Master 2 pro

Programmation temps-réel
Cours 1 et 2
Introduction et ordonnancement

Isabelle PUAUT / Rémi COZOT
Université de Rennes I

Applications temps-réel embarquées

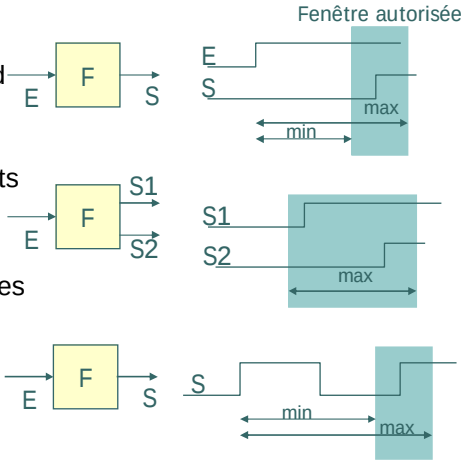
- Systèmes en interaction avec l'environnement physique
- Contraintes de temps à satisfaire
  - La date de livraison d'un résultat est un critère de correction autant que la valeur du résultat elle-même
- Systèmes dédiés (non généralistes)
  - Matériel (processeur, DSP, FPGA, ASICs, capteurs, actionneurs)
  - Logiciel
- Contraintes industrielles
  - Systèmes embarqués : contraintes de poids, taille, consommation énergétiques, coût, surface
  - Time-to-market
  - Sûreté de fonctionnement

Notion de contraintes de temps

- Définition
  - Limite quantifiée (en rapport avec le temps réel) sur le temps séparant deux événements
  - Ex : échéance de terminaison au plus tard (deadline)
- Source des contraintes de temps
  - Délai de réaction du système avant dysfonctionnement plus ou moins grave
  - Stabilité du processus physique contrôlé (ex: pendule inversé)

Exemples de contraintes de temps

- Échéance de terminaison au plus tôt ou au plus tard (deadline)
- Cohérence entre instants de production des résultats
  - Ex : synchronisation son–image
- Cadence de production des sorties
  - Ex : régularité de présentation des images

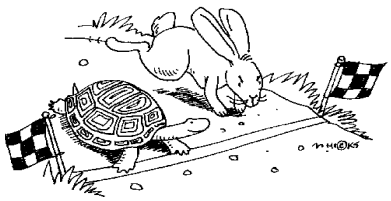


Classes de systèmes temps-réel

- Temps-réel strict/dur (hard real-time) : le non respect d'une contrainte de temps a des conséquences graves (humaines, économiques, écologiques) : besoin de garanties
- Temps-réel souple/mou (soft real-time) : on peut tolérer le non respect occasionnel d'une contrainte de temps (garanties probabilistes)

Mauvaises interprétations de la notion de temps-réel

- « Real-time is not real-fast » ou « rien ne sert de courir, il faut partir à point »



- Aller vite n'est pas l'objectif recherché

Systèmes temps-réel

Domaines d'application (1/2)

- Electronique grand public
  - Caméras numériques, appareils photo numérique
  - Multimédia, téléphonie
    - décodeurs vidéo
    - téléphones portable
    - PDA
    - consoles de jeu

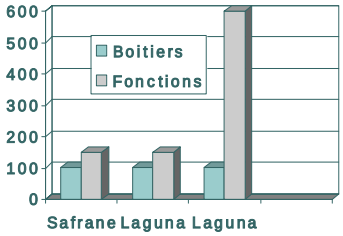
Systèmes temps-réel

Domaines d'application (2/2)

- Automobile
  - Systèmes anti-blocage de freins, contrôle moteur, informatique de confort
- Contrôle de procédés industriels
- Avionique et spatial
- Périphériques informatique
  - FAX
  - imprimantes

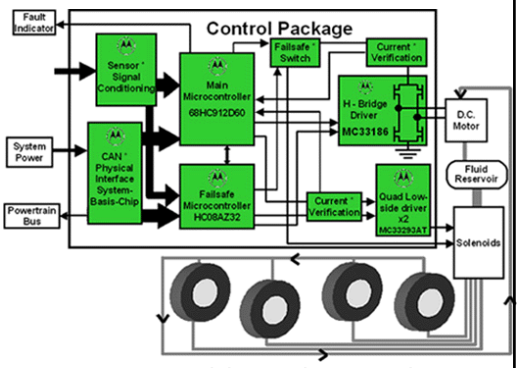
Systèmes temps-réel
Exemple : automobile

- Evolutions des fonctions logicielles dans l'automobile



(C. Balle, Journée Illiatech, octobre 2001)

- Système antiblocage (ABS)



(site Web motorola : www.motorola.com)

