Ordonnancement et ordonnançabilité monoprocesseur

Emmanuel Grolleau
LISI, ENSMA
Téléport 2, 1 Av. Clément Ader
BP 40109, 86961 Chasseneuil-Futuroscope Cedex, France
grolleau@ensma.fr

***Résumé*—Cet article présente les avancées récentes en terme de modèles de tâches, et leur utilisation dans le cadre monoprocesseur. Nous nous concentrons sur les modèles de tâches prenant en compte les décalages de dates d'activation, et notamment sur les transactions, et les modèles *multiframes* et ses généralisations, jusqu'au modèle *digraph*. Sur ces modèles complexes, il y a principalement deux types d'analyse. D'une part, l'analyse de pire temps de réponse (RTA - *Response Time Analysis*), est utilisée pour étudier des systèmes ordonnancés par des algorithmes à priorités fixes. D'autre part, l'analyse de la demande processeur (DBF - *Demand Bound Function*), est utilisée pour étudier l'ordonnançabilité globale d'un système. Nous nous intéressons ici à l'analyse de temps de réponse.**

I. Introduction

Un système temps réel s'assure de maintenir un procédé (drone, système automobile ou avionique, etc.) dans un état voulu, en assurant une réactivité du système cohérente avec les contraintes de temps inhérentes au procédé. Le système s'informe de l'état courant du procédé à l'aide de capteurs, et agit sur celui-ci à l'aide d'actionneurs. Les contraintes de temps peuvent provenir de contraintes physiques liées à des capteurs ou des actionneurs (par exemple lecture d'informations sur un bus de communication devant s'effectuer à temps, commandes envoyées devant être réactualisées à un certain rythme, etc.) ou bien être des contraintes de bout-en-bout et s'appliquant sur la chaine de traitement partant d'un ensemble de capteurs et se terminant à un ensemble d'actionneurs. Par exemple, il est commun de voir des contraintes de temps d'un ordre de grandeur de quelques dizaines de millisecondes pour la chaine de traitement suivante : (1) lecture sur une centrale inertielle de l'attitude (tangage, roulis) d'un aéronef, (2) calcul des commandes à appliquer à l'aide d'algorithmes d'asservissement, (3) applications des commandes sur les gouvernes et le moteur.

L'implémentation d'un système temps réel peut reposer sur l'hypothèse synchrone ou l'hypothèse asynchrone. Dans l'hypothèse synchrone, on considère que le système est capable de réagir en un temps que l'on peut considérer comme instantané par rapport à la dynamique du procédé. Lorsque l'on se place dans l'hypothèse asynchrone, on considère que les durées de traitement sont telles que l'on doit prendre en considération le fait qu'un ou plusieurs évènements puissent avoir lieu pendant un traitement. Dans ce cas, il convient de pouvoir préempter un traitement en cours, afin d'opérer un autre traitement. On utilise à cette fin plusieurs traitement parallèles : les tâches. Il existe deux mécanismes permettant de contrôler un fonctionnement multitâche : l'ordonnancement et le séquencement. Un séquenceur se base sur une séquence calculée hors-ligne (avant mise en œuvre du système) qui détermine quelle tâche s'exécute et à quel instant. Les méthodes utilisées pour calculer une séquence d'ordonnancement valide hors-ligne peuvent être de type séparation et évaluation [1], [2], énumératives basées sur des modèles de représentation des tâches [3], ou utilisant du *model checking* [4], ou encore des métaheuristiques. Un ordonnanceur se base sur un algorithme d'ordonnancement, qui utilise des priorités pour réaliser ses décisions d'ordonnancement en-ligne. Dans cet article, nous nous intéressons aux systèmes multitâche ordonnancés en-ligne par un ordonnanceur à priorités.

Nous présentons en section II le modèle dit de Liu et Layland avec ses extensions classiques, et les problèmes classiques de validation temporelle. En section III nous voyons les techniques de calcul de temps de réponse sous différentes hypothèses et facteurs pratiques. Enfin, la section IV présente des modèles prenant en compte des décalages de réveil dans un contexte dirigé par les évènements, et montre comment les techniques de calcul de temps de réponse ont été adaptées à ces modèles.

II. Validation temporelle

La validation temporelle est une étape primordiale dans la phase de développement d'un système temps réel critique. Le système temps réel est d'abord modélisé de sorte que le modèle ne puisse en aucun cas avoir un meilleur comportement que l'exécution réelle. Il y a donc bien un écart entre modèle et réalité, le modèle étant un pire cas de la réalité. Ensuite il faut s'assurer que pour tout comportement du modèle, les contraintes de temps sont respectées. Il s'agit donc de démontrer que le pire cas de comportement du modèle (qui est un pire cas du système modélisé) respecte les contraintes de temps.

Définition 1 (Fiablement ordonnançable): Un système S est fiablement ordonnancé par un algorithme d'ordonnancement A, si pour toute instance de système correspondant à la spécification de S, l'ordonnancement de S respecte toutes les contraintes temporelles.

Définition 2 (Ordonnançable): Un système S est ordonnançable si il existe un algorithme qui l'ordonnance fiablement.

Ainsi, si on considère de façon classique le modèle dit de Liu&Layland (L&L) [5]. Un système de tâches temps réel est $S = \{\tau_i\}_{i=1..n}$ où chaque tâche τ_i est définie par un sous-ensemble des paramètres temporels $\langle O_i, C_i, D_i, T_i, J_i\rangle$:

- T_i est la période d'activation de τ_i. Dans le cas strictement périodique, cette période est stricte : notons que cela ne peut correspondre qu'à une implémentation où τ_i est déclenchée par l'horloge interne. Dans le cas sporadiquement périodique, T_i représente l'écart minimal entre deux activations successives de la tâche. Chaque activation de la tâche crée une instance qui doit être exécutée en respectant ses contraintes de temps.
- O_i est la première date d'activation de τ_i. Cette date peut être connue, on dit alors que le système est concret, et sous cette hypothèse, on se trouve généralement dans le cas strictement périodique pour T_i. Si toutes les tâches sont activées simultanément, on parle de systèmes de tâches simultanées, alors que dans le cas contraire, on parle de systèmes de tâches différées. Lorsque O_i est inconnu, le système est dit non concret.
- C_i est la pire durée d'exécution d'une instance de la tâche par le processeur. On appelle cette durée le WCET (*Worst-Case Execution Time*). Le WCET est une estimation pire cas de la durée d'exécution d'une instance de τ_i, le WCET est souvent très nettement supérieur à la durée moyenne d'exécution [6].
- D_i est le délai critique de τ_i et représente son échéance relative : une instance de τ_i doit être terminée au pire D_i unités de temps après sa date de réveil. La durée séparant l'activation d'une instance de sa terminaison s'appelle son temps de réponse. Vérifier le respect des contraintes de temps revient donc, généralement, à s'assurer que le temps de réponse de toute instance de τ_i n'est pas plus grand que son délai critique.
- J_i est la gigue temporelle de τ_i. Ce paramètre peut apparaitre pour les systèmes de tâches concrets et strictement périodiques et représente l'incertitude quant à la date réelle à laquelle les instances sont activées. La $k^{ème}$ instance de τ_i est alors activée dans l'intervalle $[O_i + (k-1)T_i..O_i + (k-1)T_i + J_i]$.

