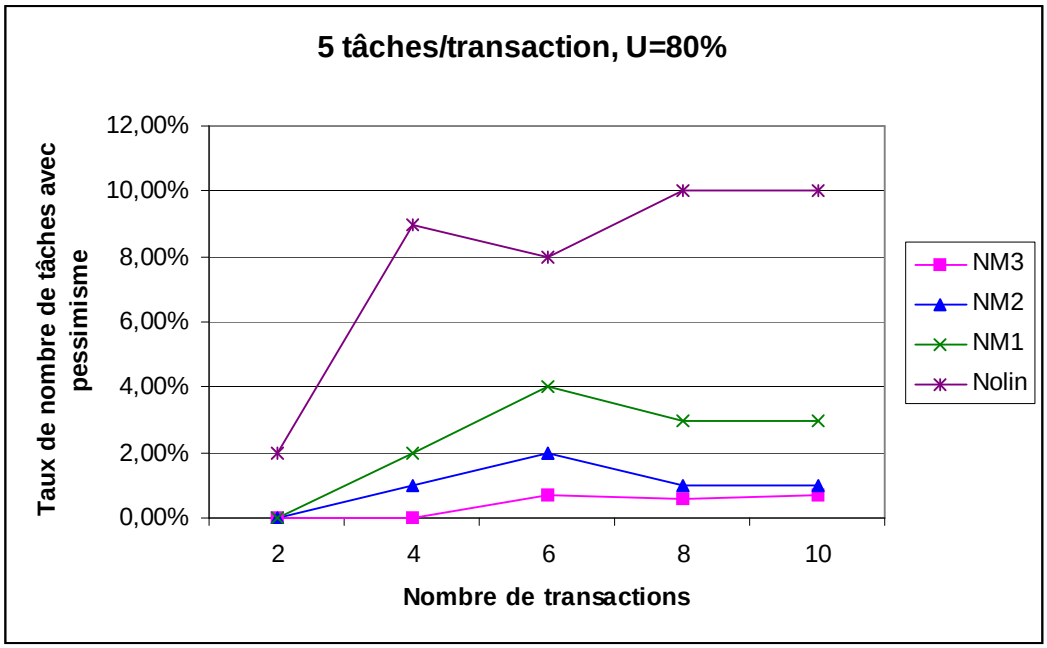

FIG. 3.8 – Influence du nombre de transactions sur le nombre de tâches avec un WCRT pessimiste.

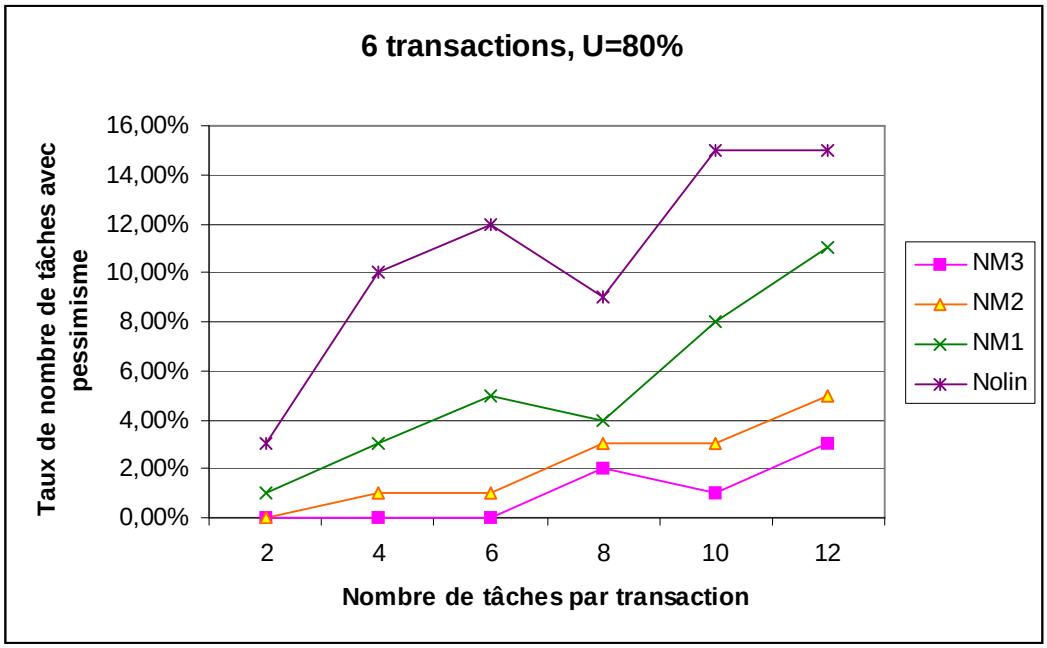

FIG. 3.9 – Influence du nombre de tâches sur le nombre de tâches avec un WCRT pessimiste.

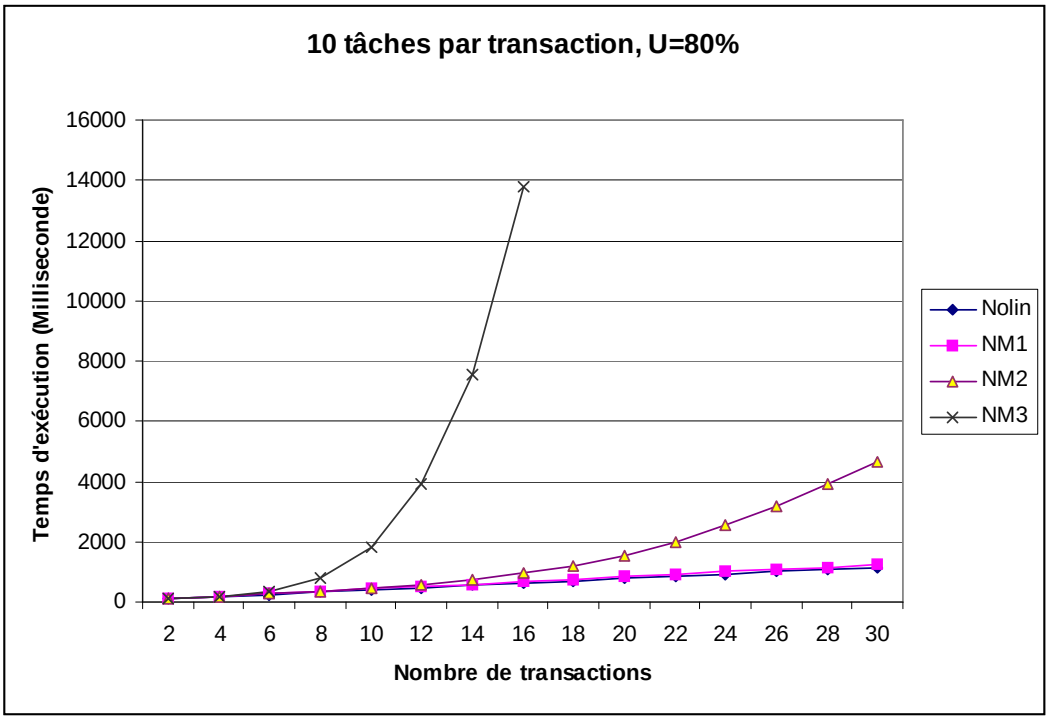

FIG. 3.10 – Influence du nombre de transactions sur le temps de calcul.

de transactions. Par exemple pour un système de 10 transactions de 30 tâches NM2 nécessite 5 secondes pour analyser une tâche, ce qui fait une différence de 1 seconde par rapport au temps mis par Nolin ou NM1 (4 secondes). Tandis que le temps de traitement nécessaire pour NM3 explose à partir de 10 transactions de 12 tâches (figure 3.11).

D'après ces résultats de simulation, nous pouvons alors toujours utiliser la méthode mixte NM1 à la place de la méthode de Nolin pour un coût similaire et une meilleure qualité de test. Pour des systèmes de taille réaliste la méthode mixte NM2 fournit un meilleur compromis qualité/temps de calcul. Tandis que la méthode NM3 est très coûteuse en temps de calcul pour une très faible amélioration en qualité de temps de réponse par rapport à NM2.

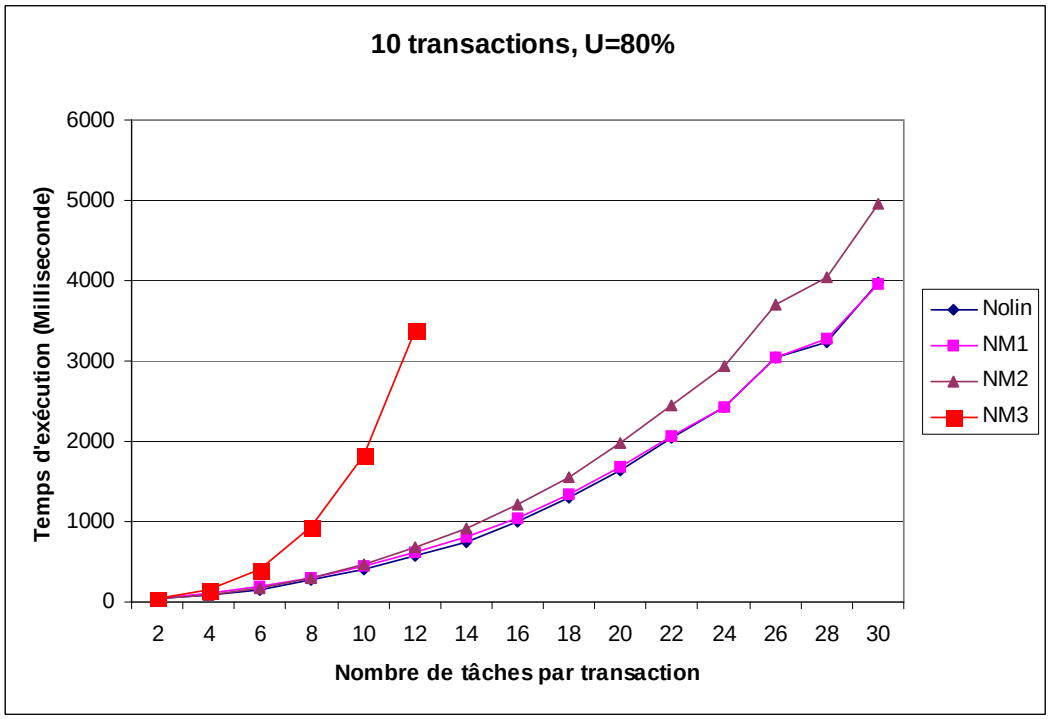

FIG. 3.11 – Influence du nombre de tâches sur le temps de calcul.

3.7 Conclusion

Dans le contexte de l'ordonnancement à priorités fixes, la valeur exacte du pire temps de réponse est donnée par une analyse exponentielle [107, 70] (condition nécessaire et suffisante d'ordonnançabilité des tâches à offsets). Afin d'éviter ce problème de complexité, une analyse approchée de temps de réponse (condition suffisante) nécessitant un temps de calcul pseudo-polynomial a été proposée par Palencia et Harbour [107, 70]. Cette analyse approchée a été améliorée en éliminant la surestimation sur le maximum des interférences des transactions, cette amélioration est réalisée par la fonction d'interférence effective proposée par Turja et Nolin [61, 62, 63]. La méthode approchée reste pessimiste, malgré l'amélioration de la fonction de maximisation d'interférence. Dans ce chapitre nous avons montré que le pessimisme de l'analyse approchée est induit par le phénomène de changement des tâches candidates initiant l'instant critique durant la calcul itératif du temps de réponse. Ce changement de tâches candidates est produit par la fonction de maximisation d'interférence des transactions.

Nous avons présenté une nouvelle méthode d'analyse [81, 82] d'évaluation de pire temps de réponse dans le contexte des transactions. Cette nouvelle méthode améliore la borne supérieure du temps de réponse fournie par la méthode d'analyse approchée de Turja-Nolin [61, 62, 63]. L'idée mise en œuvre dans cette approche est de combiner le principe du calcul exact et le principe d'approximation afin de diminuer le pessimisme de l'analyse pire cas, permettant ainsi d'améliorer la borne supérieure du temps de réponse fournie tout en conservant une complexité

pseudo-polynomiale. Cette analyse mixte de temps de réponse est paramétrée par le nombre de transactions choisies pour une analyse exacte d'interférence. Plus le nombre de transactions fixé pour une analyse exacte est élevé plus la qualité de l'analyse d'ordonnancabilité est bonne, et plus la complexité du calcul est importante. Des tests de performance ont été réalisés et ont permet les limites de l'application de cette analyse mixte pour des systèmes de taille réaliste. En résumé, les points importants mis en évidence par les tests sont :

- Les méthodes d'analyse mixte NM1 et NM2 sont considérablement moins pessimistes que la méthode approchée de Turja-Nolin. NM3 est meilleure mais avec une très légère différence par rapport à la méthode NM2.
- Le temps de traitement mis par NM1 est similaire à celui mis par la méthode approchée de Turja-Nolin. La méthode NM2 nécessite un temps de calcul légèrement supérieur à celui de NM1 et Turja-Nolin, et est moins sensible au nombre de tâches par transaction. La méthode NM3 est très gourmande en temps de calcul, son temps de traitement explose rapidement avec le nombre de tâches et de transactions.
- En terme d'application des méthodes, NM2 offre le meilleur compromis qualité/temps de calcul pour des systèmes de taille réaliste.

Chapitre 4

Identification des instants critiques pire cas, en priorités fixes

Résumé : *Nous étudions, dans ce chapitre, le problème d'identification du pire scénario, dans un ordonnancement à priorités fixes de transactions. Dans un premier temps nous étudions les transactions monotoniques et étendant les résultats de [110]. Pour ce type de transactions la tâche initiant l'instant critique est connue. Nous montrons que cette définition est limitée pour être appliquée sur le modèle général des transactions et pour identifier toutes les transactions pour lesquelles la tâche initiant l'instant critique peut être directement déterminée. Puis nous proposons la propriété d'accumulativité monotonique (AM) permettant d'identifier la tâche initiant l'instant critique dans les transaction qui la satisfont, cette propriété est plus générale, en contexte d'application et en qualité du résultat, que celle de la monotonicité. Nous montrons, ensuite l'exactitude de l'analyse approchée et mixte dans certains cas particuliers. Finalement, nous proposons la propriété de dominance des tâches, qui englobe la propriété d'AM. Cette propriété permet d'éviter l'étude de tous les instants critiques candidats initiés par des tâches dominées et ainsi permet une diminution du temps de calcul. Une étude de performance est effectuée pour chacune des propriétés présentées.*

4.1 Introduction

Le problème majeur de l'analyse exacte d'ordonnançabilté des transactions réside dans le nombre d'instants critiques candidats à étudier. En priorités fixes le pire temps de réponse d'une tâche se produit lorsqu'une tâche plus prioritaire s'active simultanément dans chaque transaction. Le problème est alors quelle tâche de chaque transaction initie l'instant critique produisant le pire temps de réponse d'une tâche analysée. Toute tâche plus prioritaire que la tâche analysée est considérée dans la création de l'instant critique pire cas. Dans ce chapitre, nous nous intéressons au problème de l'identification des tâches candidates qui peuvent réellement créer l'instant critique pire cas.

Dans le contexte des tâches MF, la propriété des tâches accumulativement monotoniques a été définie. Pour une tâche MF satisfaisant cette propriété, la trame initiant l'instant critique pire cas pour une trame moins prioritaire est connue, cette trame est appelée trame de pointe. Une tâche MF est AM si la plus grande quantité de temps processeur total requis par n'importe quelle séquence de $l \geq 1$ trames correspond à celle qui commence par la trame de pointe. Cette propriété a été utilisée comme supposition pour trouver des conditions d'ordonnançabilité moins pessimistes. Aucune étude n'a été effectuée sur la performance ou sur l'évaluation du coût de l'intégration de cette propriété dans l'analyse des tâches MF ou GMF arbitraires.

Dans le contexte des tâches à offsets, plusieurs difficultés sont rencontrées pour déterminer l'interférence maximum d'une transaction : le motif d'activations de tâches varie avec la tâche initiant la période d'activité, au contraire des tâche MF où une activation d'une trame est prévue après chaque période, une collision d'exécution des tâches à offsets et de débordement d'exécution d'une période à une autre sont possibles. En plus, la présence de la gigue rend l'état initiale de l'interférence de la transaction liée aux valeurs des gigues, cette quantité d'interférence initiale doit être prise en compte pour déterminer l'interférence maximum.

Dans ce chapitre, nous traitons le problème d'identification du pire scénario d'exécution pour une tâche, dans le contexte de l'ordonnancement à priorités fixes. Notre étude est basée sur deux idées essentielles : (1) identifier la tâche candidate initiant l'instant critique pire cas pour certains types de transactions (sections 4.2-4.3), (2) dans le cas contraire, identifier les tâches candidates n'initiant jamais l'instant critique pire cas. Dans un premier temps nous présentons, dans la section 4.2, notre étude [114, 86] effectuée sur la propriété des transactions monotoniques proposée par Traoré [111, 114, 110], dans le contexte des tâches avec gigues nulles.

4.2 Étude des transactions monotoniques

Nous commençons par une présentation, dans les sous sections 4.2.1 et 4.2.3, des résultats issus des travaux de Traoré et al [111, 114, 110]. Nous suivons, cette bref présentation de concepts, dans les sous sections suivantes, par une étude de performance et comparaison avec la méthode d'analyse approchée de Turja et Nolin.

4.2.1 Transactions monotoniques

Le concept des transactions monotoniques [111, 114] a été proposé dans le but d'accélérer le temps d'analyse de temps de réponse. L'intérêt principal de la propriété de monotonicité est de garder une seule tâche candidate pour calculer le maximum des interférences d'une transaction. En effet, dans certains cas de transactions, durant n'importe quelle période d'activité, l'interférence maximum d'une transaction correspond toujours à son interférence quand la période d'activité est initiée par une unique tâche candidate. Autrement dit, pour toute durée de période d'activité une transaction Γ_i ne change jamais la tâche initiant l'instant critique. L'exploitation de cette propriété permet d'accélérer la recherche d'instant critique pire cas (en passant le nombre de tâches candidates dans une transaction monotonique à 1).

Nous prenons l'exemple présenté sur la figure 4.1. Supposons que toutes les tâches de Γ_i soient plus prioritaires que la tâche analysée τ_{ua} et que toutes les gigues des tâches soient nulles. La table 4.1, de calcul de temps de réponse approché de la tâche τ_{ua} montre que pour n'importe quelle période d'activité de longueur t, l'interférence maximum $W_i(t)$ de Γ_i sur le temps de réponse d'une tâche moins prioritaire, correspond toujours à son interférence lorsque τ_{i1} initie l'instant critique.

Iteration	t	$W_i(t) = \max W_{ic}(t)\ (c = 1..5)$	R_{ua}
0			9
1	9	$11 = W_{i1}(t)$	20
2	20	$20 = W_{i1}(t)$	29
3	29	$29 = W_{i1}(t)$	38
4	38	$29 = W_{i1}(t)$	38

TAB. 4.1 – Analyse approchée de temps de réponse de τ_{ua} de la figure 4.1

La transaction Γ_i est dite monotonique pour toute tâche moins prioritaire τ_{ua}. Dans ce genre de situation une importante quantité de calcul pourra être évitée à condition de savoir, a priori, si la transaction est monotonique. En effet, la connaissance de la tâche τ_{ic} d'une transaction Γ_i initiant l'instant critique permet d'éviter l'examen des $(|\Gamma_i| - 1)$ tâches candidates de Γ_i. Avant de montrer comment la propriété de monotonicité d'une transaction pourra être vérifiée, nous donnons quelques définitions utiles.

Définition 7 *Une transaction Γ_i est en forme normale si les deux conditions suivantes sont satisfaites :*

- $O_{ij} + C_{ij} < O_{i(j+1)}$ *pour* $1 \leq j < |\Gamma_i|$ *et*
- $O_{i|\Gamma_i|} < T_i + O_{i1}$.

La transaction de la figure 4.1 est en forme normale car il n'y a aucun chevauchement entre l'exécution des tâches.

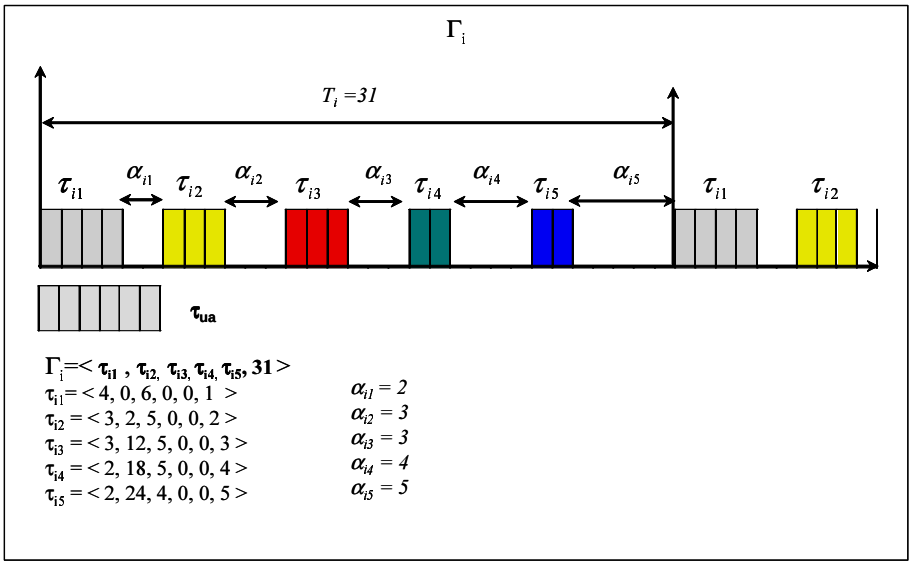

FIG. 4.1 – Exemple d'une transaction Monotonique.

4.2.2 Mise en forme normale

D'après la définition 7, le principe de mise en forme normale d'une transaction est alors de regrouper les tâches voisines qui ne respectent pas la condition de normalité (définition 7) en une tâche.

A partir d'une transaction Γ_i il est possible de trouver une transaction Γ_i^* équivalente en terme d'interférence à Γ_i et en forme normale, par la fusion des tâches voisines qui ne respectent pas les conditions de la normalité. Le processus de la normalisation est le suivant :

Soit une transaction $\Gamma_i : \{\tau_{i1}, \tau_{i2}, ..., \tau_{i|\Gamma_i|}, T_i\}$, et une tâche à analyser τ_{ua}, nous supposons que toutes les tâches de Γ_i ont une priorité supérieure à celle de τ_{ua}. Soit $\Gamma_i^* := \left\{\tau_{i1}^*, \tau_{i2}^*, ..., \tau_{i|\Gamma_i|}^*, T_i\right\}$ la transaction en forme normale de Γ_i.
Au début du processus de normalisation Γ_i^* est initialisée avec les valeurs de Γ_i :
$\Gamma_i^* := \left\{\tau_{i1}^*, \tau_{i2}^*, ..., \tau_{i|\Gamma_i|}^*, T_i\right\}$ avec $\tau_{ij}^* = \tau_{ij}$ pour $1 \leq j \leq |\Gamma_i|$.

- **Etape 1 :** Pour $1 \leq j < |\Gamma_i^*|$, si $O_{ij}^* + C_{ij}^* \geq O_{i(j+1)}^*$ alors fusionner $\tau_{i(j+1)}^*$ dans τ_{ij}^*. Ces deux tâches forment alors une seule tâche commençant à O_{ij}^* avec une durée d'exécution (WCET) égale à $C_{ij}^* + C_{i(j+1)}^*$. Renommer les tâches de la transaction dans l'ordre croissant des O_{ij}^* parce que $\tau_{i(j+1)}^*$ est supprimée.
- **Etape 2 :**
  - Si $O_{i|\Gamma_i^*|}^* + C_{i|\Gamma_i^*|}^* \geq T_i + O_{i1}^*$ alors fusionner τ_{i1}^* dans $\tau_{i|\Gamma_i^*|}^*$ avec $C_{i|\Gamma_i^*|}^* = C_{i|\Gamma_i^*|}^* + C_{i1}^*$. Renommer les tâches de la transaction et reprendre l'étape 1.
  - Sinon c'est la fin du processus de la mise en forme normale.

Ce processus converge si la charge du système est inférieure à 1. Nous montrons l'équivalence, au

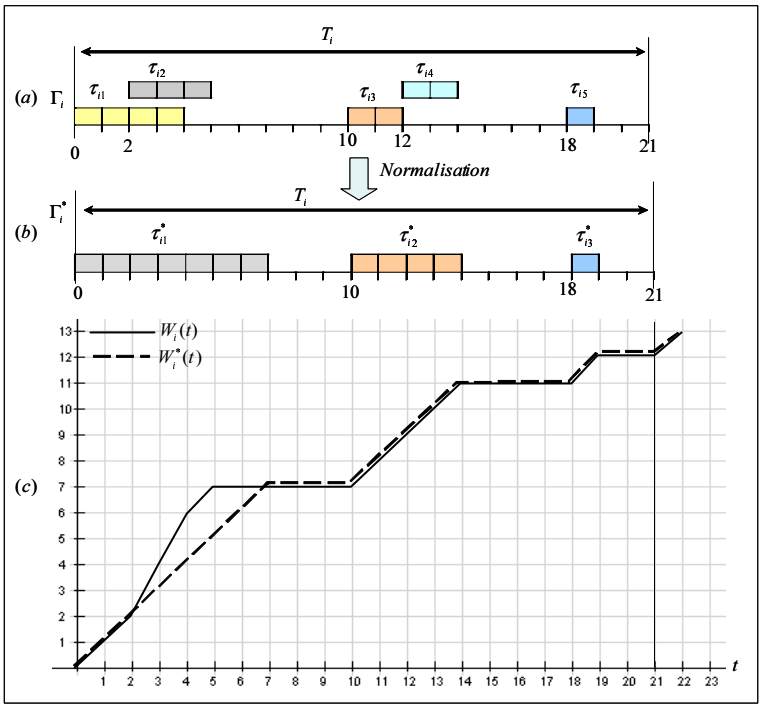

FIG. 4.2 – Interférence et forme normale.

niveau de l'interférence, entre une transaction et sa forme normale à travers l'exemple présenté sur la figure 4.2. La courbe $W_i(t)$ représente la fonction d'interférence de Γ_i. Entre l'instant $t = 2$ et $t = 4$ elle est de dérivée 2 car les deux tâches τ_1 et τ_2 demandent le processeur simultanément. Rappelons que la capacité du processeur ne peut pas dépasser le temps consacré au traitement, ainsi la dérivée de la fonction d'interférence de Γ_i doit être inférieure ou égale à 1. La courbe $W_i^*(t)$ (interférence de Γ_i^* la transaction forme normale de Γ_i) représente l'interférence de Γ_i suivant son traitement par le processeur. Nous constatons que l'interférence causée par τ_{i1} et τ_{i2} est traitée comme l'interférence d'une seule entité (sans temps creux), d'où on peut les considérer après concaténation comme une seule tâche τ_{i1}^* dans la transaction forme normale Γ_i^*. La transaction forme normale Γ_i^* d'une transaction Γ_i modélise l'interférence de Γ_i comme elle est traitée par le processeur. Par conséquent l'interférence maximum de Γ_i est égale à l'interférence maximum de la transaction Γ_i^*.

Par exemple, la transaction de la figure 4.3(b) est la transaction forme normale de la transaction présentée sur la figure 4.3(a). Après l'application du processus de mise en forme normale les tâches τ_{i2} et τ_{i3} sont fusionnées en τ_{i1}^*, les tâches τ_{i5}, τ_{i6} et τ_{i7} sont fusionnées en τ_{i3}^* ; τ_{i8}, τ_{i9} et τ_{i10} forment τ_{i4}^* et les tâches τ_{i11}, τ_{i12} et τ_{i1} sont regroupées en τ_{i5}^*. Pour cet exemple nous avons alors l'égalité suivante : $\forall t : \quad W_i(t) = W_i^*(t)$.

Définition 8 *Une transaction Γ_i est monotonique pour une tâche moins prioritaire τ_{ua} si sa*

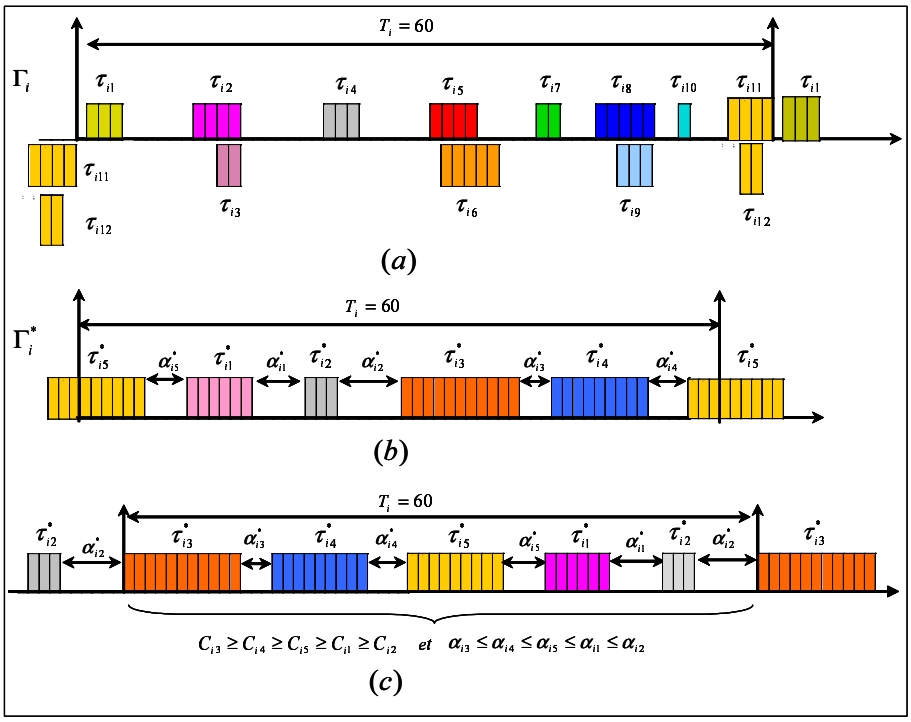

FIG. 4.3 – Processus de vérification de la monotonicité d'une transaction.

forme forme normale Γ_i^* contient un motif monotonique. La première tâche du motif monotonique est la tâche qui initie l'instant critique pour toute période d'activité.

Définition 9 *[110] : Soit $\Gamma_i : \{\tau_{i1}, \tau_{i2}, ..., \tau_{i|\Gamma_i|}, T_i\}$ une transaction, et τ_{ua} une tâche analysée moins prioritaire que toutes les tâches de Γ_i. Soit $\Gamma_i^* := \{\tau_{i1}^*, \tau_{i2}^*, ..., \tau_{i|\Gamma_i|}^*, T_i\}$ la transaction forme normale de Γ_i. Notons :*

- $\alpha_{ij}^* = O_{i(j+1)}^* - (O_{ij}^* + C_{ij}^*)$ *pour* $1 \leq j < |\Gamma_i^*|$
- $\alpha_{i|\Gamma_i^*|}^* = (T_i + O_{i1}^*) - O_{i|\Gamma_i^*|}^*$

α_{ij}^* représente le temps entre la fin d'exécution de τ_{ij} et l'activation de la tâche suivante.
Γ_i^* possède un motif monotonique si les WCET des tâches de Γ_i^* sont dans l'ordre décroissant tandis que les phases α_{ij}^* sont dans l'ordre croissant c'est à dire :

- $C_{i(p+1)}^* \leq C_{ip}^*$ *pour tout* $1 \leq p < |\Gamma_i^*|$
- $\alpha_{ip}^* \leq \alpha_{i(p+1)}^*$ *pour tout* $1 \leq p < |\Gamma_i^*|$

La transaction Γ_i de la figure 4.1, est en forme normale ($\Gamma_i = \Gamma_i^*$), et son motif monotonique débute par la première tâche τ_{i1}. En effet les C_{ij} sont en ordre croissant ($C_{i1}^* \geq C_{i2}^* \geq C_{i3}^* \geq C_{i4}^* \geq C_{i5}^*$) et les écarts α_{ij}^* sont en ordre décroissant ($\alpha_{i1}^* \leq \alpha_{i2}^* \leq \alpha_{i3}^* \leq \alpha_{i4}^* \leq \alpha_{i5}^*$).

Notons que la propriété de monotonicité d'une transaction est relative à la priorité de la tâche analysée (en effet les tâches intervenant dans la mise en forme normale dépendant de la priorité de la tâche analysée). Par exemple, si toutes les tâches, de la transaction Γ_i (Figure 4.1), sont

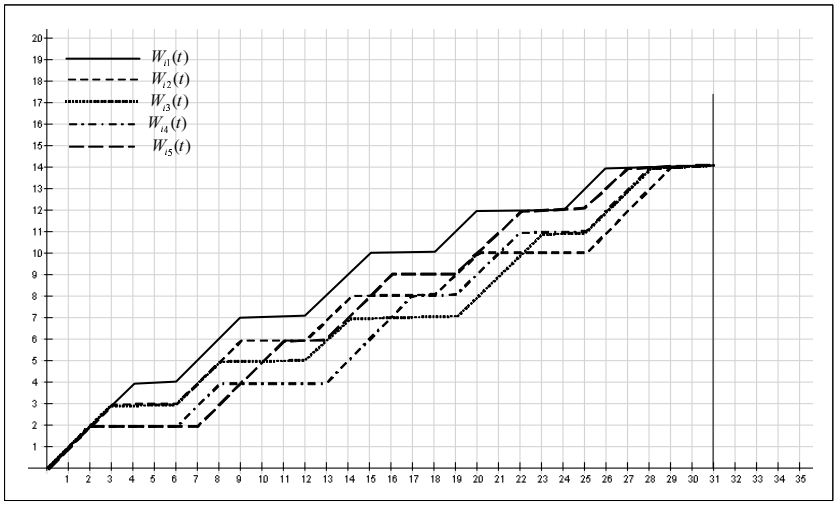

FIG. 4.4 – Interférences de la transaction monotonique Γ_i présentée dans la figure 4.1.

plus prioritaires que la tâche analysée τ_{ua}, alors Γ_i est monotonique pour τ_{ua}. Supposons que τ_{ua} est moins prioritaire que toutes les tâches de Γ_i excepté la tâche τ_{i3}, dans ce cas τ_{i3} ne contribue plus à l'interférence de Γ_i sur le temps de réponse de τ_{ua}. Γ_i est en forme normale mais elle ne vérifie pas la condition de monotonicité car la valeur de α^*_{i2} devient 9 ($\alpha^*_{i1} \leq \alpha^*_{i2} > \alpha^*_{i4} \leq \alpha^*_{i5}$) ; d'où Γ_i n'est plus monotonique pour la nouvelle τ_{ua}.

