L. David and F. Cottet, Acte de Real Time Systems (RTS'01), Ed. Teknea, Paris, France, 2001, pp. 167-187

Traitement de la gigue de tâches dépendantes dans un contexte d'ordonnancement temps réel en ligne.

L. David & F. Cottet
{david@ensma.fr} {cottet@ensma.fr}

LISI-ENSMA
BP 40109
1, Av. Clément Ader - Téléport 2
86961 Futuroscope Cedex France

Résumé : L'ordonnancement en ligne d'une application temps réel permet rarement la maîtrise de la régularité d'exécution des tâches périodiques. Cela peut s'avérer néfaste lorsque ces tâches doivent respecter strictement leur période (acquisition de données, commande d'un actionneur, etc.). Après avoir défini formellement les deux types d'irrégularités d'exécution que nous pouvons rencontrer dans les applications temps réel, nous nous proposons dans ce travail de fournir une technique de manipulation des attributs temporels des tâches périodiques, en conservant les algorithmes d'ordonnancement DM et EDF, et en garantissant l'annulation des gigues ou leur encadrement. De plus, cette étude prend en compte les configurations de tâches dépendantes.

Mots clés : Temps réel, ordonnancement en ligne, gigue temporelle, tâches périodiques dépendantes.

1 Introduction

Nous allons nous intéresser dans ce travail au temps réel à contraintes strictes, dans lequel nous adoptons le principe de découpage de l'application en un système de tâches. Parmi les contraintes temporelles qui reposent sur ces tâches, nous allons plus particulièrement étudier celles qui ont trait à la régularité d'exécution. Beaucoup d'exemples de systèmes temps réel présentent en effet de telles contraintes (systèmes de pilotage d'un procédé, les trames vidéos des systèmes multimédia, robotique de précision, etc... [HLH96, LH96]). Les périodes des tâches liées à des actionneurs ou à des acquisitions de données se doivent donc, sous certaines conditions et pour des besoins particuliers, d'être précisément respectées.

Dans un contexte d'exécutions concurrentes comme le temps réel, ceci n'est pas toujours le cas. En effet, toutes les politiques d'ordonnancement ne prennent pas en compte la traduction de cette contrainte temporelle. Dans le modèle classique de Liu et Layland [LL73], largement utilisé, les ordonnancements en ligne sont principalement orientés respect des échéances ; en effet, chaque exécution d'une tâche sur une de ses périodes est totalement indépendante, en terme d'ordonnancement, de l'exécution de cette même tâche sur une autre de ses périodes.

Dans la littérature, il existe de nombreux qualificatifs pour nommer l'irrégularité d'exécution qui en découle. On parle de gigue temporelle, de gigue, de délais imprévisibles entre deux exécutions, de distances temporelles non constantes, etc... [But97, DS95, LH96]. Il faut en outre distinguer la gigue spécifique à l'exécution d'une tâche, et la gigue de bout en bout d'un système, celle-ci mesurant en fait la variation du temps de réponse du système à un événement. Quelques méthodes d'ordonnancement, principalement du type hors ligne, existent aujourd'hui pour contrôler ce phénomène : une technique d'ordonnancement dans le cas réparti basée sur le principe du recuit simulé a ainsi été présentée par DiNatale et Stankovic dans [DS95]. Une autre solution hors-ligne au problème du respect de la cadence d'échantillonnage, se traduit par une recherche exhaustive des séquences valides respectant un critère de minimisation de la gigue [CC96, CGC96]. Lin et al. proposent enfin un modèle dans lequel les contraintes de régularité d'exécution sont prises en compte avec des algorithmes d'ordonnancement bien spécifiques [HLH96, LH96].

Dans ce travail, après avoir défini formellement les deux types de gigue pouvant être rencontrés, nous présentons un principe de traitement de la gigue dans un contexte en ligne. Pour mettre en place ce traitement, nous présentons alors un ensemble d'outils de manipulation des attributs temporels, qui nous seront nécessaires dans les sections suivantes pour décrire une technique d'annulation, ou d'encadrement de cette gigue.

Dans cette étude, nous nous plaçons dans l'hypothèse d'un environnement mono-processeur sur lequel sont exécutées des tâches périodiques, pouvant être liées par des contraintes de précédence. Nous utilisons par ailleurs le modèle de tâche de Liu et Layland [LL73] (cf. figure 1) dans lequel chaque tâche est caractérisée par 4 paramètres : $r_{i,1}$, la date de première activation, C_i, la pire durée d'exécution, D_i, son délai critique, et T_i sa période.

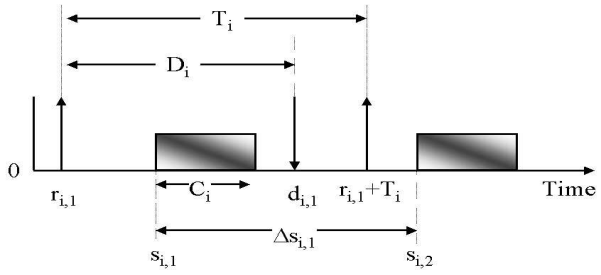

FIG. 1 – Modèle de tâche utilisé.

2 La gigue, définitions

Dans cette section, nous présentons deux types de gigues : la gigue de régularité d'une tâche et la gigue de bout en bout qui peut se définir sur un ensemble de tâches formant une activité ($< \tau_{capteur}, ..., \tau_{actionneur} >$), ou simplement se définir sur une seule tâche (τ_i), désignant dans ce cas une gigue de cohésion.

2.1 Gigue de régularité

La définition de cette gigue se fait sur la période d'étude de l'ordonnancement (H), qui représente en fait un motif d'ordonnancement des tâches qui est indéfiniment répété pendant l'exécution de l'application temps réel. Soit maintenant $s_{i,k}$ (respectivement $e_{i,k}$) la date de début d'exécution (respectivement de fin d'exécution) de la k^{ieme} instance d'une tâche τ_i (cf. figures 1 et 2).

La durée séparant les débuts d'exécution (respectivement les fins d'exécution) de deux instances successives d'une même tâche est définie par la différence :

$$\Delta s_{i,k} = s_{i,k+1} - s_{i,k} \; (resp. \; \Delta e_{i,k} = e_{i,k+1} - e_{i,k})$$

où k décrit la k^{ieme} différence numérotée dans la période d'étude. $\Delta s_{i,k}$ et $\Delta e_{i,k}$ ont comme valeurs extrêmes 0 et $2T_i$ (dans le cas de tâche à échéance sur requête avec des durées d'exécution négligeables). Introduisons ensuite :

$$j_{s_{i,k}} = \frac{|\Delta s_{i,k} - T_i|}{T_i} 100 \; \% \tag{2.1}$$

qui mesure, en pourcentage par rapport à T_i, l'irrégularité des débuts d'exécution entre la k^{ieme} instance et la $k+1^{ieme}$ instance :

- si la différence vaut T_i (tâche régulière), alors : $j_{s_{i,k}} = 0 \; \%$,
- si la différence tend vers 0 ou vers $2T_i$ (maximum d'irrégularité dans l'exécution), alors : $j_{s_{i,k}} = 100 \; \%$.

Nous obtenons bien sûr les mêmes expressions et les mêmes remarques en utilisant $e_{i,k}$. Étant donnée une tâche τ_i, la moyenne arithmétique de ces différences conduit à la définition suivante :

Définition 1 *Soit une tâche $\tau_i(r_{i,1}, C_i, D_i, T_i)$ dont les différentes fins d'exécution de ses instances sont repérées par $s_{i,k}$ ($k \in \mathbb{N}^*$). Le pourcentage de gigue temporelle moyenne de la tâche τ_i est caractérisé par :*

$$J_{Moy}(\tau_i) = \frac{1}{N_i} \sum_{k=1}^{N_i} \frac{|\Delta s_{i,k} - T_i|}{T_i} 100 \; \%, \tag{2.2}$$

où N_i est le nombre d'instances de la tâche τ_i sur une durée de H (sur la figure 2, $N_i = 4$).

