Arnaud LEGRAND
Laboratoire de l'Informatique du parallélisme
http://www.ens-lyon.fr/LIP
http://graal.ens-lyon.fr/~alegrand

Ordonnancement asymptotique de graphes de tâches identiques et indépendants sur plate-forme hétérogène

Projet *ReMaP* (CNRS, ENS Lyon et INRIA)
Algorithmique, ordonnancement et équilibrage de charge pour plate-forme hétérogène

Plan de l'exposé

1. Ordonnancement maître/esclave : modélisation et métrique classique
2. Ordonnancement de tâches indépendantes en régime permanent
   - Modélisation
   - Calcul du régime permanent optimal
   - Construction d'un ordonnancement asymptotiquement optimal
3. Ordonnancement de graphes de tâches indépendants en régime permanent
4. Conclusion

Plan de l’exposé

1. Ordonnancement maître/esclave : modélisation et métrique classique
2. Ordonnancement de tâches indépendantes en régime permanent
   - Modélisation
   - Calcul du régime permanent optimal
   - Construction d’un ordonnancement asymptotiquement optimal
3. Ordonnancement de graphes de tâches indépendants en régime permanent
4. Conclusion

Plan de l'exposé

1. Ordonnancement maître/esclave : modélisation et métrique classique
2. Ordonnancement de tâches indépendantes en régime permanent
   - Modélisation
   - Calcul du régime permanent optimal
   - Construction d'un ordonnancement asymptotiquement optimal
3. Ordonnancement de graphes de tâches indépendants en régime permanent
4. Conclusion

Plan de l’exposé

1. Ordonnancement maître/esclave : modélisation et métrique classique
2. Ordonnancement de tâches indépendantes en régime permanent
   - Modélisation
   - Calcul du régime permanent optimal
   - Construction d’un ordonnancement asymptotiquement optimal
3. Ordonnancement de graphes de tâches indépendants en régime permanent
4. Conclusion

Plan de l'exposé

1. Ordonnancement maître/esclave : modélisation et métrique classique
2. Ordonnancement de tâches indépendantes en régime permanent
   - Modélisation
   - Calcul du régime permanent optimal
   - Construction d'un ordonnancement asymptotiquement optimal
3. Ordonnancement de graphes de tâches indépendants en régime permanent
4. Conclusion

Plan de l'exposé

1. Ordonnancement maître/esclave : modélisation et métrique classique
2. Ordonnancement de tâches indépendantes en régime permanent
   - Modélisation
   - Calcul du régime permanent optimal
   - Construction d'un ordonnancement asymptotiquement optimal
3. Ordonnancement de graphes de tâches indépendants en régime permanent
4. Conclusion

Plan de l’exposé

1. Ordonnancement maître/esclave : modélisation et métrique classique
2. Ordonnancement de tâches indépendantes en régime permanent
   - Modélisation
   - Calcul du régime permanent optimal
   - Construction d’un ordonnancement asymptotiquement optimal
3. Ordonnancement de graphes de tâches indépendants en régime permanent
4. Conclusion

Ordonnancement maître/esclave

Architecture Maître-esclave

Architecture Maître/esclave Une technique simple mais efficace.

Mise en œuvre classique Un certain nombre de tâches indépendantes sont traitées par des processeurs identiques (les esclaves) sous la supervision d'un processeur particulier (le maître).

Version hétérogène Les temps de calcul et de communication des esclaves sont différents les uns des autres.

Applications Toute simulation de type Monte Carlo :

- ⇝ microphysiologie cellulaire,
- ⇝ simulations physiques,
- ⇝ conformations de protéines,. . .

Architecture Maître-esclave

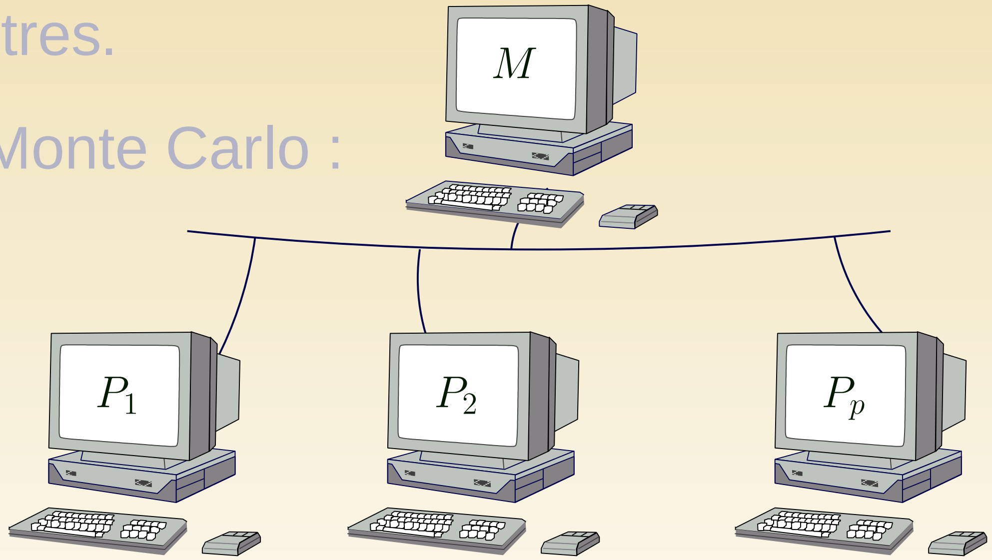
Architecture Maître/esclave Une technique simple mais efficace.

Mise en œuvre classique Un certain nombre de tâches indépendantes sont traitées par des processeurs identiques (les esclaves) sous la supervision d'un processeur particulier (le maître).

Version hétérogène Les temps de calcul et de communication des esclaves sont différents les uns des autres.

Applications Toute simulation de type Monte Carlo :

- microphysiologie cellulaire,
- simulations physiques,
- conformations de protéines,...

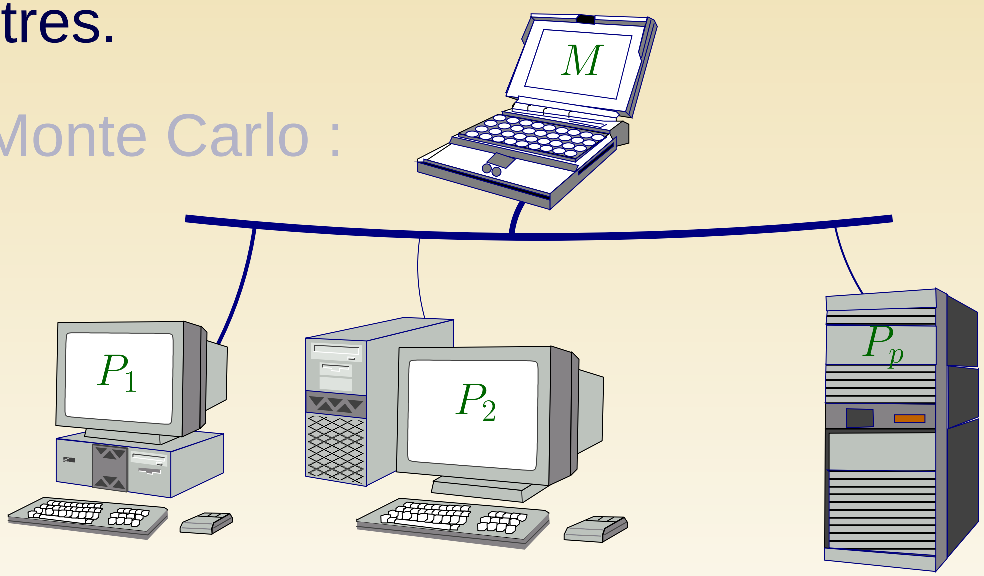


Architecture Maître-esclave

Architecture Maître/esclave Une technique simple mais efficace.

Mise en œuvre classique Un certain nombre de tâches indépendantes sont traitées par des processeurs identiques (les esclaves) sous la supervision d'un processeur particulier (le maître).

Version hétérogène Les temps de calcul et de communication des esclaves sont différents les uns des autres.



Applications Toute simulation de type Monte Carlo :

- microphysiologie cellulaire,
- simulations physiques,
- conformations de protéines,...

Architecture Maître-esclave

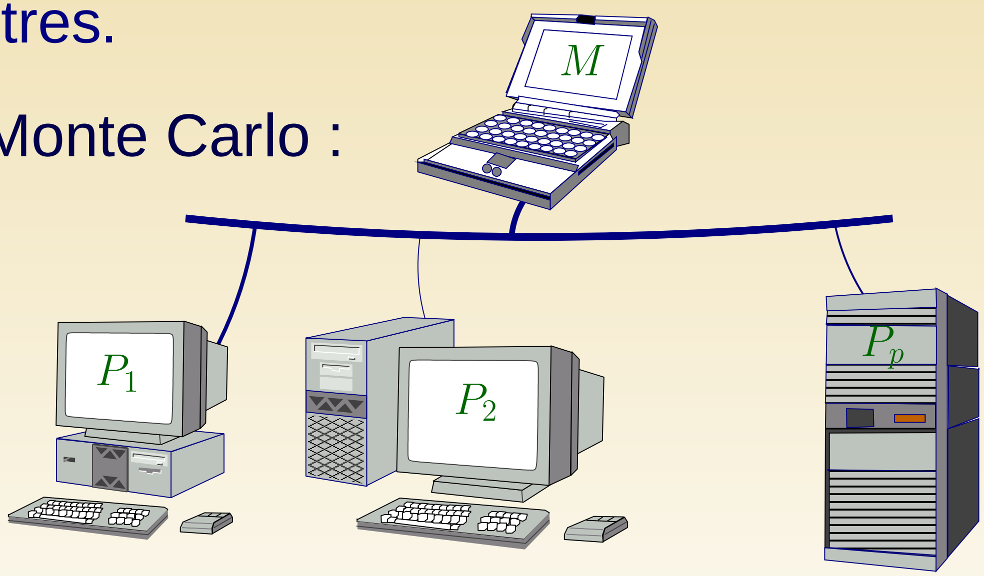
Architecture Maître/esclave Une technique simple mais efficace.

Mise en œuvre classique Un certain nombre de tâches indépendantes sont traitées par des processeurs identiques (les esclaves) sous la supervision d'un processeur particulier (le maître).

Version hétérogène Les temps de calcul et de communication des esclaves sont différents les uns des autres.

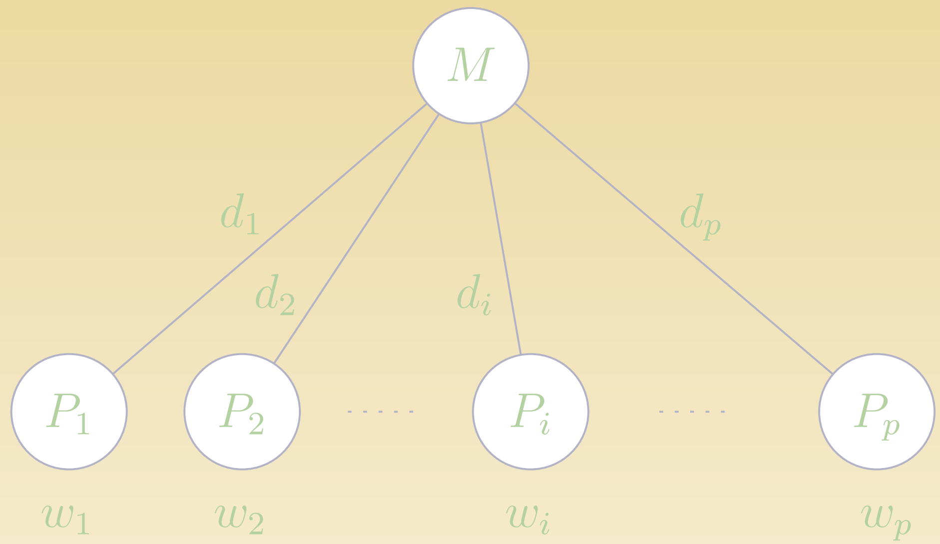
Applications Toute simulation de type Monte Carlo :

- ⇝ microphysiologie cellulaire,
- ⇝ simulations physiques,
- ⇝ conformations de protéines,. . .



Modélisation

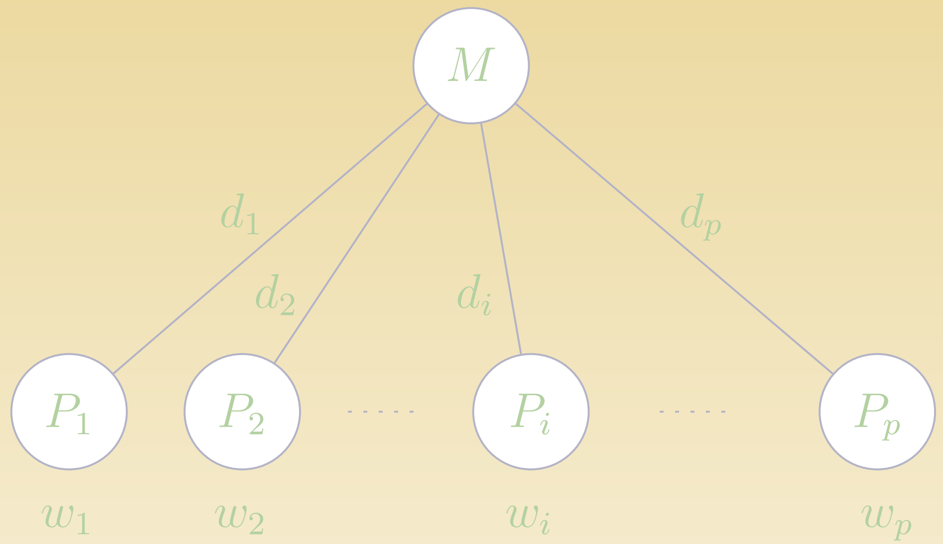
- Ensemble de tâches indépendantes à traiter par les p esclaves.
- Toutes les tâches sont identiques : elles représentent la même quantité de calcul



- Il faut un temps d_i pour transférer une tâche de M à P_i et un temps w_i pour la traiter sur P_i
- Les communications sont 1-port : M ne peut envoyer qu'une seule tâche à la fois.
- Recouvrement des calculs et des communications.

