Ordonnancement temps réel avec gestion de l'énergie renouvelable

Younes Chandarli

Directeur : Laurent George

Encadrant : Damien Masson

Octobre 2011 - Décembre 2014

2 Décembre 2014

UNIVERSITÉ PARIS-EST

LABORATOIRE D'INFORMATIQUE GASPARD-MONGE

Sous la co-tutelle de :
CNRS
ÉCOLE DES PONTS PARISTECH
ESIEE PARIS
UPEM • UNIVERSITÉ PARIS-EST MARNE-LA-VALLÉE

Plan

1. Introduction
2. Problématiques
3. État de l'art
4. Contributions
5. Conclusion

Plan

1 Introduction

2 Problématiques

3 État de l'art

4 Contributions

5 Conclusion

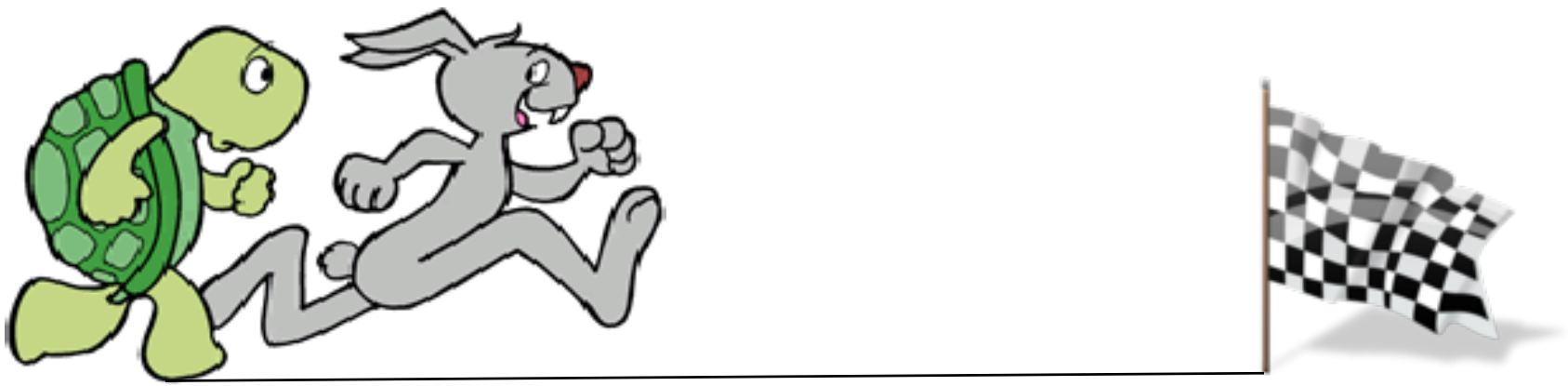
Motivation

Ordonnancement temps réel des systèmes collecteurs d'énergie sur monoprocesseur

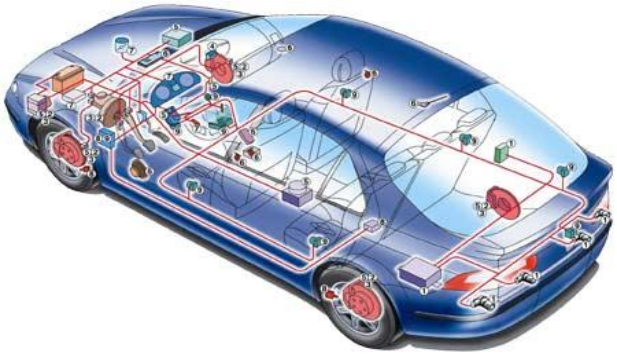
Système temps réel

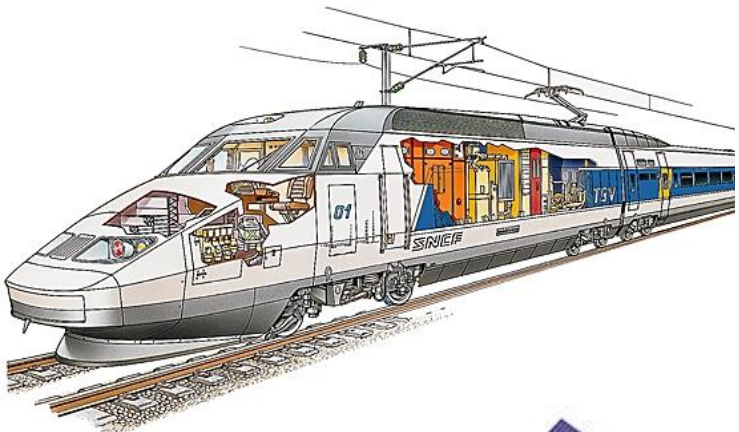
Système temps réel

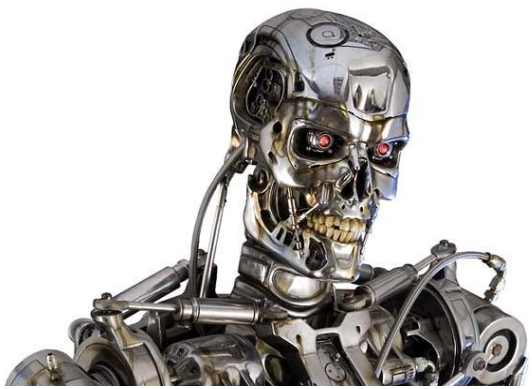
En informatique, un système est considéré comme étant temps réel si le respect de ses contraintes temporelles est aussi important que l'exactitude des résultats.



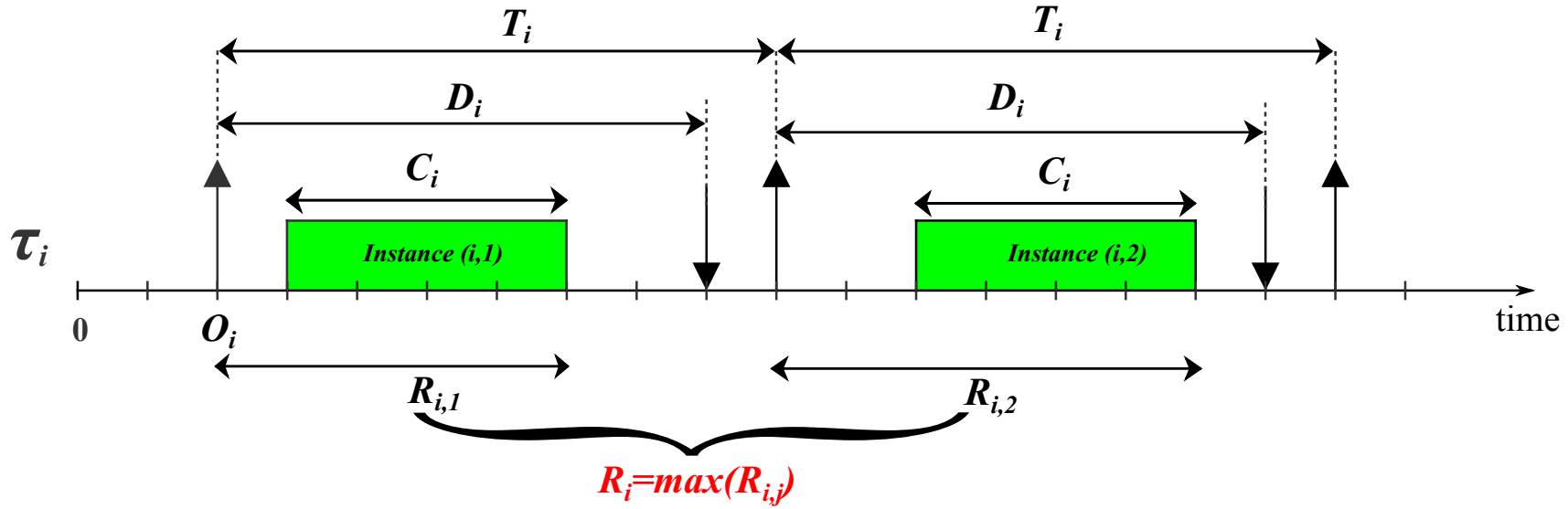
Applications temps réel







Tâches temps réel



- C_i : le pire temps d'exécution
- D_i : l'échéance relative
- T_i : la période
- O_i : la première date d'activation
- P_i : la priorité
- R_i : le pire temps de réponse

C. L. Liu, and James W. Layland,
"Hard Real-Time Computing Systems", ACM Journal, 1973.

Tâches temps réel

- Problématique : exécuter toutes les tâches en respectant leurs échéances
- Algorithmes d'ordonnancement :
  - Stratégies à priorité de tâche fixe (RM, DM)
  - Stratégies à priorité d'instance fixe (EDF)
  - Stratégies à priorités dynamiques (LLF)

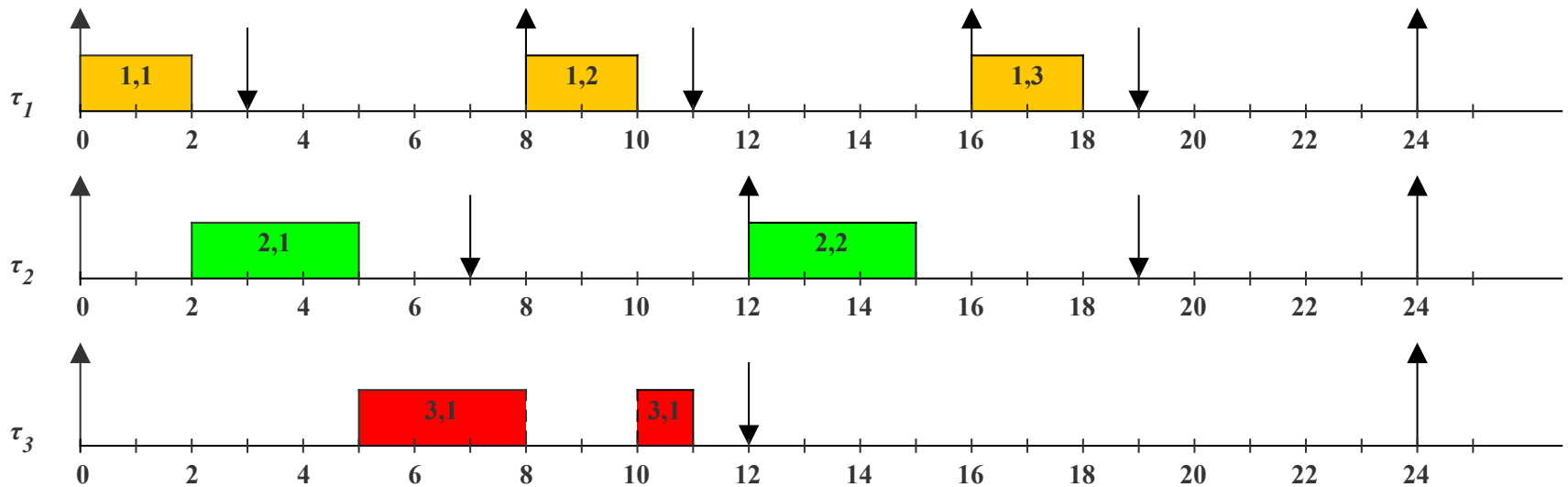


FIGURE: Un système ordonnancé en priorité de tâche fixe

Définitions

- Algorithme optimal : si un système donné n'est pas ordonnançable avec l'algorithme optimal, alors il n'existe pas de solution pour ce système
- Systèmes non-concrets : Il n'y pas d'hypothèses sur certains paramètres
- Utilisation processeur du système : $U = \sum_{i=1}^{n} C_i / T_i$
- Demande processeur d'une tâche : $wp_i(t) = \left\lceil \frac{t - O_i}{T_i} \right\rceil \times C_i$
- Slack-time ou Quantité de temps creux d'une instance : la somme des temps d'inactivité du processeur jusqu'à son échéance

Définitions

- Algorithme optimal : si un système donné n'est pas ordonnançable avec l'algorithme optimal, alors il n'existe pas de solution pour ce système
- Systèmes non-concrets : Il n'y pas d'hypothèses sur certains paramètres
- Utilisation processeur du système : $U = \sum_{i=1}^{n} C_i / T_i$
- Demande processeur d'une tâche : $wp_i(t) = \left\lceil \frac{t - O_i}{T_i} \right\rceil \times C_i$
- Slack-time ou Quantité de temps creux d'une instance : la somme des temps d'inactivité du processeur jusqu'à son échéance

Définitions

- Algorithme optimal : si un système donné n'est pas ordonnançable avec l'algorithme optimal, alors il n'existe pas de solution pour ce système
- Systèmes non-concrets : Il n'y pas d'hypothèses sur certains paramètres
- Utilisation processeur du système : $U = \sum_{i=1}^{n} C_i / T_i$
- Demande processeur d'une tâche : $wp_i(t) = \left\lceil \frac{t - O_i}{T_i} \right\rceil \times C_i$
- Slack-time ou Quantité de temps creux d'une instance : la somme des temps d'inactivité du processeur jusqu'à son échéance

Définitions

- Algorithme optimal : si un système donné n'est pas ordonnançable avec l'algorithme optimal, alors il n'existe pas de solution pour ce système
- Systèmes non-concrets : Il n'y pas d'hypothèses sur certains paramètres
- Utilisation processeur du système : $U = \sum_{i=1}^{n} C_i / T_i$
- Demande processeur d'une tâche : $wp_i(t) = \left\lceil \frac{t - O_i}{T_i} \right\rceil \times C_i$
- Slack-time ou Quantité de temps creux d'une instance : la somme des temps d'inactivité du processeur jusqu'à son échéance

Définitions

- Algorithme optimal : si un système donné n'est pas ordonnançable avec l'algorithme optimal, alors il n'existe pas de solution pour ce système
- Systèmes non-concrets : Il n'y pas d'hypothèses sur certains paramètres
- Utilisation processeur du système : $U = \sum_{i=1}^{n} C_i / T_i$
- Demande processeur d'une tâche : $wp_i(t) = \left\lceil \frac{t - O_i}{T_i} \right\rceil \times C_i$
- Slack-time ou Quantité de temps creux d'une instance : la somme des temps d'inactivité du processeur jusqu'à son échéance

Analyse de temps de réponse classique

Méthode

1. Calculer le temps de réponse des tâches dans le pire scénario,
2. Test d'ordonnançabilité exact : le système de tâches est ordonnançable si $\forall \tau_i, R_i \leq D_i$.

Pour l'ordonnancement à priorité fixe classique

1. Pire scénario : l'activation synchrone de toutes les tâches
2. Calcul du pire temps de réponse de chaque tâche τ_i avec l'algorithme itératif :

$$\begin{cases} wp_i^0 & = & C_i \\ wp_i^{m+1} & = & C_i + \sum_{j<i} \left\lceil \frac{wp_i^m}{T_j} \right\rceil \times C_j \\ R_i & = & \min_{wp_i^{m+1}=wp_i^m}(wp_i^m) \end{cases}$$

Analyse de temps de réponse classique

Méthode

1. Calculer le temps de réponse des tâches dans le pire scénario,
2. Test d'ordonnançabilité exact : le système de tâches est ordonnançable si $\forall \tau_i, R_i \leq D_i$.

Pour l'ordonnancement à priorité fixe classique

1. Pire scénario : l'activation synchrone de toutes les tâches
2. Calcul du pire temps de réponse de chaque tâche τ_i avec l'algorithme itératif :

$$\begin{cases} wp_i^0 & = & C_i \\ wp_i^{m+1} & = & C_i + \sum_{j<i} \left\lceil \frac{wp_i^m}{T_j} \right\rceil \times C_j \\ R_i & = & \min_{wp_i^{m+1}=wp_i^m}(wp_i^m) \end{cases}$$