Par ailleurs, partant de la variable $j_{s_{i,k}}$, plusieurs analyses de la gigue peuvent être faites : par exemple, le pourcentage de gigue temporelle maximum, minimum, etc... De plus, les conventions adoptées dans ce paragraphe peuvent ne pas être adaptées à certains types d'application. Aussi, à travers cette définition et cette approche, nous proposons juste une solution, parmi d'autres, pour mesurer l'irrégularité moyenne d'exécution d'une tâche.

2.2 Gigue de bout en bout

Une gigue de bout en bout d'un système temps réel mesure en fait les variations du temps de réponse du système à un événement. Nous allons donc nous intéresser à ces variations au travers les durées séparant le début d'exécution d'une instance, et la fin d'exécution d'une instance qui peut appartenir à une autre tâche. Ces durées seront notées $TR_k(a)$ où a désigne l'activité considérée.

Considérons maintenant une activité $a =< \tau_i, \tau_j >$ constituée pour l'exemple (cf. figure 3-a) de deux tâches dont les périodes ne sont pas nécessairement identiques (sauf si les tâches sont dépendantes

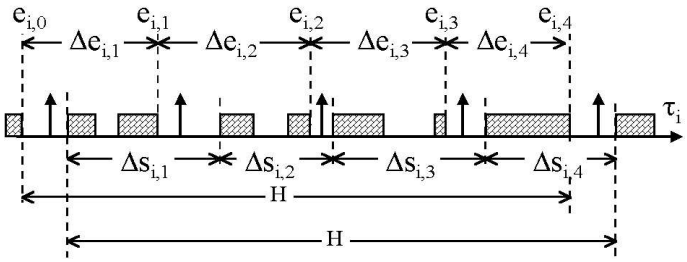

FIG. 2 – La gigue de régularité

[BB99]). Pour être le plus à jour possible dans notre traitement des données de τ_i, on choisit l'instance de τ_i qui est la plus récente pour définir la dernière valeur $TR_3(a)$ de la figure 3-a. Les mesures à rassembler sur une période d'étude, sont donc les durées entre le début d'exécution de l'instance la plus récente en terme d'exécution (τ_i), et la fin d'exécution de l'instance qui lui correspond (τ_j) ($TR_k(a)$ avec $1 \leq k \leq 3$ sur la figure 3-a).

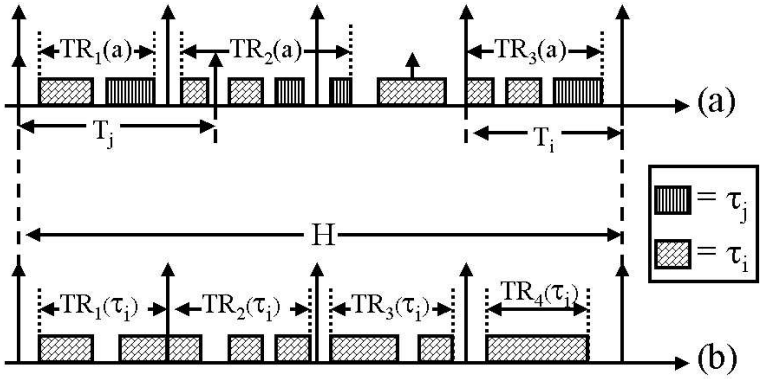

FIG. 3 – (a) : gigue de bout en bout d'une activité $a =< \tau_i, \tau_j >$ - (b) : gigue de cohésion d'une tâche.

Pour la gigue de cohésion il suffit de rassembler l'ensemble des durées séparant le début d'exécution d'une instance et sa fin d'exécution ($TR_k(\tau_i)$ avec $1 \leq k \leq 4$ sur la figure 3-b).

Nous devons ensuite comparer ces TR_k à une valeur de "référence", qui peut être en fait la moyenne arithmétique des TR_k, mais qui peut aussi être une donnée provenant du domaine de l'automatique, à savoir, le temps de réponse moyen qui correspond à une commande stable. Notons TR_{Moy} cette référence, et considérons alors l'ensemble des différences suivantes :

$$\Delta_k = \frac{|TR_k - TR_{Moy}|}{TR_{Moy}} 100\ \%$$

qui sont en fait les pendants des $j_{s_{i,k}}$ définies par l'équation 2.1. On applique alors le même principe de formalisation et on en vient à la définition suivante :

Définition 2 *Soit une activité $a =< \tau_i, ..., \tau_j >$ dont les durées séparant le début d'exécution d'une instance de τ_i et la fin d'exécution de l'instance de τ_j lui correspondant, sont repérées par $TR_k(a)$ ($k \in \mathbb{N}^*$). Le pourcentage de gigue moyenne de bout en bout de l'activité a est caractérisé par :*

$$J_{Moy}(a) = \frac{1}{N_a} \sum_{k=1}^{N_a} \frac{|TR_k(a) - TR_{Moy}(a)|}{TR_{Moy}(a)} 100\ \%, \tag{2.3}$$

où N_a est le nombre d'activités exécutées sur une durée de H (cf. figure 3-a, $N_a = 3$). Dans le cas d'une activité comportant qu'une seule tâche τ_i, ces durées sont caractérisées par le début d'exécution

et la fin d'exécution d'une seule instance, et on définit le pourcentage de gigue moyenne de cohésion de la tâche τ_i par ($N_i = 4$ sur figure 3-b) :

$$J_{Moy,c}(\tau_i) = \frac{1}{N_i} \sum_{k=1}^{N_i} \frac{|TR_k(\tau_i) - TR_{Moy}(\tau_i)|}{TR_{Moy}(\tau_i)} 100\ \%. \tag{2.4}$$

On peut en outre faire les mêmes remarques que celles du paragraphe précédent concernant cette présentation que nous voulions générale, et que nous pouvons adapter à des contextes d'applications temps réel particuliers. Nous ne traiterons dans ce travail que la gigue de régularité. Toutefois, la gigue de cohésion sera prise en considération dans la dernière section.

3 Méthodologie de traitement de la gigue

Décrivons les notations et les conventions utilisées dans cette étude. On considère deux ensembles de tâches disjoints, notés R et NR, dont l'union décrit l'ensemble des tâches d'une application temps réel. Le premier ensemble R correspond aux tâches que nous voulons régulières en terme d'exécution, quant au second, il décrit les tâches non nécessairement régulières. On notera en outre par n_R le cardinal de R. On suppose de plus que le paramètre période des tâches considérées n'est pas modifiable : il est en effet souvent conditionné par la dynamique du procédé et fait donc parti des contraintes temporelles de base du cahier des charges. Enfin, on suppose que la redéfinition éventuelle d'une date d'activation n'entraîne pas une redéfinition du délai critique : les échéances absolues de la configuration initiale ne sont donc pas nécessairement respectées pour les tâches dont on redéfinit les dates d'activation.

3.1 Principe général

L'idée principale de notre technique de traitement de la gigue peut se résumer ainsi : si deux tâches périodiques que nous souhaitons régulières en terme d'exécution possèdent des fenêtres d'exécution à intersection non nulle, alors indubitablement, l'une d'entre elles présentera de la gigue de régularité, ou de cohésion.

Illustrons cette idée à l'aide d'une configuration de tâches (τ_1, τ_2, τ_3) (cf. tableau 1) que nous ordonnançons selon les priorités externes ($Prio_i$). La séquence d'ordonnancement est alors reportée dans la figure 4 où nous remarquons que τ_2 est perturbée dans son exécution par τ_1, et que τ_3 l'est aussi par τ_1 puis τ_2 : les fenêtres d'exécution de ces tâches sont en effet à intersection non nulle, conduisant à la gigue des tâches τ_2 et τ_3. Une solution intéressante pour supprimer ces gigues se base sur la désynchronisation des dates d'activation. Le triplet $(r_{1,1}, r_{2,1}, r_{3,1}) = (1, 0, 6)$ entraîne en effet une intersection nulle des fenêtres d'exécution, conduisant à l'annulation des gigues (cf. figure 5).