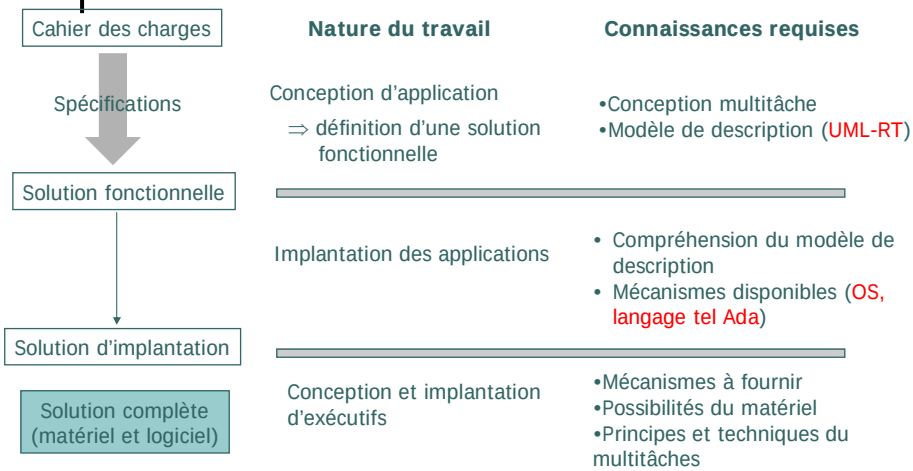
Contraintes pour la réalisation d'applications temps-réel (1/2)

- Contraintes de temps
  - Choix de l'ordre d'exécution des fonctions important
    - Ordonnancement des calculs
  - Déterminisme d'exécution (predictability)
    - On doit connaître pour tous les calculs effectués (application, OS) leur temps de calcul de manière sûre (non sous-estimée)
  - Validation des contraintes de temps à partir de la charge de travail
    - Tests pas toujours suffisants (exhaustivité)
    - Modèle du système et de la charge : analyse d'ordonnançabilité

Contraintes pour la réalisation d'applications temps-réel (2/2)

- Sûreté de fonctionnement : qualité des développements (lisibilité, maintenabilité)
  - Environnement de conception (UML-RT)
  - Langage de programmation adapté : langage de haut niveau
    - Assembleur rapide, mais non portable, difficile de vérifier les contraintes de temps (pas assez structuré)
    - Langages de plus haut niveau (ex: Ada)

Niveaux d'approche pour la conception d'applications (temps-réel)



Plan du cours / TD

- Ordonnancement des calculs et vérification d'ordonnançabilité (bases « théoriques »)
- Programmation temps-réel
  - Programmation
    - Ada95 pour le temps-réel
    - Environnements d'exécution temps-réel
  - Modélisation et spécifications
    - UML et ses extensions pour le temps-réel
- Remarques
  - Programmation temps-réel asynchrone seulement
  - Partie ordonnancement indépendante d'un environnement d'exécution particulier

Plan des TPs

- Créneaux 1 et 2
  - TPs langage Ada (Isabelle)
    - Illustration du cours
- Créneaux 3 à 5
  - Mini-projet d'application temps-réel en Ada
    - Moteur 3D Ada
      - Partie 3D fournie
      - Mise en place de comportements
        - Fréquences différentes
      - Gestion de l'interface homme – machine

Contrôle des connaissances

- Objectif
  - Valider une connaissance des concepts présentés
- Contrôle continu (pas d'examen terminal)
  - Assiduité
  - Test basique des connaissances (type QCM)
  - TPs
  - Assidu + compris => note >10

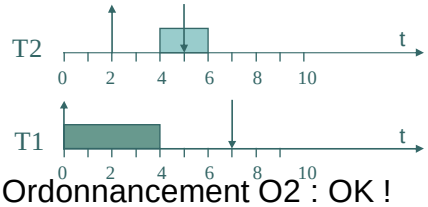
Ordonnancement temps-réel

Introduction (1/2)

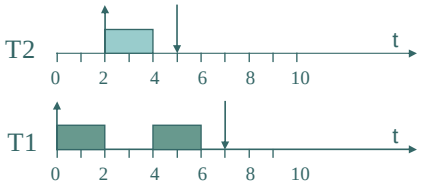
- Problème
  - Calculs à exécuter + contraintes de temps = dans quel ordre exécuter ?
- Définition
  - Ensemble des règles définissant l'ordre d'exécution des calculs sur le processeur
- Pourquoi ordonnancer ?
  - Parce que ça a un impact sur le respect des contraintes de temps
  - Exemple
    - Tâche T1 : arrivée en 0, durée 4, échéance 7
    - Tâche T2 : arrivée en 2, durée 2, échéance 5
    - Ordonnancement O1: premier arrivé, premier servi, on n'interrompt jamais une tâche
    - Ordonnancement O2: priorité, T2 plus prioritaire que

Introduction (2/2)

- Ordonnancement O1 : T2 rate son échéance



- Ordonnancement O2 : OK !