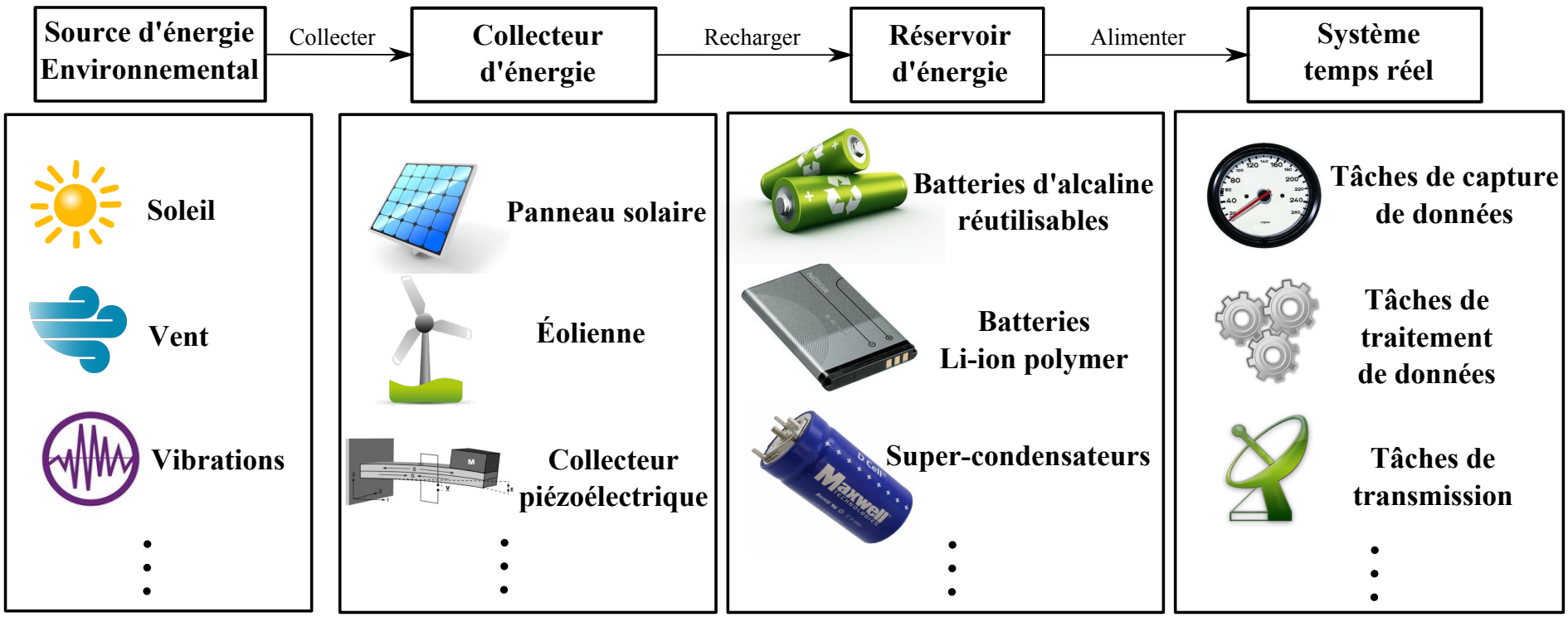
Motivation

Ordonnancement temps réel des systèmes

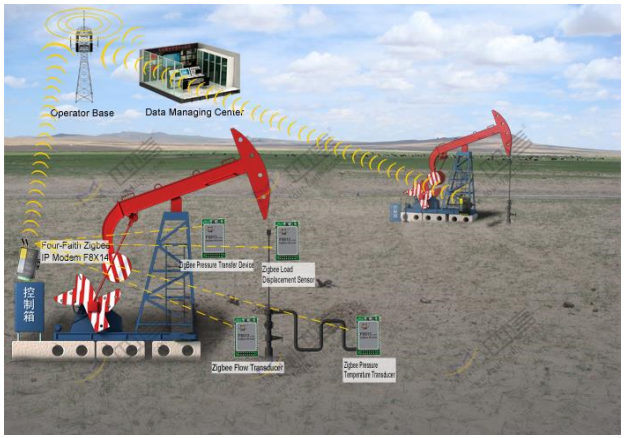
collecteurs d'énergie

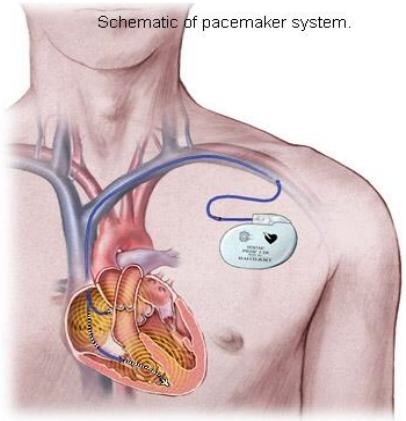
sur monoprocesseur

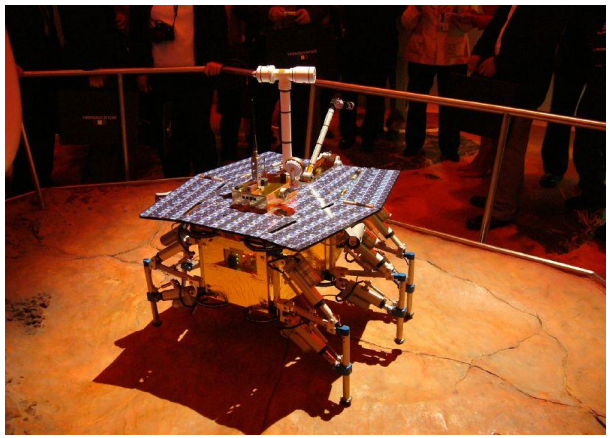
Les systèmes collecteurs d'énergie

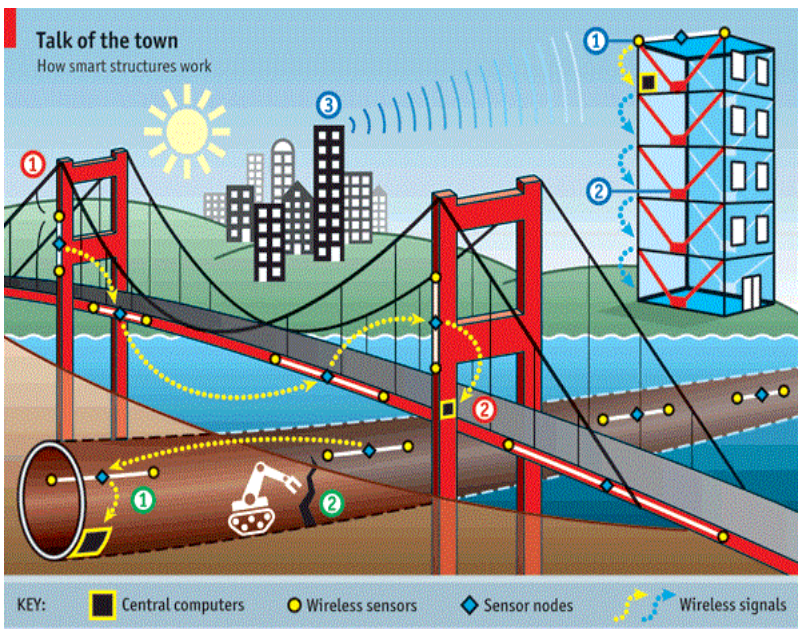


Applications des systèmes collecteurs d'énergie









Plan

1 Introduction

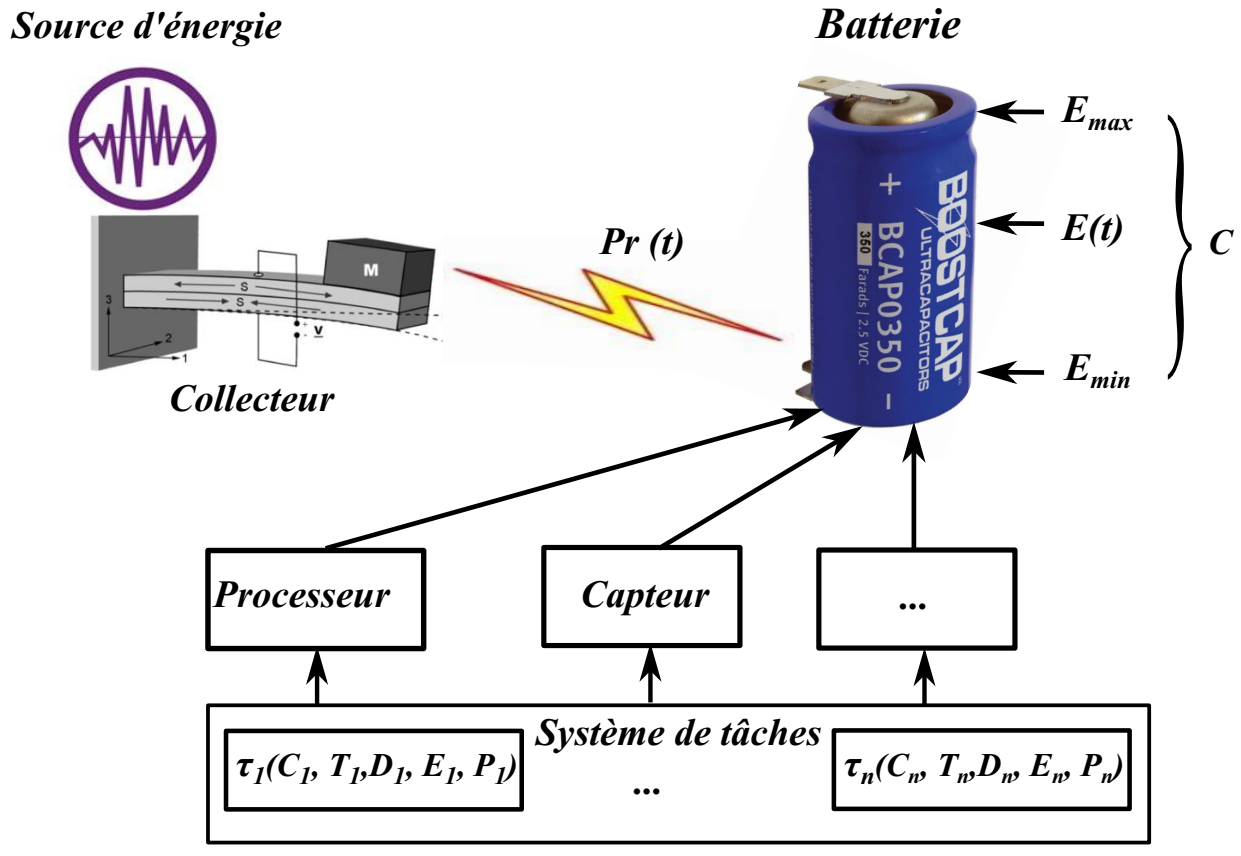
2 Problématiques

3 État de l'art

4 Contributions

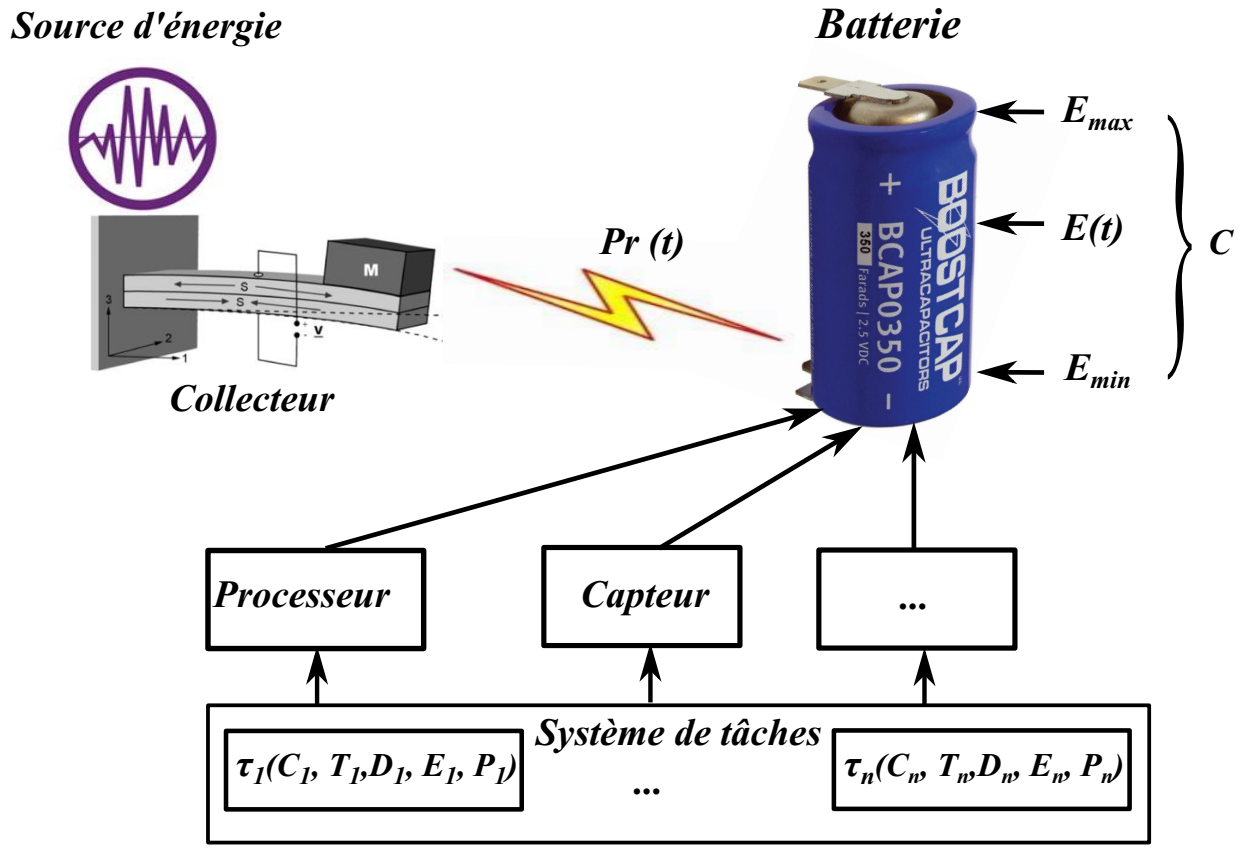
5 Conclusion

Modèle



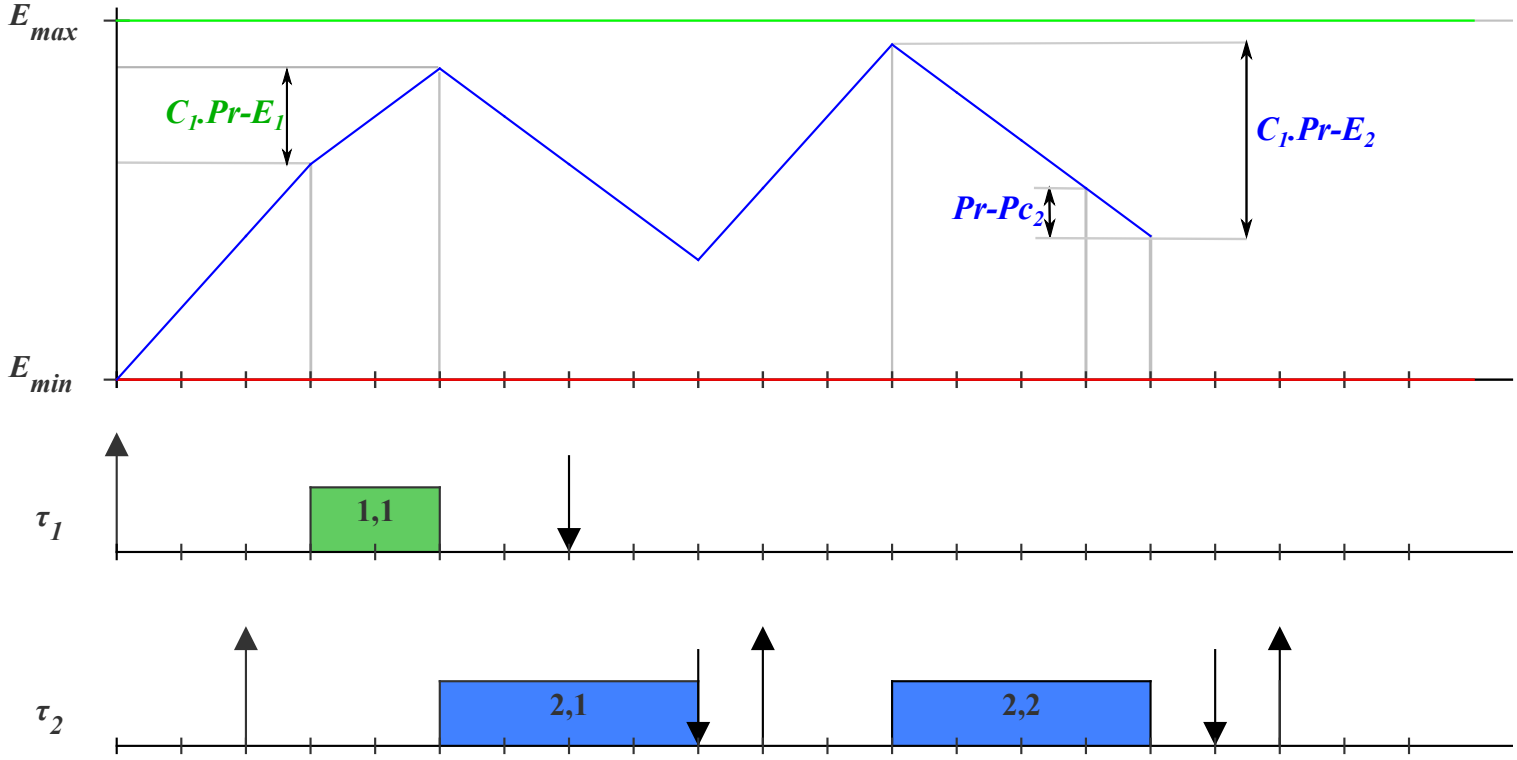
- Le modèle d'énergie
  - $P_r(t)$: la fonction de rechargement constante : $P_r(t) = P_r$
  - Taux de consommation constants et différents pour chaque tâche

Modèle



- Le modèle d'énergie
  - $P_r(t)$: la fonction de rechargement constante : $P_r(t) = P_r$
  - Taux de consommation constants et différents pour chaque tâche

Tâches temps réel avec les paramètres d'énergie

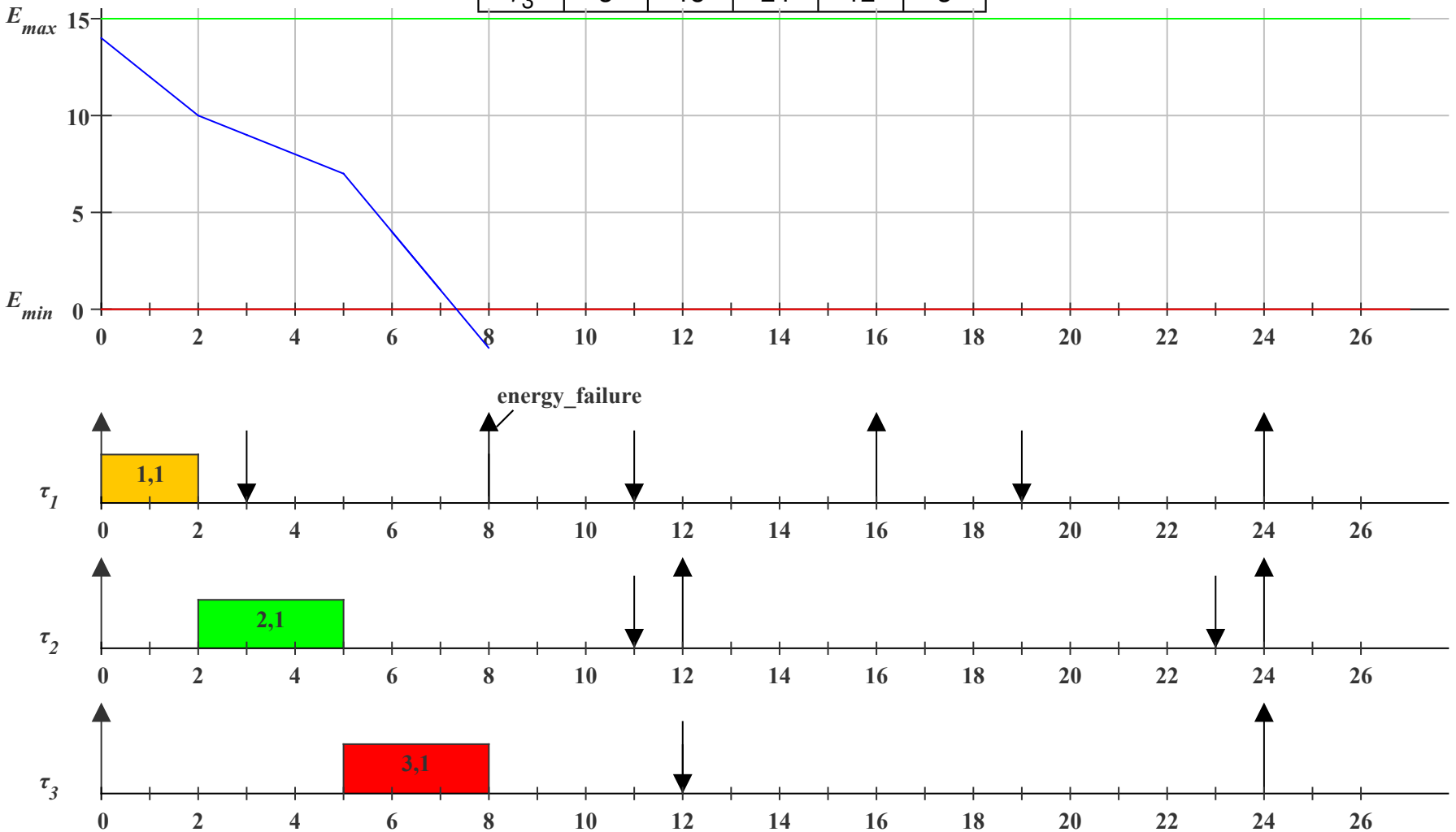


En plus des paramètres temporels classiques, la tâche τ_i est caractérisée par :

- E_i : sa pire consommation d'énergie
- Pc_i : son taux de consommation ($Pc_i = E_i / C_i$)
  - La tâche est dite consommatrice d'énergie si $Pc_i > P_r$
  - La tâche est dite génératrice d'énergie si $Pc_i \leq P_r$

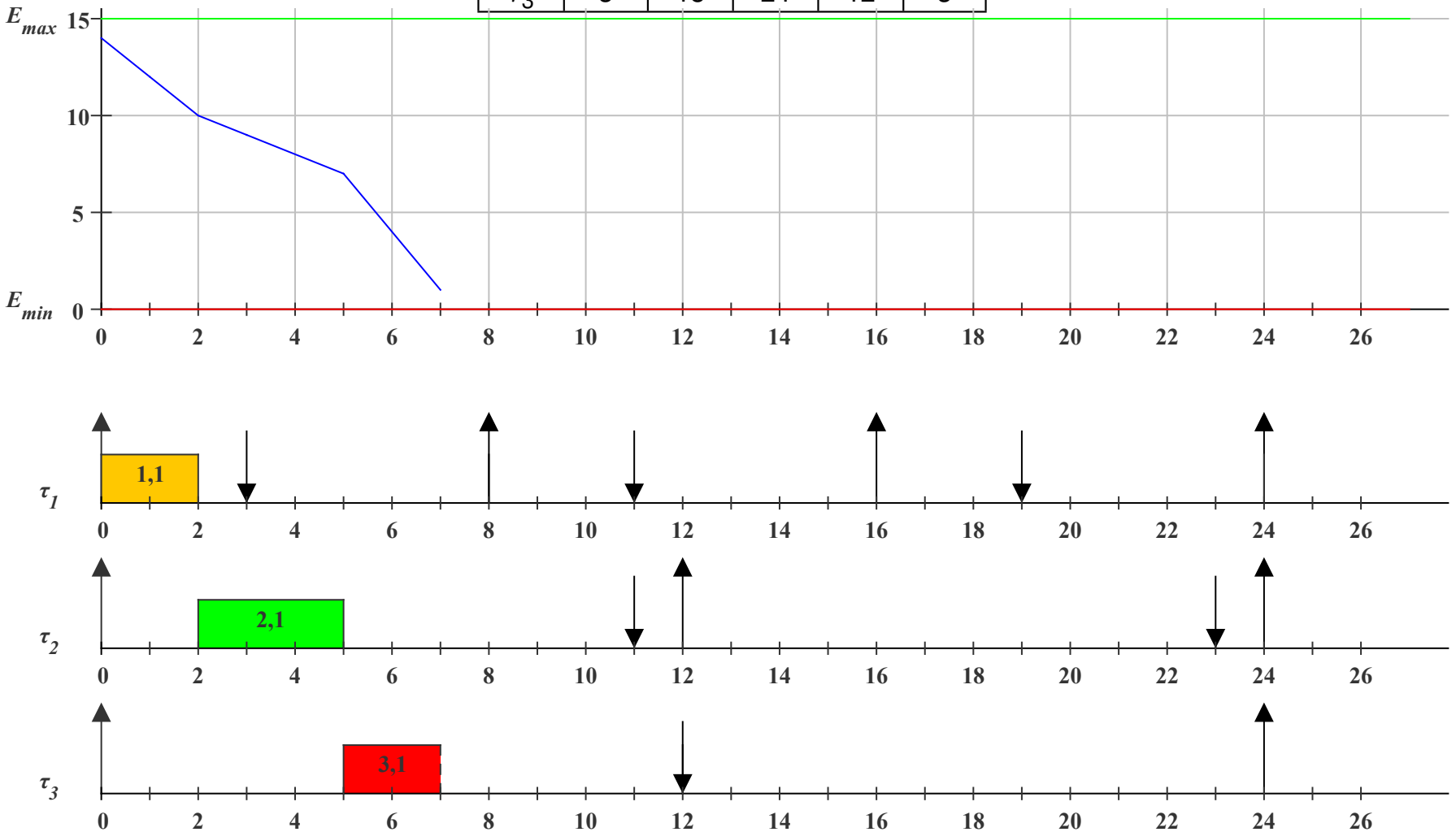
Ordonnancement des systèmes collecteurs d'énergie

-	C_i	E_i	T_i	D_i	P_i
τ_1	2	10	8	3	1
τ_2	3	12	12	11	2
τ_3	3	18	24	12	3



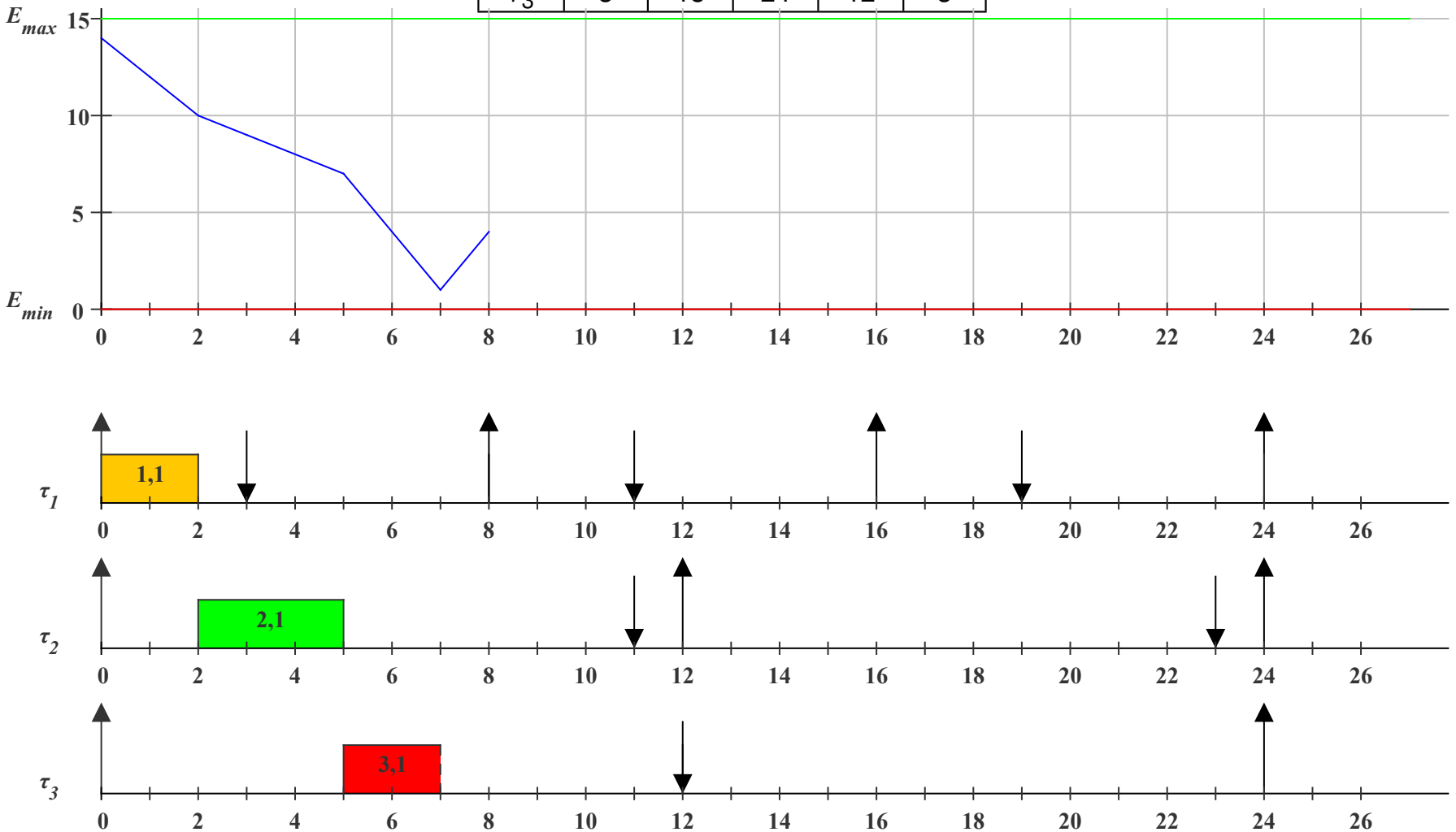
Ordonnancement des systèmes collecteurs d'énergie

-	C_i	E_i	T_i	D_i	P_i
τ_1	2	10	8	3	1
τ_2	3	12	12	11	2
τ_3	3	18	24	12	3



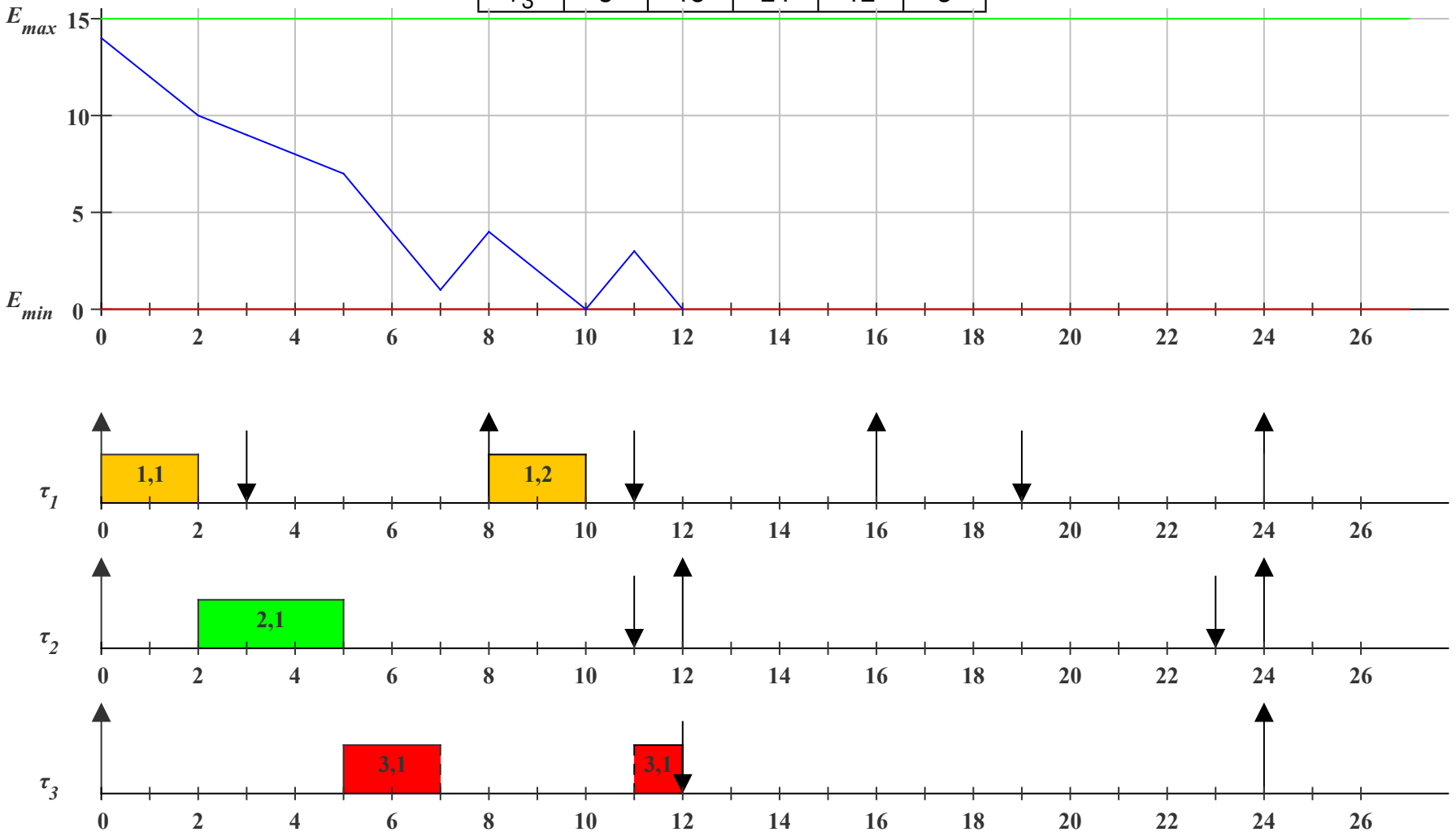
Ordonnancement des systèmes collecteurs d'énergie

-	C_i	E_i	T_i	D_i	P_i
τ_1	2	10	8	3	1
τ_2	3	12	12	11	2
τ_3	3	18	24	12	3



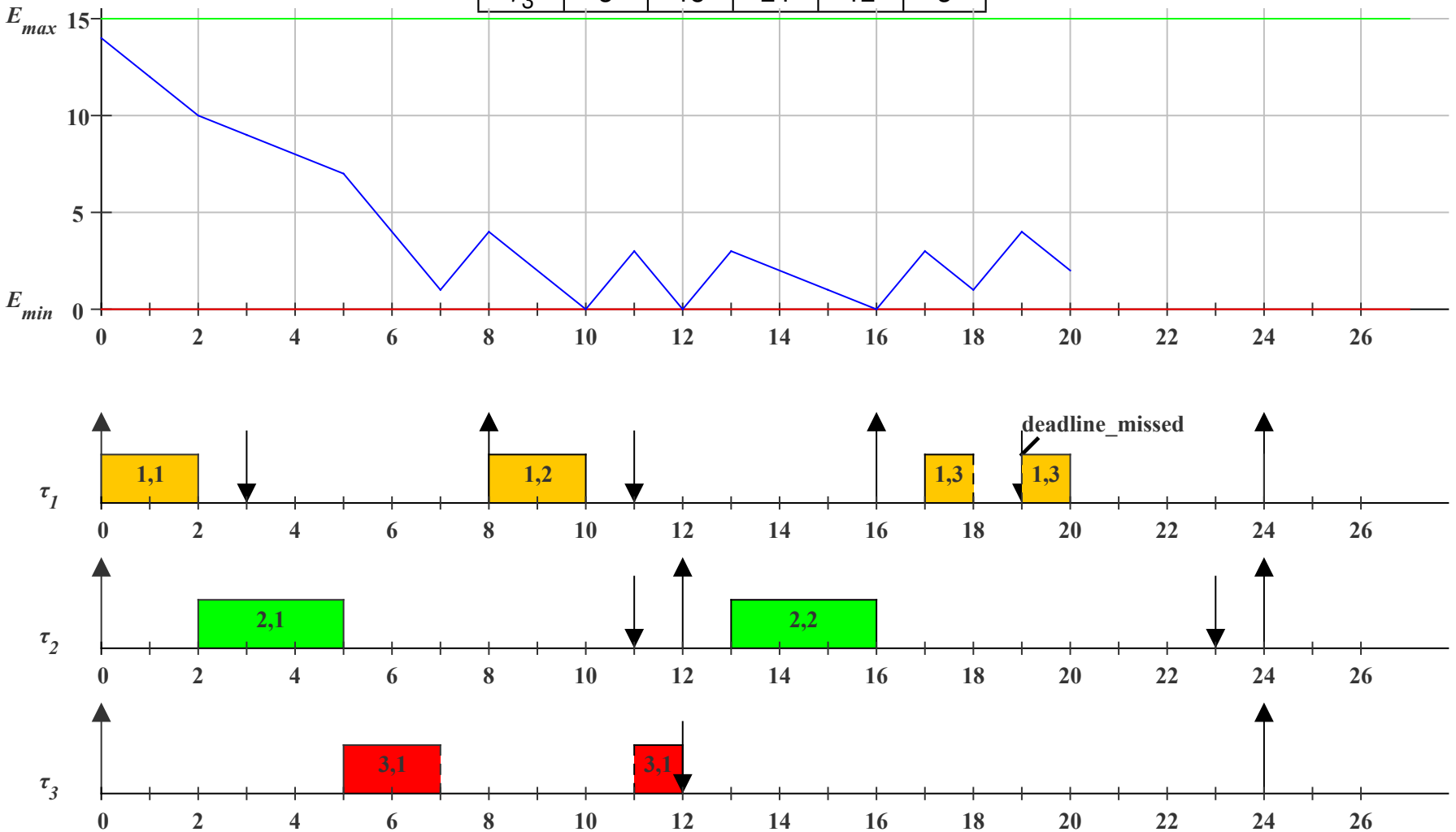
Ordonnancement des systèmes collecteurs d'énergie

-	C_i	E_i	T_i	D_i	P_i
τ_1	2	10	8	3	1
τ_2	3	12	12	11	2
τ_3	3	18	24	12	3



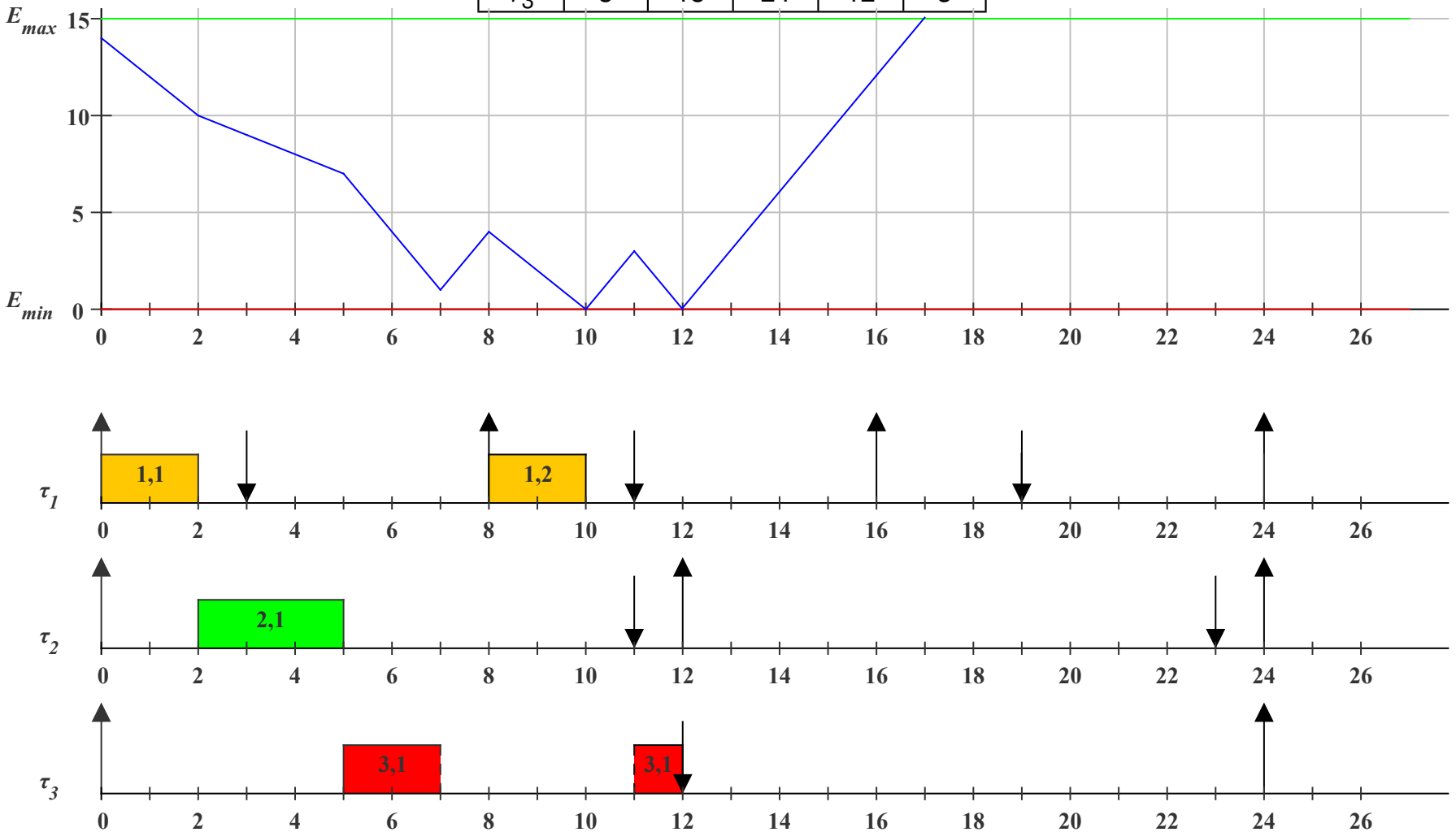
Ordonnancement des systèmes collecteurs d'énergie

