Mesures de l'équité d'une application temps-réel à l'aide de géométrie discrète

Annie Choquet-Geniet* — **Gaëlle Largeteau-Skapin**** **Abdoulaye Ouattara*****

* *LISI - ENSMA - Téléport 2 - 1 avenue Clément Ader - BP 40109 86961 Futuroscope Chasseneuil Cedex, France*

** *Laboratoire SIC, Université de Poitiers, BP 30179 86961 Futuroscope Chasseneuil Cedex, France*

*** *Université Polytechnique de Bobo Dioulasso - Burkina Fasso*

annie.geniet@univ-poitiers.fr, glargeteau@sic.univ-poitiers.fr

RÉSUMÉ. *Nous nous plaçons dans le contexte de l'analyse hors-ligne d'applications temps-réel constituées de tâches pouvant partager des ressources critiques, en environnement multiprocesseur. Nous adoptons une approche à base de géométrie discrète. Nous nous intéressons aux comportements équitables de l'application. Dans un premier temps, nous les avons caractérisés géométriquement, afin de pouvoir les extraire du modèle général. S'il n'existe pas de tels comportements, ce qui peut se produire puisqu'en présence de ressources critiques les ordonnancements équitables ne sont plus optimaux, nous souhaitons extraire les séquences les plus équitables. Nous proposons pour cela des indicateurs de qualité mesurant l'équité des séquences, qui peuvent être combinés avec le modèle géométrique.*

ABSTRACT. *We address off-line scheduling for real-time applications composed of tasks that may share critical resources, in multiprocessor environment. We adopt a discrete geometry model based approach. We are interested in the fair behaviours of the application. In a first time, we have characterized them by means of geometrical properties, so that we can then extract them from the general model. If no fair schedules exist, what may arises, since when resources are involved, fair strategies are no longer optimal, we aim to get as much fair as possible schedules. For that purpose, we propose fairness benchmarks, which can be combined with the geometrical model.*

MOTS-CLÉS : *ordonnancement multiprocesseur, P-équitable, mesure d'équité, géométrie discrète.*

KEYWORDS: *multiprocessor scheduling, PFairness, fairness measure, discrete geometry.*

JESA – 43/2009. MSR 2009, pages 1065 à 1080

1. Introduction

Les applications temps-réel ont pour vocation de garantir le bon fonctionnement et la sécurité de procédés physiques sensibles (système d'allumage d'un véhicule, avion en vol, robot, centrale nucléaire...). Elles sont assujetties à la dynamique du procédé : leur vitesse de réaction doit être adaptée à la vitesse d'évolution de celui-ci, et elles doivent à ce titre respecter les contraintes temporelles qui en découlent. Classiquement, une application temps-réel consiste en un ensemble de tâches périodiques interagissantes, qui s'exécutent de manière régulière. L'un des enjeux de la conception de ces applications est de garantir que les contraintes temporelles seront toujours respectées, et que les interactions seront correctement gérées (respect de l'exclusion mutuelle, absence d'interblocage...). Ceci relève de la problématique de l'ordonnancement. Nous considérons ici des architectures *multiprocesseur* constituées de m processeurs identiques. Nous supposons que l'ordonnancement est *global* : les tâches ne sont jamais définitivement affectées à un processeur donné, elles peuvent migrer à tout moment d'un processeur à un autre. Dans le cas général, déterminer l'existence d'une séquence valide en environnement multiprocesseur est un problème NP-difficile (Andersson *et al.*, 2000; Leung *et al.*, 1982). De plus, il a été prouvé qu'il ne peut pas exister de méthode d'ordonnancement en ligne optimale [1] (Hong *et al.*, 1992). Cependant, si les tâches sont indépendantes, les stratégies P-équitables PF, PD et PD^2, proposées par (Baruah *et al.*, 1996; Anderson *et al.*, 2004), sont des stratégies optimales. Si les tâches ne sont pas indépendantes, une approche alternative consiste à utiliser des techniques d'ordonnancement avant exécution : une séquence est calculée avant l'exécution de l'application et est ensuite implémentée dans une table au niveau du séquenceur. Ces techniques s'appuient généralement sur des approches orientées modèles. Celles-ci présentent deux atouts majeurs : tout d'abord, elles limitent au maximum la surcharge processeur liée à l'ordonnancement, puisque aucun algorithme d'ordonnancement n'est exécuté ; d'autre part, elles sont le plus souvent basées sur une exploration exhaustive de l'espace des solutions (l'ensemble de toutes les séquences possibles) et donc, s'il existe une séquence valide, elles sont assurées de la trouver. Cela confère donc à ces méthodes une puissance d'ordonnancement supérieure à celle des méthodes en-ligne (i.e. qui consiste à implémenter un algorithme qui s'exécute en même temps que l'application). Ceci a néanmoins des contreparties : les techniques de modélisation de l'application et de calcul des séquences valides ont la plupart du temps des complexités relativement élevées ; les techniques d'ordonnancement avant exécution sont moins souples que les techniques en-ligne, puisque toute évolution de l'application nécessite de reprendre en intégralité le calcul de la séquence à suivre. Plusieurs approches ont été développées dans la littérature, à base de réseaux de Petri, d'automates finis, de chaînes de Markov (Grolleau *et al.*, 2002; Largeteau *et al.*, 2001; Geniet *et al.*, 2001; Chauvière *et al.*, 2007). L'approche que nous adoptons ici, à base de géométrie discrète (Largeteau *et al.*, 2005), reprend la même

1. Un algorithme d'ordonnancement est dit optimal si, quelle que soit l'application considérée, soit il produit une séquence valide, i.e. qui respecte toutes les contraintes temporelles, soit aucun autre algorithme ne le pourra.

démarche que l'approche à base d'automates, mais de façon beaucoup plus efficace, tant au niveau de la construction du modèle que de son analyse. Ce modèle associe un objet géométrique dans l'espace (temps CPU, temps) à chaque tâche. La forme de cet objet géométrique dépend directement des contraintes temporelles auxquelles la tâche est soumise. La concurrence s'exprime par un produit cartésien. Le partage de ressource est, quant à lui, modélisé par une soustraction des points correspondant à une utilisation invalide de la ressource. Suivant les propriétés topologiques de l'objet final, nous pouvons déterminer si l'application est ordonnançable ou non, éventuellement pour une architecture donnée. Nous pouvons également préciser, dans le cas ordonnançable, le nombre minimal de processeurs nécessaires. Le principal avantage de cette méthode est que, même si l'application n'est pas ordonnançable, l'objet produit n'est pas vide, il permet alors d'analyser les causes de l'invalidité et de quantifier le taux d'invalidité du système.