Théorème 25 *[110] : Soit* $\Gamma_i := \{\tau_{i1}, \tau_{i2}, ..., \tau_{i|\Gamma_i|}, T_i\}$ *une transaction et* τ_{ua} *une tâche à analyser. Soit* Γ^*_i *la forme normale de la transaction* Γ_i*. Si* Γ_i *est monotonique pour la tâche* τ_{ua}*, alors l'instant critique de* τ_{ua} *survient quand elle est activée au même instant que la première tâche de* Γ^*_i*.*

Pour expliquer le résultat du théorème 25, nous constatons sur la figure 4.4 que, durant la première période $T_i = 31$, la courbe d'interférence $W_{i1}(t)$ de la transaction monotonique $\Gamma_i = \Gamma^*_i$ (présentée sur la figure 4.1) lorsque la tâche du point τ_{i1} initie la période d'activité est toujours supérieure ou égale à son interférence lorsque toute autre tâche $\tau_{ik} \neq \tau_{i1}$ initie la période d'activité. Puisque la fonction d'interférence est périodique alors pour tout intervalle de temps t l'interférence maximum de Γ_i correspond à celle lorsque τ_{i1} initie la période d'activité, ainsi la tâche candidate de Γ_i qui initie l'instant critique est $\tau_{i1} = \tau^*_{i1}$ (celle qui débute le motif monotonique).

Afin de vérifier si une transaction Γ_i est monotonique pour une tâche τ_{ua} deux opérations principales sont nécessaires : la mise en forme normale (ou la recherche de la transaction forme normale Γ^*_i) de la transaction Γ_i et l'opération de vérification de l'existence d'un motif monotonique dans la transaction forme normale Γ^*_i. En fonction du résultat de cette dernière opération la propriété de monotonicité est affectée ou non à la transaction Γ_i pour le niveau de priorité de la tâche analysée. Dans la suite nous présentons le processus de vérification de la monotonicité d'une transaction en forme normale.

4.2.3 Vérification de la monotonicité d'une transaction

Le processus de vérification de l'existence d'un motif monotonique est le suivant :
Pour une transaction en forme normale, $\Gamma_i^* := \left\{\tau_{i1}^*, \tau_{i2}^*, ..., \tau_{i|\Gamma_i|}^*, T_i\right\}$, il n'y a pas de différence, relativement à la plus longue période d'activité à considérer :
$\Gamma_i^* := \left\{\tau_{i1}^*, \tau_{i2}^*, ..., \tau_{i|\Gamma_i|}^*, T_i\right\}$ et
$\Gamma_i^* := \left\{\tau_{ik}^*, \tau_{i(k+1)}^*, ..., \tau_{i|\Gamma_i|}^*, \tau_{i1}^*, \tau_{i2}^*, ..., \tau_{i(k-1)}^*, T_i\right\}$.

Nous pouvons faire une rotation des tâches de la transaction Γ_i^* sans modifier l'interférence imposée par Γ_i^* sur les tâches de priorité faible. Pour cette raison, nous considérons que Γ_i est monotonique si nous pouvons trouver un motif monotonique dans Γ_i^* par rotation des tâches de Γ_i^*. Nous savons que pour un motif monotonique, nous commençons par la première tâche ayant le WCET le plus élevé. Afin de rechercher un motif monotonique, nous commençons faire un inventaire de toutes les tâches avec le WCET maximum. Puis, nous considérons alternativement chacune de ces tâches τ_{ik}^* comme la première tâche de la transaction Γ_i^* par rotation des tâches de Γ_i^* ; et nous vérifions si les conditions de monotonicité (sur C_{ij}^* et α_{ij}^*) sont respectées ; si c'est le cas, Γ_i est monotonique et τ_{ik}^* devient la première tâche de Γ_i^*. Ainsi, $\Gamma_i^* := \left\{\tau_{ik}^*, \tau_{i(k+1)}^*, ..., \tau_{i|\Gamma_i|}^*, \tau_{i1}^*, \tau_{i2}^*, ..., \tau_{i(k-1)}^*, T_i\right\}$.

La transaction (elle est en forme normale) présentée dans la figure 4.3(b) contient un motif monotonique qui commence par la tâche τ_{i3}^* et vérifiant les conditions de monotonicité. Ce motif est illustré par la figure 4.3(c). Donc la transaction de la figure 4.3(a) est monotonique pour toute tâche à analyser moins prioritaire que toutes les tâches de Γ_i.

4.2.4 Remarques

- Le temps de réponse d'une tâche analysée τ_{ua} obtenu par l'analyse approchée ou mixte est exact si toutes les transactions du système sont monotoniques pour la tâche τ_{ua}, ainsi la tâche initiant l'instant critique de chacune des transactions est connues. La propriété de monotonicité n'apporte pas d'amélioration sur le pire temps de réponse obtenu par la méthode approchée mais elle fournit une information supplémentaire sur l'exactitude du pire temps de réponse et une éventuelle accélération de calcul (section 4.2.6).
- Dans le processus de normalisation et de vérification de la monotonicité d'une transaction pour une tâche analysée, une seule opération de mise en forme et une seule opération de recherche du motif monotoniques sont nécessaires, tandis que pour la méthode approchée de Turja-Nolin, au minimum $|\Gamma_i|$ opérations de mise en forme normale plus le calcul du maximum des interférences sont nécessaires.
- L'analyse de temps de réponse en utilisant la propriétés des transactions monotoniques n'est applicable que sur des systèmes de tâches à offsets avec gigues nulles.

4.2.5 Exemple d'application

Le système de la figure 4.3 contient une seule transaction composée de douze tâches avec gigues nulles, et une tâche à analyser τ_{ua} avec une durée d'exécution $C_{ua} = 9$. Supposons que toutes les tâches du système aient une priorité supérieure à celle de la tâche τ_{ua}.

$$\Gamma_i := \{< \tau_{i1}\tau_{i2}, ..., \tau_{i12} >, 60\}$$

$$\begin{aligned}
\tau_{i1} &:< 3, 1, D_{i1}, 0, 0, 1 > & \tau_{i7} &:< 2, 36, D_{i7}, 0, 0, 7 > \\
\tau_{i2} &:< 4, 9, D_{i2}, 0, 0, 2 > & \tau_{i8} &:< 5, 43, D_{i8}, 0, 0, 8 > \\
\tau_{i3} &:< 2, 11, D_{i3}, 0, 0, 3 > & \tau_{i9} &:< 3, 46, D_{i9}, 0, 0, 9 > \\
\tau_{i4} &:< 3, 20, D_{i4}, 0, 0, 4 > & \tau_{i10} &:< 1, 49, D_{i10}, 0, 0, 10 > \\
\tau_{i5} &:< 4, 29, D_{i5}, 0, 0, 5 > & \tau_{i11} &:< 4, 56, D_{i11}, 0, 0, 11 > \\
\tau_{i6} &:< 5, 31, D_{i6}, 0, 0, 6 > & \tau_{i12} &:< 2, 57, D_{i12}, 0, 0, 12 >
\end{aligned}$$

Nous définissons Γ_i^* la forme normale de la transaction Γ_i en appliquant les opérations du processus de normalisation. Au début, nous mettons $\Gamma_i^* = \Gamma_i$, puis nous fusionnons les tâches qui ne respectent pas les conditions de normalité. Alors les tâches τ_{i2}^* et τ_{i3}^* forment une seule tâche qui sera renommée τ_{i1}^* ; les tâches τ_{i5}^*, τ_{i6}^* et τ_{i7}^* sont regroupées dans une seule tâche τ_{i3}^* ; les tâches τ_{i8}^*, τ_{i8}^* et τ_{i10}^* sont regroupées dans τ_{i4}^* et les tâches τ_{i11}^*, τ_{i12}^* et τ_{i1}^* sont regroupées dans τ_{i1}^*. La tâche τ_{i4}^* a été renommée en τ_{i2}^*. D'où Γ_i^* la forme normale de Γ_i qui est composée seulement de cinq tâches.

$$\Gamma_i^* : \{< \tau_{i1}^*, \tau_{i2}^*, ..., \tau_{i5}^* >, 60\}$$

$$\begin{aligned}
\tau_{i1}^* &:< 6, 9, D_{i1}, 0, 0, 1 > & \tau_{i2}^* &:< 3, 20, D_{i2}, 0, 0, 2 > \\
\tau_{i3}^* &:< 11, 29, D_{i3}, 0, 0, 3 > & \tau_{i4}^* &:< 9, 43, D_{i4}, 0, 0, 4 > \\
\tau_{i5}^* &:< 9, 56, D_{i5}, 0, 0, 5 >
\end{aligned}$$

les phases entre les tâches de la forme normale sont :
$\alpha_{i1}^* = 5, \alpha_{i2}^* = 6, \alpha_{i3}^* = 3, \alpha_{i4}^* = 4, \alpha_{i5}^* = 4$

Un motif monotonique commence toujours par une tâche ayant la durée d'exécution maximum. D'où nous vérifions la monotonicité du motif qui commence par la tâche τ_{i3}^*, ce motif vérifie la condition appliquée sur les durées d'exécutions des tâches et celle appliquée sur les phases entre les tâches. Nous avons

$$\begin{aligned}
C_{i3}^* \geq C_{i4}^* \geq C_{i5}^* \geq C_{i1}^* \geq C_{i2}^* \\
\alpha_{i3}^* \leq \alpha_{i4}^* \leq \alpha_{i5}^* \leq \alpha_{i1}^* \leq \alpha_{i2}^*
\end{aligned}$$

La transaction Γ_i est alors monotonique pour la tâche à analyser, et l'instant critique est toujours initié par la première tâche du motif monotonique τ_{i3}^*. Connaître la tâche initiant l'instant

Iteration	t	$W_i(t) = W^*_{i3}(t)$	R_{ua}
0			9
1	9	11	20
2	20	20	29
3	29	29	38
4	38	29	38

TAB. 4.2 – Calcul de temps de réponse

critique élimine tout calcul d'interférence correspondant aux autres tâches candidates. Nous appliquons le calcul itératif du temps de réponse de la tâche τ_{ua}.

Nous remarquons une grande différence en nombre d'opérations effectuées entre la méthode des transactions monotoniques et la méthode approchée de Turja-Nolin. Une seule opération de normalisation et de vérification de la monotonicité de la transaction est nécessaire pour trouver la tâche initiant l'instant critique, ainsi que pour calculer l'interférence de Γ_i. Tandis que la méthode de Nolin aura besoin au minimum de 12 opérations d'ordre et de normalisation.

4.2.6 Évaluation de performance

Afin de montrer l'amélioration en temps de calcul de temps de réponse qui pourra être apportée par la propriété des transactions monotoniques, nous avons implémenté la méthode approchée de Turja-Nolin [62, 63] sans utiliser et en utilisant la propriété de monotonicité des transactions. Les temps présentés dans les graphes correspond aux temps mis par les deux méthodes d'analyse pour calculer le pire temps de réponse d'une tâche analysée moins prioritaire que toutes les tâches des systèmes générés aléatoirement. Les systèmes générés ne contiennent que des transactions monotoniques pour la tâche analysée. Dans les courbes du résultat présentés au-dessous, chaque point est obtenu par une moyenne de 1000 systèmes de transactions monotoniques générés aléatoirement. Les tests proposés ci-dessous ont été réalisés sur un ordinateur de type PC, équipé d'un processeur Pentium IV et disposant de 1Go de mémoire vive.

4.2.6.1 Description du simulateur

Le générateur des systèmes de transactions monotoniques prend en entrée plusieurs paramètres réglables : charge totale du système (en pourcentage %), le nombre de transactions par système et le nombre de tâches par transaction. En fonction de ces paramètres les autres propriétés des tâches sont générées :

- La charge totale de système est distribuée sur les transactions suivant l'algorithme UUniFast (voir l'annexe B.5) présenté en [19].
- Les périodes T_i des transactions sont générées aléatoirement entre 1000 et 1000000 (distribution uniforme)

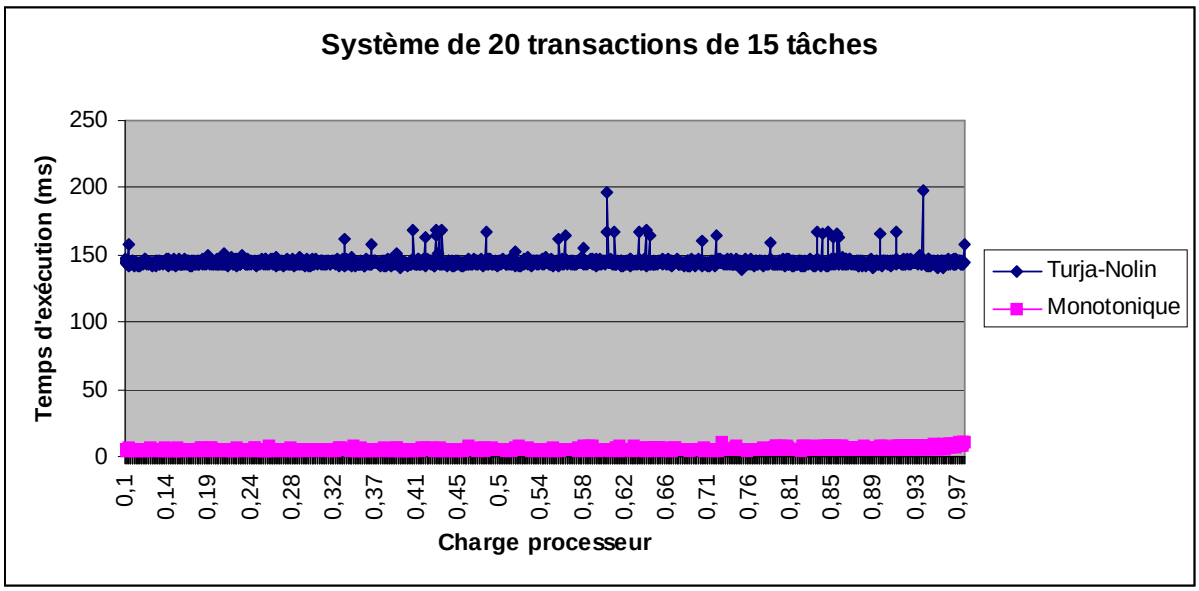

FIG. 4.5 – Influence de la charge totale du système sur le temps d'exécution.

- La charge d'une transaction Γ_i est distribuée sur les tâches suivant l'algorithme UUnifast, les C_{ij} sont calculés en fonction de T_i et la charge.
- L'offset est distribué aléatoirement dans la période i.e. dans l'intervalle $[0, T_i]$ (distribution uniforme) et est affété aux tâches d'une façon à respecter les conditions de monotonicité de la transaction.
- Les délais critiques (D_{ij}) sont générés aléatoirement entre C_i et T_i par une distribution uniforme.
- Les priorités des tâches sont affectées suivant la politique d'ordonnancement DM.
- Le facteur de blocage B_{ij} est nul.
- la gigue sur activation J_{ij} est nulle car notre définition de la propriétés de monotonicité est applicable seulement sur des transactions avec gigues nulles.

4.2.6.2 Résultat de la simulation

Dans la simulation nous avons fait varier les trois paramètres d'entrée de notre générateur de transactions monotoniques. Les tests sont faits sur le calcul du temps de réponse d'une tâche indépendante moins prioritaire que toutes les tâches des systèmes générés. En fonction de chaque paramètre variant dans les tests, nous présentons l'évolution de temps de traitement nécessaire pour les deux implémentations réalisées pour l'obtention de temps de réponse.

4.2.6.2.1 ***Influence de la charge totale du système :*** Les systèmes générés pour ce test sont composés de 20 transactions monotoniques, chacune contient 15 tâches. La charge totale du système varie entre 10% et 90%. On remarque sur la figure 4.5 que la charge du système

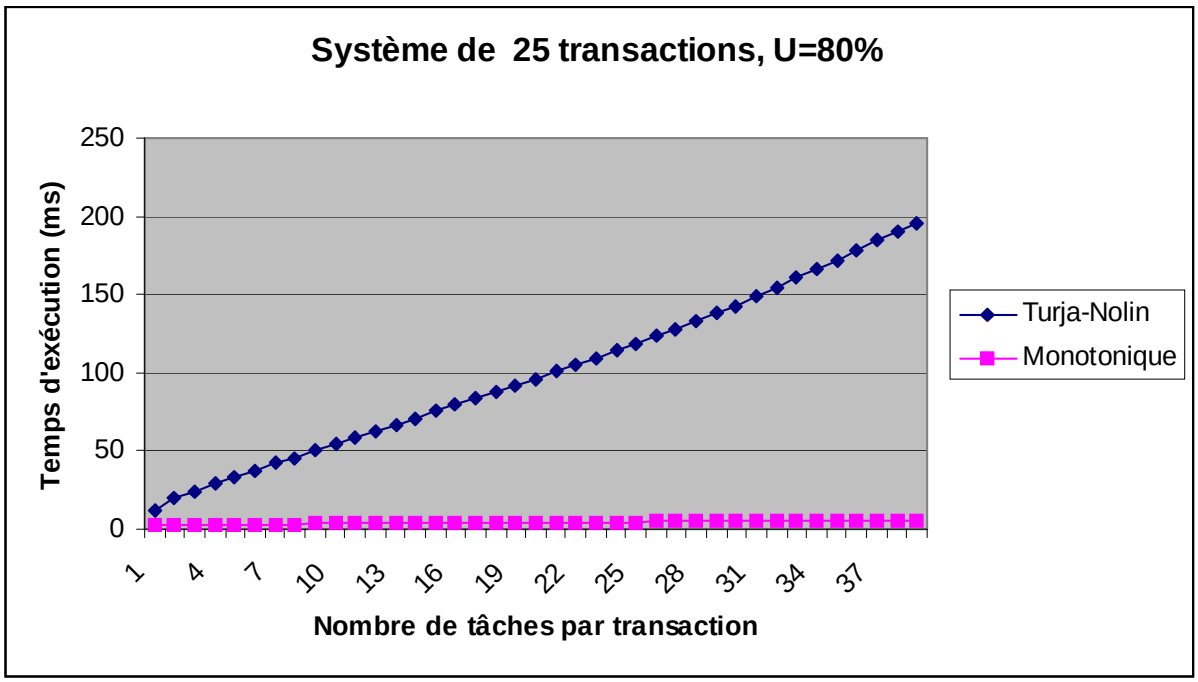

FIG. 4.6 – Influence du nombre de tâches par transaction sur le temps d'exécution.

n'a aucune influence sur le temps de traitement pour les deux implementations réalisées. Cette stabilité est due au nombre fixe de transactions et de tâches par transaction, la complexité des méthodes est définie en fonction de ces deux nombres. La difference entre les deux temps de traitement necessaires pour les deux méthodes est trés importante, ce qui fait que le gain en temps de calcul apporté par utilisation de la propriété de monotonicité est trés important par rapport au temps mis par le processus de vérification de la monotonicité.

4.2.6.2.2 ***Influence du nombre de tâches par transaction :*** Afin de montrer l'influence du nombre de tâches par transaction sur le temps de traitement nécessaire pour les deux implémentations, nous avons fait les tests sur des systèmes de transactions monotoniques avec une charge totale égale à 80%, et le nombre de transactions a été fixé à 25. Le nombre de tâches par transaction a été varié entre 1 et 40.

La figure 4.6 montre l'influence très importante du nombre de tâches sur le temps de traitement de la méthode approchée de Turja-Nolin, le temps de traitement est en croissance linéaire en fonction du nombre de tâches. Effectivement en augmentant le nombre de tâches, le nombre de tâches candidates augmente et par conséquence le nombre d'opérations de mise en forme normale et du calcul d'interférence augmente. Par contre le temps de traitement de l'implémentation en utilisant la propriétés des transactions monotoniques n'est pas influencé par le nombre de tâches, puisque le nombre d'opérations de mise en forme et de vérification de la monotonicité ne change pas car la monotonicité concerne les transactions.

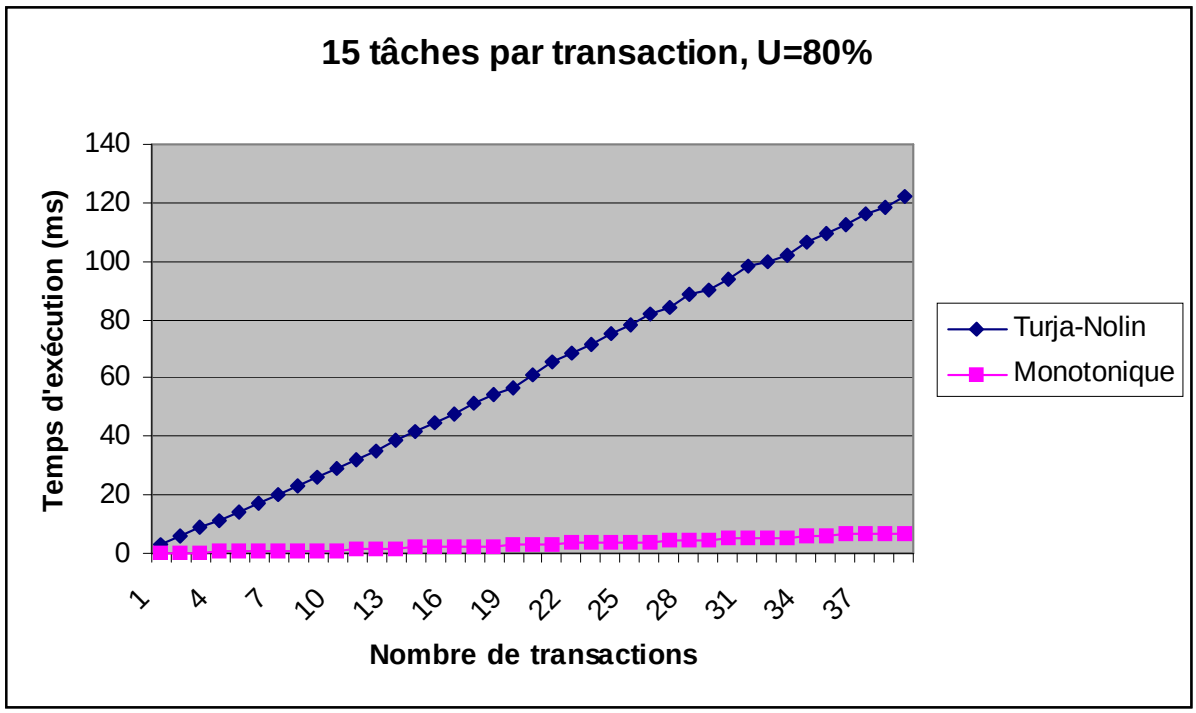

FIG. 4.7 – Influence du nombre de transactions sur le temps d'exécution.

4.2.6.2.3 ***Influence du nombre de transactions :*** De la même manière, pour simuler l'influence du nombre de transactions sur le temps de traitement des deux implémentations réalisées, les systèmes sont générés en variant le nombre de transactions entre 1 et 40, le nombre de tâches par transaction a été fixé à 15, et la charge du système à 80%. La figure 4.7 presente l'evolution du temps de traitement en fonction du nombre de transactions. Le temps de traitement de la méthode approximative de Turja-Nolin est en croissance linéaire en fonction du nombre de transactions par contre l'influence du nombre de transactions sur le temps necessaire pour la methode des transactions monotoniques est trés faible car le processus de vérification est de complexité trés faible.

D'après ces résultats de simulation, on peut conclure que le temps de traitement mis par le processus de vérification de la monotonicté des transactions est négligeable par rapport au temps mis par la méthode d'analyse approchée de Turja-Nolin. D'où l'injection de la vérification de la monotonicité des transactions dans la méthode approchée de Turja-Nolin n'aura aucune influence sur le temps de traitement de la méthode. De plus, cette propriété de monotonicité apporte une information supplémentaire sur l'exactitude des temps de réponse qui pourra être utile pour prendre des réponses négatives sur l'ordonnançabilité des systèmes. Une tâche avec un temps de réponse approché supérieur à son délai critique tel que toutes les transactions du système sont monotoniques pour elle, n'est pas ordonnançable car le temps de réponse obtenu par la méthode approchée est exact dans ce cas.

4.2.7 Limites de la propriété de monotonicité

La propriété de la monotonicité est importante pour l'identification de l'instant critique produisant le pire temps de réponse ou pour réduire le nombre d'instants critiques à considérer par la méthode exacte ou par la méthode mixte. Cependant citons plusieurs lacunes pour la définition et l'application de cette propriété :

- L'utilisation de la propriété des transactions monotonique n'est applicable que sur des systèmes de tâches avec gigues nulles. La présence des gigues provoque une différence entre les transactions forme normale correspondante à chacune des tâches candidates initiant la période d'activité. Par conséquent, la définition d'une transaction monotonique et le processus de vérification présentés précédemment deviennent difficiles à établir.
- En présence de gigue, la tâche de pointe ne correspond pas forcément à la tâche ayant le pire temps d'exécution C_{ip} car l'état initial de la fonction d'interférence est définie par les parties d'interférence induites par la gigue I_{ic}^{set1}.
- La définition des transactions monotoniques ne couvre pas toutes les transactions ayant une unique tâche initiant l'instant critique pour toute période d'activité. C'est-à-dire qu'il y a des transactions pour lesquelles la tâche initiant l'instant critique est unique mais qui ne sont pas monotoniques par définition. En effet, la définition des transactions monotoniques est basée sur l'ordre (structure) des tâches dans une transaction, pour trouver la tâche initiant l'instant critique pire cas, tandis que le calcul de pire temps de réponse est basé sur les pires interférences des transactions. Ce constat montre la nécessité de trouver une propriété plus générale (section 4.3).

  La figure 4.8 présente une transaction Γ_1 composée de trois tâches avec gigues nulles. Chaque courbe illustrée dans cette figure décrit la fonction d'interférence effective de Γ_1 sur le temps de réponse d'une tâche moins prioritaire lorsque chacune de ses tâches candidates initie l'instant critique. Nous déterminons le temps de réponse d'une tâche sous analyse τ_{ua} avec $C_{ua} = 8$ en appliquant la fonction approchée d'interférence.

Iteration	t	W_{11}	W_{12}	W_{13}	W_i	R_{ua}
0						5
1	5	1	**3**	2	**3**	8
2	8	4	**5**	3	**5**	10
3	8	4	**5**	3	**5**	10

TAB. 4.3 – WCRT de τ_{ua} dans la transaction Γ_1 de la figure 4.8.

$$\forall t : W_{12}(t) \geq W_{11}(t) \; et \; W_{12}(t) \geq W_{13}(t)$$

d'où

$$\forall t : W_1(t) = W_{12}(t)$$

La transaction présentée dans la figure 4.8 est en forme normale, elle ne contient aucun motif monotonique vérifiant les conditions de monotonicité (C_{1j} décroissant et α_{1j} crois-

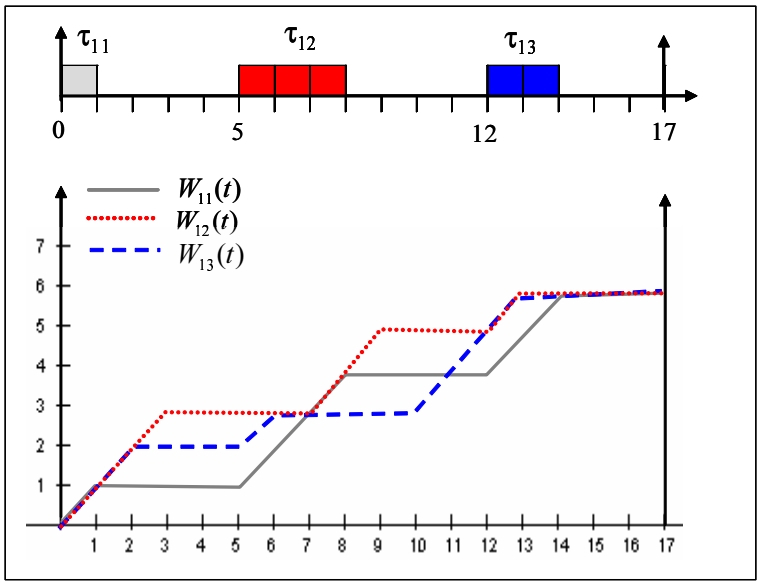

FIG. 4.8 – Exemple d'une transaction Accumulativement monotonique.

sant) puisque $\alpha_{12} > \alpha_{13}$. Donc, Γ_1 n'est pas monotonique par définition [111]. Tandis qu'en regardant les courbes de la figure 4.8 qui illustrent l'évolution de la fonction d'interférence de la transaction pour chacune de ses tâches candidates, il n'y a qu'une courbe (correspondant à la tâche candidate τ_{12}) qui enveloppe toutes les autres courbes d'interférence, pour n'importe quelle période d'activité. Cela signifie que la tâche candidate correspondante à cette courbe supérieure est toujours celle qui initie l'instant critique produisant le pire cas. Pour notre exemple de la figure 4.8, l'interférence de la transaction Γ_1 quand la tâche τ_{12} initie l'instant critique est toujours le maximum des interférences de Γ_1.

Afin de combler ces lacunes et d'évaluer l'apport de l'identification des tâches initiant l'instant critique pire cas pour certains transactions, sur l'analyse de temps de réponse dans le cas général (transactions quelconques) nous proposons la propriété des transactions Accumulativement Monotoniques (AM). Cette nouvelle propriété est complète (dans le sens où elle est appliquée sur des tâches avec gigues non nulles, et détecte toutes les transactions n'ayant qu'une seule tâche candidate initiant l'instant critique pire cas), elle est alors une extension de la propriété de la monotonicité. Toute transaction monotonique est forcément accumulativement monotonique.

4.3 Transactions Accumulativement Monotoniques

Dans cette section, nous introduisons la propriété des transactions Accumulativement Monotonique (AM). Cette propriété permet d'identifier la tâche initiant l'instant critique pour les transactions qui la satisfont. Nous donnons une définition plus générale des transactions AM, cette définition englobe toutes les transactions sans et avec gigues et celles possédant une seule tâche initiant l'instant critique.

Définition 10 *Soit* $\Gamma_i =< \tau_{i1}, \tau_{i1}, ..., \tau_{i|\Gamma_i|} >$ *une transaction et* τ_{ua} *une tâche à analyser.* Γ_i *est accumulativement monotonique pour* τ_{ua} *si et seulement si pour toute période d'activité de longueur t, l'interférence maximum de* Γ_i *est toujours produite lorsque la même tâche candidate* τ_{ip} *initie l'instant critique.*
Formellement : Γ_i *est accumulativement monotonique si et seulement si :*

$$\exists p \in hp_i(\tau_{ua}) : \forall c \in hp_i(\tau_{ua}), \forall t : W_{ic}(t) \leq W_{ip}(t) \tag{4.1}$$

Nous appelons la tâche τ_{ip} qui initie l'instant critique pour une transaction accumulativement monotonique une tâche de pointe. Par exemple la tâche τ_{12} de la transaction Γ_i illustrée dans la figure 4.8 est une tâche de pointe.