-	C_i	E_i	T_i	D_i	P_i
τ_1	2	10	8	3	1
τ_2	3	12	12	11	2
τ_3	3	18	24	12	3



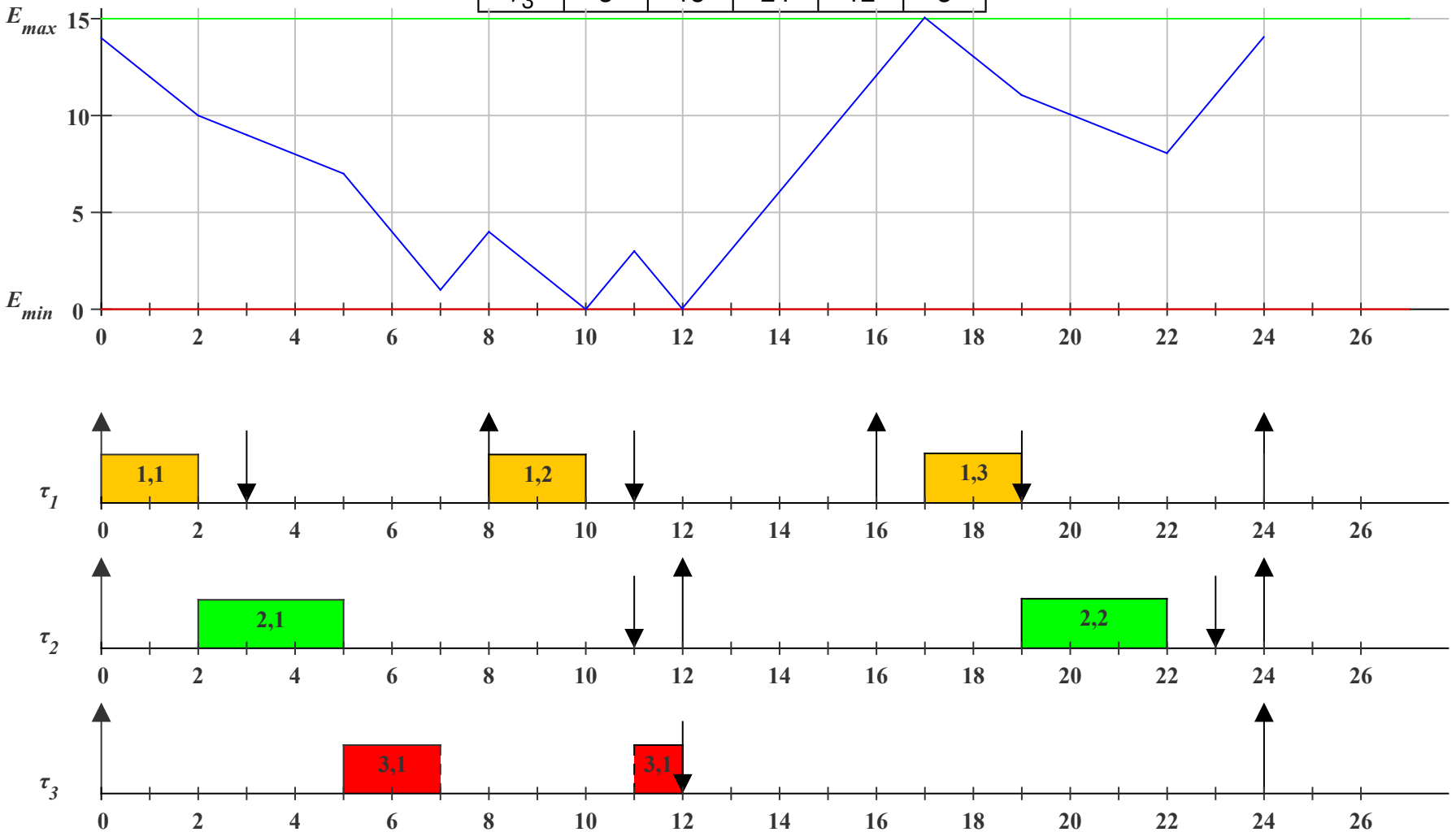
Ordonnancement des systèmes collecteurs d'énergie

-	C_i	E_i	T_i	D_i	P_i
τ_1	2	10	8	3	1
τ_2	3	12	12	11	2
τ_3	3	18	24	12	3



Ordonnancement des systèmes collecteurs d'énergie

-	C_i	E_i	T_i	D_i	P_i
τ_1	2	10	8	3	1
τ_2	3	12	12	11	2
τ_3	3	18	24	12	3



Les hypothèses

- Ordonnancement préemptif à priorité de tâche fixe
- Taux de rechargement constant ($P_r(t) = P_r$)
- Pc_i constant et différent pour chaque tâche
- Échéances contraintes ou implicites ($\forall_i \; D_i \leq T_i$)

Définitions

- Utilisation d'énergie du système : $Ue = \sum_{i=1}^{n} E_i/(T_i \times P_r)$
- Demande d'énergie d'une tâche : $we_i(t) = \left\lceil \frac{t-O_i}{T_i} \right\rceil \times E_i$
- Slack-energy ou surplus d'énergie d'une instance : l'excédant énergétique entre l'instant courant et l'échéance de l'instance.

Problématiques

1. **Les algorithmes** : Comment peut-on ordonnancer un ensemble de tâches à priorité fixe sur un seul processeur de telle sorte que les échéances de toutes les tâches soient respectées et que leur demande d'énergie soit satisfaite tout en gardant le niveau du réservoir entre E_{min} et E_{max} ?

2. **Les conditions d'ordonnançabilité** : Existe-t-il une condition nécessaire et suffisante pour vérifier l'ordonnançabilité d'un système de tâches donné ?

3. **La capacité minimale du réservoir** : Quelle est la capacité minimale de la batterie qui préserve l'ordonnançabilité d'un système donné ?

Plan

1 Introduction

2 Problématiques

3 État de l'art

4 Contributions

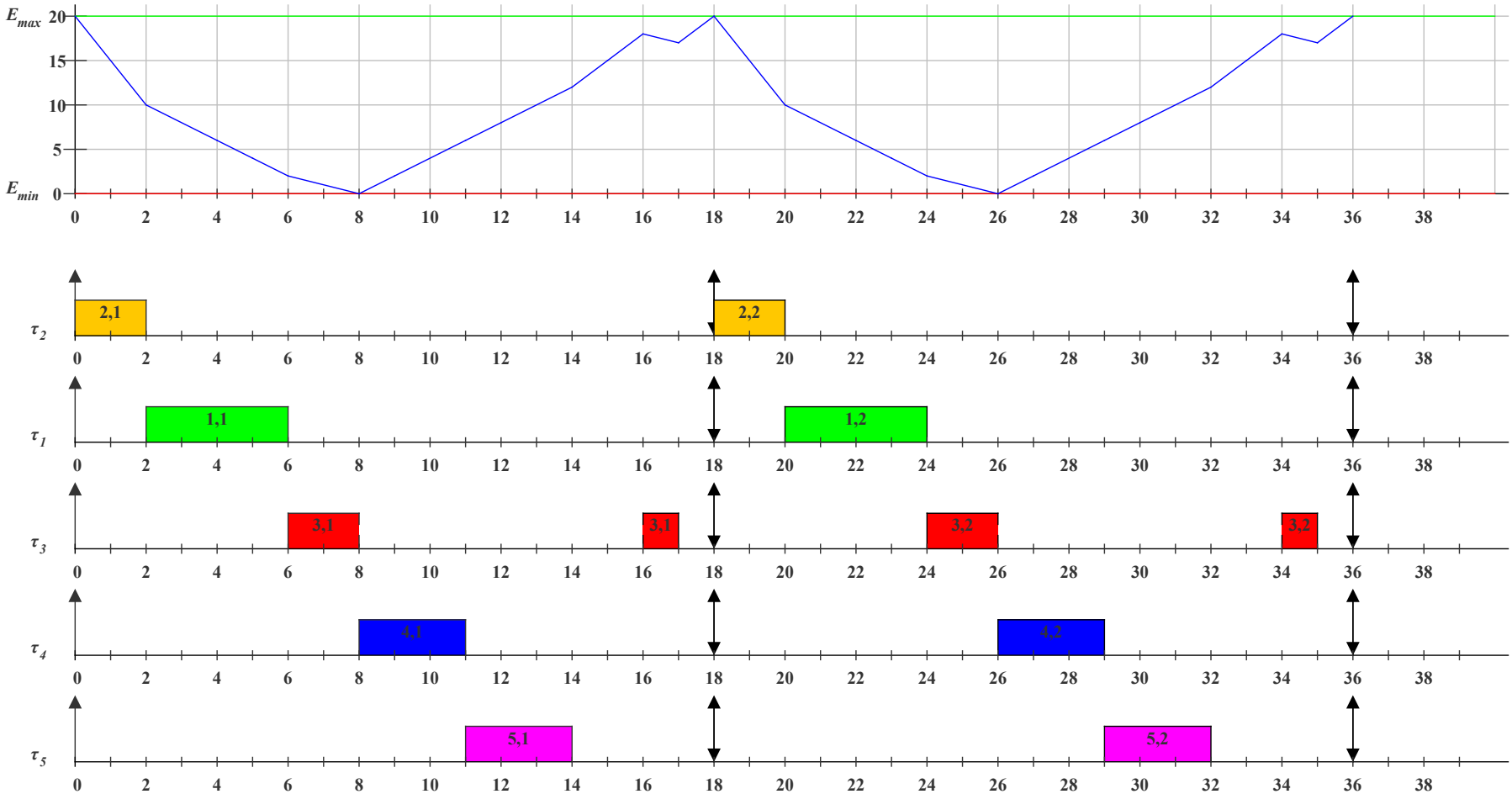
5 Conclusion

État de l'art

Les principaux algorithmes disponibles dans l'état de l'art :

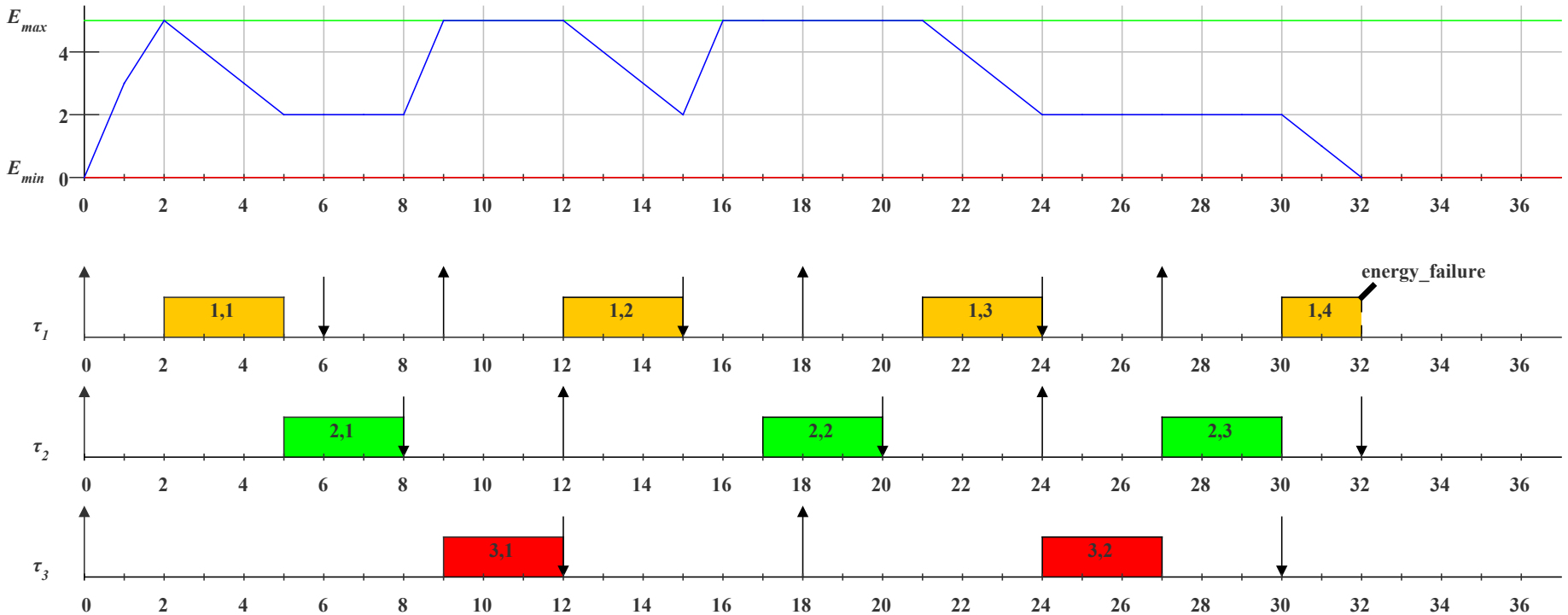
- *EDL* : EDF as Late as possible (1999)
- *FBA* : Frame-Based Algorithm (2001)
- *LSA* : Lazy Scheduling Algorithm(2006)
- *EDeg* : EDF with energy garantee (2011)
- PFP_{ST} : Preemptive Fixed-Priority with Slack-Time (2011)

L'algorithme *FBA* (le modèle Frame-Based)



A. Allavena and D. Mossé,
"Scheduling of Frame-based Embedded Systems with Rechargeable Batteries",
Workshop in conjunction with RTAS, 2001.

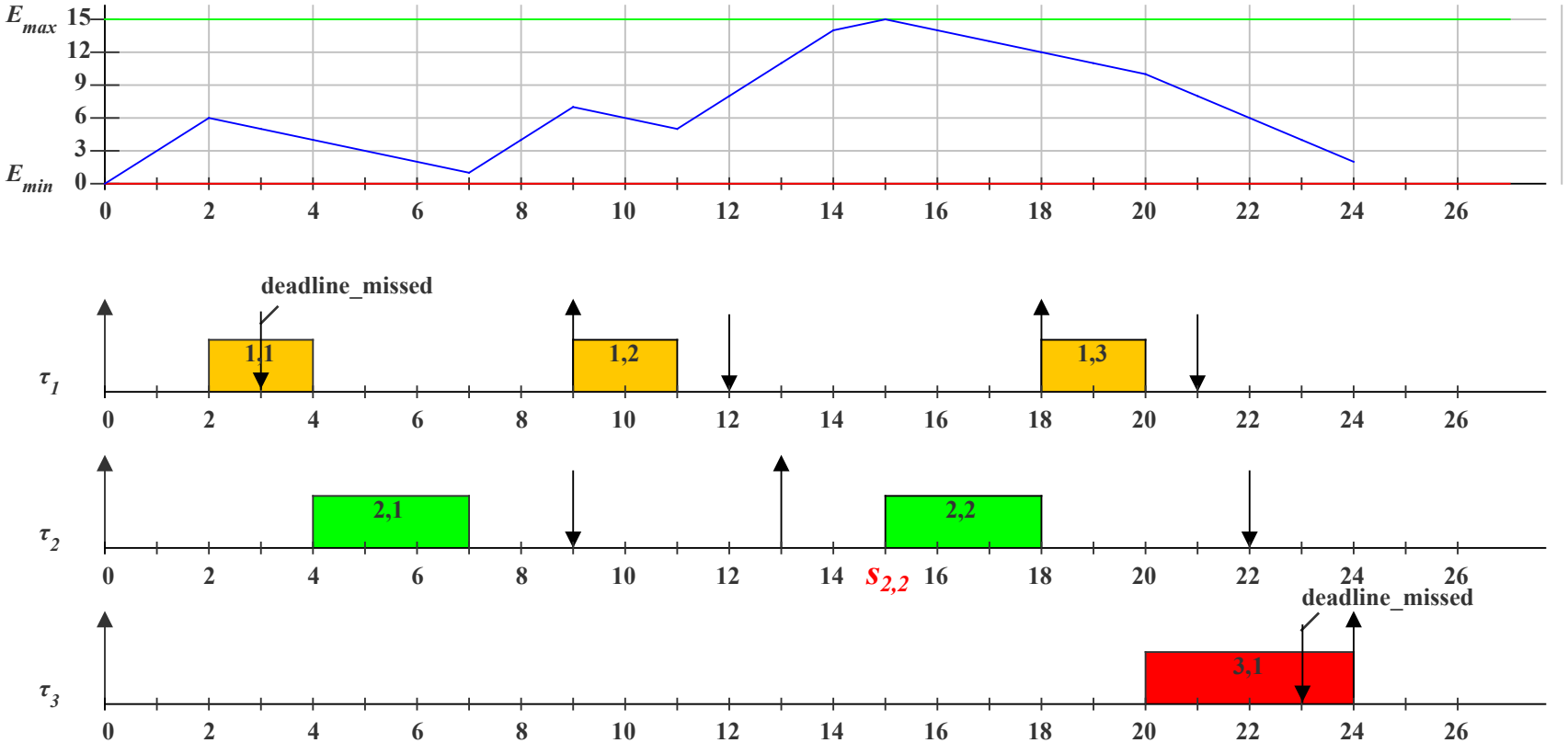
L'algorithme *EDL* : EDF au plus tard



M. Chetto,
"The EDL Server for Scheduling Periodic and Soft AperiodicTasks with Resource Constraints",
Real-Time Systems Journal, 1999.

L'algorithme *LSA*

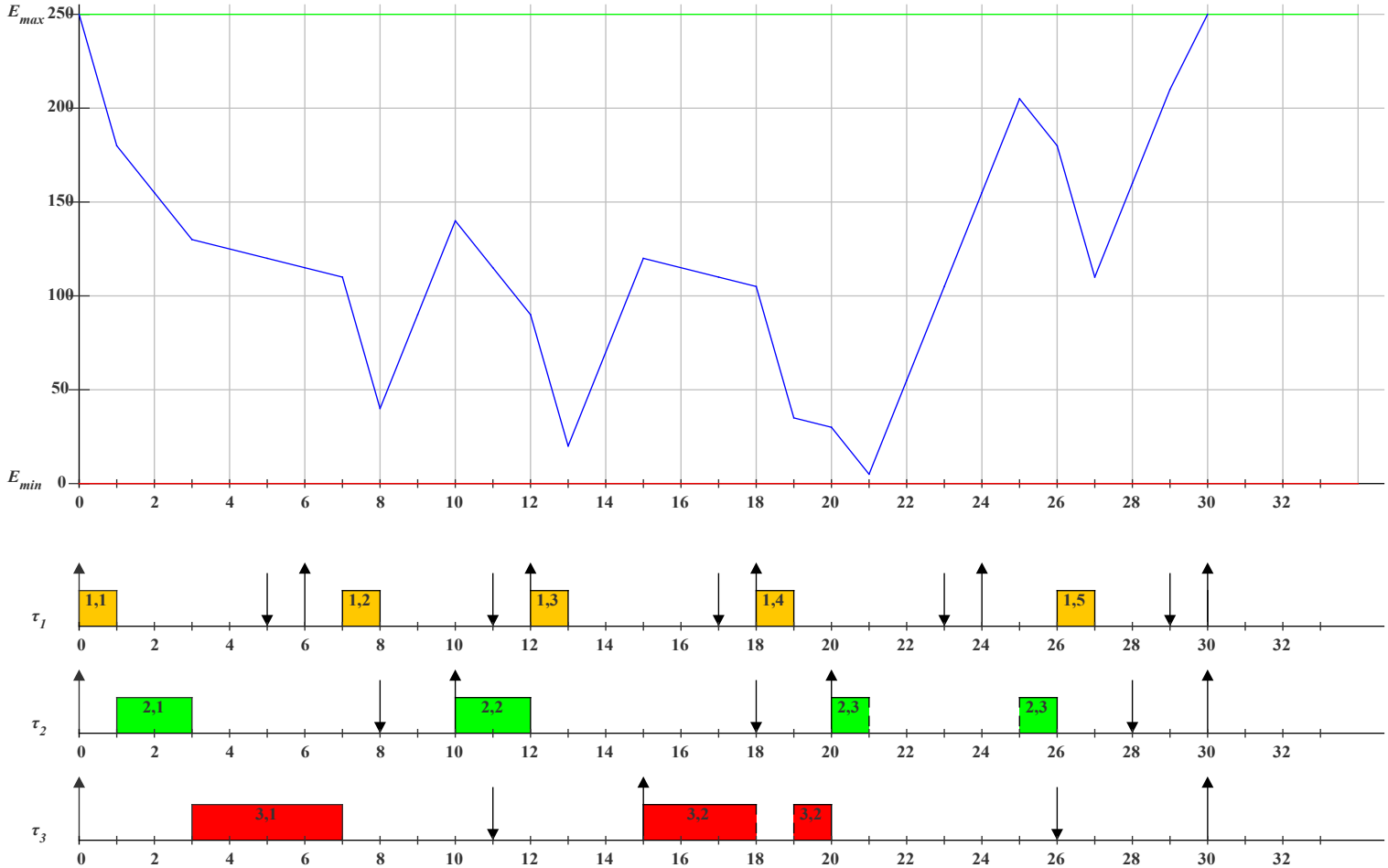
Même taux de consommation pour toutes les tâches



C. Moser and L. Thiele and L. Benini and D. Brunelli,
"Real-Time Scheduling with Regenerative Energy",
ECRTS, 2006.

L'algorithme *EDeg* : EDF avec garanties d'énergie

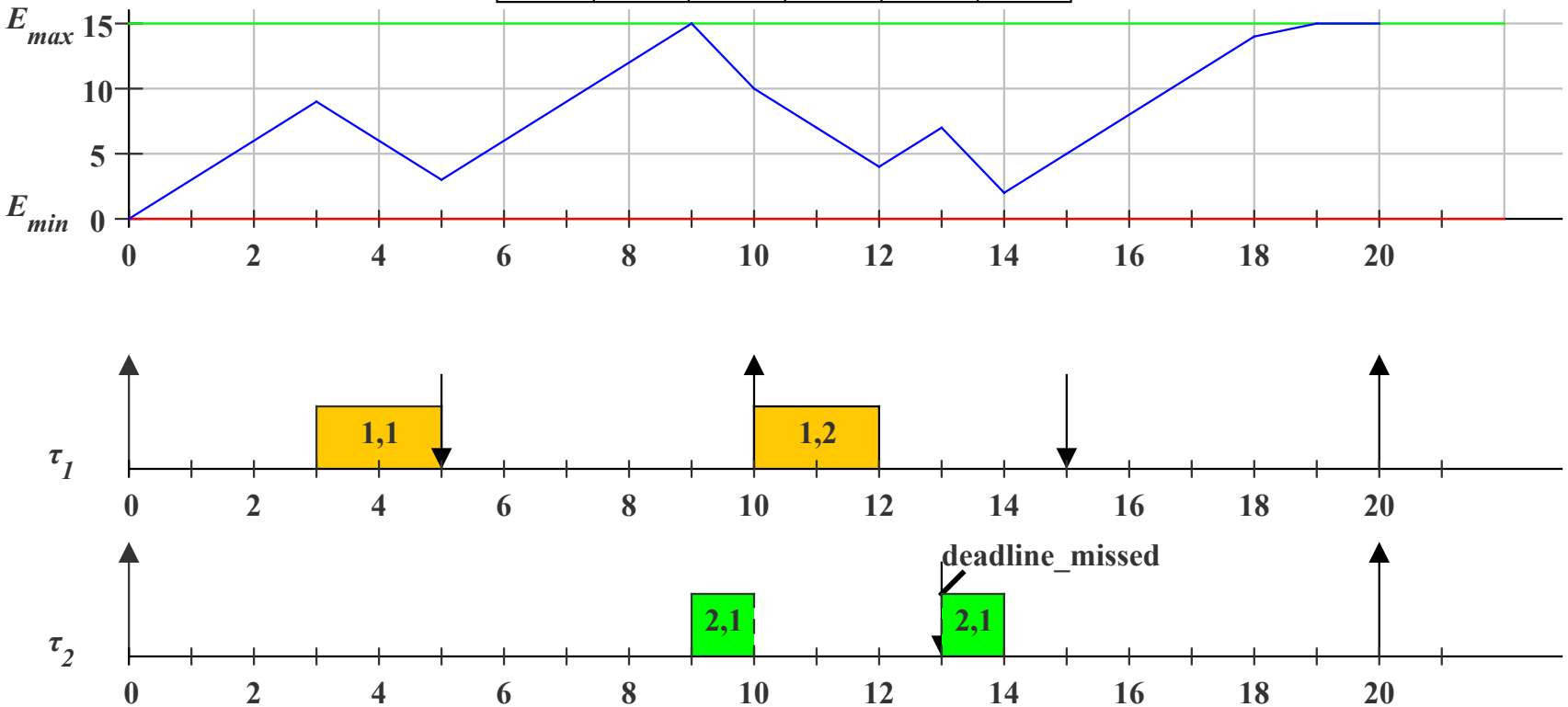
Utilise la notion de Slack-energy et Slack-time



H. EL Ghor, and M. Chetto and R.H. Chehade,
"A Real-time Scheduling Framework for Embedded Systems with Environmental Energy Harvesting",
Computers & Electrical Engineering Journal, 2011.

L'algorithme PFP_{ST} (hybride au plus tôt / au plus tard)

-	C_i	E_i	T_i	D_i	P_i
τ_1	2	12	10	5	1
τ_2	2	16	20	13	2



M. Chetto, and D. Masson and S. Midonnet,
"Fixed Priority Scheduling Strategies for Ambient Energy-Harvesting Embedded systems", Computers & Electrical Green Computing and Communications, 2011.

- *FBA* : optimal uniquement pour le modèle Frame-Based
- *LSA*, *EDL* et *EDeg* : optimaux dans certains cas dans la classe des algorithmes à priorité d'instance fixe (EDF)
- Pas d'algorithme optimal ni de test d'ordonnançabilité en priorité de tâche fixe

Plan

1 Introduction

2 Problématiques

3 État de l'art

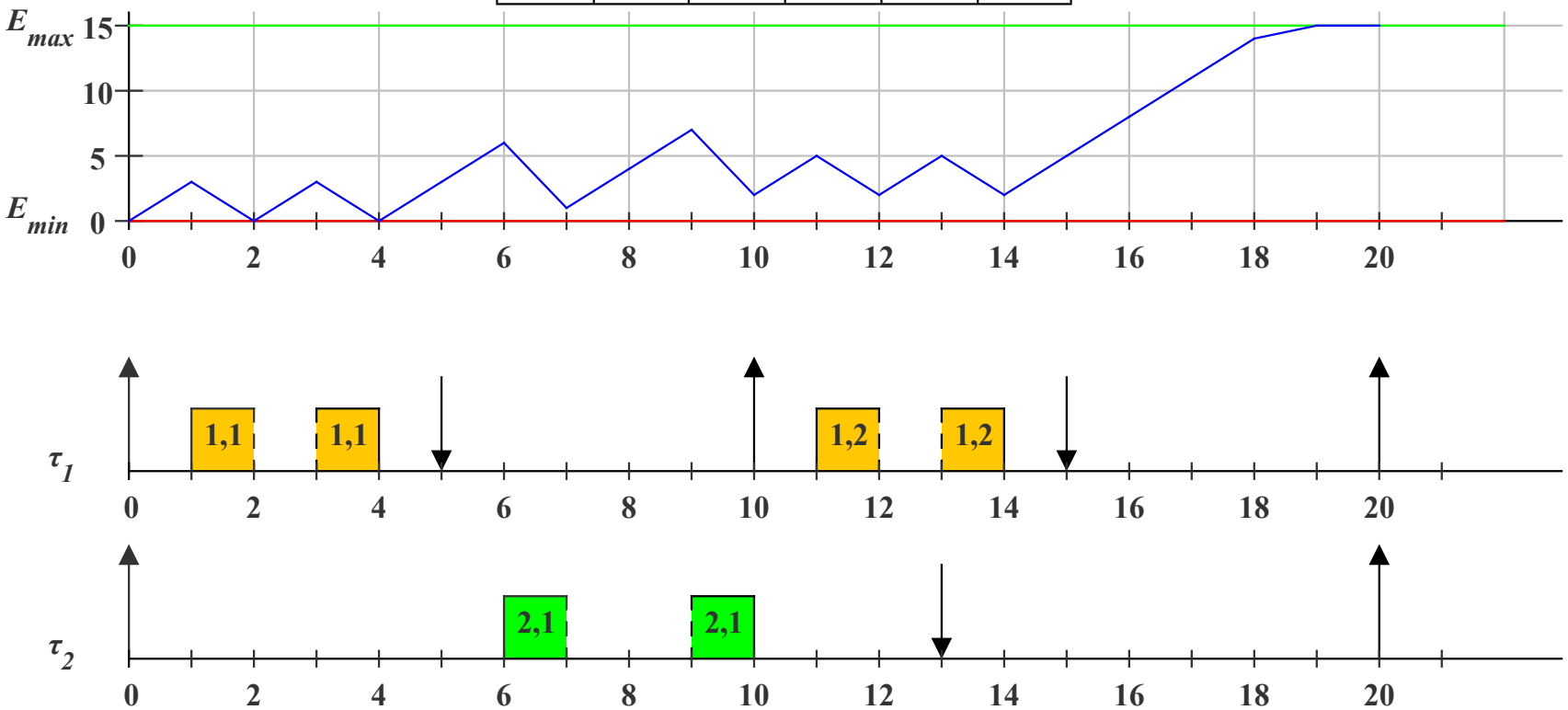
4 Contributions
- Systèmes non-concrets et tâches consommatrices : PFP_{ASAP}
- Systèmes concrets et tâches consommatrices et génératrices : tests d'ordonnançabilité
- Systèmes concrets et tâches consommatrices et génératrices : optimalité
- Systèmes avec contraintes thermiques
- L'outil de simulation YARTISS

Plan

1. Introduction
2. Problématiques
3. État de l'art
4. Contributions
   - Systèmes non-concrets et tâches consommatrices : PFP_{ASAP}
   - Systèmes concrets et tâches consommatrices et génératrices : tests d'ordonnançabilité
   - Systèmes concrets et tâches consommatrices et génératrices : optimalité
   - Systèmes avec contraintes thermiques
   - L'outil de simulation YARTISS
5. Conclusion

L'algorithme PFP_{ASAP} (au plus tôt)

-	C_i	E_i	T_i	D_i	P_i
τ_1	2	12	10	5	1
τ_2	2	16	20	13	2



L'algorithme PFP_{ASAP}

- Exécute les tâches quand l'énergie disponible dans la batterie est suffisante,
- Recharge autant que nécessaire pour exécuter une seule unité de temps.