Notre objectif est d'une part de caractériser géométriquement les comportements équitables afin de pouvoir les extraire du modèle général, s'il en existe. D'autre part, nous souhaitons, s'il n'existe pas de comportements équitables compatibles avec les règles de manipulation de ressources, sélectionner les comportements qui sont les plus équitables possible. En effet, outre qu'elles sont efficaces pour des tâches indépendantes, les stratégies P-équitables présentent des atouts : elles garantissent une exécution régulière des tâches, ce qui peut être très intéressant pour certains types d'applications, en particulier dans le domaine multimédia ; elles permettent une gestion efficace des tâches apériodiques contraintes (Choquet-Geniet *et al.*, 2009). Pour cela nous construisons un modèle géométrique P-équitable qui modélise l'ensemble des comportements P-équitables de l'application, puis nous montrons comment prendre en compte le partage de ressources. Si l'objet obtenu ne possède pas les bonnes propriétés topologiques, il n'existe pas de séquences équitables respectant les règles d'accès aux ressources pour l'architecture cible. Dans ce cas, nous considérons l'ensemble de toutes les séquences valides, d'où nous souhaitons extraire les séquences les plus équitables. Pour cela, nous proposons plusieurs modèles numériques permettant de quantifier l'équité d'une séquence.

L'organisation du papier est la suivante : dans la section 2, nous présentons l'ordonnancement P-équitable. En section 3, nous présentons le modèle géométrique sur lequel nous nous appuyons. Puis en section 4, nous présentons les mesures d'équité que nous proposons. Enfin, la section 5 donne quelques conclusions et perspectives.

2. Ordonnancement P-équitable

Dans la suite, pour tout réel x, $\lfloor x \rfloor$ désigne la partie entière inférieure de x et $\lceil x \rceil$ la partie entière supérieure.

Nous rappelons que nous considérons des plateformes comportant m processeurs identiques. Nous considérons des applications composées de n tâches périodiques à échéances sur requêtes $\tau_i(C_i, P_i)$. C_i représente la pire durée d'exécution de la

tâche et P_i sa période. Nous supposons que les tâches sont soumises à des contraintes d'échéances strictes, et qu'elles peuvent partager des ressources communes (variables partagées, segments mémoire ...). Une tâche consiste en un ensemble infini d'instances activées aux instants $k.P_i$ ($k \in \mathbb{N}$) et à chaque nouvelle activation, l'instance précédente doit avoir terminé son exécution. Nous supposons que le parallélisme est interdit : une tâche ne peut pas s'exécuter sur plusieurs processeurs simultanément. Nous supposons enfin que les paramètres temporels sont connus et déterministes.

Formellement, un ordonnancement est une application $S : \mathbb{N} \times \{1, \ldots, n\} \rightarrow \{0, 1\}$ telle que $\sum_{i=1}^{n} S(t, i) \leq m$. Intuitivement, $S(t, i) = 1$ signifie que la tâche τ_i est exécutée pendant l'intervalle de temps $[t, t + 1[$ sur un processeur.

Les stratégies P-équitables ont été développées pour les systèmes multiprocesseur. Ce sont des stratégies optimales pour des systèmes de tâches indépendantes en environnement multiprocesseur. L'idée de base est que les tâches s'exécutent de manière régulière, à une vitesse égale à leur facteur d'utilisation ($u_i = \frac{C_i}{P_i}$). Cela signifie qu'à l'instant t, la tâche τ_i doit avoir exécuté $\frac{C_i}{P_i}.t$ unités de temps processeur. Mais le nombre d'unités de temps processeur devant être entier, ce nombre est approximé soit par $\lfloor u_i.t \rfloor$ soit par $\lceil u_i.t \rceil$. De manière formelle :

Définition 1. *Un ordonnancement S est P-équitable si et seulement si*

$$\forall t \in \mathbb{N}, \forall i \in \{1, \ldots, n\}, -1 < u_i.t - \sum_{j=0}^{t-1} S(j, i) < 1.$$

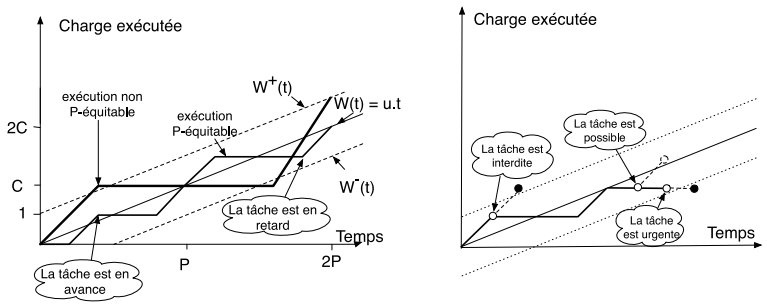

Figure 1. *Exécutions P-équitable et non P-équitable.*

La figure 1 illustre la notion d'équité. Pour chaque tâche, la courbe représentant la charge déjà exécutée doit être située entre les deux droites d'équation $y = \frac{C_i}{P_i}.t \pm 1$.

A chaque instant t, si le point $C_i(t)$ est au-dessus de la droite idéale, la tâche est dite **en avance**, si elle est sur la droite idéale, elle est dite **ponctuelle** et si elle est en dessous de la droite idéale, elle est dite **en retard**. Pour construire un ordonnancement P-équitable, à chaque instant $t \in \mathbb{N}$, l'ensemble des tâches $\{\tau_1, \tau_2, \ldots, \tau_n\}$ est partitionné en trois sous-ensembles (voir figure 1) :

– Les tâches **urgentes** qui sont les tâches en retard qui, si elles n'étaient pas exécutées à l'instant t, seraient à l'instant $t+1$ sous la droite limite inférieure.