TAB. 1 – Exemple de configuration de tâche.

	$r_{i,1}$	C_i	D_i	T_i	$Prio_i$
τ_1	0	3	16	16	3
τ_2	0	1	4	4	2
τ_3	0	2	8	8	1

Plus généralement, le principe de notre méthode va ainsi consister à désynchroniser les intervalles d'exécution des tâches régulières en jouant d'une part, sur les dates d'activation, et d'autre part, sur les délais critiques, ceci en tenant compte des éventuelles contraintes de précédence. Cette démarche est ainsi résumée par quatre étapes que nous particulariserons pour le cas de l'annulation de la gigue, ou le cas de son encadrement.

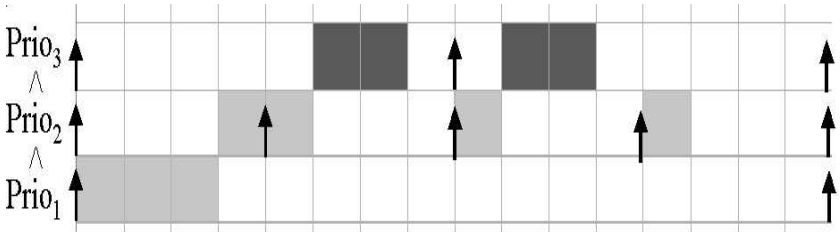

FIG. 4 – Perturbation de τ_2 par τ_1, de τ_3 par τ_1 puis τ_2 : gigue sur τ_2 et τ_3.

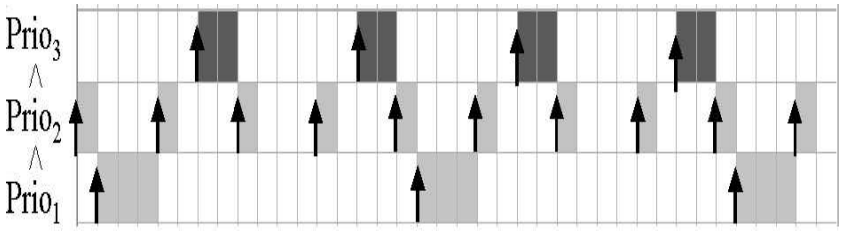

FIG. 5 – Séquence d'ordonnancement sans gigue.

1. On recherche dans un premier temps les dates d'activations des tâches régulières τ_i de sorte que si elles démarrent leurs exécutions aux dates d'activation et les terminent W_i unités de temps processeur plus tard ($W_i \in \mathbb{N}^*$, $0 < W_i \leq D_i$), alors nous garantissons une exécution sans chevauchement (W_i correspond à la largeur de la fenêtre d'exécution de la tâche).
2. On fait ensuite les choix définitifs pour les dates d'activation en fonction des contraintes de précédence sur les dates d'activation et en fonction des solutions possibles de l'étape 1. On exprime également les conditions de précédence sur les délais critiques qui conduisent alors à un ensemble de possibilités.
3. On traduit ensuite la propriété selon laquelle on souhaite des intervalles d'exécution d'une durée maximum de W_i pour chaque tâche régulière τ_i, propriété qui permet alors de faire nos choix définitifs sur les délais critiques en prenant en compte ceux proposés dans l'étape 2.
4. Dans la quatrième étape enfin, nous devons vérifier si la nouvelle configuration de tâche proposée est toujours ordonnançable dans le contexte d'ordonnancement fixé.

Cette méthode peut être implémentée de deux façons différentes : soit on souhaite annuler totalement la gigue de régularité et dans ce cas, on va interdire la préemption des tâches de R par les tâches de NR en posant $W_i = C_i$, soit on souhaite simplement l'encadrer, auquel cas, nous allons autoriser des préemptions dans une certaine limite, fixée par la fenêtre de largeur W_i.

3.2 Outils nécessaires au traitement de la gigue

Trois outils sont nécessaires au traitement de la gigue : le premier s'intéresse au décalage des intervalles d'exécution des tâches de R, ceci au travers d'une redéfinition des dates de première activation. Le deuxième outil prend en compte, si nécessaire, le problème des précédences. Le troisième outil enfin s'intéresse à redéfinir les priorités des tâches.

3.2.1 Désynchronisation des dates d'activation

Soit $(\tau_1, ..., \tau_n)$ un ensemble de tâches à désynchroniser. Pour toutes les tâches τ_i, on définit un paramètre W_i, qui caractérise depuis les dates d'activation des instances de τ_i, le délai au terme duquel, il est prévu que l'exécution de l'instance de la tâche soit terminée (cf. figure 6). Le problème de désynchronisation des fenêtres d'exécution, dont l'obtention est détaillée dans [DC00, DCG00b],

revient alors à chercher un n-uplet $(r_{1,1}, ..., r_{n,1})$ qui vérifie[1] :

$$\forall (i,j) \in [|1;n|]\ i < j$$

$$\begin{array}{c} r_{i,1} - r_{j,1} \neq 0\, mod(T_i \wedge T_j) \\ et \\ \left(\begin{array}{c} r_{i,1} - r_{j,1} \neq 1\, mod(T_i \wedge T_j) \\ r_{i,1} - r_{j,1} \neq 2\, mod(T_i \wedge T_j) \\ ... \\ r_{i,1} - r_{j,1} \neq W_j - 1\, mod(T_i \wedge T_j) \end{array} \right) \ et \ \left(\begin{array}{c} r_{j,1} - r_{i,1} \neq 1\, mod(T_i \wedge T_j) \\ r_{j,1} - r_{i,1} \neq 2\, mod(T_i \wedge T_j) \\ ... \\ r_{j,1} - r_{i,1} \neq W_i - 1\, mod(T_i \wedge T_j) \end{array} \right) \end{array} \quad (3.1)$$

où T_i et T_j sont nécessairement non premiers entre eux.

Si on trouve ainsi un n-uplet vérifiant ces conditions, et si les durées des intervalles d'exécution des tâches τ_i $(1 \leq i \leq n)$ sont égales à W_i, alors les fenêtres d'exécution sont garanties de ne pas se chevaucher.

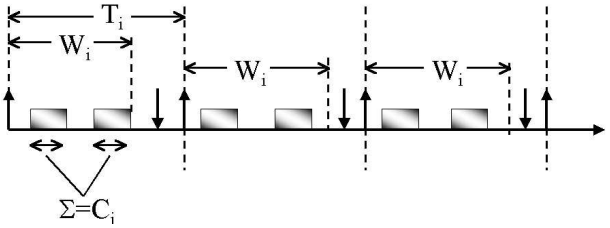

FIG. 6 – Définition des paramètres pour le problème de désynchronisation.

Ce problème est complexe surtout lorsque le nombre n augmente, aussi, même si des outils analytiques nous permettent de préciser des solutions dans les conditions particulières (cf. [DCG00b]), nous faisons le plus souvent appel à une technique d'énumération de toutes les solutions (cf. [DCG00a, DC00]).

3.2.2 Prise en compte des contraintes de précédence

La contrainte de base d'une activité consiste en une exécution d'une séquence de tâches dans un ordre prédéfini. En général, les tâches mises en jeu sont dépendantes les unes des autres et ont donc toutes une même période T qui est alors aussi celle de l'activité. Dans cette étude, nous supposons que c'est le cas, et nous choisissons pour les précédences les conditions contraignant le moins les attributs temporels, ainsi, selon [Bla76], pour que τ_i précède τ_j, il suffit que :

$$\left\{ \begin{array}{c} r_{i,1} \leq r_{j,1} \\ Prio_i > Prio_j \end{array} \right. . \quad (3.2)$$

La deuxième condition sur les priorités est bien sûr conditionnée par l'algorithme d'ordonnancement choisi, ainsi, en DM par exemple , nous avons : $D_i < D_j$.