PFP_{ASAP} est un algorithme énergétiquement non-oisif

Le processeur est inactif uniquement quand l'énergie n'est pas suffisante pour exécuter au moins une unité de temps de la plus prioritaire des tâches actives.

Le pire scénario

Hypothèses

- Le système est non-concret énergétiquement [1]
- Toutes les tâches sont des consommatrices

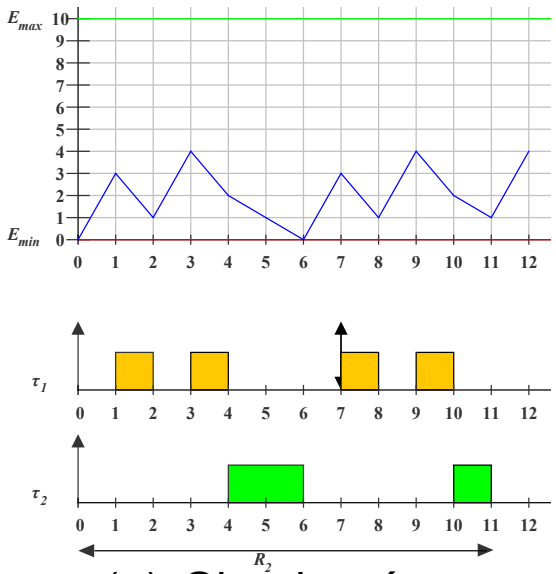
Théorème

Sous ces hypothèses, le pire scénario qu'une tâche peut subir sous PFP_{ASAP} se produit à chaque fois qu'elle est activée simultanément avec toutes les tâches plus prioritaires alors que la batterie est à son seuil minimal E_{min}.

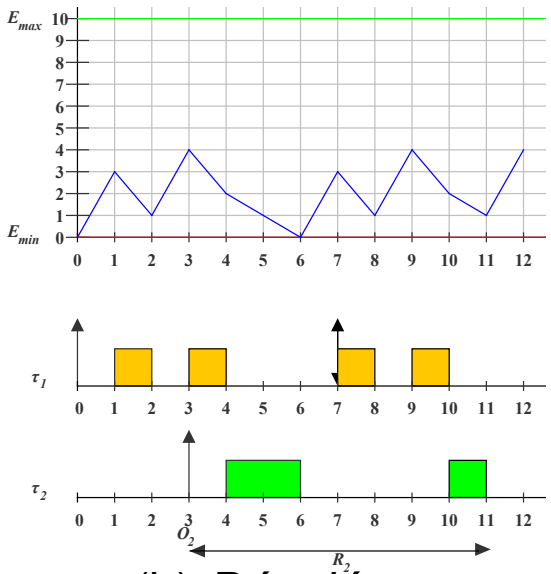
1. Pas d'hypothèses sur les premières dates d'activation et le niveau initial de la batterie

Le pire scénario

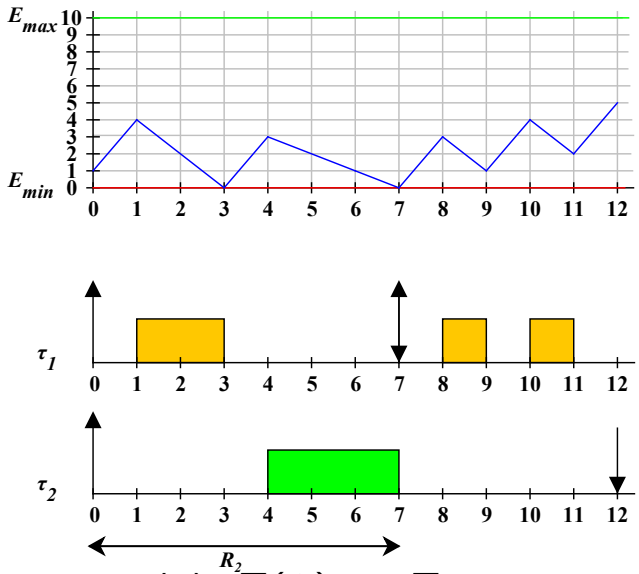
L'idée de la preuve : calculer le temps de réponse exact avec une formule et l'appliquer dans différents scénarios :



(a) Simultanées



(b) Décalées



(c) $E(0) > E_{min}$

$$w_i(t) = max\left(\left\lceil \frac{\sum_{j\leq i}\left\lceil \frac{t-O_j}{T_j}\right\rceil \times E_j - E(0)}{P_r}\right\rceil, \sum_{j\leq i}\left\lceil \frac{t-O_j}{T_j}\right\rceil \times C_j\right)$$

Optimalité

Théorème

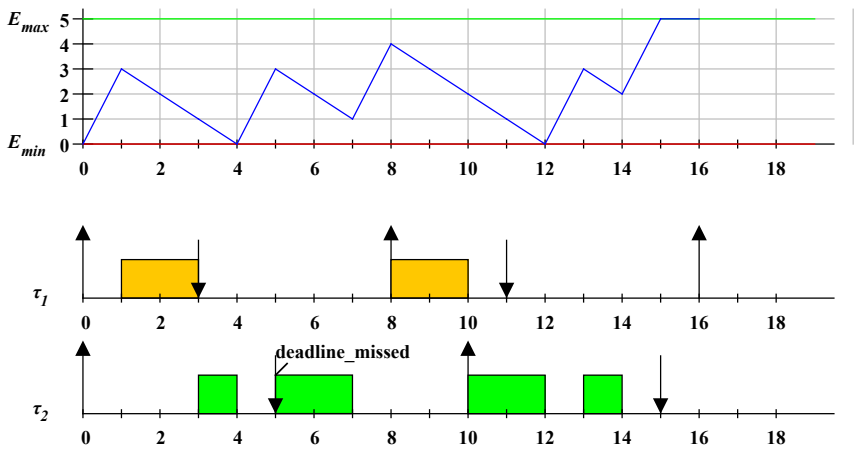
L'algorithme PFP_{ASAP} est optimal pour les systèmes de tâches non-concrets énergétiquement composés de tâches consommatrices seulement.

Optimalité

Théorème

L'algorithme PFP_{ASAP} est optimal pour les systèmes de tâches non-concrets énergétiquement composés de tâches consommatrices seulement.

L'idée de la preuve



Si $D_k \times P_r < \sum_{j \leq k} \left\lceil \frac{D_k}{T_j} \right\rceil \times E_j$, alors τ_k dépasse son échéance.

Un dépassement d'échéance se produit dans le pire cas seulement si l'énergie rechargée n'est pas suffisante pour satisfaire la demande d'énergie jusqu'à l'échéance dépassée.

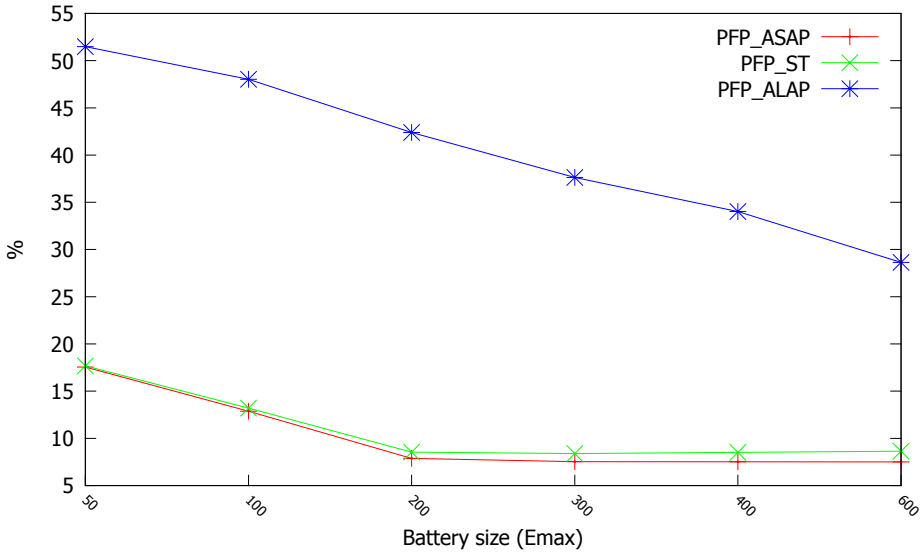
Condition d'ordonnançabilité nécessaire et suffisante

- Par une analyse de temps de réponse
- L'utilisation de la formule (1) dans le pire scénario

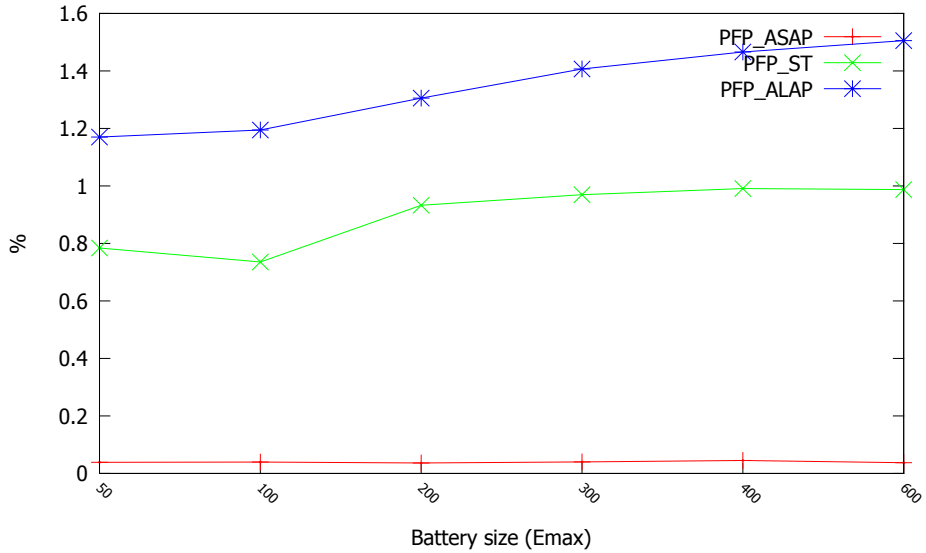
$$w_i(t) = max\left(\left\lceil \frac{\sum_{j\leq i}\left\lceil \frac{t-O_j}{T_j}\right\rceil \times E_j - E(0)}{P_r}\right\rceil, \sum_{j\leq i}\left\lceil \frac{t-O_j}{T_j}\right\rceil \times C_j\right) \quad (1)$$

- L'étude de la première instance est suffisante car le système est étudié dans le pire scénario et les échéances sont contraintes ou implicites

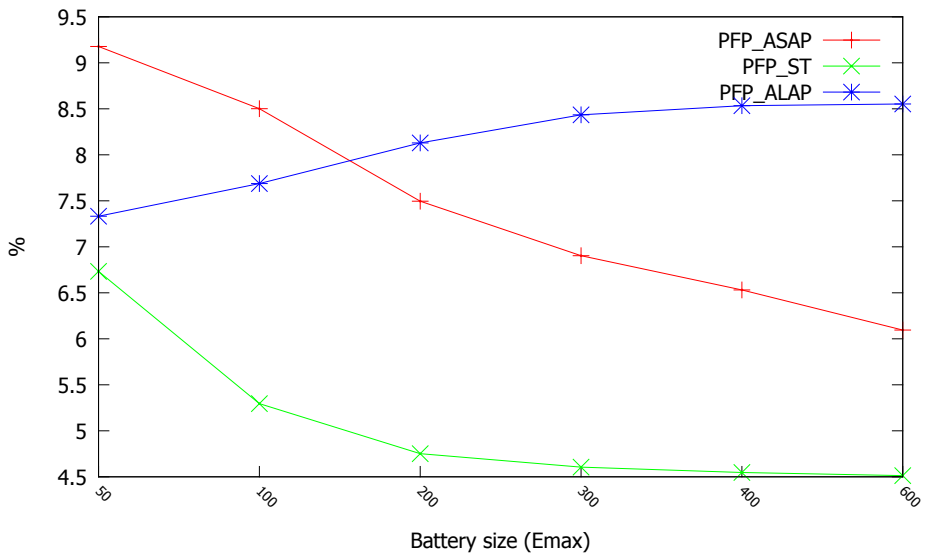
Évaluation



(a) Taux d'échec



(b) Coût de l'algorithme



(c) Nombre de préemptions

Publication

Ce travail est le fruit d'une collaboration avec Dr. Y. Abdeddaïm et a fait l'objet d'une publication dans la conférence ECRTS :

Yasmina Abdeddaïm, Younès Chandarli, and Damien Masson,
"The Optimality of PFP_{ASAP} Algorithm for Fixed-Priority Energy-Harvesting Real-Time Systems"

Plan

1. Introduction
2. Problématiques
3. État de l'art
4. Contributions
   - Systèmes non-concrets et tâches consommatrices : PFP_{ASAP}
   - Systèmes concrets et tâches consommatrices et génératrices : tests d'ordonnançabilité
   - Systèmes concrets et tâches consommatrices et génératrices : optimalité
   - Systèmes avec contraintes thermiques
   - L'outil de simulation YARTISS
5. Conclusion

Analyse d'ordonnançabilité avec approximation des temps de réponse

Ce qui change dans le modèle :

- un ensemble $\Gamma = \Gamma_c \cup \Gamma_g$ de tâches sporadiques à priorité fixe $\tau_i = (O_i, C_i, Pc_i, E_i, T_i, D_i, P_i)$ avec $D_i \leq T_i$:
  - Les tâches consommatrices : $\Gamma_c = \{\tau_i \in \Gamma, Pc_i > P_r\}$
  - Les tâches génératrices : $\Gamma_g = \{\tau_i \in \Gamma, 0 \leq Pc_i \leq P_r\}$

Objectif

Construire un test d'ordonnançabilité sous PFP_{ASAP} pour les systèmes de tâches composés de tâches consommatrices et de tâches génératrices d'énergie.

Analyse d'ordonnançabilité avec approximation des temps de réponse

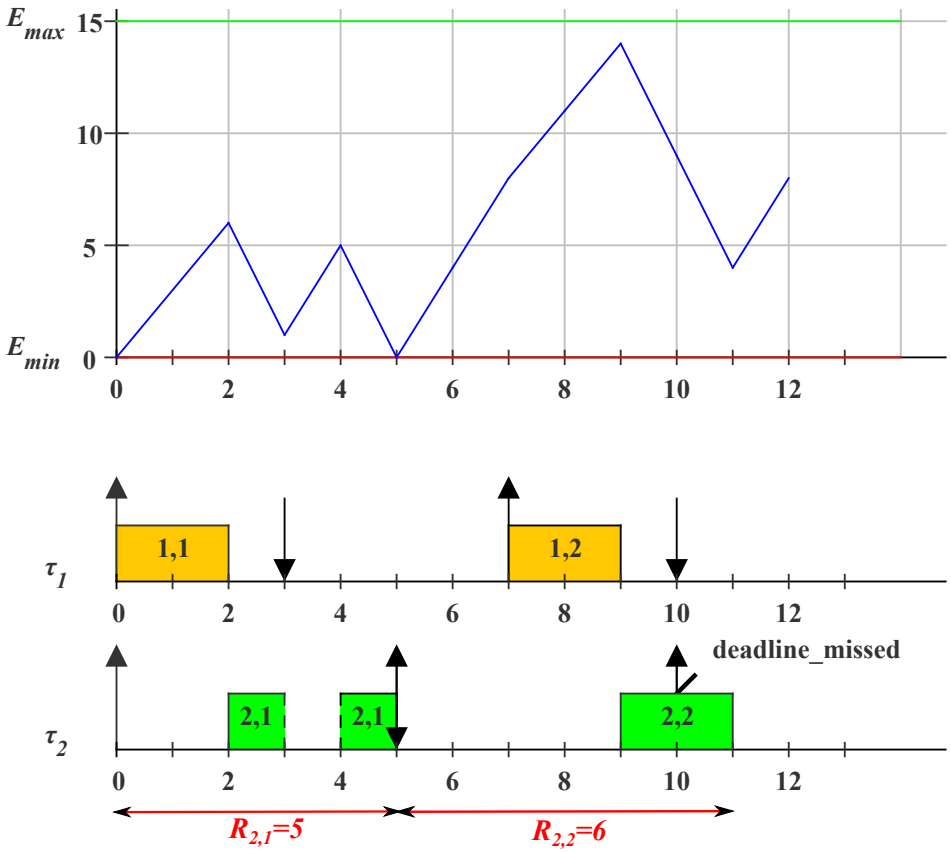
Ce qui change dans le modèle :

- un ensemble $\Gamma = \Gamma_c \cup \Gamma_g$ de tâches sporadiques à priorité fixe $\tau_i = (O_i, C_i, Pc_i, E_i, T_i, D_i, P_i)$ avec $D_i \leq T_i$:
  - Les tâches consommatrices : $\Gamma_c = \{\tau_i \in \Gamma, Pc_i > P_r\}$
  - Les tâches génératrices : $\Gamma_g = \{\tau_i \in \Gamma, 0 \leq Pc_i \leq P_r\}$

Objectif

Construire un test d'ordonnançabilité sous PFP_{ASAP} pour les systèmes de tâches composés de tâches consommatrices et de tâches génératrices d'énergie.

Analyse de temps de réponse sous PFP_{ASAP}



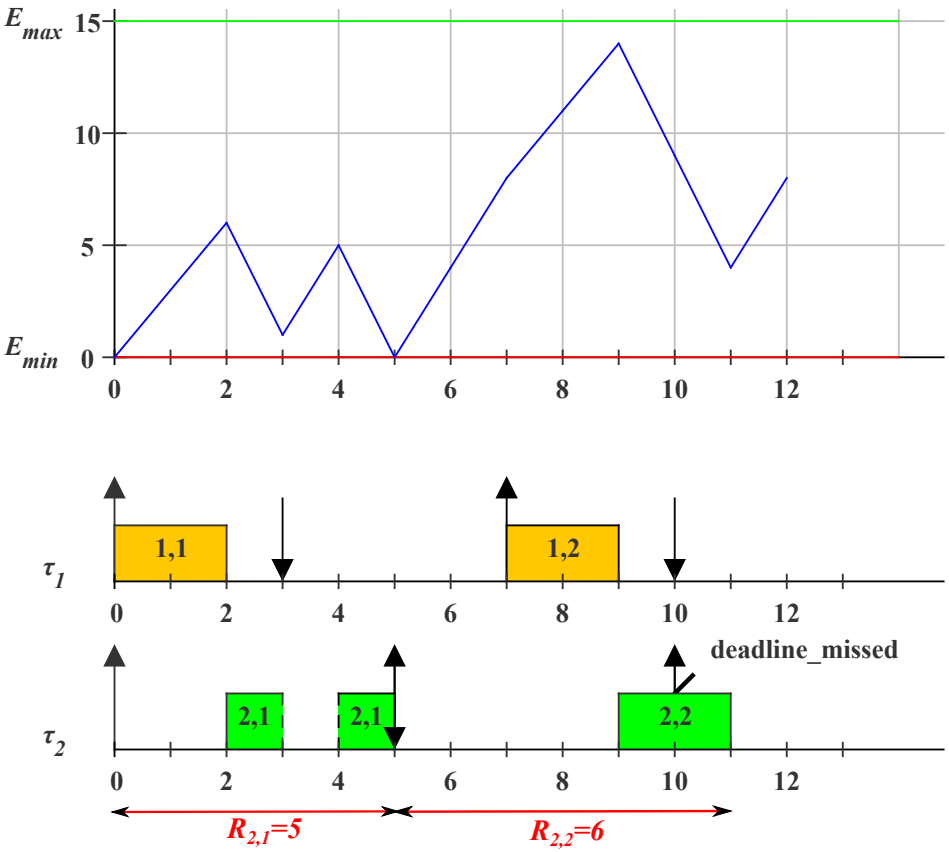
$P_r = 4$ $E_{max} = 15$

-	C_i	E_i	T_i	D_i	Pc_i	P_i
τ_1	2	2	7	3	1	1
τ_2	2	18	5	5	9	2

Analyse de temps de réponse sous PFP_{ASAP}

Le pire scénario

L'activation synchrone de toutes les tâches n'est plus le pire scénario.



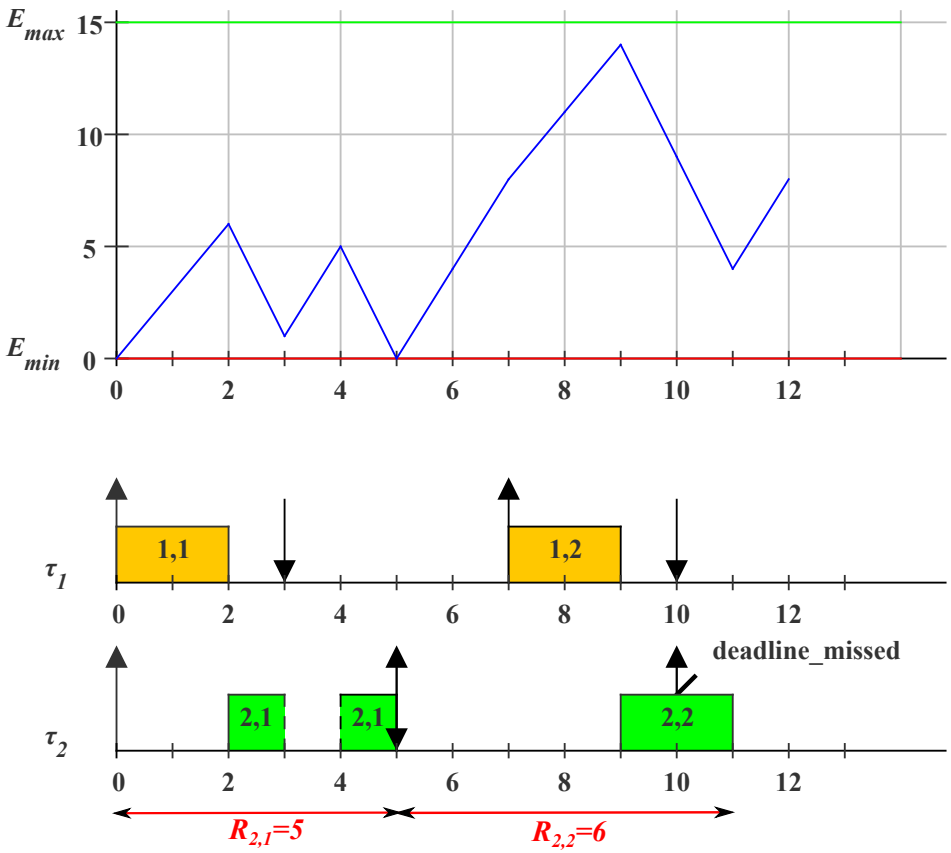
$P_r = 4$ $E_{max} = 15$

-	C_i	E_i	T_i	D_i	Pc_i	P_i
τ_1	2	2	7	3	1	1
τ_2	2	18	5	5	9	2

Analyse de temps de réponse sous PFP_{ASAP}

- Le pire scénario n'est pas connu
- Le test précédent devient seulement nécessaire

Borner par le haut R_i pour construire un test d'ordonnançabilité suffisant.



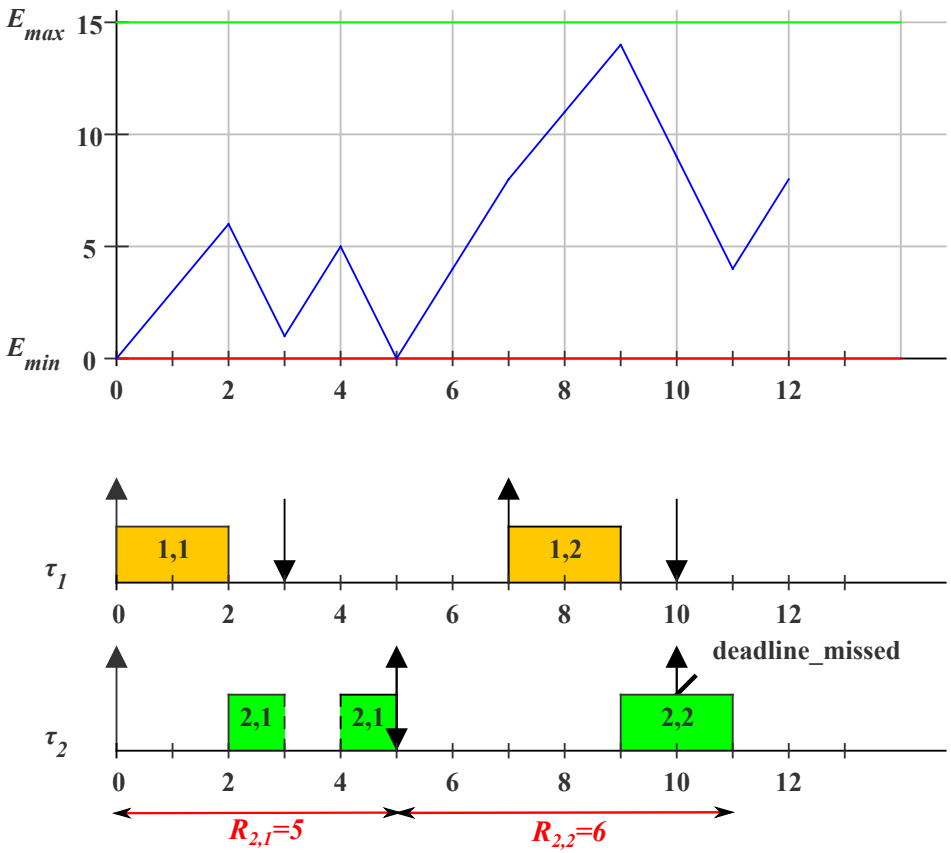
$P_r = 4$ $E_{max} = 15$

-	C_i	E_i	T_i	D_i	Pc_i	P_i
τ_1	2	2	7	3	1	1
τ_2	2	18	5	5	9	2

Analyse de temps de réponse sous PFP_{ASAP}

- Le pire scénario n'est pas connu
- Le test précédent devient seulement nécessaire

Borner par le haut R_i pour construire un test d'ordonnançabilité suffisant.



$P_r = 4$ $E_{max} = 15$

-	C_i	E_i	T_i	D_i	Pc_i	P_i
τ_1	2	2	7	3	1	1
τ_2	2	18	5	5	9	2

Borner le temps de réponse

- On a besoin d'une fonction monotone et croissante $F(i, w)$ qui borne par le haut le durée du pire temps de réponse dans un intervalle de longueur w ;
- La borne supérieure R_i^{UB} du pire temps de réponse R_i correspond au plus petit $w > 0$ qui satisfait $F(i, w) = w$.
- On définit pour chaque tâche τ_i un scénario virtuel qui :
  1. maximise les interférences des tâches plus prioritaires,
  2. maximise le nombre de périodes de rechargement requises.