– Les tâches **interdites** qui sont les tâches en avance qui, si elles étaient exécutées à l'instant t, seraient à l'instant $t+1$ au-dessus de la droite limite supérieure.

– Les tâches **possibles** dont ni l'exécution ni la non exécution à l'instant t ne conduit à violer la condition de P-équité.

Puis toutes les tâches urgentes sont exécutées. Les tâches possibles sont ensuite triées, et les $m - \mid Urgentes \mid$ premières d'entre elles sont exécutées.

Plusieurs algorithmes P-équitables ont été proposés dans la littérature (PF, PD et PD^2 (Baruah *et al.*, 1996; Anderson *et al.*, 2004)). Ils diffèrent dans la façon dont les tâches possibles exécutées sont sélectionnées. Notons enfin que, pour des raisons d'implémentation, la P-équité est souvent définie à l'aide de fenêtres d'exécution des sous-tâches unitaires. Toute tâche τ_i est décomposée en une infinité de sous-tâches τ_i^j de durée 1. La $j^{ème}$ ($j > 0$) sous-tâche doit alors s'exécuter dans une fenêtre temporelle I_i^j délimitée par $r_i^j = \lfloor \frac{j-1}{u_i} \rfloor$ et $d_i^j = \lceil \frac{j}{u_i} \rceil$. Ces fenêtres d'exécution sont déduites de la définition. La figure 2 donne un exemple de fenêtres et le principe de leur détermination à partir des droites W et W^+. Définir les fenêtres revient à paver la zone de validité (située entre les deux droites limite). L'exécution d'une sous-tâche correspond à un segment de droite de pente 1. Chaque segment de pente 1 doit donc être situé à l'intérieur de son pavé.

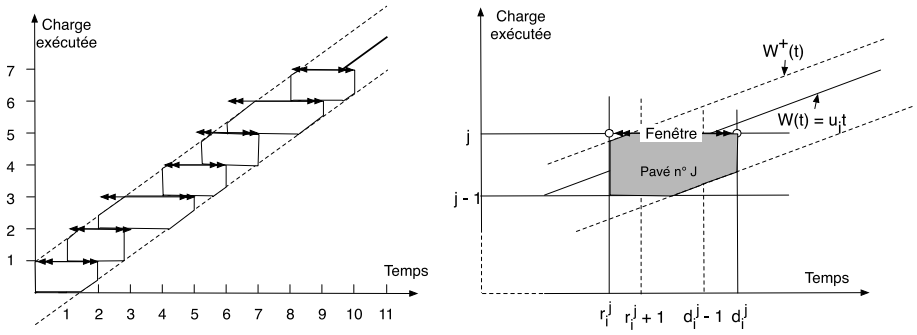

Figure 2. *Fenêtres d'exécution pour la tâche <8, 11> - calcul des fenêtres et des pavés*

3. Définitions du modèle géométrique de représentation

Cette section présente le modèle géométrique associé à l'ensemble des états valides d'une tâche qui garantissent une utilisation équitable de la ressource processeur. Le modèle géométrique d'une tâche est présenté en détail dans (Largeteau *et al.*, 2004; Largeteau *et al.*, 2005; Largeteau-Skapin *et al.*, 2008), le modèle géométrique de l'équité est présenté dans (Largeteau-Skapin *et al.*, 2009).

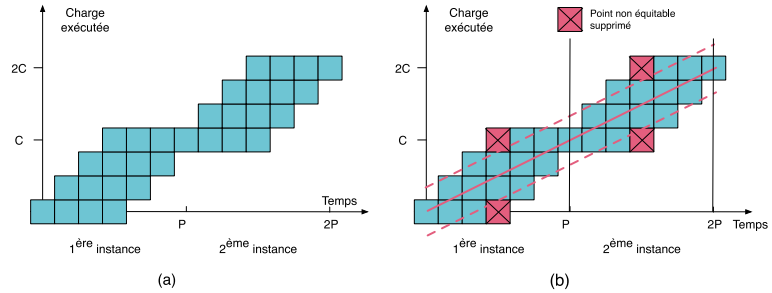

Figure 3. *a) modèle géométrique ; b) modèle de l'équité pour la tâche* $< 3, 7 >$

3.1. *Espace de représentation des états d'une tâche*

Nous considérons pour chaque tâche, un espace E à 2 dimensions dans lequel l'ordonnée correspond à l'avancement de la tâche (i.e. la charge cumulée qu'elle a déjà exécutée) et l'abscisse correspond au temps. Lors de son exécution, les états successifs de la tâche τ dépendent de l'attribution ou non du CPU. Chaque état de la tâche correspond à un point de l'espace E et une séquence d'exécution de la tâche est représentée par une courbe (la *"trajectoire de la tâche"*) de l'espace (temps,temps CPU). La tâche ne peut être dans deux états à la fois et son état ne peut pas être indéfini. Elle doit également respecter ses contraintes temporelles $< C, P >$. Ceci se traduit par un certain nombre de contraintes géométriques : nous notons $r_i = i.P$ le réveil de la $(i+1)^{\text{ème}}$ instance de la tâche ($r_{i+1} = r_i + P$ est son échéance).

Définition 2. *On appelle* ***trajectoire valide*** *de* $\tau < C, P >$ *toute application croissante* $Tr_\tau\ \mathbb{N}^+ \rightarrow \mathbb{N}^+$ *telle que :* $\begin{cases} Tr_\tau([r_i, r_{i+1}]) & = & [i.C, (i+1).C] \\ Tr_\tau(t+1) & = & Tr_\tau(t) \text{ ou } Tr_\tau(t)+1 \end{cases}$
L'ensemble des trajectoires valides est noté $TV(\tau)$.

Nous pouvons noter qu'il y a isomorphisme entre les trajectoires et les séquences d'ordonnancement de la tâche. Nous pouvons donc employer indifféremment l'un ou l'autre terme. Le modèle géométrique d'une tâche, noté $Gr(\tau)$, est l'ensemble des points d'une trajectoire valide de τ, i.e. l'ensemble des états valides temporellement et atteignables de τ (cf. figure 3).