Théorème 26 *Soit* $\Gamma =< \Gamma_1, \Gamma_2, ..., \Gamma_{|\Gamma|} >$ *un système de transactions, et* τ_{ua} *une tâche analysée moins prioritaire que toutes les tâches du système. Si toutes les transactions du système sont AM pour* τ_{ua} *alors le temps de réponse de* τ_{ua} *obtenu par les méthodes approchée ou mixte est exact, l'instant critique produisant le pire cas est initié par la tâche de pointe de chacune des transactions.*

Preuve :
Nous notons $R_{ua}(f(t))$ le temps de réponse de la tâche τ_{ua}, calculé par la recherche du premier point fixe (calcul itératif), en utilisant la fonction d'interférence du système $f(t)$. Sans perte de généralité, noous supposons que toutes les tâches de toutes les transactions sont plus prioritaires que la tâche analysée τ_{ua}, et que toutes les transactions sont AM pour τ_{ua}. Puisque nous avons montré dans le chapitre 3 que le temps de réponse mixte est toujours compris entre le temps de réponse exact et le temps de réponse approchée de Nolin, il suffit alors, pour prouver le résultat du théorème 26, de prouver seulement que le temps de réponse approché de Nolin est exact. Nous pouvons noter :

– Le temps de réponse obtenue par la méthode approchée de Nolin :

$$\begin{aligned} R_{ua}^{Nolin} &= R_{ua}\left(f^{Nolin}(t)\right) \\ &= R_{ua}\left(\max_{\forall c_1 \in \Gamma_1} W_{1c_1}(t) + ... + \max_{\forall c_k \in \Gamma_k} W_{kc_k}(t) + ... + \max_{\forall c_n \in \Gamma_n} W_{nc_n}(t)\right) \end{aligned}$$

– Et le temps de réponse obtenu par la méthode exacte :

$$\begin{aligned} R_{ua}^{exact} &= \max_{\forall c_1 \in \Gamma_1, ..., c_k \in \Gamma_k, ..., c_n \in \Gamma_n} R_{ua}\left(f^{Exacte}(t)\right) \\ &= \max_{\forall c_1 \in \Gamma_1, ..., c_k \in \Gamma_k, ..., c_n \in \Gamma_n} R_{ua}\left(W_{1c_1}(t) + ... + W_{kc_k}(t) + ... + W_{nc_n}(t)\right) \end{aligned}$$

Rappelons que les fonctions d'interférences définies ci-dessus sont monotones. Pour un calcul itératif du premier point fixe, basé sur deux fonctions monotones différentes $f(t)$ et $g(t)$, nous avons la propriété suivante :

$$Si \ \ \forall t : \ \ f(t) \leq g(t) \ \ alors \ \ R_{ua}(f(t)) \leq R_{ua}(g(t)) \tag{4.2}$$

Pour une transaction AM Γ_i avec une tâche du pointe τ_{ip_i} nous avons $\max_{\forall c_i \in \Gamma_i} W_{1c_i}(t) = W_{ip_i}$, par conséquent :

$$R_{ua}^{Nolin} = R_{ua}\left(W_{1p_1}(t) + ... + W_{kp_k}(t) + ... + W_{np_n}(t)\right) \tag{4.3}$$

D'après l'équation 4.2 nous obtenons les égalités suivantes :

$$\begin{aligned} R_{ua}^{exact} &= \max_{\forall c_1 \in \Gamma_1, ..., c_k \in \Gamma_k, ..., c_n \in \Gamma_n} \left(R_{ua}\left(W_{1c_1}(t) + ... + W_{kc_k}(t) + ... + W_{nc_n}(t)\right)\right) \\ &= R_{ua}\left(\max_{\forall c_1 \in \Gamma_1, ..., c_k \in \Gamma_k, ..., c_n \in \Gamma_n} \left(W_{1c_1}(t) + ... + W_{kc_k}(t) + ... + W_{nc_n}(t)\right)\right) \end{aligned} \tag{4.4}$$

Puisque l'interférence d'une transaction est indépendante de toute autre transaction du système alors on obtient :

$$\begin{aligned} & \max_{\forall c_1 \in \Gamma_1, ..., c_k \in \Gamma_k, ..., c_n \in \Gamma_n} \left(W_{1c_1}(t) + ... + W_{kc_k}(t) + ... + W_{nc_n}(t)\right) \\ = & \max_{\forall c_1 \in \Gamma_1} W_{1c_1}(t) + ... + \max_{\forall c_k \in \Gamma_k} W_{kc_k}(t) + ... + \max_{\forall c_n \in \Gamma_n} W_{nc_n}(t) \\ = & \ W_{1p_1}(t) + ... + W_{kp_k}(t) + ... + W_{np_n}(t) \end{aligned}$$

Par conséquent le temps de réponse exact (équation 4.4) devient :

$$R_{ua}^{exact} = R_{ua}\left(W_{1p_1}(t) + ... + W_{kp_k}(t) + ... + W_{np_n}(t)\right)$$

Ce qui est équivalent au temps de réponse approché de Nolin (équation 4.3).
□

Il est indispensable de trouver un algorithme de vérification de la propriété d'AM peu coûteux en temps de calcul afin de pouvoir exploiter cette propriété pour améliorer la qualité de l'analyse d'ordonnançabilité. Dans la section suivante, nous proposons un algorithme de vérification reposant sur la périodicité de la fonction d'interférence et sur la représentation statique de l'interférence [62, 63] que nous allons adapter à nos besoins.

4.3.1 Processus de vérification de l'Accumulativité monotonique

L'idée est simple, elle consiste à stocker, pour chaque point d'interférence, une information identifiant la tâche candidate correspondante, ensuite vérifier si les points spécifiant l'interférence maximum correspondent tous à une même tâche candidate.

Turja et Nolin ont montré que l'interférence d'une transaction pendant la première période peut être différente de celle produite durant la deuxième période, et que la fonction d'interférence dans

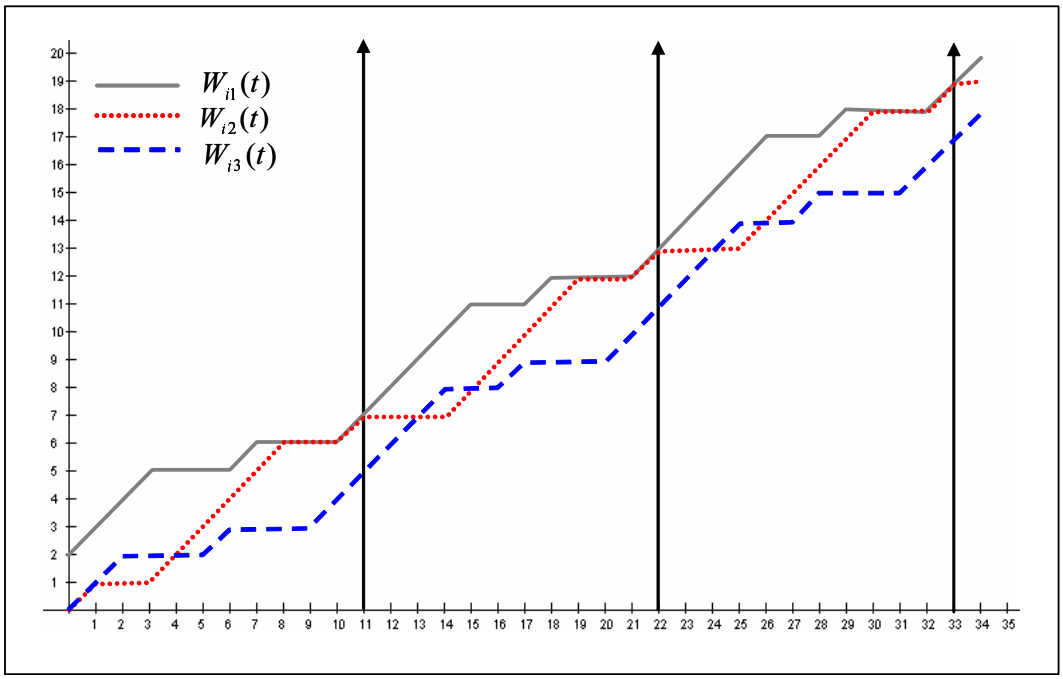

FIG. 4.9 – Périodicité de la fonction d'interférence effective.

la deuxième période se répète pour les périodes suivantes. Par conséquent, notre test d'accumulativité monotonique (équation 4.1) se limite dans l'intervalle des deux premières périodes. La figure 4.9 illustre bien la périodicité de l'interférence effective de la transaction Γ_i de la table 4.4, et montre que l'interférence de Γ_i durant la première période est différente de celle produite durant la période suivante. Pour $t > T_i$ l'interférence de Γ_i est répétitive avec une période égale à T_i.

Afin de faciliter la présentation de l'algorithme, nous considérons comme exemple la transaction présentée dans la table 4.4 pour mettre en évidence toutes les étapes de calcul : représentation statique de la fonction d'interférence, et vérification de la propriété d'accumulativité monotonique d'une transaction. Γ_i est composée de trois tâches avec une période $T_i = 11$, dont certaines ont une gigue non nulle. Rappelons que Φ_{ijc} donne la date de la première activation de τ_{ij}, après l'instant critique initié par la tâche τ_{ic}.

Tâche	O_{ij}	C_{ij}	J_{ij}	Φ_{ij1}	Φ_{ij2}	Φ_{ij3}
τ_{i1}	0	3	0	0	0	0
τ_{i2}	6	1	1	0	0	0
τ_{i3}	10	2	2	2	0	0

TAB. 4.4 – Caractéristiques d'une transaction AM.

L'interférence d'une transaction Γ_i lorsque une tâche τ_{ic} initie l'instant critique pourra être représentée statiquement dans deux tables (P_{ic} et P'_{ic}), l'une pour la première période et l'autre

pour la deuxième. Chacune de ces tables contient les coordonnées des points (date, interférence) représentants un changement de l'interférence. La fonction d'interférence maximum d'une transaction est représentée statiquement par deux tables (P_i et P'_i), ces deux tables sont déduites par la fonction « Envelopper » sur tous les points de l'ensemble des tables d'interférence P_{ic} et P'_{ic} de Γ_i pour chacune de ses tâches candidates τ_{ic}.

Nous proposons une nouvelle structure pour les points décrivant l'évolution de l'interférence. Cette structure permet de stocker plus d'informations sur l'interférence aidant à déterminer, à la fin de la phase de pré-calcul, les tâches candidates produisant la pire interférence. Chaque point d'une table de stockage contient les cordonnées suivantes, la longueur de la période d'activité x, l'interférence correspondante y, et l'indice de la tâche candidate initiant l'instant critique c.

$$point :< x, y, c >$$

Un point est traduit comme suit : Pour une période d'activité de longueur x, une interférence totale égale à y est imposée par la transaction considérée Γ_i lorsque la tâche τ_{ic} initie l'instant critique.

4.3.1.1 Pre-calcul de l'interférence

Soit τ_{ua} une tâche à analyser et Γ_i une transaction, sans perte de généralité nous supposons que toutes les tâches de Γ_i sont plus prioritaires que la tâche τ_{ua}. Notons P_{ic} et P'_{ic} les tables dans lesquelles on va stocker les informations sur la fonction d'interférence de Γ_i, lorsque τ_{ic} initie l'instant critique, durant la première et la deuxième période respectivement. Nous reprenons les mêmes opérations de pré-calcul de l'interférence présentées dans le chapitre 2 : opération d'ordre, fusion et l'opération de séparation. Pour chaque tâche candidate τ_{ic} la transaction Γ_i est alors mise en forme normale après l'exécution des trois opérations de restructuration.

Comme nous l'avons vu dans le chapitre 2, le motif de périodicité de l'interférence maximum $W_i(\tau_{ua}, t)$ est divisé en deux phases, la première correspond à la première période qui ne doit pas prendre en compte la tâche séparée (tâche débordante), la deuxième phase correspond à la deuxième période (et les périodes suivantes) dans lesquelles la tâche séparée est prise en compte dans le calcul d'interférence. L'interférence maximum $W_i(\tau_{ua}, t) = \max_{\forall c \in hp_i(\tau_{ua})} W_{ic}(\tau_{ua}, t)$ d'une transaction Γ_i contient deux parties d'interférence :

$$W_i(\tau_{ua}, t) = J_i^{ind}(\tau_{ua}) + T_i^{ind}(\tau_{ua}, t) \tag{4.5}$$

$$W_i(\tau_{ua}, t) = J_i^{ind} + T_i^{ind}(\tau_{ua}, t) \tag{4.6}$$

– La partie statique induite par la gigue notée J_i^{ind}, est forcément le maximum des parties I_{ic}^{set1}. Elle correspond aux instances activées avant la période d'activité (ensemble set1).

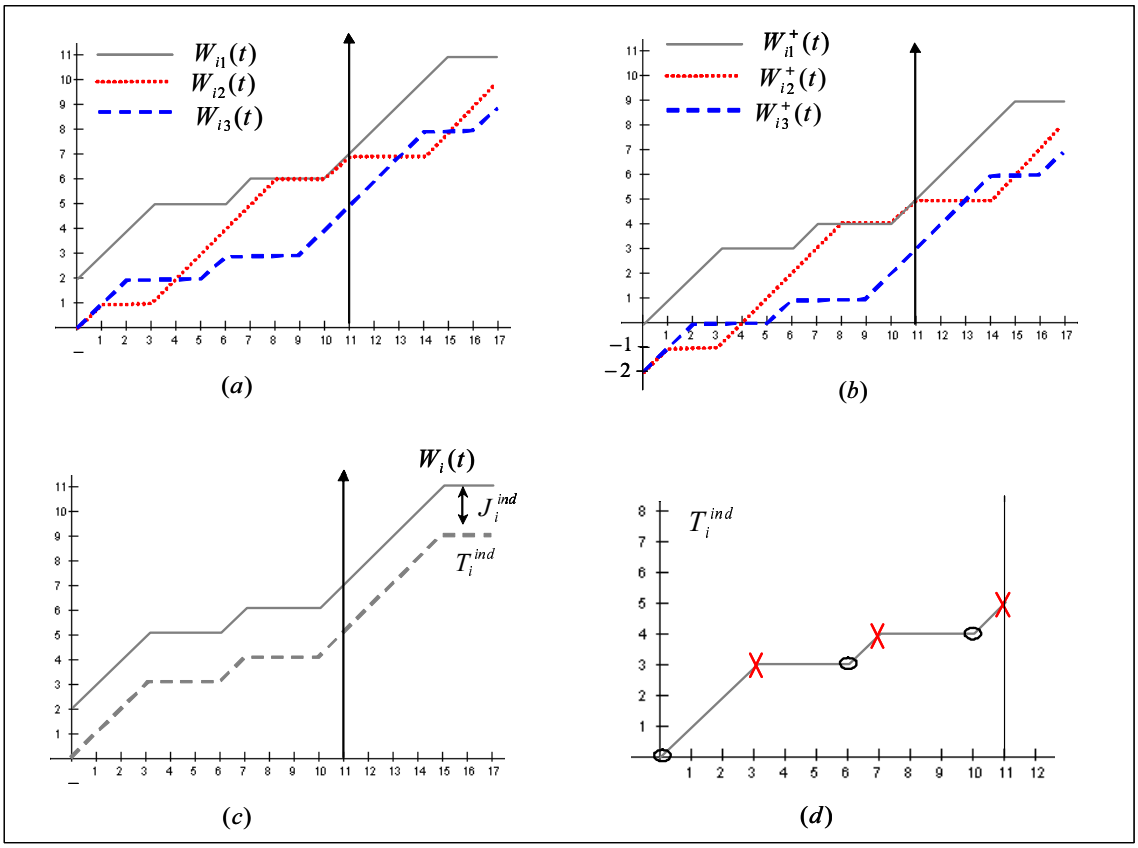

FIG. 4.10 – $W_i(\tau_{ua}, t)$, $T_i^{ind}(t)$ et $W_{ic}^+(\tau_{ua}, t)$.

$$J_i^{ind} = \max_{\forall c \in hp_i(\tau_{ua})} \sum_{\forall j \in hp_i(\tau_{ua})} I_{ijc}^{set1} \tag{4.7}$$

– La partie dynamique induite par le temps notée $T_i^{ind}(t)$, cette partie est causée par les instances de l'ensemble set2, elle possède un motif périodique de période T_i

$$T_i^{ind}(t) = \max_{\forall c \in hp_i(\tau_{ua})} W_{ic}^{+}(\tau_{ua}, t) \tag{4.8}$$

$$W_{ic}^{+}(\tau_{ua}, t) = \sum_{\forall j \in hp_i(\tau_{ua})} \left(I_{ijc}^{set1} + I_{ijc}^{set2}(t) \right) - J_i^{ind} \tag{4.9}$$

La figure 4.10 montre la relation entre ces différentes formules. En conclusion, il suffit seulement de stocker des informations sur les points convexes (marqués par des croix dans la figure 4.10(d)) de la partie d'interférence dépendante de temps $T_i^{ind}(t)$, durant les deux premières périodes. Une représentation statique de l'interférence $W_{ic}^{+}(t)$ est nécessaire pour chaque tâche candidate.

4.3.1.2 Représentation statique

Les points de la table P_{ic} stockant l'interférence $W_{ic}^{+}(t)$ produite, durant la première période, par la transaction Γ_i lorsque τ_{ic} initie l'instant critique sont déterminés par la formule 2.29.

$$\begin{aligned}
P_{ic}[1].x &= 0 \\
P_{ic}[1].y &= \sum_{\forall j \in hp_i(\tau_{ua})} I_{ijc}^{Set1} - J_i^{ind}(\tau_{ua}) \\
&\ldots \\
P_{ic}[k].x &= \Phi_{ikc} + C_{ik} \quad k \in 2..|\Gamma_i| \\
P_{ic}[k].y &= P_{ic}[k-1].y + C_{ik} \quad k \in 2..|\Gamma_i|
\end{aligned}$$

Pour notre exemple, ces points correspondent aux points convexes des courbes de la figure 4.10(b). Rappelons qu'on stocke une information supplémentaire déterminant la tâche candidate associée à chaque point.

$$P_{i1} =< (3,3,1),(7,4,1),(11,5,1) >$$
$$P_{i2} =< (1,-1,2),(8,4,2),(11,5,2) >$$
$$P_{i3} =< (2,0,3),(6,1,3),(11,3,3) >$$

Une fois que toutes les informations sur la fonction d'interférence W_{ic}^{+} de Γ_i sont stockées dans l'ensemble des tables P_{ic} associée à chacune des tâches candidates τ_{ic}, les points décrivant l'interférence maximum $T_i^{ind}(\tau_{ua}, t)$ sont déduits, après d'avoir réuni toutes les tables P_{ic}, en gardant seulement les points représentant la fonction d'interférence maximum. Cette opération est assurée par la relation « Envelopper » (équations 2.30-2.31). Un point $P_i[a]$ Enveloppe un autre

point $P_i[b]$ (notons $P_i[a] \succ P_i[b]$), si la présence de $P_i[a]$ implique que $P_i[b]$ n'est pas un point convexe pour la courbe représentant le maximum des interférences. Formellement, la relation « Envelopper » est définie par :

$$P_i[a] \succ P_i[b] \quad SSI$$

$$P_i[a].y \geq P_i[b].y \ \wedge \ (P_i[a].x - P_i[a].y \leq P_i[b].x - P_i[b].y)$$

Pour notre exemple, au premier temps, P_i est l'union de toutes les tables P_{ic} :

$$P_i =< (3,3,1),(7,4,1),(11,5,1),(1,-1,2),(8,4,2),(11,5,2),(2,0,3),(6,1,3),(11,3,3) >$$

Nous ne gardons dans P_i que les points représentants les points convexes maximum, en appliquant la relation « Envelopper ». Nous obtenons :

$$P_i =< (3,3,1),(7,4,1),(11,5,1) >$$

Ces points correspondent aux points convexes de la figure 4.10(d).

De la même manière, la table P'_{ic} stockant les informations sur l'interférence W^+_{ic} durant la deuxième période est calculée par les mêmes formules sauf que le point initial est donné par la formule 2.32.

$$P'_{ic}[1].y = P_{ic}[|\Gamma_i|].y - \max_{x \in \Gamma_i} P_{ix}[|\Gamma_i|].y \tag{4.10}$$

Notons que la tâche séparée est prise en compte dans l'interférence durant la deuxième période. Une fois que toutes les tables P'_{ic} sont calculées, la table P'_i stockant les points de l'interférence maximum est déduite par la relation « Envelopper ». Pour notre exemple, nous avons les tables suivantes :

$$P'_{i1} =< (0,0,1),(4,4,1),(7,5,1),(11,6,1) >$$

$$P'_{i2} =< (0,0,2),(8,5,2),(11,6,2) >$$

$$P'_{i3} =< (0,-2,3),(3,3,3),(6,4,3),(11,6,3) >$$

Après leur union dans la table P'_i et l'application de la relation « Envelopper » on obtient :

$$P'_i =< (0,0,1),(4,4,1),(7,5,1),(11,6,1) >$$

4.3.1.3 Vérification de l'accumulativité monotonique

Le processus du vérification consiste à vérifier si l'interférence maximum de Γ_i durant les deux premières périodes correspond toujours à une seule tâche candidate. C'est-à-dire vérifier si tous les points de P_i et P_i' correspondant à une même tâche candidate τ_{ip} (la coordonnée c dans la structure des points). Si c'est le cas alors la transaction Γ_i est AM et son interférence maximum correspond toujours a son interférence lorsque la tâche τ_{ip} initie l'instant critique (pour l'algorithme, voir l'annexe B.3).
Formellement une transaction Γ_i est accumulativement monotonique si et seulement si :

$$p = P_i[1].c,\ \forall k \in 2..|P_i|\,,\ \forall k' \in 1..|P_i'|\ :\ P_i[k].c = p\ \wedge\ P_i'[k'].c = p \tag{4.11}$$

Dans notre exemple, tous les points de P_i et P_i' correspondent à la même tâche candidate τ_{i1}, d'où la transaction Γ_i est accumulativement monotonique et sa tâche de pointe est τ_{i1}.

$$\forall k \in 1..|P_i|\,,\ \forall k' \in 1..|P_i'|\ :\ P_i[k].c = 1\ \wedge\ P_i'[k'].c = 1$$

Notons que la tâche de pointe, si elle existe, correspond à la tâche initiant l'instant critique produisant la pire interférence induite par la gigue J_i^{ind}. J_i^{ind} représente l'état initial de l'interférence maximum $W_i(t)$.

$$J_i^{ind} = I_{ip}^{set1} \tag{4.12}$$

4.3.2 Remarques

Toutes les méthodes d'analyse approchées et mixtes reposent sur la représentation statique de l'interférence. Le processus de vérification de la propriété de l'AM injecté dans l'algorithme des pré-calculs de l'interférence [62, 63] ne change alors rien au niveau de la la complexité de ces méthodes d'analyse. Plus le changement de la structure des points, un simple parcours des deux tables P_i et P_i' est nécessaire pour vérifier si la propriété d'AM est satisfaite ou non.

Puisque le temps de réponse d'une tâche analysée, obtenu soit par la méthode de Nolin soit par la méthode mixte est exact lorsque toutes les transactions sont AM pour cette tâche, alors la condition suffisante d'ordonnançabilité de la tâche analysée devient une condition nécessaire et suffisante. Dans le cas d'analyse mixte ou exacte, les instants critiques initiées par les tâches candidates (exceptée la tâche de pointe) d'une transaction AM ne seront pas considérées dans le calcul de pire temps de réponse. Ainsi une amélioration en temps de traitement est possible.

4.3.3 Expérimentation

Dans notre simulation, nous allons évaluer et quantifier l'amélioration apportée par l'exploitation de la propriété des transactions accumulativements monotoniques sur la qualité de l'analyse de temps de réponse pour les différentes méthodes, en nombre d'instants critiques (également en temps d'exécution) et en quantité des tâches pour lesquelles la condition suffisante devient

nécessaire et suffisante d'ordonnançabilité (nombre de tâches dans un système pour lesquelles nous savons que le temps de réponse calculé par les méthodes approchées ou mixtes sont exacts). Pour cela, nous avons implémenté les algorithmes d'analyse suivants :

- Nolin : La méthode approchée de Turja-Nolin [61, 62, 63]
- NM1 : La méthode mixte [81, 82] avec un nombre de transactions $E = 1$ pour lesquelles nous appliquons une fonction d'interférence exacte.
- NM2 : La méthode mixte avec $E = 2$.
- NM2 : La méthode mixte avec $E = 3$.
- Exact : La méthode exacte [107, 70].

Et d'autre part les mêmes méthodes précédentes mais avec l'utilisation du concept des transactions AM :

- Nolin_AM : La méthode approximative de Nolin [62] en exploitant la propriété des transactions AM.
- NM1_AM : La méthode mixte [81, 82] avec $E = 1$ en exploitant la propriété des transactions AM.
- NM2_AM : La méthode mixte avec $E = 2$ en exploitant la propriété des transactions AM.
- NM2_AM : La méthode mixte avec $E = 3$ en exploitant la propriété des transactions AM.
- Exact_AM : La méthode exacte [107, 70] en exploitant la propriété des transactions AM.

L'implémentation a été réalisée avec les équations complètes de calcul de temps de réponse présentées dans [70], (i.e. en prenant en compte l'interférence des tâches plus prioritaires appartenant à la transaction Γ_u contenant la tâche analysée et en calculant le temps de réponse de toutes les instances de τ_{ua} activées dans la période d'activité). Dans les courbes du résultat, chaque point est obtenu par une moyenne de 100 systèmes de transactions générés aléatoirement. Les tâches à offsets sont générées aléatoirement par le même simulateur présenté dans la section 3.6.1 du chapitre précédent.

4.3.3.1 Critères d'évaluation

Pour évaluer et quantifier les améliorations apportées par la propriété des transactions AM pour les différentes méthodes d'analyse de temps de réponse, nous nous sommes focalisés sur quatre critères :

- *Taux du nombre de tâches avec temps de réponse exact (%) :*
  Cette métrique estime la proportion des tâches dans un systèmes pour lesquelles toutes les transactions du système sont AM. C'est-à-dire le nombre de tâches pour lesquelles nous savons que le temps de réponse obtenu par la méthode approchée ou par la méthode mixte sont exactes. Ce critère permet de comparer les taux d'application de la condition suffisante et la condition nécessaire et suffisante.

- *Taux d'amélioration en nombre d'instants critiques (%) :*
  Cette métrique mesure le taux d'amélioration en nombre d'instants critiques de chacune des méthodes implémentées en exploitant la propriété des transactions AM par rapport au celui considéré par la même méthode d'origine. Il évalue le pourcentage d'instants critiques

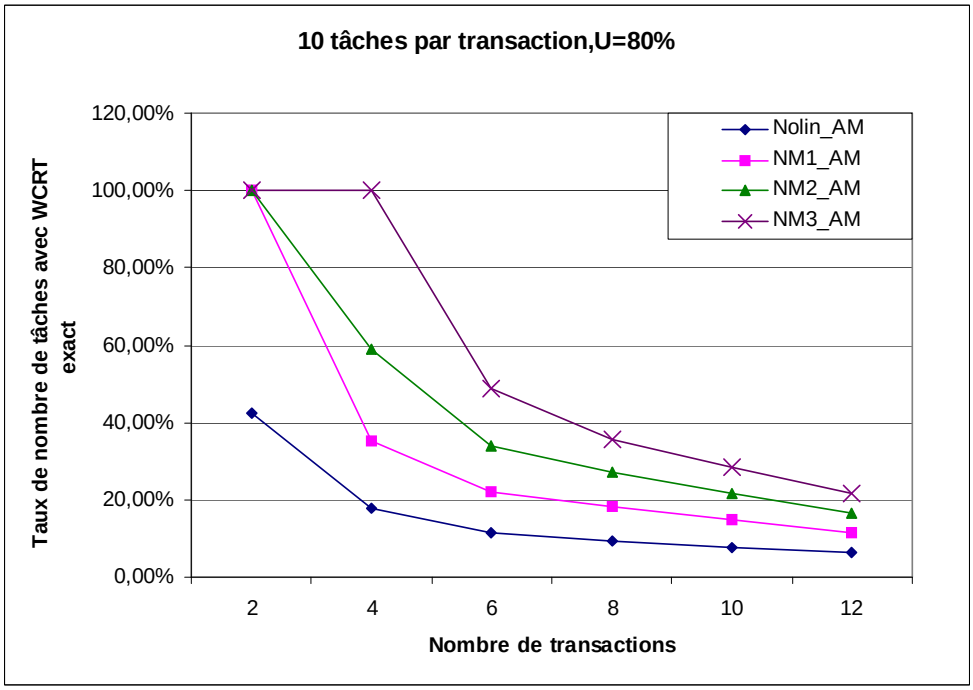

FIG. 4.11 – Influence du nombre de transactions sur le taux de tâches avec WCRT exact.

non considérés par rapport au nombre total d'instants critiques devant être traités par la méthode d'origine. Pour une transaction AM une seule tâche candidate (tâche de pointe) qui initie l'instant critique. Notons Nb_IC^M le nombre d'instants critiques considérés pour une méthode M et $Nb_IC^{M\_AM}$ le nombre considéré par la méthode M en utilisant les transactions AM, le taux d'amélioration est obtenu par :

$$(Nb\_IC^M - Nb\_IC^{M\_AM})/Nb\_IC^M \tag{4.13}$$

– *Amélioration en temps d'exécution (seconde) :*
Cette métrique mesure l'amélioration en temps de traitement apporté par le gain en nombre d'instants critiques dû à l'utilisation de la propriété AM. Elle évalue le temps moyen mis par chacune des méthodes pour analyser un système de transactions.

– *Taux du nombre de transactions AM pour une tâche analysée (%) :*
Cette métrique mesure la proportion du nombre des transactions AM pour une tâche analysée (moins prioritaire) dans un système de transaction quelconque.

4.3.3.2 Résultat de la simulation

Pour toutes les figures présentant les résultats de la simulation, les tests sont faits sur des configurations de tâches à offsets avec une charge processeur totale égale à 80%. Afin de simuler l'influence des tâches sur la performance des méthodes d'analyse, le nombre de transactions est

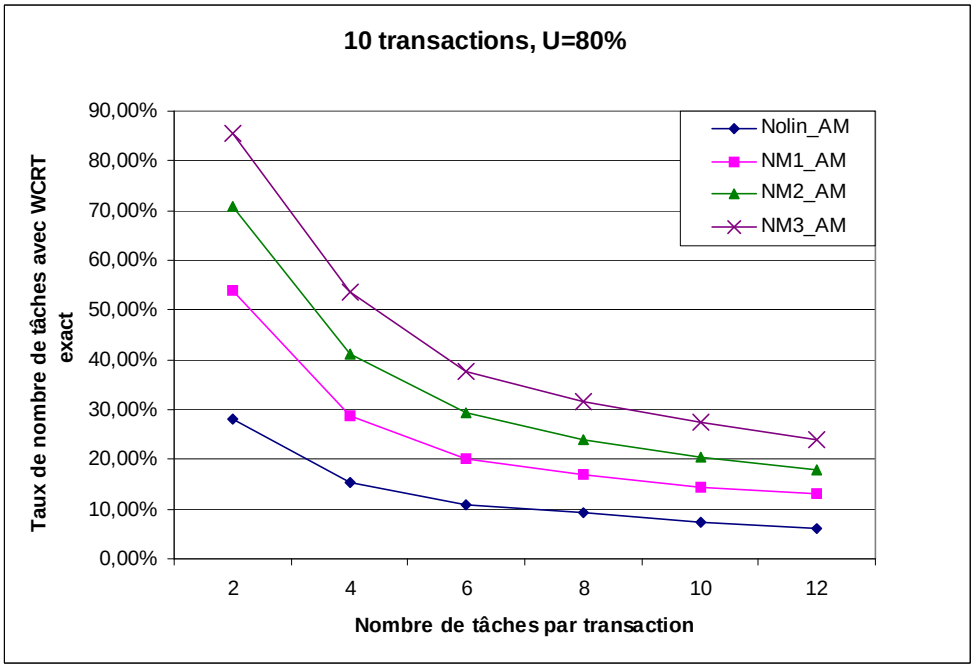

FIG. 4.12 – Influence du nombre de tâches sur le taux de tâches avec WCRT exact.

fixé à 10 transactions et le nombre de tâches varie entre 2 et 12. L'inverse est considérée pour simuler l'influence du nombre de transactions sur la performance des méthodes.

Les figures 4.11-4.12 présentent le taux du nombre des tâches pour lesquelles nous possédons l'information de l'exactitude des temps de réponse fournis par les différentes méthodes d'analyse. Nous constatons une décroissance du taux de tâches avec WCRT exact en augmentant le nombre de transactions (figure 4.11) et en augmentant le nombre de tâches (figure 4.12). Cette décroissance n'est pas continue, une légère stabilité est constatée à partir de 8 transactions (figure 4.11) et à partir de 8 tâches par transactions (4.12). Par exemple, pour un système de 10 transactions de 10 tâches la méthode approchée de Turja-Nolin est exacte pour 9% de tâches, la méthode mixte NM1 est exacte pour 15%, NM2 est exacte pour 20% et NM3 est exacte pour 28%. Notons que le nombre de transactions AM pour une tâche, dans un système, est le même quelle que soit la méthode d'analyse considérée, et que la difference des taux du nombre de tâches avec WCRT exacte entre les méthodes est due au mixage de l'analyse exacte et approchée pour les méthodes mixtes NM1, NM2 et NM3. La condition necessaire et suffisante d'ordonnançabilité pourra etre appliquée sur une quantité importante des tâches.

Les figures 4.13-4.14 montrent le gain moyen en nombre d'instants critiques traités pour la méthode exacte et la méthode mixte. La méthode de Turja-Nolin ne figure pas dans les graphes car elle est complètement approchée et il n'y a pas de notion d'instant critique ou de scénario. Une croissance de taux d'amélioration en nombre d'instants critiques est constatée avec le nombre de transactions. A partir de 8 transactions de 10 tâches on constate une stabilité de

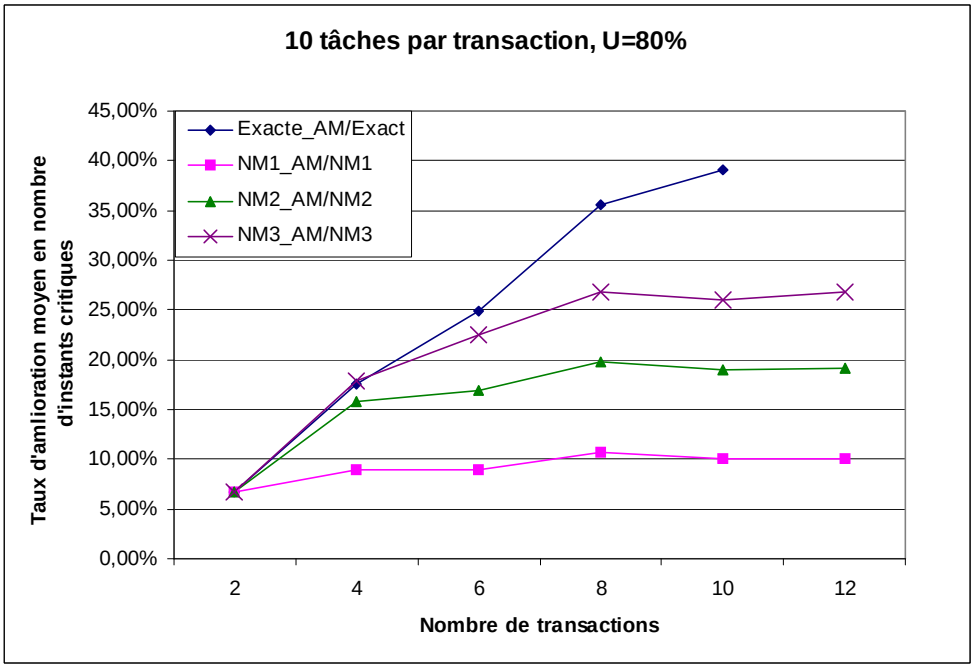

FIG. 4.13 – Influence du nombre de transactions sur le taux d'amélioration moyen en nombre d'instants critiques.