Classification des ordonnancements

- Critères de classification
  - Instant ou l'ordre d'exécution est décidé : hors-ligne / en-ligne
  - Possibilité d'interruption d'une tâche par une autre : préemptif / non préemptif
  - Optimalité : quel est le « meilleur » ordonnancement

Classification (1/5)

Ordonnancement « hors-ligne » / « en-ligne »

- Hors-ligne
  - La séquence d'ordonnancement est pré-calculée avant l'exécution effective (on dit aussi « time-driven » scheduling)
  - A l'exécution, l'ordonnanceur est un simple séquenceur (« cyclic scheduler »)

Classification (2/5)
Ordonnancement « hors-ligne » / « en-ligne »

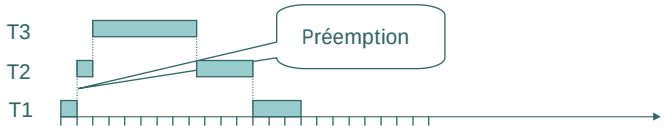
- Ordonnancements « en-ligne »
  - Les décisions d'ordonnancement sont prise au cours de l'exécution
  - A l'exécution, l'ordonnanceur implante un algorithme d'ordonnancement permettant de savoir à tout instant quel tâche exécuter : besoin d'un exécutif multitâche
  - Généralement, ordonnancements conduits par la priorité

Classification (3/5)
Ordonnancement non préemptif / préeemptif

- Non préemptif
  - On n'interrompt jamais l'exécution d'une tâche en cours au profit d'une autre tâche
- Préemptif
  - La tâche en cours peut perdre involontairement le processeur au profit d'une autre tâche (jugée plus urgente)
- Remarques : orthogonal par rapport à la classification en-ligne / hors-ligne

Classification (4/5)
Ordonnancement non préemptif / préeemptif

- Exemple d'ordonnancement préemptif : conduit par la priorité
  - Algorithme d'ordonnancement : sélectionner la tâche la plus prioritaire
  - prio(T3) > prio(T2) > prio(T1) (ici, priorités fixes)
  - T1, T2 et T3 arrivent respectivement aux dates 1, 2, et 3

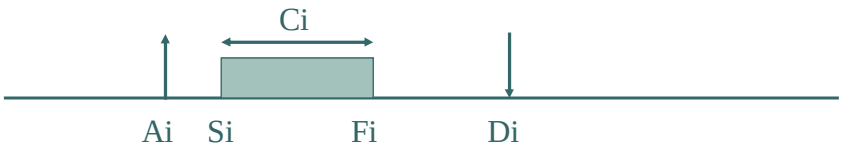


Classification (5/5)
Optimalité

- Optimal vs heuristique
  - Optimal : si ne trouve pas de solution au problème d'ordonnancement posé, alors aucun autre algorithme de la même classe ne peut en trouver
  - Heuristique : pas de garantie d'optimalité
- Remarques
  - Cette notion d'optimalité ne permet pas de déterminer la meilleure classe d'algorithme pour un problème donné

Notations (1/2)

- Arrivée Ai
- Temps de calcul maximum sans préemption Ci
- Echéance (deadline) : relative ou absolue Di
- Date de début (start time) Si
- Date de fin (finish time) Fi
- Retard (lateness) Li = (Fi - Di)



Notations (2/2)

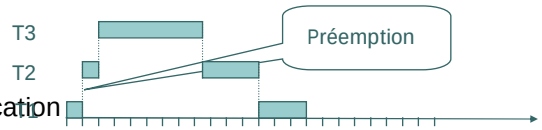
- Arrivées des tâches
  - Périodiques : arrivée à intervalles réguliers (Pi)
    - Date d'activation initiale, offset Oi
    - Si pour tout i,j Oi=Oj, tâches synchrones
    - Si Di = Pi, tâche à échéance sur requête
  - Sporadiques : on connaît une borne minimale sur l'intervalle entre deux arrivées
  - Apériodiques : tout le reste
- Remarque : exécution de tâches périodiques cyclique sur l'intervalle
  - [0,PPCM(Pi)] pour les tâches synchrones
  - [min(Oi), max(Oi,Oj+Dj) + 2 * PPCM(Pi)] dans le cas contraire

Quelques ordonnancements temps-réel (1/3)