Définition 3. *Le domaine de validité* $Gr(\tau)$ *des états d'une tâche* τ *est défini par :*

$$Gr(\tau) = \bigcup_{\psi \in TV(\tau)} \{(t, \psi(t))\}.$$

Pour être équitable, une tâche doit exécuter, entre r_i et r_{i+1}, les C unités de temps de son code le plus régulièrement possible. Le taux d'exécution idéal est alors $\frac{C}{P}$. Un état d'une tâche est dit équitable s'il est valide et ne s'éloigne pas de l'exécution

idéale. Nous pouvons définir géométriquement les états équitables de la façon suivante (cf figure 3) :

Définition 4. *L'ensemble des états équitables de la tâche* $\tau < C, P >$ *est défini par :*

$$Gr_E(\tau) = \{(t, y) \in \mathbb{N}^2,\ -P < Ct - Py < P\}$$

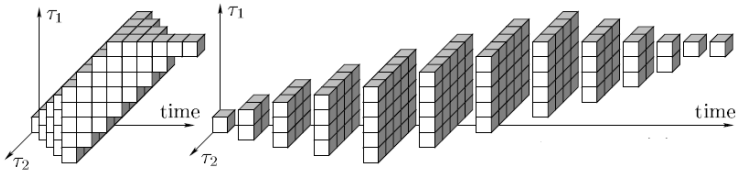

Figure 4. *Représentation géométrique du fonctionnement de deux tâches*

3.2. *Intégration de la concurrence*

L'état du système $\Gamma = (\tau_i)_{i\in[1,n]}$ à un instant t est défini par l'état d'avancement de chacune des tâches τ_i à cet instant t. La validité d'un état à l'instant t du système dépend de la validité des états de chacune de ses tâches à l'instant t. Le modèle géométrique de l'application peut donc se définir à partir des modèles de chaque tâche :

Définition 5. *L'ensemble des états valides de* Γ *est l'ensemble* $\Omega(\Gamma)$ *défini par :*

$$\Omega(\Gamma) = \{(t, x_1, ..., x_n) / \ \forall i \in [1, n],\ (t, x_i) \in Gr(\tau_i)\}$$

$T(\Gamma)$ *est l'ensemble des trajectoires de* $\Omega(\Gamma)$ *issues de l'origine dont les projections sur chacun des plans associés aux tâches est une trajectoire valide pour la tâche considérée.*

Nous montrons dans (Largeteau *et al.*, 2005) que Ω peut être obtenu par application des opérations de produit cartésien et d'intersection. Pour l'exemple de la figure 4, l'espace dans lequel est modélisée la concurrence est à 3 dimensions (2 tâches plus le temps). Le schéma de droite présente une vue éclatée du modèle. Le modèle de l'équité pour l'ensemble du système s'obtient à partir des modèles de l'équité pour chacune des tâches qui le composent :

Définition 6. *L'ensemble des états équitables du système* $\Gamma = (\tau_i)_{i\in[1,n]}$ *est :*

$$\Omega_E(\Gamma) = \{(t, y_1, ...y_n) \in \mathbb{N}^{n+1} \text{ tel que } \forall i \in [1, n], (t, y_i) \in Gr_E(\tau_i)\}$$

3.3. *Partage de ressources*

Nous considérons ici des ressources critiques, qui doivent s'utiliser en exclusion mutuelle. Une solution correcte doit respecter les sections critiques associées à leur

utilisation. Si pour un état du système, deux tâches sont toutes les deux à l'intérieur d'une section critique associée à la même ressource, cet état est invalide. Les états invalides peuvent être définis géométriquement. On note $nb_k(1)$ le temps CPU avant l'utilisation de la ressource R dans le code de τ_k, et $nb_k(2)$ le temps CPU depuis le début de la tâche jusqu'à la fin de la section critique ($nb_k(2) - nb_k(1)$ correspond à la durée de la section critique). Soit $(\tau_i)_{i\in I}$ l'ensemble des tâches partageant une ressource R. Les états de $(\tau_i)_{i\in I}$ liés à une utilisation commune de la ressource par ces tâches doivent être invalidés.

Définition 7. *La représentation géométrique des utilisations incorrectes de la ressource R est définie par :* $Gr(R) = \{(t, x_1, \ldots, x_n) \in \Omega(\Gamma) / \ \exists j_1, j_2 \in I,$
$\lfloor \frac{x_{j_1}}{P_{j_1}} \rfloor . P_{j_1} + nb_{j_1}(1) < x_{j_1} < \lfloor \frac{x_{j_1}}{P_{j_1}} \rfloor . P_{j_1} + nb_{j_1}(2)$ *and*
$\lfloor \frac{x_{j_2}}{P_{j_2}} \rfloor . P_{j_2} + nb_{j_2}(1) < x_{j_2} < \lfloor \frac{x_{j_2}}{P_{j_2}} \rfloor . P_{j_2} + nb_{j_2}(2)\}$.

L'ensemble des états respectant les contraintes de partage de ressources correspond à l'intersection de Ω avec le complémentaire de $Gr(R)$.

Définition 8. *Si le système $\Gamma = (\tau_i)_{i\in[1,n]}$ utilise une ressource R, le modèle géométrique $\Omega_R(\Gamma)$ de l'application est donné par :* $\Omega_R(\Gamma) = \Omega(\Gamma) \setminus Gr(R)$.
De même, le modèle géométrique de l'équité Ω_{ER} est : $\Omega_{ER}(\Gamma) = \Omega_E(\Gamma) \setminus Gr(R)$.
On note enfin $T_R(\Gamma)$ et $T_{ER}(\Gamma)$ l'ensemble des trajectoires de $\Omega_R(\Gamma)$ et de $\Omega_{ER}(\Gamma)$

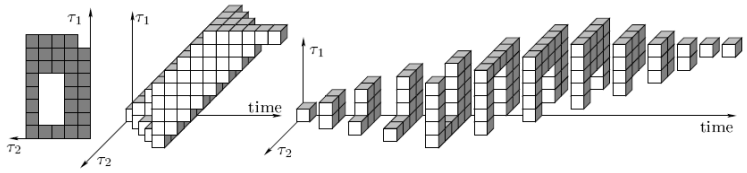

Figure 5. *Représentation géométrique d'une application avec partage de ressource*

La figure 5 présente le système de la figure 4 auquel a été ajouté un partage de ressource. Ce partage est modélisé par un trou dans le modèle initial. Le premier schéma présente la zone d'exclusion mutuelle dans l'espace (τ_1, τ_2), le second présente le modèle de l'application, et le troisième une vue éclatée du modèle.