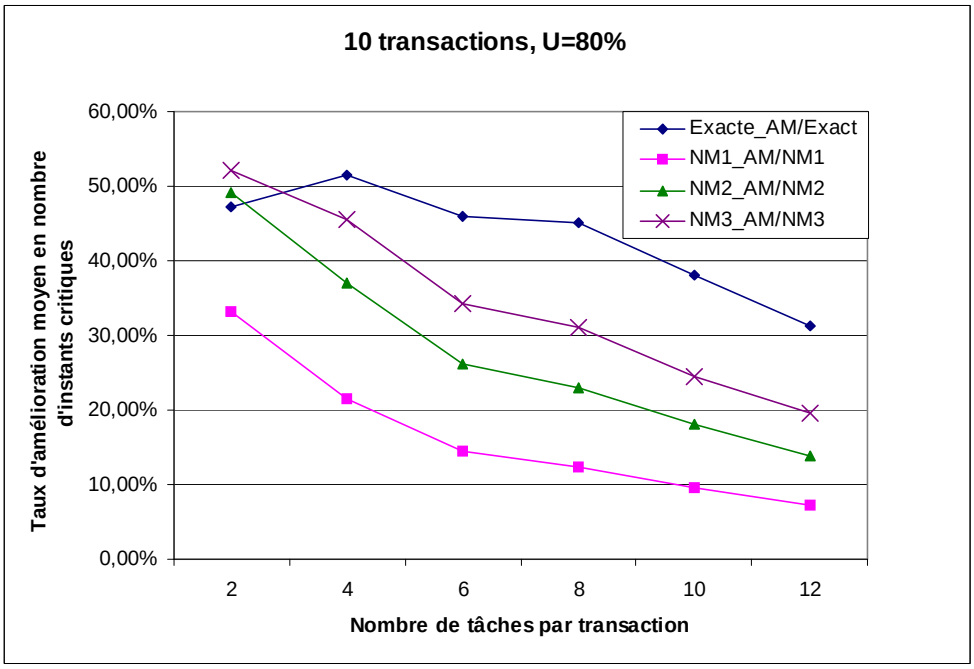

FIG. 4.14 – Influence du nombre de tâches sur le taux d'amélioration moyen en nombre d'instants critiques.

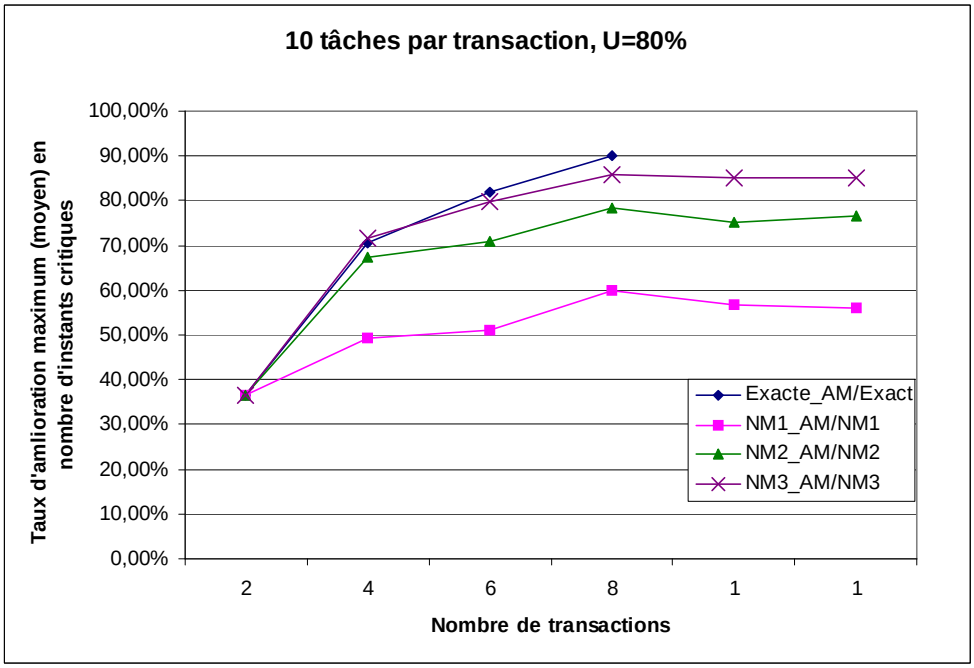

FIG. 4.15 – Influence du nombre de transactions sur le taux d'amélioration maximum en nombre d'instants critiques.

taux d'amélioration vers 10% pour NM1, 20% pour NM2 et 21% pour NM3. Cette stabilité est engendrée par la stabilité de la probabilité d'accumulativité monotonique d'une transaction pour une tâche moins prioritaire (figure 4.19). Tandis que la figure 4.14 montre une décroissance du taux d'amélioration en nombre instants critiques avec le nombre de tâches par transaction. L'augementation du nombre de tâches dans une transaction diminue la probabilité d'être AM pour une tâche moins prioritaire (figure 4.20) ce qui explique cette décroissance de gain en nombre d'instants critiques. Malgré cette décroissance les gains sont à peu près égaux aux taux calculés en fonction du nombre de transactions. Par exemple pour un système de 10 transactions de 10 tâches, nous avons un gain de 10% pour NM1, 18% pour NM2, et 24% pour NM3.

Les figures 4.15-4.16 présentent l'amélioration maximum (moyen) en instants critiques qui pourra être engendrée pour une tâche. Le même comportement de taux moyen est constaté dans ce cas, les taux maximum moyen arrivent jusqu'au 52% pour NM1, 75% pour NM2, 85% pour NM3 et 93% pour la méthode exacte. Il est clair que l'on peut avoir une amélioration de 100% pour les tâches où toutes les transactions sont AM pour elle.

Le gain en nombre d'instants critiques est mis en valeur par l'évaluation et la comparaison des temps nécessaires pour analyser un système par les méthodes d'analyse. Les figures 4.17-4.18 présentent l'évolution de temps mis par les différentes méthode d'analyse en fonction du nombre de transactions et en fonction du nombre de tâches respectivement. Nous constatons que le temps mis par la méthode approchée Nolin est similaire à celui de la même méthode en exploitant la

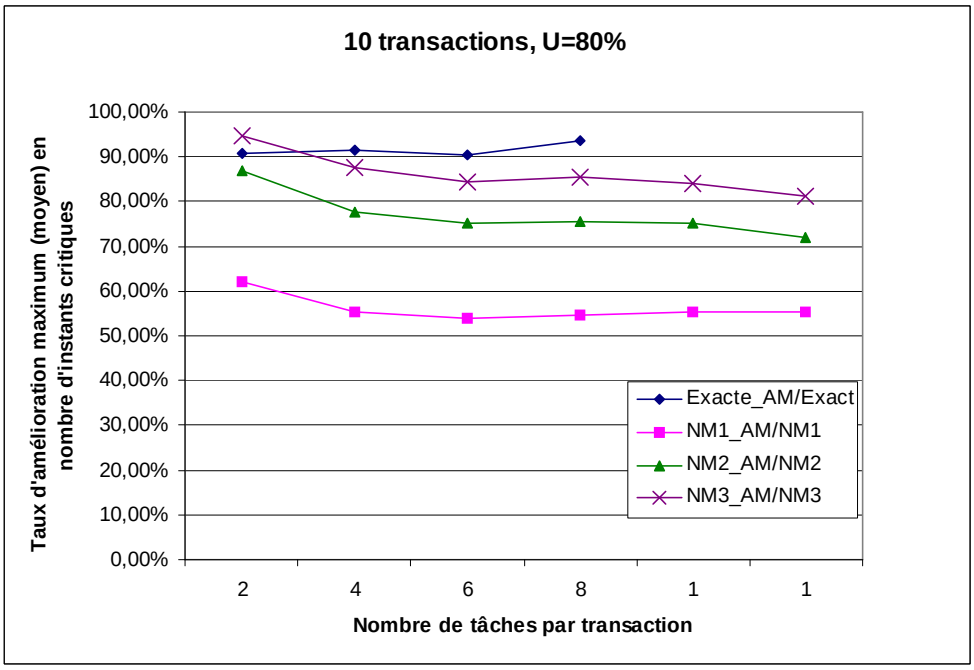

FIG. 4.16 – Influence du nombre de tâches sur le taux d'amélioration maximum en nombre d'instants critiques.

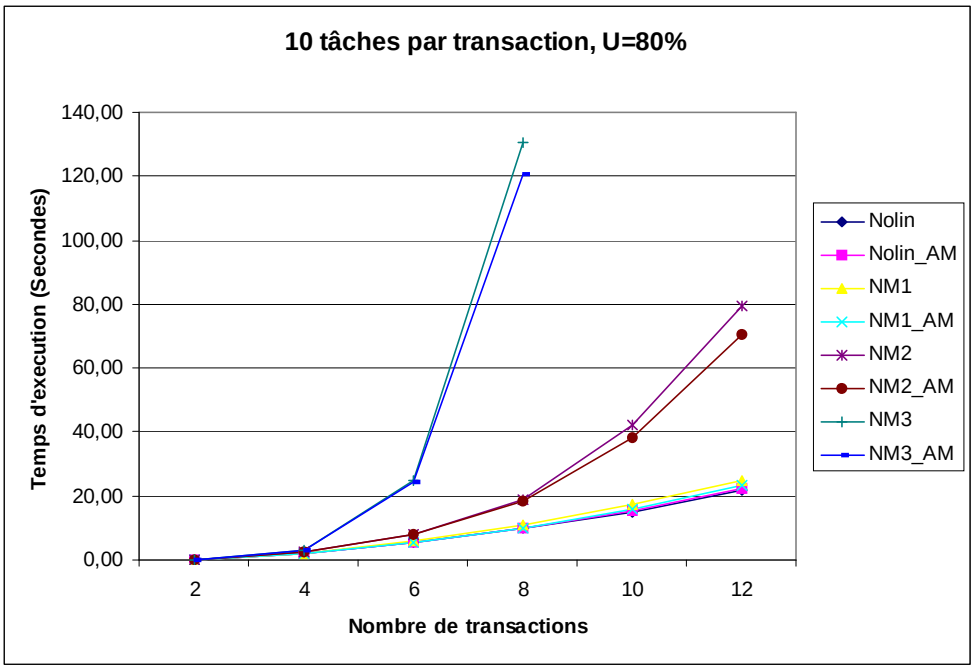

FIG. 4.17 – Influence du nombre de transactions sur le temps d'exécution.

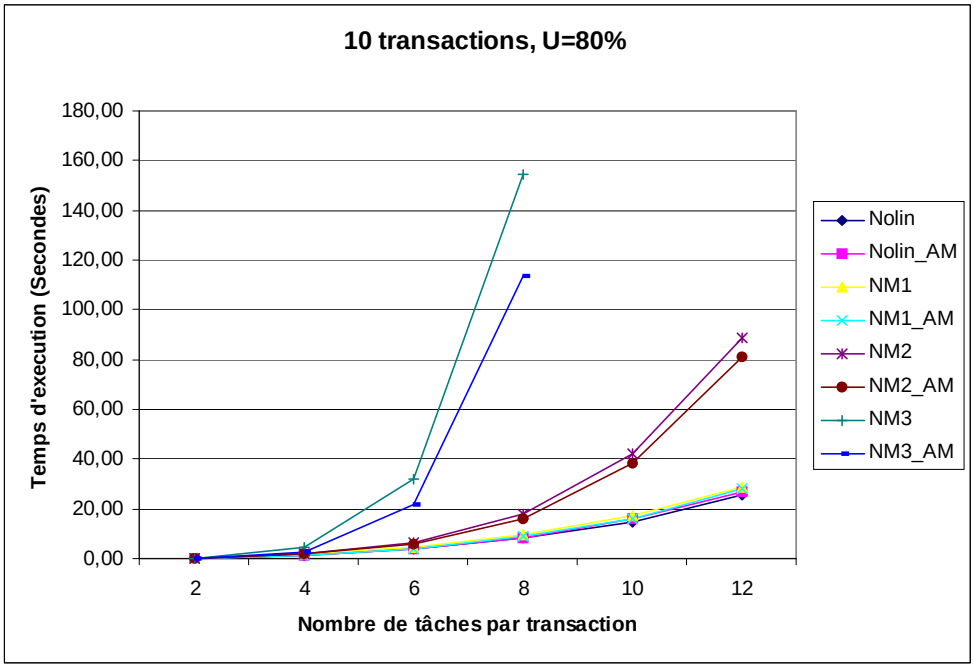

FIG. 4.18 – Influence du nombre de tâches sur le temps d'exécution.

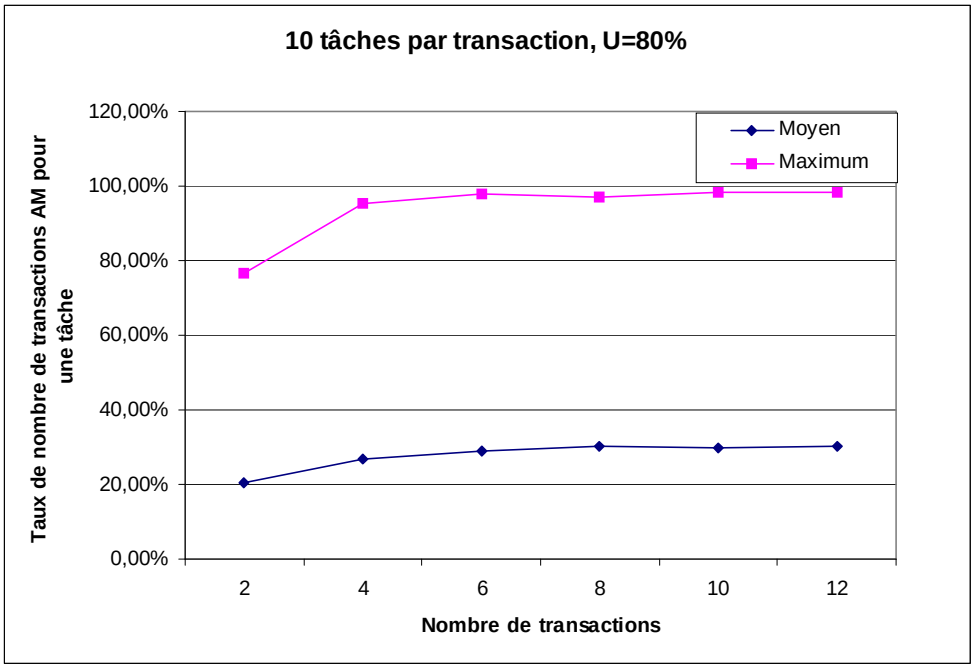

FIG. 4.19 – Le taux de nombre de transactions AM pour une tâche sous analyse.

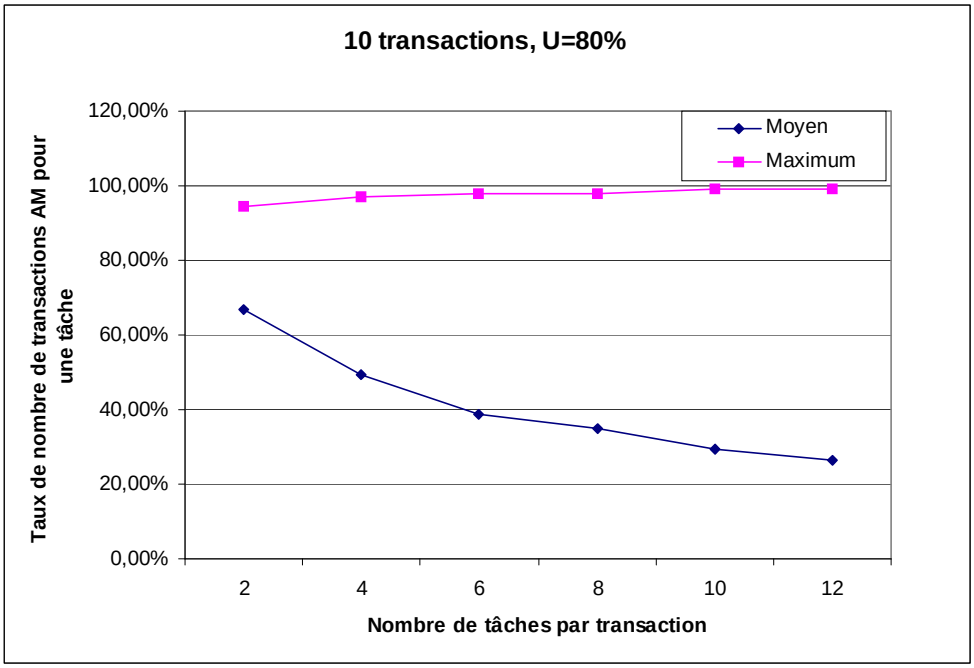

FIG. 4.20 – Influence du nombre de tâches sur le taux du nombre de transactions AM pour une tâche sous analyse.

propriété d'accumulativement monotonique Nolin_AM ce qui confirme l'influence nulle du processus de vérification sur la complexité des méthodes d'analyse. Une accélération de calcul est apportée par la propriété des transactions AM sur la méthode mixte, NM2 et NM3. Par exemple pour analyser un système de 10 transactions de 12 tâche, la méthode NM2_AM a mis 80s au lieu de 90s (NM2). NM3_AM a mis 117s au lieu de 157s (NM3) pour analyser un système de 10 transactions de 8 tâches.

Les figures 4.19-4.20 présentent les taux moyen et maximum du nombre de transactions qui sont AM pour une tâche indépendante moins prioritaire que toutes les tâches de système. Les figures montrent que dans un système de 12 transactions de 10 tâches, le taux moyen (resp. maximum) de transactions AM est de 25% de l'ensemble des transactions de système (resp. 99%).

4.3.4 Bilan

En conclusion de cette section, la propriété des transactions accumulativements monotoniques permet de créer une condition nécessaire et suffisante d'ordonnançabilité sur un nombre important de tâches dans un système analysé. En plus, elle permet d'accélérer le calcul pour les méthodes mixtes et la méthode exacte. Cette amélioration en temps de calcul est due à la connaissance a priori de le tâche initiant l'instant critique produisant le pire cas, pour certaines transactions (transactions AM). La propriété AM est assez fréquente puisque nous la trouvons en moyenne sur 25% des transactions d'un système (figure 4.19) qui peuvent etre AM pour une

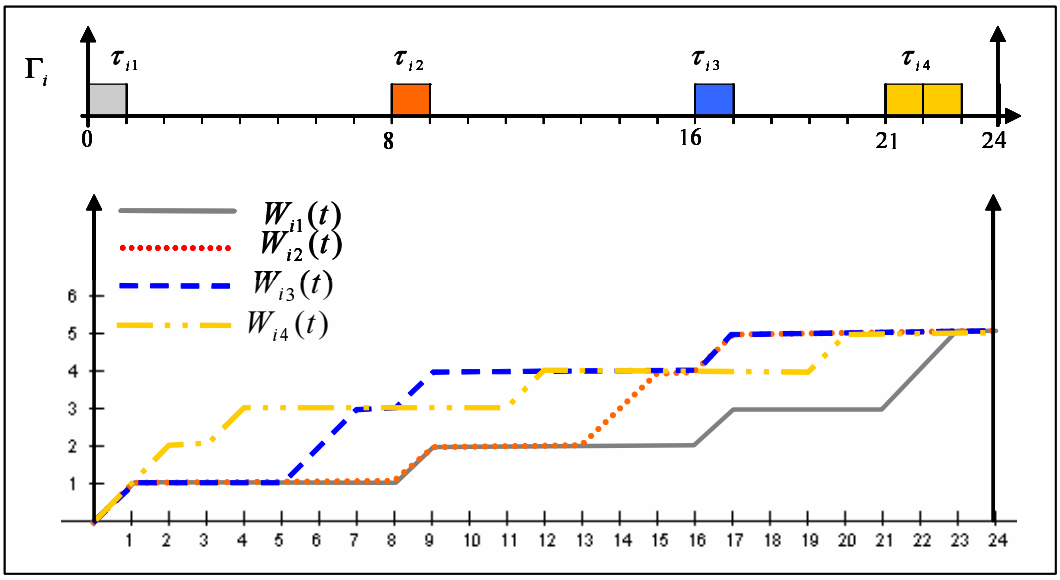

FIG. 4.21 – Exemple d'une transaction contenant des tâches dominées.

tâche moins prioritaire. Afin d'accélerer plus le calcul des méthodes d'analyse, nous proposons dans la section suivante une propriété moins forte et plus générale, celle des tâches dominées.

4.4 Tâches dominées

Contrairement à la propriété des transactions AM, la propriété des tâches dominées consiste à trouver les tâches qui ne peuvent jamais initier l'instant critique produisant le pire cas. En effet, si une transaction est non AM, cela ne signifie pas forcément que toutes ses tâches peuvent participer à la production du pire temps de réponse d'une tâche moins prioritaire. Nous montrons ce concept à travers un exemple.

Soit Γ_i la transaction présentée sur la figure 4.21, et τ_{ua} une tâche à analyser avec une durée d'exécution $C_{ua} = 6$, Nous supposons que toutes les tâches de Γ_i sont plus prioritaires que τ_{ua}. La table 4.5 résume les étapes du calcul itératif du temps de réponse de τ_{ua} par la méthode approchée de Turja-Nolin [70, 61, 63].

Iteration	t	W_{11}	W_{12}	W_{13}	W_{14}	W_i	R_{ua}
0							6
1	6	1	1	2	**3**	**3**	9
2	9	2	2	**4**	3	**4**	10
3	10	2	2	**4**	3	**4**	10

TAB. 4.5 – WCRT approché de τ_{ua} du système de la figure 4.21.

Le temps de réponse de τ_{ua} est de 10 unités de temps (table 4.5). Nous remarquons que la tâche initiant l'instant critique change alternativement entre τ_{i3} et τ_{i4} d'où la transaction Γ_i est non AM. En revanche nous constatons que les tâches τ_{i1} et τ_{i2} n'ont jamais initié l'instant critique. Les courbes d'interférence de Γ_i présentées sur la figure 4.21 montrent bien que l'interférence de Γ_i lorsque τ_{i1} ou τ_{i2} initie l'instant critique ne produit jamais le pire cas (interférence maximum) ; plus précisément l'interférence de Γ_i lorsque τ_{i3} initie l'instant critique est toujours supérieure ou égale à celle produite lorsque τ_{i1} ou τ_{i2} initie l'instant critique.

$$\forall t : W_{i3}(t) \geq W_{i1}(t) \;\; et \;\; W_{i3}(t) \geq W_{i2}(t)$$

Il est évident alors que la considération de la tâche τ_{i1} et τ_{i2} dans la construction du pire scénario est inutile car ces deux tâches n'initient jamais l'instant critique produisant le pire cas. Nous appelons ce type de tâches des tâches dominées.

Dans ce cas, la connaissance a priori des tâches dominées, est très utile pour éliminer tout calcul ou traitement inutile d'un instant critique initié par une de ces tâches. Dans la suite, nous donnons quelques définitions et montrons comment cette classe de tâches pourra être identifiée.

4.4.1 Définitions

Définition 11 ***(Tâche Dominée)***

Une tâche τ_{id} d'une transaction Γ_i est dite tâche dominée par une autre tâche τ_{ip} de la même transaction si et seulement si dans n'importe quelle période d'activité de longueur t, l'interférence de Γ_i lorsque τ_{id} initie l'instant critique est inférieure ou égale à son interférence lorsque la tâche τ_{ip} initie l'instant critique. Formellement τ_{id} est dominée par la tâche τ_{ip} si et seulement si :

$$\forall t : \; W_{id}(\tau_{ua}, t) \leq W_{ip}(\tau_{ua}, t) \tag{4.14}$$

Les tâches τ_{i1} et τ_{i2} de la transaction Γ_i présentées dans la figure 4.21 sont des tâches dominées par la tâche τ_{i3} car, d'après les courbes d'interférence (périodique),

$$\forall t : \; W_{i1}(\tau_{ua}, t) \leq W_{i3}(\tau_{ua}, t) \; et \; W_{i2}(\tau_{ua}, t) \leq W_{i3}(\tau_{ua}, t)$$

Si les interférences d'une transaction pour deux tâches candidates différentes sont égales pour tout intervalle de temps (i.e. une tâche domine l'autre et inversement), alors nous choisissons arbitrairement l'une des deux comme tâche dominée.
Une tâche candidate τ_{ic} d'une transaction Γ_i est dite critique pour une tâche à analyser τ_{ua} si elle n'est dominée par aucune autre tâche de Γ_i. Formellement τ_{ic} est une tâche candidate critique si et seulement si :

$$\forall k \in hp_i(\tau_{ua}), k \neq c \; : \; \exists t : \; W_{ic}(\tau_{ua}, t) > W_{ik}(\tau_{ua}, t) \tag{4.15}$$

Les tâches τ_{i3} et τ_{i4} de la transaction Γ_i présentées dans la figure 4.21 sont des tâches candidates critiques car elles ne sont pas dominées par d'autres tâches de Γ_i.

Il est évident d'après la définition d'une tâche dominée que cette dernière ne pourra jamais être une tâche candidate critique. Autrement dit, elle ne pourra jamais initier l'instant critique produisant le pire cas puisque'elle ne crée jamais une interférence maximum de la transaction à laquelle elle appartient.

Soit Γ un système de n transactions Γ_i $(i = 1..n)$. L'analyse exacte de temps de réponse d'une tâche moins prioritaire que toutes les tâches du système nécessite l'examen de $|\Gamma_1| \times |\Gamma_2| \times ... \times |\Gamma_n| = |\Gamma_i|^n$ instants critiques. Supposons qu'une tâche soit dominée dans une transaction quelconque (par exemple Γ_1), le nombre d'instants critiques éliminés est égal à $|\Gamma_2| \times ... \times |\Gamma_n| = |\Gamma_i|^{n-1}$. Et par conséquent, le nombre d'instants critiques à considérer est égal alors à $(|\Gamma_1| - 1) \times |\Gamma_2| \times ... \times |\Gamma_n|$.

Notons que le concept des tâches dominées est plus général que le concept des transactions AM tel qu'une transaction Accumulativement monotonique est un cas particulier des tâches dominées, et elle peut être définie en fonction des tâches dominées comme suit.

Définition 12 ***(Transaction AM)***

Une transaction Γ_i est accumulativemet monotonique pour une tâche analysée τ_{ua} s'il existe une tâche τ_{ip} qui domine toutes les autres tâches de Γ_i (toutes les tâches de Γ_i sont dominées par une seule tâche τ_{ip}).

4.4.2 Procédure de détection d'une tâche dominée

Afin de mettre en œuvre le concept des tâches dominées, nous proposons un procédure qui détecte si une tâche τ_{id} est dominée par une autre tâche τ_{ip}. L'idée est la même que celle proposée pour les transactions AM. Il suffit de comparer les points convexes des courbes d'interférence correspondant à chaque paire de tâches candidates. Puisque l'interférence effective d'une transaction est périodique [62, 63], la fonction d'interférence durant la première période peut être différente de celles des autres périodes, alors la vérification de dominance (comparaison d'interférence) est réduite aux deux premières périodes.

Une fois, les points décrivant l'interférence d'une transaction Γ_i stockés dans les table P_{ic} et P'_{ic} pour chacune de ses tâches candidates τ_{ic}, la vérification de la dominance est possible entre toute paire de tâches candidates. Une tâche τ_{id} est dominée par une tâche τ_{ip} si et seulement si tous les points de P_{id} sont « Enveloppés » par les points de P_{ip}, et tous les points de P'_{id} sont « Enveloppés » aussi par les points de P'_{ip}.

Formellement, une tâche τ_{id} est dominée par une tâche τ_{ip} Ssi :

$$\forall b = 1..|P_{id}| : \ \exists a \in [1..|P_{ip}|] : \ P_{ip}[a] \succ P_{id}[b] \ et$$

$$\forall b = 1..\left|P'_{id}\right| : \ \exists a \in \left[1..\left|P'_{ip}\right|\right] : \ P'_{ip}[a] \succ P'_{id}[b]$$

Dans notre exemple de la figure 4.21, les tables stockant l'interférence de Γ_i pour τ_{i1}, τ_{i2} et τ_{i3} sont :

$$P_{i1} = P'_{i1} =< (1,1,1),(9,2,1),(17,3,1),(23,5,1) >$$
$$P_{i2} = P'_{i2} =< (1,1,2),(9,2,2),(15,4,2),(17,5,2) >$$
$$P_{i3} = P'_{i3} =< (1,1,3),(7,3,2),(9,4,2),(17,5,2) >$$

Les courbes d'interférence W_{i1} et W_{i2} présentées sur la figure 4.21 montrent bien que tous les points de P_{i1} (coins convexes de la courbe W_{i1} et de W_{i2}) sont tous enveloppés par les points de P_{i3} (coins convexes de la courbe W_{i3}) alors τ_{i3} domine τ_{i1} et domine aussi τ_{i2} (on peut voir graphiquement la transitivité de la relation Envelopper).

L'identification des tâches candidates critiques (pouvant initier l'instant critique produisant le pire cas) consiste alors à éliminer toutes les tâches dominées de l'ensemble de tâches créant l'instant critique. En effet, une tâche candidate est critique si elle n'est pas dominée par une autre tâche de la même transaction. Le processus de test de la dominance entre les tâches d'une même transaction est injecté dans le processus de représentation statique d'interférence, adapté dans nos travaux faits sur les transactions AM. A la fin du calcul de l'ensemble des tables P_{ic} et P'_{ic}, chaque paire de tâches est vérifiée pour voir si l'une domine l'autre. Les tâches dominées sont exclues de l'ensemble des tâches critiques.

Il est évident que la relation de dominance est transitive, en effet, si une tâche τ_{ik} domine une tâche τ_{il}, et τ_{il} à son tour domine une autre tâche τ_{im} alors la tâche τ_{ik} domine forcement la tâche τ_{im} ; cette propriétés permet d'éviter quelques tests de dominance inutiles. Par exemple, dans la figure 4.21 la tâche τ_{i2} domine τ_{i1} et τ_{i3} domine τ_{i2}, alors forcément τ_{i3} domine τ_{i1}.

Au niveau de la complexité des méthodes d'analyse, la recherche des tâches dominées ne provoque pas une augmentation de la complexité de calcul, car elle repose sur la représentation statique d'interférence. Après avoir stocké toutes les informations sur l'interférence d'une transaction, un test de dominance est effectué. Il y a donc $O(n^2)$ comparaisons, où n est le nombre de tâches dans la transaction considérée. L'algorithme d'implémentation est présenté dans l'annexe B.4.

4.4.3 Exemple

A travers cet exemple nous montrons le gain en nombre d'instants critiques apporté par l'élimination des instants critiques initiés par des tâches dominées. Nous considérons que toutes les tâches du système, présentées sur la figure 4.22, sont plus prioritaires que la tâche analysée τ_{ua}. Les transactions Γ_1 et Γ_2 sont AM pour τ_{ua} avec la tâche de pointe τ_{11} et τ_{21} respectivement (τ_{11} (resp τ_{21}) domine toutes les tâches de Γ_1 (resp Γ_2)). La tâche τ_{42} de la transaction Γ_4 est dominée par la tâche τ_{41}, d'où Γ_4 ne contient que trois tâches candidates critiques τ_{41} ,τ_{43} et τ_{44} (la tâche τ_{42} n'initie jamais le pire instant critique).

Notons $|\Gamma_i|^*$ le nombre de tâches candidates critiques appartenant à Γ_i, il est clair que $|\Gamma_i|^*$ est toujours inférieur ou égal au nombre de tâches candidates de Γ_i $|\Gamma_i|$. Le nombre d'instants critiques pouvant produire le pire cas dépend alors de $|\Gamma_i|^*$ au lieu de $|\Gamma_i|$.

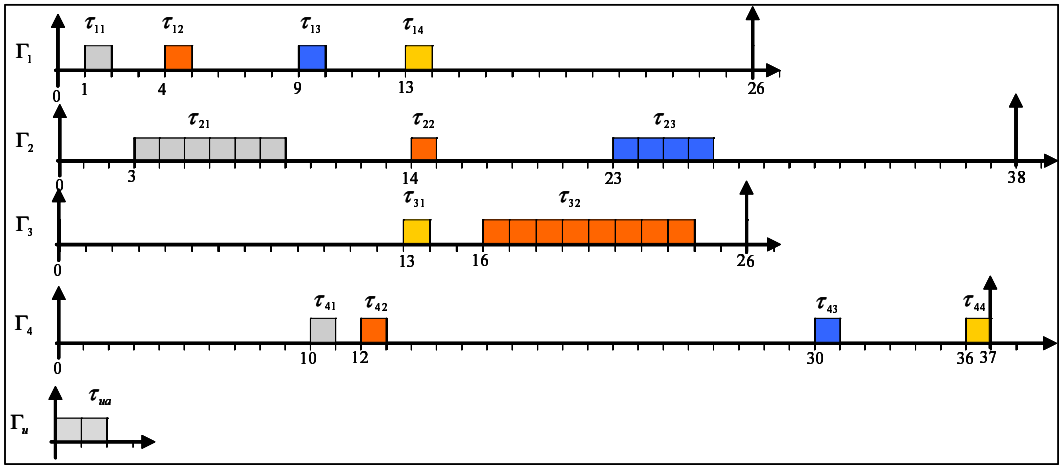

FIG. 4.22 – Exemple d'un système de transactions avec des tâches dominées.

Scénario Γ_i	tâche candidate τ_{ic}	$R_{ua,ic}$	$R_{ua,i}$	R_{ua}
Γ_1	Transaction AM			70
Γ_2	Transaction AM			
Γ_3	τ_{31}	55	70	
	τ_{32}	70		
Γ_3	τ_{41}	52	71	
	τ_{42}	Dominée		
	τ_{43}	52		
	τ_{44}	71		

TAB. 4.6 – Calcul de WCRT de τ_{ua} par la méthode mixte avec $E = 1$.

Afin de calculer le WCRT de τ_{ua} avec la méthode exacte le nombre d'instants critiques à traiter est $|\Gamma_1| \times |\Gamma_2| \times |\Gamma_3| \times |\Gamma_4| = 4 \times 3 \times 2 \times 4 = 96$, Ce nombre est réduit, en éliminant les instants initiées par des tâches dominées, à $|\Gamma_3| \times |\Gamma_4|^* = 2 \times 3 = 6$ instants critiques (une amélioration de 93%). Pour la méthode mixte avec le nombre de transactions $E = 1$ pour lesquelles un calcul exact d'interference est appliqué $E = 1$, le nombre d'instants critiques est reduit de $|\Gamma_1| + |\Gamma_2| + |\Gamma_3| + |\Gamma_4| = 4 + 3 + 2 + 4 = 13$ à $|\Gamma_3|^* + |\Gamma_4|^* = 2 + 3 = 5$ (amélioration de 61%). L'analyse mixte avec $E = 2$ pour la methode mixte le nombre d'instants critiques est de 6 au lieu de 62. Notons que le temps de réponse de τ_{ua} obtenu par les methodes mixte est égal à sa valeur exacte $R_{ua} = 70$ (table 4.6), en revanche la borne obtenue par la methode approchée Turja-Nolin est pessimiste ($R_{ua} = 71$).