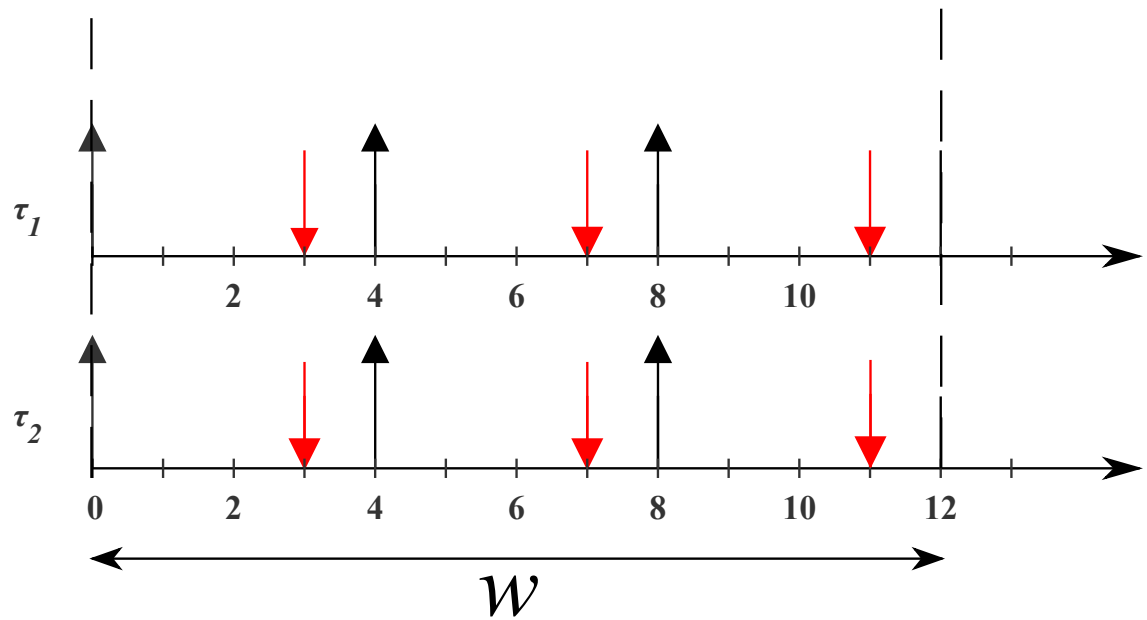
Borner le temps de réponse

- On a besoin d'une fonction monotone et croissante $F(i, w)$ qui borne par le haut le durée du pire temps de réponse dans un intervalle de longueur w ;
- La borne supérieure R_i^{UB} du pire temps de réponse R_i correspond au plus petit $w > 0$ qui satisfait $F(i, w) = w$.
- On définit pour chaque tâche τ_i un scénario virtuel qui :
  1. maximise les interférences des tâches plus prioritaires,
  2. maximise le nombre de périodes de rechargement requises.

Maximiser les interférences

Pour chaque tâche τ_i activée dans une fenêtre de taille w, le nombre maximal d'instances de tâches plus prioritaires qui sont activées dans cette fenêtre est :

$$\sum_{j<i} \left\lceil \frac{w}{T_j} \right\rceil = \sum_{j<i, \tau_j \in \Gamma_c} \left\lceil \frac{w}{T_j} \right\rceil + \sum_{j<i, \tau_j \in \Gamma_g} \left\lceil \frac{w}{T_j} \right\rceil$$

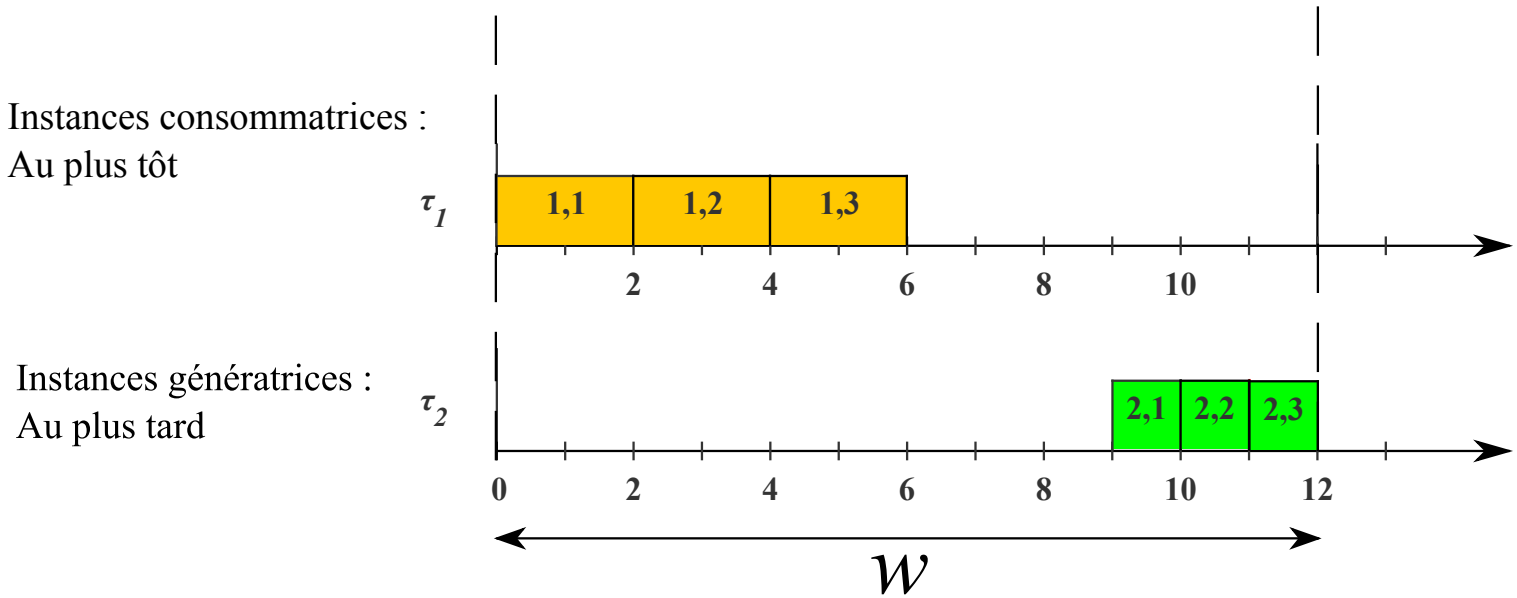


Maximiser les rechargement

Pour borner par le haut le temps de rechargement dans une fenêtre de taille w, on considère un scénario virtuel où :

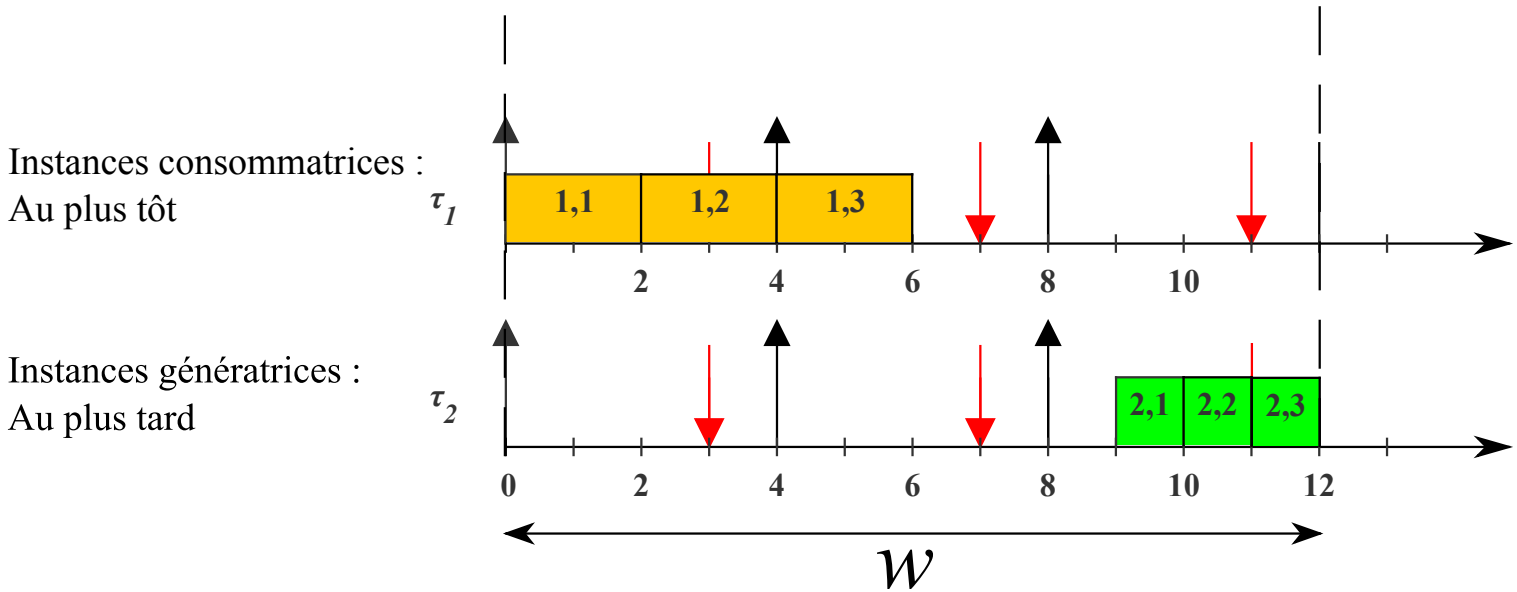
1. La batterie est au seuil minimal E_{min} au début de la fenêtre,
   - pour minimiser le budget énergie de l'intervalle w.
2. Toutes les instances consommatrices sont exécutées avant toutes les instances génératrices d'énergie.
   - pour maximiser les temps de rechargement

Borne supérieure R_i^{UB1}



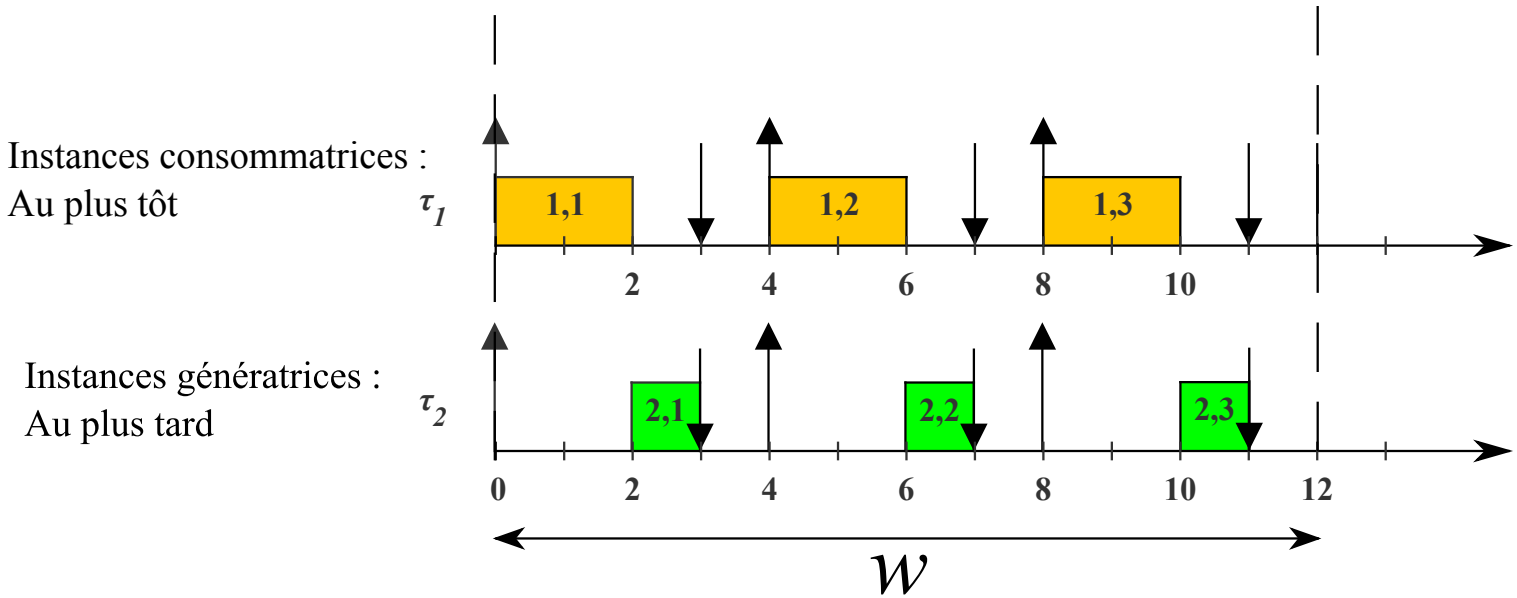
$$F^{UB1}(i,w) = \underbrace{\left\lceil \sum_{j \le i, \tau_j \in \Gamma_c} \left\lceil \frac{w}{T_j} \right\rceil \times ((Pc_j - P_r) \times C_j / P_r) \right\rceil}_{\text{le rechargement maximal}} + \underbrace{\sum_{j \le i, \tau_j \in \Gamma_c} \left\lceil \frac{w}{T_j} \right\rceil \times C_j}_{\text{tâches consommatrices}} + \underbrace{\sum_{j \le i, \tau_j \in \Gamma_g} \left\lceil \frac{w}{T_j} \right\rceil \times C_j}_{\text{tâches génératrices}}$$

Une borne supérieure plus fine R_i^{UB2}



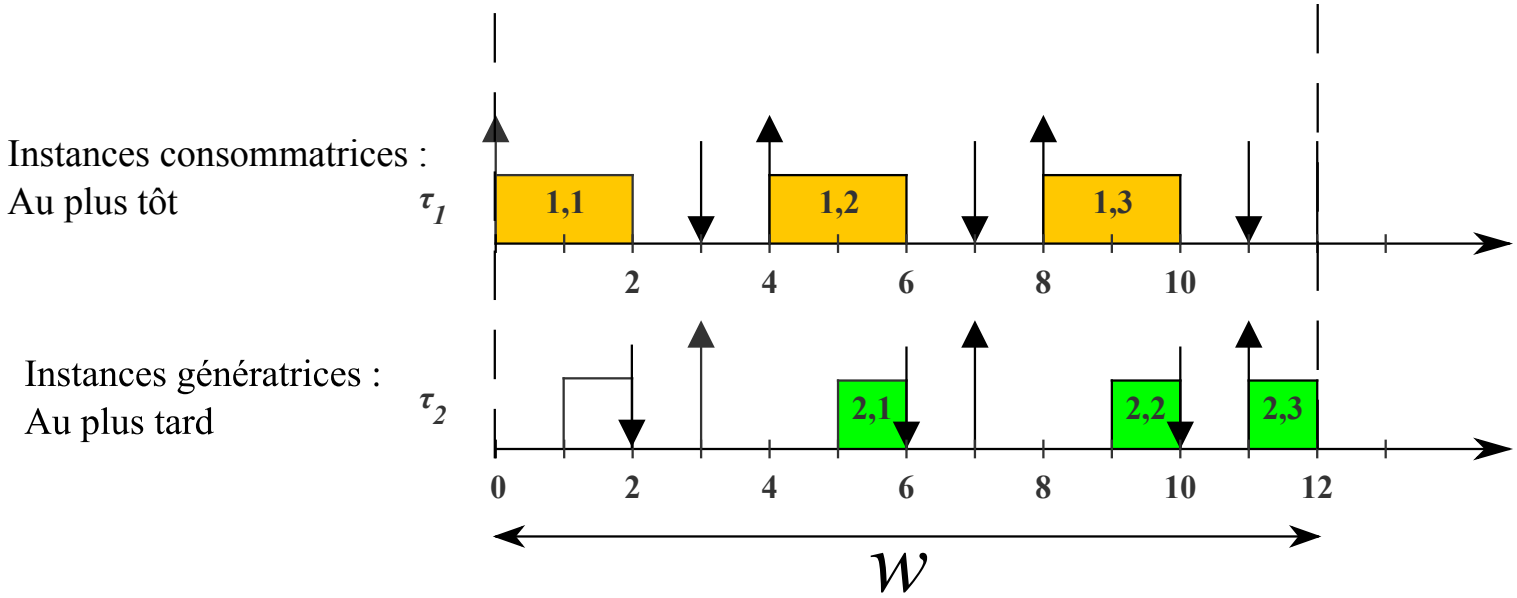
- Les instances consommatrices sont exécutées au plus tôt
- Les instances génératrices sont exécutées au plus tard

Une borne supérieure plus fine R_i^{UB2}



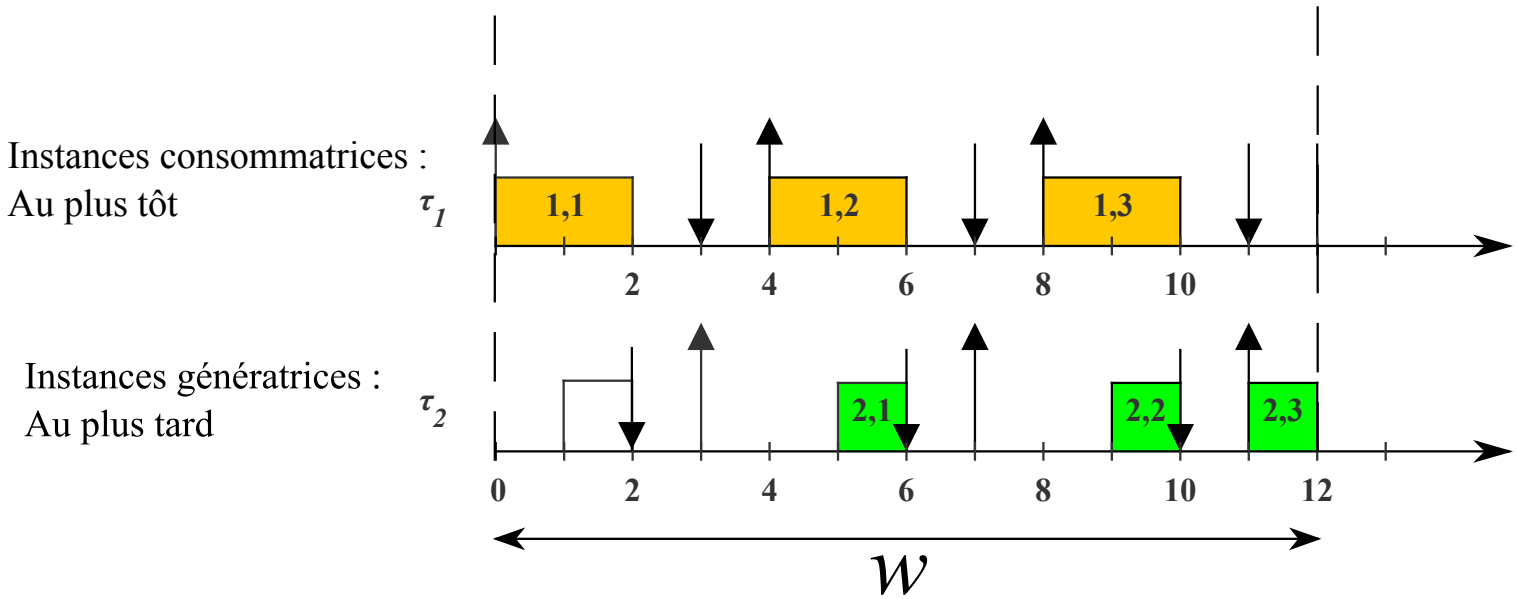
- Les instances consommatrices sont exécutées au plus tôt
- Les instances génératrices sont exécutées au plus tard

Une borne supérieure plus fine R_i^{UB2}



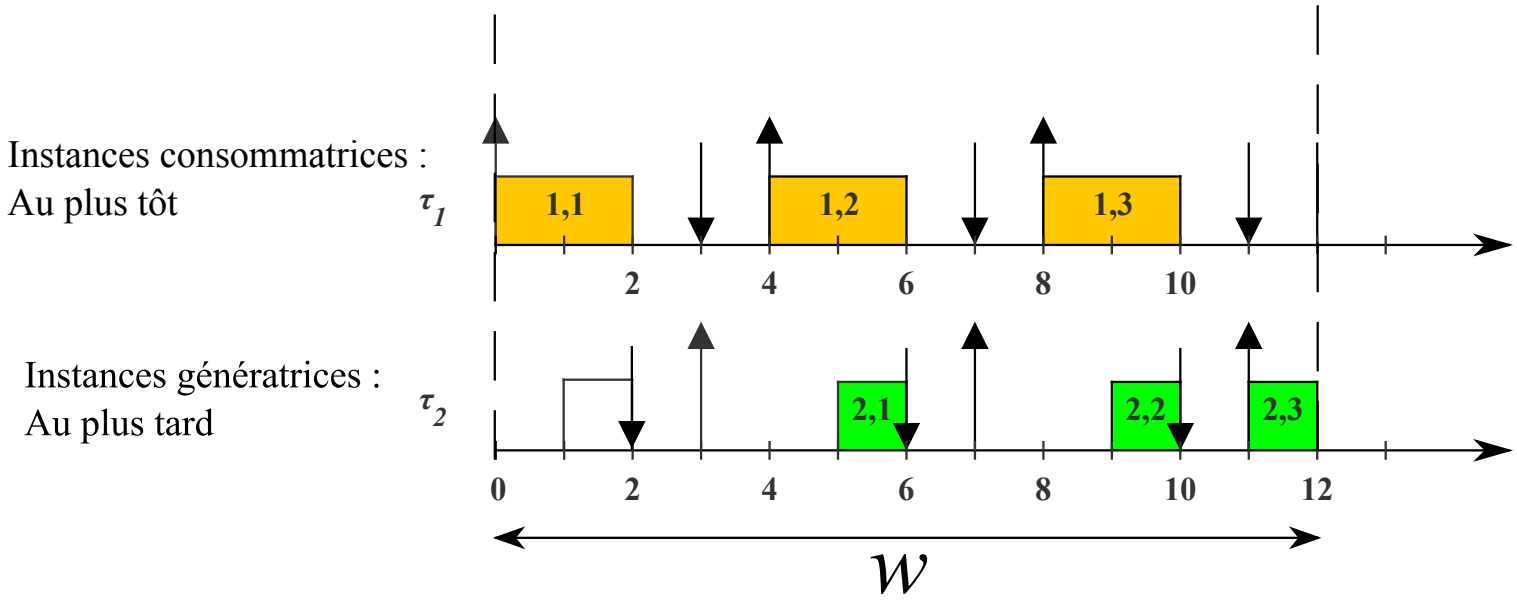
- Les instances consommatrices sont exécutées au plus tôt
- Les instances génératrices sont exécutées au plus tard

Test d'ordonnançabilité suffisant *UB2*



- Pour calculer $F^{UB2}(i, w)$, on calcule le pire temps de réponse de la séquence virtuelle :

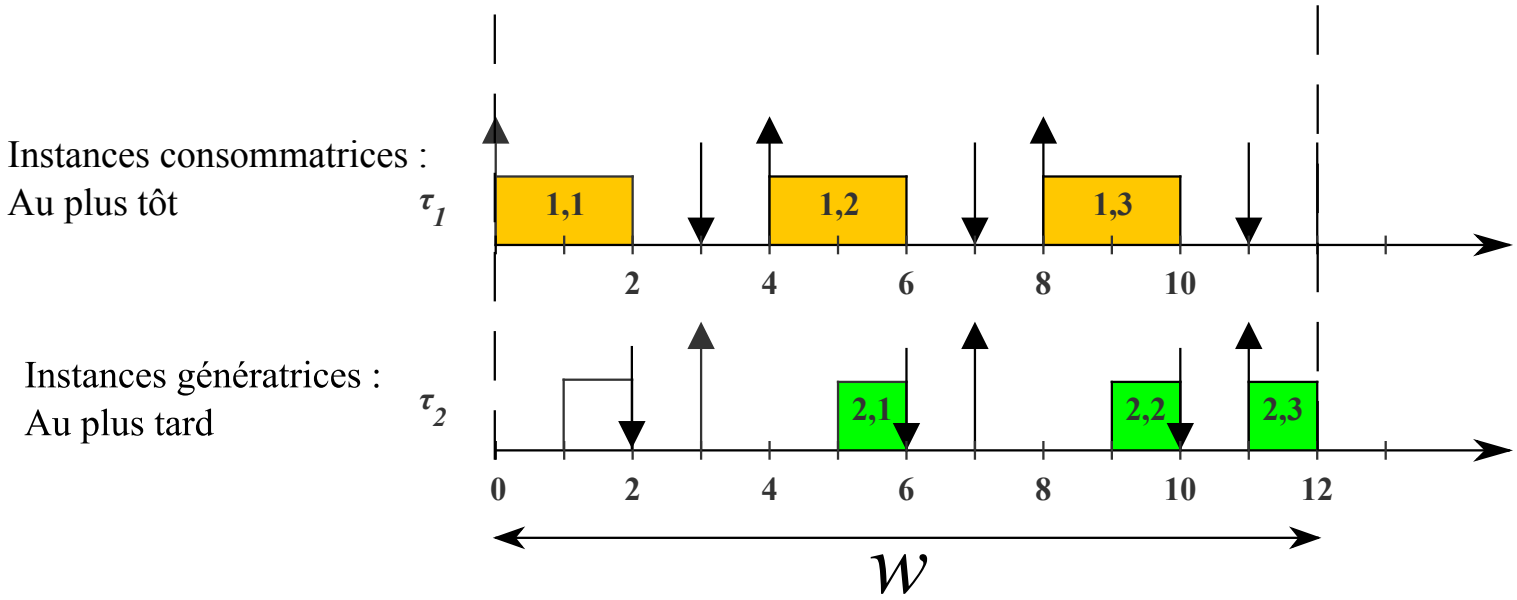
Test d'ordonnançabilité suffisant *UB2*



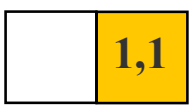
- Pour calculer $F^{UB2}(i, w)$, on calcule le pire temps de réponse de la séquence virtuelle :

□

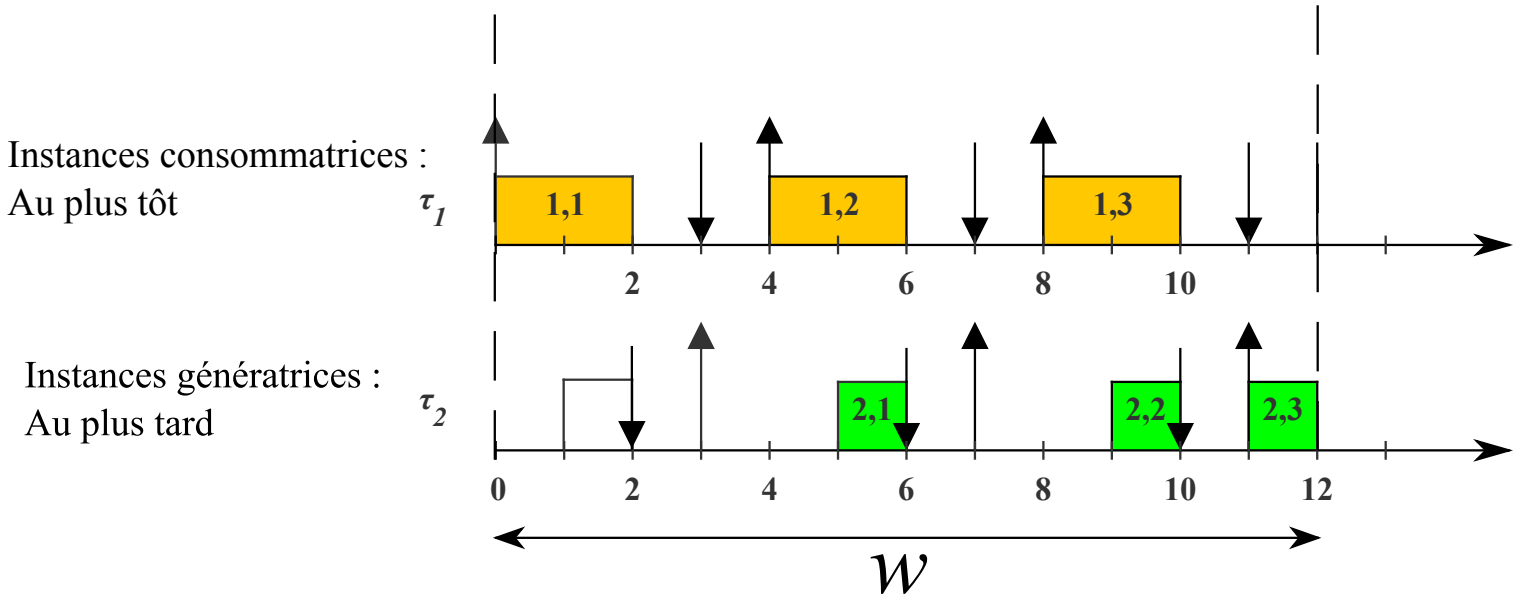
Test d'ordonnançabilité suffisant *UB2*



- Pour calculer $F^{UB2}(i, w)$, on calcule le pire temps de réponse de la séquence virtuelle :