3.2.3 Redéfinitions des priorités

Dans ce paragraphe, on souhaite redéfinir les priorités des tâches de R afin de les rendre plus prioritaires que celles de NR. Cette redéfinition des priorités dépend de l'algorithme d'ordonnancement choisi : en DM, les priorités étant inversement proportionnelles aux délais critiques, les conditions à respecter sont alors :

$$(\forall \tau_i \in R)\ C_i \leq D_i < \min_{\tau_j \in NR}(D_j), \quad (3.3)$$

[1] $\wedge$ représente le symbole mathématique du pgcd et, *mod*, le symbole modulo.

en ED, les priorités sont dynamiques, il est donc difficile de prévoir à priori quelle sera la priorité d'une tâche τ_i de R face à une tâche τ_j de NR à un instant t. Il existe cependant une condition, certes forte mais permettant d'imposer à priori cette redéfinition des priorités. Il suffit pour cela d'imposer :

$$(\forall \tau_i \in R)\ D_i = C_i, \tag{3.4}$$

ainsi, lorsqu'une tâche τ_i de R demandera le processeur elle devra nécessairement l'obtenir sans quoi il y aura faute temporelle et la configuration ne sera pas ordonnançable.

4 Annulation de la gigue de régularité

Dans cette section, en utilisant les outils décrits ci-avant, nous allons d'abord rappeler les résultats obtenus pour une configuration de tâches indépendantes. Puis, nous verrons le cas de configuration de tâches dépendantes, que nous illustrerons enfin au travers un exemple.

4.1 Cas des tâches indépendantes

Comme DiNatale et Stankovic l'ont soulignés dans [DS95], le contrôle de la gigue est très délicat dans le cadre de tâches pouvant être préemptées pendant leur exécution. Aussi, avons-nous construit notre technique en imposant aux tâches régulières de s'exécuter sans préemption, pour, au final, obtenir un ordonnancement des tâches régulières du type de celui de la figure 7. Illustrons alors cette

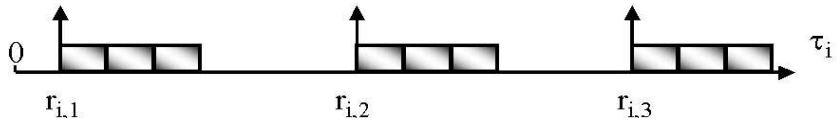

FIG. 7 – Exécution régulière et sans préemption d'une tâche de R.

technique au travers d'un exemple [DCG00a, DC00]. Soit une configuration de tâches dont deux sont à contraintes de régularité stricte (τ_{Acq_1}, τ_{Acq_2}) (cf. tableau 2). L'ordonnancement de ces tâches selon DM ou ED conduit à des gigues sur les tâches régulières qui ne sont pas pas négligeables (cf. tableau 2). On modifie alors étape par étape les attributs temporels de ces tâches.

TAB. 2 – Exemple de configuration de tâches.

Tâches	$r_{i,1}$	C_i	D_i	T_i	Gigues (%) DM / ED
τ_{Acq_1}(Acquisition)	0	2	16	16	11,5 / 7,7
τ_{Trait_1}(Traitement)	0	1	16	16	
τ_{Acq_2}(Acquisition)	0	2	32	32	12,5 / 8,9
$\tau_{Trait_{2-1}}$(Traitement)	0	4	31	32	
$\tau_{Trait_{2-2}}$(Traitement)	0	1	32	32	
τ_{Trait_3}(Traitement)	0	4	14	14	
τ_{Cont_3}(Contrôle)	0	1	13	14	
τ_{Cont_4}(Contrôle)	0	1	8	8	

Etape 1 : On considère le problème de désynchronisation de la section 3.2.1 pour les deux tâches régulières avec $W_i = C_i$. A l'aide d'un algorithme d'énumération des solutions, on obtient alors un ensemble de 377 couples solutions à ce problème. Choisissons par exemple, le couple solution $(r_{Acq_1,1}, r_{Acq_2,1}) = (0, 2)$.

Etape 3 : Pour garantir une exécution des tâches régulières (R) sans préemption, il suffit de les rendre plus prioritaires que les tâches sans contraintes de régularité (NR). On applique alors

les relations décrites dans la section 3.2.3 : en DM, l'équation 3.3 donne, dans les conditions les moins restrictives, des délais critiques tels que : $(D_1, D_2) = (7, 7)$, en ED, on obtient de l'équation 3.4 les délais critiques suivants : $(D_1, D_2) = (2, 2)$.

Etape 4 : On vérifie l'ordonnancement de la nouvelle configuration à l'aide d'un outil de simulation d'ordonnancement mono-processeur,

et on obtient finalement, une configuration ordonnançable en DM et en ED avec le résultat prévu sur les tâches régulières : leur gigue sont annulées.

4.2 Cas des tâches dépendantes

Dans notre technique de traitement de la gigue de régularité, nous allons ici prendre en considération des tâches dépendantes les unes des autres. En identifiant par τ_R, une tâche de R et par τ_{NR}, une tâche de NR, nous avons reporté dans la figure 8 les différents types possibles de précédence. Nous les avons ainsi numérotés de $P1$ à $P4$: $P1 : \tau_R \rightarrow \tau_{NR}$, $P2 : \tau_R \rightarrow \tau_R$, $P3 : \tau_{NR} \rightarrow \tau_{NR}$ et $P4 : \tau_{NR} \rightarrow \tau_R$.

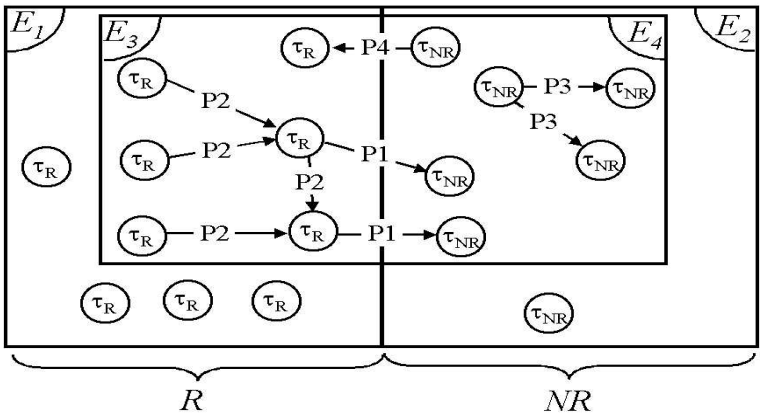

FIG. 8 – Application temps réel constituée d'un ensemble de tâches classées par catégorie : $E_1 \cup E_3 = R$, $E_2 \cup E_4 = NR$, $E_3 \cup E_4 =$ tâches mises en jeu dans une relation de précédence, $E_3 =$ tâches de R mises en jeu dans une relation de précédence, $E_4 =$ tâches de NR mises en jeu dans une relation de précédence.

Décrivons maintenant les 4 étapes à effectuer sur les tâches pour que celles à contraintes de régularité (τ_R) aient finalement une gigue de régularité nulle.

Etape 1 : Souhaitant toujours un ordonnancement des tâches régulières du type de celui de la figure 7, garant d'une gigue de régularité nulle, on applique alors la technique de désynchronisation des dates d'activation avec $W_i = C_i$ (cf. section 3.2.1). On obtient ainsi, si elles existent, de nouvelles dates d'activation pour les tâches régulières ($E_1 \cup E_3$ dans la figure 8). Notons ainsi l'ensemble de ces solutions par $\left\{ \left(r_{1,1}, ..., r_{n_R,1}\right)_j \, j \in [|1; p|] \right\}$ où p est le nombre de solutions à ce problème de désynchronisation.