4.4.4 Expérimentation

Nous évaluons, dans cette section, les améliorations, en nombre d'instants critiques et surtout en temps de calcul mis par les méthodes d'analyse, apportées par l'exploitation de la propriété de la dominance. Comme pour les transactions AM, nous avons implémenté d'une part les algorithmes d'analyse d'origine suivantes :

- Nolin : La méthode approximative de Nolin [61, 62, 63].
- NM1 : La méthode mixte [81, 82] avec un nombre de transactions $E = 1$ pour lesquelles nous appliquons une fonction d'interférence exacte.
- NM2 : La méthode mixte avec $E = 2$.
- NM2 : La méthode mixte avec $E = 3$.
- Exact : La méthode exacte [107, 70].

Et d'autre part les mêmes méthodes précédentes mais avec l'exploitation de la propriété des tâches dominées :

- Nolin_DOM : La méthode approximative de Nolin [62] en exploitant la propriété tâches dominées.
- NM1_DOM : La méthode mixte [81, 82] avec $E = 1$, en exploitant la propriété des tâches dominées.
- NM2_DOM : La méthode mixte avec $E = 2$ en exploitant la propriété tâches dominées.
- NM2_DOM : La méthode mixte avec $E = 3$ en exploitant la propriété tâches dominées.
- Exact_DOM : La méthode exacte [107, 70] en exploitant la propriété tâches dominées.

Les tests de simulation sont faits sur des systèmes de transactions générés aléatoirement par le générateur de tâches présenté dans la section 3.6.1 du chapitre précédent.

Dans un premier temps nous évaluons le taux de nombre de tâches dominées dans une transaction en fonction du nombre de tâches. Ensuite nous simulons l'influence du nombre de transactions et du nombre de tâches par transaction sur : (1) le taux d'amélioration en nombre d'instants critiques à analyser grâce à l'élimination de ceux initiés par des tâches candidates dominées, (2) l'améliorations en temps de calcul induite par l'élimination des instants critiques dominés. Ces deux dernières métriques sont décrites dans la section 4.3.3.1 des critères d'évaluation pour les transactions AM. Ces tests sont effectués pour chacune des méthodes implémentées afin de

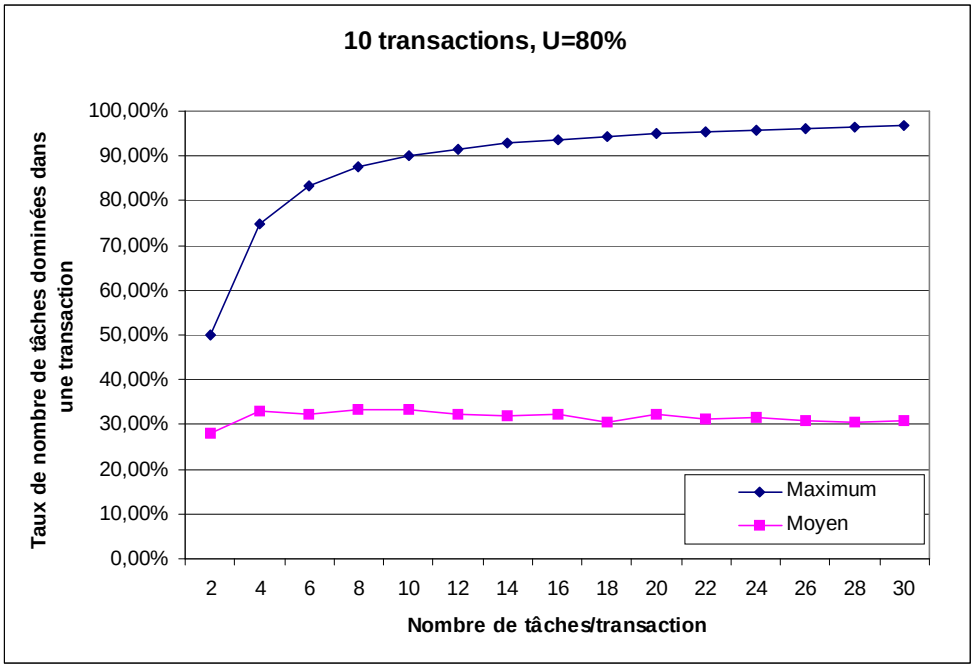

FIG. 4.23 – Influence du nombre de tâches sur le taux du nombre de tâches dominées dans une transaction.

faciliter la comparaison.

4.4.4.1 Évaluation du taux de nombre de tâches dominées dans une transaction

Cette métrique mesure le taux du nombre de tâches candidates qui sont dominées parmi le nombre total des tâches candidates d'une transaction quelconque. Soit une transaction Γ_i contenant N tâches candidates dont D sont dominées, alors le taux de nombre de tâches dominées dans cette transaction est donné par :

$$(D/N)$$

La figure 4.23 présente le taux de nombre de tâches dominées moyen et maximum (en moyenne) d'une transaction dans un système quelconque. Notons que dans ce graphe, chaque point correspond à la moyenne des résultats de 100 systèmes de 10 transactions. Nous constatons que l'augmentation du nombre de tâches dans une transaction n'a pas une influence remarquable sur le nombre de tâches dominées. Le nombre moyen de tâches dominées dans une transaction de k tâches ($k = 2..30$) est environ de 32% du nombre total de tâches candidates. Ce taux est important vu le nombre d'instants critiques pouvant etre évités (figures 4.24-4.25). Le taux maximum (en moyenne) de tâches dominées est éstimé à environ 96% du nombre total de tâches d'une transaction. Cette valeur réprésente la moyenne du nombre maximum de tâches dominées d'une transaction dans un système.

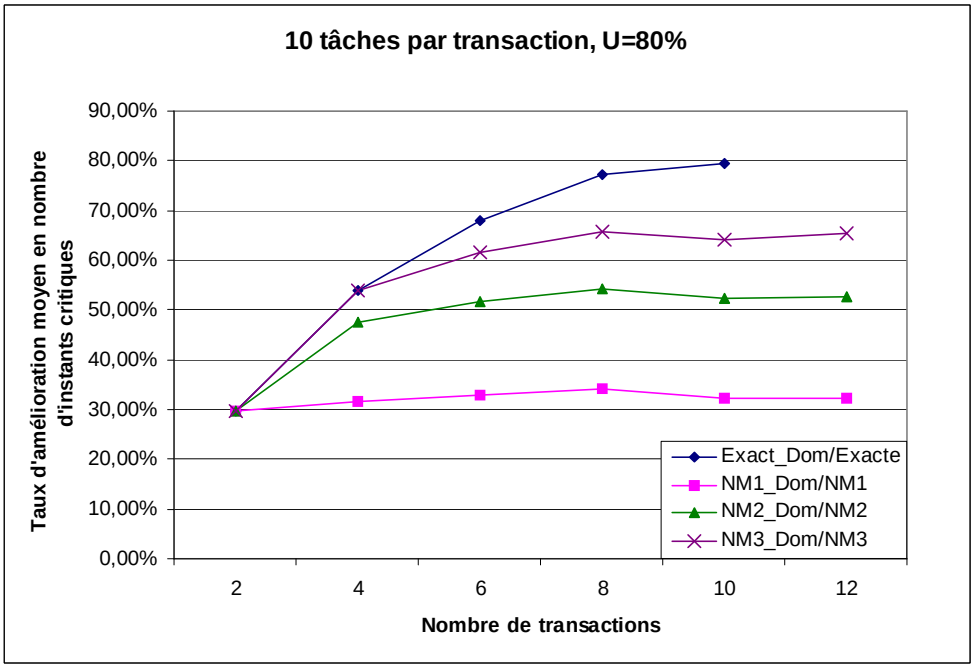

FIG. 4.24 – Influence du nombre de transactions sur le taux d'amélioration moyen en nombre d'instants critiques.

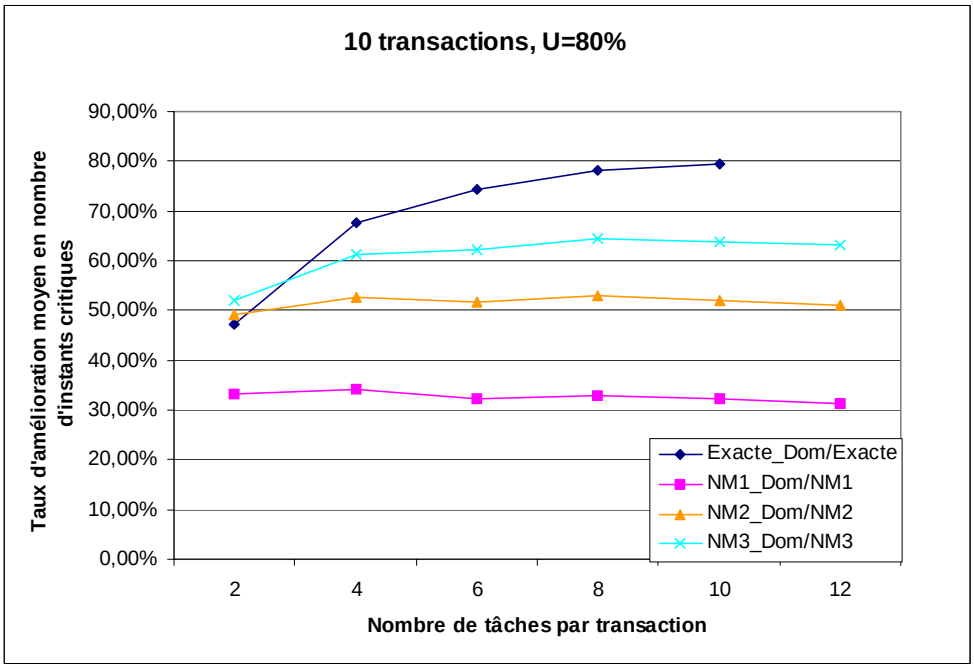

FIG. 4.25 – Influence du nombre de tâches sur le taux d'amélioration moyen en nombre d'instants critiques.

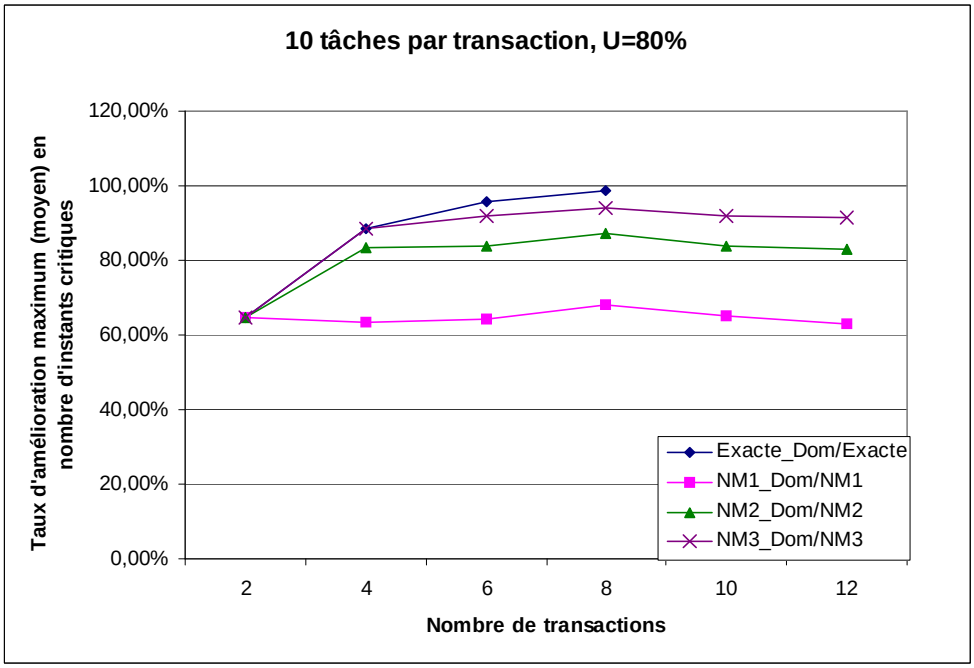

FIG. 4.26 – Influence du nombre de transactions sur le taux d'amélioration maximum en nombre d'instants critiques.

4.4.4.2 Évaluation de l'amélioration en nombre d'instants critiques

Cette métrique représente le taux d'instants critiques éliminés du calcul. Pour une transaction, les tâches candidates dominées ne seront pas considérées dans la création des instants critiques possibles. Notons Nb_IC^M le nombre d'instants critiques considérés pour une méthode M et $Nb_IC^{M\_DOM}$ le nombre considéré par la méthode M en éliminant les instants critiques initiés par des tâches dominées. Le taux d'amélioration est obtenu par :

$$(Nb\_IC^M - Nb\_IC^{M\_DOM})/Nb\_IC^M \tag{4.16}$$

Les figures 4.24-4.25 (resp, figures 4.26-4.27) montrent une forte amélioration moyenne (resp, maximum) en nombre d'instants critiques considérés pour les différentes méthodes d'analyse exacte et mixte. Les figures 4.24-4.26 présentent le comportement de cette amélioration en fonction de la variation du nombre de transactions par système. Tandis que les figures 4.25-4.27 présentent l'influence du nombre de tâches par transaction sur ces taux d'amélioration. L'amélioration moyenne comme l'amélioration maximale sont stables et ne montrent aucune sensibilité aux variations du nombre de tâches et du nombre de transactions. Cette stabilité est expliquée par le pourcentage stable de nombre de tâches dominées par transactions (figure 4.23). Pour la méthode mixte le taux d'amélioration est meilleur en augmentant le nombre de transactions E sur lesquelles une analyse exacte est appliquée. Nous retrouvons une amélioration moyenne vers

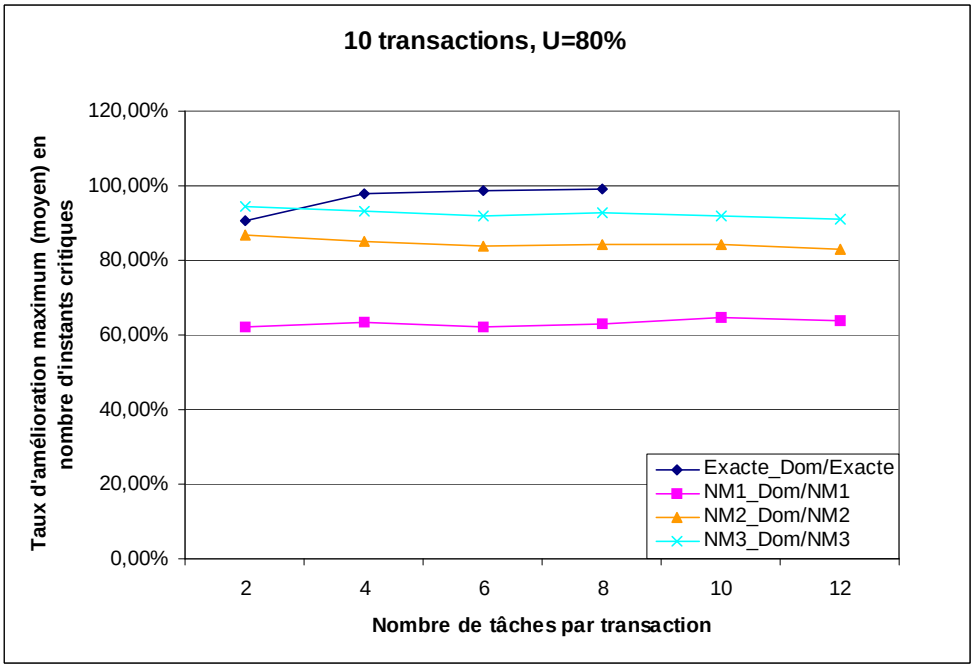

FIG. 4.27 – Influence du nombre de tâches sur le taux d'amélioration maximum en nombre d'instants critiques.

32% d'instants citiques pour NM1, 51% pour NM2, 62% pour NM3 et une amélioration moyenne de 80% pour la méthode exacte. Tandis que l'amélioration maximum peut arriver jusqu'à 62% pour NM1, 82% pour NM2, 90% pour NM3 et 98% pour la méthode exacte.

4.4.4.3 Évaluation du temps d'exécution

Les figures 4.28-4.29 présentent le temps d'exécution, pour analyser un système, mis par les différentes méthodes implémentées, sans et avec la propriété de la dominance des tâches. Nous constatons que les méthodes mixtes NM2_Dom et NM3_Dom (utilisant la propriété de la dominance) sont plus rapides que celles d'origine NM1 et NM2. L'écart entre le temps mis par les méthodes utilisant la dominance de tâches et le temps mis par les méthodes d'origine correspondantes croît avec le nombre de tâches et le nombre de transactions. La figure 4.28 montre, par exemple, que pour analyser un système de 12 transactions de 10 tâches, la méthode NM2 met 80 secondes, par contre, en utilisant les tâches dominées, NM2_Dom ne met que 50 secondes, ce qui représente un gain de 30 secondes (37%). L'accélration de la méthode NM3 est trés importante où pour un système de 8 transactions de 10 tâches la méthode met 42s (NM3_Dom) au lieu de mettre 155s (NM3). Cette accélération de calcul est bien meilleure que celle induite par la propriété des transactions AM présenté dans les figures 4.17-4.18 (section 4.3.3.2) et confirme le résultat obtenu sur le gain en nombre d'instants critiques.

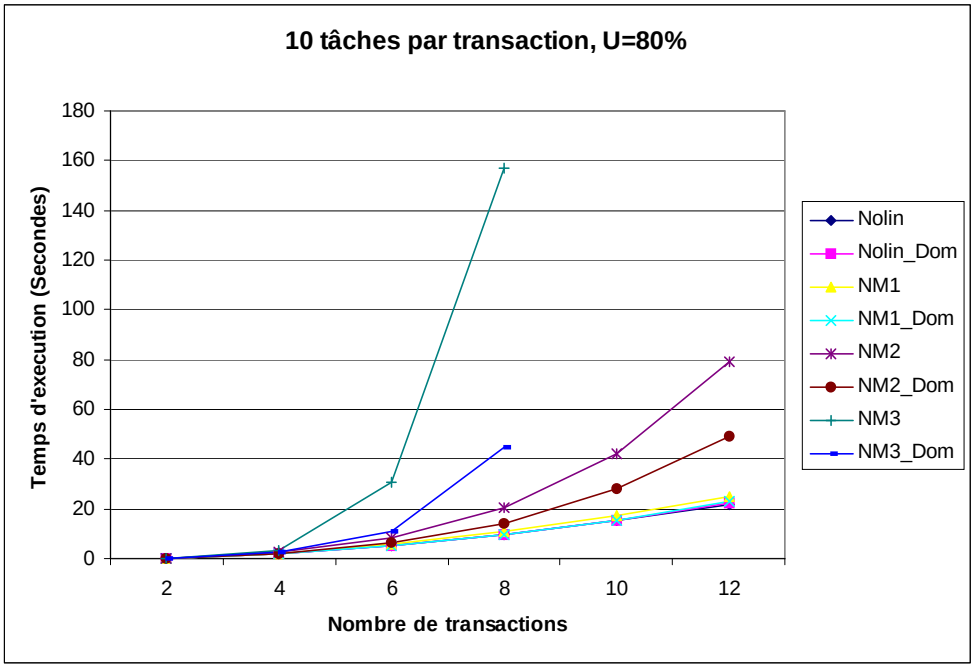

FIG. 4.28 – Influence du nombre de transactions sur le temps d'exécution.

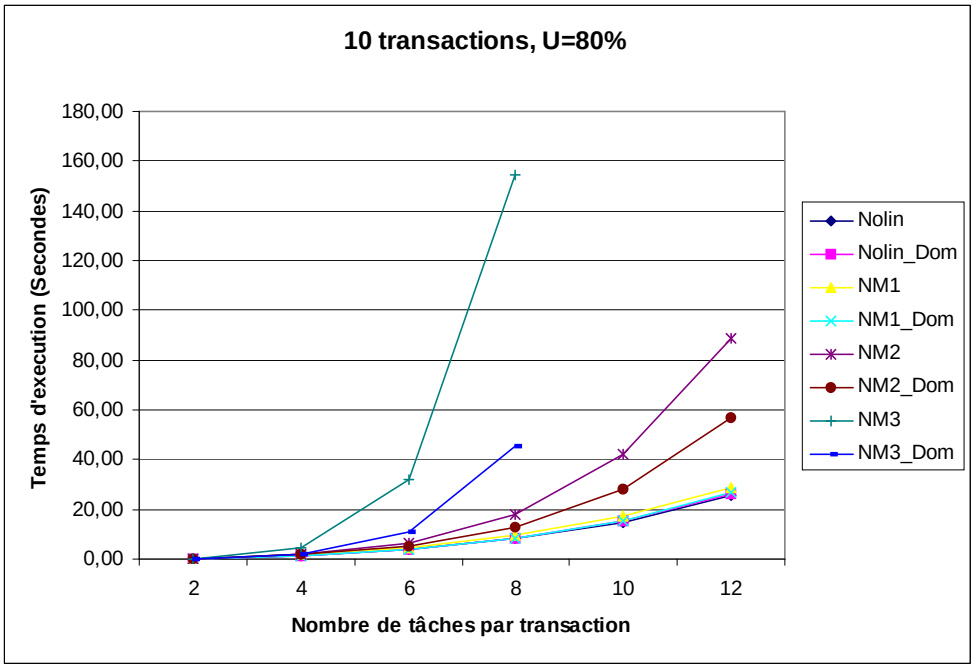

FIG. 4.29 – Influence du nombre de tâches sur le temps d'exécution.

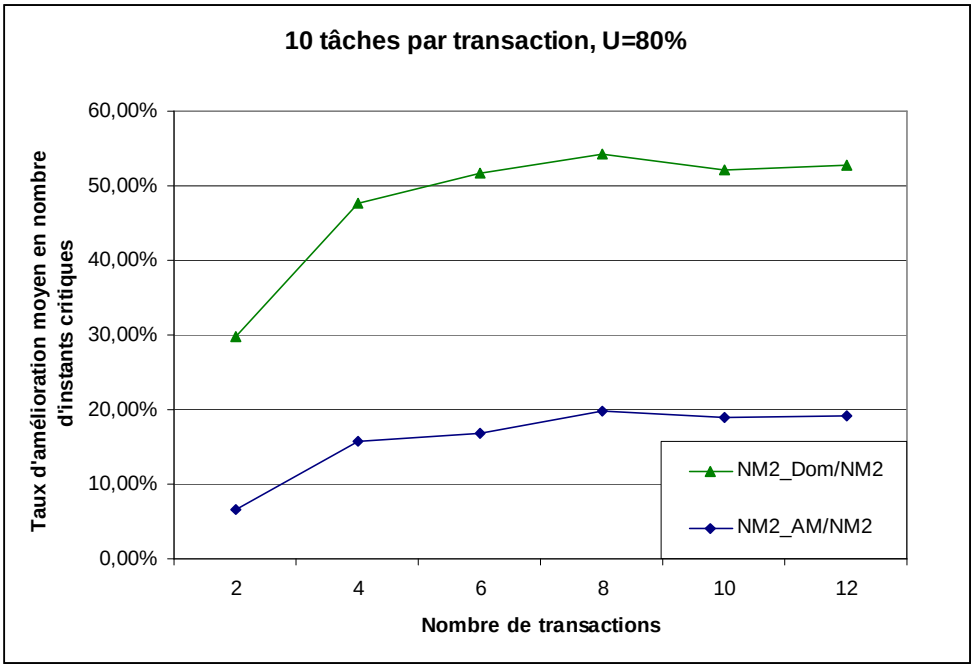

FIG. 4.30 – Taux moyen d'amélioration en nombre d'instants critiques.

4.5 Bilan

La propriété des transactions AM est un cas particulier de la propriété des tâches dominées. Une transaction AM est une transaction qui contient une tâche de pointe dominant toutes autres tâches de la transaction. Notons que les deux propriétés mènent au même taux d'information supplémentaire sur l'exactitude des temps de réponse (condition nécessaire et suffisante) car cette information est basée essentiellement sur la propriété d'AM. Par contre l'amélioration en nombre d'instants critiques et en temps d'exécution sont basées essentiellement sur la propriétés de dominance. Afin de comparer les performances de ces deux propriétés, les figures 4.30 et 4.31 regroupent les deux résultats obtenus pour la méthode mixte NM2, pour les deux propriétés, sur le nombre d'instants critiques et sur le temps d'exécution respectivement.

Nous constatons que la propriété de dominance des tâches est toujours meilleure, en terme d'instants critiques, que celle d'AM des transactions. Un écart constant de 30% d'instants critiques est enregistré entre les deux méthodes. Cet écart est dû au nombre important de tâches dominées dans les transactions qui ne satisfont pas la propriété AM. Par consequent le temps d'execution de la méthode est significativement moins important en utilisant la propriété de dominance que celui utilisant la propriété d'AM.

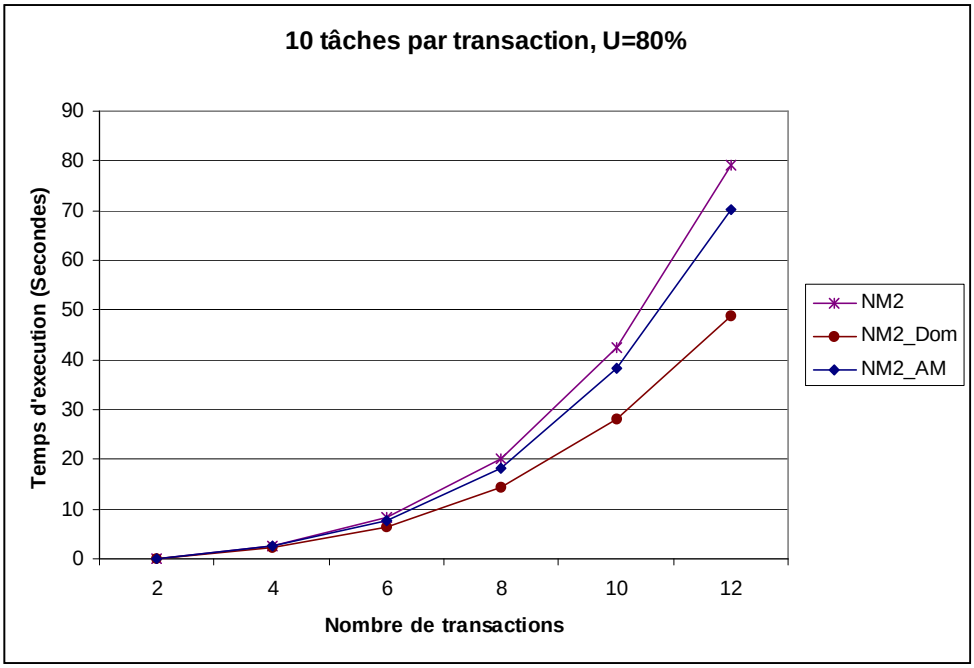

FIG. 4.31 – Comparaison des temps d'exécution.

4.6 Conclusion

Dans l'analyse d'ordonnançabilité présentée précédemment toute tâche plus prioritaire que la tâche analysée est considérée dans la construction de l'instant critique pire cas. Nous avons présenté, dans ce chapitre, quelques propriétés pour les transactions comme pour les tâches permettant d'identifier les tâches candidates pouvant initier l'instant critique pire cas.

Dans un premier temps, nous avons étudié la propriété de monotonicité pour des transactions avec gigues nulles. Une transaction satisfaisant la condition de monotonicité est dite monotonique et sa tâche candidate initiant l'instant critique pire cas est connue, cette tâche est appelée tâche de pointe. Une information sur l'exactitude de temps de réponse est obtenue lorsque toutes les transactions du système sont monotoniques pour la tâche analysée. Des tests ont été effectués et ont montré que l'injection du test de monotonicité dans l'analyse n'a aucune influence sur le temps de calcul de la méthode approchée de Turja-Nolin. Le temps de calcul peut être significativement optimisé dans le cas où toutes les transactions sont monotoniques. Plusieurs lacunes ont été énumérées comme l'absence de gigue et la non détection de nombreux cas de transactions ayant une seule tâche candidate initiant l'instant critique pire cas.

Afin de remédier à ces lacunes, nous avons présenté la propriété des transactions accumulativement monotoniques (AM), cette propriété est plus générale que la propriété de monotonicité. Toute transaction monotonique est forcément AM. Une procédure de vérification de l'AM des transactions a été proposée, elle est basée essentiellement sur la périodicité de l'interférence et sa représentation statique. Les points essentiels conclus lors des tests de performance effectués

sont les suivants :

- Un nombre important de tâches, dans un système, ont un temps de réponse approché exacte avec une méthode approchée. Cette information transforme la condition suffisante d'ordonnançabilité à une condition nécessaire et suffisante pour ces tâches. Près de 20% des tâches d'un système ont un temps de réponse exacte donné par la méthode mixte NM2, cette méthode a été adoptée dans le chapitre précédent.
- La connaissance des tâches de pointe des transactions AM permet d'éviter le traitement d'un nombre important d'instants critiques candidats. Nous trouvons une amélioration de 25% d'instants critiques pour la méthode NM2.
- Le gain en nombre d'instants critiques conduit à une accélération du temps de calcul surtout pour les méthodes mixtes NM2 et NM3.

Afin d'accélérer encore plus les méthodes d'analyse, nous avons présenté la propriété de dominance, cette fois-ci pour les tâches. Elle permet de détecter les tâches candidates ne pouvant jamais initier l'instant critique pire cas. Cette propriété est plus générale que les deux propriétés présentés précédemment (monotonicité et AM) où une transaction AM ou monotonique est une transaction dans laquelle toutes les tâches sont dominées par une seule tâche de pointe. Les tests effectués ont montré que cette propriété a apporté une amélioration, en nombre d'instants critiques, plus importante que celle apportée par la propriété d'AM des transactions. Un taux d'amélioration moyen de 65% d'instants critiques est enregistré pour la méthode mixte NM2. Ainsi une accélération considérable de calcul est enregistrée pour les méthodes d'analyse mixte NM2 et NM3.

La propriété de dominance de tâches (englobant la propriété de monotonicité et d'AM des transactions), soutient fortement l'application de la méthode d'analyse mixte NM2, vue l'amélioration apportée en temps de calcul plus le taux d'application de la condition nécessaire et suffisante.

En perspectives, nous pensons pouvoir améliorer la qualité de l'analyse d'ordonnançabilité, en fixant un nombre seuil d'instant critique pour appliquer chacune des méthodes d'analyse mixte et exacte. En effet, pour une tâche analysée, en fonction du nombre d'instants critiques à considérer, après avoir éliminé tous les instant initiés par des tâches dominées, choisir la méthode d'analyse exacte ou la méthode d'analyse mixte (NM2, NM3, ...). Cela pourra maîtriser le temps de traitement, tout en améliorant la qualité du test d'ordonnançabilité.

Chapitre 5

Analyse d'ordonnançabilité des transactions avec priorités dynamiques EDF

Résumé : *Dans ce chapitre, nous adressons l'ordonnancement avec EDF des transactions temps réel. En nous basant sur l'analyse de la demande processeur, nous proposons un test exact d'ordonnançabilité avec une complexité pseudo-polynomiale. Le principe est de vérifier que pour tout intervalle de temps de longueur t, la demande processeur du système est inférieure ou égale à t. Les instants à tester correspondent aux échéances de toutes les tâches présentes dans la période d'activité.*

5.1 Introduction

Dans le contexte des transactions à priorités dynamiques EDF, une seule analyse définissant un test exact d'ordonnançabinlité a été proposée dans la littérature [71] avec une complexité exponentielle. Une analyse, avec une complexité pseudo-polynomiale, fournissant une condition suffisante d'ordonnançabilité, a été proposée par Palencia et Harbour [71]. Ces travaux se basent essentiellement sur la technique d'analyse de temps de réponse. Dans ce chapitre, nous proposons une condition nécessaire et suffisante d'ordonnançabilité des transactions ordonnancées par EDF, avec une complexité pseudo-polynomiale. Pour cela, nous allons utiliser la technique d'analyse de la demande processeur présentée dans le chapitre 1. Cette technique consiste à prouver que, pour tout intervalle de temps t donné, la demande processeur est inférieure ou égale à la longueur de l'intervalle de temps t. Cette technique fournit un test exact de l'ordonnançabilité globale d'un système de transactions avec EDF. Cette méthode, pour un système donné, ordonnancé par EDF, retourne une réponse booléenne (ordonnançable ou non) concernant son ordonnançabilité, mais aucune information sur le pire temps de réponse des tâches.

La méthode que nous proposons consiste à étendre les résultats de l'analyse de la demande processeur des tâches classiques indépendantes, au cas des transactions. Le test d'ordonnançabilité des tâches indépendantes par l'analyse de la demande processeur, consiste à vérifier que toutes les échéances survenant dans la période d'activité incluant de toutes les tâches sont respectées. Rappelons que la demande processeur, dans un intervalle de temps $[t_1, t_2]$, est la durée cumulée des exécutions des requêtes qui ont leur réveil et leur échéance dans l'intervalle considéré. La période d'activité, pour les tâches classiques indépendantes, est initiée par l'activation simultanée de toutes les tâches du système. Cette définition de période d'activité est pessimiste pour les tâches à offset car certaines tâches ne peuvent jamais s'activer au même instant.

Nous prenons l'exemple d'une transaction série [112], consistant en une acquisition de données série. Le processus est activé par un évènement externe (arrivée d'une trame de température). La figure 5.1 présente une tâche d'acquisition de la température (avec une durée d'exécution de 3 unités de temps et un délai critique de 5 unités de temps) activée à l'arrivée de l'évènement externe, cette tâche est suivie par une tâche d'acquisition de la pression (avec durée d'exécution de 3 unités de temps et un délai critique de 5 unités de temps) activée 5 unités de temps (offset) après l'arrivée de la trame de température. Les deux tâches d'acquisition sont suivies par une tâche de traitement de données activée 10 unités de temps après l'arrivée du premier évènement.