- Selon un choix hors-ligne de l'ordre d'exécution
  - Génération de l'ordre pour le respect des contraintes de temps (associe ordonnancement et vérification d'ordonnançabilité)
  - Suppose la connaissance de l'ensemble des tâches
  - Exploration exhaustive : problème NP-complet
    - Méthodes d'élagage (pruning) des ordres possibles [Bratley, 1971]
    - Recherche orientées ou heuristiques [Spring, 1987]

Quelques ordonnancements temps-réel (2/3)

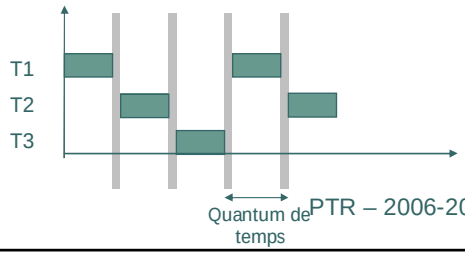
- Selon un choix en-ligne de l'ordre d'exécution : Conduits par la priorité
  - Sélection de la tâche la plus prioritaire
  - En général, préemptif
  - Exemple: prio(T3) > prio(T2) > prio(T1)



  - Classification
    - Priorités fixes (statiques) : indépendants du temps
    - priorités dynamiques : évoluent avec le temps
  - Très utilisé dans le monde du temps-réel

Quelques ordonnancements temps-réel (3/3)

- Selon un choix en-ligne de l'ordre d'exécution
  - Temps partagé avec politique du tourniquet (round-robin)
    - Algorithme généraliste
    - Souvent utilisé en complément du mécanisme de priorité pour les tâches de priorités égales (équité entre ces tâches)

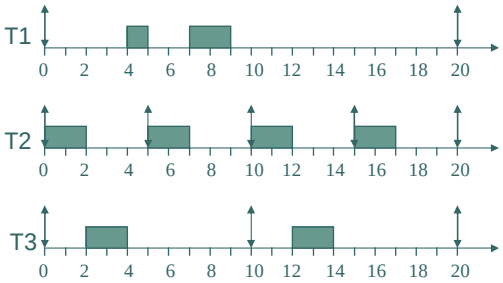


Ordonnancement préemptif à priorité fixe
Rate Monotonic (1/2)

- Pour tâches périodiques seulement
- Définition [Liu & Layland, 1973]
  - La priorité d'une tâche est inversement proportionnelle à sa période d'activation
- Propriété
  - Optimal dans la classe des algorithmes à priorités fixes pour des tâches périodiques indépendantes à échéance sur requête (Di=Pi).

Ordonnancement préemptif à priorité fixe
Rate Monotonic (2/2)

Tâche	Pi	Ci	Di
T1	20	3	20
T2	5	2	5
T3	10	2	10



Prio(T2) > Prio(T3) > Prio(T1)
Exécution cyclique
(PPCM des périodes)

Ordonnancement préemptif à priorité fixe
Deadline Monotonic

- Pour tâches périodiques
- Définition [Leung & Whitehead, 1985]
  - La priorité d'une tâche est inversement proportionnelle à son échéance relative (conflits résolus arbitrairement)
- Propriété
  - Optimal dans la classe des algorithmes à priorités fixes pour des tâches périodiques indépendantes à échéance ≤ période

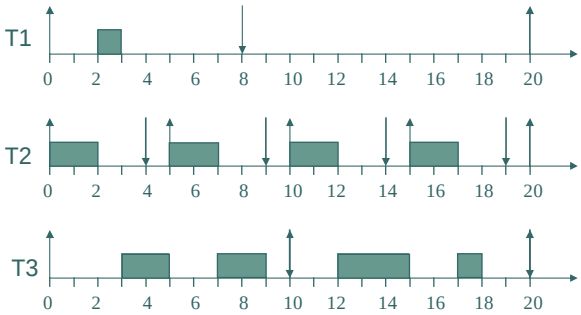
Ordonnancement préemptif à priorité dynamique
Earliest Deadline First (1/3)

- Earliest Deadline First (EDF)
- Applicable pour tâches périodiques et non périodiques
- Définition [Liu & Layland, 1973]
  - A un instant donné, la tâche la plus prioritaire (parmi les tâches prêtes) est celle dont l'échéance absolue est la plus proche
  - Préemptif
- Propriété
  - Optimal dans la classe des algorithmes préemptifs pour des configurations de tâches périodiques indépendantes avec échéance ≤ période

Ordonnancement préemptif à priorité dynamique
Earliest Deadline First (2/3)

Tâche	Pi	Ci	Di
T1	20	1	8
T2	5	2	4
T3	10	4	10

(2 préemptions en 5 et 15)



Vérification des contraintes de temps (analyse d'ordonnançabilité)

Vérification des contraintes temporelles
Introduction (1/3)