Etape 2 : Il faut maintenant prendre en compte les éventuelles relations de précédence présentes dans la configuration de tâches. Pour chaque relation, les conditions de la section 3.2.2 doivent être appliquées :

– en DM et en ED, la première condition concerne les dates de première activation : elles doivent vérifier :

$$(\forall (i,j) \in \mathbb{N}^2)(\tau_i \rightarrow \tau_j) \begin{cases} r_{i,1} \leq r_{j,1} & (DM) \\ r_{i,1} < r_{j,1} & (ED) \end{cases} \tag{4.1}$$

Parmi les p n_R-uplets de l'étape 1, on choisit alors ceux qui vérifient également ces conditions (à noter que parmi ces choix, il n'y aura pas de dates d'activation identiques). Il reste ensuite

les conditions qui concernent les dates d'activation non incluses dans les n_R-uplets, à savoir celles des tâches de NR qui participent à une relation de précédence (région E_4 de la figure 8) : ces conditions permettent de définir à leur tour un ensemble de dates d'activation possibles pour les tâches de E_4.

Passons maintenant aux conditions sur les délais critiques et distinguons l'algorithme d'ordonnancement choisi :

- en DM, nous devons choisir des délais critiques de sorte que :

$$(\forall (i;j) \in \mathbb{N}^2)(\tau_i \rightarrow \tau_j)\ D_i < D_j. \tag{4.2}$$

  Ces conditions sont cependant inutiles dans le cas de deux tâches régulières liées par une précédence. En effet, le choix des dates d'activation effectué ci-dessus, combiné à notre future propriété selon laquelle les tâches régulières ne se chevauchent pas entre elles pendant leurs exécution, suffit à assurer la précédence.
- En ED, les seules conditions à respecter sont les suivantes :

$$(\forall (i;j) \in \mathbb{N}^2)(\tau_i \rightarrow \tau_j)\ D_i = C_i, \tag{4.3}$$

  ce qui définit de manière unique les nouveaux délais critiques des tâches mises en jeu dans les relations de précédence. Une remarque va cependant s'imposer pour ED :

Remarque 1 *Restreindre les délais critiques des tâches τ_i à C_i va indubitablement diminuer les chances d'ordonnançabilité de la nouvelle configuration (nous le verrons en pratique dans l'exemple qui suit). Une suggestion pour diminuer ce risque pourrait être de désynchroniser les tâches mises en jeu dans une relation de précédence, de sorte qu'il existe un intervalle entre les dates d'activation de deux tâches se succédant, qui permette au moins d'insérer l'exécution de la tâche qui précède. Cette suggestion est néanmoins écartée dans cette étude pour la clarté de la description.*

Etape 3 Pour garantir une exécution sans préemption des tâches de R, il suffit en fait de les rendre plus prioritaires que les tâches de NR. En effet, pendant l'exécution d'une tâche τ_i de R, on ne pourra pas la faire préempter par une tâche de NR à cause des priorités, et on ne pourra pas non plus la faire préempter par d'autres tâches de R, car elles sont déjà désynchronisées entre elles.

- dans le cas DM, d'après la section 3.2.3, la redéfinition des priorités se traduit par un nouveau délai critique D_i^* pour les tâches de R selon la condition :

$$(\forall \tau_i \in R)\ C_i \leq D_i^* \leq \min_{\tau_j \in NR}(D_i; D_j - 1). \tag{4.4}$$

  Ainsi, la tâche τ_i obtient un nouveau délai critique qui est strictement inférieur aux délais critiques des tâches NR. Cette redéfinition appelle cependant la remarque suivante :

Remarque 2 *Imposer $D_i^* = \min_{\tau_j \in NR}(D_i; D_j - 1)$ ou imposer $D_i^* = C_i$ revient exactement au même problème d'ordonnancement. En effet, les différentes exécutions des tâches R sont toujours désynchronisées les unes par rapport aux autres selon les résultats de la première étape, donc, les seules fautes temporelles qui pourraient apparaître sont celles mettant en jeu une tâche régulière et une tâche non nécessairement régulière : le fait alors d'imposer par l'équation 4.4 la borne maximale ou la borne minimale implique des priorités plus grandes pour les tâches R qui suffisent alors à lever cette éventualité. Il est néanmoins intéressant de définir l'ensemble des nouveaux délais critiques possibles, de sorte qu'un éventuel critère analytique d'ordonnançabilité puisse par exemple être vérifié dans le cas le moins contraint.*

Cette redéfinition (4.4) appelle également une remarque sur sa compatibilité avec l'équation 4.2 :

- en effet, en DM, l'équation 4.4 impose en fait : $D_{\tau_R} < D_{\tau_{NR}}$ qui peut être contradictoire avec les équations de 4.2. La seule façon d'éviter l'incompatibilité entre les deux redéfinitions, c'est d'interdire les relations de précédence du type $P4 : \tau_{NR} \rightarrow \tau_R$, qui typiquement nécessiterait une condition $D_{\tau_{NR}} < D_{\tau_R}$. L'élimination du traitement de ce type de précédence ($P4$) n'est pas réellement une restriction car il apparaît conceptuellement difficile de faire précéder une tâche que l'on veut régulière, par une tâche dont l'exécution est fluctuante en terme d'occurrence. Une fois cette précédence mise de côté, les mécanismes de choix des délais critiques des tâches de R se font comme dans le cas de la précédence : parmi les n_R-uplets solutions des équations 4.4, c'est à dire parmi un ensemble du type $\left\{(D_1, ..., D_{n_R})_j \, j \in [|1; q|]\right\}$ où q est le nombre de solutions aux équations 4.4, on choisit ceux qui répondent également aux conditions des contraintes de précédence (équations 4.2).
- En ED, on rajoute aux conditions de l'étape 2, les conditions :

$$(\forall \tau_i \in R)\ D_i^* = C_i.$$

Ces conditions sont entachées de la même remarque faite dans le paragraphe précédent (remarque 1) où l'on évoqué leur caractère restrictif et leurs conséquences sur l'ordonnançabilité de la configuration.

Etape 4 Etant donné l'ensemble des nouveaux attributs temporels possibles pour les tâches régulières et pour les tâches mises en jeu dans une relation de précédence : on choisit alors une solution parmi celles disponibles, puis, dans une dernière étape, nous vérifions l'ordonnançabilité de la nouvelle configuration en faisant appel, par exemple, à une simulation.

4.3 Exemple complet

Considérons une application temps réel simplifiée qui piloterait en partie un aéronef. Cette application est décrite dans la figure 9 et les différents paramètres temporels des tâches sont présentés dans le tableau 3. Elle présente les deux types de tâches R et NR et tous les types autorisés de précédence. Dans ce tableau 3, nous avons également reportés les gigues temporelles des tâches régulières, obtenues en ordonnançant directement la configuration. Nous allons alors dans cet exemple dérouler les différentes étapes qui vont conduire à annuler la gigue des tâches régulières (τ_{R_1}, τ_{R_2}, τ_{R_3}) et qui vont conduire à respecter les contraintes de précédence.

TAB. 3 – Description des paramètres de tâches d'une application temps réel.

	$r_{i,1}$	C_i	D_i	T_i	Gigues (%) DM - ED	Commentaires
τ_{NR_1}	0	2	32	32		(NR) Tâche de mesure de la température.
τ_{NR_2}	0	1	32	32		(NR) Tâche de calcul de la vitesse de l'aéronef.
τ_{R_1}	0	3	32	32	2,6 - 2,6	(R) Tâche de mesure de la pression.
τ_{NR_3}	0	1	16	16		(NR) Tâches pour piloter la température et la pression cabine.
τ_{R_2}	0	2	16	16	10 - 10	(R) Tâche de commande des organes de vol.
τ_{R_3}	0	1	16	16	10 - 10	(R) Tâche de d'acquisition des ordres de pilotage.
τ_{NR_4}	0	2	10	10		(NR) Tâche d'alarme (rapide).
τ_{NR_5}	0	3	56	56		(NR) Tâche d'affichage.

Etape 1 : Désynchronisation des dates de premières activation Dans un premier temps, nous décalons les dates de première activation les unes par rapport aux autres de sorte qu'aucune exécution d'une tâche régulière ne vienne perturber une exécution d'une autre tâche régulière.