On considère généralement qu'une tâche n'est pas réentrante (i.e. une instance d'une tâche τ_i ne peut pas commencer son exécution avant que toutes ses instances précédentes ne soient terminées), ce qui est cohérent avec le fonctionnement de tous les exécutifs temps réel.

Même en ne considérant que des tâches indépendantes, pour lesquelles la préemption est autorisée à tout moment, il existe, à partir de ces paramètres temporels, plusieurs modèles de comportements très différents. Par exemple, si les tâches sont concrètes, strictement périodiques, et à échéance sur requête (i.e. $D_i = T_i$), alors les tâches du système peuvent être définies par $\langle O_i, C_i, T_i\rangle$. Dans ce cas, la $k^{ème}$ instance de τ_i est activée à l'instant $(k-1)T_i$ et doit être terminée avant la date $(k-1)T_i + D_i$. Si les tâches sont non-concrètes, à échéance sur requête, et sporadiquement périodiques, les tâches du système peuvent être définies par $\langle C_i, T_i\rangle$. On ne connait pas la date d'activation de la $k^{ème}$ instance, cependant, on sait qu'elle ne peut pas avoir lieu moins de T_i unités de temps après l'instance $k-1$. Soit $O_{i,k}$ cette date, elle devra alors se terminer avant la date $O_{i,k} + D_i$.

Dans certains cas simples, il existe des formules algébriques, calculables en temps polynomial, permettant de vérifier l'ordonnançabilité d'un système [7]. Cependant, ces formules algébriques sont le plus souvent des conditions suffisantes mais non nécessaires, de plus, elles ne sont pas adaptées aux systèmes intégrant des paramètres réalistes (partage de ressources critiques, précédences entre les tâches, tâches concrètes différées, etc.).

Il existe donc deux techniques de base de complexité pseudo-polynomiale : sous certaines hypothèses simples, ces techniques sont exactes (i.e. la réponse donnée est nécessaire et suffisante), alors que dans des cas plus généraux, la réponse obtenue ne peut être qu'une condition suffisante. La première technique est la fonction de la demande (DF - *Demand Function*) [8]. Cette technique ne considère pas d'algorithme d'ordonnancement en particulier, mais traduit la possibilité pour un ordonnancement optimal de respecter toutes les échéances du système. La seconde technique de base est l'analyse de temps de réponse (RTA - *Response-Time Analysis*) [9][10], et permet d'étudier le temps de réponse des tâches étant donné un ordonnancement à priorités fixes. Notons que ces techniques sont de la même complexité que les problèmes de base qu'elles résolvent : en effet, le problème d'ordonnançabilité d'un système de tâches non-concrète et sporadiquement périodiques (ou de façon équivalent strictement périodiques et simultanées) est NP-difficile au sens faible [11]. Quant au problème de calcul de temps de réponse, il est de même complexité [12].

III. Calcul de temps de réponse

La RTA permet de calculer le pire temps de réponse d'une tâche pour un ordonnancement à priorités fixes. Les ordonnanceurs à priorités fixes sont les plus répandus dans l'industrie. En effet, la très grande majorité des exécutifs commerciaux (normes POSIX [13], OSEK [14], etc.) ne permettent que d'accorder aux tâches une priorité fixe. Seuls quelques exécutifs académiques, et les exécutifs conformes à l'annexe D.2.6 de la norme Ada 2005 [15] et ultérieur, permettent d'utiliser un ordonnanceur à priorités dynamiques. Dans cette section, nous nous plaçons dans un contexte à priorités fixes.

A. Modèle classique de tâches

Le calcul de temps de réponse se base sur l'instant critique ainsi que sur les deux théorèmes suivants :

Définition 3 (Instant critique [5], [16]): L'instant critique pour une tâche a lieu lorsqu'elle se réveille simultanément avec toutes les tâches les plus prioritaires.

L'instant critique détermine donc le pire cas pour une tâche dans un ordonnancement à priorités fixes : il arrive lorsque l'interférence (retard) due aux tâches plus prioritaires est maximisée. L'interférence d'une tâche peut se visualiser

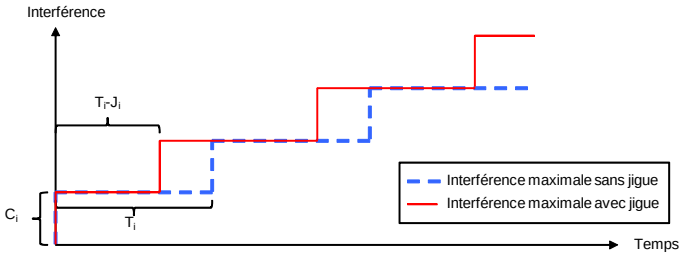

Fig. 1. Interférence maximale d'une tâche τ_i sur les tâches moins prioritaires

sous forme d'une courbe en escalier (voir figure 1). Chaque marche de cet escalier représente le réveil de la tâche, et donc son interférence sur les tâches moins prioritaires. Cependant, lorsque les tâches peuvent souffrir d'une gigue d'activation (paramètre J_i), l'interférence qu'aura une tâche τ_i sur la tâche étudiée est maximisée lorsqu'elle subit initialement sa gigue d'activation. En effet, dans ce cas, cette tâche interfère avec la tâche étudiée à l'instant initial, puis à l'instant $T_i - J_i$, puis à l'instant $2T_i - J_i$, etc. D'où une modification de la définition 3 lorsqu'il y a des gigues d'activation.

Définition 4 (Instant critique [16]): En présence de gigue d'activation, l'instant critique pour une tâche a lieu lorsqu'elle se réveille simultanément avec toutes les tâches les plus prioritaires qui ont subi leur pire gigue d'activation.

On voit clairement sur la figure 1 que plus la gigue d'activation est importante, plus l'interférence qu'elle peut faire subir aux tâches moins prioritaires est élevée.

Le calcul du pire temps de réponse se base sur la notion de période d'activité : une période d'activité de niveau i est un intervalle de temps pendant lequel le processeur est toujours occupé à traiter des tâches de priorité supérieure ou égale à celle de τ_i. Considérons un système constitué de $n \geq 2$ tâches. Sans perte de généralité, nous supposons que les indices de tâches sont ordonnés par priorité décroissante (i.e. la priorité de τ_1 est la plus élevée). Nous nous intéressons à la période d'activité de niveau 2, donc nous ne considérons que les tâches τ_1 et τ_2. Supposons que les paramètres des tâches soient donnés uniquement par leur WCET et leur période (système sporadiquement périodique, tâches à échéance sur requête), avec $\tau_1 \langle C_1 = 1, T_1 = 4 \rangle$ et $\tau_2 \langle C_2 = 10, T_2 = 14 \rangle$. D'après la définition 3, l'instant critique pour τ_2 correspond à un réveil simultané avec τ_1.