- Rôle
  - Formules mathématiques ou algorithmes permettant de vérifier (prouver) que les tâches respecteront leurs contraintes de temps (ex: échéances)
- Classification
  - Vérification hors-ligne (avant exécution)
    - Critères analytiques calculables (CN, CS, CNS)
    - Cibles = systèmes temps-réel strict
  - Vérification en-ligne (pendant exécution)
    - Tests d'acceptation en-ligne pour savoir si on accepte de nouvelles tâches
    - Risque de rejet de tâches → cibles = systèmes temps-réel souple

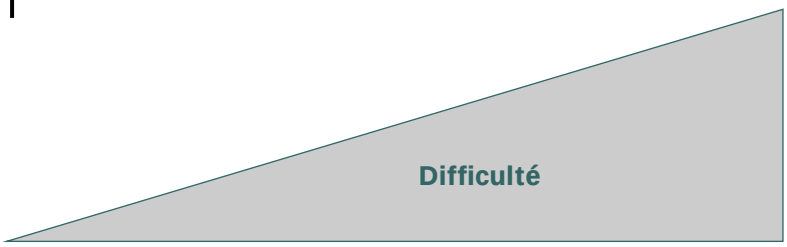
Vérification des contraintes temporelles
Introduction (2/3)

- Données d'entrée des méthodes de verification : modèle du système
  - Connaissance des informations sur les tâches (modèle de tâches)
    - Arrivée des tâches: périodique, sporadique, apériodique (dates d'arrivées)
    - Synchronisations : précédences, exclusions entre tâches
    - Temps d'exécution au pire-cas (WCET)
  - Ces données sont connues statiquement pour les analyses d'ordonnançabilité hors-ligne
- Sorties
  - Verdict sur le respect des contraintes de temps

Vérification des contraintes temporelles
Introduction (3/3)

- Conditions nécessaires et/ou suffisantes
  - Condition nécessaire C
    - Il faut que C soit vérifiée pour que les contraintes de temps soient respectées (système faisable)
    - C non vérifiée ⇨ système non faisable
    - C vérifiée ⇨ système peut être faisable (ou pas)
  - Condition suffisante
    - Il suffit que C soit vérifiée pour que le système soit faisable
    - C vérifiée ⇨ système est faisable (à coup sûr)
    - C non vérifiée ⇨ système peut être faisable (ou pas)
  - Condition nécessaire et suffisante
    - C vérifiée équivalent à montrer la faisabilité du système

Vérification des contraintes temporelles
Complexité de la vérification (1/2)



Monoprocesseur
Tâches périodiques, synchrones, indépendantes

Multiprocesseur

Système distribué, tâches périodiques et sporadiques, asynchrones, dépendantes

De manière générale, le problème d'ordonnançabilité est NP-difficile

Vérification des contraintes temporelles
Complexité de la vérification (2/2)

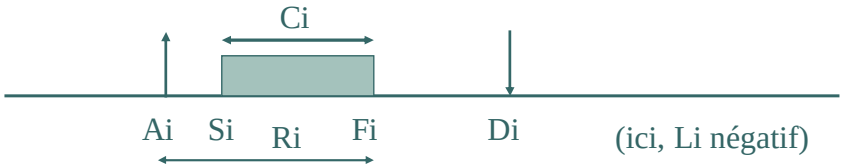
Monoprocesseur			Multiprocesseur		
Indep.	NoPr |Lmax Pr | Lmax NoPr,Ri | Lmax	Poly Poly NP-hard	**NoPr, 2 proc**	NoPr,prec,Ci=1 | Lmax NoPr,prec, ress NoPr, Ci <> 1 | Lmax NoPr, Ci=1, 1 ress	Poly NP-hard NP-hard NP-hard
Préc.	NoPr,prec | Lmax NoPr, prec, Ri | Lmax Pr, prec, Ri | Lmax	Poly NP-hard Poly	**NoPr, N Proc**	NoPr, prec, Ci=1	NP-hard
Ress.	Per + sémas Pr,Ri,Ci=1 | Lmax NoPr, prec, Ri, Ci=1 | Lmax	NP-hard Poly Poly	**Pr, N proc**	Pr | mbmiss	NP-hard

Vérification de contraintes temporelles
Plan

- Faisabilité pour tâches apériodiques à dates d'arrivée connues, non préemptif
- Faisabilité pour tâches périodiques et ordonnancements préemptifs à priorités fixes
  - Rappel des notations et hypothèses
  - Simulation
  - Condition nécessaire d'ordonnançabilité
  - Vérification pour ordonnancements à priorités fixes
    - Condition pour Rate Monotonic (CS) et Deadline Monotonic (CS)
    - Condition générale pour algorithmes à priorités fixes : Response Time Analysis (CNS)
- Faisabilité pour tâches périodiques et ordonnancements préemptifs à priorités