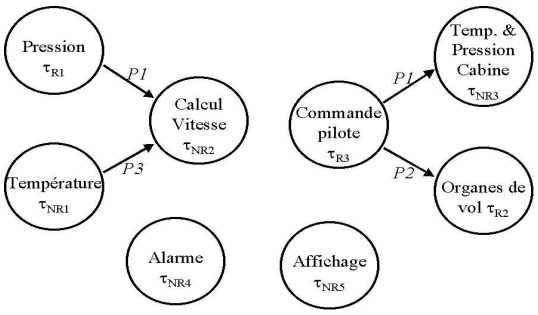

FIG. 9 – Application temps réel.

Nous cherchons donc les triplets $(r_{R_1,1}, r_{R_2,1}, r_{R_3,1})$ qui répondent au problème de désynchronisation avec $W_{R_1} = 3$, $W_{R_2} = 2$ et $W_{R_3} = 1$. Cela nous conduit alors à trouver 3696 solutions différentes ($(r_{R_1,1}, r_{R_2,1}, r_{R_3,1}) = (0, 3, 5)$ ou $(r_{R_1,1}, r_{R_2,1}, r_{R_3,1}) = (0, 4, 3)$ par exemple).

Etape 2 : Prise en compte des précédences pour les $r_{i,1}$: En DM, pour respecter les relations $P1$ et $P2$, on doit choisir parmi les $r_{R_i,1}$ $(1 \leq i \leq 3)$ de l'étape 1, ceux qui vérifient les relations suivantes : $r_{R_1,1} \leq r_{NR_2,1}$, $r_{R_3,1} \leq r_{R_2,1}$ et $r_{R_3,1} \leq r_{NR_3,1}$, où $r_{NR_2,1}$ et $r_{NR_3,1}$ sont à ajuster également. On choisit par exemple le triplet : $(r_{R_1,1}, r_{R_2,1}, r_{R_3,1}) = (0, 4, 3)$ pour les tâches régulières. Quant aux dates de premières activations des tâches NR, on peut faire d'une part les choix suivants : $r_{NR_2,1} = 1$, $r_{NR_3,1} = 4$, et d'autre part, en vue de respecter la relation $P3$ qui impose : $r_{NR_1,1} \leq r_{NR_2,1}$, on peut prendre $r_{NR_1,1} = 0$. En ED, ce sont les mêmes relations qui interviennent, à ceci près que nous avons à faire à des inégalités strictes.

Prise en compte des précédences pour les D_i : En DM, les relations à satisfaire pour les priorités sont les suivantes :

$$D_{R_1} < D_{NR_2} \quad , \quad D_{NR_1} < D_{NR_2} \quad et \quad D_{R_3} < D_{NR_3} \tag{4.5}$$

laissant de côté la relation $D_{R_3} < D_{R_2}$ qui n'est pas nécessaire pour faire respecter la contrainte de précédence pour les tâches régulières (cf. explications dans 4.2, étape 2). En ED, les conditions à satisfaire pour les tâches impliquées dans une relation de précédence sont :

$$Pour\ toute\ relation\ du\ type : \ \tau_i \rightarrow \tau_j, \ alors\ D_i = C_i \tag{4.6}$$

Ces conditions établies, nous allons dans le paragraphe suivant, faire les choix définitifs pour les valeurs des D_i.

Etape 3 : Changement de priorité des tâches régulières : En DM, la relation 3.3 impose que le délai critique d'une tâche R doit être strictement inférieur au délai critique de toutes les autres tâches NR. Ce qui consiste à imposer que :

$$D_{R_1} < D_{NR_j} \quad , \quad D_{R_2} < D_{NR_j} \quad et \quad D_{R_3} < D_{NR_j} \tag{4.7}$$

pour j indiçant toutes les tâches NR. Nous pouvons alors proposer les solutions suivantes pour les nouvelles valeurs des D_i, solutions qui vérifient les deux ensembles de conditions 4.5 et 4.7 : $D_{R_1} = 9$, $D_{R_2} = 9$, $D_{R_3} = 9$, $D_{NR_1} = 31$, $D_{NR_2} = 32$ (inchangé) et $D_{NR_3} = 16$ (inchangé également). En ED, les conditions à imposer sur les délais critiques reviennent en fait à définir pour toutes les tâches régulières : $D_i = C_i$.

Etape 4 : Test d'ordonnançabilité Il s'agit maintenant de vérifier si la nouvelle configuration de tâches est bien ordonnançable en DM ou en ED. Dans le cadre de DM, nous obtenons finalement le résultat escompté : les gigues des tâches τ_{Acq_1} et τ_{Acq_2} sont annulées et les contraintes de précédence sont respectées (cf. figure 10). En ED par contre, nous obtenons une configuration non ordonnançable qui renforce l'idée exprimée précédemment dans la remarque 1. On peut cependant essayer de relâcher les contraintes sur les délais critiques pour les mettre au même niveau que ceux de DM, on obtient alors une configuration ordonnançable dans laquelle les précédences sont respectées, contrairement aux gigues des tâches régulières qui ne sont pas annulées. En fait, dans cette éventualité, comme les manipulations des attributs temporels ne sont plus celles exigées par notre technique dans le cadre de ED, les précédences peuvent alors ne plus être respectées et les gigues ne plus être annulées. Cet exemple souligne ainsi le caractère restrictif de la technique dans le cadre de ED, comparée à une utilisation dans le cadre de DM où les modifications des paramètres sont moins contraignantes.

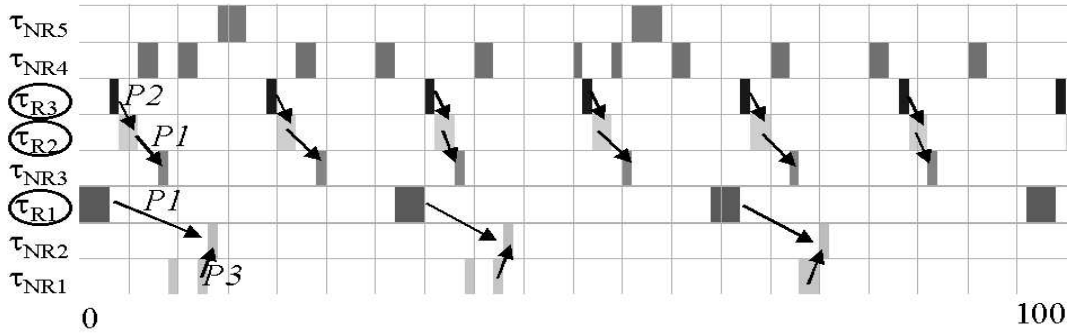

FIG. 10 – La séquence d'ordonnancement après la modification des paramètres dans le contexte de DM : les exécutions des tâches R sont régulières et les précédences de l'application sont respectées.

4.4 Conclusion : besoin d'une méthode plus flexible

Le problème de recherche des $\{r_{i,1}\}$ pour la phase de désynchronisation est un problème complexe. De plus, la nouvelle configuration de tâches, surtout pour le cas ED, présente des attributs temporels très contraints pour répondre à notre technique. Ces contraintes, nous l'avons vu au travers de l'exemple précédent, peuvent nous conduire à des configurations non ordonnançables surtout lorsque le nombre de tâches régulières augmente. D'un point de vue pratique, au lieu d'imposer une gigue nulle aux tâches de R, il serait plus judicieux de relâcher les contraintes et d'obtenir seulement un encadrement des gigues. D'ailleurs, dans les spécifications d'une application temps réel, on rencontre communément de telles contraintes [Bat98].