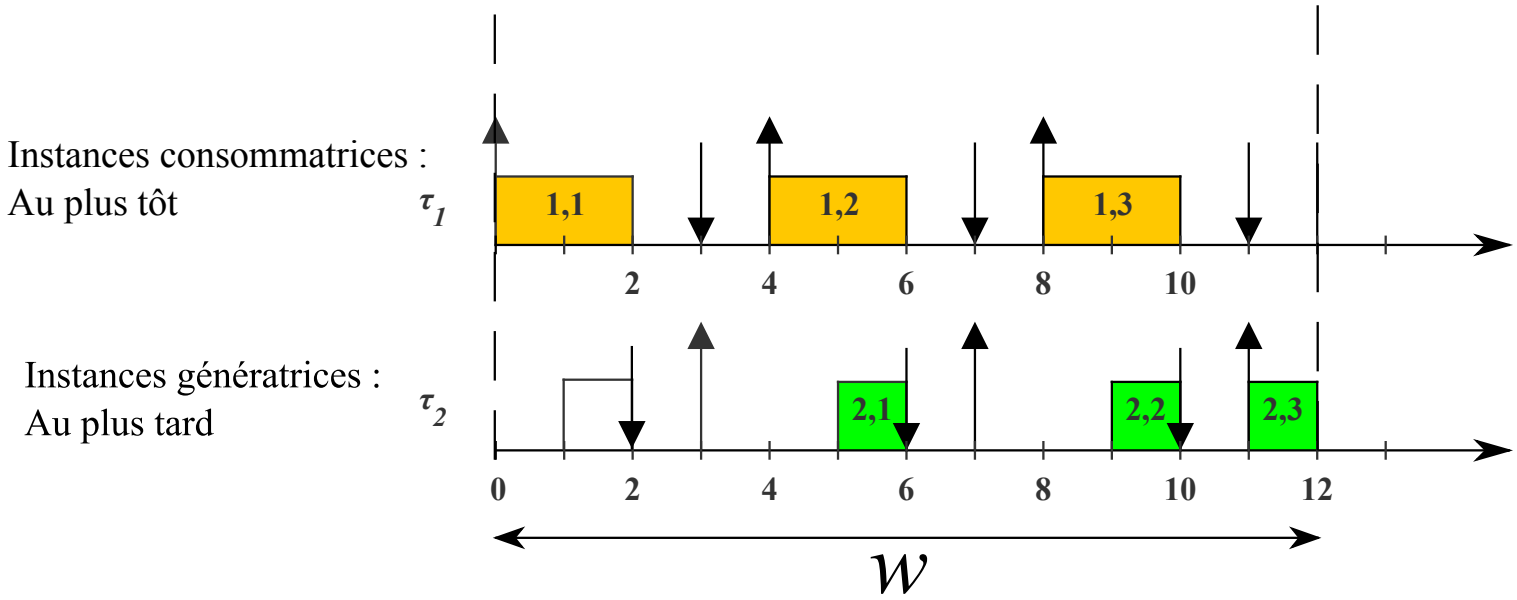


Test d'ordonnançabilité suffisant *UB2*



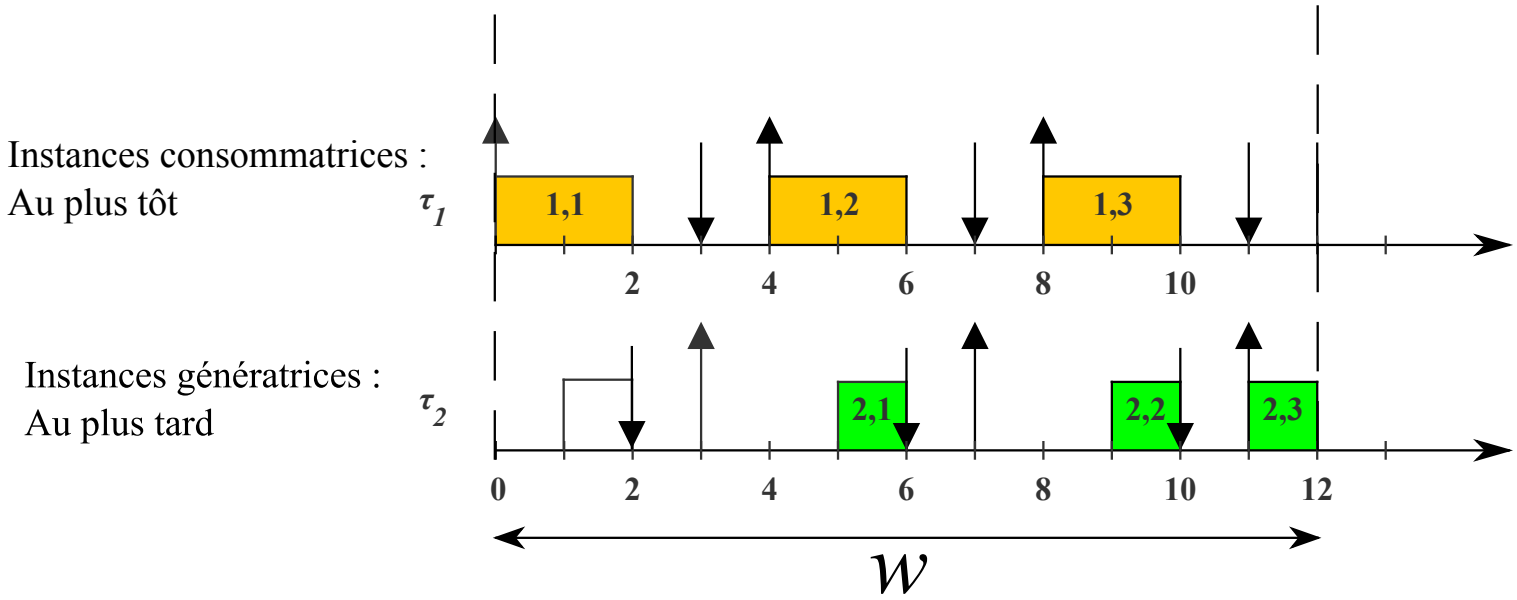
- Pour calculer $F^{UB2}(i, w)$, on calcule le pire temps de réponse de la séquence virtuelle :

Test d'ordonnançabilité suffisant *UB2*

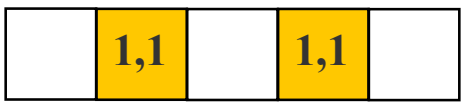


- Pour calculer $F^{UB2}(i, w)$, on calcule le pire temps de réponse de la séquence virtuelle :

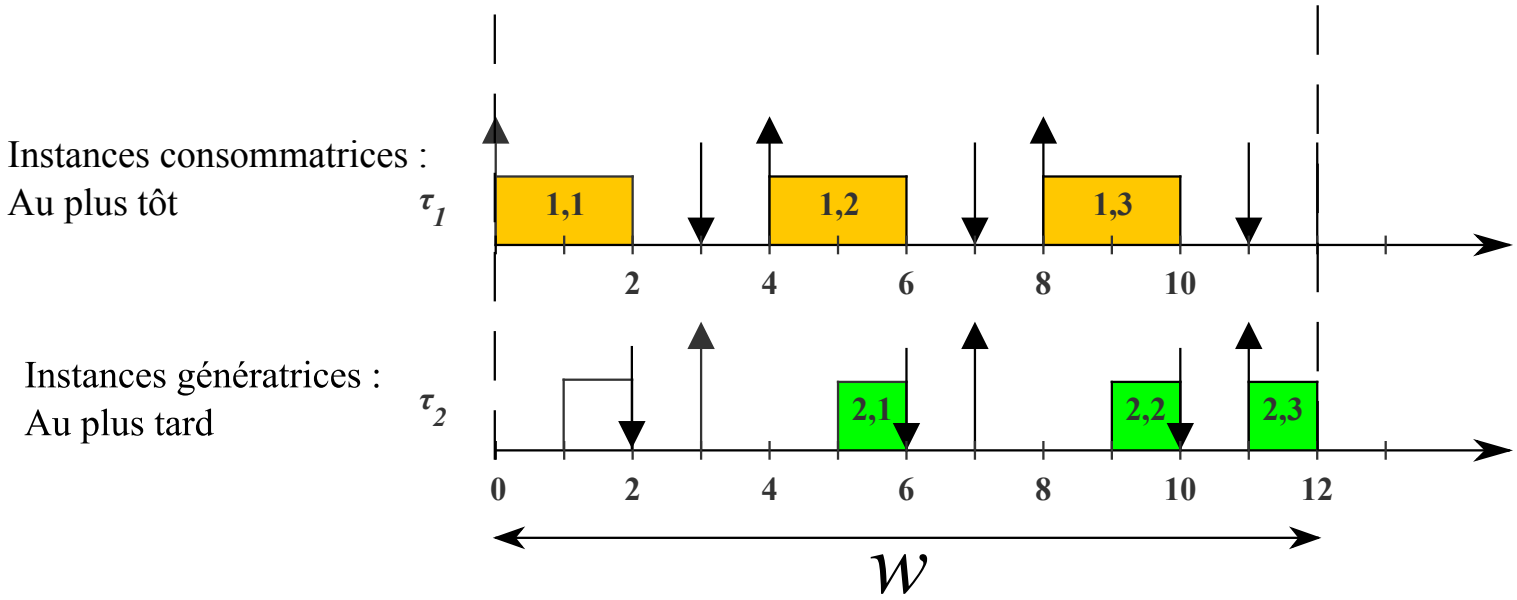
Test d'ordonnançabilité suffisant *UB2*



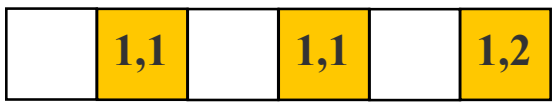
- Pour calculer $F^{UB2}(i, w)$, on calcule le pire temps de réponse de la séquence virtuelle :



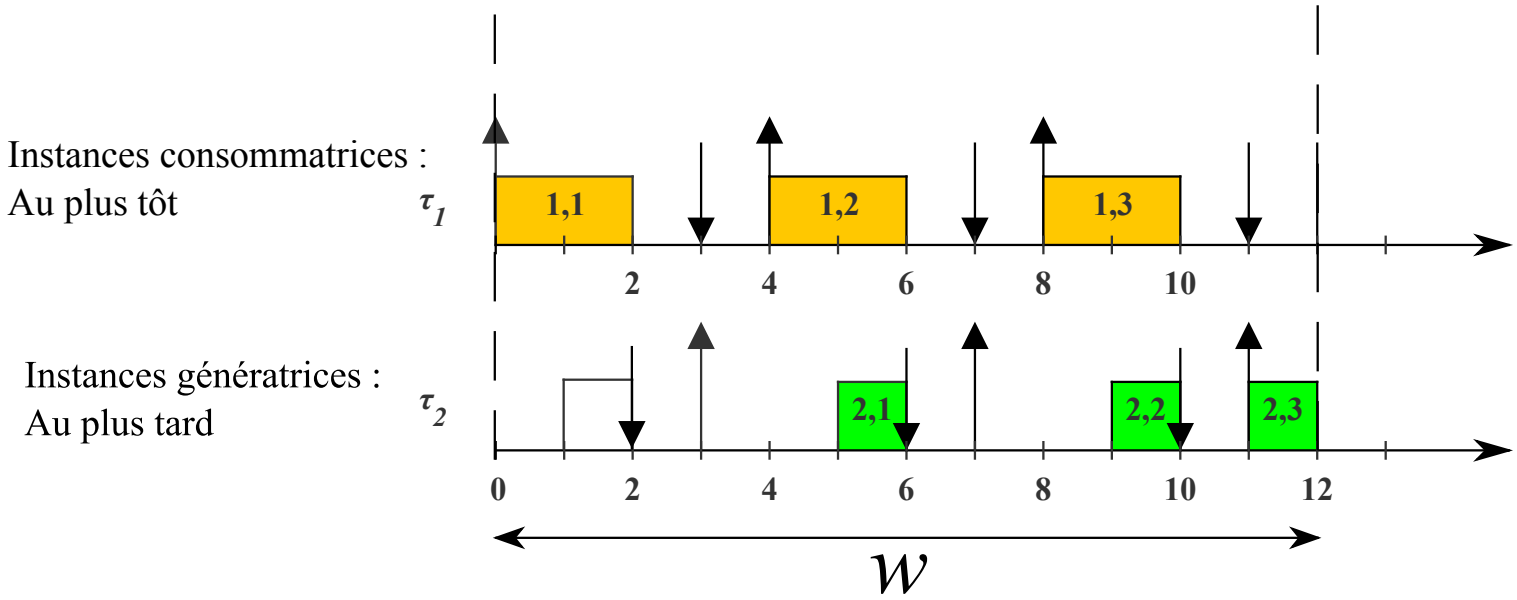
Test d'ordonnançabilité suffisant *UB2*



- Pour calculer $F^{UB2}(i, w)$, on calcule le pire temps de réponse de la séquence virtuelle :

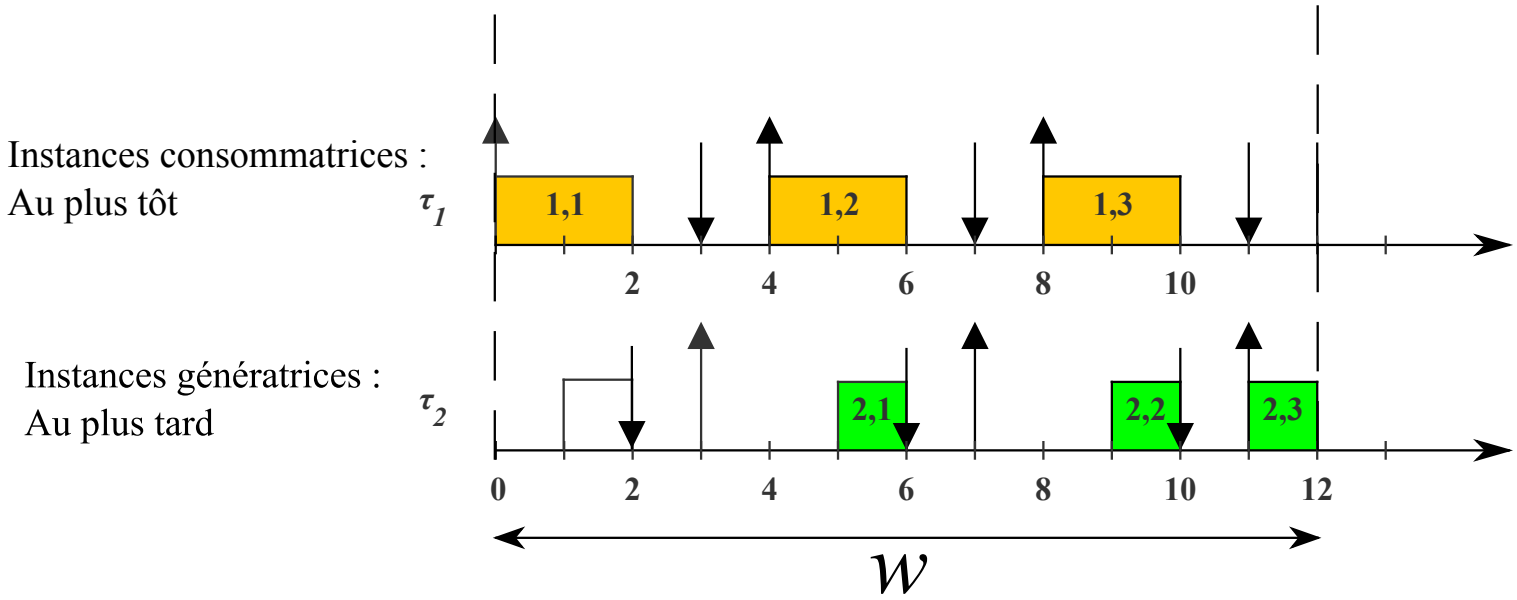


Test d'ordonnançabilité suffisant *UB2*



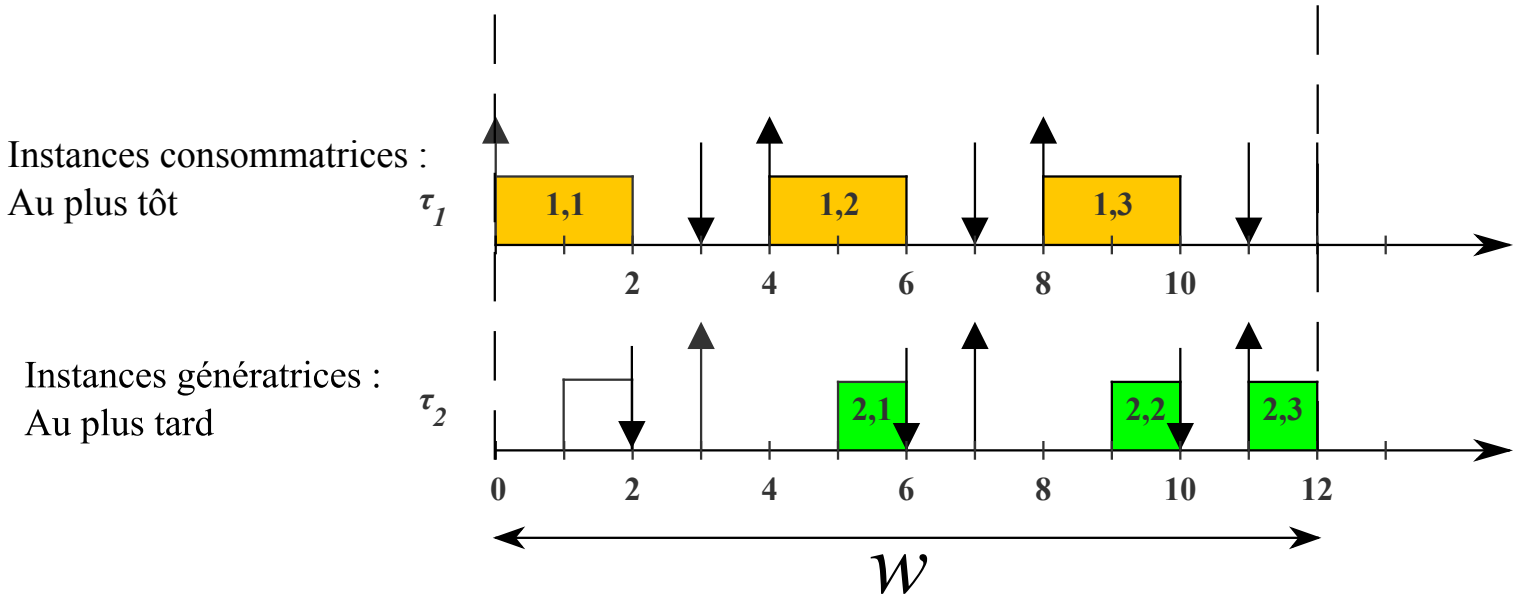
- Pour calculer $F^{UB2}(i, w)$, on calcule le pire temps de réponse de la séquence virtuelle :

Test d'ordonnançabilité suffisant *UB2*

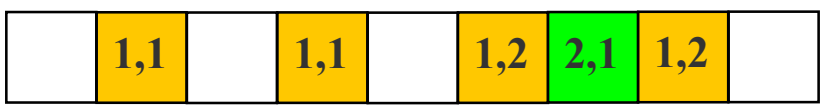


- Pour calculer $F^{UB2}(i, w)$, on calcule le pire temps de réponse de la séquence virtuelle :

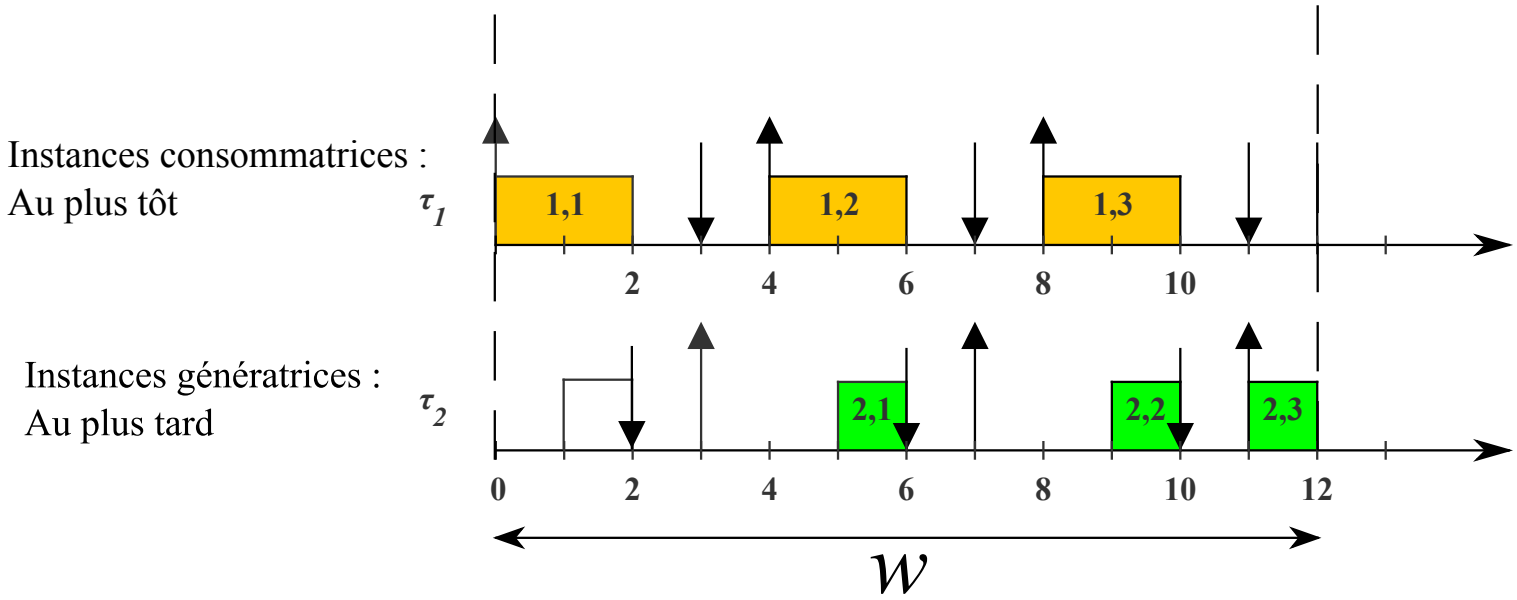
Test d'ordonnançabilité suffisant *UB2*



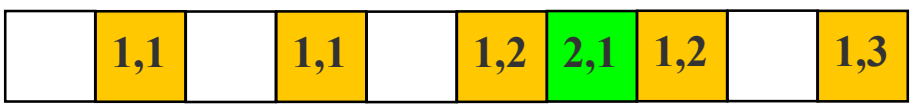
- Pour calculer $F^{UB2}(i, w)$, on calcule le pire temps de réponse de la séquence virtuelle :



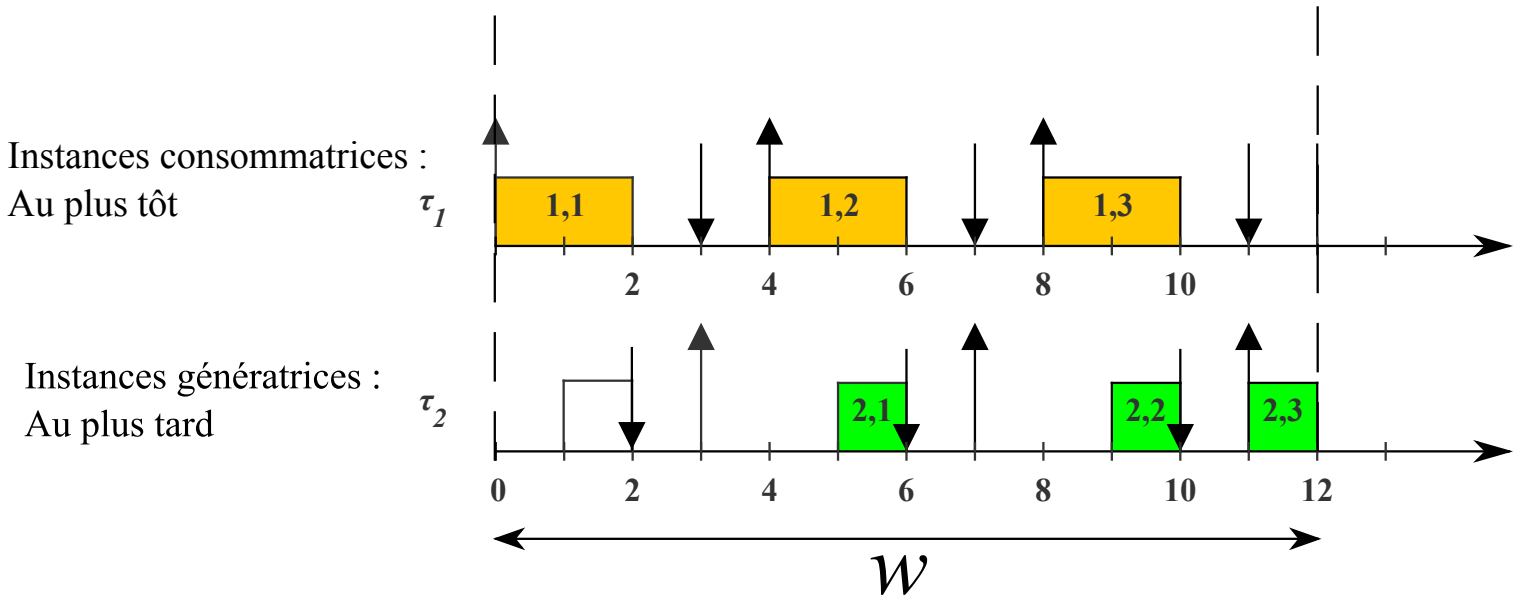
Test d'ordonnançabilité suffisant *UB2*



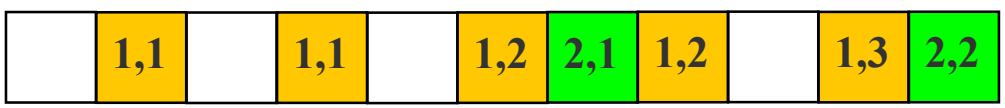
- Pour calculer $F^{UB2}(i, w)$, on calcule le pire temps de réponse de la séquence virtuelle :



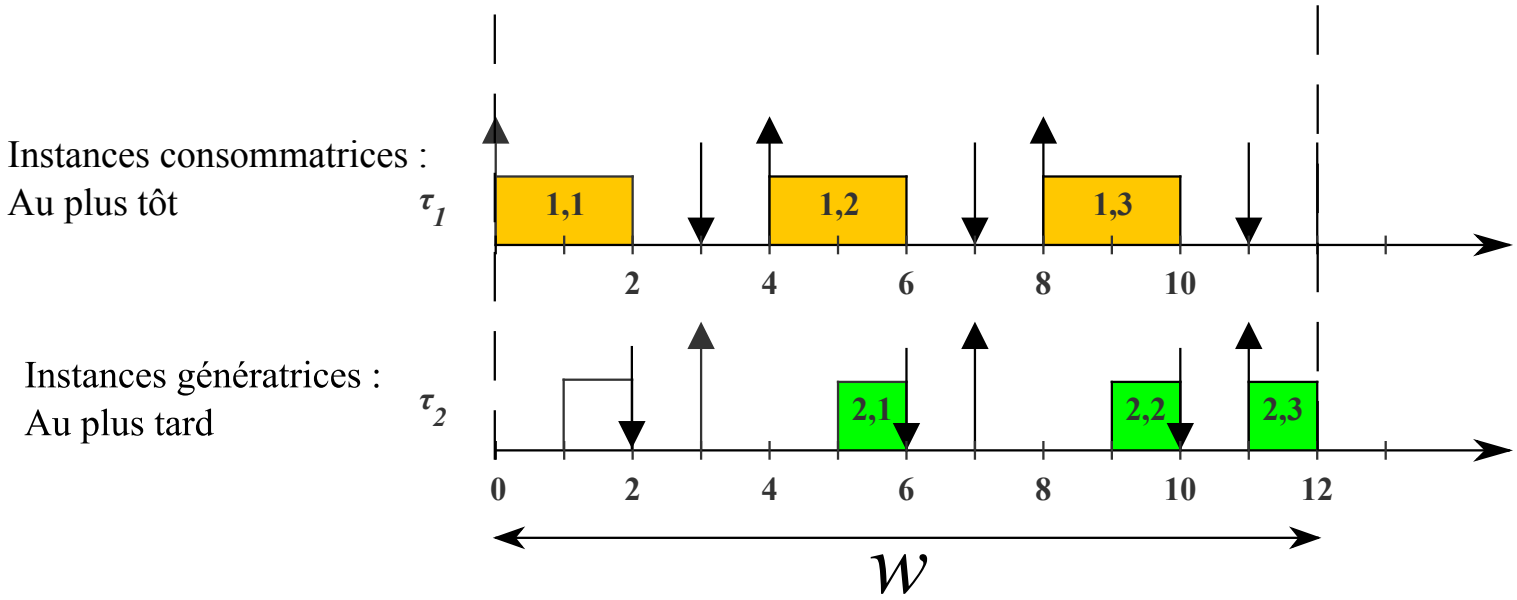
Test d'ordonnançabilité suffisant *UB2*



- Pour calculer $F^{UB2}(i, w)$, on calcule le pire temps de réponse de la séquence virtuelle :