Analyse d'ordonnançabilité
Rappel des notations (1/2)

- Arrivée Ai
- Temps de calcul maximum sans préemption Ci
- Echéance (deadline) : relative ou absolue Di
- Date de début (start time) et fin (finish time) Si,Fi
- Temps de réponse Ri = Fi - Ai
- Retard (lateness) Li = (Fi - Di)



Analyse d'ordonnançabilité
Rappel des notations (2/2)

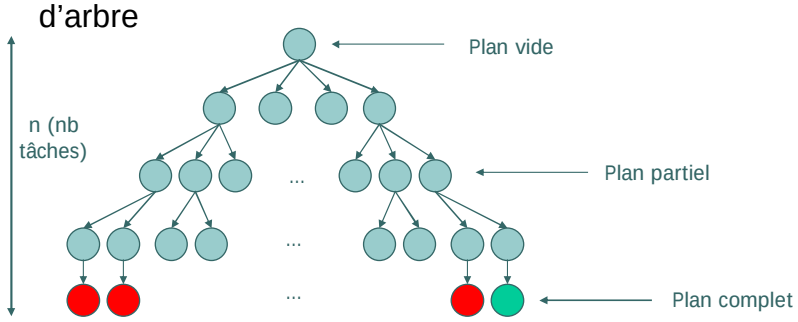
- Arrivées des tâches
  - Périodiques : arrivée à intervalles réguliers (Pi)
    - Date d'activation initiale, offset Oi
    - Si pour tout i,j Oi=Oj, tâches synchrones
    - Si Di = Pi, tâche à échéance sur requête
- Instant critique : date pour laquelle une arrivée de tâches produira son pire temps de réponse

Faisabilité pour tâches apériodiques en non préemptif

- Problème à résoudre
  - Entrées : ensemble de tâches connu. Pour chaque Ti, on connaît :
    - Ci : temps d'exécution
    - Ai : date absolue d'arrivée
    - Di : échéance absolue
  - Sorties
    - Verdict : faisable ou infaisable
    - Calendrier d'exécution (Si pour chaque tâche)
- Complexité du problème : NP-complet

Faisabilité pour tâches apériodiques en non préemptif

- Le problème exprimé sous forme d'une exploration d'arbre



Faisabilité pour tâches apériodiques en non préemptif

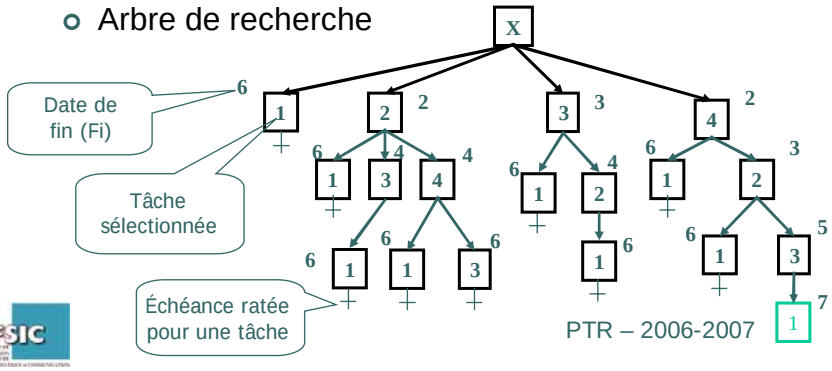
- Solutions
  - Solution optimale : exploration exhaustive de toutes les séquences d'ordonnancement possibles O(n n!) : trop complexe dans le cas général
  - Solution suboptimales : réduction de l'arbre de recherche
    - Elagage (pruning)
      - Exemple : algorithme de Bratley
    - Heuristiques pour guider le parcours de l'arbre
- Complexité de la recherche → orienté vers la vérification hors-ligne

Faisabilité pour tâches apériodiques en non préemptif
Algorithme de Bratley (1971) (1/3)

- Principe
  - A un nœud de l'arbre est associé un plan provisoire
  - Exploration à partir d'un nœud en considérant toutes les tâches encore possibles
  - On élague une branche (pruning) dans deux cas :
    - on a trouvé un plan dans lequel les échéances sont respectées
    - l'ajout de tout noeud sur le chemin courant entraîne un dépassement d'échéance

Faisabilité pour tâches apériodiques en non préemptif Algorithme de Bratley (1971) (2/3)

- Exemple
  - A1 = 4 A2 = 1 A3 = 1 A4 = 0
  - C1 = 2 C2 = 1 C3 = 2 C4 = 2
  - D1 = 7 D2 = 5 D3 = 6 D4 = 4
- Arbre de recherche



Faisabilité pour tâches apériodiques en non préemptif Algorithme de Bratley (1971) (3/3)