5 Encadrement de la gigue de régularité

La technique décrite dans cette section est sensiblement similaire à celle de la section précédente : nous allons en fait être moins contraignants sur les tâches, de sorte que plus de configurations soient ordonnançables et de sorte que les gigues soient non plus annulées mais bornées. La principale raison pour laquelle la technique précédente conduit à des configurations trop contraintes, peut être justifiée par notre idée d'imposer une exécution sans préemption des tâches régulières. Dans cette section, nous allons donc autoriser la préemption des tâches de R, à l'intérieur d'intervalles d'exécution basés sur les spécifications des gigues, plus précisément, sur les gigues de cohésion des tâches régulières. En effet, la définition de la gigue de cohésion se fait sur les durées séparant le début et la fin d'exécution d'une même instance (cf. section 2), définissant alors une largeur moyenne, maximum ou minimum (etc...), des intervalles d'exécution des instances, intervalles sur lesquels la préemption est autorisée. Ainsi considérant la gigue maximum de cohésion notée $J_{max,c}(\tau_i)$ d'une tâche τ_i, elle définit alors la largeur

maximum autorisée pour l'exécution d'une instance de τ_i (cf. figure 11). La quantité $J_{max,c}(\tau_i) - C_i$ définit par ailleurs la durée à l'intérieur de ces intervalles pendant laquelle le processeur n'est pas allouée à τ_i (cf. figure 11).

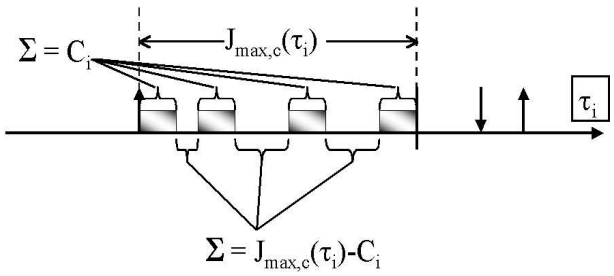

FIG. 11 – La gigue maximum de cohésion d'une tâche ($J_{max,c}(\tau_i)$) et la durée de préemption possible ($J_{max,c}(\tau_i) - C_i$).

De la même façon que dans la section précédente, nous allons alors imposer à chaque tâche τ_i de l'ensemble R de s'exécuter dans des intervalles commençant à ses dates d'activation ($r_{i,j}$ $j \geq 1$) et se terminant $W_i = J_{max,c}(\tau_i)$ unités de temps processeur plus tard. En outre, l'intersection de ces intervalles devra être nulle, ce qui implique que les éventuelles préemptions des tâches régulières (R) soient exclusivement dues à des tâches non nécessairement régulières (NR). D'un point de vue conception, ceci est justifiable puisque les tâches régulières sont le plus souvent, mises en jeu dans des activités pour lesquelles l'exécution des tâches doit être effectuée dans un ordre prédéfini afin de pouvoir garantir la cohérence temporelle des données.

5.1 Résultats sur la gigue temporelle moyenne

A l'issue de notre technique d'encadrement, les intervalles dans lesquels les tâches τ_i de R sont autorisées à s'exécuter débutent aux différentes dates d'activation de τ_i et se terminent $J_{max,c}(\tau_i)$ unités de temps processeur plus tard. Considérons donc une telle tâche τ_i. Sur ces intervalles, le pire cas pour la gigue est définie par l'obtention de la plus petite durée puis de la plus longue durée séparant les fins d'exécution de deux instances successives (cf. figure 12).

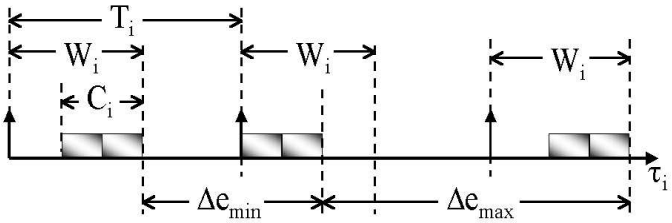

FIG. 12 – Le cas du maximum de gigue dans l'exécution de la tâche τ_i.

Selon les conventions adoptées dans la section 2, nommons par Δe_{min} et par Δe_{max} ces deux durées (cf. figure 12). Nous avons alors les relations suivantes :

$$\Delta e_{max} = T_i + W_i - C_i, \quad \Delta e_{min} = T_i + C_i - W_i,$$

qui vont nous permettent d'encadrer la gigue temporelle moyenne. Pour cela, supposons que sur une période d'étude H, il y ait autant de durées Δe_{min} et que de durées Δe_{max}. Soit N ce nombre, nous avons alors selon l'équation 2.2, une gigue temporelle moyenne telle que :

$$J_{Moy}(\tau_i) = \frac{1}{2N} \frac{N\,|\Delta e_{min} - T_i| + N\,|\Delta e_{max} - T_i|}{T_i} 100\ \%,$$

qui conduit à :

$$J_{Moy}(\tau_i) = \frac{J_{max,c}(\tau_i) - C_i}{T_i} 100\ \%.$$

En résumé donc, nous imposons dans un premier temps aux tâches régulières de s'exécuter dans des intervalles de largeur $W_i = J_{max,c}(\tau_i)$, démarrant aux différentes dates d'activation des tâches τ_i pour ensuite obtenir deux résultats sur leurs gigues :

- la gigue de cohésion spécifiée ne sera effectivement jamais dépassée pendant l'exécution,
- et la gigue temporelle moyenne sera bornée par la valeur $J_{Moy}(\tau_i)$ décrite ci-dessus.

5.2 Description détaillée de la technique

Rappelons que les périodes des tâches participant à une relation de précédence sont identiques et que nous conservons les descriptions des relations de précédence du paragraphe 4.2.

Etape 1 Il s'agit dans un premier temps de résoudre le problème de désynchronisation avec $W_i = J_{max,c}(\tau_i)$ pour les tâches de R (région $E_3 \cup E_4$ de la figure 8). Cela donne alors un ensemble de n_R-uplets du type : $\left\{(r_{1,1}, ..., r_{n_R,1})_j \; j \in [|1; p|]\right\}$, où p est le nombre de solutions au problème.

Etape 2 On procède de la même façon que dans la section précédente : parmi ces p n_R-uplets, on choisit ainsi ceux qui vérifient les conditions 4.1 (à noter que parmi ces choix, il n'y aura pas de dates d'activation identiques). Sur ces mêmes conditions, on rassemble l'ensemble des solutions possibles pour les dates d'activation des tâches de la région E_4 (cf. figure 8). On passe ensuite aux délais critiques :

- en DM, nous devons vérifier les équations 4.2, qui sont cependant inutiles pour le cas de deux tâches régulières liées par une précédence. La définition des délais critiques des tâches de la région E_3, dans l'étape 3, suffira en effet à assurer la précédence pour les tâches de E_3.
- en ED, il faut vérifier les équations 4.3 qui impliquent alors une exécution de la tâche τ_i sans préemption et ce, dès son activation. Ainsi, dans le contexte ED, pour toutes les tâches précédant une autre tâche, la largeur d'exécution est réduite à $W_i = C_i$. Dans ce contexte, il est alors nécessaire de refaire l'étape 1 en prenant en compte cette restriction pour le problème des désynchronisations. Le début de l'étape 2, celui qui concerne les dates d'activations, doit alors aussi être reconduit.

Etape 3 Dans cette étape, nous redéfinissons les délais critiques des tâches τ_i de R de sorte qu'elles s'exécutent dans les intervalles de largeur $W_i = J_{max,c}(\tau_i)$. On pose ainsi : $D_i^* = W_i = J_{max,c}(\tau_i)$ qui garantit que τ_i ne sera préemptée (par une tâche de NR) que pendant une durée maximum de $J_{max,c}(\tau_i) - C_i$, sans quoi il y aura faute temporelle.

- Cette redéfinition garantit également en DM le respect des contraintes de précédence des tâches de E_3. En effet, à l'issue des choix dans l'étape 2 des dates d'activation (respect du problème de désynchronisation et respect des contraintes de précédence), cette redéfinition conduit à des exécutions des tâches de R dans le bon ordre et sans chevauchement.
- Dans le contexte ED, pour chaque tâche τ_i de R qui précède n'importe quelle type de tâche, à la place de cette définition, on garde la condition 4.3 qui impose $D_i^* = C_i$.