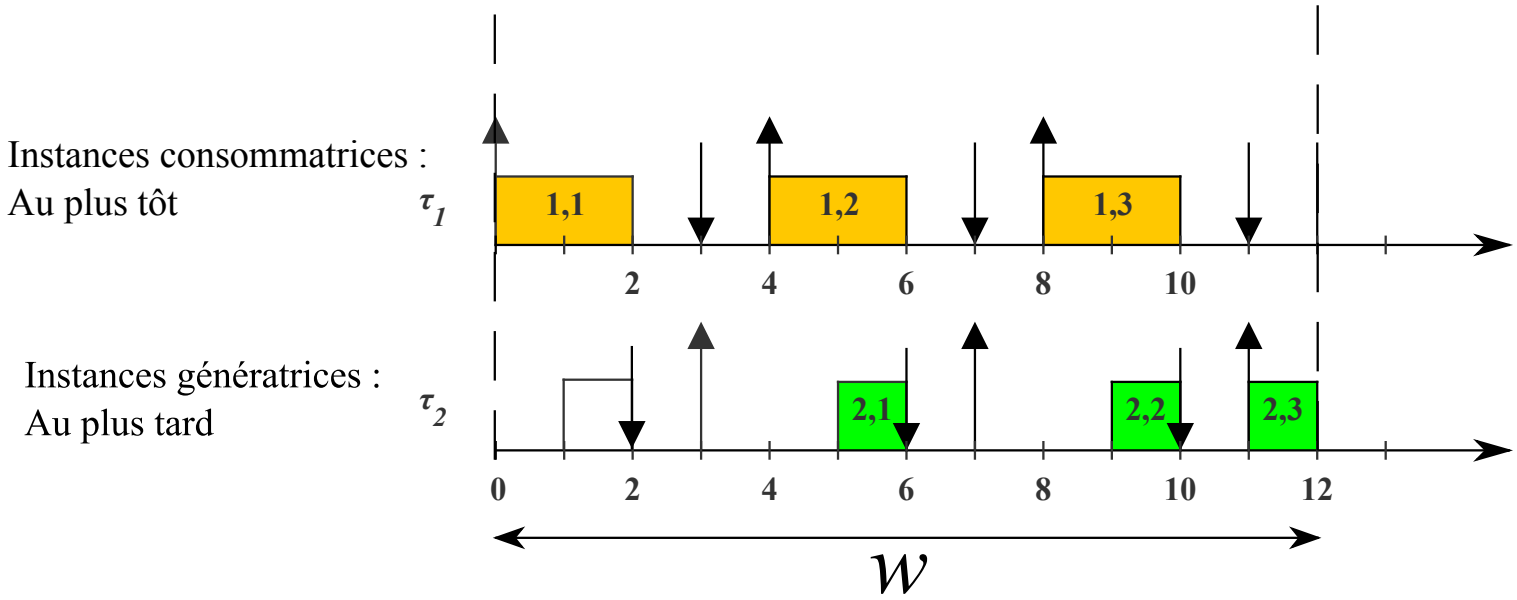


Test d'ordonnançabilité suffisant *UB2*



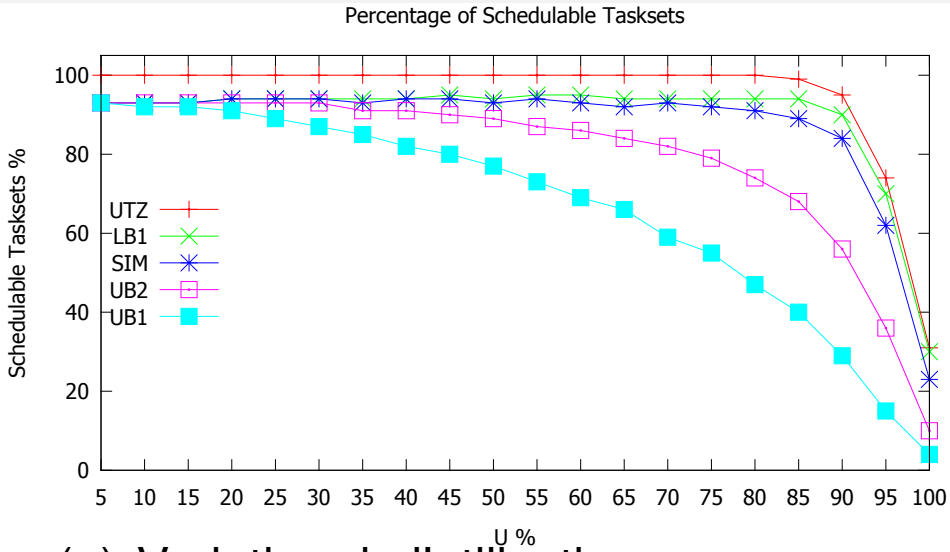
- Pour calculer $F^{UB2}(i, w)$, on calcule le pire temps de réponse de la séquence virtuelle :

Test d'ordonnançabilité suffisant *UB2*

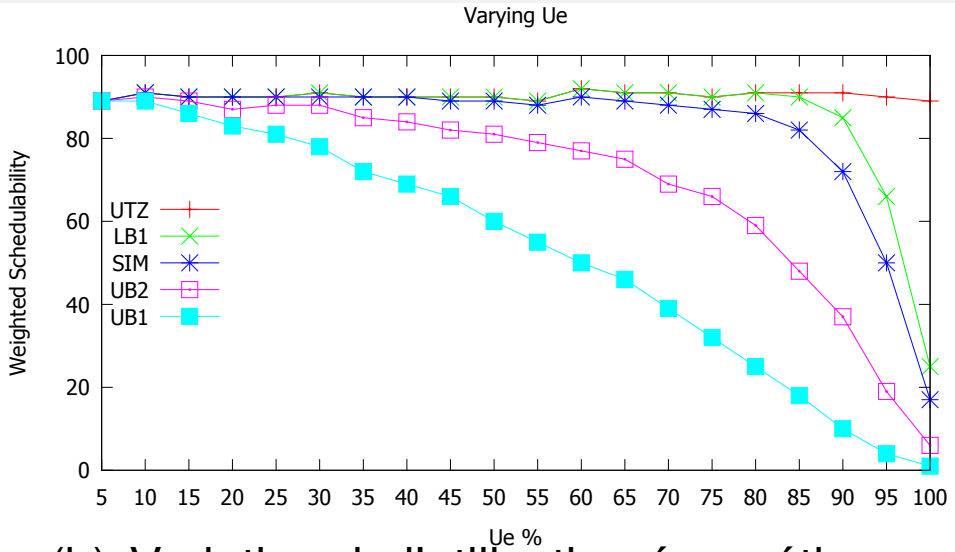


- Pour calculer $F^{UB2}(i, w)$, on calcule le pire temps de réponse de la séquence virtuelle :

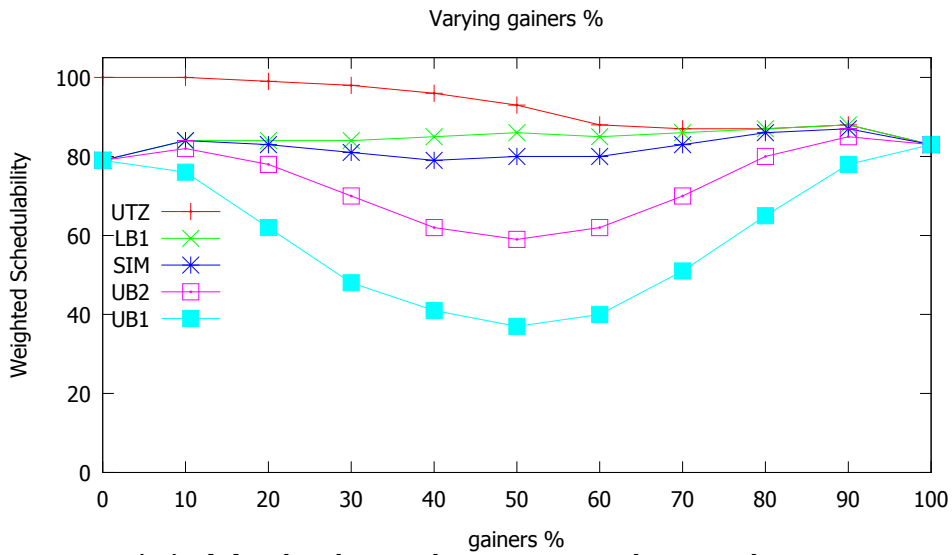
Évaluation



(a) Variation de l'utilisation processeur



(b) Variation de l'utilisation énergétique



(c) Variation de types de taches

Publication

Ce travail est le fruit d'une collaboration avec Prof. R. Davis et Dr. Y. Abdeddaïm et a fait l'objet d'une publication dans la conférence RTNS :

Yasmina Abdeddaïm, Younès Chandarli, R.I. Davis and Damien Masson,
"Schedulability Analysis for Fixed Priority Real-Time Systems with Energy-Harvesting"

RTNS

2014

Prix du meilleur article de la conférence

Une version étendue sera soumise à *RTS Journal*

Plan

1. Introduction
2. Problématiques
3. État de l'art
4. Contributions
   - Systèmes non-concrets et tâches consommatrices : PFP_{ASAP}
   - Systèmes concrets et tâches consommatrices et génératrices : tests d'ordonnançabilité
   - Systèmes concrets et tâches consommatrices et génératrices : optimalité
   - Systèmes avec contraintes thermiques
   - L'outil de simulation YARTISS
5. Conclusion

Recherche d'algorithme optimal

Nous avons exploré plusieurs idées d'algorithme :

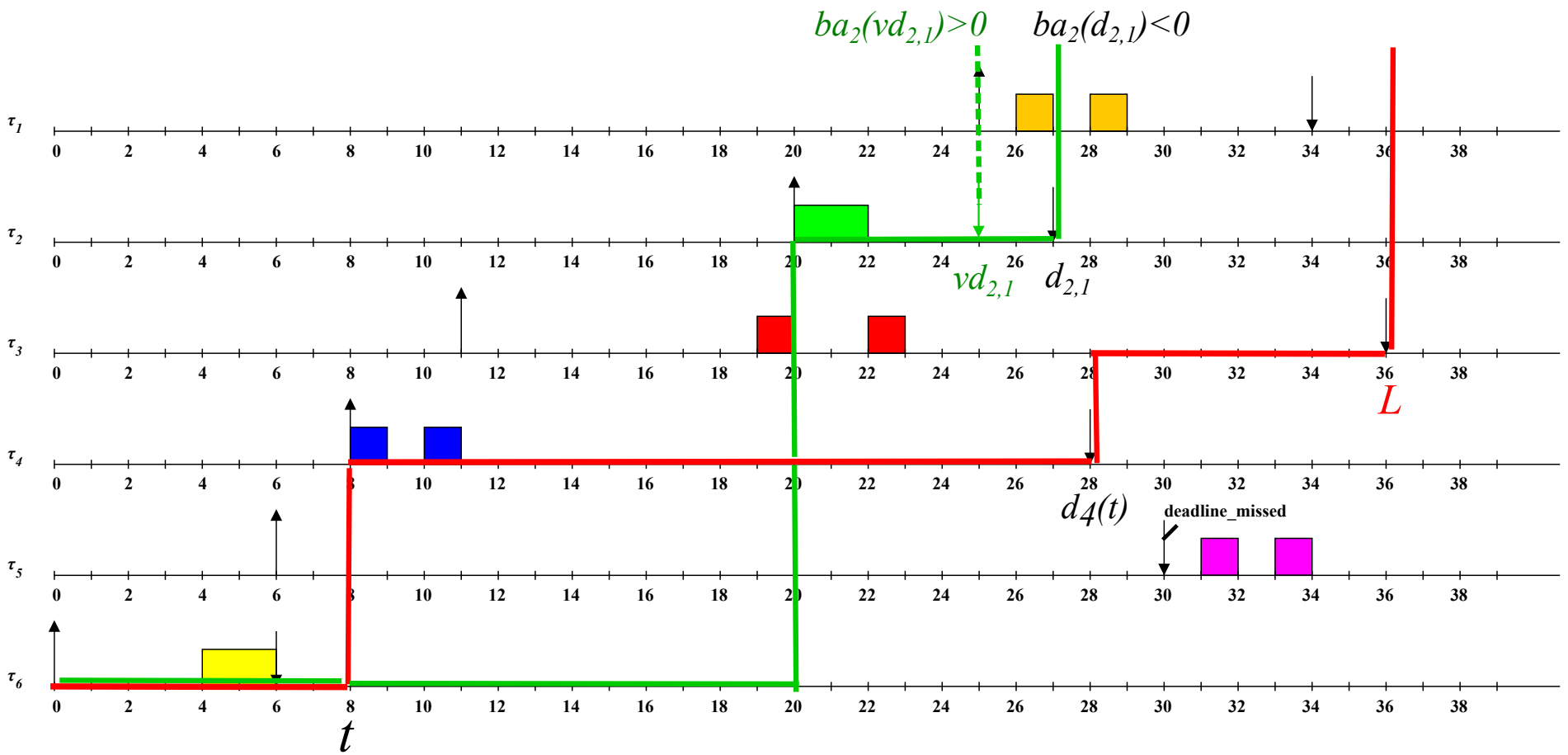
- FPC_{ASAP}
- $FPLSA$
- $FPLeg$
- FP_{lh}

Aucun de ces algorithmes n'est optimal avec une complexité raisonnable :

Conjecture

La problématique de l'ordonnancement temps réel des systèmes collecteurs d'énergie à priorité fixe est un problème NP-difficile.

La difficulté de trouver un algorithme optimal



Plan

1. Introduction
2. Problématiques
3. État de l'art
4. Contributions
   - Systèmes non-concrets et tâches consommatrices : PFP_{ASAP}
   - Systèmes concrets et tâches consommatrices et génératrices : tests d'ordonnançabilité
   - Systèmes concrets et tâches consommatrices et génératrices : optimalité
   - Systèmes avec contraintes thermiques
   - L'outil de simulation YARTISS
5. Conclusion

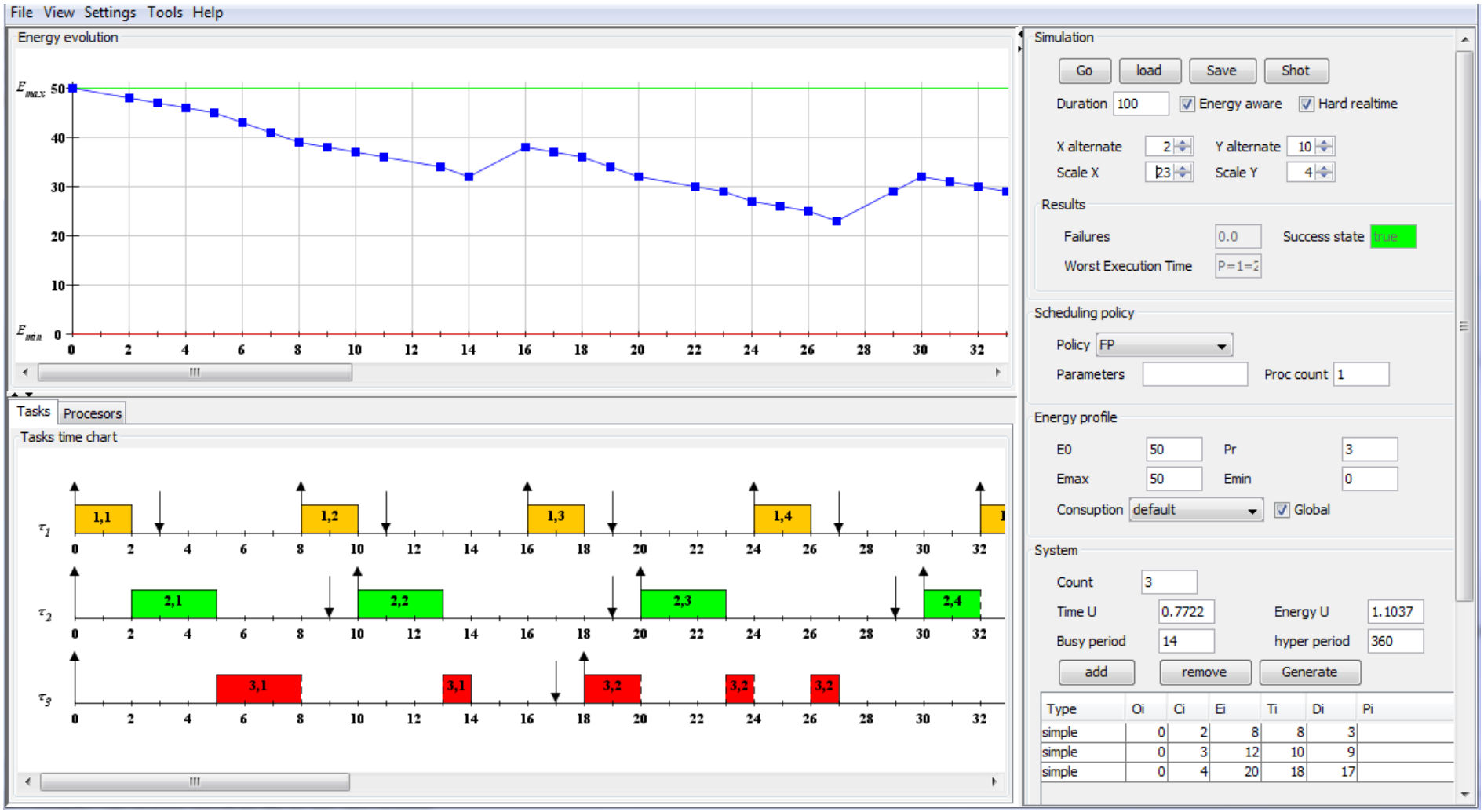
Systèmes avec contraintes thermiques

- La température ne doit jamais dépasser un seuil maximal T_{max}
- T_{max} est l'équivalent de E_{min}
- La problématique d'ordonnancement temps réel présente des similitudes avec celle des systèmes collecteurs d'énergie
- Ce travail a été entamé dans le cadre d'une collaboration avec Prof. N. Fisher

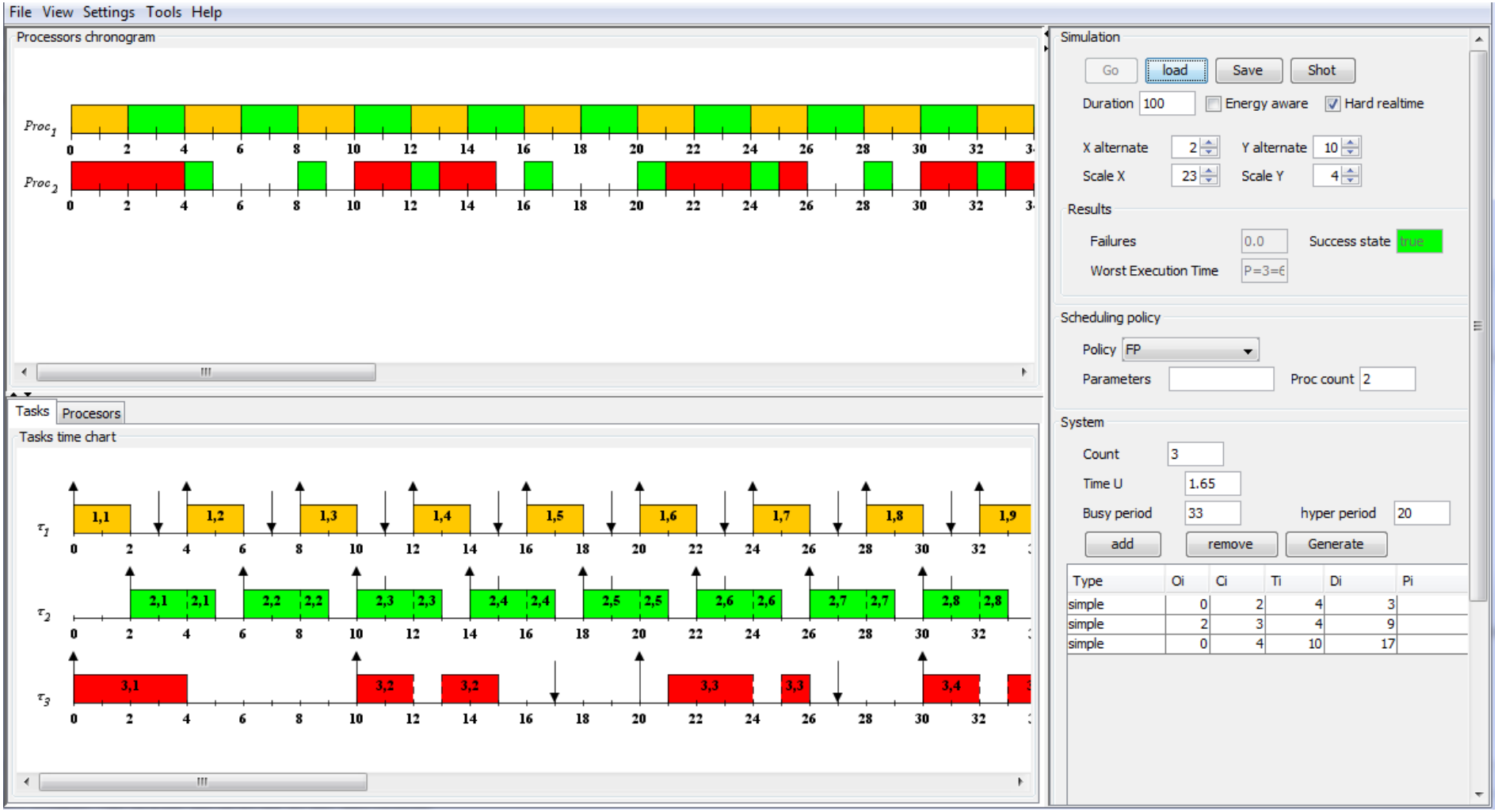
Plan

1. Introduction
2. Problématiques
3. État de l'art
4. Contributions
   - Systèmes non-concrets et tâches consommatrices : PFP_{ASAP}
   - Systèmes concrets et tâches consommatrices et génératrices : tests d'ordonnançabilité
   - Systèmes concrets et tâches consommatrices et génératrices : optimalité
   - Systèmes avec contraintes thermiques
   - L'outil de simulation YARTISS
5. Conclusion

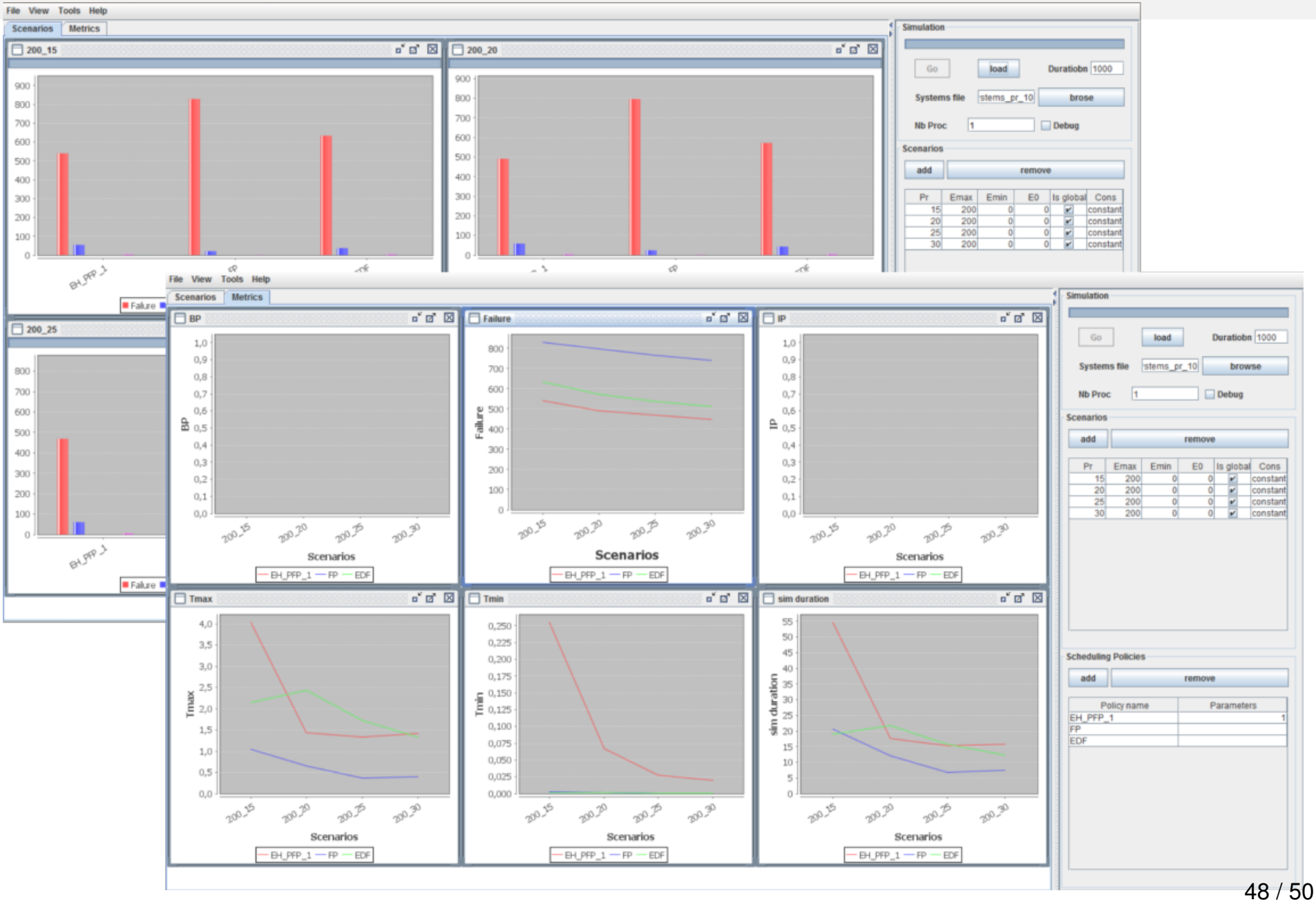
YARTISS


File View Settings Tools Help
Energy evolution
E_{max}
E_{min}
Tasks
Procesors
Tasks time chart
1,1
1,2
1,3
1,4
2,1
2,2
2,3
2,4
3,1
3,1
3,2
3,2
3,2
Simulation
Go
load
Save
Shot
Duration 100
Energy aware
Hard realtime
X alternate 2
Y alternate 10
Scale X 23
Scale Y 4
Results
Failures 0.0
Success state true
Worst Execution Time P=1=2
Scheduling policy
Policy FP
Parameters
Proc count 1
Energy profile
E0 50
Pr 3
Emax 50
Emin 0
Consuption default
Global
System
Count 3
Time U 0.7722
Energy U 1.1037
Busy period 14
hyper period 360
add
remove
Generate
Type Oi Ci Ei Ti Di Pi
simple 0 2 8 8 3
simple 0 3 12 10 9
simple 0 4 20 18 17

YARTISS


File View Settings Tools Help
Processors chronogram
Proc1
Proc2
Tasks
Procesors
Tasks time chart
τ1
τ2
τ3
Simulation
Go
load
Save
Shot
Duration 100
Energy aware
Hard realtime
X alternate 2
Y alternate 10
Scale X 23
Scale Y 4
Results
Failures 0.0
Success state true
Worst Execution Time P=3=€
Scheduling policy
Policy FP
Parameters
Proc count 2
System
Count 3
Time U 1.65
Busy period 33
hyper period 20
add
remove
Generate
Type Oi Ci Ti Di Pi
simple 0 2 4 3
simple 2 3 4 9
simple 0 4 10 17

YARTISS


File View Tools Help
Scenarios Metrics
200_15
200_20
200_25
Simulation
Go
load
Durationn 1000
Systems file
stems_pr_10
brose
Nb Proc 1
Debug
Scenarios
add
remove
Pr Emax Emin E0 Is global Cons
15 200 0 0 constant
20 200 0 0 constant
25 200 0 0 constant
30 200 0 0 constant
BP
Failure
IP
Tmax
Tmin
sim duration
Scenarios
EH_PFP_1 FP EDF
Scheduling Policies
Policy name Parameters
EH_PFP_1 1
FP
EDF

Publication

YARTISS a fait l'objet d'une publication dans le workshop WATERS qui est associé à la conférence ECRTS :

Younes Chandarli, Frederic Fauberteau, Damien Masson, Serge Midonnet and Manar Qamhieh.
"YARTISS : A Tool to Visualize, Test, Compare and Evaluate Real-Time Scheduling Algorithms". WATERS 2012.

WATERS

Plan

1 Introduction

2 Problématiques

3 État de l'art

4 Contributions

5 Conclusion

Conclusion

1. PFP_{ASAP} : un algorithme optimal pour les systèmes non-concrets de tâches consommatrices
   - Une condition d'ordonnançabilité pour la même famille de systèmes
2. Deux conditions d'ordonnançabilité suffisantes basées sur des bornes supérieures du pire temps de réponse des tâches des systèmes pouvant contenir des tâches génératrices d'énergie
3. Une étude approfondie sur l'existence d'un algorithme optimal pour le modèle général
4. Début d'exploration des éventuelles similitudes avec les systèmes sous contraintes thermiques
5. YARTISS : un outil de simulation modulaire et extensible

Perspectives

Pistes d'approfondissement :

- Étudier le modèle sous d'autres hypothèses : profils de rechargement et de consommation
- Continuer l'exploration des similitudes avec les systèmes sous contraintes thermiques
- Étudier le modèle sur des plateformes multiprocesseur

Problèmes ouverts :

- Prouver que le problème d'ordonnancement à priorité fixe des systèmes collecteurs d'énergie est un problème NP-Difficile

Merci pour votre attention

Production scientifique

Publications :

- Deux articles de conférence
- Un atelier (Workshop)
- Deux articles courts (WIP)
- Un journal en cours de rédaction

Distinctions :

- Prix du meilleur article de la conférence RTNS 2014

Logiciels :

- Le développent de YARTISS

Communication :

- Présentation des articles aux conférences
- Présentation des papiers courts sous forme de posters
- Présentation de YARTISS dans plusieurs séminaires

Mobilité internationale :

- Séjour de 2 mois à *Wayne State University* à Detroit aux États-Unis pour une collaboration avec Prof. Nathan Fisher

Remerciements

Ma famille et mes amis ...