- Propriétés de l'algorithme
  - Non optimal (s'arrête au premier ordonnancement faisable, pas nécessairement le meilleur)
  - En moyenne, technique d'élagage très efficace, mais complexité au pire est toujours O(n n!)
  - Utilisable pour de la vérification hors-ligne seulement

Analyse d'ordonnançabilité pour tâches périodiques Hypothèses

- Hypothèses explicites
  1. Tâches périodiques de période Pi
  2. Temps d'exécution pire-cas Ci constant
  3. Echeance sur requête (Di = Pi)
  4. Tâches indépendantes
  5. Tâches synchrones (Oi=0)
- Hypothèses implicites
  5. Pas de suspensions des tâches
  6. Une tâche peut être lancée dès son arrivée (Ai) : pas de gigue au démarrage
  7. Surcoûts du système d'exploitation (démarrage tâche, préemption) nuls

Analyse d'ordonnançabilité pour tâches périodiques
Vérification par simulation

- Périodicité des tâches → ordonnancement cyclique (pas la peine de simuler à l'infini)
- Durée de la période d'étude (ou pseudo-période)
  - Tâches synchrones : [0,PPCM(Pi)]
  - Tâches échelonnées (au moins un offset Oi non nul)
    - [min(Oi), max(Oi,Oj+Dj) + 2 * PPCM(Pi)]
- Evaluation
  - Valide quel que soit l'algorithme d'ordonnancement
  - Période d'étude peut être très longue
  - Adapté aux ordonnancements hors-ligne

Analyse d'ordonnançabilité pour tâches périodiques
Condition nécessaire d'ordonnançabilité

- Notion de charge processeur

$$U = \sum_{i=1}^{n} \frac{Ci}{Pi}$$

- Condition nécessaire d'ordonnançabilité : $U \leq 1$
  - Cette condition n'est pas une condition suffisante (pas pour tous les ordonnancements)

Analyse d'ordonnançabilité pour tâches périodiques
Faisabilité pour Rate Monotonic (1/2)

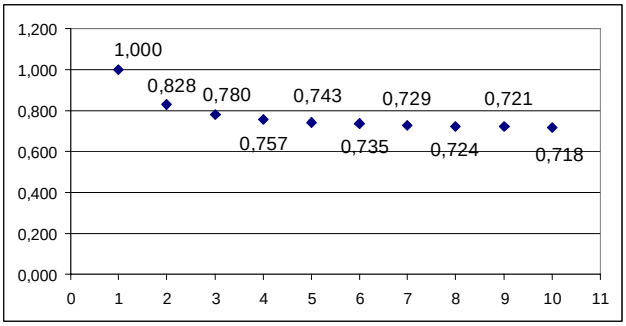
- Définition RM : la priorité d'une tâche est inversement proportionnelle à sa période
- Propriété RM : optimal parmi les algorithmes à priorités fixes pour des tâches périodiques indépendantes à échéance sur requête (Di=Pi).
- Condition de faisabilité

$$\sum_{i=1}^{n} \frac{Ci}{Pi} \leq n(2^{\frac{1}{n}}-1)$$

  - faible complexité O(n)
  - condition suffisante seulement
  - s'applique aussi quand les tâches ne sont pas synchrones

Analyse d'ordonnançabilité pour tâches périodiques
Faisabilité pour Rate Monotonic (2/2)

- Variation du seuil de charge en fonction de n



- Cas particulier : périodes des tâches sont des harmoniques (pour tout i,j tq Pi<Pj, Pj = k Pi avec k entier) : Ulub = 1

Analyse d'ordonnançabilité pour tâches périodiques
Faisabilité pour Deadline Monotonic

- Hypothèse précédentes sauf H3 : Di ≤ Pi
- Définition DM : priorité inversement proportionnelle à l'échéance relative
- Propriété DM
  - Optimal dans la classe des algorithmes à priorités fixes pour des tâches périodiques indépendantes à échéance ≤ période
- Condition de faisabilité $\sum_{i=1}^{n} \frac{Ci}{Di} \leq n(2^{\frac{1}{n}}-1)$
  - faible complexité O(n)
  - condition suffisante seulement (si Σ (Ci/Di) > U_{lub}, le jeu de tâches peut respecter ses échéances)
  - s'applique aussi quand les tâches ne sont pas synchrones

Analyse d'ordonnançabilité pour tâches périodiques
Response Time Analysis (1/4)

- Response time analysis (RTA, analyse de temps de réponse) [Joseph & Pandya, 1986, Audsley & al, 1993]
  - Applicable pour tous les ordonnancements à priorité statique, quel que soit l'assignation des priorités (RM, DM, autre)
  - Condition nécessaire et suffisante
  - Utilisable pour tout Di (même si non inférieur à Pi : dans ce cas, RM et DM ne sont plus optimaux)
  - S'applique aussi quand les tâches ne sont pas synchrones, mais est alors une condition suffisante seulement
- Principe