Modélisation

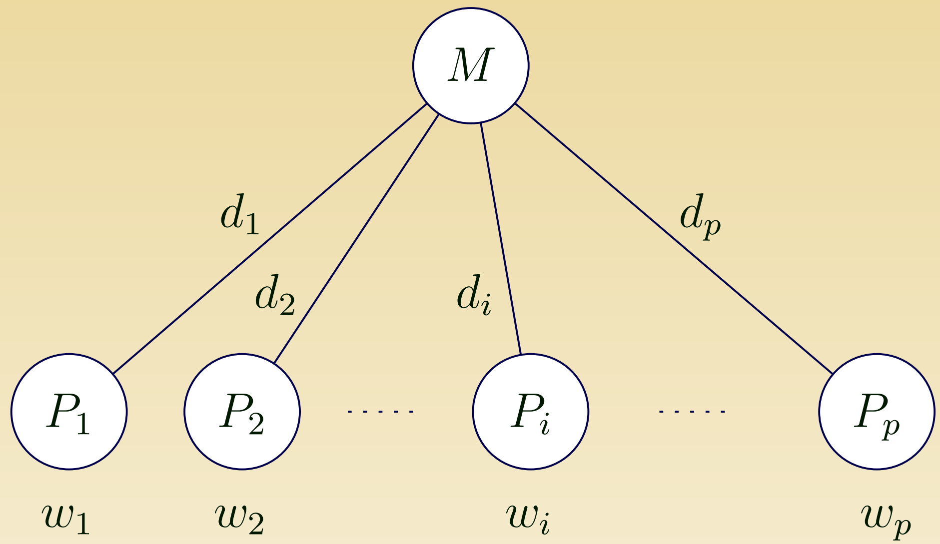
- Ensemble de tâches indépendantes à traiter par les p esclaves.
- Toutes les tâches sont identiques : elles représentent la même quantité de calcul



- Il faut un temps d_i pour transférer une tâche de M à P_i et un temps w_i pour la traiter sur P_i
- Les communications sont 1-port : M ne peut envoyer qu'une seule tâche à la fois.
- Recouvrement des calculs et des communications.

Modélisation

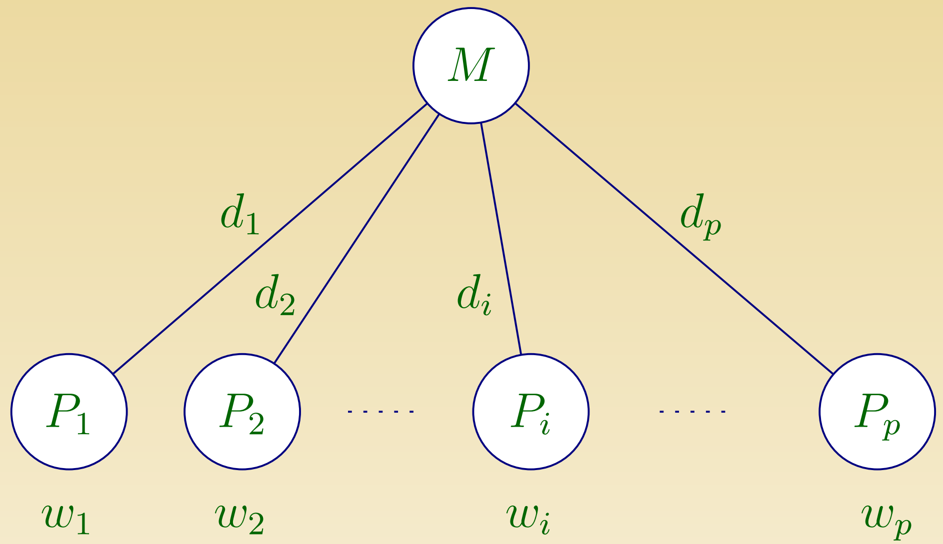
- Ensemble de tâches indépendantes à traiter par les p esclaves.
- Toutes les tâches sont identiques : elles représentent la même quantité de calcul



- Il faut un temps d_i pour transférer une tâche de M à P_i et un temps w_i pour la traiter sur P_i
- Les communications sont 1-port : M ne peut envoyer qu'une seule tâche à la fois.
- Recouvrement des calculs et des communications.

Modélisation

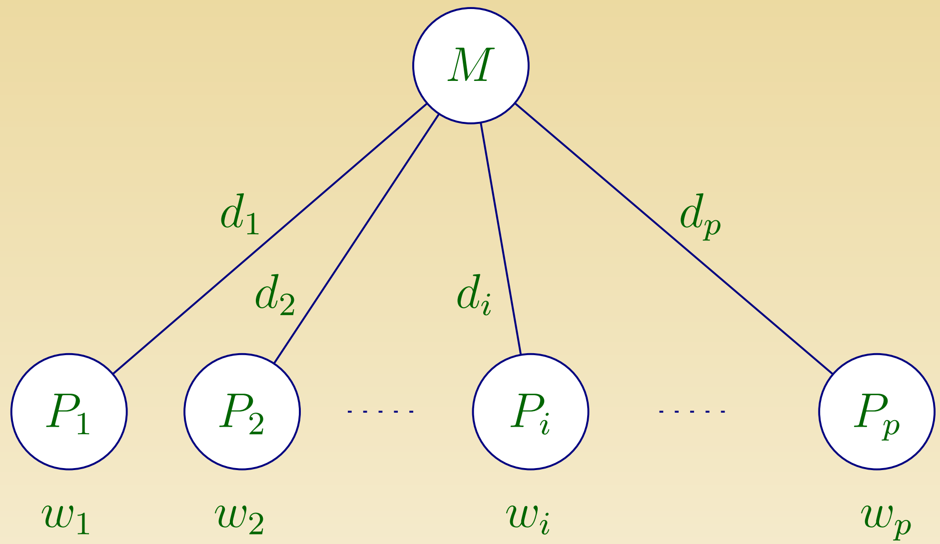
- Ensemble de tâches indépendantes à traiter par les p esclaves.
- Toutes les tâches sont identiques : elles représentent la même quantité de calcul



- Il faut un temps d_i pour transférer une tâche de M à P_i et un temps w_i pour la traiter sur P_i
- Les communications sont 1-port : M ne peut envoyer qu'une seule tâche à la fois.
- Recouvrement des calculs et des communications.

Modélisation

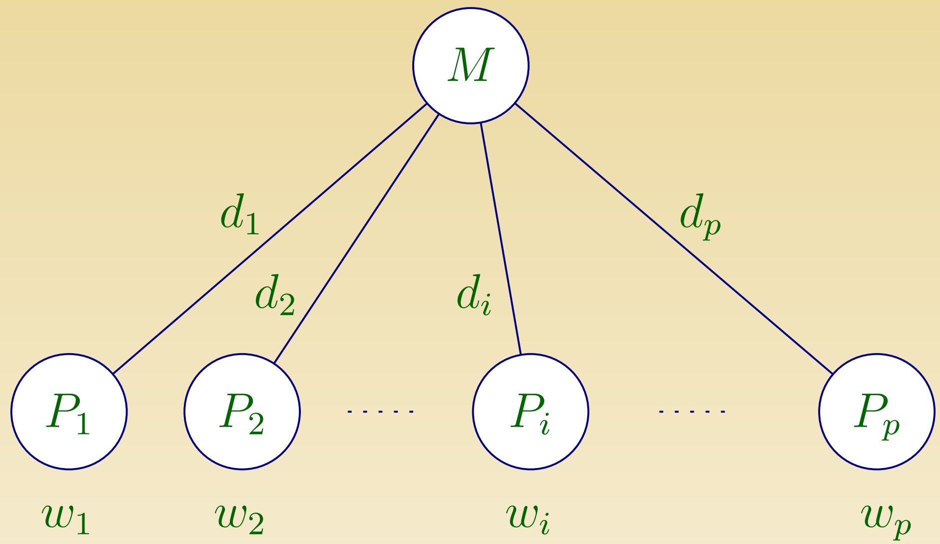
- Ensemble de tâches indépendantes à traiter par les p esclaves.
- Toutes les tâches sont identiques : elles représentent la même quantité de calcul



- Il faut un temps d_i pour transférer une tâche de M à P_i et un temps w_i pour la traiter sur P_i
- Les communications sont 1-port : M ne peut envoyer qu'une seule tâche à la fois.
- Recouvrement des calculs et des communications.

Modélisation

- Ensemble de tâches indépendantes à traiter par les p esclaves.
- Toutes les tâches sont identiques : elles représentent la même quantité de calcul



- Il faut un temps d_i pour transférer une tâche de M à P_i et un temps w_i pour la traiter sur P_i
- Les communications sont 1-port : M ne peut envoyer qu'une seule tâche à la fois.
- Recouvrement des calculs et des communications.

Résultats de complexité

Définition (MasterSlave ($P_1(d_1, w_1), \ldots, P_p(d_p, w_p), n$)). Étant donné une plate-forme maître-esclave de caractéristique $(d_1, w_1), \ldots, (d_p, w_p)$, quel est le temps minimal nécessaire au traitement de n tâches ?

MasterSlave ($P_1(d_1, w_1), \ldots, P_p(d_p, w_p), n$) peut être résolu en temps $O(n^2p^2)$ avec un algorithme glouton non trivial [BLR02].

Si le graphe d'interconnexion de plate-forme est une chaîne ou une pieuvre, le problème reste polynomial [Dut03b].

En revanche si la plate-forme est un arbre, le problème devient NP-complet [Dut03a].

Résultats de complexité

Définition (MasterSlave ($P_1(d_1, w_1), \ldots, P_p(d_p, w_p), n$)). Étant donné une plate-forme maître-esclave de caractéristique $(d_1, w_1), \ldots, (d_p, w_p)$, quel est le temps minimal nécessaire au traitement de n tâches ?

MasterSlave ($P_1(d_1, w_1), \ldots, P_p(d_p, w_p), n$) peut être résolu en temps $O(n^2 p^2)$ avec un algorithme glouton non trivial [BLR02].

Si le graphe d'interconnexion de plate-forme est une chaîne ou une pieuvre, le problème reste polynomial [Dut03b].

En revanche si la plate-forme est un arbre, le problème devient NP-complet [Dut03a].

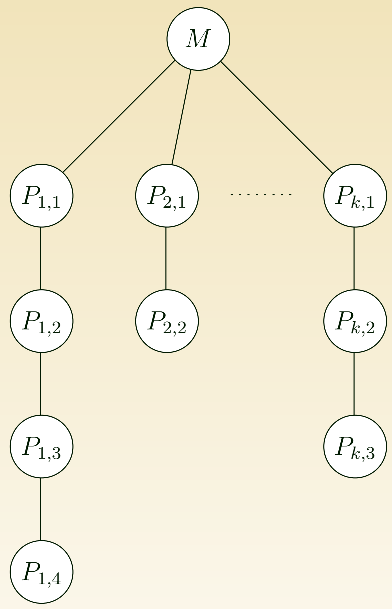
Résultats de complexité

Définition (MasterSlave ($P_1(d_1, w_1), \ldots, P_p(d_p, w_p), n$)). Étant donné une plate-forme maître-esclave de caractéristique $(d_1, w_1), \ldots, (d_p, w_p)$, quel est le temps minimal nécessaire au traitement de n tâches ?

MasterSlave ($P_1(d_1, w_1), \ldots, P_p(d_p, w_p), n$) peut être résolu en temps $O(n^2p^2)$ avec un algorithme glouton non trivial [BLR02].

Si le graphe d'interconnexion de plate-forme est une chaîne ou une pieuvre, le problème reste polynomial [Dut03b].

En revanche si la plate-forme est un arbre, le problème devient NP-complet [Dut03a].