L'équipe d'encadrement :

- Mon directeur de thèse Laurent George
- Mon encadrant Damien Masson

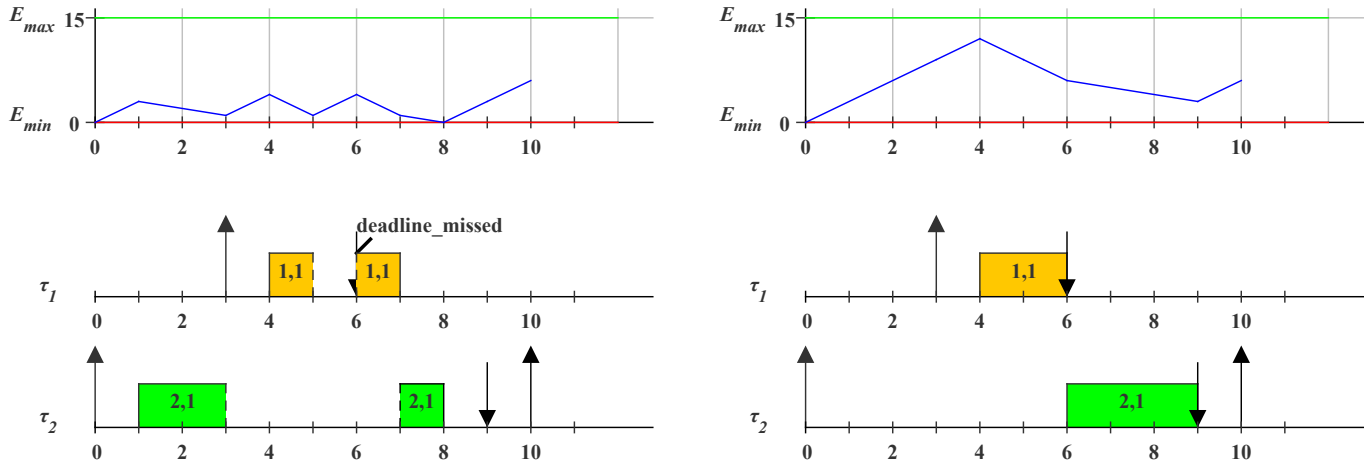
Les collaborateurs :

- Yasmina Abdeddaïm
- Rob Davis
- Nathan Fisher

Et tous ceux que j'ai pu rencontrer pendant cette thèse :

- Les amis et les collègues doctorants et titulaires
- L'équipe administrative du LIGM et de l'ESIEE
- L'équipe administrative de l'école doctorale

Recherche d'algorithme optimal



Pour le modèle général à priorité fixe :

- PFP_{ASAP}, PFP_{ALAP} et PFP_{ST} ne sont pas optimaux

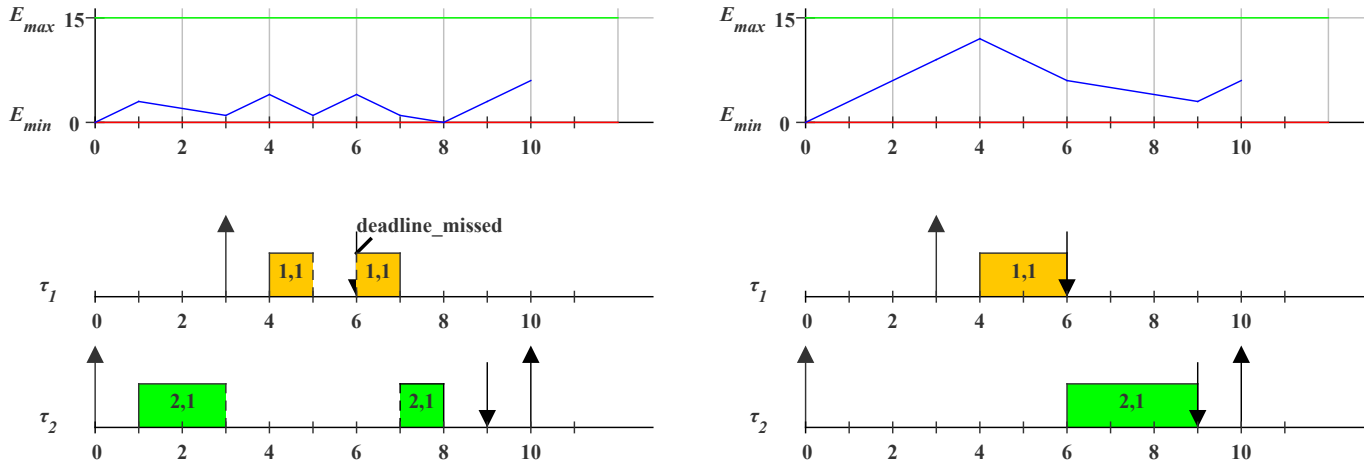
La non-oisiveté énergétique

Aucun algorithme non-oisif énergétiquement n'est optimal.

La clairvoyance

L'algorithme optimal doit être clairvoyant temporellement et énergétiquement.

Recherche d'algorithme optimal



Pour le modèle général à priorité fixe :

- PFP_{ASAP}, PFP_{ALAP} et PFP_{ST} ne sont pas optimaux

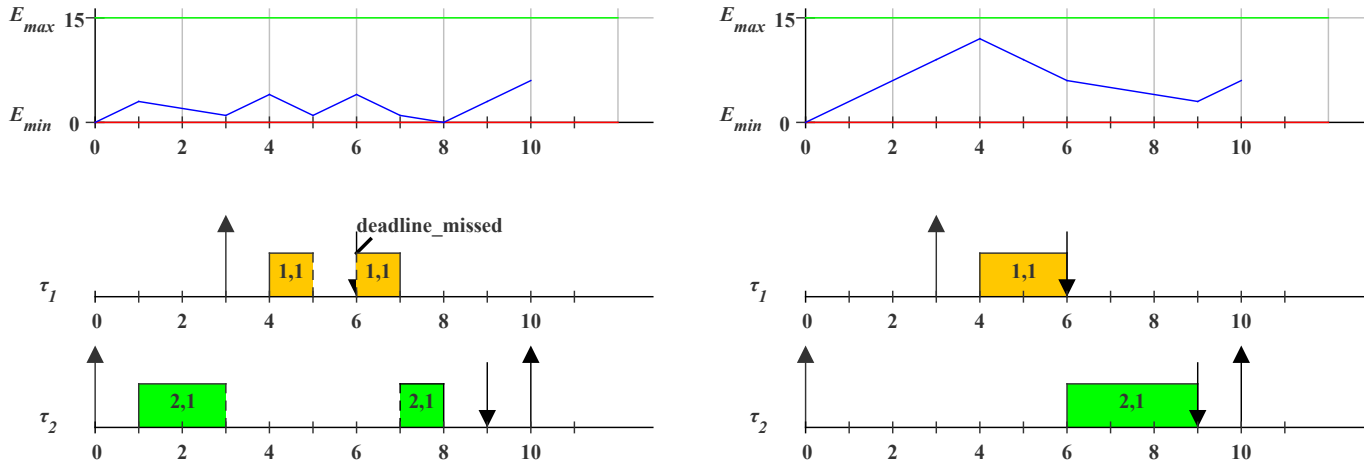
La non-oisiveté énergétique

Aucun algorithme non-oisif énergétiquement n'est optimal.

La clairvoyance

L'algorithme optimal doit être clairvoyant temporellement et énergétiquement.

Recherche d'algorithme optimal



Pour le modèle général à priorité fixe :

- PFP_{ASAP}, PFP_{ALAP} et PFP_{ST} ne sont pas optimaux

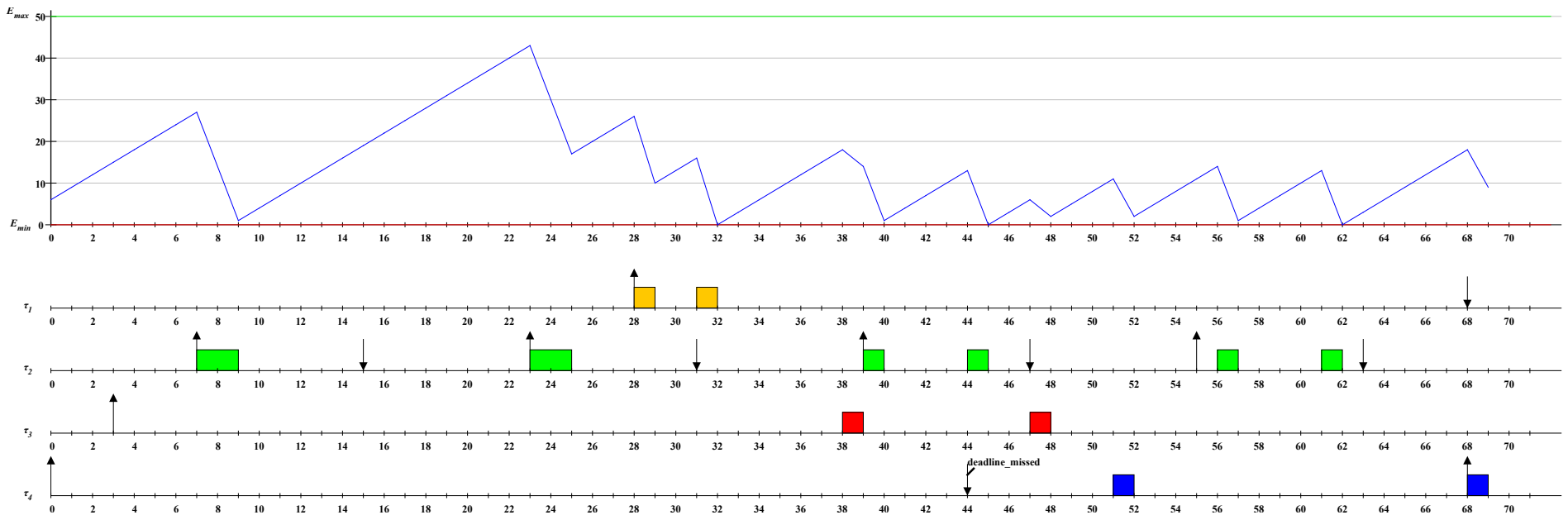
La non-oisiveté énergétique

Aucun algorithme non-oisif énergétiquement n'est optimal.

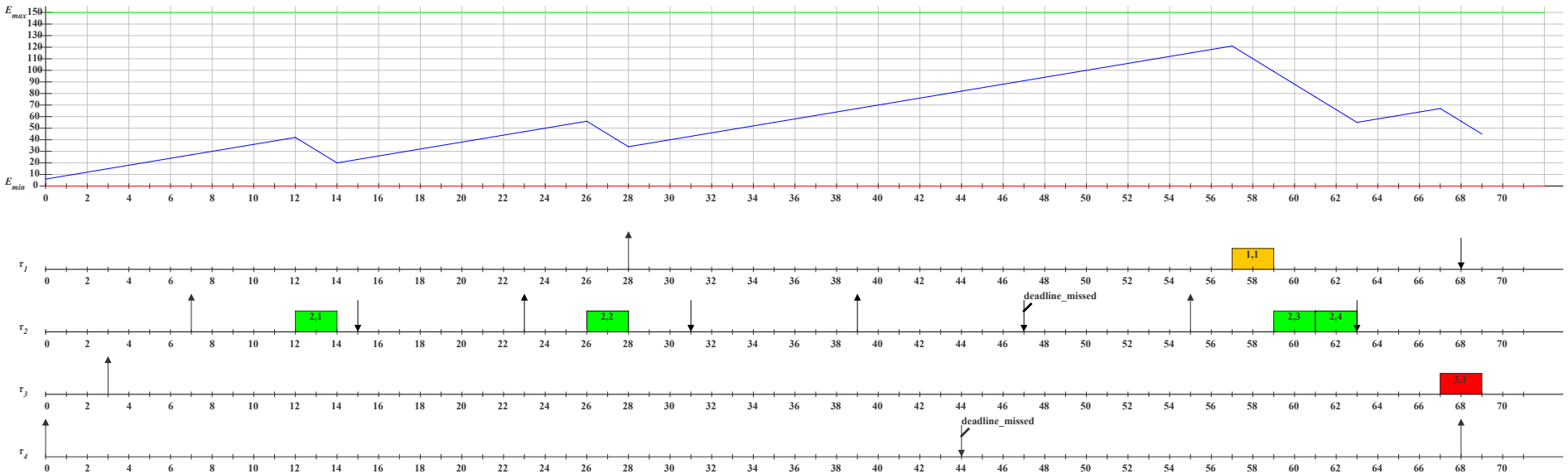
La clairvoyance

L'algorithme optimal doit être clairvoyant temporellement et énergétiquement.

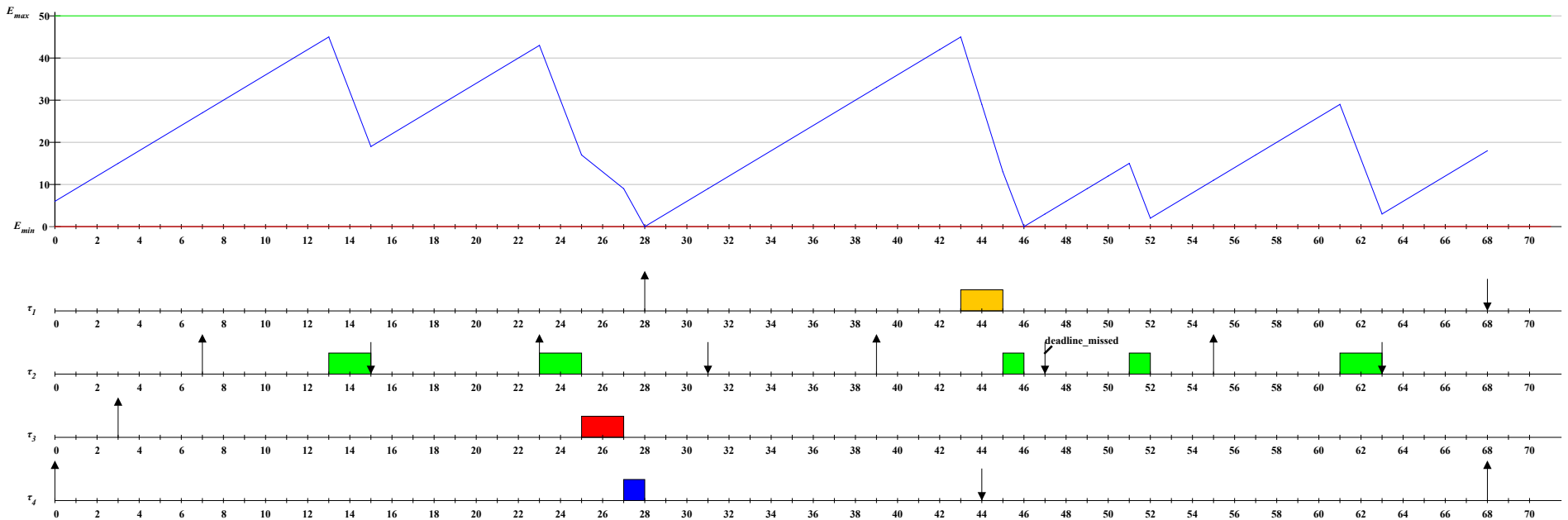
L'algorithme FPC_{ASAP}



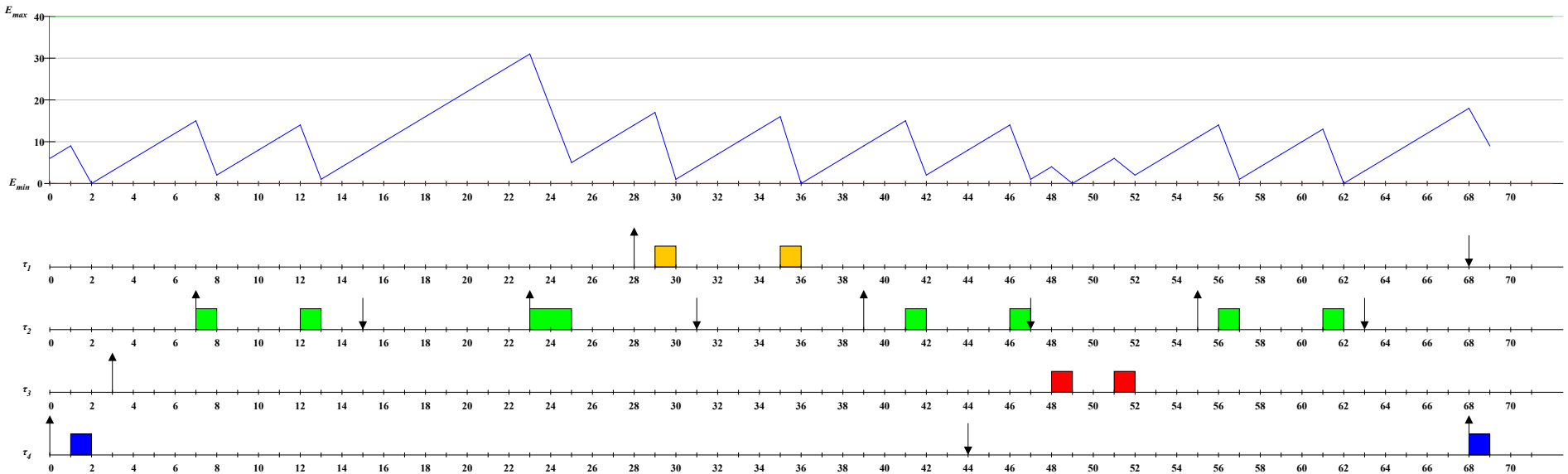
L'algorithme *FPLSA*



L'algorithme *FPLeg*



L'algorithme FP_{lh}

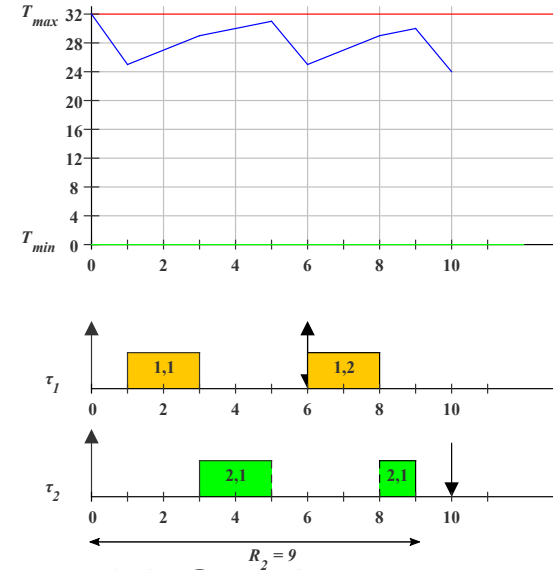


Systèmes avec contraintes thermiques

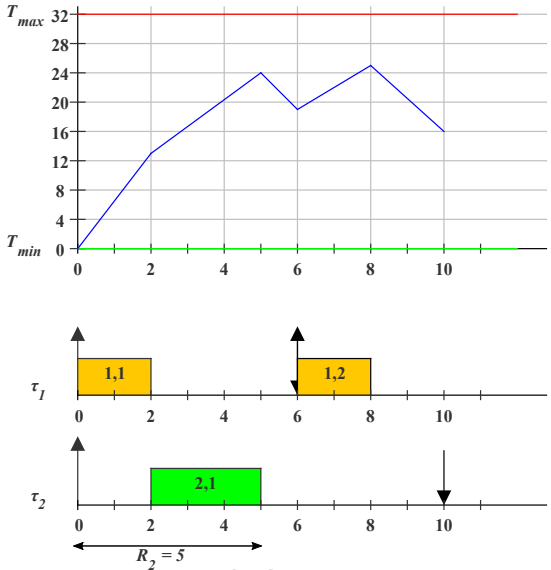
$$T'(t) = a - b \times T(t) \tag{2}$$

$$T(t) = \frac{a}{b} + \left(T(t_0) - \frac{a}{b}\right) \times e^{-b(t-t_0)} \tag{3}$$

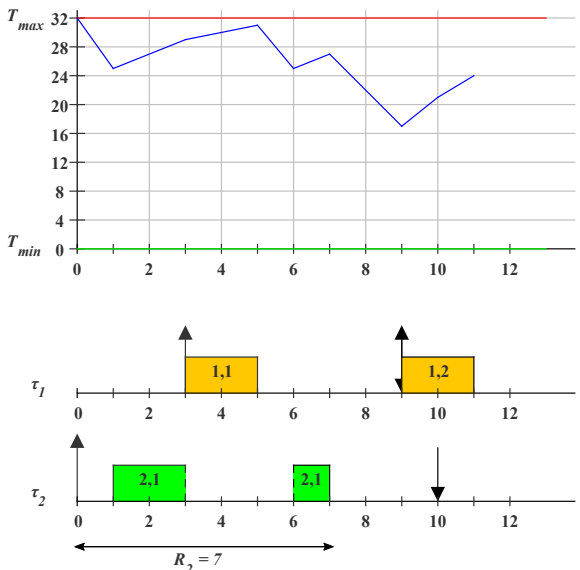
$$T(t) = T(t_0) \times e^{-b(t-t_0)} \tag{4}$$



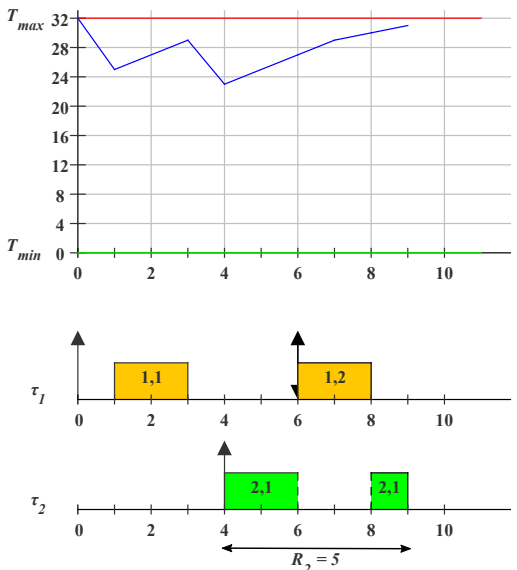
(a) Synchronous



(b) $T(0) < T_{max}$

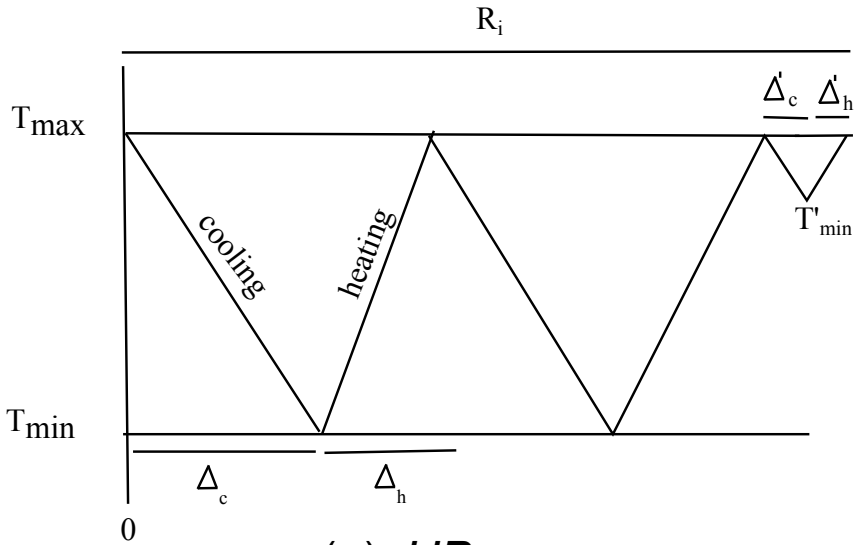


(c) Negative shifting

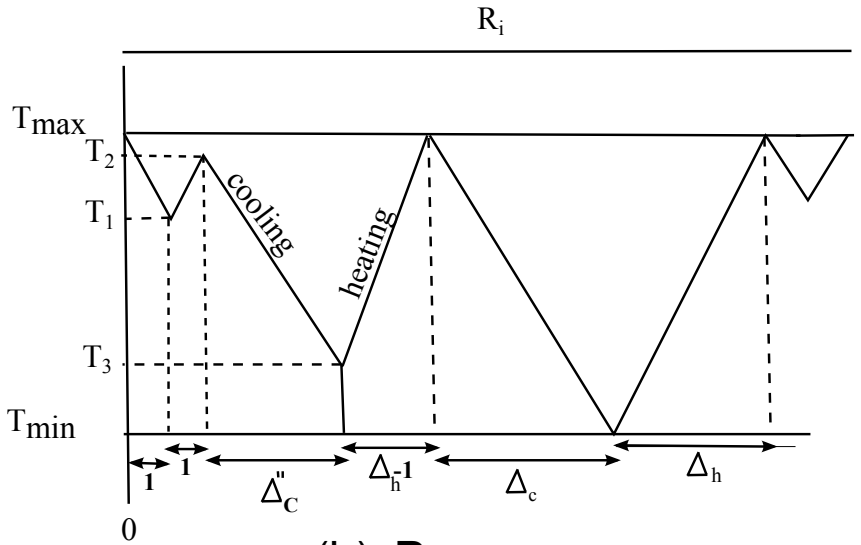


(d) Positive shifting

Borne $UB_{T_{min}}$

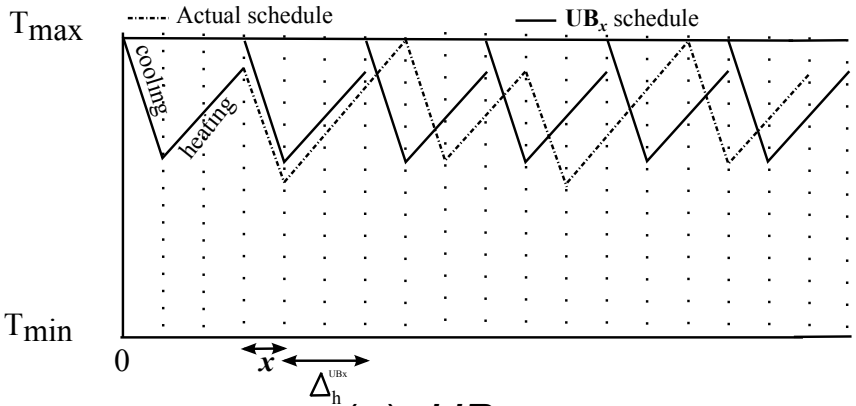


(a) $UB_{T_{min}}$

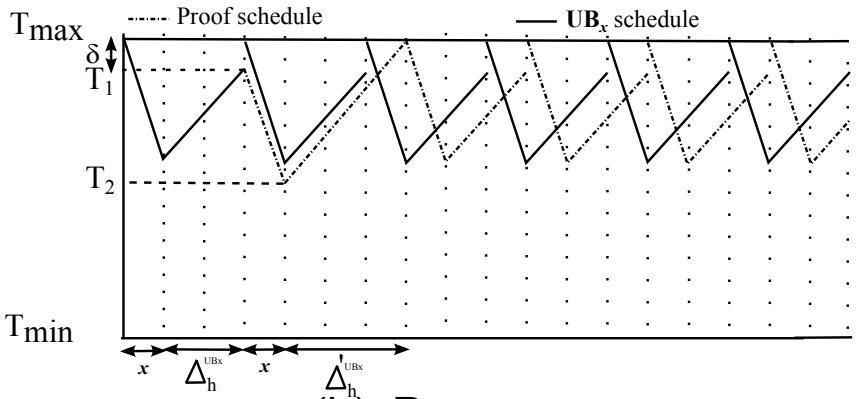


(b) Preuve

Borne UB_x

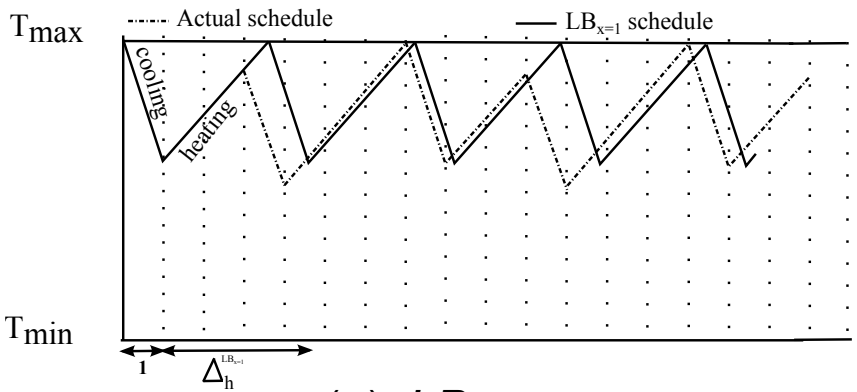


(a) UB_x



(b) Preuve

Borne $LB_{x=1}$



(a) $LB_{x=1}$