Sur la figure 2, nous voyons qu'initialement, à l'instant critique, τ_1 et τ_2 sont activées, ainsi la demande est $C_1 + C_2 = 11$, et le processeur doit traiter ces 11 unités de temps de demande. La capacité de traitement du processeur est représentée par la droite $temps = demande\ processeur$, puisqu'il peut traiter une unité de temps à chaque unité de temps. A l'instant $T_1 = 4$, la tâche τ_1 est activée à nouveau, et requière $C_1 = 1$ unité de temps de traitement supplémentaire (notons que sous l'axe des abscisses, la présence du nom d'une tâche représente son activation au plus tôt à cette date). Il en va de même aux instants $2T_i = 8$, puis $3T_i = 12$. Notons que la fonction escalier de demande processeur représente la demande maximale possible, i.e. les tâches demandent leur pire durée d'exécution à chaque réveil, leurs réveils sont

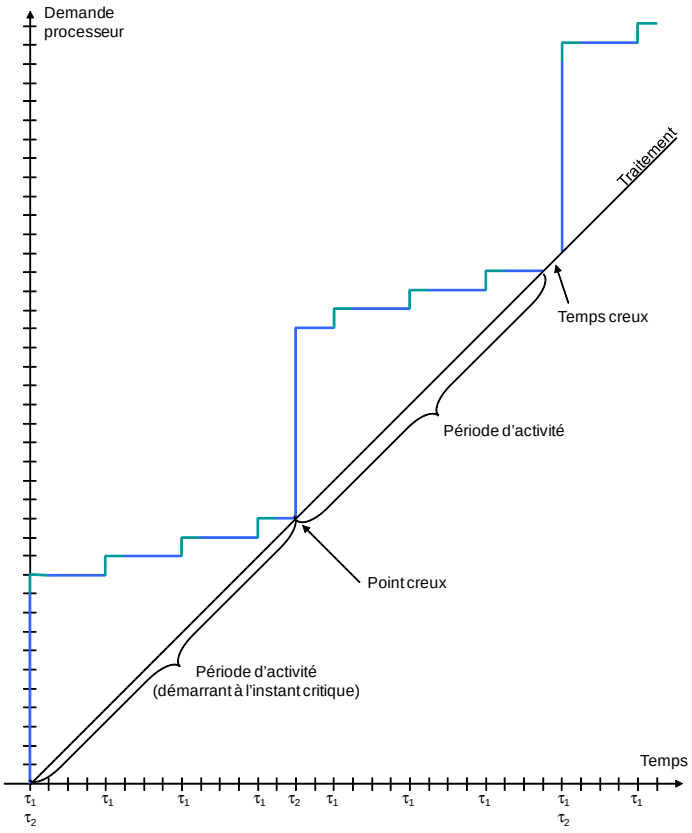

Fig. 2. Période d'activité, temps creux, point creux

les plus fréquents possible. A l'instant 14, un point creux a lieu : la fonction de la demande processeur rencontre l'axe de traitement. Cela signifie qu'à cet instant, toute la demande ayant eu lieu a été satisfaite. Par conséquent, comme nous avons représenté la période d'activité de niveau 2, à l'instant 14, sous l'hypothèse que la demande a été la pus importante possible, la tâche τ_2 se termine. Ainsi, sur la figure 2, la période d'activité initiée par l'instant critique dure 14 unités de temps, et s'achève par un point creux. Un point creux suffit à terminer une période d'activité; un temps creux peut bien entendu terminer aussi une période d'activité. Un point creux (de niveaux de priorité i) correspond à un instant où le processeur a traité la totalité des demandes des tâches de priorité aussi grande que la priorité de τ_i.

Une second période d'activité est représentée sur la figure 2, celle-ci est plus courte que la première puisqu'elle ne dure que 13 unités de temps. Cela est normal, d'après le Théorème 1.

Théorème 1 (Plus longue période d'activité [16], [17]): La plus longue période d'activité est initiée par un instant critique.

Une période d'activité permet de calculer, à l'aide d'une fonction en escalier représentant la demande processeur, l'instant auquel une tâche se termine en supposant le pire scénario d'activation des tâches.

Théorème 2 (Worst-case response time - WCRT [16], [17]): Le pire temps de réponse d'une tâche advient dans la plus longue période d'activité.

Par conséquent, l'étude de la première période d'activité représentée sur la figure 2 suffit à connaitre le pire temps de réponse de la tâche τ_2. D'un point de vue général, l'étude de la période d'activité de niveau i initiée par l'instant critique permet de déterminer le pire temps de réponse (WCRT) de la tâche τ_i. Cela permet de déduire directement si un système est fiablement ordonnançable par une affectation de priorités donnée. Notons que la demande processeur est la somme de toutes les demandes processeur des tâches de priorité au moins aussi grande que celle de τ_i. On appelle aussi interférence la demande processeur des tâches de priorité supérieure à la tâche étudiée. Le terme anglais correspondant est *Request Bound Function*, on parle donc de l'étude de **RBF**. Plusieurs questions peuvent se poser ici :

a) Comment calculer la longueur de la période d'activité, quelle est la complexité du problème ?
b) Peut-on approximer polynomialement la date d'intersection entre RBF et traitement processeur (i.e. le premier point creux) ?
c) Que se passe-t-il si plusieurs instances de τ_i se trouvent dans cette période d'activité ?
d) Cette technique s'adapte-t-elle a des contextes plus généraux, comme au cas où des ressources critiques sont partagées ?

Nous étudions ces questions dans la suite de cette section.

B. Calcul de longueur de période d'activité

Pour commencer, nous supposons que le système étudié est sans gigue d'activation.

La complexité du calcul de longueur de période d'activité est NP-difficile au sens faible [12]. Cela signifie qu'il existe des algorithmes pseudo-polynomiaux, mais pas d'algorithme polynomial permettant de résoudre ce problème. L'algorithme utilisé par calculer la RBF est le suivant :

1) A l'instant initial considéré, la tâche τ_i doit être exécutée, donc la période d'activité ne peut pas être plus courte que son WCET C_i, on note $W_i^{(0)} = C_i$ cette durée.
2) Si la période d'activité dure au moins $W_i^{(0)}$, alors tout réveil de tâche de priorité supérieure (on note $hp(i)$ l'ensemble de indices des tâches de priorité supérieure à celle de τ_i) dans l'intervalle semi-ouvert $[0 - W_i^{(0)}[$ fait aussi partie de la période d'activité. Pour une tâche donnée τ_j avec $j \in hp(i)$, l'interférence de τ_j, et donc son influence dans un intervalle de temps ouvert $[0, t[$ est au plus :
$$RBF_j(t) = \left\lceil \frac{t}{T_j} \right\rceil C_j \quad (1)$$
La durée de la période d'activité, si elle dure au moins $W_i^{(0)} = C_i$, dure donc au moins $W_i^{(1)}$ avec :
$$W_i^{(1)} = C_i + \sum_{j \in hp(i)} RBF_j(W_i^{(0)})$$
3) Par conséquent, la période d'activité maximale dure au moins $[0 - W_i^{(1)}[$, il faut reprendre le point 2 jusqu'à arriver à un point fixe (i.e. l'intervalle ne croît plus), afin de trouver la durée de la période d'activité, et par là, la position du premier point creux. Nous notons $W_i^{(*)}$ cette durée. Remarquons que τ_i n'est pris en compte qu'une seule fois dans le calcul.