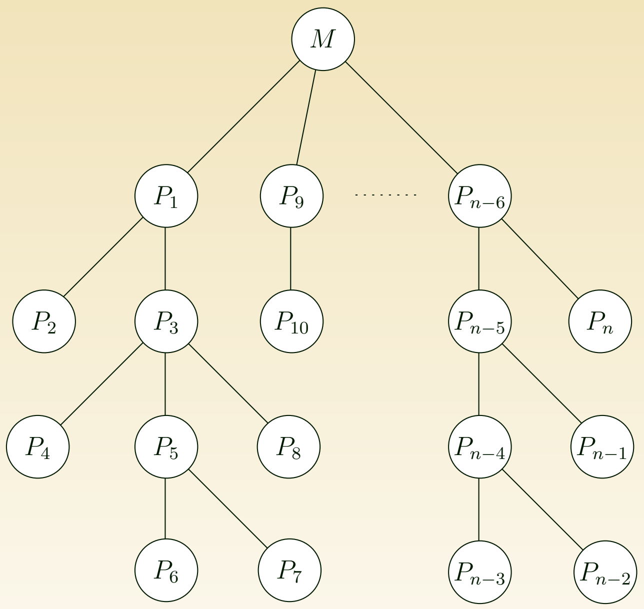


Résultats de complexité

Définition (MasterSlave ($P_1(d_1, w_1), \ldots, P_p(d_p, w_p), n$)). Étant donné une plate-forme maître-esclave de caractéristique $(d_1, w_1), \ldots, (d_p, w_p)$, quel est le temps minimal nécessaire au traitement de n tâches ?

MasterSlave ($P_1(d_1, w_1), \ldots, P_p(d_p, w_p), n$) peut être résolu en temps $O(n^2p^2)$ avec un algorithme glouton non trivial [BLR02].

Si le graphe d'interconnexion de plate-forme est une chaîne ou une pieuvre, le problème reste polynomial [Dut03b].
En revanche si la plate-forme est un arbre, le problème devient NP-complet [Dut03a].



Une modélisation peu adaptée...

La difficulté provient de la métrique utilisée : le makespan.

Cette métrique n'est pas adaptée à une plate-forme distribuée à grande échelle :

- la modélisation d'une grille de calcul ainsi que la mesure des différents paramètres est extrêmement délicate ;
- étant donné le temps de déploiement et la difficulté de mise en œuvre de telles plates-formes, le nombre de tâches à traiter est généralement très grand.

En se plaçant en régime permanent, on peut considérer des modèles de plates-formes bien plus réalistes tout en construisant des ordonnancements efficaces.

Une modélisation peu adaptée...

La difficulté provient de la métrique utilisée : le makespan.

Cette métrique n'est pas adaptée à une plate-forme distribuée à grande échelle :

- la modélisation d'une grille de calcul ainsi que la mesure des différents paramètres est extrêmement délicate ;
- étant donné le temps de déploiement et la difficulté de mise en œuvre de telles plates-formes, le nombre de tâches à traiter est généralement très grand.

En se plaçant en régime permanent, on peut considérer des modèles de plates-formes bien plus réalistes tout en construisant des ordonnancements efficaces.

Une modélisation peu adaptée...

La difficulté provient de la métrique utilisée : le makespan.

Cette métrique n'est pas adaptée à une plate-forme distribuée à grande échelle :

- la modélisation d'une grille de calcul ainsi que la mesure des différents paramètres est extrêmement délicate ;
- étant donné le temps de déploiement et la difficulté de mise en œuvre de telles plates-formes, le nombre de tâches à traiter est généralement très grand.

En se plaçant en régime permanent, on peut considérer des modèles de plates-formes bien plus réalistes tout en construisant des ordonnancements efficaces.

Une modélisation peu adaptée...

La difficulté provient de la métrique utilisée : le makespan.

Cette métrique n'est pas adaptée à une plate-forme distribuée à grande échelle :

- la modélisation d'une grille de calcul ainsi que la mesure des différents paramètres est extrêmement délicate ;
- étant donné le temps de déploiement et la difficulté de mise en œuvre de telles plates-formes, le nombre de tâches à traiter est généralement très grand.

En se plaçant en régime permanent, on peut considérer des modèles de plates-formes bien plus réalistes tout en construisant des ordonnancements efficaces.

Une modélisation peu adaptée…

La difficulté provient de la métrique utilisée : le makespan.

Cette métrique n'est pas adaptée à une plate-forme distribuée à grande échelle :

- la modélisation d'une grille de calcul ainsi que la mesure des différents paramètres est extrêmement délicate ;
- étant donné le temps de déploiement et la difficulté de mise en œuvre de telles plates-formes, le nombre de tâches à traiter est généralement très grand.

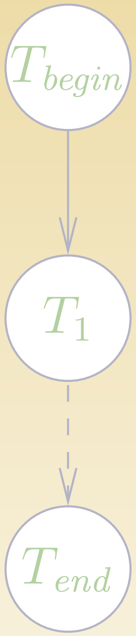
En se plaçant en régime permanent, on peut considérer des modèles de plates-formes bien plus réalistes tout en construisant des ordonnancements efficaces.

Tâches indépendantes en régime permanent : modélisation

Graphe de tâches

On dispose de n problèmes $\mathcal{P}^{(1)}, \mathcal{P}^{(2)}, \ldots, \mathcal{P}^{(n)}$ à résoudre avec n grand.

Chaque problème correspond à une copie d'un même graphe de tâches $G_A = (V_A, E_A)$, le graphe d'application.

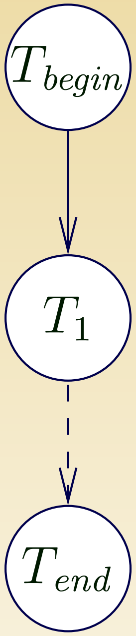


T_{begin} et T_{end} sont fictives : elles permettent de modéliser la distribution des fichiers d'entrée ou le rapatriement des fichiers de sortie.

Graphe de tâches

On dispose de n problèmes $\mathcal{P}^{(1)}, \mathcal{P}^{(2)}, \ldots, \mathcal{P}^{(n)}$ à résoudre avec n grand.

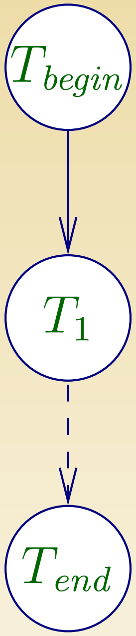
Chaque problème correspond à une copie d'un même graphe de tâches $G_A = (V_A, E_A)$, le graphe d'application.



T_{begin} et T_{end} sont fictives : elles permettent de modéliser la distribution des fichiers d'entrée ou le rapatriement des fichiers de sortie.

Graphe de tâches

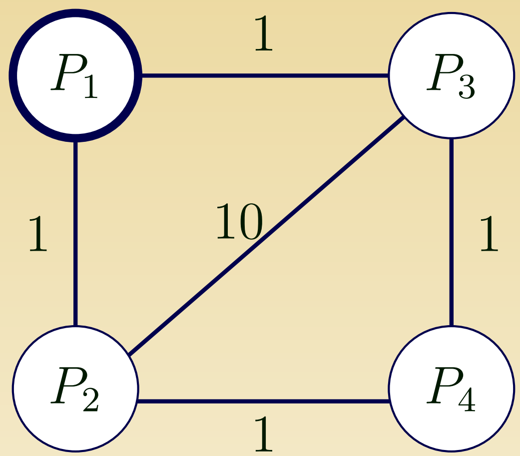
On dispose de n problèmes $\mathcal{P}^{(1)}, \mathcal{P}^{(2)}, \ldots, \mathcal{P}^{(n)}$ à résoudre avec n grand. Chaque problème correspond à une copie d'un même graphe de tâches $G_A = (V_A, E_A)$, le graphe d'application.



T_{begin} et T_{end} sont fictives : elles permettent de modéliser la distribution des fichiers d'entrée ou le rapatriement des fichiers de sortie.

Graphe de Plateforme

La plate-forme est représentée à l'aide d'un graphe $G_P = (V_P, E_P)$ appelé graphe de plate-forme.

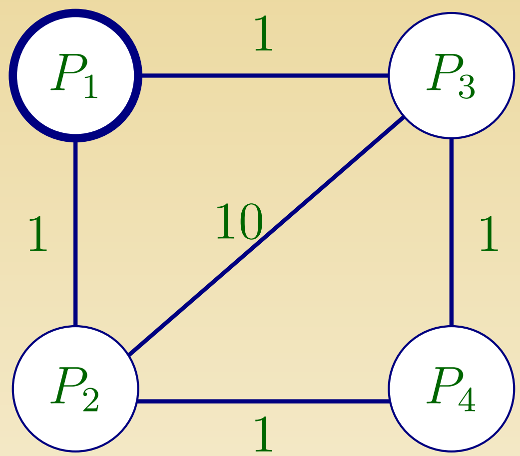


Chaque arête $P_i \to P_j$ est étiquetée par $c_{i,j}$: le temps nécessaire à l'envoi d'un message de taille unitaire entre P_i et P_j.

Modèle de communications : recouvrement complet, modèle 1-port en entrée et en sortie.

Graphe de Plateforme

La plate-forme est représentée à l'aide d'un graphe $G_P = (V_P, E_P)$ appelé graphe de plate-forme.

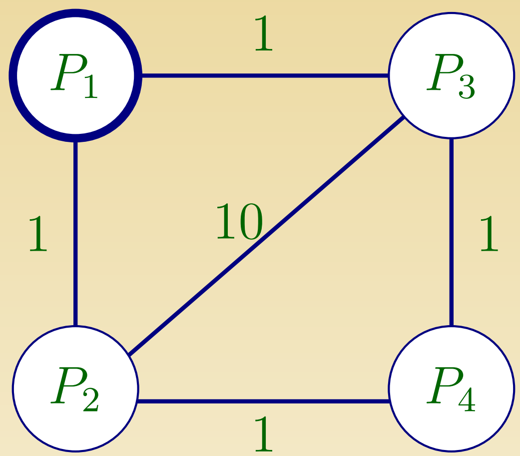


Chaque arête $P_i \to P_j$ est étiquetée par $c_{i,j}$: le temps nécessaire à l'envoi d'un message de taille unitaire entre P_i et P_j.

Modèle de communications : recouvrement complet, modèle 1-port en entrée et en sortie.

Graphe de Plateforme

La plate-forme est représentée à l'aide d'un graphe $G_P = (V_P, E_P)$ appelé graphe de plate-forme.

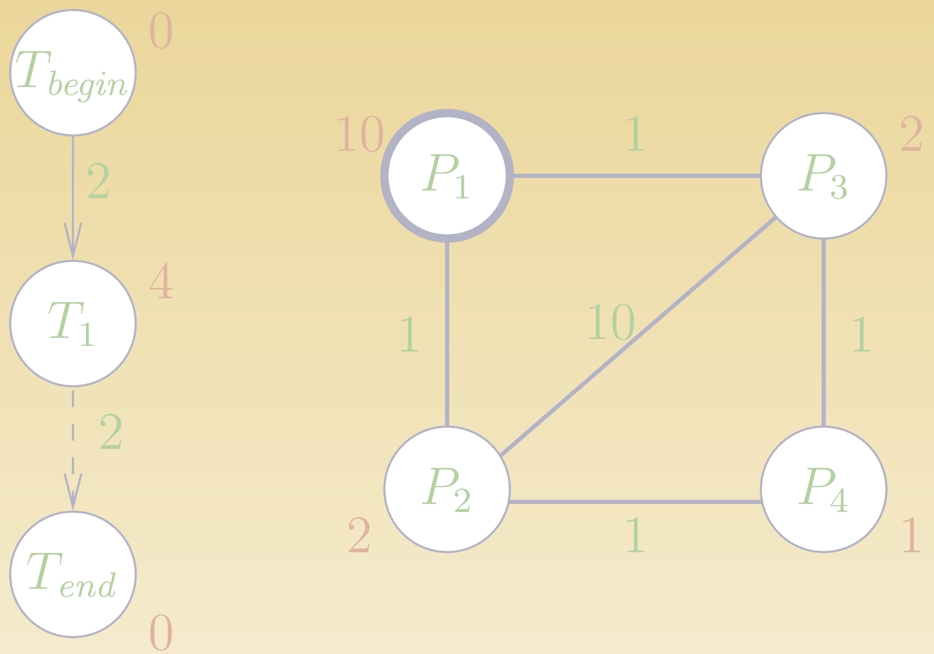


Chaque arête $P_i \to P_j$ est étiquetée par $c_{i,j}$: le temps nécessaire à l'envoi d'un message de taille unitaire entre P_i et P_j.

Modèle de communications : recouvrement complet, modèle 1-port en entrée et en sortie.

Temps de communications et de calcul

Il faut $w_{i,k}$ unités de temps au processeur P_i pour traiter la tâche T_k.