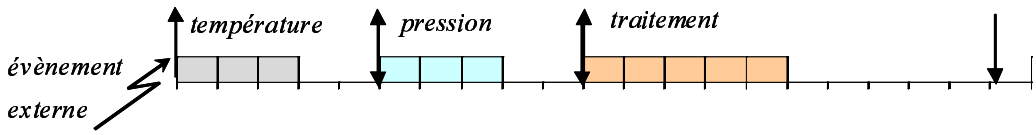

FIG. 5.1 – Exemple d'une transaction série.

Il est clair que ce système de tâches est ordonnançable car les trois tâches ne s'activent jamais au

même moment. Cependant, si nous considérons que toutes les tâches s'activent simultanément (période d'activité pour tâches indépendantes classiques) alors l'ensemble de tâches n'est pas ordonnançable (première échéance dépassée à l'instant 5) à cause de la surestimation importante de la demande processeur imposée par les tâches, qui ne peut jamais être générée dans la réalité. D'où la considération des conditions d'ordonnançabilité des tâches classiques périodiques pour les tâches à offset rend le test pessimiste. Afin d'éliminer ce pessimisme, nous devons prendre en compte les relations d'offset entre les tâches, durant l'analyse.

La section 5.2 présente le principe de test par analyse de la demande processeur et son application sur les systèmes de transactions. Dans la section 5.3, nous présentons le test d'ordonnançabilité, dans une période d'activité donnée, ensuite nous proposons un test exact avec une complexité pseudo-polynomiale, dans la section 5.4. Une amélioration de la rapidité de calcul est apportée par une technique d'implémentation que nous présentons dans la section 5.5 puis une étude de complexité est présentée dans la section 5.5.3.

5.2 Test d'ordonnançablité par analyse de demande processeur

Rappelons que le test d'ordonnançabilité par EDF des tâches classiques indépendantes, basée sur l'analyse de la demande processeur, est donné par le théorème suivant :

Théorème 27 *[16] Un système de tâches synchrone* Γ *est ordonnançable par EDF sur une architecture mono-processeur si et seulement si :*

$$\forall L^* \leq L, dbf(0, L^*) \leq L^* \tag{5.1}$$

tel que L^* *est un échéance d'une instance d'une tâche et* L *la longueur de la plus longue période d'activité*

Dans le contexte des tâches sporadiques, les tâches sont non-concrètes car leur date de première activation n'est pas connue. Le pire scénario d'exécution des tâches sporadiques non concrètes se produit lorsque toutes les tâches s'activent simultanément et ensuite arrivent à leur rythme maximal (i.e. périodiquement avec une période fixe égale au temps minimum séparant deux activations). D'où un système de tâches sporadiques est ordonnançable si et seulement si le système de tâches indépendantes synchrones correspondant est ordonnançable.

Corollaire 1 *Un système de tâches sporadiques non-concret est ordonnançable si et seulement si le système de tâches indépendantes synchrones correspondant est ordonnançable (théorème 27)*

Donc pour analyser l'ordonnançabilité des tâches sporadiques nous nous référons à l'analyse des tâches classiques indépendantes. Puisque le pire scénario d'exécution est connu, nous avons préféré noter $df(t)$ la demande processeur maximale au lieu de dbf (théorème 15). Le théorème 27 montre qu'afin de vérifier l'ordonnançabilité du système il est nécessaire de vérifier que toutes les échéances t présentes dans la plus longue période d'activité sont supérieures ou égales à la demande processeur maximum $df(t)$ causée par le système dans t.

Pour les transactions, le pire scénario est l'un parmi plusieurs candidats. D'où la pire demande processeur d'un système est le maximum de ses demandes causées pour tous les scénarios possibles (qu'on va noter plus tard dbf). Afin de pouvoir appliquer ce test sur un modèle de tâche autre que le modèle de L&L [55], quelques hypothèses, appelées hypothèses d'indépendances, définies par Baruah et al [12], doivent être satisfaites par le modèle de tâche étudié.

- Le comportement d'exécution d'une tâche ne dépend pas du comportement des autres tâches dans le système : chaque tâche est une entité indépendante, peut-être activée séparément par des événements extérieurs. Il n'est pas admissible pour une tâche de générer directement une instance en répondant à une autre tâche génératrice d'instances (aucune interaction entre les tâches).
- On doit pouvoir spécifier les contraintes de la charge processeur sans faire aucune référence au temps absolu : par exemple une spécification telle que la tâche τ génère une instance à l'instant 3 est inadmissible. Les tâches vérifiant cette propriété sont les tâches avec une date de première activations non connue (non-concrètes) où toutes les spécifications temporelles sont faites relativement à la date de la première activation.

Dans le contexte des transactions, le comportement d'exécution de chaque transaction est indépendant du comportement de toute autre transaction dans le système et aucun partage de ressource n'est considéré. Chaque transaction est activée par l'arrivée d'un évènement externe (date d'arrivée non connue a priori). Les transactions sont alors non-concrètes et toutes les spécifications temporelles sont relatives à la date d'arrivée de l'évènement externe. D'où le modèle de transactions satisfait les deux hypothèses d'indépendance, et par conséquent, nous pouvons utiliser le théorème 27, pour tous les scénarios (périodes d'activité) possibles, afin d'analyser l'ordonnançabilité des transactions avec l'algorithme EDF.

Prendre en compte les relations d'offset, existant entre les tâches d'une même transaction, génère plusieurs difficultés qui doivent être résolues :

- Déterminer la période d'activité dans laquelle le test d'ordonnançabilité (théorème 27) doit être effectué.
- Déterminer la fonction de la demande processeur d'une transaction, durant n'importe quel intervalle de temps t.
- Minimiser la complexité du test (applicable pour des systèmes relativement grands) et le temps de calcul.

Pour des transactions ordonnancées par EDF, une période d'activité est produite lorsqu'une tâche τ_{ic} de chaque transaction Γ_i du système s'active simultanément, après avoir été retardée par une valeur maximale de gigue. D'après cette définition, et comme nous l'avons vu pour la RTA présentée dans le chapitre 2, plusieurs périodes d'activités sont possibles, par la combinaison des tâches des transactions initiant la période d'activité. Une solution naïve pour tester l'ordonnançabilité d'un système consiste à effectuer le test d'ordonnançabilité pour toutes les périodes d'activité candidates. Malheureusement le nombre de périodes d'activité, à considérer, et qui est égal à $\prod_{\forall \Gamma_i} |\Gamma_i|$, rend le test inapplicable, à cause de la complexité exponentielle. Ce test est présenté plus tard par le théorème 28.

Avant d'adresser ce problème, nous présentons comment la demande processeur peut être calculée pour une transaction, dans une période d'activité donnée.

5.3 Fonction de la demande processeur

Pour un intervalle de temps t, l'analyse de la demande processeur consiste à capturer la demande (interférence) des tâches activées dans t et qui doivent finir leur exécution dans l'intervalle t (généralement on fixe à 0 le début de la période d'activité). Il est clair que le calcul de la pire demande processeur d'un système dans une période d'activité t, nécessite, bien évidement, de calculer la pire demande (interférence) de chacune des transactions Γ_i (donc de chaque tâche τ_{ij} dans Γ_i) du système.

Notons que la présence de la gigue dans les transactions permet à des instances de tâches activées hors l'intervalle de temps t de contribuer à la demande processeur du système dans t. D'où la définition 4 (chapitre 1) n'est pas applicable directement sur les transactions. Dans la suite nous analysons le mode d'activation des tâches afin de pouvoir déterminer toutes les instances activées dans une période d'activité.

Pour une période d'activité donnée, nous notons τ_{ic} la tâche candidate d'une transaction Γ_i, qui initie cet instant critique. Nous fixons $t = 0$ la date de début de la période d'activité (date d'activation de τ_{ic} après avoir été retardée par une valeur maximale de la gigue). Dans la suite, les indices de type x_{ijc} sont relatifs à la tâche τ_{ij} dans le scénario où τ_{ic} initie la période d'activité.

Puisque plusieurs instances d'une tâche τ_{ij} peuvent interférer dans une période d'activité de longueur t, les instances de τ_{ij} activées dans la période d'activité de longueur t peuvent être divisées en deux ensembles.

- Set1 : Les instances activées avant le début de la période d'activité et peuvent être retardées par une certaine valeur du gigue d'une façon que leurs activations coïncident avec la date de début de la période d'activité.
- Set2 : Les instances activées après le début de la période d'activité.

D'après le résultat présenté dans l'analyse de temps de réponse proposé par Palencia et Harbour [71] (théorème 3), la pire contribution de τ_{ij} dans la période d'activité se produit lorsque toutes les activations de l'ensemble Set1 ont une valeur de gigue de telle sorte qu'elles s'activent au début de la période d'activité et lorsque toutes les activations dans Set2 ont une valeur nulle de gigue. D'où les activations considérées dans Set1 sont toutes les instances de τ_{ij} activées (sans retard de gigue) dans $[-J_{ij}, 0[$.

D'après ces résultats, nous donnons une définition de la fonction de la demande processeur, d'une transaction, pour une période d'activité débutant à la date d'activation simultanée d'une tâche candidate τ_{ic} dans chaque transaction Γ_i.

Définition 13 *(fonction de la demande processeur). Soit Γ_i une transaction, une période d'acti-*

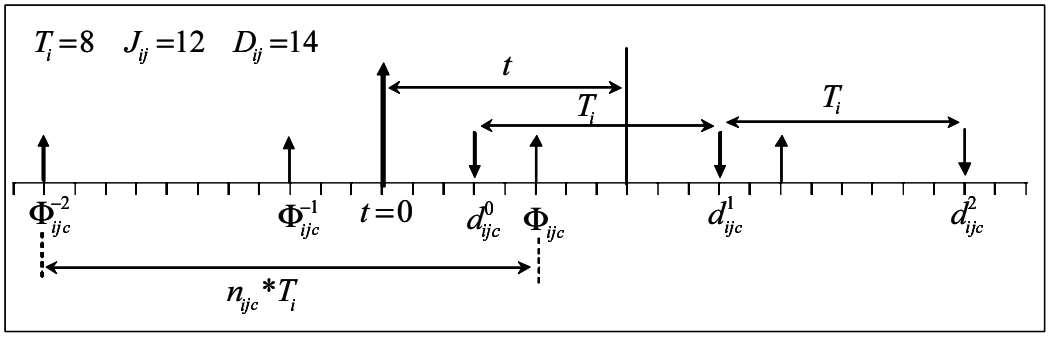

FIG. 5.2 – Motif d'activations et d'échéances dans une période d'activité.

vité qui est initiée par l'activation d'une tâche τ_{ic} de Γ_i, et t un nombre entier positif. La fonction de la demande processeur $df_{ic}(t)$ donne la demande en temps d'exécution cumulative causée par les instances de toutes les tâches τ_{ij}, de Γ_i, qui ont une date d'activation dans $[-J_{ij}, t[$ et une échéance dans l'intervalle $]0, t]$.

D'après la définition de la demande processeur, et afin de faciliter les calculs, nous déterminons, dans un premier temps, la demande processeur causée par les tâches activées avant ou à la date t, sans prendre en compte les échéances, ensuite nous ne gardons que celles des tâches ayant des échéances avant ou à la date t. Notons $df_{ijc}(t)$ la fonction de la demande processeur d'une tâche τ_{ij} dans la période d'activité qui coïncide avec l'activation de la tâche τ_{ic}.

Nous nous intéressons donc à la demande processeur causée par les deux ensembles Set1 et Set2. Afin de déterminer le nombre d'instances d'une tâche τ_{ij}, dans chacune des ensembles Set1 et Set2, nous reposons sur la phase, notée Φ_{ijc}, entre cette tâche et la date de début de la période d'activité. Nous focalisons sur le motif d'activation d'une tâche τ_{ij} (figure 5.2) ; Φ_{ijc} dénote la phase entre τ_{ij} et la date de début de la période d'activité coïncidant avec l'activation de la tâche candidate τ_{ic} ; i.e. la date d'activation de la première instance de τ_{ij} de Set2 dans la période d'activité survient après Φ_{ijc} unités de temps du début de la période ($t = 0$), et les activations suivantes arrivent périodiquement chaque T_i. Notons que $0 \leq \Phi_{ijc} < T_i$.

$$\Phi_{ijc} \quad = (O_{ij} - (O_{ic} + J_{ic})) \mod T_i \tag{5.2}$$

Contrairement au cas du contexte des priorités fixes, la demande processeur des instances de l'ensemble Set1 dépend de la valeur maximale de la gigue et de la longueur de la période d'activité t. Tandis que celle de l'ensemble Set2 dépend seulement de la longueur de la période d'activité. Par conséquent la pire demande processeur de τ_{ij} causée avant la date t est divisées en deux parties. df_{ijc}^{set1} la demande causée par les instances de Set1 avec une échéance avant ou à t, et df_{ijc}^{set2} la demande causée par les instances de Set2 avec échéance avant t. Soit n_{ijc} le nombre d'instances de τ_{ij} appartenant à Set1. Dans l'exemple de la figure 5.2, on a $n_{ijc} = 2$ instances qui sont activées avant le début de la période d'activité et retardées par une valeur de gigue de sorte à s'activer réellement au début de la période d'activité.

$$n_{ijc} = \left\lfloor \frac{J_{ij} + \Phi_{ijc}}{T_i} \right\rfloor \tag{5.3}$$

Parmi les n_{ijc} activations dans Set1, nous ne devons considérer, dans la fonction de la demande processeur, que les instances qui ont un échéance survenant avant ou à la date t. Dans la figure 5.2, seulement la première instance (Φ_{ijc}^{-2}) de Set1 a une échéance d_{ijc}^0 qui survient avant l'intervalle de temps $t = 8$. D'où le nombre d'activations, appartenant à Set1, ayant leurs échéances à ou avant t est obtenu par :

$$\left\lfloor \frac{n_{ijc}T_i + t - D_{ij} - \Phi_{ijc}}{T_i} \right\rfloor + 1 \tag{5.4}$$

Notons que pour une tâche τ_{ij}, nous avons une condition nécessaire triviale $C_{ij} + J_{ij} \leq D_{ij}$. Par conséquent, la fonction df_{ijc}^{set1} déterminant la demande processeur des instances appartenant à l'ensemble Set1 est définie comme suit :

$$df_{ijc}^{set1}(t) = \left(\min \left(n_{ijc}, \left\lfloor \frac{n_{ijc}T_i + t - D_{ij} - \Phi_{ijc}}{T_i} \right\rfloor + 1 \right) \right)_0 C_{ij} \tag{5.5}$$

$(x)_0$ signifie $\max\{x, 0\}$.

Avec le même principe, nous calculons la demande causée par les instances appartenant à l'ensemble Set2. Nous savons que Φ_{ijc} est la date à laquelle la première instance est activée, dans la période d'activité ; et les instances suivantes s'activent périodiquement après la première instance. La fonction $df_{ijc}^{set2}(t)$ est alors donnée par la formule suivante :

$$df_{ijc}^{set2}(t) = \left(\min \left(\left\lceil \frac{t - \Phi_{ijc}}{T_i} \right\rceil, \left\lfloor \frac{t - D_{ij} - \Phi_{ijc}}{T_i} \right\rfloor + 1 \right) \right)_0 C_{ij} \tag{5.6}$$

Notons que l'équation précédente peut retourner une valeur négative parce que le délai critique peut être supérieur à la période. Il est clair que pour $t \leq d_{ijc}^0 + (n_{ijc} - 1)T_i$, $df_{ijc}^{set2}(t)$ est nul car seules les échéances des instances de l'ensemble Set1 surviennent dans cet intervalle de temps. Et pour $t \geq d_{ijc}^0 + (n_{ijc} - 1)T_i$, $df_{ijc}^{set1}(t) = n_{ijc}C_{ij}$.

$$df_{ijc}(t) = df_{ijc}^{set1}(t) + df_{ijc}^{set2}(t) \tag{5.7}$$

$df_{ijc}(t)$ est la demande processeur causée par la tâche τ_{ij} dans une période d'activité de longueur t, lorsque son début coïncide avec l'activation de la tâche τ_{ic}. D'où la demande totale $df_{ic}(t)$, causée par la transaction Γ_i, est égale à la somme des demandes de toutes les tâches de la transaction :

$$df_{ic}(t) = \sum_{\forall j} \left(df_{ijc}^{set1} + df_{ijc}^{set2} \right) \tag{5.8}$$

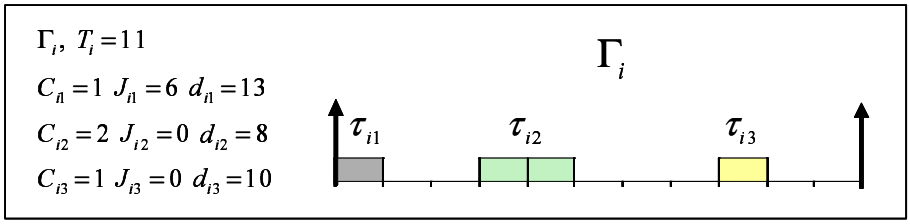

FIG. 5.3 – Exemple d'une transaction avec gigues.

La fonction de la demande processeur d'une transaction peut être représentée graphiquement par une courbe à escalier, où chaque point d'escalier correspond à une échéance d'une ou plusieurs instances de tâches de la transaction, et la demande processeur cumulée, à cette date, est augmentée par la somme des durées d'exécution des instances correspondantes (instances devant terminer leur exécutions à cette date). La figure 5.4 présente la courbe de la demande processeur de la transaction Γ_i donnée dans la figure 5.3, lorsque la tâche τ_{i1} initie la période d'activité.

Maintenant, nous avons tous les ingrédients pour vérifier l'ordonnançabilité du système, dans une période d'activité donnée. Pour cela, il reste à vérifier si toutes les échéances, de chaque tâche dans le système, présentes dans la période d'activité considérée, sont respectées, en utilisant la formule suivante :

$$\forall L^* \leq L, \sum_{\forall \Gamma_i} df_{ic}(L^*) \leq L^* \tag{5.9}$$

Où L est la longueur de la période d'activité considérée. L est obtenue par la recherche itérative de point fixe, en utilisant la demande processeur comme en contexte à priorité fixe (sans prendre en compte les échéances des instances).

$$\begin{aligned} L^{(0)} &= \sum_{\forall i, \forall j} \left\lfloor \frac{J_{ij} + \Phi_{ijc_i}}{T_i} \right\rfloor C_{ij} \\ L^{(n+1)} &= \sum_{\forall i, \forall j} \left(\left\lfloor \frac{J_{ij} + \Phi_{ijc_i}}{T_i} \right\rfloor + \left\lceil \frac{L^{(n)} - \Phi_{ijc_i}}{T_i} \right\rceil \right) C_{ij} \end{aligned} \tag{5.10}$$

Soit ψ_{ic} l'ensemble d'échéances de tâches de Γ_i, présentes dans la période d'activité initiée par l'activation d'une tâche candidate τ_{ic}, de Γ_i. C'est l'union d'échéances d_{ij} des instances de l'ensemble Set1 et ceux de Set2 présentes dans L.

$$\begin{aligned} \psi_{ic} = &\{d_{ij} = \Phi_{ijc} - k.T_i + D_{ij} \leq L : \forall j = 1..|\Gamma_i|, k = 0..n_{ijc}\} \cup \\ &\{d_{ij} = \Phi_{ijc} + k.T_i - D_{ij} \leq L : \forall j = 1..|\Gamma_i|\} \end{aligned} \tag{5.11}$$

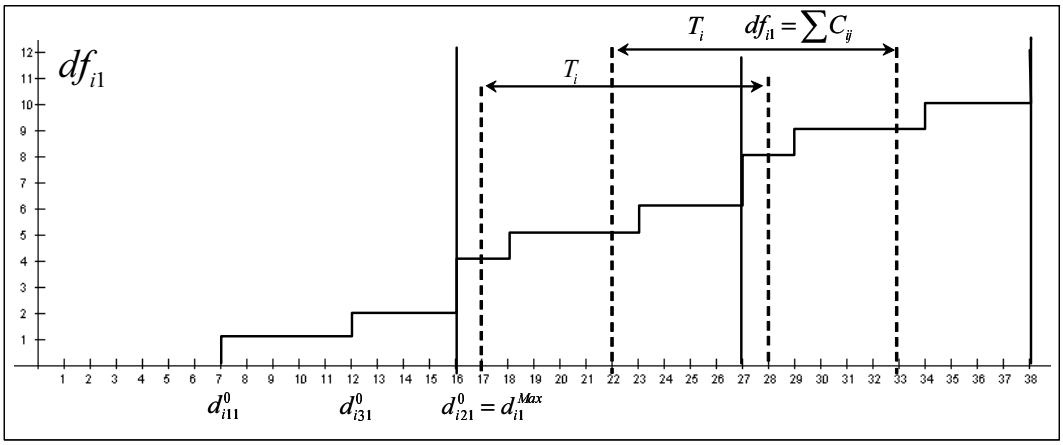

FIG. 5.4 – Fonction de la demande processeur d'une transaction, pour une période d'activité donnée.

Comme nous l'avons dit précédemment, le principal problème de l'analyse d'ordonnançabilité des tâches à offset est que nous ne savons pas quelle tâche τ_{ic} dans chaque transaction Γ_i doit être considérée pour créer la période d'activité pire cas. Une solution intuitive, consiste à vérifier l'ordonnançabilité du système dans toutes les périodes d'activités possibles en parcourant toutes les combinaisons possibles des tâches dans chaque transaction. Le théorème suivant donne le test exact d'ordonnançabilité d'un système de transactions ordonnancées par EDF.

Théorème 28 *Soit* $C :< c_1, c_2, ..., c_{|S|} >$ *une combinaison des indices des tâches candidates* τ_{ic_i} *dans chaque transaction* Γ_i *initiant une période d'activité donnée.* Θ *est l'ensemble de toutes les combinaisons possibles* C *(avec* $i \in 1..|S|$ *et* $c_i \in 1..|\Gamma_i|$*). S est ordonnançable par EDF si et seulement si*

$$\forall t \leq L : \ dbf(t) = \max_{C \in \Theta} \sum_{\forall \Gamma_i} df_{ic_i}(t) \leq t \tag{5.12}$$

Preuve
Le test d'ordonnançabilité consiste à vérifier que pour tout intervalle de temps t (échéance), dans tous les scénarios (périodes d'activité) possibles, la demande processeur est inférieure ou égale à t. Il est clair que si le maximum des demandes correspondant à tous les scénarios est inférieur ou égal à t alors la demande de chaque scénario l'est.
□

Évidement, le nombre important de combinaisons ($\prod_{i=1}^{|S|} |\Gamma_i|$ périodes d'activités) rend inapplicable l'analyse de toutes les combinaisons. Nous pouvons résoudre ce problème de complexité, en utilisant la fonction de la borne de la demande processeur qui calcule la demande processeur maximum d'une transaction pour n'importe quel intervalle de temps t. Cette technique fournit le même résultat que l'analyse exacte du Théorème 29.

5.4 Analyse pseudo-polynomiale

Nous définissons $dbf_i(t)$ la fonction de la demande processeur maximum causée par une transaction Γ_i dans un intervalle de temps de longueur t. Cette fonction mesure alors la demande maximum de toutes les demandes que Γ_i peut causer, en considérant chacune de ses tâches τ_{ic} comme celle qui initie la période d'activité.

Définition 14 *(Demand Bound Function.) Soit Γ_i une transaction. La fonction de la demande maximum $dbf_i(t)$ dénote la demande cumulative maximum en temps processeur causée par les instances de toutes les tâches τ_{ij}, de Γ_i, qui ont une date d'activation dans $[-J_{ij}, t[$ et un échéance dans l'intervalle de temps t. Formellement :*

$$dbf_i(t) = \max_{\forall \tau_{ic} \in \Gamma_i} df_{ic}(t) \tag{5.13}$$

La nouvelle méthode d'analyse consiste à effectuer, uniquement, un test d'ordonnançabilité du système, dans la plus longue période d'activité, en utilisant la demande processeur maximum, au lieu de traiter toutes les périodes d'activité. Autrement dit, pour toutes les échéances t des tâches appartenant à la plus longue période d'activité, vérifier si la demande maximum $dbf(t)$, du système, est inférieure ou égale à la longueur de l'intervalle t. Ce test est équivalent à l'analyse exacte présentée dans la section précédente, i.e. il fournit une condition nécessaire et suffisante d'ordonnançabilité (théorème 29)

Théorème 29 *Un système de transactions est ordonnançable si et seulement si $\sum_{\forall \Gamma_i} dbf_i(t) \leq t$ pour tout nombre positif t.*

Preuve :
La preuve consiste à montrer algébriquement l'équivalence entre la partie gauche de l'inégalité 5.12, du test exact (théorème 28), et celle du théorème 29. Pour un intervalle de temps de longueur t, notons $df_{ic_i}(t)$ la demande processeur dans t de Γ_i lorsque la tâche candidate τ_{ic_i} initie la période d'activité. Supposons $\theta =< df_{1c_1}(t), df_{2c_2}(t), ..., df_{nc_{|S|}}(t) >$ l'ensemble des demandes processeurs de toutes les transactions du système $< \Gamma_1, \Gamma_2, ..., \Gamma_{|S|} >$, pour une période d'activité initiée par l'activation d'une tâche τ_{ic_i} dans chaque transaction Γ_i.

Notons $\Theta = \{\theta : \forall i \in 1..|S|,\ c_i \in 1..|\Gamma_i|\}$ l'ensemble de tous les scénarii possibles (périodes d'activités). La valeur de la fonction de la demande processeur (équation 5.12) du système, dans le test exact est donnée par :

$$dbf(t) = \max_{\theta \in \Theta} \left(\sum_{i=1}^{|S|} df_{ic_i}(t) \right)$$

$df_{ic_i}(t)$ est le i^{eme} élément de l'ensemble θ. i.e. la demande processeur de transaction Γ_i pour une combinaison donnée θ. Puisque chaque demande processeur $df_{ic_i}(t)$ d'une transaction Γ_i est indépendante de toute autre transaction $\Gamma_j \neq \Gamma_i$, alors les éléments de tous les ensembles θ sont indépendants. D'où on peut permuter les opérations de la somme et de la maximisation. La formule de la demande processeur du système, utilisée dans le test, devient :

$$dbf(t) = \sum_{i=1}^{|S|} \left(\max_{\theta \in \Theta} df_{ic_i}(t) \right)$$

Dans chaque combinaison θ le i^{eme} élément correspond toujours à la demande processeur de la transaction Γ_i. D'où cet élément appartient toujours à l'ensemble des demandes possibles que Γ_i peut produire $\{df_{ic_i} : c_i \in 1..|\Gamma_i|\}$. Pour un i donné alors, nous avons :

$$\max_{\theta \in \Theta} df_{ic_i}(t) = \max_{c_i=1..|\Gamma_i|} df_{ic_i}(t)$$

La demande processeur du système devient :

$$dbf(t) = \sum_{i=1}^{|S|} \left(\max_{c_i=1..|\Gamma_i|} df_{ic_i}(t) \right)$$

Par définition de la fonction de la demande processeur (équation 5.13), nous pouvons réécrire la formule comme suit :

$$dbf(t) = \sum_{i=1}^{|S|} dbf_i(t)$$

Alors, le test exact, présenté par l'équation 5.12 est équivalent au test suivant (théorème 29)

$$\forall t \leq L, \sum_{\forall \Gamma_i} \left(\max_{c \in 1..|\Gamma_i|} df_{ic}(t) \right) \leq t \tag{5.14}$$

□

D'après le résultat du théorème 29, l'utilisation de la fonction de la demande processeur maximum de chaque transaction, dans le test, fournit une condition nécessaire et suffisante d'ordonnancabilité. Ce résultat est intéressant en comparaison avec le résultat présenté dans [71] où les auteurs focalisent sur l'analyse de temps de réponse des transactions avec EDF et proposent une condition suffisante d'ordonnançabilité avec une complexité pseudo-polynomiale.

La procédure du test d'ordonnançabilité d'un système S, consiste alors à déterminer s'il existe une date d'échéance L^*, dans la plus longue période d'activité de longueur L, telle que la demande processeur totale avant L^* est supérieure à la longueur L^*.

$$\exists L^* \leq L : \left(\sum_{\Gamma_i \in S} dbf_i(L^*) \right) > L^* \ ? \tag{5.15}$$

La longueur de la plus longue période d'activité est calculée en utilisant la fonction d'interférence maximum (chapitre 2), dans le contexte des priorités fixes (sans prendre en considération les échéances des tâches)

$$L^{(n+1)} = \sum_{\forall \Gamma_i} W_i(L^{(n)}) \tag{5.16}$$

$W_i(t)$ désigne l'interférence cumulative maximum causée par toutes les instances (activées dans t) de toutes les tâches de la transaction Γ_i, dans n'importe quel intervalle de temps de longueur t.

$$W_i(t) = \max_{\forall \tau_{ic}} \left\{ \sum_{\forall j} \left(\left\lfloor \frac{J_{ij} + \Phi_{ijc}}{T_i} \right\rfloor + \left\lceil \frac{t - \Phi_{ijc}}{T_i} \right\rceil \right) C_{ij} \right\} \tag{5.17}$$

Pour notre exemple de la figure 5.3, nous calculons la demande processeur de la transaction Γ_i, dans un intervalle de temps $t = 23$, lorsque chacune des tâches τ_{i1}, τ_{i2}, ou τ_{i3} initie la période d'activité. Alors le maximum parmi toutes les demandes est considéré comme une borne de la demande de Γ_i. Nous détaillons le calcul de la demande de la tâche τ_{i1} lorsqu'elle initie la période d'activité :

$$df_{i11}^{set1}(23) = \left(\min \left(1, \left\lfloor \frac{1 \times 11 + 23 - 5 - 13}{11} \right\rfloor + 1 \right) \right)_0 \times 1 = 1$$

$$df_{i11}^{set2}(23) = \left(\min \left(\left\lceil \frac{23 - 5}{11} \right\rceil, \left\lfloor \frac{23 - 5 - 13}{11} \right\rfloor + 1 \right) \right)_0 \times 1 = 1$$

D'où

$$df_{i11}(23) = df_{i11}^{set1}(23) + df_{i11}^{set2}(23) = 2$$

Avec le même principe, nous calculons la demande causée par τ_{i2} ($df_{i21}(23) = 2$), et celle causée par τ_{i3} ($df_{i31}(23) = 2$). La demande causée par la transaction Γ_i, lorsque τ_{i1} initie la période d'activité, est $df_{i1}(23) = 2 + 2 + 2 = 6$ (Figure 5.4).

Lorsque τ_{i2} initie la période d'activité, Γ_i produit une demande de $df_{i2}(23) = 7$ unités de temps ; tandis que lorsque τ_{i3} l'initie la demande processeur produite par Γ_i est de $df_{i3}(23) = 5$ unités de temps. D'où la demande maximum, causée par Γ_i, dans un intervalle de temps de 23 unités de temps, est $dbf_i(23) = \max\{df_{i1}(23), df_{i2}(23), df_{i3}(23)\} = 7$.

Cette méthode a un complexité pseudo-polynomiale, le calcul de la borne de la demande processeur d'une transaction a une complexité $O(|\Gamma_i|^2)$ car pour chaque tâche candidate τ_{ic} qui initie la période d'activité, nous devons calculer la demande causée par chacune des tâches τ_{ij} de Γ_i. Soit X le nombre des échéances à vérifier dans la plus longue période d'activité, alors la complexité de la méthode d'analyse est $O(X \ |\Gamma_i|^2 \, |S|)$.

5.5 Analyse d'ordonnançabilité accélérée

Nous remarquons que lors du test de l'ordonnançabilité d'un système, pour chaque échéance à vérifier, le calcul de la demande processeur maximum $dbf_i(t)$, causée par une transaction Γ_i, nécessite de traiter tous les paires possibles (τ_{ic}, τ_{ij}), de tâches de Γ_i. Ce calcul répétitif des demandes $df_{ijc}(t)$ pour chaque paire de tâches peut être évité : ainsi une optimisation en complexité est possible. L'idée est similaire à celle employée dans le contexte de transactions à priorités fixes, elle consiste à trouver un motif répétitif de la fonction df_{ic}, ainsi que pour la fonction maximum dbf_i. Pour cela, nous proposons, dans cette section, un nouvel algorithme d'implémentation efficace qui réduit significativement le temps de traitement mis par la méthode d'analyse. Au premier temps, nous déterminons un motif statique et périodique de la fonction de la demande maximum des transactions. Ce motif permet de stocker la demande des transactions dans des tables qui seront utilisées ultérieurement durant les différentes étapes du test, pour déduire facilement la $dbf(t)$ pour n'importe quel intervalle de temps t.