La formule générale de calcul de plus grande durée de période d'activité dans le cas où τ_i n'apparait qu'une seule fois dans la période d'activité est donc le plus petit point fixe [9] de :
$$W_i^{(*)} = C_i + \sum_{j \in hp(i)} RBF_j(W_i^{(*)})$$

Si $W_i^{(*)}$ n'est pas assez grand pour que τ_i ait démarré sa seconde instance, alors le pire temps de réponse de τ_i, d'après le Théorème 2, est le temps de réponse de l'instance de τ_i ayant démarré à l'instant critique. Cela implique que le pire temps de réponse possible pour la tâche τ_i est : $WCRT_i = W_i^{(*)}$.

Cependant, si $W_i^{(*)} > T_i$, cela signifie qu'une seconde instance de τ_i est présente dans la période d'activité initiée par l'instant critique. Il en résulte, d'après le Théorème 2, que cette seconde instance peut avoir un pire temps de réponse que la première instance, et qu'il faut donc le déterminer. C'est le principe de la formule proposée par [10], qui est une extension de la formule de [9] au cas des instances suivantes.

Même si il y a plus d'une instance de τ_i dans la période d'activité étudiée de niveau i, alors la durée de la période d'activité $W_i^{(*)}(1)$ limitée à une seule instance de τ_i est donnée par la formule de [9] sur l'équation 2. Le pire temps de réponse pour la première instance de la tâche étudiée en est directement déduit par l'équation 3 :
$$W_i^{(*)}(1) = C_i + \sum_{j \in hp(i)} RBF_j(W_i^{(*)}(1)) \quad (2)$$
$$WCRT_i(1) = W_i^{(*)}(1) \quad (3)$$

Si $WCRT_i(1) \geq T_i$, alors il faut calculer la longueur de la période d'activité contenant deux instances de τ_i :
$$W_i^{(*)}(2) = 2C_i + \sum_{j \in hp(i)} RBF_j(W_i^{(*)}(2))$$

Et le pire temps de réponse pour cette instance est donné par l'écart entre le réveil de cette instance (date T_i) et la date de fin de la période d'activité :
$$WCRT_i(2) = W_i^{(*)}(2) - T_i$$

On est amené alors à calculer le temps de réponse de chaque instance présente dans la période d'activité qui est potentiellement le pire temps de réponse pour la tâche τ_i. La formule générale de [10] est alors donnée, k correspondant au nombre d'instances de τ_i présentes, par :
$$W_i^{(*)}(k) = kC_i + \sum_{j \in hp(i)} RBF_j(W_i^{(*)}(k)) \quad (4)$$
$$WCRT_i(k) = W_i^{(*)}(k) - (k-1)T_i \quad (5)$$
$$WCRT_i = \max_k (WCRT_i(k)) \quad (6)$$

Tant que $WCRT_i(k) > T_i$, on calcule $WCRT_i(k+1)$, puisque dans ce cas, la $k+1^{ème}$ instance de τ_i fait aussi partie de la période d'activité initiée par l'instant critique, et

se retrouve de ce fait candidate potentielle au pire temps de réponse.

C. Approximation

Il peut être intéressant, notamment lorsque les systèmes de tâches sont très grands, ou bien lorsque l'on souhaite faire de l'analyse de sensibilité (i.e. on s'intéresse au domaine de variation des paramètres de sorte à conserver l'ordonnançabilité du système) [18], [19], d'avoir un test polynomial donnant une valeur approchée du pire temps de réponse.

On trouve des bornes approchées calculables en temps linéaire dans [20], [21], cependant, [22] montre que ces approximations ne possèdent pas de borne d'erreur constante, et que, d'un point de vue général, les bornes de temps de réponse calculables en temps linéaire ne peuvent pas garantir de borne d'erreur constante. L'augmentation de capacité du processeur à une vitesse deux fois plus rapide est le prix à payer pour que ces bornes ne soient pas pessimistes.

Par conséquent des schémas d'approximation ont été proposés. Un schéma d'approximation est un algorithme pouvant s'approcher de la réponse exacte (algorithme optimal) avec un facteur d'au plus $1 + \epsilon$ de l'optimal. Un schéma d'approximation complet (*Fully Polynomial Time Approximation Scheme* - FPTAS) est un schéma d'approximation pour lequel l'algorithme est polynomial en la taille de l'entrée et $1/\epsilon$. Des FPTAS pour l'approximation du calcul du temps de réponse ont été proposés dans [23], [24]. Ces FPTAS se basent sur une modification de la fonction RBF afin d'en considérer une approximation.

D. Prise en compte de facteurs pratiques

Les facteurs pratiques altèrent le modèle de tâches indépendantes afin de prendre en compte un fonctionnement interdépendant des tâches. Ainsi, des portions de tâches en exclusion mutuelle, des contraintes de précédence, des parties non préemptibles, des tâches qui se suspendent en attendant le résultat d'une opération d'entrée/sortie, des gigues d'activation, etc., sont autant de facteurs pratiques qui doivent être pris en compte afin de ne pas sous-estimer le pire temps de réponse des tâches. La RBF est extrêmement intéressante vis à vis des facteurs pratiques et dans bon nombre des cas suscités, la RBF a été adaptée. Cependant, le problème de calcul de temps de réponse devient NP-difficile au sens fort dans bien des cas, et par conséquent, la technique de RTA basée sur la RBF adaptée n'est plus qu'une condition suffisante d'ordonnançabilité (i.e. le pire temps de réponse calculé peut ne pas être atteignable).

1) Parties non préemptibles et exclusion mutuelle: Le fait que les tâches ne soient pas toujours préemptibles (que ce soit à cause du partage de ressources critiques, ou bien du fait que certaines tâches sont non préemptibles), rend le problème d'ordonnançabilité NP-difficile au sens fort (cas non-préemptif [25], cas préemptif [26]). De plus, il peut se produire des anomalies d'ordonnancement, c'est à dire que le cas le pire ne correspond pas forcément à l'utilisation par les tâches de leur WCET, ou aux réveils les plus fréquents. Dans le cas du partage de ressources, des protocoles de gestion de ressources ont été proposés [27], [28], de façon à éliminer le problème d'inversion de priorité, et à borner la pire durée de blocage des tâches. La pire durée de blocage d'une tâche τ_i, notée B_i, représente la pire durée pendant laquelle τ_i peut être bloquée par une tâche de priorité inférieure. Etant donné que la RTA présentée pour les tâches indépendantes prend en compte l'interférence des tâches plus prioritaires sur une tâche étudiée, si cette tâche peut aussi subir une interférence (on parle alors de blocage) de la part de tâches de priorité inférieure, alors il suffit de modifier légèrement la formule de calcul de la durée maximale de la période d'activité.

$$W_i^{(*)}(k) = B_i + kC_i + \sum_{j \in hp(i)} RBF_j(W_i^{(*)}(k))$$

Une tâche τ_i ne pouvant être bloquée qu'une seule fois par période d'activité, le facteur de blocage est simplement ajouté une fois à la fonction escalier représentant la demande processeur de niveau i et permettant de calculer la longueur de la période d'activité.