3.4. *Prise en compte de l'architecture et décision d'ordonnançabilité*

Une fois l'objet construit, afin de prendre en compte le nombre de processeurs, nous devons étudier les propriétés de connexité. Sur une trajectoire, on regarde combien de tâches sont actives lorsque l'on passe d'un point au point suivant (voir la figure 6).

Définition 9. *S'il y a au plus k tâches actives entre deux états, on dit que la trajectoire est k-simultanée entre ces deux points.*

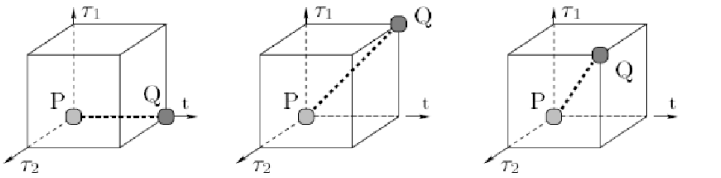

Figure 6. *k-simultanéité de deux tâches*

Une trajectoire est dite k-simultanée si elle est k-simultanée entre deux points consécutifs quelconques.
Un ensemble $\Omega_R(\Gamma)$ (resp. $\Omega_E(\Gamma)$) est dit k-simultané s'il existe au moins une trajectoire ψ de $T_R(\Gamma)$ (resp. $T_{ER}(\Gamma)$) k-simultanée.

Theorem 1. *Si l'ensemble $\Omega_R(\Gamma)$ (resp. $\Omega_{ER}(\Gamma)$) est k-simultané alors il existe un ordonnancement temporellement valide (resp. équitable) de l'application sur k processeurs.*

La complexité spatiale est par construction fonction du PPCM des périodes. Une fois le modèle construit, les différentes analyses possibles sont effectuées en O(n).

Remarque : pour un ensemble de tâches $\Gamma = (\tau_i)_{i\in[1,n]}$, l'ensemble $\Omega(\Gamma)$ est nécessairement n-simultané. Si l'ensemble $\Omega_{ER}(\Gamma)$ n'est pas n-simultané c'est que le partage de la ressource R pose problème.

4. Mesures d'équité

La partie précédente a montré comment décider de l'existence de séquences (ou trajectoires les deux termes étant équivalents) équitables compatibles avec l'architecture cible et respectant les règles d'accès aux ressources. Lorsqu'il n'en existe pas, il est possible qu'il existe tout de même des trajectoires valides. Autrement dit, on peut avoir $\Omega_R(\Gamma)$ m-simultané alors que $\Omega_{ER}(\Gamma)$ ne l'est pas. De plus, une trajectoire valide peut être plus ou moins équitable en ce sens qu'elle peut plus ou moins respecter les principes d'une exécution régulièrement répartie. Notre objectif est de poser les fondements d'un outil de sélection de comportements en fonction de critères quantitatifs d'équité. Nous proposons des indicateurs de qualité mesurant l'équité des séquences. Nous présentons des mesures adaptées à chacune des deux représentations (charge exécuté ou sous-tâches). Nous commençons par considérer une tâche seule, à laquelle nous associons une mesure d'invalidité, la mesure de validité est alors le complément à 1 de cette mesure. Si cette mesure est nulle, la tâche a un comportement équitable. Puis, nous étendons notre approche à l'application dans sa globalité.

Dans ce qui suit, P est l'hyper-période de l'application : $P = ppcm(P_1, P_2, \ldots, P_n)$, $n_i(P)$ est le nombre d'instances de τ_i dans l'intervalle $[0, P[$: $n_i(P) = \frac{P}{P_i}$, et $nbs_i(P)$ est le nombre de sous-tâches dont la fenêtre d'exécution est dans l'intervalle $[0, P[$: $nbs_i = n_i(P).C_i$.

Mesure fondée sur le nombre de points hors zone

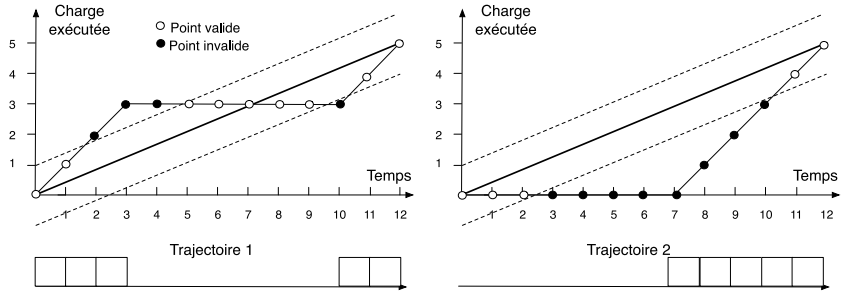

Figure 7. *Points valides de deux trajectoires pour la tâche* $< 5, 12 >$

Nous considérons la trajectoire de la tâche sur l'intervalle $[0, P[$, et nous définissons :

$$Mes_{inv_1}(\tau_i, S, P) = \frac{\#\, points(t, Charge(t)), 0 \leq t < P, \text{hors zone de validité}}{\#\text{ total de points}}.$$

La zone de validité est la zone contenue entre les deux droites W_- et W^+. Considérons la tâche $< 5, 12 >$. La figure 7 présente deux trajectoires de cette tâche sur l'intervalle $[0, 12[$. Chaque trajectoire comporte 12 points. Pour la première trajectoire, il y a 4 points hors de la zone de validité et il y en a 8 pour la seconde. Les mesures d'invalidité associée à ces trajectoires sont donc respectivement égales à $1/3$ et $2/3$. La trajectoire 1 est donc plus équitable que la trajectoire 2. Cette première mesure est simple à mettre en œuvre. Il suffit de considérer les points en dehors de l'objet géométrique qui représente les trajectoires équitables.