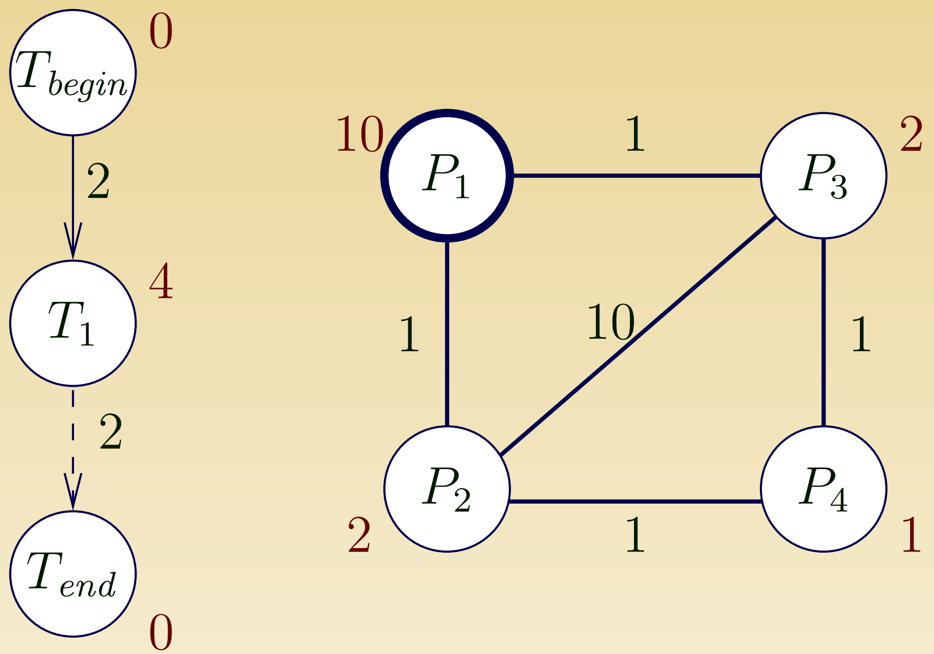


Chaque arête $e_{k,l} : T_k \rightarrow T_l$ de G_A est étiquetée par un coût de communication $data_{k,l}$ représentant la quantité de données créées par T_k et utilisées par T_l.

Pour transférer $e_{k,l}$ entre P_i et P_j, il faut donc un temps égal à $data_{k,l} \times c_{i,j}$.

Temps de communications et de calcul

Il faut $w_{i,k}$ unités de temps au processeur P_i pour traiter la tâche T_k.

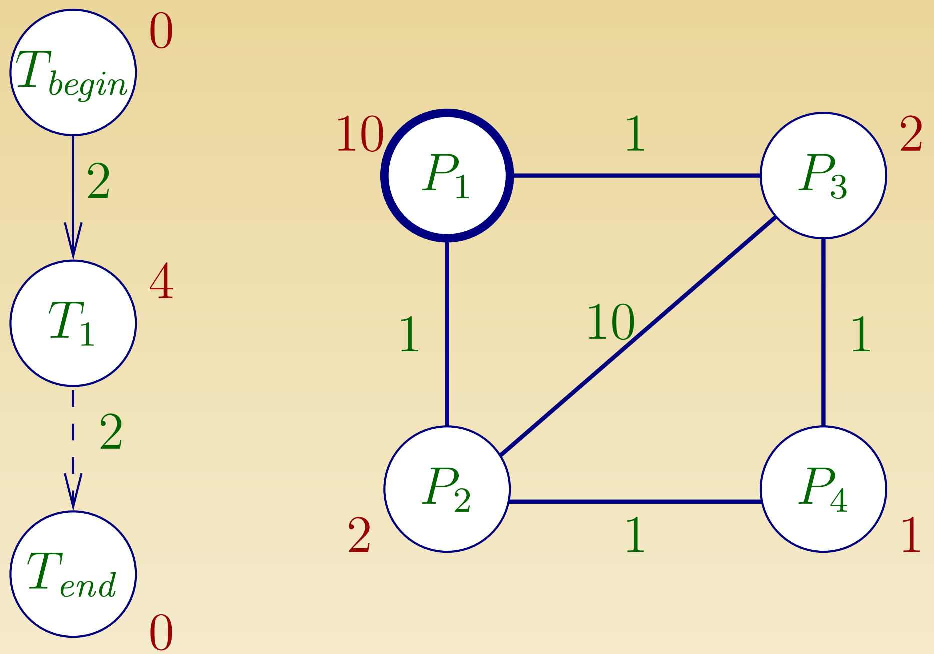


Chaque arête $e_{k,l} : T_k \to T_l$ de G_A est étiquetée par un coût de communication $data_{k,l}$ représentant la quantité de données créées par T_k et utilisées par T_l.

Pour transférer $e_{k,l}$ entre P_i et P_j, il faut donc un temps égal à $data_{k,l} \times c_{i,j}$.

Temps de communications et de calcul

Il faut $w_{i,k}$ unités de temps au processeur P_i pour traiter la tâche T_k.

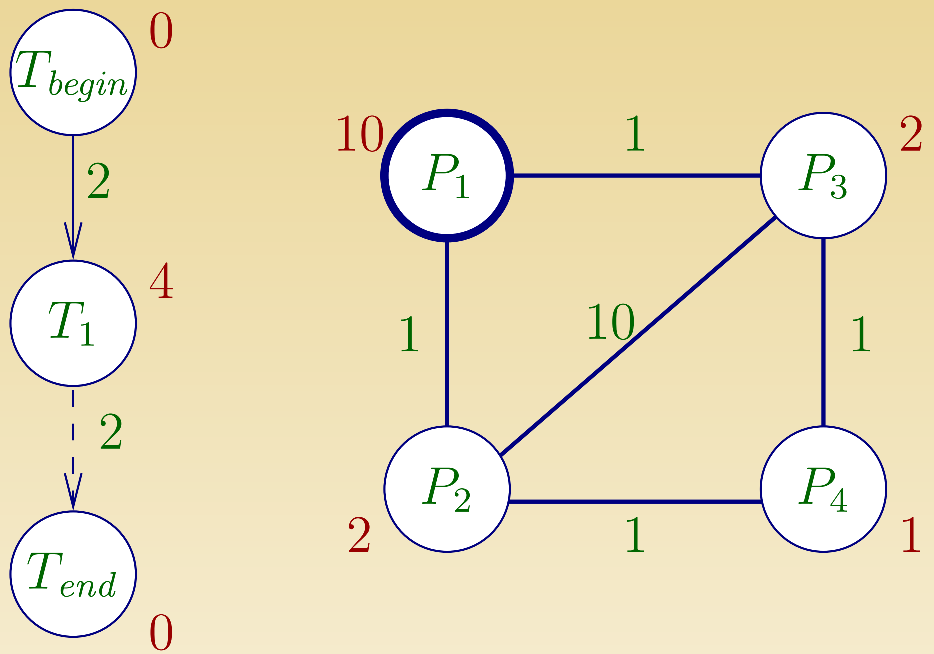


Chaque arête $e_{k,l} : T_k \rightarrow T_l$ de G_A est étiquetée par un coût de communication $data_{k,l}$ représentant la quantité de données créées par T_k et utilisées par T_l.

Pour transférer $e_{k,l}$ entre P_i et P_j, il faut donc un temps égal à $data_{k,l} \times c_{i,j}$.

Temps de communications et de calcul

Il faut $w_{i,k}$ unités de temps au processeur P_i pour traiter la tâche T_k.



Chaque arête $e_{k,l} : T_k \rightarrow T_l$ de G_A est étiquetée par un coût de communication $data_{k,l}$ représentant la quantité de données créées par T_k et utilisées par T_l.

Pour transférer $e_{k,l}$ entre P_i et P_j, il faut donc un temps égal à $data_{k,l} \times c_{i,j}$.

Un peu de vocabulaire

Définition (Allocation). Une allocation est composée d'une application $\pi : V_A \mapsto V_P$ et d'une application $\sigma : E_A \mapsto \{\text{chemin dans } G_P\}$.

Définition (Ordonnancement). Un ordonnancement associé à une allocation (π, σ) est composée d'une application $t_\pi : V_A \mapsto \mathbb{R}$ et d'une application $t_\sigma : E_A \times E_P \mapsto \mathbb{R}$ vérifiant :

- les contraintes de ressources
- les contraintes 1-port
- les contraintes de précédence

Un peu de vocabulaire

Définition (Allocation). Une allocation est composée d'une application $\pi : V_A \mapsto V_P$ et d'une application $\sigma : E_A \mapsto \{\text{chemin dans } G_P\}$.

Définition (Ordonnancement). Un ordonnancement associé à une allocation (π, σ) est composée d'une application $t_\pi : V_A \mapsto \mathbb{R}$ et d'une application $t_\sigma : E_A \times E_P \mapsto \mathbb{R}$ vérifiant :

- les contraintes de ressources
- les contraintes 1-port
- les contraintes de précédence

Un peu de vocabulaire

Définition (Allocation). Une allocation est composée d'une application $\pi : V_A \mapsto V_P$ et d'une application $\sigma : E_A \mapsto \{\text{chemin dans } G_P\}$.

Définition (Ordonnancement). Un ordonnancement associé à une allocation (π, σ) est composée d'une application $t_\pi : V_A \mapsto \mathbb{R}$ et d'une application $t_\sigma : E_A \times E_P \mapsto \mathbb{R}$ vérifiant :

- les contraintes de ressources
- les contraintes 1-port
- les contraintes de précédence

Un peu de vocabulaire

Définition (Allocation). Une allocation est composée d'une application $\pi : V_A \mapsto V_P$ et d'une application $\sigma : E_A \mapsto \{\text{chemin dans } G_P\}$.

Définition (Ordonnancement). Un ordonnancement associé à une allocation (π, σ) est composée d'une application $t_\pi : V_A \mapsto \mathbb{R}$ et d'une application $t_\sigma : E_A \times E_P \mapsto \mathbb{R}$ vérifiant :

- les contraintes de ressources
- les contraintes 1-port
- les contraintes de précédence

Un peu de vocabulaire

Définition (Allocation). Une allocation est composée d'une application $\pi : V_A \mapsto V_P$ et d'une application $\sigma : E_A \mapsto \{\text{chemin dans } G_P\}$.

Définition (Ordonnancement). Un ordonnancement associé à une allocation (π, σ) est composée d'une application $t_\pi : V_A \mapsto \mathbb{R}$ et d'une application $t_\sigma : E_A \times E_P \mapsto \mathbb{R}$ vérifiant :

- les contraintes de ressources
- les contraintes 1-port
- les contraintes de précédence

Calcul du régime permanent optimal

Définitions

$cons(P_i, T_k)$: nombre moyen de tâches de type T_k traitées par unité de temps sur le processeur P_i.

$$\forall P_i, \forall T_k \in V_A,\ 0 \leqslant cons(P_i, T_k) \times w_{i,k} \leqslant 1 \tag{1}$$

$sent(P_i \rightarrow P_j, e_{k,l})$: nombre moyen de fichiers de type $e_{k,l}$ envoyés de P_i à P_j par unité de temps.

$$\forall P_i, P_j,\ 0 \leqslant sent(P_i \rightarrow P_j, e_{k,l}) \times (data_{k,l} \times c_{i,j}) \leqslant 1 \tag{2}$$

Définitions

$cons(P_i, T_k)$: nombre moyen de tâches de type T_k traitées par unité de temps sur le processeur P_i.

$$\forall P_i, \forall T_k \in V_A,\ 0 \leqslant cons(P_i, T_k) \times w_{i,k} \leqslant 1 \qquad (1)$$

$sent(P_i \to P_j, e_{k,l})$: nombre moyen de fichiers de type $e_{k,l}$ envoyés de P_i à P_j par unité de temps.

$$\forall P_i, P_j,\ 0 \leqslant sent(P_i \to P_j, e_{k,l}) \times (data_{k,l} \times c_{i,j}) \leqslant 1 \qquad (2)$$

Définitions

$cons(P_i, T_k)$: nombre moyen de tâches de type T_k traitées par unité de temps sur le processeur P_i.

$$\forall P_i, \forall T_k \in V_A,\ 0 \leqslant cons(P_i, T_k) \times w_{i,k} \leqslant 1 \tag{1}$$

$sent(P_i \to P_j, e_{k,l})$: nombre moyen de fichiers de type $e_{k,l}$ envoyés de P_i à P_j par unité de temps.

$$\forall P_i, P_j,\ 0 \leqslant sent(P_i \to P_j, e_{k,l}) \times (data_{k,l} \times c_{i,j}) \leqslant 1 \tag{2}$$

Définitions

$cons(P_i, T_k)$: nombre moyen de tâches de type T_k traitées par unité de temps sur le processeur P_i.

$$\forall P_i, \forall T_k \in V_A,\ 0 \leqslant cons(P_i, T_k) \times w_{i,k} \leqslant 1 \qquad (1)$$

$sent(P_i \to P_j, e_{k,l})$: nombre moyen de fichiers de type $e_{k,l}$ envoyés de P_i à P_j par unité de temps.

$$\forall P_i, P_j,\ 0 \leqslant sent(P_i \to P_j, e_{k,l}) \times (data_{k,l} \times c_{i,j}) \leqslant 1 \qquad (2)$$

Équations du régime permanent

Modèle 1-port en sortie : les émissions du processeur P_i sont effectuées séquentiellement vers ses voisins.

$$\forall P_i, \sum_{P_i \to P_j} \sum_{e_{k,l} \in E_A} \left(sent(P_i \to P_j, e_{k,l}) \times data_{k,l} \times c_{i,j} \right) \leqslant 1 \qquad (3)$$

Modèle 1-port en entrée : les réceptions du processeur P_i sont effectuées séquentiellement.

$$\forall P_i, \sum_{P_j \to P_i} \sum_{e_{k,l} \in E_A} \left(sent(P_j \to P_i, e_{k,l}) \times data_{k,l} \times c_{j,i} \right) \leqslant 1 \qquad (4)$$

Recouvrement des calculs et des communications : les calculs et les communications sont indépendants.