Analyse d'ordonnançabilité pour tâches périodiques
Response Time Analysis (2/4)

- Introduction au calcul de temps de réponse Ri
  - Ri = Ci (temps d'exécution pire-cas de Ti)
    + Ii (interférences dues à des préemptions par tâches plus prioritaires)
  - Nombre maximal de préemptions subies par une instance de Ti : $\sum_{j \in hp(i)} \left\lceil \frac{Ri}{Pj} \right\rceil$
  - Interférence correspondante (Cj pour chaque préemption par Tj) $Ii = \sum_{j \in hp(i)} \left\lceil \frac{Ri}{Pj} \right\rceil Cj$
  - Temps de réponse Ri $Ri = Ci + \sum_{j \in hp(i)} \left\lceil \frac{Ri}{Pj} \right\rceil Cj$
- Pas de solution simple (Ri des deux cotés)
  - Solution Ri = plus petite valeur vérifiant l'équation
  - Pas besoin de vérifier l'équation ∀ Ri ≤ Di (slt qd nb préemptions augmente)

Analyse d'ordonnançabilité pour tâches périodiques
Response Time Analysis (3/4)

- Algorithme de calcul des Ri (itératif)

$$w_i^0 = C_i$$
$$w_i^{k+1} = C_i + \sum_{j \in hp(i)} \left\lceil \frac{w_i^k}{P_j} \right\rceil C_j$$

- Quand la série converge, on a calculé le temps de réponse Ri
- Le système est ordonnançable quand Ri ≤ Di

Analyse d'ordonnançabilité pour tâches périodiques
Response Time Analysis (4/4)

- Exemple

	Ci	Pi	prio
T4	3	10	
T3	1	5	
T2	2	20	
T1	2	20	

$$w_1^0 = C_1 = 2$$

$$w_1^1 = C_1 + \left\lceil \frac{C_1}{P_4} \right\rceil C_4 + \left\lceil \frac{C_1}{P_3} \right\rceil C_3 + \left\lceil \frac{C_1}{P_2} \right\rceil C_2 = 2 + \left\lceil \frac{2}{10} \right\rceil 3 + \left\lceil \frac{2}{5} \right\rceil 1 + \left\lceil \frac{2}{20} \right\rceil 2 = 2+3+1+2 = 8$$

$$w_1^2 = C_1 + \left\lceil \frac{8}{P_4} \right\rceil C_4 + \left\lceil \frac{8}{P_3} \right\rceil C_3 + \left\lceil \frac{8}{P_2} \right\rceil C_2 = 2 + \left\lceil \frac{8}{10} \right\rceil 3 + \left\lceil \frac{8}{5} \right\rceil 1 + \left\lceil \frac{8}{20} \right\rceil 2 = 2+3+2+2 = 9$$

$$w_1^3 = C_1 + \left\lceil \frac{9}{P_4} \right\rceil C_4 + \left\lceil \frac{9}{P_3} \right\rceil C_3 + \left\lceil \frac{9}{P_2} \right\rceil C_2 = 2 + \left\lceil \frac{9}{10} \right\rceil 3 + \left\lceil \frac{9}{5} \right\rceil 1 + \left\lceil \frac{9}{20} \right\rceil 2 = 2+3+2+2 = 9$$

Analyse d'ordonnançabilité pour tâches périodiques
Faisabilité pour EDF (Di=Pi)

- Définition EDF [Liu & Layland, 1973]
  - A un instant donné, la tâche la plus prioritaire (parmi les tâches prêtes) est celle dont l'échéance absolue est la plus proche
- Propriété EDF
  - Optimal dans la classe des algorithmes à priorités dynamiques pour des configurations de tâches périodiques indépendantes avec échéance ≤ période
- Condition de faisabilité (Di=Pi)
  - faible complexité O(n)
  - condition nécessaire et suffisante
  - s'applique aussi quand les tâches ne sont pas synchrones

$$\sum_{i=1}^{n} \frac{Ci}{Pi} \leq 1$$

Vérification d'ordonnançabilité
Bilan

- De nombreux résultats théoriques en ordonnancement temps-réel
  - Difficile d'en faire le tour en peu de temps
  - Ici, on n'a présenté que les principaux résultats
- Attention aux hypothèses d'application
  - Simples formules ou algorithmes …
- Utiliser les résultats théoriques existants (degré de confiance par rapport aux méthodes de test, utilisables dans les phases préliminaires du développement)
- Peu d'outils commerciaux utilisant ces résultats