5.5.1 La périodicité de la fonction de la demande

Puisque dans l'ordonnancement avec EDF, la demande processeur dans un intervalle de temps est essentiellement liée aux date d'échéances des tâches, nous nous focalisons sur la périodicités des échéances afin de trouver un motif périodique de la fonction de la demande processeur. Nous savons que les dates d'activations d'une tâche sont périodiques, par conséquent les dates d'échéances le sont aussi. Soit une période d'activité donnée, initiée par l'activation d'une tâche τ_{ic} d'une transaction Γ_i. Notons d^0_{ijc} la première date d'échéance, présente dans la période d'activité, d'une tâche τ_{ij} appartenant à Γ_i. d^0_{ijc} est alors la date d'échéance de la première instance appartenant à l'ensemble Set1 s'il n'est pas vide, sinon elle correspond à la date d'échéance de la première instance de l'ensemble Set2. La première échéance de τ_{ij} (dans la période d'activité) est donnée par :

$$d^o_{ijc} = \Phi_{ijc} + D_{ij} - n_{ijc}T_i \tag{5.18}$$

Par conséquent, nous pouvons déduire la demande processeur de τ_{ij}, durant n'importe période d'activité de longueur $t = d^0_{ijc} + k \times T_i$ par :

$$df_{ijc}(t) = (k+1) \times C_{ij}$$

Nous constatons que, durant chaque intervalle de longueur T_i après la première échéance d^0_{ijc}, la demande (non cumulative) causée par τ_{ij} vaut toujours C_{ij} (une activation (échéance) chaque T_i unités de temps). Notons d^{Max}_{ijc} le maximum parmi toutes les premières échéances de toutes les tâches de Γ_i (figure 5.4).

$$d^{Max}_{ic} = \max_{\forall \tau_{ij}} \left\{ d^0_{ijc} \right\} \tag{5.19}$$

Utilisant ce constat, nous déduisons que la fonction de la demande processeur d'une transaction est périodique après la date d^{Max}_{ic}, et la demande processeur (non cumulative) causée par

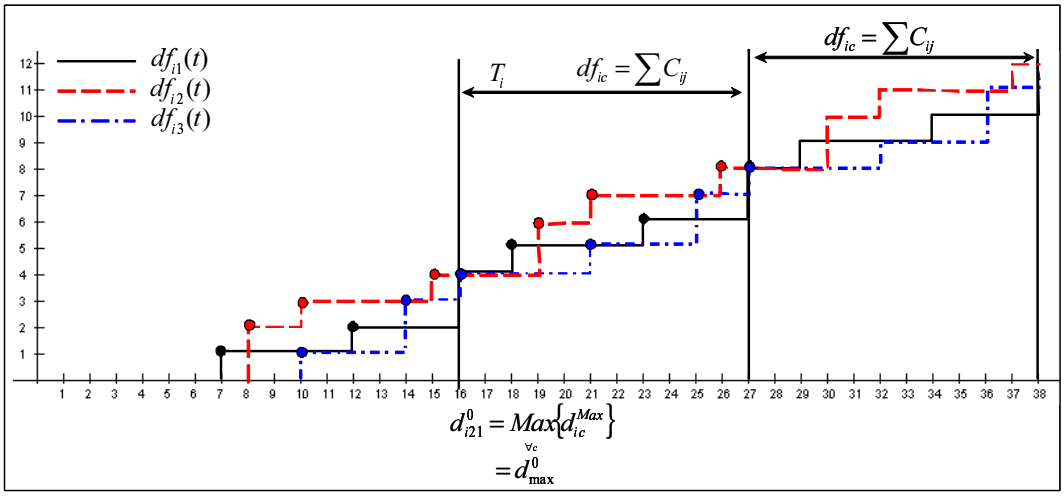

FIG. 5.5 – Motif de la fonction de la demande processeur d'une transaction pour toutes les périodes d'activités.

une transaction Γ_i, durant tout intervalle de longueur T_i (après une date $t \geq d_{ic}^{Max}$) est égale à $\sum_{\forall \tau_{ij}} C_{ij}$. La figure 5.4 montre que, après n'importe quelle date $t > d_{ic}^{Max}$, la demande de Γ_i, lorsque τ_{i1} initie la période d'activité, est répétitive (avec une période égale à T_i), ce qui permet d'obtenir une représentation statique de la fonction df_{ic} durant tout temps $t \geq d_{ic}^{Max}$.

Afin de représenter statiquement la fonction $dbf_i(t)$ (utilisée dans le test), qui est le maximum des demandes processeur possibles de Γ_i, nous devons, d'abord, représenter statiquement toutes ses demandes df_{ic}, en considérant chacune de ses tâches comme celle initiant la période d'activité, durant le même intervalle de temps (le plus long intervalle durant lequel df_{ic} est non répétitive). Pour une déduction simple de dbf_i, durant un intervalle de temps arbitraire t (Théorème 30) nous représentons statiquement les fonctions de la demande pour chaque tâche candidate τic durant l'intervalle de temps $d^0_{max} + T_i$.

$$d^0_{max} = \max_{\forall c, \forall j} \left\{ d^0_{ijc} \right\} = \max_{\forall c} \left\{ d^{Max}_{ic} \right\} \tag{5.20}$$

5.5.2 Représentation statique

Pour chaque période d'activité, initiée par une tâche τ_{ic}, nous définissons un ensemble ou une table P_{ic} de points, où chaque point $P_{ic}[k]$ possède deux coordonnées : x (représente une date d'échéance) et y (représente la demande processeur cumulative causée par les tâches de Γ_i ayant une activation et une échéance avant x), décrivant la valeur de la demande cumulative de Γ_i, à une échéance, lorsque l'activation de τ_{ic} initie la période d'activité. Les points dans P_{ic} correspondent aux points convexes de la fonction $df_{ic}(t)$ illustrés par des points dans la figure 5.5. Notons Ω_{ic} l'ensemble ordonné et non redondant, des couples date d'échéance d (survenant avant ou à la date $d^0_{max} + T_i$) et demande correspond C, causée seulement par les instances de toutes les tâches de Γ_i ayant la même échéance d.

$$\Omega_{ic} = \left\{ d^0_{ijc} + k \times T_i \leq d^0_{max} + T_i,\ \forall j, \forall k = 0, 1... \right\} \tag{5.21}$$

L'évolution de la fonction de la demande processeur bf_{ic}, durant l'intervalle de temps $d^0_{max} + T_i$ est stockée dans la table P_{ic} de sorte que l'on ait les équations suivantes :

$$\begin{aligned} P_{ic}[1].x &= \Omega_{ic}(1).d \\ P_{ic}[1].y &= \Omega_{ic}(1).c \\ &... \\ P_{ic}[k+1].x &= \Omega_{ic}(k+1).d \\ P_{ic}[k+1].y &= P_{ic}[k].y + \Omega_{ic}(k+1).c \end{aligned} \tag{5.22}$$

Pour notre exemple de la figure 5.5, les tables P_{ic} suivantes stockent les coordonnées des points convexes, pour chaque courbe correspondant à la fonction df_{ic} , lorsque chacune des tâches τ_{i1}, τ_{i2} ou τ_{i3} initie la période d'activité.

$$P_{i1} =< (7,1), (12,2), (16,4), (18,5), (23,6), (27,8) >$$

$$P_{i2} =< (8,2), (10,3), (1,4), (19,6), (21,7), (26,8) >$$

$$P_{i3} =< (10,1), (14,3), (16,4), (21,5), (25,7), (27,8) >$$

Une fois toutes les informations, sur toutes les fonctions df_{ic}, stockées dans les tables P_{ic}, la représentation statique de la fonction de la demande maximum dbf_i, dans une table qu'on note P_i est comme suit :
la table P_{ic} est initialisée par l'union de toutes les tables P_{ic}.

$$P_i = \bigcup_{\forall \tau_{ic}} P_{ic} \tag{5.23}$$

Afin de déterminer les points de P_i correspondant au points convexes de la fonction $dbf_i(t)$ (la figure 5.6 représente la courbe dbf_i qui est l'enveloppe supérieure des courbes df_{ic} de la figure 5.5), qui décrit la demande processeur maximum, nous définissons la relation « Envelopper » :
Un point $P_i[a]$ enveloppe un autre point $P_i[b]$ (noté $P_i[a] \succ P_i[b]$) si la présence de $P_i[a]$ implique que $P_i[b]$ n'est pas un point convexe de la fonction de la demande maximum. Graphiquement, les points de la table P_i sont illustrés par des points dans la figure 5.6.

$$\begin{aligned} P_i[a] \succ P_i[b] \ \ &\textit{si et seulement si} \\ (P_i[a].x \leq P_i[b].x \ \wedge \ P_i[a].y &\geq P_i[b].y) \end{aligned} \tag{5.24}$$

En utilisant la relation Envelopper, nous supprimons de l'ensemble P_i tous les points enveloppés (qui ne peuvent pas représenter la fonction dbf_i i.e. les points convexes de la courbe représentant dbf_i) :

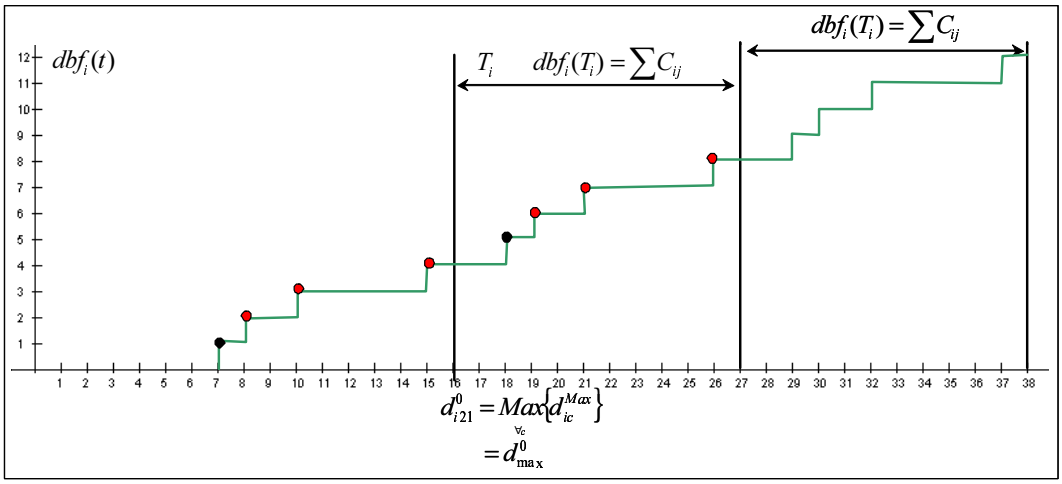

FIG. 5.6 – Fonction de la demande processeur maximum d'une transaction.

$$Supprimer \;\; P_i[b] \;\; de \;\; P_i \;\; si \;\; \exists a \neq b : P_i[a] \succ P_i[b] \tag{5.25}$$

Pour notre transaction de la figure 5.5. Nous ne gardons dans la table $P_i = p_{i1} \cup p_{i2} \cup p_{i3}$ que les points représentant le maximum des dbf_i. D'où nous avons :

$$P_i =< (7.1), (8.2), (10.3), (15.4), (18.5), (19.6), (21.7), (26.8) >$$

A cette étape, La demande maximum dbf_i est stockée dans P_i. Il reste alors à montrer comment la périodicité de dbf_i sera exploitée afin de calculer la demande processeur.
En utilisant la table P_i, la demande maximum causée par une transaction Γ_i, durant un intervalle de temps $t \leq d^0_{max} + T_i$, est obtenue par la fonction de recherche rapide, dans P_i, suivante :

$$\begin{aligned} dbf_i(t) &= P_i[n].y \\ n &= \max\{m : P_i[m].x \leq t\} \end{aligned} \tag{5.26}$$

Nous montrons, via le théorème suivant, comment la borne de la demande processeur de Γ_i peut être déduite, en utilisant la table P_i, pour tout intervalle t.

Théorème 30 *Durant un intervalle de temps $t = t' + m \times T_i$, où $d^0_{max} \leq t' < d^0_{max} + T_i$, la borne de la demande causée par une transaction Γ_i est : $dbf_i(t) = dbf_i(t') + m \times \sum_{\forall \tau_{ij}} C_{ij}$.*

Preuve :
Nous pouvons prouver, facilement, cette déduction, par une équivalence algébrique : Soit $t = d^0_{max} + m \times T_i + t^*$, où $t^* < T_i$, la longueur d'une période d'activité. Notons $dbf_i(a, b) = dbf_i(b) - dbf_i(a)$ la demande causée durant l'intervalle de temps commençant à la date a, dans une période d'activité, et terminant à la date b. Puisqu'après la date $t > d^0_{max}$, la demande causée par une transaction est périodique avec une période T_i et durant chaque intervalle de temps de longueur

T_i une demande de $\sum_{\forall \tau_{ij}} C_{ij}$ unités de temps est produite (Figure 5.5), alors la demande produite entre $d^0_{max} + kT_i$ et $d^0_{max} + kT_i + t^*$ est équivalent à celle produite entre d^0_{max} et $d^0_{max} + t^*$.

$$dbf_i(d^0_{max} + kT_i, d^0_{max} + kT_i + t^*) = dbf_i(d^0_{max}, d^0_{max} + t^*)$$

D'où

$$\begin{aligned}
dbf_i(t) &= dbf_i(0, d^0_{max}) + m \times \sum_{\forall \tau_{ij}} C_{ij} + \ dbf_i(d^0_{max}, d^0_{max} + t^*) \\
dbf_i(t) &= dbf_i(d^0_{max}) + m \times \sum_{\forall \tau_{ij}} C_{ij} + \ dbf_i(d^0_{max} + t^*) - dbf_i(d^0_{max}) \\
dbf_i(t) &= m \times \sum_{\forall \tau_{ij}} C_{ij} + \ dbf_i(d^0_{max} + t^*) \\
dbf_i(t) &= m \times \sum_{\forall \tau_{ij}} C_{ij} + \ dbf_i(t')
\end{aligned}$$

Avec

$$t' = d^0_{max} + t^* \ \ and \ \ d^0_{max} \leq t' < d^0_{max} + T_i$$

□

Pour le même exemple de la figure 5.3, nous calculons la borne de la demande processeur de Γ_i, pour $t = 32$. Nous utilisons la table P_i pour calculer la demande pour tout $t < d^0_{max} + T_i = 27$. Nous avons $t = 32 = 21 + 1 \times T_i = 21 + 11$ ($m = 1$ et $d^0_{max} = 16 \leq 21 < d^0_{max} + T_i = 27$. D'où $dbf_i(32) = dbf_i(21) + 1 \times \sum_{\forall \tau_{ij}} C_{ij} = 7 + 4 = 11$ (Figure 5.6). De la table P_i, nous déduisons que $dbf_i(21) = P_i[7].y = 7$.

Afin de valider un système S, une seule opération de représentation statique est effectuée pour chaque transaction, ainsi durant les tests sur la demande, pour toute échéance, une simple déduction de dbf_i est effectuée, au lieu de recalculer la demande pour chaque paire de tâches (τ_{ic}, τ_{ij}). L'ensemble ψ des dates d'échéances à vérifier est réduit aux dates auxquelles la fonction dbf_i, pour toute transaction Γ_i, change de valeur. Par conséquent, nous devons vérifier que, pour tout échéance d, appartenant à ψ, la borne de la demande causée par le système est inférieure ou égale à d.

$$\psi = \bigcup_{\forall \Gamma_i \in S} \psi_i \tag{5.27}$$

$$\begin{aligned}
\psi_i = \{d = P_i[k].x \ / \ k = 1..|P_i|\} \cup \\
\left\{ \ d = P_i[k].x + m \times T_i \ / P_i[k].x > d^0_{max} \ \wedge \ m = 1..\right\}
\end{aligned}$$

5.5.3 Complexité de la méthode

Dans le contexte des tâches sporadiques ordonnancées par l'algorithme EDF, [15, 10] a prouvé que pour un facteur d'utilisation $U < 1$ la longueur de la période d'activité est bornée par $\ell = \left(\frac{U}{1-U} \max_{i=1}^{N} \{T_i - D_i\}\right)$ et la complexité de l'analyse d'ordonnancabilité est $O(N.\ell)$, avec N le nombre de tâches et $U = \sum_{i=1}^{N}(C_i/T_i)$ le facteur d'utilisation du système.

L'analyse d'ordonnançabilité des systèmes de transactions est divisée en deux grandes phases. La première est la représentation statique de la fonction de la demande processeur maximum, pour chaque transaction Γ_i du système, durant un intervalle de temps $t \leq d^0_{max} + T_i$. Cette opération a, pour une transaction Γ_i, une complexité de :

$$O\left(\left(1 + \frac{d^0_{max} - d^0_{min}}{T_i}\right) |\Gamma_i|^2 . \log\left(\left(1 + \frac{d^0_{max} - d^0_{min}}{T_i}\right) |\Gamma_i|\right)\right)$$

La deuxième phase est la détermination de l'ordonnancabilité du système, en utilisant la recherche dans les tables P_i, obtenues dans la première phase. La recherche d'une valeur dans une table P_i nécessite un temps de complexité

$$O\left(\log\left(\left(1 + \frac{d^0_{max} - d^0_{min}}{T_i}\right) |\Gamma_i|\right)\right)$$

Notons X le nombre d'échéances (de l'ensemble ψ) qui doivent être vérifiées. D'où la complexité totale de la deuxième phase est

$$O\left(X \log\left(\left(1 + \frac{d^0_{max} - d^0_{min}}{T_i}\right) |\Gamma_i|\right)\right)$$

Dans le cas où $D_{ij} < T_i$ la complexité de calcul est réduite à $O\left(|\Gamma_i|^2 . \log(|\Gamma_i|)\right)$ pour la phase de la représentation statique de la demande processeur, et pour le test d'ordonnançabilité à $O(X \log |\Gamma_i|)$. Par conséquent, la complexité totale devient $O\left(|\Gamma_i|^2 . \log(|\Gamma_i|) + X \log |\Gamma_i|\right)$.

Pour comparaison, nous présentons la complexité de l'analyse de temps de réponse (RTA), des tâches à offset, proposée par Harbour [71], fournissant une condition suffisante d'ordonnançabilité. Comme nous l'avons présenté dans le chapitre 2, cette analyse consiste à calculer une borne supérieure du pire temps de réponse pour chaque tâche dans le système, en utilisant une fonction d'interférence maximum (approximation). Pour une tâche analysée τ_{ua}, notons : X le nombre d'échéances (instants critiques), de toutes les tâches du système, dans la plus longue période d'activité ; Y le nombre d'itérations parcourues durant le processus du calcul du temps de réponse (recherche itérative d'un point fixe) d'une instances de τ_{ua}, pour un instant critique donné ; et Z le nombre d'instances de τ_{ua} activées dans la période d'activité correspond à l'instant critique considéré.
Le calcul de l'interférence maximum, d'une transaction, durant un intervalle de temps t, a une complexité $O(|\Gamma_i|^2)$. L'analyse de temps de réponse consiste à calculer le temps de réponse de toutes les instances, activées dans la période d'activité, de la tâche analysée, et cela pour tous les instants critiques possibles dans la période d'activité. D'où la complexité de l'algorithme RTA, pour τ_{ua} est :

$$O(X \times Y \times Z \times |S| \times |\Gamma_i|^2)$$

L'analyse est effectuée pour chacune de tâches du système afin d'avoir une décision sur l'ordonnançabilité du système, avec une complexité $O(X \times Y \times Z \times |S|^2 \times |\Gamma_i|^3)$.

Nous remarquons que la méthode que nous avons proposée dans ce chapitre fournit un test d'ordonnançabilité exact avec une complexité pseudo-polynomiale. Par contre le test pseudo-polynomial existant, basé sur l'analyse de pire temps de réponse est suffisant.

5.6 Conclusion

Pour les transactions ordonnancées par EDF, il n'existait qu'une analyse suffisante d'ordonnançabilité avec une complexité pseudo-polynomiale. Le test exact basé sur les temps de réponse est inapplicable à cause de sa complexité exponentielle. Dans ce chapitre, nous avons proposé un test exact d'ordonnançabilité des transactions avec une complexité pseudo-polynomiale. Ce test est basé sur l'analyse de la demande processeur, il fournit une réponse sur l'ordonnançabilité globale de système, sans aucune information sur les temps de réponse des tâches. Le temps de calcul a été optimisé par la méthode d'implémentation présentée dans la section 5.5 qui consiste à représenter statiquement la fonction de la demande processeur dans des tables ; ces tables sont utilisées par la suite pour déduire la demande du système pour n'importe quel intervalle de temps.

Conclusion

Dans le travail présenté dans ce mémoire, nous avons contribué au test d'ordonnançabilité des transactions temps réel. Le modèle des transactions est considéré sans aucune restriction sur les gigues et sur les délais critiques (cas le plus général). Notons que ces résultats sont applicables sur le modèle de tâches multiframes et multiframes généralisées, car il a été montré que ces deux modèles sont des cas particuliers des transactions. Nous avons prouvé formellement et confirmé par des évaluations expérimentales la qualité des résultats des méthodes et des propriétés proposées.

Dans la partie État de l'art de ce mémoire, nous avons introduit les système temps réel et la problématique de l'ordonnancement, suivi par les résultats de base, pour le test d'ordonnançabilité des systèmes de tâches basés sur le modèle de Liu et Layland (tâches indépendantes). Dans certains types d'applications, il existe une relation de décalage d'activations entre les tâches ce qui rend pessimistes les tests effectués sur le modèle de Liu et Layland. Le modèle des transactions permet de prendre en considération les décalages d'activation. Cependant, dans le cas priorités fixes les tests exacts ont une complexité exponentielle tandis que les tests avec une complexité pseudo-polynomiale sont pessimistes (approchés).

Notre contribution à l'ordonnancements des transactions concerne les deux contextes d'ordonnancement : priorités fixes et priorités dynamique EDF. En priorités fixes nous avons focalisé sur l'analyse de temps de réponse, les principaux résultats sont :

- Optimisation de la qualité du test approché en conservant un calcul pseudo-polynomial : la méthode d'analyse mixte, proposée dans le chapitre 3, améliore la borne supérieure du temps de réponse fournie par la méthode d'analyse approchée de Turja-Nolin. Elle consiste à combiner le principe du calcul exact et le principe d'approximation afin de diminuer le pessimisme de l'analyse pire cas, permettant ainsi d'améliorer la borne supérieure du temps de réponse fournie tout en conservant une complexité pseudo-polynomiale. Notons que cette méthode est paramétrable, plus le nombre de transactions considérées pour une analyse exacte est élevé, meilleure est la qualité de l'analyse d'ordonnançabilité, et plus la complexité du calcul est importante. L'évaluation expérimentale a montré que la méthode mixte avec deux transactions choisies pour une analyse exacte (NM2) offre le meilleur compromis qualité/temps de calcul pour des systèmes de taille réaliste (de quelques dizaines au quelques centaines de tâches).
- Identification du pire scénario (analyse exacte) : pour certaines transactions, nous pouvons identifier la tâche candidate qui initie l'instant critique pire cas. Ces transactions sont

celles qui vérifient la propriété d'accumulativité monotonique. Par conséquent, si toutes les transactions sont accumulativement monotoniques pour une tâche analysée alors l'instant critique pire cas est identifiable dans un temps pseudo-polynomial et la condition suffisante de son ordonnançabilité est nécessaire et suffisante.
– Optimisation en temps de calcul : nous avons défini la propriété de dominance des tâches. Toute tâche dominée par une autre tâche est éliminée comme candidate à initier l'instant critique, évitant ainsi tous les instants initiés par elle. Notons que la propriété d'accumulativité monotonique des transactions AM est cas particulier de la propriété de tâches dominées. L'évaluation expérimentale a montré que le gain en nombre d'instants critiques conduit à une accélération importante du temps de calcul surtout pour les méthodes mixtes NM2 et NM3 et que la propriété d'AM peut confirmer que le test approché donne un résultat exact pour un taux considérable des tâches dans le système.

Dans le cas d'ordonnancement à priorités dynamiques EDF, nous avons proposé un test exact basé sur l'analyse de la demande processeur avec une complexité pseudo-polynomiale. Il fournit une réponse booléenne sur l'ordonnançabilité globale du système, sans aucune information sur les temps de réponse des tâches. Afin d'optimiser le temps de calcul, nous avons proposé une implémentation qui consiste à représenter statiquement la fonction de la demande processeur dans des tables ; puis les tables sont utilisées pour déduire la demande du système pour n'importe quel intervalle de temps.

En perspective de ces travaux, dans le contexte d'ordonnancement à priorités fixes, nous pensons pouvoir améliorer la qualité de l'analyse d'ordonnançabilité, en fixant un nombre seuil d'instants critiques pour appliquer chacune des méthodes d'analyse mixte et exacte. Il s'agit, pour une tâche analysée, en fonction du nombre d'instants critiques à considérer, après avoir éliminé tous les instant initiés par des tâches dominées, de choisir alternativement la méthode d'analyse exacte ou la méthode d'analyse mixte (NM2, NM3, ...). Cela permettrait de maîtriser le temps de traitement, tout en améliorant la qualité du test d'ordonnançabilité.

Une autre étude est envisagée : l'idée est d'étendre notre analyse d'ordonnançabilité exacte des transactions avec priorités dynamiques EDF en prenant en compte des ressources partagées. Nous souhaitons aussi étudier l'optimalité de l'algorithme d'ordonnancement EDF pour les transactions.

Les transactions sont non concrètes, mais dans le cas réel, un système peut comprendre des tâches indépendantes concrètes. Notons que l'analyse des tâches concrètes différées est NP-Difficile. Donc l'étude de l'ordonnançabilité des systèmes de transactions en présence de tâches concrètes devient plus complexe. L'idée est d'étudier la possibilité de modéliser les offsets d'activations des tâches concrètes par une transaction afin d'améliorer la qualité de l'analyse. Un autre problème envisagé est d'attribuer les offsets aux tâches concrètes de façon à assurer l'ordonnançabilité du système.

Aux vu des résultats que nous avons obtenus pour l'analyse d'ordonnançabilité des transactions,

nous nous proposons d'étudier comment adapter l'analyse holistique à la présence de transactions.

ANNEXES

Annexe A

Formules complètes

A.1 Méthode approchée de Turja et Nolin

Nous présentons les formules complètes du calcul approché du temps de réponse d'une tâche analysée τ_{ua} par la méthode de Turja et Nolin lorsqu'une tâche τ_{uc_u} initie l'instant critique.

La phase d'activation, (date de la première activation après l'instant critique) d'une tâche τ_{ij} lorsqu'une tâche candidate τ_{ic} initie l'instant critique, est donnée par :

$$\Phi_{ijc} = (O_{ij} - (O_{ic} + J_{ic}))\ mod\ T_i \tag{A.1}$$

L'interférence effective causée par une τ_{ij} lorsque l'instant critique est initié par τ_{ic}.

$$W_{ijc,Nolin}(t) = \left\lfloor \frac{J_{ij}+\Phi_{ijc}}{T_i} \right\rfloor C_{ij} + \left\lceil \frac{t-\Phi_{ijc}}{T_i} \right\rceil C_{ij} - x_{ijc}(t^*) \tag{A.2}$$

Où

$$\begin{aligned} t^* &= t - \Phi_{ijc} \\ x_{ijc}(t^*) &= \begin{cases} C_{ij} - (t^*\ mod\ T_i) & \text{si } t^* > 0 \wedge (0 < t^* mod T_i) < C_{ij}) \\ 0 \text{ sinon} \end{cases} \end{aligned}$$

L'interférence effective de Γ_{ic} est donnée par :

$$W_{i,Nolin}(\tau_{ua}, t) = \sum_{\forall j \in hp_i(\tau_{ua})} \left(\left\lfloor \frac{J_{ij}+\Phi_{ijc}}{T_i} \right\rfloor C_{ij} + \left\lceil \frac{t-\Phi_{ijc}}{T_i} \right\rceil C_{ij} - x_{ijc}(t-\Phi_{ijc}) \right) \tag{A.3}$$

La fonction d'interférence maximum (approchée) de Γ_i :

$$W_{i,Nolin}(\tau_u, t) = \max_{\forall c \in hp_i(\tau_{ua})} W_{ic,Nolin}(\tau_u, t) \tag{A.4}$$

La position $p_{0,ua}$ de la première instance de τ_{ua} présente dans la période d'activité est :

$$p_{0,ua} = -\left\lfloor \frac{J_{ua} + \Phi_{uac_u}}{T_u} \right\rfloor + 1 \tag{A.5}$$

La longueur L de la période d'activité est :

$$\begin{aligned} L^{(k)}_{ua,Nolin} &= B_{ua} + \left(\left\lceil \frac{L^{(k-1)}_{ua,Nolin} - \Phi_{uac_u}}{T_u} \right\rceil - p_{0,ua} + 1 \right) C_{ua} + W_{uc_u,Nolin}(\tau_{ua}, L^{(k-1)}_{ua,Nolin}(p)) + \\ &\sum_{\forall i} W_{ic_i,Nolin}(\tau_{ua}, L^{(k-1)}_{ua,Nolin}) \end{aligned} \tag{A.6}$$

La position de la dernière instance de τ_{ua} présente dans la période d'activité est donnée par :

$$p_{L,ua} = \left\lfloor \frac{L_{ua,Nolin} - \Phi_{uac_u}}{T_u} \right\rfloor \tag{A.7}$$

Afin de déterminer l'instant de fin d'exécution de l'instance p, nous calculons la longueur de la période d'activité associée à l'instance p

$$\begin{aligned} L^{(k)}_{ua,Noin}(p) &= B_{ua} + (p - p_{0,ua} + 1)\, C_{ua} + W_{uc,Nolin}\left(\tau_{ua}, L^{(k-1)}_{ua,Noin}(p)\right) + \\ &\sum_{\forall i \neq u} W_{i,Nolin}\left(\tau_{ua}, L^{(k-1)}_{ua,Nolin}(p)\right) \end{aligned} \tag{A.8}$$

La date d'activation $a_{ua}(p)$ d'une instance de position p est donnée par :

$$a_{ua}(p) = (p-1)T_u + \Phi_{uac_u} \tag{A.9}$$

Le temps de réponse $R_{ua}(p)$ de l'instance p est donné par la différence entre la date de fin d'exécution et celle de son activation :

$$R_{ua}(p) = L_{ua,Nolin}(p) - a_{ua}(p) \tag{A.10}$$

Le temps de réponse de τ_{ua} lorsqu'une tâche τ_{uc_u} initie l'instant critique est alors :

$$R_{ua} = \max_{p=p0,ua...pL,ua} R_{ua}(p) \tag{A.11}$$

Le pire temps de réponse est le maximum des temps de réponse obtenus pour chacune des tâches candidates τ_{uc} de Γ_u

$$R_{ua} = \max_{c_u \in hp_u(\tau_{ua})} \left\{ \max_{p=p0,ua...pL,ua} R_{ua}(p) \right\} \tag{A.12}$$

A.2 Méthode d'analyse mixte

Nous présentons les formules complètes du calcul mixte, du temps de réponse d'une tâche analysée τ_{ua} lorsqu'une analyse exacte d'interférence d'une transaction Γ_e est effectuée (soit τ_{ec_e} la tâche initiant l'instant critique). Nous considérons une tâche τ_{uc_u}, de Γ_u qui initie l'instant critique.

Les formules du calcul d'interférence sont les même que celles de l'analyse approchée de Turja et Nolin.
La position $p_{0,ua}$ de la première instance de τ_{ua} présente dans la période d'activité est :

$$p_{0,ua} = -\left\lfloor \frac{J_{ua} + \Phi_{uac_u}}{T_u} \right\rfloor + 1 \tag{A.13}$$

La longueur L de la période d'activité est donnée par :

$$\begin{aligned} L^{(k)}_{ua,Mixte} &= B_{ua} + \left(\left\lceil \frac{L^{(k-1)}_{ua,Mixte} - \Phi_{uac_u}}{T_u} \right\rceil - p_{0,ua} + 1 \right) C_{ua} + W_{uc_u,Nolin}(\tau_{ua}, L^{(k-1)}_{ua,Mixte}(p)) + \\ & W_{ec_e,Nolin}(\tau_{ua}, L^{(k-1)}_{ua,Mixte}(p)) + \sum_{\forall i \neq e,u} W_{ic_i,Nolin}(\tau_{ua}, L^{(k-1)}_{ua,Mixte}) \end{aligned} \tag{A.14}$$

La position de la dernière instance de τ_{ua} présente dans la période d'activité est donnée par :

$$p_{L,ua} = \left\lfloor \frac{L_{ua,Mixte} - \Phi_{uac_u}}{T_u} \right\rfloor \tag{A.15}$$

Afin de déterminer l'instant de fin d'exécution de l'instance p, nous calculons la longueur de la période d'activité associée à l'instance p :

$$\begin{aligned} L^{(k)}_{ua,Mixte}(p) &= B_{ua} + (p - p_{0,ua} + 1)\, C_{ua} + W_{uc,Nolin}\left(\tau_{ua}, L^{(k-1)}_{ua,Mixte}(p)\right) + \\ & W_{ec_e,Nolin}\left(\tau_{ua}, L^{(k-1)}_{ua,Mixte}(p)\right) + \sum_{\forall i \neq e,u} W_{i,Nolin}\left(\tau_{ua}, L^{(k-1)}_{ua,Mixte}(p)\right) \end{aligned} \tag{A.16}$$

La date d'activation $a_{ua}(p)$ d'une instance de position p est donnée par :

$$a_{ua}(p) = (p-1)T_u + \Phi_{uac_u} \tag{A.17}$$

Le temps de réponse $R_{ua}(p)$ de l'instance p est donné par la différence entre la date de fin d'exécution et celle de son activation :

$$R_{ua}(p) = L_{ua,Mixte}(p) - a_{ua}(p) \tag{A.18}$$

Le temps de réponse de τ_{ua} lorsque une tâche τ_{uc_u} et τ_{ec_e} initient l'instant critique est alors :

$$R_{ua} = \max_{p=p0,ua...pL,ua} R_{ua}(p) \tag{A.19}$$

Le pire temps de réponse est le maximum des temps de réponse obtenus pour chacune des tâches candidates τ_{uc} de Γ_u

$$R_{ua} = \max_{c_u \in hp_u(\tau_{ua}), c_e \in hp_e(\tau_{ua})} \left\{ \max_{p=p0,ua...pL,ua} R_{ua}(p) \right\} \tag{A.20}$$

Annexe B

Algorithmes

B.1 Méthode de Turja-Nolin

Cet algorithme stocke, dans des tables, toutes les fonctions "interférence effective" d'une transaction Γ_i correspondantes à chacune des tâches candidates, puis stocke l'interférence maximum dans deux tables P_i et P_i' (points convexes) puis dans V_i et V_i' (points concaves).

Debut
Calculer J_i^{ind}
Pour c Allant de 1 jusqu'a $|\Gamma_i|$ **faire**
  Opération ordre
  Opération fusion
  Opération séparation
  Calculer P_{ic} **:** tâche séparée n'est pas prise en compte
  Pour j Allant de 1 jusqu'a $|\Gamma_i|$ **faire**
    $P_i[j] = \ interférence$ (équation 2.29)
  Fin Pour
  Calculer P'_{ic} **:** tâche séparée prise en compte
  Opération ordre
  Opération fusion
  Pour j Allant de 1 jusqu'a $|\Gamma_i|$ **faire**
    $P'_i[j] = \ interférence$ (équation 2.32 et 2.29)
  Fin Pour
  $P_i = P_i \ \cup \ P_{ic}$;
  $P'_i = P'_i \ \cup \ P_{ic}$;
Fin Pour
$P_i = \ Envelopper \ (P_i)$;
$P'_i = \ Envelopper \ (P'_i)$;
Calculer V_i (à partir de P_i)
Calculer V'_i (à partir de P'_i)
Fin.