$$\forall P_i, \sum_{T_k \in V_A} cons(P_i, T_k) \times w_{i,k} \leqslant 1 \qquad (5)$$

Équations du régime permanent

Modèle 1-port en sortie : les émissions du processeur P_i sont effectuées séquentiellement vers ses voisins.

$$\forall P_i, \sum_{P_i \to P_j} \sum_{e_{k,l} \in E_A} \left(sent(P_i \to P_j, e_{k,l}) \times data_{k,l} \times c_{i,j} \right) \leqslant 1 \tag{3}$$

Modèle 1-port en entrée : les réceptions du processeur P_i sont effectuées séquentiellement.

$$\forall P_i, \sum_{P_j \to P_i} \sum_{e_{k,l} \in E_A} \left(sent(P_j \to P_i, e_{k,l}) \times data_{k,l} \times c_{j,i} \right) \leqslant 1 \tag{4}$$

Recouvrement des calculs et des communications : les calculs et les communications sont indépendants.

$$\forall P_i, \sum_{T_k \in V_A} cons(P_i, T_k) \times w_{i,k} \leqslant 1 \tag{5}$$

Équations du régime permanent

Modèle 1-port en sortie : les émissions du processeur P_i sont effectuées séquentiellement vers ses voisins.

$$\forall P_i, \sum_{P_i \to P_j} \sum_{e_{k,l} \in E_A} \left(sent(P_i \to P_j, e_{k,l}) \times data_{k,l} \times c_{i,j} \right) \leqslant 1 \tag{3}$$

Modèle 1-port en entrée : les réceptions du processeur P_i sont effectuées séquentiellement.

$$\forall P_i, \sum_{P_j \to P_i} \sum_{e_{k,l} \in E_A} \left(sent(P_j \to P_i, e_{k,l}) \times data_{k,l} \times c_{j,i} \right) \leqslant 1 \tag{4}$$

Recouvrement des calculs et des communications : les calculs et les communications sont indépendants.

$$\forall P_i, \sum_{T_k \in V_A} cons(P_i, T_k) \times w_{i,k} \leqslant 1 \tag{5}$$

Équations du régime permanent

Modèle 1-port en sortie : les émissions du processeur P_i sont effectuées séquentiellement vers ses voisins.

$$\forall P_i, \sum_{P_i \to P_j} \sum_{e_{k,l} \in E_A} \left(sent(P_i \to P_j, e_{k,l}) \times data_{k,l} \times c_{i,j} \right) \leqslant 1 \tag{3}$$

Modèle 1-port en entrée : les réceptions du processeur P_i sont effectuées séquentiellement.

$$\forall P_i, \sum_{P_j \to P_i} \sum_{e_{k,l} \in E_A} \left(sent(P_j \to P_i, e_{k,l}) \times data_{k,l} \times c_{j,i} \right) \leqslant 1 \tag{4}$$

Recouvrement des calculs et des communications : les calculs et les communications sont indépendants.

$$\forall P_i, \sum_{T_k \in V_A} cons(P_i, T_k) \times w_{i,k} \leqslant 1 \tag{5}$$

Équations du régime permanent

Modèle 1-port en sortie : les émissions du processeur P_i sont effectuées séquentiellement vers ses voisins.

$$\forall P_i, \sum_{P_i \to P_j} \sum_{e_{k,l} \in E_A} \left(sent(P_i \to P_j, e_{k,l}) \times data_{k,l} \times c_{i,j} \right) \leqslant 1 \quad (3)$$

Modèle 1-port en entrée : les réceptions du processeur P_i sont effectuées séquentiellement.

$$\forall P_i, \sum_{P_j \to P_i} \sum_{e_{k,l} \in E_A} \left(sent(P_j \to P_i, e_{k,l}) \times data_{k,l} \times c_{j,i} \right) \leqslant 1 \quad (4)$$

Recouvrement des calculs et des communications : les calculs et les communications sont indépendants.

$$\forall P_i, \sum_{T_k \in V_A} cons(P_i, T_k) \times w_{i,k} \leqslant 1 \quad (5)$$

Équations du régime permanent

Modèle 1-port en sortie : les émissions du processeur P_i sont effectuées séquentiellement vers ses voisins.

$$\forall P_i, \sum_{P_i \to P_j} \sum_{e_{k,l} \in E_A} \left(sent(P_i \to P_j, e_{k,l}) \times data_{k,l} \times c_{i,j} \right) \leqslant 1 \tag{3}$$

Modèle 1-port en entrée : les réceptions du processeur P_i sont effectuées séquentiellement.

$$\forall P_i, \sum_{P_j \to P_i} \sum_{e_{k,l} \in E_A} \left(sent(P_j \to P_i, e_{k,l}) \times data_{k,l} \times c_{j,i} \right) \leqslant 1 \tag{4}$$

Recouvrement des calculs et des communications : les calculs et les communications sont indépendants.

$$\forall P_i, \sum_{T_k \in V_A} cons(P_i, T_k) \times w_{i,k} \leqslant 1 \tag{5}$$

Loi de conservation

Soit P_i un processeur et $e_{k,l}$ une arête du graphe de tâches.

Reçues : $\displaystyle\sum_{P_j \to P_i} sent(P_j \to P_i, e_{k,l})$

Créées : $cons(P_i, T_k)$

Consommées : $cons(P_i, T_l)$

Émises : $\displaystyle\sum_{P_i \to P_j} sent(P_i \to P_j, e_{k,l})$

$$\forall P_i, \forall e_{k,l} : T_k \to T_l \in E_A,$$
$$\sum_{P_j \to P_i} sent(P_j \to P_i, e_{k,l}) + cons(P_i, T_k) = \sum_{P_i \to P_j} sent(P_i \to P_j, e_{k,l}) + cons(P_i, T_l) \quad (6)$$

Loi de conservation

Soit P_i un processeur et $e_{k,l}$ une arête du graphe de tâches.

Reçues : $\sum_{P_j \to P_i} sent(P_j \to P_i, e_{k,l})$

Créées : $cons(P_i, T_k)$

Consommées : $cons(P_i, T_l)$

Émises : $\sum_{P_i \to P_j} sent(P_i \to P_j, e_{k,l})$

$$\forall P_i, \forall e_{k,l} : T_k \to T_l \in E_A,$$
$$\sum_{P_j \to P_i} sent(P_j \to P_i, e_{k,l}) + cons(P_i, T_k) = \sum_{P_i \to P_j} sent(P_i \to P_j, e_{k,l}) + cons(P_i, T_l) \quad (6)$$

Loi de conservation

Soit P_i un processeur et $e_{k,l}$ une arête du graphe de tâches.

Reçues : $\displaystyle\sum_{P_j \to P_i} sent(P_j \to P_i, e_{k,l})$

Créées : $cons(P_i, T_k)$

Consommées : $cons(P_i, T_l)$

Émises : $\displaystyle\sum_{P_i \to P_j} sent(P_i \to P_j, e_{k,l})$

$$\forall P_i, \forall e_{k,l} : T_k \to T_l \in E_A,$$
$$\sum_{P_j \to P_i} sent(P_j \to P_i, e_{k,l}) + cons(P_i, T_k) = \sum_{P_i \to P_j} sent(P_i \to P_j, e_{k,l}) + cons(P_i, T_l) \quad (6)$$

Loi de conservation

Soit P_i un processeur et $e_{k,l}$ une arête du graphe de tâches.

Reçues : $\sum_{P_j \to P_i} sent(P_j \to P_i, e_{k,l})$

Créées : $cons(P_i, T_k)$

Consommées : $cons(P_i, T_l)$

Émises : $\sum_{P_i \to P_j} sent(P_i \to P_j, e_{k,l})$

$$\forall P_i, \forall e_{k,l} : T_k \to T_l \in E_A, \quad \sum_{P_j \to P_i} sent(P_j \to P_i, e_{k,l}) + cons(P_i, T_k) = \sum_{P_i \to P_j} sent(P_i \to P_j, e_{k,l}) + cons(P_i, T_l) \tag{6}$$

Loi de conservation

Soit P_i un processeur et $e_{k,l}$ une arête du graphe de tâches.

Reçues : $\displaystyle\sum_{P_j \to P_i} sent(P_j \to P_i, e_{k,l})$

Créées : $cons(P_i, T_k)$

Consommées : $cons(P_i, T_l)$

Émises : $\displaystyle\sum_{P_i \to P_j} sent(P_i \to P_j, e_{k,l})$

$$\forall P_i, \forall e_{k,l} : T_k \to T_l \in E_A,$$
$$\sum_{P_j \to P_i} sent(P_j \to P_i, e_{k,l}) + cons(P_i, T_k) =$$
$$\sum_{P_i \to P_j} sent(P_i \to P_j, e_{k,l}) + cons(P_i, T_l) \quad (6)$$

Loi de conservation

Soit P_i un processeur et $e_{k,l}$ une arête du graphe de tâches.

Reçues : $\displaystyle\sum_{P_j \to P_i} sent(P_j \to P_i, e_{k,l})$

Créées : $cons(P_i, T_k)$

Consommées : $cons(P_i, T_l)$

Émises : $\displaystyle\sum_{P_i \to P_j} sent(P_i \to P_j, e_{k,l})$

$$\forall P_i, \forall e_{k,l} : T_k \to T_l \in E_A,$$
$$\sum_{P_j \to P_i} sent(P_j \to P_i, e_{k,l}) + cons(P_i, T_k) =$$
$$\sum_{P_i \to P_j} sent(P_i \to P_j, e_{k,l}) + cons(P_i, T_l) \quad (6)$$

Borne supérieure du débit d'un ordonnancement cyclique

MAXIMISER $\rho = \sum_{i=1}^{p} cons(P_i, T_{end})$,

SOUS LES CONTRAINTES

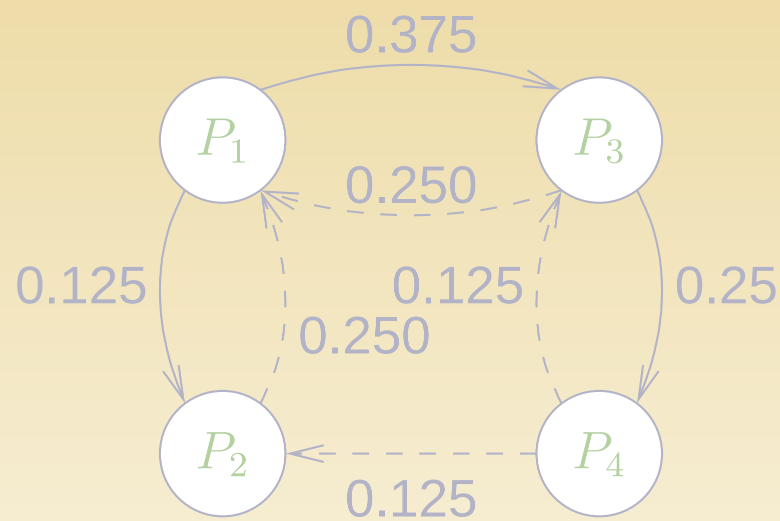
$$
\left\{
\begin{array}{ll}
\text{(7a)} & \forall P_i, \forall T_k \in V_A,\ 0 \leqslant cons(P_i, T_k) \times w_{i,k} \leqslant 1 \\
\text{(7b)} & \forall P_i, P_j,\ 0 \leqslant sent(P_i \to P_j, e_{k,l}) \times (data_{k,l} \times c_{i,j}) \leqslant 1 \\
\text{(7c)} & \forall P_i,\ \displaystyle\sum_{P_i \to P_j} \sum_{e_{k,l} \in E_A} \big(sent(P_i \to P_j, e_{k,l}) \times data_{k,l} \times c_{i,j} \big) \leqslant 1 \\
\text{(7d)} & \forall P_i,\ \displaystyle\sum_{P_j \to P_i} \sum_{e_{k,l} \in E_A} \big(sent(P_j \to P_i, e_{k,l}) \times data_{k,l} \times c_{j,i} \big) \leqslant 1 \\
\text{(7e)} & \forall P_i,\ \displaystyle\sum_{T_k \in V_A} cons(P_i, T_k) \times w_{i,k} \leqslant 1 \\
\text{(7f)} & \forall P_i, \forall e_{k,l} \in E_A : T_k \to T_l, \\
& \qquad \displaystyle\sum_{P_j \to P_i} sent(P_j \to P_i, e_{k,l}) + cons(P_i, T_k) = \\
& \qquad\qquad \displaystyle\sum_{P_i \to P_j} sent(P_i \to P_j, e_{k,l}) + cons(P_i, T_l)
\end{array}
\right.
$$

Retour sur notre exemple

Répartition des calculs

	$cons(P_i, T_1)$
P_1	0.025
P_2	0.125
P_3	0.125
P_4	0.250
Total	21 tâches/ 40 secondes

Communications

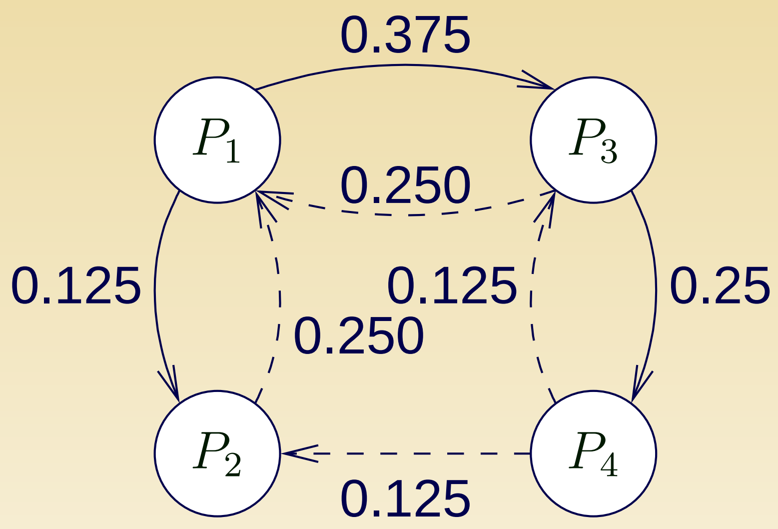


Retour sur notre exemple

Répartition des calculs

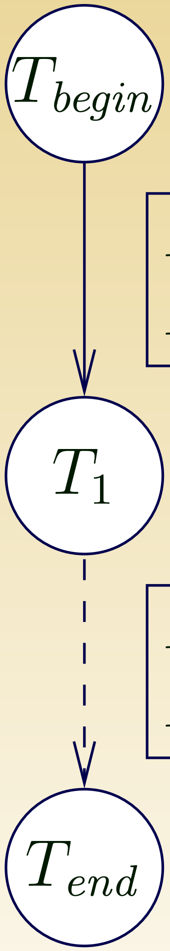
	$cons(P_i, T_1)$
P_1	0.025
P_2	0.125
P_3	0.125
P_4	0.250
Total	21 tâches/ 40 secondes

Communications



Décomposition en allocations

Régime permanent = superposition de plusieurs allocations.