Etape 4 Nous vérifions l'ordonnançabilité de la nouvelle configuration.

5.3 Exemple

Reprenons la structure de l'exemple de la section précédente (paragraphe 4.3 : tableau 3, figure 9) et supposons que les gigues maximum de cohésion spécifiées pour les tâches régulières, sont les suivantes : $J_{max,c}(\tau_{R_1}) = 11$, $J_{max,c}(\tau_{R_2}) = 4$, $J_{max,c}(\tau_{R_3}) = 1 = C_{R_3}$ (préemption interdite pour τ_{R_3}). Pour appliquer notre technique à cet ensemble de tâches, nous allons distinguer l'algorithme d'ordonnancement choisi puisque, nous l'avons vu, la manipulation des paramètres temporels est

sensiblement différentes suivant que l'on choisit DM ou ED. Illustrons en détail la technique dans le cas de ED, sachant qu'elle ne pose pas de problème particulier pour le contexte DM.

Etape 1 : Dans les relations de précédence de cette application, il y a deux tâches régulières qui précèdent d'autres tâches, appartenant à R ou à NR. Aussi, selon les considérations évoquées dans l'étape 3 du paragraphe 5.2, l'exécution de ces tâches régulières va se faire sans préemption. On applique alors le principe de désynchronisation (cf. section 3.2.1) aux tâches régulières : τ_{R_1}, τ_{R_2} et τ_{R_3} avec les paramètres suivants : $W_{R_1} = C_{R_1} = 3$, $W_{R_2} = J_{max,c}(\tau_{R_2}) = 4$ et $W_{R_3} = C_{R_3} = 1$. La recherche de tous les triplets possibles à l'aide d'une technique d'énumérations, conduit à caractériser 2160 triplets solutions différents ($(r_{R_1,1}, r_{R_2,1}, r_{R_3,1}) = (0, 3, 7)$ ou $(r_{R_1,1}, r_{R_2,1}, r_{R_3,1}) = (2, 8, 7)$ par exemple).

Etape 2 : Pour respecter les relations de l'application temps réel (cf. figure 9), on doit choisir parmi les différentes dates de première activation possibles établies dans le paragraphe précédent, des $r_{R_i,1}$ qui vérifient les relations suivantes :

$$\begin{array}{lll} r_{R_1,1} < r_{NR_2,1} & et & r_{R_3,1} < r_{R_2,1} \\ r_{R_3,1} < r_{NR_3,1} & et & r_{NR_1,1} < r_{NR_2,1} \end{array} \tag{5.1}$$

(cf. section 3.2.2 pour détail). On choisit alors par exemple le triplet : $(r_{R_1,1}, r_{R_2,1}, r_{R_3,1}) = (2, 8, 7)$ pour les tâches régulières, et $r_{NR_2,1} = 3$, $r_{NR_3,1} = 8$, $r_{NR_1,1} = 0$ pour les tâches non nécessairement régulières présentes dans ces conditions.

Quant aux conditions sur les D_i, on doit respecter :

$$D_{R_1} = C_{R_1} \quad , \quad D_{NR_1} = C_{NR_1} \quad et \quad D_{R_3} = C_{R_3} \tag{5.2}$$

de sorte qu'en cas de demande commune du processeur, les tâches précédant soient prioritaires.

Etape 3 : En ED, il reste le nouveau délai critique de la tâche τ_{R_2} à définir : $D_{R_2} = J_{max,c}(\tau_{R_2}) = 4$. Nous pouvons alors proposer les solutions suivantes pour les D_i : $D_{NR_1} = 2$ et $(D_{R_1}, D_{R_2}, D_{R_3}) = (3, 4, 1)$.

Etape 4 : On obtient au final une configuration ordonnançable en ED, avec des précédences respectées et des gigues bornées, qui sont en fait annulées. En utilisant cette configuration pour le contexte DM, nous obtenons le même résultat avec en outre des gigues moyennes bornées : $J_{Moy}(\tau_{R_1}) = 2,54$ %, $J_{Moy}(\tau_{R_2}) = 0$ %, et $J_{Moy}(\tau_{R_3}) = 0$ %.

6 Conclusion

Nous avons vu au travers de cette étude qu'il était possible d'ordonnancer une configuration de tâches en respectant les contraintes de régularité d'exécution et les contraintes de précédence, dans les contextes d'ordonnancement DM ou ED. La technique se décrit ainsi en quatre étapes : désynchronisation des dates d'activation, prise en compte des précédences, restriction des fenêtres d'exécution à une largeur limite et enfin, vérification de l'ordonnançabilité. Ces modifications successives des paramètres des tâches, permettant alors en DM et en ED d'annuler ou d'encadrer la gigue de certaines tâches, ont été automatisées. A noter que dans le cadre de ED, les chances d'ordonnançabilité sont plus faibles que dans le cadre de DM.

Dans une prochaine étape, il s'agira de généraliser ces mécanismes au traitement de la gigue de bout en bout et de voir le cas de configuration de tâches partageant des ressources critiques, et intégrant des tâches apériodiques.

Références

[Bat98] I. J. Bate, *Scheduling and timing analysis for safety critical real-time systems*, Ph.D. thesis, Department of Computer Science, University of York, nov. 1998, pp. 159–184.

[BB99] I. Bate and A. Burns, *An approach to task attribute assignment for uniprocessor systems*, Proc. of the 11^{th} EuroMicro on Real-time Systems (York, England), June 1999, pp. 46–53.

[Bla76] J. Blazewicz, *Modelling and performance evolution of computers systems*, ch. Scheduling dependent tasks with different arrival times to meet deadlines, North-Holland Publishing Company, 1976.

[But97] G.C. Buttazzo, *Hard real-time computing systems*, ch. Periodic task scheduling, pp. 77–108, Kluwer Academic Publishers, 1997.

[CC96] S.-T. Cheng and C.-M. Chen, *A cyclic scheduling approach for relative timing requirements*, Proc. of the 3^{rd} IEEE Real-Time Applications Worshop, 1996, pp. 160–163.

[CGC96] A. Choquet-Geniet, D. Geniet, and F. Cottet, *Exhaustive computation of the scheduled task execution sequences of a real-time application*, Proc. of the 4^{th} International Symposium on Formal Techniques in Real-Time and Fault-Tolerant Systems (Uppsala, Sweden), 1996, pp. 246–262.

[DC00] L. David and F. Cottet, *Le traitement de la gigue dans un contexte en ligne*, Tech. Report 00 009, LISI-ENSMA, Futuroscope, France, 2000.

[DCG00a] L. David, F. Cottet, and E. Grolleau, *Gigue temporelle et ordonnancement par échéance dans les applications temps réel*, Proc. of the IEEE Conf. Inter. Francophone d'Automatique (Lille, France), 2000, pp. 681–686.

[DCG00b] L. David, F. Cottet, and E. Grolleau, *Maîtrise de la gigue temporelle avec les algorithmes d'ordonnancement DM et ED*, Tech. Report 00 004, LISI-ENSMA, Futuroscope, France, 2000.

[DS95] M. DiNatale and A. Stankovic, *Applicability of simulated annealing methods to real-time scheduling and jitter control*, Proc. of Real-Time Systems Symposium (Pisa), 1995, pp. 190–199.

[HLH96] C.-C. Han, K.-J. Lin, and C.-J. Hou, *Distance-constrained scheduling and its applications to real-time systems*, IEEE Transactions on Computers **45(7)** (1996), 814–826.

[LH96] K.-J. Lin and A. Herkert, *Jitter control in time-triggered systems*, Proc. of the 29^{th} International Conf. on System Sciences (Hawaiï, U.S.A.), 1996, pp. 451–459.

[LL73] C.L. Liu and J.W. Layland, *Scheduling algorithms for mutltiprogramming in real-time environnement*, Journal of the ACM **20(1)** (1973), 46–61.