Algorithme 1: Représentation statique de l'interférence maximum d'une transaction Γ_i

L'algorithme 2 présente le calcul approché du pire temps de réponse d'une tâche analysée τ_{ua}. Avant de commencer l'analyse de temps de réponse une représentation statique de l'interférence est effectuée.

Debut
Pour i Allant de 1 jusqu'a $|\Gamma|$ **faire**
  Représentation statique de l'interférence (Algorithme 1) ;
Fin Pour
$R_{ua} = C_{ua}$;

Pour c Allant de 1 jusqu'à $|\Gamma_u|$ **faire**
  Calculer Φ_{uac} ;
  Calculer $p_{0,ua}$;
  Calcul itératif de la longueur $L_{ua,Nolin}$ de la période d'activité :
  Tant que (Non fin) **faire**
    Déduire l'interférence des transactions des tables V_i et V_i' ;
    calculer $L^{(k)}_{ua,Nolin}$; (équation A.6) ;
  Fait
  Calculer $p_{L,ua}$ (équation A.7) ;
  Calcul de temps de réponse de toutes les instances :
  Pour p allant de $p_{0,ua}$ jusqu'à $p_{L,ua}$ **faire**
    Calcul itératif de la longueur $L_{ua,Nolin}(p)$ de la période d'activité associée à p :
    Tant que (Non fin) **faire**
      Déduire l'interférence des transactions des tables V_i et V_i' ;
      Calculer $L^{(k)}_{ua,Noin}(p)$; (équation A.8) ;
    Fait
    $R_{ua}(p) = L_{ua,Nolin}(p) - (p-1)T_u - \Phi_{uac}$;
    $R_{ua} = \max(R_{ua}, R_{ua}(p))$;
  Fin Pour

Fin Pour

Fin.

Algorithme 2: Algorithme du calcul approché du pire temps de réponse d'une tâche τ_{ua} (Turja et Nolin)

B.2 Méthode Mixte

L'algorithme 3 présente le calcul mixte complet du pire temps de réponse d'une tâche analysée τ_{ua}.

Debut
Pour i Allant de 1 jusqu'a $|\Gamma|$ **faire**
  Représentation statique de l'interférence (Algorithme 1) ;
Fin Pour
Pour e Allant de 1 jusqu'à $|\Gamma|$ $(e \neq u)$ **faire**
  $R_{ua}(e) = C_{ua}$;
  Pour c_e Allant de 1 jusqu'à $|\Gamma_e|$ **faire**
    Pour c_u Allant de 1 jusqu'à $|\Gamma_u|$ **faire**
      Calculer Φ_{uac} ;
      Calculer $p_{0,ua}$;
      Calcul itératif de la longueur $L_{ua,Mixte}$ de la période d'activité :
      Tant que (Non fin) **faire**
        Déduire l'interférence des transactions des tables V_i et V_i' ;
        Déduire l'interférence de Γ_e de la table P_{ec_e} et P'_{ec_e} ;
        calculer $L^{(k)}_{ua,Mixte}$; (équation A.14) ;
      Fait
      Calculer $p_{L,ua}$ (équation A.15) ;
      Calcul de temps de réponse de toutes les instances :
      Pour p allant de $p_{0,ua}$ jusqu'à $p_{L,ua}$ **faire**
        Calcul itératif de la longueur $L_{ua,Mixte}(p)$ de la période d'activité associée à p :
        Tant que (Non fin) **faire**
          Déduire l'interférence des transactions des tables V_i et V_i' ;
          Déduire l'interférence de Γ_e de la table P_{ec_e} et P'_{ec_e} ;
          Calculer $L^{(k)}_{ua,Mixte}(p)$; (équation A.16) ;
        Fait
        $R_{ua}(p) = L_{ua,Nolin}(p) - (p-1)T_u - \Phi_{uac}$;
        $R_{ua}(e) = \max(R_{ua}(e), R_{ua}(p))$;
      Fin Pour
    Fin Pour
  Fin Pour
  $R_{ua} = \min(R_{ua}, R_{ua}(e))$;
Fin Pour
Fin.

Algorithme 3: Algorithme du calcul mixte du pire temps de réponse d'une tâche τ_{ua}.

B.3 Transactions AM

Supposons que toutes les tâches de toutes les transactions du système soient plus prioritaires que la tâche sous analyse. L'algorithme suivant effectue le test de l'accumulativité monotonique des transactions d'un système.

Debut
Pour i Allant de 1 jusqu'a $|\Gamma|$ **faire**
  Représentation statique de l'interférence (Algorithme 1) ;
  Vérification de la propriété d'AM de Γ_i :
  $p = P_i[1].c$;
  AM = vrai :
  $j = 2$;
  Tant que (AM and $J \leq |P_i|$) **faire**
    Si ($P_i[j].c \neq p$) **Alors**
      AM = faux
    Fin Si
    $j++$
  Fait
  $j = 1$
  Tant que (AM and $J \leq |P'_i|$) **faire**
    Si ($P'_i[j].c \neq p$) **Alors**
      AM = faux
    Fin Si
    $j++$
  Fait
  Si (AM) **Alors**
    NombreAM++ ;
  Fin Si

Fin Pour

Fin.

Algorithme 4: Algorithme de pré-calcul d'interférence avec vérification de l'AM

B.4 Tâches dominées

Supposons que toutes les tâches de toutes les transactions du système soient plus prioritaires que la tâche sous analyse. Cet algorithme présente le pré-calcul de l'interférence des transactions du système et la détection des tâches dominées.

Debut
Pour i Allant de 1 jusqu'a $|\Gamma|$ **faire**
  Représentation statique de l'interférence (Algorithme 1) ;
  Détection des tâches dominées :
  $TachesDominées_i = \phi$;
  Pour j Allant de 1 jusqu'a $|\Gamma_i| - 1$ **faire**
    Si ($\tau_{ij} \notin TachesDominées_i$) **Alors**
      Pour k Allant de $j + 1$ jusqu'a $|\Gamma_i|$ **faire**
        Si ($\tau_{ik} \notin TachesDominées_i$) **Alors**
          Si (τ_{ij} domine τ_{ik} ou l'inverse) **Alors**
            Inserer la tâche dominée dans l'ensemble $TachesDominées_i$
          Fin Si
        Fin Si
      Fin Pour
    Fin Si
  Fin Pour
  $TachesCrtiques_i$= l'ensemble de tâches de Γ_i - $TachesDominées_i$;
  Si (Taille($TachesCrtiques_i$)=1) **Alors**
    Γ_i est AM ;
    nombreAM++ ;
  Fin Si

Fin Pour

Fin.

Algorithme 5: Algorithme de pré-calcul d'interférence avec détection des tâches dominées

B.5 Algorithme UUniFast

UUnifast est un algorithme efficace, proposé par Bini et Butazzo [19], pour générer des ensembles de tâches avec une distribution uniforme. Cet algorithme génère une valeur $nextSumU$ (charge) pour $n - 1$ variables. Ensuite il met la première charge égale à la différence entre $sumU$ et la valeur générée. $sumU$ contient la somme des charge de $n - 1$ ($nextSumU$) variables, puis on régénère $nextSumU$ la charge de $n - 2$ variables et affectons à la deuxième variable une charge égale à la différence entre $sumU$ et $nextSumU$, et ainsi de suite.

Function vectU = UUniFast(n, U)
$sumU = U$;
Pour i Allant de 1 jusqu'a $n-1$ **faire**
  $nextSumU = sumU \times rand^{(1/(n-i))}$;
  $vectU(i) = sumU$ - $nextSumU$;
  $sumU = nextSumU$;
Fin Pour
$vectU(n) = USum$;

La complexité de cet algorithme est $O(n)$.

Bibliographie liée à l'étude

[83] : A. Rahni and E. Grolleau and M. Richard. An efficient response time analysis for real-time transactions with fixed priority assignment, *International Journal of Systems and Software Engineering (ISSE)*,2008 (à paraître).

[84] : A. Rahni and E. Grolleau and M. Richard. Feasibility analysis of non-concrete real-time transactions with EDF assignement priority, *16th International Conference on Real-Time and Network Systems*, Rennes France, October 16-17, 2008.

[82] : A. Rahni and E. Grolleau and M. Richard. New worst-case response time analysis technique for real-time transactions, *ISoLa Workshop On Leveraging Applications of Formal Methods, Verification and Validation Isola2007* Poitiers France, December 12-14, 2007.

[81] : A. Rahni and E. Grolleau and M. Richard. Méthode d'évaluation du pire temps de réponse de tâches à offset. 5^{eme} *École d'été temps réel ETR2007* Nantes France, 3-7 Septembre, 2007.

[86] : A. Rahni and K. Traore and E. Grolleau and M. Richard. Comparaison of two worst-case response time analysis methods for real-time transactions, *Junior researchers workshop on real-time computing*, March 29-30, 2007.

[114] : K. Traore and E. Grolleau and A. Rahni and M. Richard. Response-time analysis of tasks with offsets, *12th IEEE International Conference on Emerging Technologies and Factory Automation ETFA'06*, September 2006.

[80] : A. Rahni and E. Grolleau and M. Richard. A survey on schedulability analysis of multiframe tasks and tasks with offset. *Rapport interne 2008-002 LISI/ENSMA*, Université de Poitiers".

[85] : A. Rahni and M. Richard and E. Grolleau. Nouvelle approche approximative d'analyse de temps de réponse, *Rapport interne 2007-001 LISI/ENSMA , Université de Poitiers*, 2007.

Bibliographie

[1] Road vehicles -interchange of digital information- controller area network (can) for high speed communications. *ISO International Standard 11898*, 1993.

[2] Road vehicles -low-speed serial data communication- part2 : low-speed controller area network (can). *ISO International Standard 11519-2*, 1994.

[3] Real-time executive for multiprocessor systems, rtems, c applications. *U.S.army missile command, Redstone Arsenal, Alabama 35898-5254*, page 254, 1996.

[4] Osek operating system, version 2.2. *http ://www.osek-vdx.org*, 2002.

[5] Y.-C. L. Altunbasak, Y. Mersereau, and R.M. Multiframe error concealment for mpeg-coded video delivery over error-prone networks. *Image Processing, IEEE Transactions*, Nov 2002.

[6] N. Audsley. Optimal priority assignement and feasibility of static priority tasks with arbitrary start times. *Technical Report YCS-164, University of York*, page 31, 1991.

[7] N. Audsley, A. Burns, R. Davis, K. Tindell, and A. Wellings. Fixed priority preemptive scheduling : an historical perspective. *Real-Time Systems 8*, pages 129–154, 1995.

[8] N. Audsley, A. Burns, M. Richardson, and A. Wellings. Deadline monotonic scheduling theory. *Real-Time Programming*, pages 55–60, 23-26 juin 1992.

[9] T. Baker. Stack-based scheduling of real-time processes. *Journal of Real-Time Systems*, 3 :67–99, 1991.

[10] S. Baruah. *The uniprocessor scheduling of sporadic real-time tasks.* Phd thesis, Department of Computer Science. The University of Texas at Austin., 1993.

[11] S. Baruah. Dynamic- and static-priority scheduling of recurring real-time tasks. *Journal of Real-Time Systems 24(1)*, pages 93–128, 2003.

[12] S. Baruah, D. Chen, S. Gorinsky, and A. Mok. Generalized multiframe tasks. *The International Journal of Time-Critical Computing Systems*, (17) :5–22, 1999.

[13] S. Baruah, D. Chen, and A. Mok. Static-priority scheduling of multiframe tasks. *11th Euromicro Conference on Real-Time Systems*, page p.0038, 1999.

[14] S. Baruah and J. Goossens. *Handbook of scheduling : algorithms models and performance analysis.* Chapman Hall/CRC Press, 2004. Chapter : Scheduling real-time tasks : Algorithms and complexity, ISBN 1-58488-397-9.

[15] S. Baruah, A. Mok, and L. Rosier. The preemptive scheduling of sporadic real-time tasks on one processor. *Proceedings of the 11th Real-Time Systems Symposium*, pages 182–190, 1990.

[16] S. Baruah, L. Rosier, and R. Howell. Algorithms and complexity concerning the preemptive scheduling of periodic real-time tasks on one processor. *The Journal of Real-Time Systems 2*, 1990.

[17] J. Beauquier and B. Bérard. Système d'exploitation- concepts et algorithmes. *McGraw-Hill*, 1990.

[18] I. Benkermi. *Modèle et algorithme d'ordonnancement pour architectures reconfigurables dynamiquement.* Thèse, Université de RENNES 1, Janvier 2007.

[19] E. Bini and G. Buttazzo. Biasing effects in schedulablity measures. *IEEE Proceedings of the 16th Euromicro Conference on Real-Time Systems (ECRTS04), Catania, Italy*, (16), July 2004.

[20] G. Buttazzo. *Hard real-time computing systems : predictable scheduling algorithms and applications.* Kluwer Academic Publisher, Boston, 1997.

[21] M. Chen and K. Lin. Dynamic priority ceiling : a concurrency control for real-time systems. *Real-Time Systems*, 2(4) :325–346, 1990.

[22] S. Cheng, J. Stankovic, and K. Ramamrithan. Scheduling algorithms for hard real-time systems-a brief survey. *Tutorial on Hard Real-Time Systems, IEEE Computer Society Press*, pages 150–73, 1988.

[23] P. Chevochot and I. Puaut. An approach for fault-tolerance in hard-time distributed systems. *Proc. 18th IEEE Symposium on Reliable Distributed Systems (SRDS'99)(short paper), Lausanne, Switzerland*, pages 292–293, October 1999.

[24] P. Chevochot and I. Puaut. Tolérance aux fautes dans les systèmes répartis temps-réel strict. *Techniques et Sciences Informatiques (TSI), 18(8) :837-870*, Octobre 1999.

[25] A. Choquet-Geniet and E. Grolleau. Minimal schedulability interval for real time systems of periodic tasks with offsets. *Theoretical of Computer Sciences*, 310 :117–134, 2004.

[26] F. Cottet. Les systèmes informatiques temps réel. *Cours de maitrise informatique de Poitiers*, 1997.

[27] F. Cottet, J. Delacroix, C. Kaiser, and Z. Mammeri. *Ordonancement temps réel*. Hermès Editions, 2000.

[28] F. Cottet, J. Delacroix, C. Kaiser, and Z. Mammeri. *Scheduling in real-time systems*. Wiley, England, 2002.

[29] F. Cottet and E. Grolleau. *Systèmes temps réel de contrôle-commande : conception et implémentation*. Dunod, Paris, 2005.

[30] D. Decotigny. *Une infrastructure de simulation modulaire pour l'évaluation de performances de systèmes temps-réel*. Thèse, Institut de Recherche en Informatique et Systèmes Aléatoires (IRISA)/INRIA Rennes, Université RENNES I, 07 avril 2003.

[31] M. Dertouzos. Control robotics : the procedural control of physical processors. *Proceedings of the IFIP Congress*, pages 807–813, October 1974.

[32] M. Dertouzos and A. Mok. Multiprocessor on-line scheduling of hard real-time tasks. *IEEE Transactions on Software Engineering, 15(12)*, pages 1497–1506, Dec 1989.

[33] P. Deschizeaux. Temps et évènement dans les systèmes distribués de contrôle de procédé. *R.A.I.R.O. Automatique/Systems Analysis and Control. 16(3) :259-74*, 1982.

[34] R. Devillers and J. Goossens. Liu and layland's schedulability test revisited. *Information Processing Letters*, 73(5-6) :157–161, 2000.

[35] R. Devillers and J. Goossens. Liu and layland's schedulability test revisited 73. *Information Processing Letters*, (6-5) :157–161, 2000.

[36] J. Elloy. Les contraintes du temps réel dans les systèmes industriels répartis. *RGE N2/91*, pages 26–34, Février 1991.

[37] E. Fersman. A generic approach to schedulability analysis of real-time systems. *PhD Thesis, Uppsala University*, December 2003.

[38] L. George, N. Rivièrre, and M. Spuri. Preemptive and non-preemptive real-time uniprocessor scheduling. *Technical Report 2966, INRIA Rocquencourt*, Sep 1996.

[39] A. Girault, H. Kalla, and Y. Sorel. Une heuristique d'ordonnancement et de distribution tolérante aux pannes pour systèmes temps réel embarqués. *Modélisation des Systèmes Réactifs, MSR'03, Metz, France. Hermes*, pages 145–160, Octobre 2003.

[40] B. Girod and M. Flierl. Multi-frame motion-compensated video compression for the digital set-top box. *ICIP, New York*, 2002.

[41] E. Grolleau. *Ordonnancement temps réel hors-ligne optimal à l'aide de réseaux de Petri en environnement monoprocesseur et multiprocesseur*. Thèse, ENSMA-Université Poitiers, 1999.

[42] C. Han and H. Yan. A better polynomial-time scedulability test for real-time fixed-priority scheduling algorithms. *Proc. IEEE Real-Time Systems Symp*, pages 36–45, December 1997.

[43] M. G. Harbour and J. Palencia. Response time analysis for tasks scheduled under edf within fixed priorities. *Real-Time Systems Symposium, RTSS2003*, pages 200–209, December 2003.

[44] D. Harel and A. Pnuelli. On the developpement of reactive systems in logic and models of concurrent systems, nato asi. *Computer Science*, pages 404–498, 1985.

[45] K. Jeffay and D. Stone. Accounting for interrupt handling costs in dynamic priority task systems. *Real-Time Systems Symposium*, pages 212–221, 1993.

[46] J.Engblom and A. Ermedahl. Modeling complex flows for worst-case execution time analysis. *Proceedings of the 21st IEEE Real-Time Systems Symposium (RTSS 2000), Orlando, Florida, USA*, December 2000.

[47] M. Joseph and P. Pandya. Finding response time in a real-time system. *Computer Journal*, 29(5) :390–395, October 1986.

[48] K. Kim and M. Naghibdadeh. Prevention of task overruns in real-time non-preemptive multiprogramming systems. *Proc of Perf*, pages 267–276, 1980.
[49] T. Kuo, L. Chang, Y. Liu, and K. Lin. Efficient on-line schedulability tests for real-time systems. *IEEE Trans. On Software Engineering*, 29-08 2003.
[50] J. Labetoulle. Un algorithme optimal pour la gestion des processus en temps réel. *Revue Francaise d'Automatique, Informatique et Recherche Opérationnelle*, pages 11–17, Fevr 1974.
[51] J. Lehoczky. Fixed priority scheduling of periodic tasks sets with arbitrary deadlines. *Proceedings of Real-Time Systems Symposium*, pages 166–171, 1990.
[52] J. Lehoczky, L. Sha, and Y. Ding. The rate monotonic scheduling algorithm : exacte characterisation and average case behaviour. *Proceedings of the Real-Time Systems Symposium*, 1989.
[53] J. Lehoczky, L. Sha, J. Strosnider, and H. Tokuda. Fixed priority scheduling theory for hard real-time systems. *Foundations of Real-Time Computing : scheduling and resource management, Kluwer Acadamic Publishers*, pages 1–30, 1991.
[54] J.-T. Leung and J. Whitehead. On the complexity of fixed-priority scheduling of periodic real-time tasks. *Performance Evaluation, 2*, pages 237–250, 1982.
[55] C. Liu and J. Layland. Scheduling algorithms for multiprogramming in real-time environnement. *Journal of ACM*, 1(20) :46–61, October 1973.
[56] J. Liu. *Real-time systems*. Prentice Hall, 2000.
[57] W.-C. Lu, K.-J. Lin, H.-W. Wei, and W.-K. Shih. New schedulability conditions for real-time multiframe tasks. *19th Euromicro Conference on Real-Time Systems (ECRTS07)*, pages 39–50, 2007.
[58] J. Maki-Turja, K. Hannien, and M. Nolin. Efficient development of real-time systems using hybrid scheduling. *International conference on embedded systems and applications (ESA)*, June 2005.
[59] J. Maki-Turja and M. Nolin. Improved analysis for real-time tasks with offsets- advanced model. *MRTS Report, Malardalen Real-Time Research Center, Malardalen University*, May 2003.
[60] J. Maki-Turja and M. Nolin. Faster response time analysis of tasks with offsets. *Proc 10th IEEE Real-Time Technology and Applications Symposium (RTAS)*, May 2004.
[61] J. Maki-Turja and M. Nolin. Tighter response time analysis of tasks with offsets. *Proc 10th International Conference on Real-Time Computing and Applications (RTCSA'04)*, August 2004.
[62] J. Maki-Turja and M. Nolin. Fast and tight response-times for tasks with offsets. *17th EUROMICRO Conference on Real-Time Systems IEEE Palma de Mallorca Spain*, July 2005.
[63] J. Maki-Turja and M. Nolin. Efficient implementation of tight response-times for tasks with offsets. *Real-Time Systems Journal, Springer Netherlands*, 16 February 2008.
[64] P. Martineau. *Ordonnancement en-ligne dans les systèmes informatiques temps réel*. Thèse, Ecole Centrale de Nantes, 1994.
[65] A. Mok. *Fundamental design problems of distributed systems for the hard-real-time environment.* Phd thesis, Laboratory for Computer Science. Massachusetts Institute of Technology. Available as Technical Report No. MIT/LCS/TR-297., 1983.
[66] A. Mok. *Fundamental design problems for the hard real-time environments*. Phd, MIT, May 1983.
[67] A. Mok and D. Chen. A multiframe model for real-time tasks. *Proceeding of the 17th Real-Time Systems Symposium, Washington*, pages 22–29, December 1996.
[68] A. Mok and D.Chen. A multiframe model for real-time tasks. *IEEE Transactions on software Engineering*, pages 635–645, October 1997.
[69] J. Palencia, J. G. Garcia, and M. G. Harbour. On the schedulability analysis for distributed hard-time systems. *Proceedings of the 9th Euromicro Workshop on Real-Time Systems, Toledo, Spain*, pages 136–143, Jaun 1997.
[70] J. Palencia and M. G. Harbour. Schedulability analysis for tasks with static and dynamic offsets. *Proc IEEE Real-time System Symposium (RTSS)*, (19), December 1998.
[71] J. Palencia and M. G. Harbour. Offset-based response time analysis of distributed systems scheduled under edf. *Euromicro conference on real-time systems. Porto, Portugal*, June 2003.
[72] C. Park and A. Shaw. Experiment with a program timing tool based on source-level timing schema. *IEEE Computer*, 24(5) :48–57, 1991.
[73] R. Pellizzoni. *Efficient feasibility analysis of real-time asynchronous task sets*. Master's thesis, Università di Pisa, 2004.

[74] R. Pellizzoni and G. Lipari. A new sufficient feasibility test for asynchronous real-time periodic task sets. *Proceedings of the 16th Euromicro Conference on Real-Time Systems, Catania, Italy*, 2004.

[75] R. Pellizzoni and G. Lipari. Improved schedulability analysis of real-time transactions with earliest deadline scheduling. *Proceedings of the 11th IEEE Real Time and Embedded Technology and Applications Symposium (RTAS05)*, pages 66–75, 2005.

[76] R. Pellizzoni and G. Lipari. Improved schedulability analysis of real-time transactions with earliest deadline scheduling. *Journal of Computer and System Sciences*, 2006.

[77] T. Pop, P. Eles, and Z. Peng. Schedulability analysis for distributed heterogenous time/event triggered real time systems. *Proceedings of the 15th Euromicro conference on real-time systems*, July 2003.

[78] P. Puschner and R. Nossal. Testing the results of static worst-case execution-time analysis. *Proc IEEE Real-Time Systems Symposium, Madrid, Spain*, 1 :134–143, December 2-4 1998.

[79] P. Pusher and C. Koza. Calculating the maximum execution time of real-time programs. *Real-Time Systems*, 1(2) :159–176, December 1989.

[80] A. Rahni, E. Grolleau, and M. Richard. A survey on schedulability analysis of multiframe tasks and tasks with offset. *Rapport interne 2008-002 LISI/ENSMA, Université de Poitiers.*

[81] A. Rahni, E. Grolleau, and M. Richard. Méthode d'évaluation du pire temps de réponse de tâches à offset. *5 eme Ecole d'été temps réel ETR2007 Nantes France*, 3-7 Septembre 2007.

[82] A. Rahni, E. Grolleau, and M. Richard. New worst-case response time analysis technique for real-time transactions. *ISoLa Workshop On Leveraging Applications of Formal Methods, Verification and Validation Isola2007 Poitiers France*, December 12-14 2007.

[83] A. Rahni, E. Grolleau, and M. Richard. An efficient response time analysis for real-time transactions with fixed priority assignment. *International Journal of Systems and Software Engineering (ISSE) (à paraître)*, 2008.

[84] A. Rahni, E. Grolleau, and M. Richard. Feasibility analysis of non-concrete real-time transactions with edf assignement priority. *16th International Conference on Real-Time and Network Systems, Rennes, France*, October 16-17 2008.

[85] A. Rahni, M. Richard, and E. Grolleau. Nouvelle approche approximative d'analyse de temps de réponse. *Rapport interne 2007-001 LISI/ENSMA , Université de Poitiers.*

[86] A. Rahni, K. Traore, E. Grolleau, and M. Richard. Comparaison of two worst-case response time analysis methods for real-time transactions. *Junior researchers workshop on real-time computing*, March 29-30 2007.

[87] J. Regher, A. Reid, K. Webb, M. Parker, and J. Lepreau. Evolving real-time systems using hierarchical scheduling and concurrency analysis. *Proceedings of the 24th IEEE real-time systems symposium (RTSS), IEEE Computer Society, Los Alamitos*, December 2003.

[88] M. Richard. *Contribution à la validation des systèmes temps réel distribués : ordonnancement à priorités fixe et placement.* Thèse, ENSMA-Université Poitiers, 2002.

[89] P. Richard. Analyse du temps de réponse des systèmes temps réel. *Actes de l'Ecole d'été Temps Réel*, 2003.

[90] P. Richard. Analyse du temps de réponse et de la demande processeur en ordonnancement temps réel de tâches périodiques. *Ecole d'été Temps Réel (ETR'05)*, 2005.

[91] F. Ridouard, F. C. P. Richard, and K. Traore. Some results on scheduling tasks with self-suspensions. *Journal of Embedded Computing*, 2006.

[92] F. Ridouard, P. Richard, and F. Cottet. Negative results for schedulings independent hard real-time tasks with self-suspensions. *Proceedings of the 25th IEEE International Real-time Systems Symposium (RTSS04)*, December 2004.

[93] F. Ridouard, P. Richard, and F. Cottet. Scheduling independent tasks with self-suspension. *Proceedings of the 13rd RTS Embedded Systems(RTS05)*, (13), April 2005.

[94] S. Saad-Bouzefrane. *Etude temporelled'application temps réel distribuées à contraintes strictes basée sur analyse d'ordonnançabilité.* Thèse, Ecole Nationale Superieure de Mécanique et d'Aérotechnique, 1998.

[95] O. Serlin. Scheduling of time critical processes. *Proc. Spring Joint Computers Conference*, pages 925–932, 1972.

[96] L. Sha, T. Abdelzaher, K.-E. Arzen, A. Cervin, T. Baker, A. Burns, G. Buttazzo, M. Caccamo, J. Lehoczky, and A. Mok. Real time scheduling theory : a historical perspective. *Real-Time Syst 28(2/3)*, pages 101–155, 2004.
[97] L. Sha, R. Rajkumar, and J. Lehoczky. Priority inheritance protocols : an approach to real-time synchronization. *Tech. Rep. CMU-CS-87-181, Carnegie-Mellon*, page 23, 1987.
[98] L. Sha, R. Rajkumar, and J. Lehoczky. Priority inheritance protocols : an approach to real-time synchronization. *IEEE Transactions on Computers*, 39(9) :1175–1185, 1990.
[99] A. Shaw. Reasoning about time in higher-level language software. *IEE Transactions on Software Engineering*, 15(7) :875–889, December 1989.
[100] M. Sjodin and H. Hansson. Improved response-time analysis calculations. *Real-Time Systems Symposium*, pages 399–408, 1998.
[101] B. Sprunt, L. Sha, and J. Lehoczky. Aperiodic tasks scheduling for real-time systems. *Journal of Real-Time Systems*, 1 :27–60, Jan 1989.
[102] M. Spuri. Analysis of deadline scheduled real-time systems. *RR-2772, INRIA, France*, 1996.
[103] M. Spuri. Holistic analysis of deadline scheduled real-time distributed systems. *RR-2873, INRIA, France*, 1996.
[104] J. Stankovic. Misconceptins about real-time computing. *IEEE Computer*, pages 10–19, 1988.
[105] H. Takada and K. Sakamura. Schedulability of generalized multiframe task sets under static pririty assignment. *Proceedings-Fourth International Workshop on Real-Time Computing Systems and Applications*, (27-29) :80–86, October 1997.
[106] K. Tindell. Using offset information to analyse static priority pre-emptively scheduled task sets. *Technical Report YCD-182,Dept of Computer Science, Unoversity of York, England*, 1992.
[107] K. Tindell. Adding time-offsets to schedulability analysis. *Technical Report YCS 221, Dept of Computer Science, University of York, England*, January 1994.
[108] K. Tindell. *Fixed priority scheduling of hard real-time systems*. Thèse, University of York, 1994.
[109] K. Tindell, A. Burns, and A. Wellings. An extensible approach for analysing fixed priority hard real-time tasks . *YCS 189 Department of Computer Science, University of York*, 1992.
[110] K. Traore. *Analyse et validation des applications temps Réel en présence de transactions : application au pilotage d'un drone miniature*. Thèse, ENSMA-Université Poitiers, 2007.
[111] K. Traore, E. Grolleau, and F. Cottet. Characterization and analysis of tasks with offsets : monotonic transactions. *Proc 12th International Conference on Embedded and Real-Time Computing Systems and Applications. RTCSA'06, Sydney, Australia*, (12), August 16-18th 2006.
[112] K. Traore, E. Grolleau, and F. Cottet. Shedulability analysis of serial transactions. *Real-Time and Network Systems RTNS'06 Poitiers France*, pages 141–149, May 30-31 2006.
[113] K. Traore, E. Grolleau, and F. Cottet. Simpler analysis of serial transactions using reverse transactions. *Conference on Autonomic and Autonomous Systems, ICAS 06, Silicon Valley, USA, IEEE Computer Society Press*, July 19-21 2006.
[114] K. Traore, E. Grolleau, A. Rahni, and M. Richard. Response-time analysis of tasks with offsets. *12th IEEE International Conference on Emerging Technologies and Factory Automation ETFA'06*, September 2006.
[115] Y. Trinquet and J. Elloy. Systèmes d'exploitation temps réel-principes. *Techniques de l'ingénieur R-8 050-7*.
[116] J. Xu and D. Parnas. On satifying timing constraints in hard-real-time systems. *Software Engineering Notes, 16(4) :132-46*, December 1991.
[117] L. Zaffalon. *Programmation concurrente et temps réel*. Presses Polytechniques et Universitaires Romandes(PPUR), 2007.
[118] A. Zuhily. Exact response time analysis for multiframe tasks. *Technical Report YCS-2007-410,Dept of Computer Science, University of York, England*, 2007.

Résumé

Un système temps réel critique nécessite une validation temporelle utilisant un test d'ordonnançabilité avant sa mise en œuvre. Cette thèse traite le problème d'ordonnancement des taches à offset (transactions) sur une architecture monoprocesseur, en priorités fixes et en priorités dynamiques. Les méthodes existantes pour un test exact ont une complexité exponentielle et seules existent des méthodes approchées, donc pessimistes, qui sont pseudo-polynomiales. En priorités fixes nous proposons des méthodes pseudo-polynomiales, basées sur l'analyse de temps de réponse qui sont moins pessimistes que les méthodes existantes. Nous présentons quelques propriétés (accumulativité monotonique, dominance de tâches) rendant exacte les méthodes d'analyse approchées pour certains cas de systèmes, et optimisant le temps de calcul. En priorités dynamiques, nous proposons un test d'ordonnançabilité exact avec une complexité pseudo-polynomiale. Ce test est basé sur l'analyse de la demande processeur. Les qualités des résultats de nos méthodes sont confirmées par des évaluations expérimentales.

Mots-clés: Systèmes temps réel, ordonnancement, monoprocesseur, tâches à offsets, transactions, tâches multiframes, Analyse de temps de réponse, demande processeur, transactions Accumulativement monotonique, tâches dominées.

Abstract

The validation process is an important step in the development of a real-time application. It consists in proving that, whatever happens, the scheduling policy guarantees that all the temporal constraints are met. In this thesis we study the schedulability test of tasks with offsets, on uniprocessor system, with fixed and dynamic priority assignments. The exact tests has an exponential complexity and only a sufficient (pessimistic) feasibility test with a pseudo-polynomial complexity has been proposed. Note that all existing tests are based on the response time analysis (RTA) technique. In the fixed priority context, we propose a new pseudo-polynomial RTA method, less pessimistic than the existing methods. We present some properties (accumulatively monotonic transactions, dominated tasks) that make exact the approximate methods, for certain cases, and optimize the run time computation. In the dynamic priority context, we provide a pseudo-polynomial exact schedulability test, based on the processor demand analysis. We provide experimental evaluations of the proposed tests and their optimization.

Keywords: Real-Time systems, scheduling, uniprocessor, tasks with offsets, transactions, multiframe tasks, response-time analysis, processor demand, Accumulatively monotonic transactions, dominated task.

Secteur de recherche: Informatique

LABORATOIRE D'INFORMATIQUE SCIENTIFIQUE ET INDUSTRIELLE
Ecole Nationale Supérieure de Mécanique et d'Aérotechnique
Téléport2- 1, avenue Clément Ader -BP 40109 - 86961 Chasseneuil-Futuroscope Cédex
Tél: 05.49.49.80.63 - Fax: 05.49.49.80.64