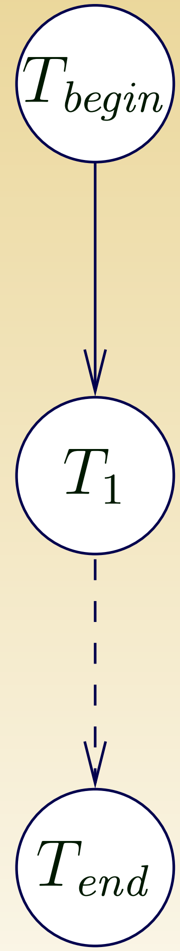
$$\boxed{P_1 : 0.525}$$

$$\boxed{\begin{array}{ll} P_1 \rightarrow P_2 : 0.125 & P_3 \rightarrow P_4 : 0.250 \\ P_1 \rightarrow P_3 : 0.375 & \end{array}}$$

$$\boxed{\begin{array}{ll} P_1 : 0.025 & P_3 : 0.125 \\ P_2 : 0.125 & P_4 : 0.250 \end{array}}$$

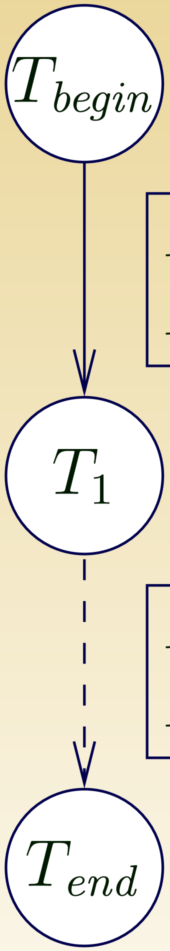
$$\boxed{\begin{array}{ll} P_4 \rightarrow P_2 : 0.125 & P_3 \rightarrow P_1 : 0.250 \\ P_4 \rightarrow P_3 : 0.125 & P_2 \rightarrow P_1 : 0.250 \end{array}}$$

$$\boxed{P_1 : 0.525}$$



Décomposition en allocations

Régime permanent = superposition de plusieurs allocations.



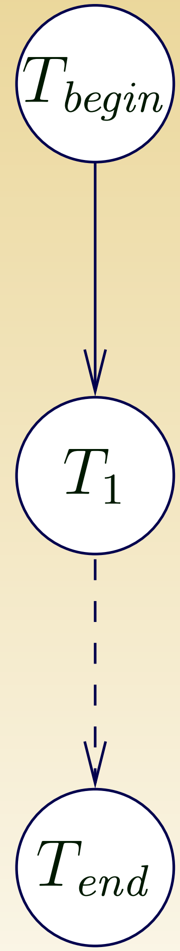
$\boxed{P_1 : 0.525}$

$\boxed{\begin{array}{ll} P_1 \rightarrow P_2 : 0.125 & P_3 \rightarrow P_4 : 0.250 \\ P_1 \rightarrow P_3 : 0.375 & \end{array}}$

$\boxed{\begin{array}{ll} P_1 : 0.025 & P_3 : 0.125 \\ P_2 : 0.125 & P_4 : 0.250 \end{array}}$

$\boxed{\begin{array}{ll} P_4 \rightarrow P_2 : 0.125 & P_3 \rightarrow P_1 : 0.250 \\ P_4 \rightarrow P_3 : 0.125 & P_2 \rightarrow P_1 : 0.250 \end{array}}$

$\boxed{P_1 : 0.525}$



P_1

Décomposition en allocations

Régime permanent = superposition de plusieurs allocations.

T_{begin} $\boxed{P_1 : 0.525}$

$$\boxed{\begin{array}{ll} P_1 \rightarrow P_2 : 0.125 & P_3 \rightarrow P_4 : 0.250 \\ P_1 \rightarrow P_3 : 0.375 & \end{array}}$$

T_1 $\boxed{\begin{array}{ll} P_1 : 0.025 & P_3 : 0.125 \\ P_2 : 0.125 & P_4 : 0.250 \end{array}}$

$$\boxed{\begin{array}{ll} P_4 \rightarrow P_2 : 0.125 & P_3 \rightarrow P_1 : 0.250 \\ P_4 \rightarrow P_3 : 0.125 & P_2 \rightarrow P_1 : 0.250 \end{array}}$$

T_{end} $\boxed{P_1 : 0.525}$

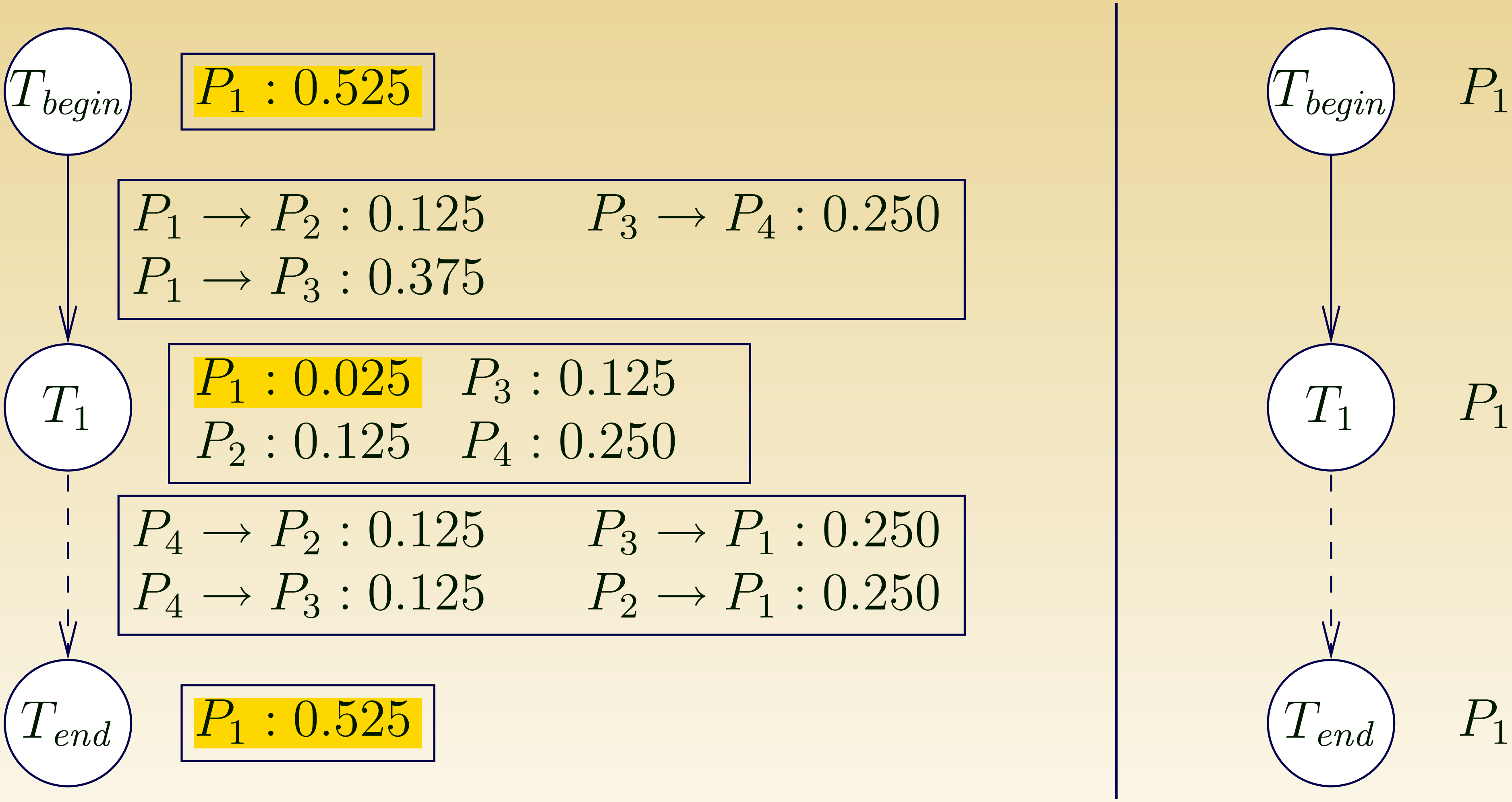
T_{begin}

T_1 P_1

T_{end} P_1

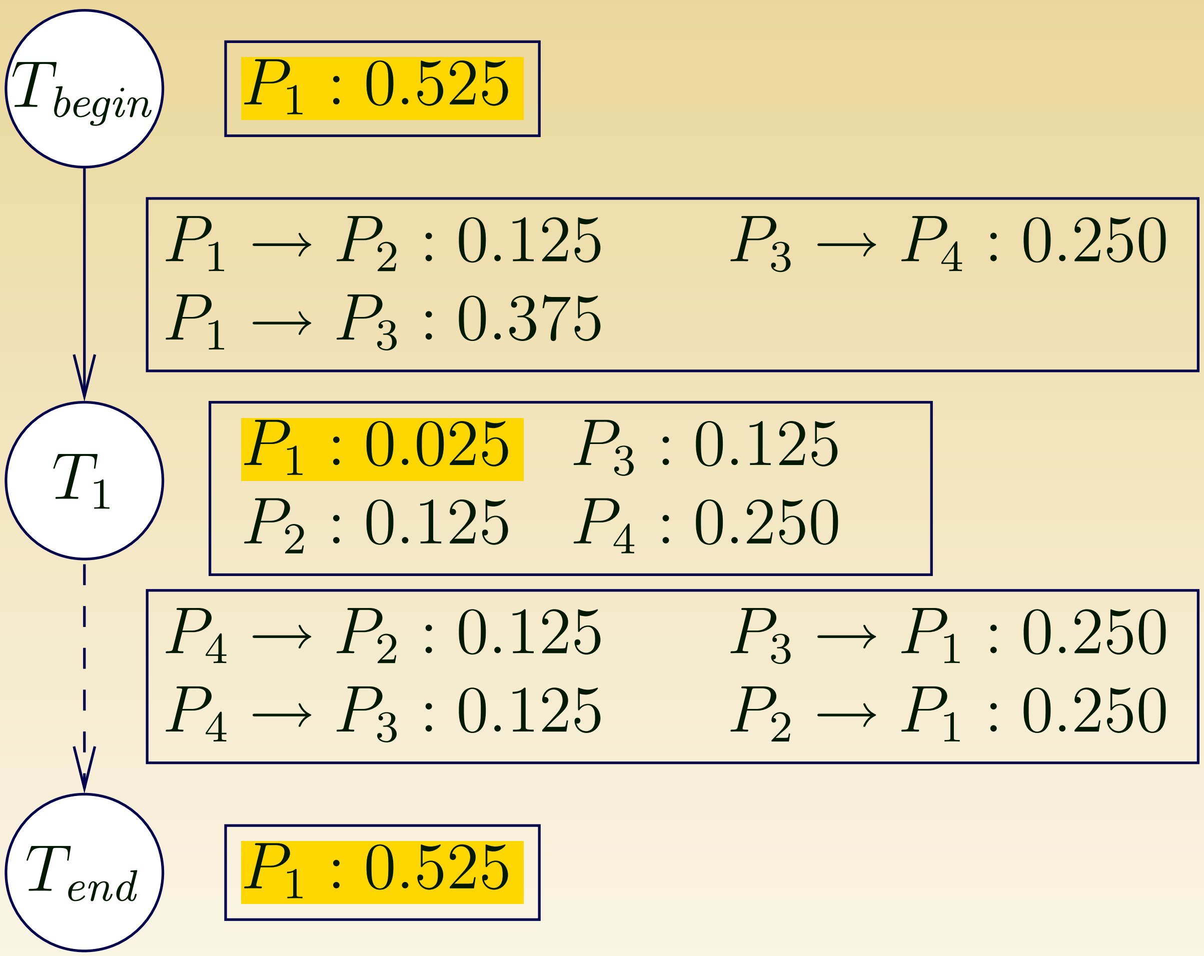
Décomposition en allocations

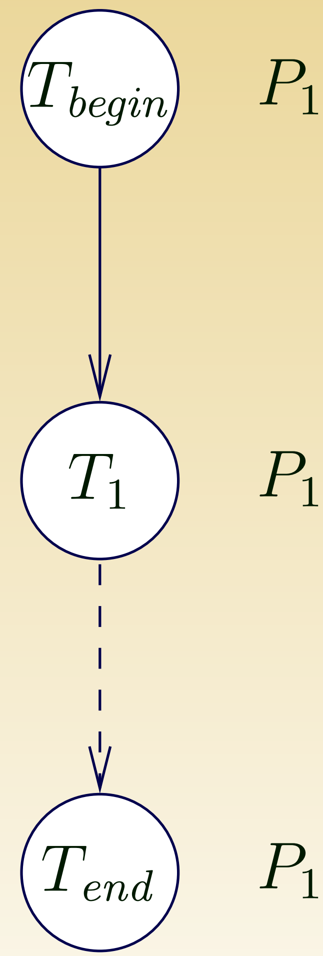
Régime permanent = superposition de plusieurs allocations.