2) Gigue d'activation: La prise en compte d'une gigue d'activation est plus simple, car le problème de calcul de longueur de période d'activité reste dans ce cas NP-difficile au sens faible. La différence par rapport au cas des tâches indépendantes réside dans le choix de l'instant critique. Sans gigue d'activation, l'origine des temps considérée pour l'instant critique est 0 et correspond au réveil simultané de toutes les tâches prises en compte dans la période d'activité. En posant encore un instant critique ayant lieu à la date 0, d'après la définition 4 [16], en présence de gigue d'activation, chaque tâche prise en compte τ_j est activée à une date $-J_j$. Par conséquent il suffit de modifier l'équation 1 de la façon suivante :

$$RBF_j(t) = \left\lceil \frac{t + J_j}{T_j} \right\rceil C_j$$

Le calcul du pire temps de réponse d'une tâche doit prendre en compte le fait que la tâche étudiée τ_i est réveillée dans le pire des cas à la date $-J_i$, ce qui implique que le temps de réponse d'une instance de τ_i est une modification simple de l'équation 5 :

$$WCRT_i(k) = W_i^{(*)}(k) - (k-1)T_i + J_i$$

3) Autres facteurs pratiques: Les contraintes de précédence dites simples considèrent que les tâches soumises à des contraintes de précédence ont la même période. Même dans ce cas simple, le calcul de temps de réponse n'a pas de solution exacte pseudo-polynomiale connue. Ainsi [29] propose une modification des formules de RTA pour les adapter à des chaines non réentrantes de précédence de tâches. Cette méthode a été adaptée au cas de graphes réentrants de précédences dans [30].

Lorsque certaines tâches peuvent se suspendre, le problème d'ordonnançabilité devient NP-difficile au sens fort, il en va donc de même quant au calcul de temps de réponse [31].

E. Systèmes de tâches à dates de réveil connues

Tous les résultats présentés jusqu'à présent considèrent l'existence d'un instant critique, le plus défavorable vis à vis du réveil de la tâche dont on étudie le temps de réponse.

Cependant, dans certains cas, cet instant critique ne peut pas avoir lieu.

Il est courant, dans l'industrie du transport (automobile, aéronautique notamment), de réaliser une programmation dirigée par le temps. Dans ce cas, les tâches sont strictement périodiques. Afin d'éviter l'instant critique, et ainsi améliorer les temps de réponse des tâches, le concepteur pourra être amené à lisser la charge, c'est à dire à s'arranger pour que les réveils des tâches coïncident peu. La validation d'un modèle de tâches concrètes différées (et donc strictement périodiques) est un problème NP-difficile au sens fort [32]. Seule la simulation peut être utilisée pour tirer parti de l'amélioration d'ordonnançabilité apportée par l'élimination de l'instant critique. Cependant, celle-ci nécessite une étude des périodes d'activité situées sur une durée de l'ordre du plus petit commun multiple des périodes. La durée de simulation nécessaire a été notamment étudiée dans [33], [34], [35].

Lorsque certains facteurs gouvernant les réveils des tâches sont connus, mais pas tous, il existe d'autres modèles, plus avancés, qui font l'objet de la section suivante.

IV. Modèle des transactions et modèles multiframes

Dans le cas d'une programmation dirigée par les évènements, si l'on souhaite prendre en compte le décalage de réveils, deux modèles principaux ont été étudiés : les transactions et les modèles multiframes. Cette partie montre les différences des deux familles de modèles, ainsi que leurs points communs lors de l'analyse de temps de réponse, qui se base sur la RTA.

A. Transactions

Définition 5 (Transaction): [36] Un système de transactions Γ, est constitué de transactions Γ_i avec $i \in \{1..|\Gamma|\}$. Une transaction Γ_i possède une période T_i et contient des tâches τ_{ij} avec $j \in \{1..|\Gamma_i|\}$. Chaque tâche τ_{ij} est caractérisée par $\langle C_{ij}, O_{ij}, D_{ij}, J_{ij}, B_{ij}, P_{ij} \rangle$ où :

- T_i est la période minimale séparant deux activations successives de la transaction. Une transaction est non concrète, activée par un évènement,
- C_{ij} est le WCET de la tâche τ_{ij},
- O_{ij} est l'*offset*, ou décalage, du réveil de la tâche par rapport au réveil de la transaction,
- J_{ij} est la gigue d'activation de la tâche par rapport à son réveil prévu. Cela signifie que si la transaction Γ_i a été activée à la date t_0, alors la tâche τ_{ij} est activée entre les dates $t_0 + O_{ij}$ et $t_0 + O_{ij} + J_{ij}$. La gigue peut être arbitrairement grande au besoin,
- D_{ij} est le délai critique de τ_{ij}, et la tâche doit être terminée à la date $t_0 + O_{ij} + D_{ij}$, notons que ce paramètre n'est pas relié à d'autres paramètres, et peut être arbitrairement grand,
- B_{ij} représente la pire durée de blocage due à un protocole de gestion de ressources si les tâches partagent des ressources critiques, et P_{ij} est la priorité de τ_{ij} si l'on utilise un ordonnancement à priorités fixes.

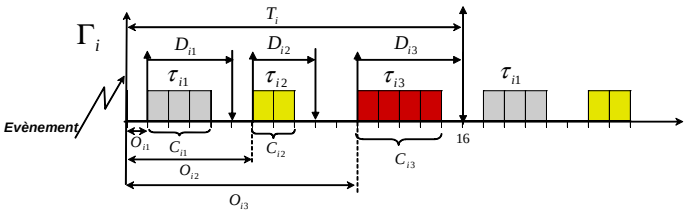

Fig. 3. Une transaction Γ_i

Le modèle des transactions est particulièrement intéressant pour représenter des tâches attendant un évènement déclencheur, qui est toujours suivi d'autres évènements relatifs au premier. Par exemple, il est particulièrement bien adapté à la modélisation de traitements effectués sur réception de données de type série [37]. La figure 3 représente une transaction composée de trois tâches.

B. Modèles multiframes

Le modèle multiframe a été introduit dans [38], [39]. Afin de permettre une meilleure compréhension des subtiles différences entre les modèles multiframes et le modèle des transactions, nous utiliserons pour présenter le modèle multiframe des notations similaires à celles utilisées pour les transactions, et qui s'éloignent en cela des notations classiques. Initialement l'idée sous-jacente au modèle multiframe était de représenter des tâches dont la durée d'exécution variait suivant un *pattern* régulier et connu. Une tâche multiframe contient des *frames* (sous-tâches) séparées de façon régulière et à échéance sur requête. Le modèle multiframe a été généralisé au modèle GMF (*Generalized Multiframe Model*) dans [40]. Comme pour les transactions, les "sous-tâches" (i.e. tâches d'une transaction, ou frames d'une tâche multiframe), le modèle GMF permet de représenter des échéances arbitraires et des séparations non régulières.

Définition 6 (Tâche GMF): [40] Une tâche GMF Γ_i est constituée d'un ensemble de frames. Chaque frame τ_{ij}, avec $j = 0..N_i - 1$ est caractérisée par un triplet $< C_{ij}, D_{ij}, S_{ij} >$ avec :

- C_{ij} le WCET de la frame τ_{ij},
- D_{ij} le délai critique de τ_{ij}, et soit t l'instant où τ_{ij} est activée, son échéance est donnée par $t + D_{ij}$,
- S_{ij} est la séparation minimale entre τ_{ij} et la frame suivante $\tau_{i,(j+1)modN_i}$,
- le délai minimal séparant deux instances sucessives d'une même frame τ_{ij} est donc donnée par $\sum_{j=0}^{N-1} S_{ij}$.