Mesure fondée sur les distances verticales

Considérons les trajectoires de la figure 8 pour la tâche $< 10, 30 >$. La trajectoire 2 est plus régulière que la trajectoire 1. Pourtant, il y a 22 points hors zone pour la trajectoire 1 contre 26 pour la trajectoire 2. La mesure précédente ne reflète pas cette impression de régularité. En fait, dans le premier cas, on s'écarte beaucoup de l'équité alors que dans le second, après un petit décalage au départ, la progression reprend un rythme régulier. Nous affinons donc la mesure précédente en comparant pour chaque point l'état d'avancement de la tâche avec son état d'avancement idéal à l'aide d'un calcul de distance entre le point (t, charge(t)) et le point (t, ideal(t)), appelée **distance verticale** à la droite $W(t)$ (voir figure 9). Elle est égale à $\mid charge(t) - W(t) \mid = \mid charge(t) - u_i.t \mid$. La mesure d'invalidité associée à la trajectoire 1 est alors égale à 78,3 et celle de la trajectoire 2 à 34,3. Cependant, cette mesure n'est pas comprise

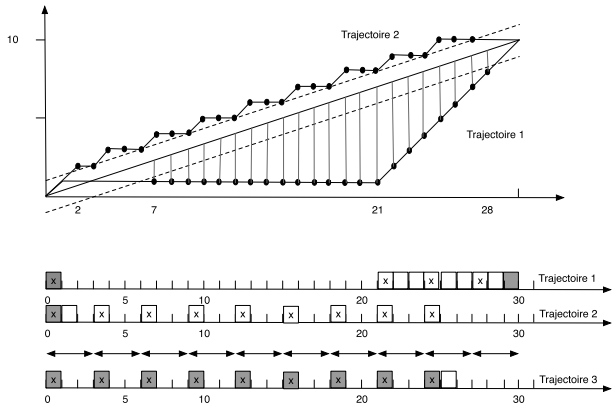

Figure 8. *Trajectoires de la tâche <10, 30>*

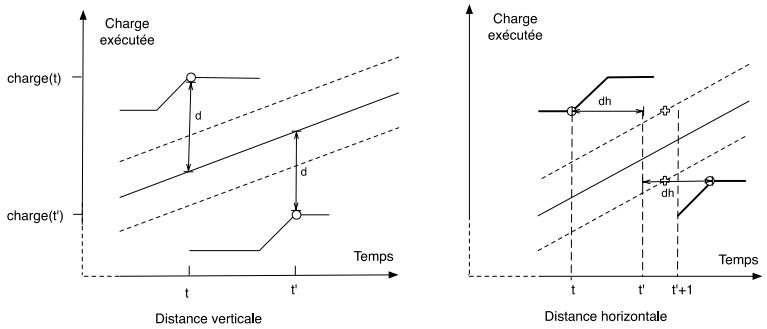

Figure 9. *Distance verticale d'un point hors zone à la droite W(t) - Distance horizontale*

entre 0 et 1. Pour la normaliser, il faut la diviser par la pire valeur. Celle-ci correspond aux cas où la tâche s'exécute tout au début de sa période ou tout à la fin. La mesure associée est ici égale à $98,3$. Donc les mesures finales d'invalidité sont égales à $0,797$ pour la trajectoire 1 et à $0,349$ pour la trajectoire 2. Formellement, nous avons :

$$Mes_{inv_2}(\tau_i, S, P) = \frac{\sum\limits_{points\ hors\ zone} \mid charge(t) - u_i.t \mid}{n_i(P).(\sum\limits_{\lceil \frac{1}{1-u_i} \rceil}^{C_i} (u_i.t - t) + \sum\limits_{C_i+1}^{\lfloor \frac{C_i-1}{u_i} \rfloor+1} (u_i.t - C_i))}.$$

Mesure fondée sur les fenêtres d'exécution

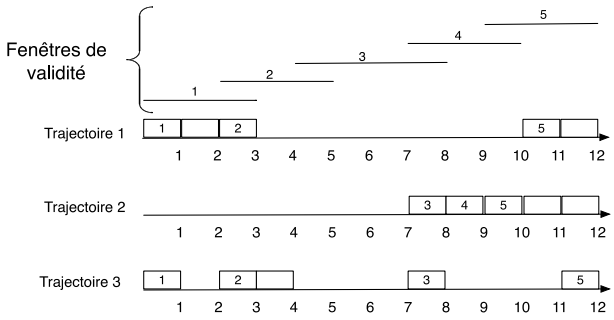

Figure 10. *Trajectoires pour la tâche <5, 12> et fenêtres de validité*

Nous adoptons maintenant le point de vue des fenêtres d'exécution et la décomposition des tâches en sous-tâches unitaires. Dans un ordonnancement équitable, on peut attacher de manière bijective à chaque sous-tâche la fenêtre dans laquelle elle s'exécute. Si l'ordonnancement n'est pas équitable, une telle construction n'est pas possible, on ne peut construire qu'une fonction injective partielle. Le taux d'invalidité est alors mesuré à l'aide du nombre de sous-tâches qui n'ont pas pu être associées à une fenêtre. Nous construisons donc une fonction injective croissante de l'ensemble des sous-tâches vers l'ensemble des fenêtres. Reprenons l'exemple de la figure 8. Nous associons les sous-tâches marquées d'une croix à une fenêtre. Ceci nous donne comme mesures d'invalidité $6/10$ pour la trajectoire 1 et $1/10$ pour la trajectoire 2, ce qui répond mieux à nos observations. Dans cet exemple, les fenêtres sont disjointes, le calcul est donc immédiat. Mais les fenêtres peuvent se chevaucher (de une unité de temps). Considérons les trajectoires proposée figure 10 pour la tâche <5, 12>. Sur chaque sous-tâche est indiqué, quand c'est possible, le numéro de la fenêtre associée à la sous-tâche. Nous obtenons comme mesure d'invalidité $2/5$ pour les trajectoires 1 et 2 et $1/5$ pour la trajectoire 3.