Décomposition en allocations

Régime permanent = superposition de plusieurs allocations.

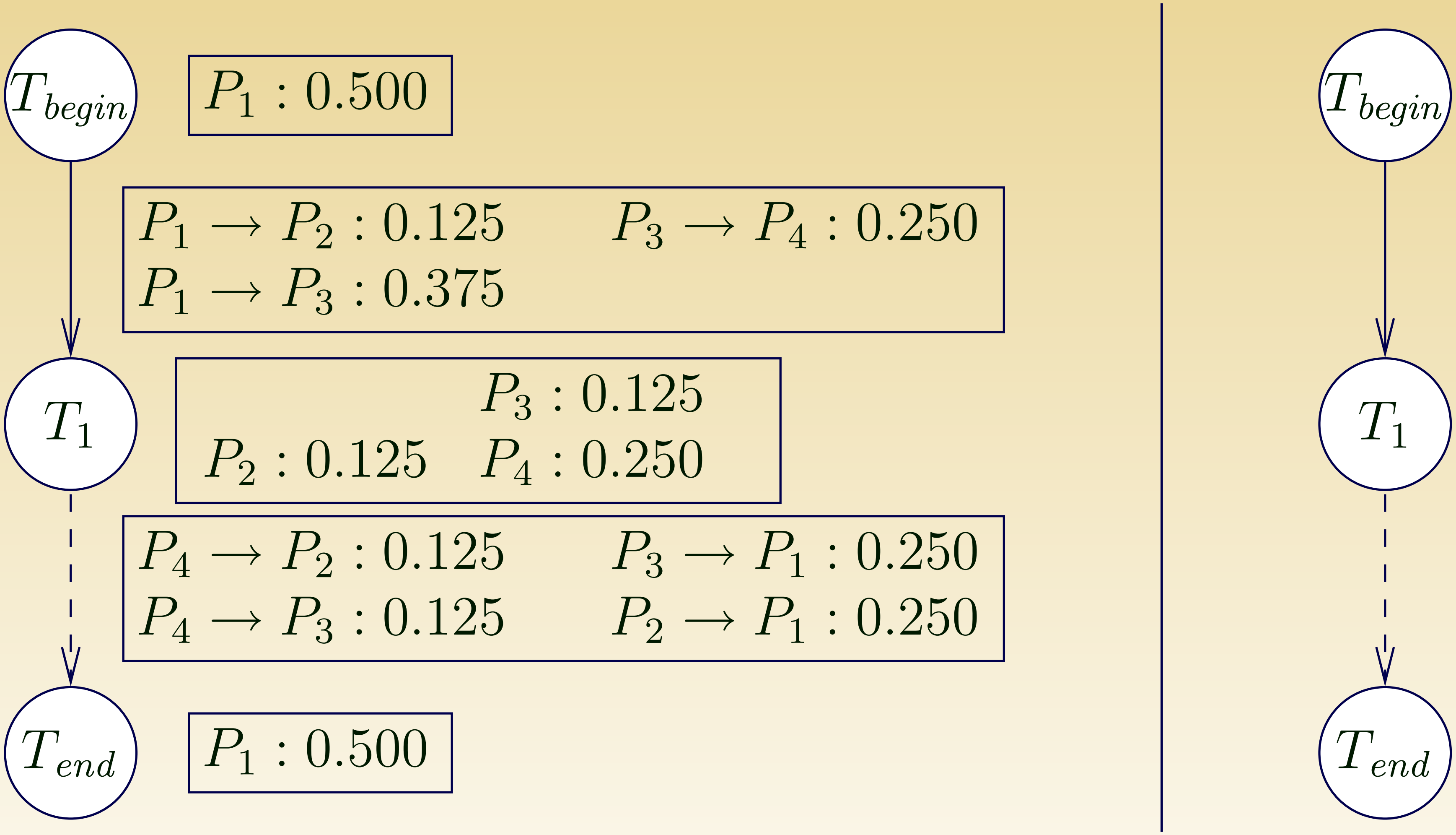




$\mathcal{A}_1 : 0.025$

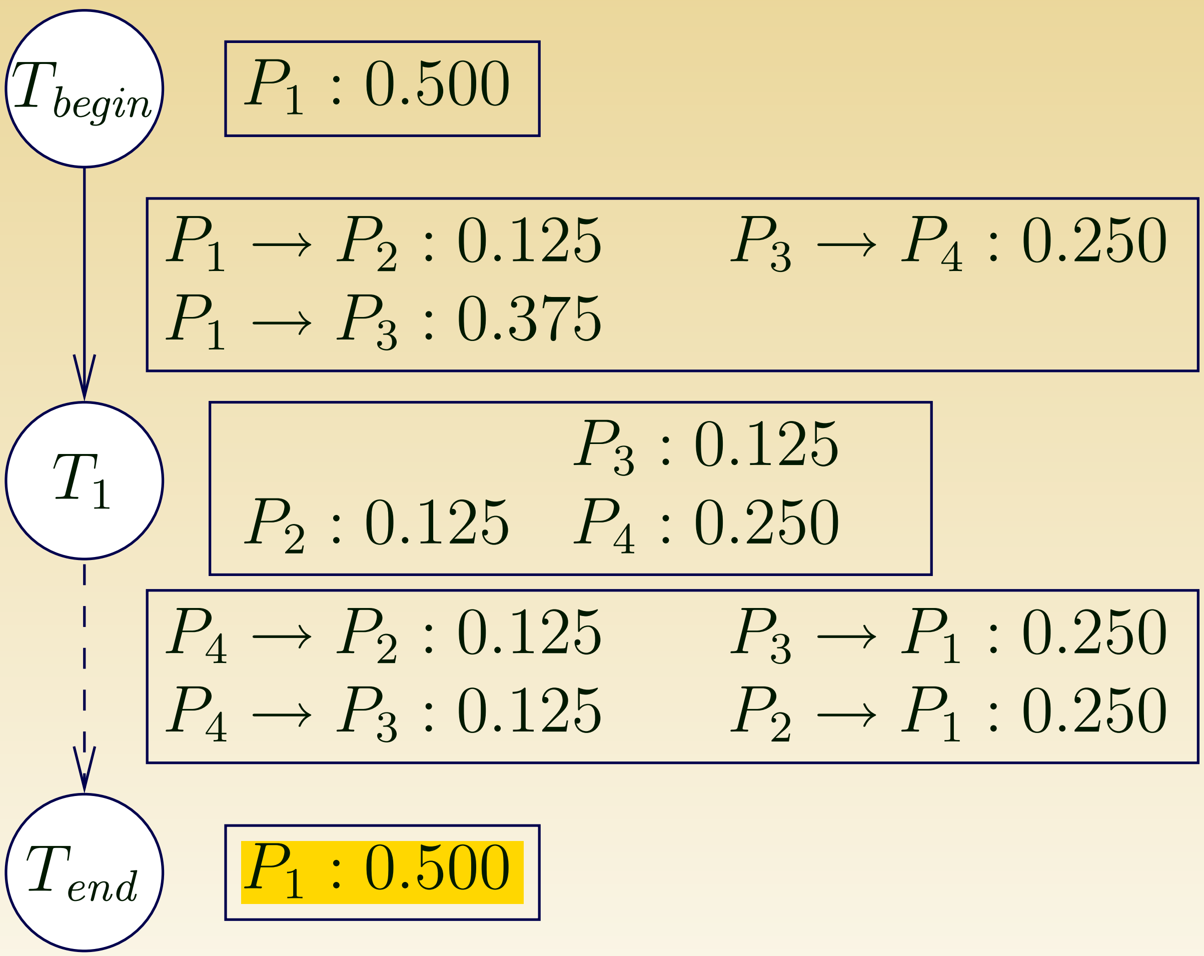
Décomposition en allocations

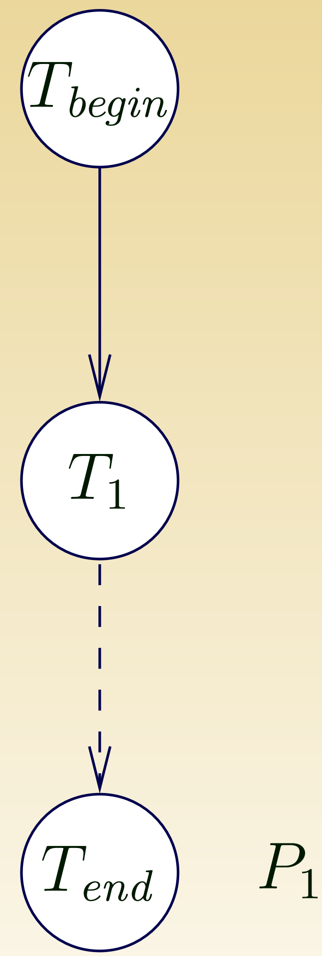
Régime permanent = superposition de plusieurs allocations.



Décomposition en allocations

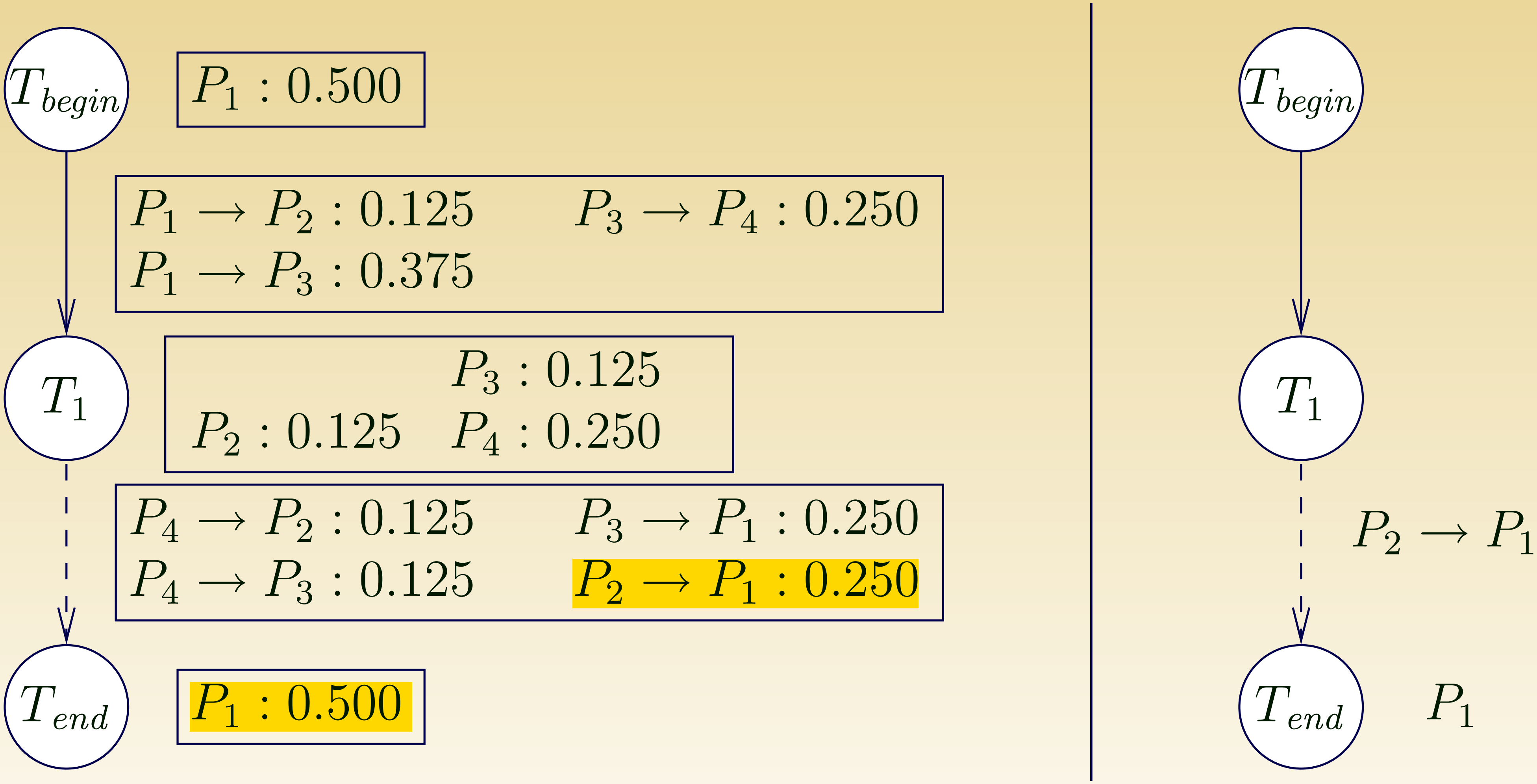
Régime permanent = superposition de plusieurs allocations.