Il y a deux différences majeures entre modèle GMF et transactions :

- une transaction est activée par un évènement, cependant les tâches composant la transaction sont nécessairement activées dans l'intervalle $[t_0 + O_{ij}..t_0 + O_{ij} + J_{ij}]$ si la transaction est activée à la date t_0. Une frame d'une tâche GMF est activée par évènement, ainsi la frame τ_{ij+1} est activée dans l'intervalle ouvert $[t_{ij} + S_{ij}.. + \infty[$, en considérant que la frame précédente de la tâche a été activée à la date t_{ij}.

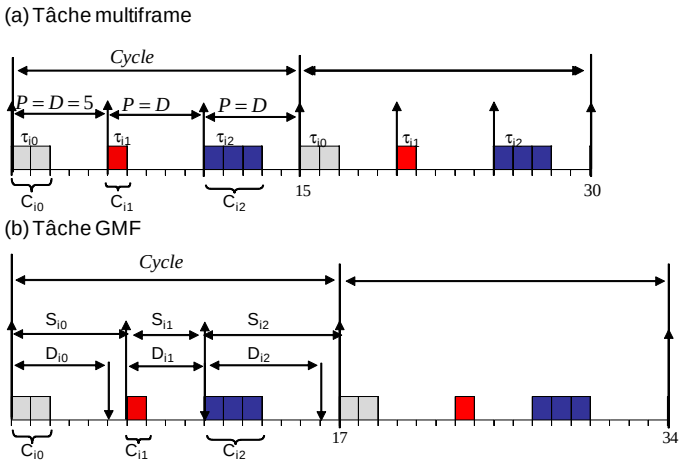

Fig. 4. Une tâche multiframe (a) et une tâche GMF (b)

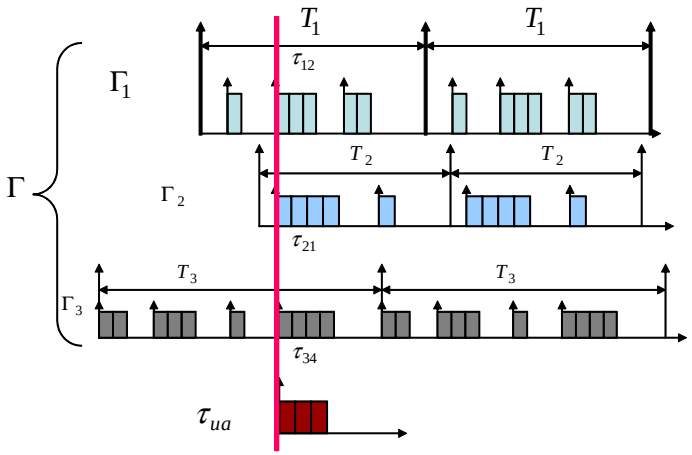

Fig. 5. Un instant critique candidat

- échéance relative : dans le modèle des transactions, l'échéance absolue est calculée relativement à la date d'activation sans gigue (i.e. à la date $t_0+O_{ij}+D_{ij}$). Dans le modèle GMF, l'échéance est calculée relativement à l'activation de la frame (i.e. $t_{ij} + D_{ij}$).

Par conséquent, les modèles sont non comparables dans les comportements qu'ils peuvent modéliser. Cependant, lorsque l'on étudie le pire temps de réponse, ou l'ordonnançabilité, c'est le comportement le plus défavorable qui est considéré. Si l'on restreint le modèle GMF à fonctionner avec le comportement créant l'interférence la plus élevée possible sur les autres tâches, alors les frames sont déclenchées au plus tôt, et le cas considéré est le cas strictement périodique. Dans ce cas, on obtient le même comportement au plus tôt pour une transaction Γ'_i, avec une période $T'_i = \sum_{j=0}^{N-1} S_{ij}$, composée de N tâches τ'_{ij} avec $O'_{i0} = 0$ et pour $j = 1..N-1$, $O'_{ij} = \sum_{k=0}^{j-1} S_{ik}$, et pour tout $j = 0..N-1$, $D'_{ij} = D_{ij}$, et $J'_{ij} = 0$.

Par conséquent, les méthodes permettant de calculer la pire interférence d'un système de tâches multiframe, ou d'un système de transaction, sur une tâche dont on veut calculer le temps de réponse, utilisent des techniques similaires.

Dans le cas du modèle GMF, et du modèle multiframe, deux hypothèses sont communément formulées :

- *Frame Separation* (FS) : les frames successives sont séparées (i.e. $D_{ij} \leq S_{ij}$),
- *localized Monotonic Absolute Deadline* (l-MAD) : l'échéance d'une frame ne peut pas avoir lieu avant celle d'une frame dont le réveil la précédait.

Le problème d'ordonnançabilité du modèle GMF sous l'hypothèse l-MAD a été étudié dans [40]. L'analyse de temps de réponse a été étudiée dans [41] sous l'assomption FS.

C. Calcul de temps de réponse

Etant donné que les études de temps de réponse de transactions ne reposent pas sur les restictions FS ou l-MAD, elles sont utilisables dans le contexte des transactions et dans le modèle GMF. L'analyse exacte de temps de réponse des transactions a été proposée initialement dans [42].

Le principe de l'analyse exacte de pire temps de réponse repose comme nous l'avons vu jusqu'à présent sur la notion d'instant critique.

Théorème 3 (Instant critique [43]): L'instant critique pour une tâche analysée τ_{ua} arrive lorsqu'une tâche candidate τ_{ic} de plus grande priorité est activée simultanément dans chaque transaction $\Gamma_i, i \neq u$, après avoir subi leur pire gigue d'activation.

Le problème est qu'il y a plusieurs instants critiques possibles. Ainsi, sur la figure 5, on ne sait pas quel scénario est le pire pour la tâche considérée. En théorie, si chacune des tâches représentées est de priorité plus grande que celle de la tâche analysée τ_{ua}, il y a $3 * 2 * 4 = 24$ instants critiques à étudier. Dans le cas général, il y a un nombre exponentiel de scénarios à étudier, et c'est le plus grand temps de réponse calculé parmi ces scénarios qui est le pire temps de réponse de la tâche étudiée. Un calcul approché de pire temps de réponse de complexité pseudo-polynomiale a donc été proposé dans [44], et amélioré dans [45], [46].

Le principe des calculs approchés de temps de réponse est de regrouper en une seule courbe RBF la pire interférence de chaque transaction. Ainsi, considérant l'exemple donné sur la figure 5, trois RBF différentes peuvent être considérées en fonction de la tâche de Γ_1 se réveillant à l'instant critique, comme cela est représenté sur la figure 6. Les techniques approchées reposent sur le calcul de la courbe composée des valeurs maximales des différentes RBF.

D. Modèles conditionnels

Plus récemment, le modèle GMF a été étendu à différents modèles prenant des aspects conditionnels en compte. Dans ces modèles, une frame peut avoir plusieurs successeurs alternatifs, et c'est en-ligne que le successeur sera choisi. Il convient donc d'étudier le cas le pire lors de l'étude d'ordonnançabilité. Les modèles sont représentés par des graphes qui expriment les alternatives, chaque nœud est une frame, et chaque chemin du graphe correspond en fait à une tâche GMF.