De manière formelle, la fonction (partielle) qui associe une fenêtre à une sous-tâche est définie de la manière suivante. Soit une tâche $\tau_i < C_i, P_i >$. On considère les fenêtres d'exécution sur un intervalle de temps $[0, P]$. La fonction f est définie sur $\{\tau_i^j / 1 \leq j \leq nbs_i(P)\}$ de manière inductive par :

– $f(\tau_{i,1}) = I_i^{j_0}$ où j_0 est le plus petit indice tel que τ_i^1 s'exécute dans $I_i^{j_0}$.

– On suppose f définie pour $\tau_i^1, \ldots, \tau_i^{j-1}$.

– Si τ_i^j s'exécute dans une unique fenêtre I_i^k alors

- s' il existe $r < j$ tel que $f(\tau_i^r) = I_i^k$ alors $f(\tau_i^j) = \perp$,

- sinon $f(\tau_i^j) = I_i^k$.

– Si τ_i^j s'exécute dans $I_i^k \cap I_i^{k+1}$, alors

- s'il existe $r < j$ tel que $f(\tau_i^r) = I_i^k$, alors $f(\tau_i^j) = I_i^{k+1}$

- sinon $f(\tau_i^j) = I_i^k$.

La mesure d'invalidité est alors définie par [2] : $Mes_{inv_3}(\tau_i, P) = \frac{|\{\tau_{i,j} / f(\tau_i^j = \perp\}|}{nbs_i(P)}$.

Mesure fondée sur les distances horizontales

Reprenons la tâche $< 10, 30 >$, et considérons la troisième trajectoire. Elle possède la même mesure d'invalidité que la trajectoire 2, selon la mesure précédente. Cependant, pour cette trajectoire, toutes les sous-tâches sauf une s'exécutent dans leur fenêtre de validité. Il semblerait donc normal que la mesure puisse les départager. Pour cela, nous raffinons la mesure précédente en déterminant pour chaque tâche la distance à sa fenêtre d'exécution. Cette distance est définie par : $dh(\tau_i^j, S) = Max\{0, r_i^j - b_i^j, f_i^j - d_i^j\}$ où b_i^j est la date de début de la sous-tâche et f_i^j sa date de terminaison ($f_i^j = b_i^j + 1$). Cette distance est appelée la **distance géométrique horizontale**, illustrée figure 9. On fait donc la somme des distances horizontales pour toutes les sous-tâches qui s'exécutent en dehors de leur fenêtre. Il faut enfin, comme dans le cas 2, normaliser cette somme. La mesure d'invalidité est alors définie par $Mes_{inv_4}(\tau_i, P) = \frac{\sum_{j=1}^{nbs_i(P)} dh(\tau_i^j, S)}{\sum_{j=1}^{nbs_i(P)} \lfloor \frac{j-1}{u_i} \rfloor - (j-1)}$.

Reprenons l'exemple de la figure 10. Pour la trajectoire 1, la distance obtenue est égale à $\frac{0+1+2+1+0}{12} = \frac{1}{3}$, pour la trajectoire 2, elle est égale à $\frac{5+4+2+1+0}{12} = 1$ et pour la trajectoire 3, on trouve $\frac{0+0+1+0+0}{12} = \frac{1}{12}$. Là encore, le modèle géométrique pourra servir de support aux calculs, puisqu'il s'agira de déterminer pour les segments hors zone la distance d'une de leurs extrémités à l'objet géométrique représentant la zone de validité. Le tableau 1 résume les valeurs de ces quatre mesures pour les trois séquences de la figure 8, avec $P = 30$.

	Mes_{inv_1}	Mes_{inv_2}	Mes_{inv_3}	Mes_{inv_4}
seq_1	0.73	0.797	0.6	0.8
seq_2	0.87	0,349	0.1	0.29
seq_3	0,07	0.139	0.1	0.02

Tableau 1. *Mesures pour les trajectoires (ou séquences) de la figure 8*

Mesures d'invalidité d'une séquence

Nous disposons de mesures permettant d'analyser de manière individuelle le degré d'équité de chacune des tâches au sein de la séquence. Afin de qualifier la séquence dans sa globalité, nous devons ensuite synthétiser les résultats obtenus. Pour cela, nous calculons la moyenne des taux individuels pondérés par leur facteur d'utilisation. Ces

2. $|A|$ désigne le cardinal de l'ensemble A.

facteurs de pondération permettent de tenir compte de la place occupée par la tâche dans l'application. Donc $Taux_{inv_k}(\tau) = \frac{\sum_{i=1}^{n} u_i . Mes_{inv_k}(\tau_i, P)}{\sum_{i=1}^{n} u_i}$.

5. Conclusion

Nous avons présenté une modélisation à base de géométrie discrète des ordonnancements P-équitables pour des applications constituées de tâches à départs simultanés et à échéances sur requêtes, partageant des ressources critiques. La construction proposée permet de déterminer s'il existe des séquences P-équitables compatibles avec les règles de gestion des ressources critiques. Lorsqu'il n'en existe pas, le modèle permet de représenter l'ensemble des séquences valides.Nous avons alors proposé des critères de sélection de comportements le plus équitable possible. Cette sélection s'appuie sur des mesures d'invalidité qui quantifient l'écart par rapport à des exécutions équitables. Nous avons proposé deux types de techniques de mesure qui s'adpatent à chacune des deux visions des ordonnancements équitables (courbe de charge exécutée versus sous-tâches). La mesure mes_{inv_1} est calculée à l'aide d'un dénombrement simple à partir du modèle géométrique. Il est ensuite décliné en une seconde version qui affine la vision binaire de la première mesure : on tient compte pour les points ne respectant pas l'équité de l'écart avec l'exécution idéale, au moyen de calculs de distances. La troisième mesure s'appuie sur la décomposition en sous-tâches de l'application, et sur l'association injective des sous-tâches à une fenêtre d'exécution. Si l'exécution n'est pas suffisamment répartie, certaines sous-tâches ne pourront pas être associées à une fenêtre. Leur nombre caractérise alors la non équité de la séquence. Enfin, la dernière mesure tient compte de la distance de chaque sous-tâche s'exécutant hors de sa fenêtre à sa fenêtre. Cette mesure s'appuie sur des distances qui peuvent être déterminées à l'aide du modèle géométrique. Les simulations réalisées tendent à montrer que la mesure mes_{inv_4} donne les résultats les plus proches de l'idée intuitive de l'équité. Ce travail constitue la première étape de la création d'un atelier de détermination de séquences presqu'équitables et d'aide à la conception. La prochaine étape consistera en l'intégration de ces mesures dans l'outil GemSMARTS (Largeteau *et al.*, 2004), puis en la réalisation d'un banc de tests permettant de les comparer et de déterminer leurs performances en fonction des caractéristiques des applications.