Dans [47], le modèle *Recurring Real-Time* (RRT) est proposé. Une tâche y est représentée par un graphe orienté sans

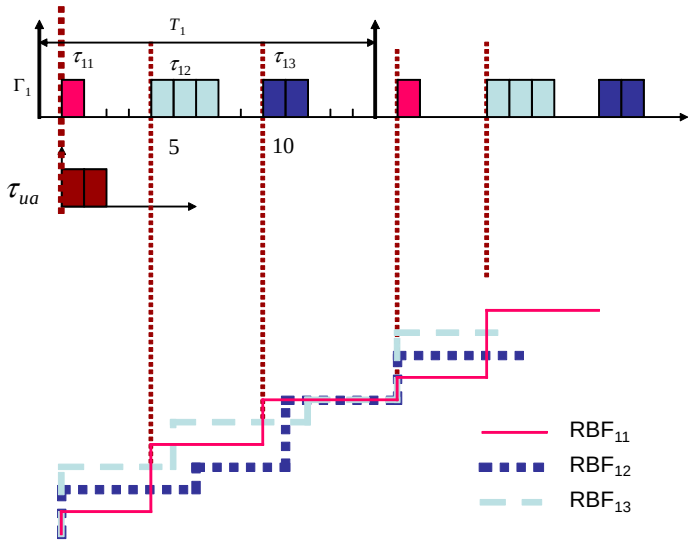

Fig. 6. Trois courbes d'interférence de la transaction Γ_1

cycle, possédant un unique nœud source et un unique nœud puits. Une période est donnée pour le graphe, qui est la période globale pour la tâche. Une tâche GMF peut être vue comme un cas particulier d'une tâche RRT avec un graphe qui est une chaîne. Sous la contrainte FS, l'analyse d'ordonnaçabilité et la RTA sont explorées dans [47].

Le ncRRT (*non-cyclic* RRT) est introduit dans [48]: cette extension permet d'avoir plusieurs nœuds puits, et ne contient pas de période globale (cette période peut cependant être calculée en parcourant le graphe, mais elle peut différer d'un chemin à un autre). Le problème d'ordonnançabilité du modèle ncRRT est étudié, sous la contrainte FS, dans [48].

Ce modèle a récemment été étendu en modèle digraphe dans [49]. Dans ce modèle, il n'y a pas de distinction de nœud source ou puits, et le point d'entrée du modèle peut être n'importe quel nœud (frame) du graphe. Là encore, le problème d'ordonnançabilité a été étudié sous l'hypothèse FS.

L'analyse de ces modèles (RRT, ncRRT, digraphe) a lieu en deux phases :

1) exploration des chemins du graphe, avec construction, si l'on étudie le pire temps de réponse, de la RBF associée, si l'on étudie l'ordonnançabilité, de la DBF associée. Au fur et à mesure que l'on parcoure tous les chemins, on conserve la DBF maximale ou la RBF maximale suivant le problème étudié. Notons que le fait de considérer la valeur maximale de la DBF ne permet d'obtenir qu'une valeur approchée, pas forcément atteignable, de la durée de la plus longue période d'activité, et par conséquent qu'une valeur approchée du pire temps de réponse calculé à partir de cette durée. Au contraire, l'étude de la RBF dans le cadre de l'étude du problème d'ordonnançabilité donne un test exact d'ordonnançabilité du système.
2) confronter la tâche étudiée aux bornes maximales de RBF ou de DBF suivant le problème étudié. Pour tous ces modèles, le problème de calcul de temps de réponse est NP-difficile au sens fort (pas d'algorithme pseudo-polynomial exact), alors que le problème d'ordonnançabilité est NP-difficile au sens faible.

V. Conclusion

Nous avons vu que la technique de calcul de temps de réponse basée sur les formules de Joseph et Pandya, et celles de Lehoczky, sont très flexibles et ont été modifiées simplement pour s'adapter à différents facteurs pratiques. Le principe de calcul de plus longue période d'activité de niveau i pour déduire un pire temps de réponse a été adapté au calcul de temps de réponse dans ces modèles plus fins : les transactions, le modèle GMF, et les modèles conditionnels qui généralisent le modèle GMF comme le RRT. Cependant, le nombre exponentiel d'instants critiques potentiels rend le problème de calcul de temps de réponse NP-difficile au sens fort. L'analyse d'ordonnançabilité est un problème plus simple pour tous ces modèles puisqu'elle reste NP-difficile au sens faible.

References

[1] J. Xu and D. Parnas, "On satisfying timing constraints in hard real-time systems," *IEEE Transactions on Computers*, vol. 19(1), pp. 70–84, 1993.

[2] J. Xu, "Multiprocessor scheduling of processes with release times, deadlines, precedence, and exclusion relations," *Transactions on Software Engineering*, vol. 19(2), pp. 139–153, 1993.

[3] E. Grolleau and A. Choquet-Geniet, "Off-line computation of real-time schedules using petri nets," *Discrete Event Dynamic Systems*, vol. 12(3), pp. 311–333, 2002.

[4] Z. Gu, "Solving real-time scheduling problems with model-checking," in *Embedded Software and Systems* (L. Yang, X. Zhou, W. Zhao, Z. Wu, Y. Zhu, and M. Lin, eds.), vol. 3820 of *Lecture Notes in Computer Science*, pp. 186–197, Springer Berlin / Heidelberg, 2005.

[5] C. Liu and J. Layland, "Scheduling algorithms for multiprogramming in a hard-real-time environment," *Journal of the ACM*, vol. 20, no. 1, pp. 46–61, 1973.

[6] "Special issue on worst-case execution-time analysis," *Real-Time Systems*, vol. 18, no. 2/3, 2000.

[7] E. Grolleau, "Tutorial on real-time scheduling," in *Ecole d'été temps réel, ETR'07*, (Nantes), 2007.

[8] K. Jeffay and D. Stone, "Accounting for interrupt handling costs in dynamic priority task systems," *Real-Time Systems Symposium*, pp. 212–221, 1993.

[9] M. Joseph and P. Pandya, "Finding response times in real-time system," *The Computer Journal*, vol. 29(5), pp. 390–395, 1986.

[10] J. Lehoczky, "Fixed priority scheduling of periodic task sets with arbitrary deadlines," in *IEEE Real-Time Systems Symposium*, pp. 201–213, 1990.

[11] F. Eisenbrand and T. Rothvoß, "Edf-schedulability of synchronous periodic task systems is conp-hard," in *Proc. ACM-SIAM Symposium on Discrete Algorithms (SODA10)*, (Austin, Texas), pp. 1029–1034, 2010.

[12] F. Eisenbrand and T. Rothvoß, "Static-priority real-time scheduling: Response time computation is np-hard," in *Proc. 29th Real-Time Systems Symposium*, (Barcelona, Spain), pp. 397–406, 2008.

[13] POSIX.1-2008, "The open group base specifications." Also published as IEEE Std 1003.1-2008, July 2008.

[14] ISO 17356, "OSEK (offene systeme und deren schnittstellen für die elektronik in kraftfahrzeugen)," 1997.

[15] S. T. Taft, R. A. Duff, R. L. Brukardt, E. Ploedereder, and P. Leroy, *Ada 2005 Reference Manual. Language and Standard Libraries: International Standard ISO/IEC 8652/1995(E) with Technical Corrigendum 1 and Amendment 1 (Lecture Notes in Computer Science)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2007.

[16] N. Audsley, "Optimal priority assignment and feasibility of static priority tasks with arbitrary start times," Tech. Rep. YCS 164, University of York, 1991.

[17] N. Audsley, A. Burns, R. Richardson, K. Tindell, and A. Wellings, "Applying new scheduling theory to static priority preemptive scheduling.," *Software Engineering Journal, 8(5):284-292*, September 1993.

[18] S. Vestal, "Fixed-priority sensitivity analysis for linear compute time models," *IEEE Transactions on Software Engineering*, vol. 20(4), pp. 308–317, 1994.

[19] E. Bini, M. Di Natale, and G. Buttazzo, "Sensitivity analysis for fixed-priority real-time systems," *Real-Time Systems*, vol. 39, pp. 5–30, 2008.

[20] M. Sjödin and H. Hansson, "Improved response-time analysis calculations," in *Proc. Real-Time Systems Symposium*, pp. 399–408, 1998.

[21] E. Bini and S. K. Baruah, "Efficient computation of reponse time bounds under fixed-priority scheduling," in *15th International Conference on Real-Time and Network Systems, RTNS'07*, (Nancy, France), March 29-30 2007.

[22] E. Bini, T. H. C. Nguyen, P. Richard, and S. K. Baruah, "A response-time bound in fixed-priority scheduling with arbitrary deadlines," *IEEE Transactions on Computers*, vol. 58, pp. 279–286, 2009.

[23] N. Fisher and S. Baruah, "A fully polynomial-time approximation scheme for feasibility analysis in static-priority systems with arbitrary relative deadlines," *Real-Time Systems, Euromicro Conference on*, vol. 0, pp. 117–126, 2005.

[24] T. H. C. NGuyen, P. Richard, and E. Bini, "Approximation techniques for response-time analysis of static-priority tasks," *Real-Time Systems*, vol. 43, pp. 147–176, 2009.

[25] K. Jeffay, D. F. Stanat, and C. U. Martel, "On non-preemptive scheduling of periodic and sporadic tasks," in *IEEE Real-Time Systems Symposium*, 1991.

[26] A. Mok, *Fundamental Design Problems of Distributed Systems for the Hard-Real-Time Environment*. PhD thesis, Laboratory for Computer Science, Massachusetts Institute of Technology, 1983.

[27] L. Sha, R. Rajkumar, and J. Lehoczky, "Priority inheritance protocols : an approach to real-time synchronization," *IEEE Transactions on Computers*, vol. 39, no. 9, pp. 1175–1185, 1990.

[28] C. Kaiser, "Exclusion mutuelle et ordonnancement par priorité," *Technique et Science Informatiques*, vol. 1, pp. 59–68, 1982.

[29] M. Harbour, M. Klein, and J. Lehoczky, "Fixed priority scheduling of periodic tasks with varying execution priority," in *Proceedings of Real-Time Systems Symposium*, pp. 116–128, San Antonio (Texas), 1991.

[30] M. Richard, P. Richard, E. Grolleau, and F. Cottet, "Contraintes de précédence et ordonnancement mono-processeur," in *Proc. Real-Time Systems*, pp. 121–138, 2002.

[31] F. Ridouard, P. Richard, and F. Cottet, "Negative results for scheduling independent hard real-time tasks with self-suspensions," *Proceedings of the 25th IEEE International Real-time Systems Symposium (RTSS04)*, December 2004.

[32] J. Leung and J. Whitehead, "On the complexity of fixed-priority scheduling of periodic real-time tasks," *Performance Evaluation(2)*, pp. 237–250, 1982.

[33] J. Leung and M. L. Merrill, "A note on preemptive scheduling of periodic, real-time tasks," *Information Processing Letters*, 1980.

[34] A. Choquet-Geniet and E. Grolleau, "Minimal schedulability interval for real time systems of periodic tasks with offsets," *Theoretical of Computer Sciences*, vol. 310, pp. 117–134, 2004.

[35] J. Goossens and R. Devillers, "Feasibility intervals for the deadline driven scheduler with arbitrary deadlines," in *Proceedings of the six International Conference on Real-time Computing Systems and Applications*, pp. 54–61, IEEE Computer Society Press, 1999.

[36] K. Tindell, "Adding time-offsets to schedulability analysis," *Technical Report YCS 221, Dept of Computer Science, University of York, England*, January 1994.

[37] K. Traore, E. Grolleau, and F. Cottet, "Simpler analysis of serial transactions using reverse transactions," in *Proc. International Conference on Autonomic and Autonomous Systems ICAS '06*, p. 11, July 16–18, 2006.

[38] A. Mok and D.Chen, "A multiframe model for real-time tasks," *IEEE Transactions on software Engineering*, pp. 635–645, October 1997.

[39] A. Mok and D. Chen, "A multiframe model for real-time tasks," in *Proc. of the 17th Real-Time Systems Symposium*, (Washignton), pp. 22–29, December 1996.

[40] S. Baruah, D. Chen, S. Gorinsky, and A. Mok, "Generalized multiframe tasks," *The International Journal of Time-Critical Computing Systems*, vol. (17), pp. 5–22, 1999.

[41] H. Takada and K. Sakamura, "Schedulability of generalized multiframe task sets under static priority assignment," *Proceedings-Fourth International Workshop on Real-Time Computing Systems and Applications*, pp. 80–86, October 1997.

[42] K. Tindell, "Using offset information to analyse static priority preemptively scheduled task sets," Tech. Rep. YCS-182, Dept. of Computer Science, University of York, England, 1992.

[43] K. Tindell, "Adding time-offsets to schedulability analysis," Tech. Rep. YCS-221, Dept. of Computer Science, University of York, England, 1994.

[44] J. Palencia and M. G. Harbour, "Schedulability analysis for tasks with static and dynamic offsets," in *Proc 19th IEEE Real-Time System Symposium*, 1998.

[45] J. Mäki-Turja and M. Nolin, "Efficient implementation of tight response-times for tasks with offsets," *Real-Time Systems Journal, Springer Netherlands*, 16 February 2008.

[46] A. Rahni, E. Grolleau, and M. Richard, "An efficient response time analysis for real-time transactions with fixed priority assignment," *Innovations in Systems and Software Engineering Journal (ISSE), Springer London*, vol. 5 (3), pp. 197–209, September 2009.

[47] S. K. Baruah, "Dynamic- and static-priority scheduling of recurring real-time tasks," *Real-Time Systems*, vol. 24, pp. 93–128, 2003. 10.1023/A:1021711220939.

[48] S. Baruah, "The non-cyclic recurring real-time task model," *Real-Time Systems Symposium, IEEE International*, vol. 0, pp. 173–182, 2010.

[49] M. Stigge, P. Ekberg, N. Guan, and W. Yi, "The digraph real-time task model," in *Proc. 17th Real-Time and Embedded Technology and Applications Symposium*, IEEE, 2011.