Une autre application intéressante de ces résultats sera de disposer d'outils de sélection de séquences où les temps creux seront disposés le plus équitablement possible. En effet, une répartition équitable des temps creux de l'application permet une prise en compte efficace de tâches apériodiques à constraintes strictes (voir (Choquet-Geniet *et al.*, 2009)).

6. Bibliographie

Anderson J., Holman P., Srinivasan A., « Fair scheduling of real time tasks on multiprocessors », *Handbook of scheduling : Algorithms, Models and Performance analysis*p. 31.1-31.21, 2004.

Andersson B., Jonsson J., « Fixed-priority preemptive multiprocessor scheduling : to partition or not to partition », *Proceedings of the conference on Real-Time Computing Systems and Applications*, p. 337-346, December, 2000.

Baruah S., Cohen N., Plaxton C., Varvel D., « Proportionate progress : a notion of fairness in resource allocation », *Algorithmica*, vol. 15, p. 600-625, 1996.

Chauvière B., Geniet D., « Une approche Markovienne pour l'étude de systèmes temps-réel à contraintes strictes », *Techniques et Sciences Informatiques*, vol. 26, n° 10, p. 1269-1303, October, 2007.

Choquet-Geniet A., Fotsing C., Malo S., Scheduling of real-time applications with variable utilization factor using a PFair based aperiodic server, Technical report, LISI, ENSMA and University of Poitiers, March, 2009.

Geniet D., Largeteau G., « Validation Temporelle de Systèmes de Tâches Temps-Réel Strictes à Durées Variables à l'Aide de Langages », *Proceedings of MSR'2001*, 2001.

Grolleau E., Choquet-Geniet A., « Off line computation of real time schedules by means of Petri nets », *Journal of Discrete Event Dynamic Systems*, vol. 12, p. 311-333, 2002.

Hong K., Leung J., « On-Line Scheduling of Real-Time Tasks », *IEEE transactionson Computers*, vol. 41, n° 10, p. 1326-1331, 1992.

Largeteau G., Chauvière B., Geniet D., « Une approche géométrique pour la validation d'applications temps réel à contraintes strictes. », *Real Time Systems'2004*, p. 15-33, 2004.

Largeteau G., Geniet D., Andres E., « Discrete Geometry Applied in Hard Real-Time Systems Validation. », *DGCI 2005 12th International Conference, Poitiers*, vol. 3429 of *LNCS*, Springer Verlag, p. 23-33, 2005.

Largeteau G., Geniet D., Dubernard J., « Validation of Distributed Periodic Real-Time Systems Using CAN Protocol with Finite Automata », *Proc. of* 5^{th} *S.C.I.*, I.I.I.S., 2001.

Largeteau-Skapin G., Geniet A., Ouattara A., Using discrete geometry to model PFair scheduling algorithm for real-time systems applications, Technical report, Xlim-SIC and LISI, university of Poitiers and ENSMA, 2009.

Largeteau-Skapin G., Geniet D., « Quantification du taux d'invalidité d'applications temps réel à contraintes strictes », *Technique et Science Informatiques*, vol. 27, n° 5, p. 589-625, Mai, 2008.

Leung J., Whitehead J., « On the Complexity of Fixed-Priority Scheduling of Periodic Real-Time Tasks », *Performance Evaluation*p. 237-250, 1982.

ANNEXE POUR LE SERVICE FABRICATION

A FOURNIR PAR LES AUTEURS AVEC UN EXEMPLAIRE PAPIER
DE LEUR ARTICLE ET LE COPYRIGHT SIGNE PAR COURRIER
LE FICHIER PDF CORRESPONDANT SERA ENVOYE PAR E-MAIL

1. ARTICLE POUR LA REVUE :
   JESA – 43/2009. MSR 2009

2. AUTEURS :
   Annie Choquet-Geniet[*] *— Gaëlle Largeteau-Skapin*[**]
   Abdoulaye Ouattara[***]

3. TITRE DE L'ARTICLE :
   Mesures de l'équité d'une application temps-réel à l'aide de géométrie discrète

4. TITRE ABRÉGÉ POUR LE HAUT DE PAGE MOINS DE 40 SIGNES :
   Mesures d'équité

5. DATE DE CETTE VERSION :
   2 octobre 2009

6. COORDONNÉES DES AUTEURS :

– adresse postale :

[*] LISI - ENSMA - Téléport 2 - 1 avenue Clément Ader - BP 40109
86961 Futuroscope Chasseneuil Cedex, France

[**] Laboratoire SIC, Université de Poitiers, BP 30179
86961 Futuroscope Chasseneuil Cedex, France

[***] Université Polytechnique de Bobo Dioulasso - Burkina Fasso

annie.geniet@univ-poitiers.fr, glargeteau@sic.univ-poitiers.fr

– téléphone : 00 00 00 00

– télécopie : 00 00 00 00 00

– e-mail : revues@lavoisier.fr

7. LOGICIEL UTILISÉ POUR LA PRÉPARATION DE CET ARTICLE :
   LaTeX, avec le fichier de style `article-hermes.cls`,
   version 1.23 du 17/11/2005.

8. FORMULAIRE DE COPYRIGHT :
   Retourner le formulaire de copyright signé par les auteurs, téléchargé sur :
   `http://www.revuesonline.com`

SERVICE ÉDITORIAL – HERMES-LAVOISIER
14 rue de Provigny, F-94236 Cachan cedex
Tél. : 01-47-40-67-67
E-mail : revues@lavoisier.fr
Serveur web : http://www.revuesonline.com