Décomposition en allocations

Régime permanent = superposition de plusieurs allocations.



Décomposition en allocations

Régime permanent = superposition de plusieurs allocations.

T_{begin}

$P_1 : 0.500$

$P_1 \longrightarrow P_2 : 0.125 \qquad P_3 \longrightarrow P_4 : 0.250$

$P_1 \longrightarrow P_3 : 0.375$

T_1

$P_3 : 0.125$

$P_2 : 0.125 \qquad P_4 : 0.250$

$P_4 \longrightarrow P_2 : 0.125 \qquad P_3 \longrightarrow P_1 : 0.250$

$P_4 \longrightarrow P_3 : 0.125 \qquad P_2 \longrightarrow P_1 : 0.250$

T_{end}

$P_1 : 0.500$

T_{begin}

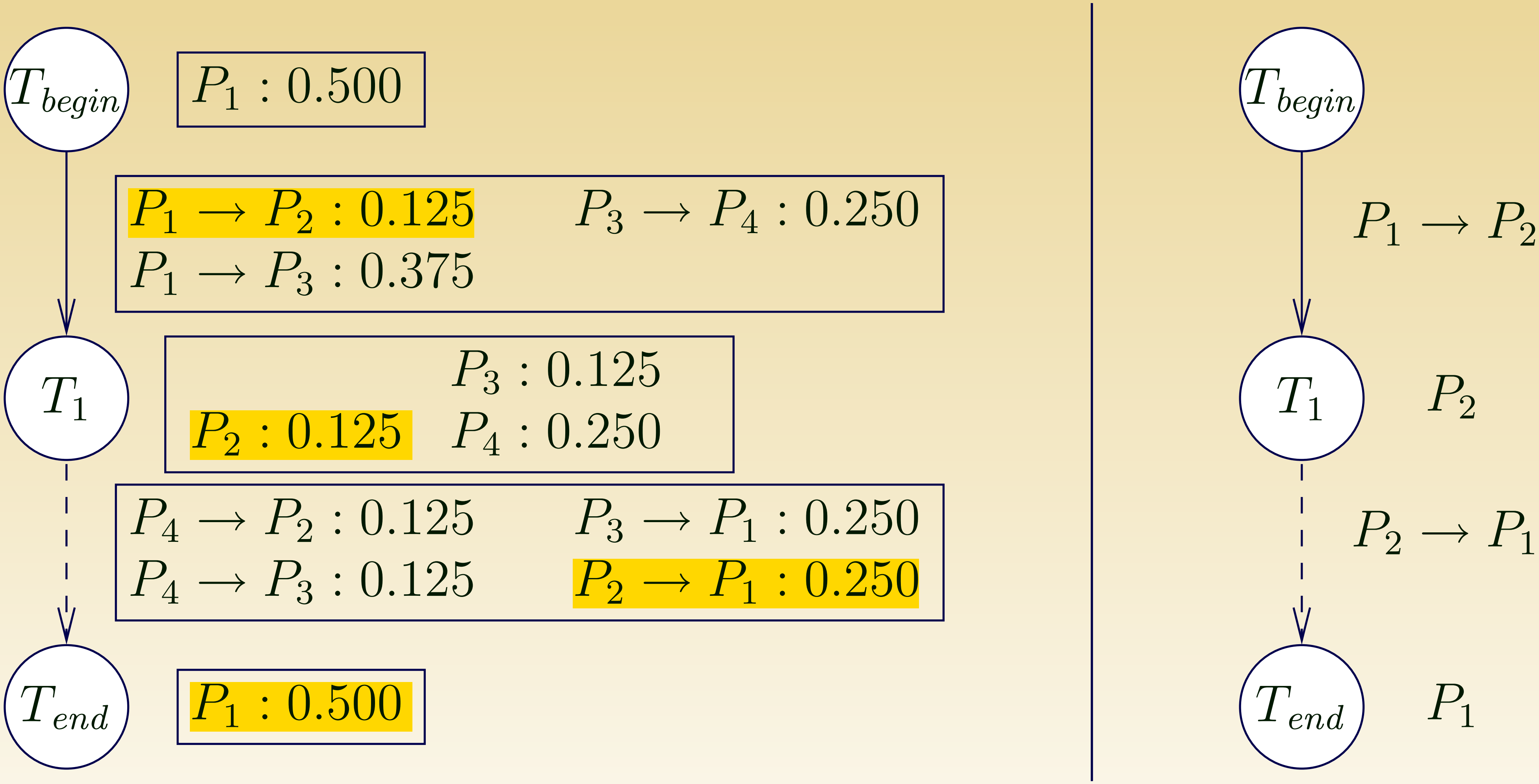
$T_1 \qquad P_2$

$P_2 \longrightarrow P_1$

$T_{end} \qquad P_1$

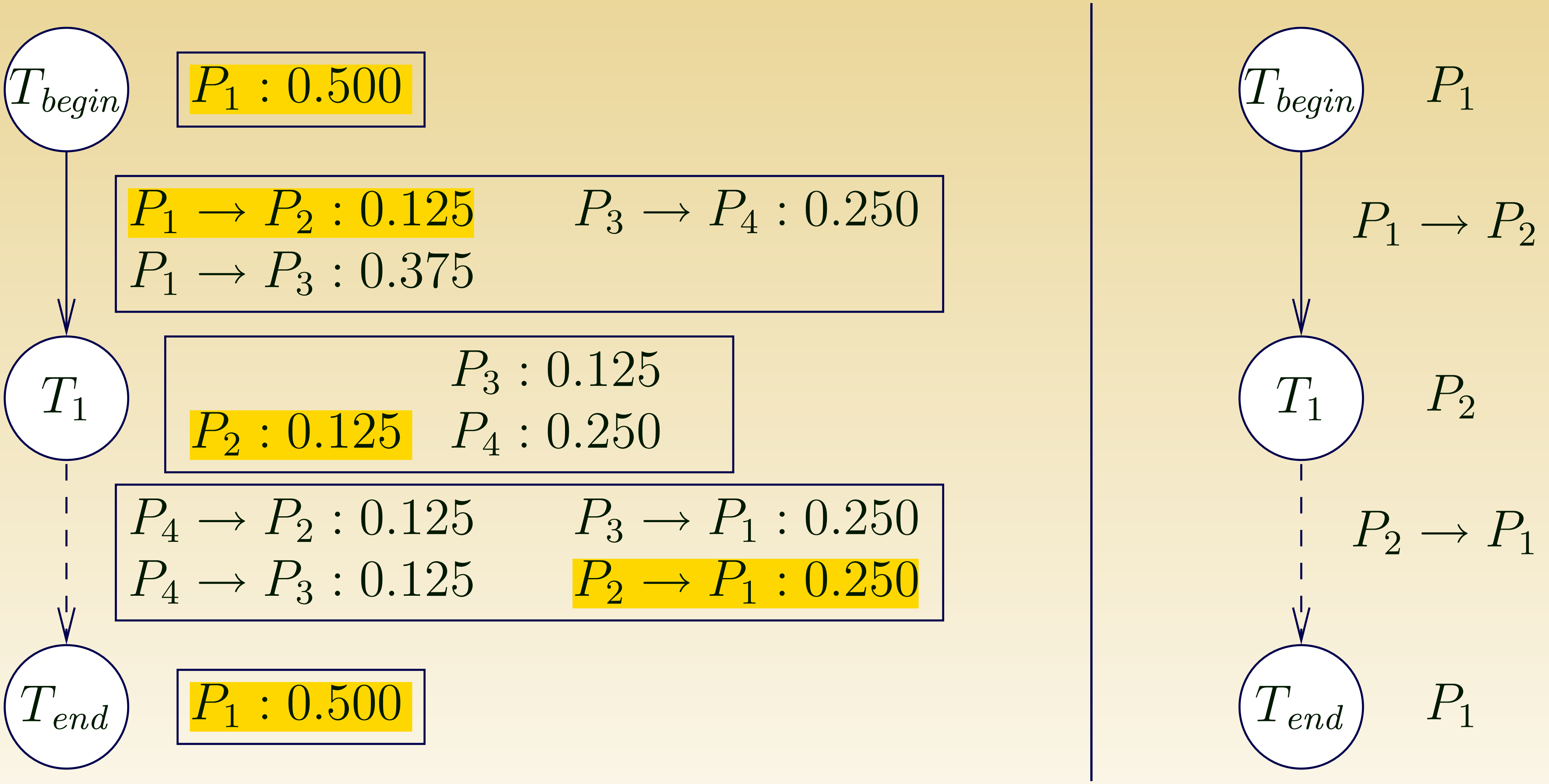
Décomposition en allocations

Régime permanent = superposition de plusieurs allocations.



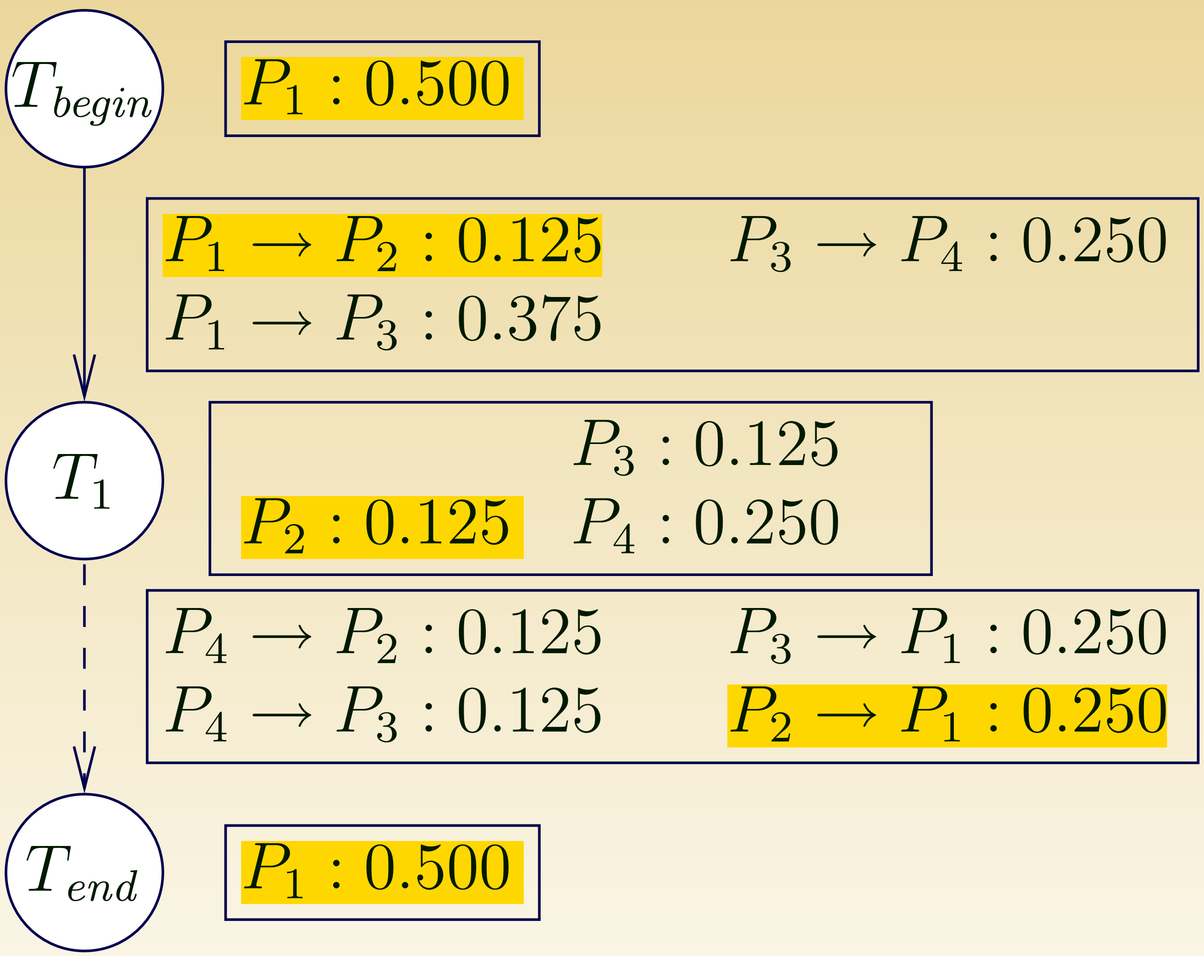
Décomposition en allocations

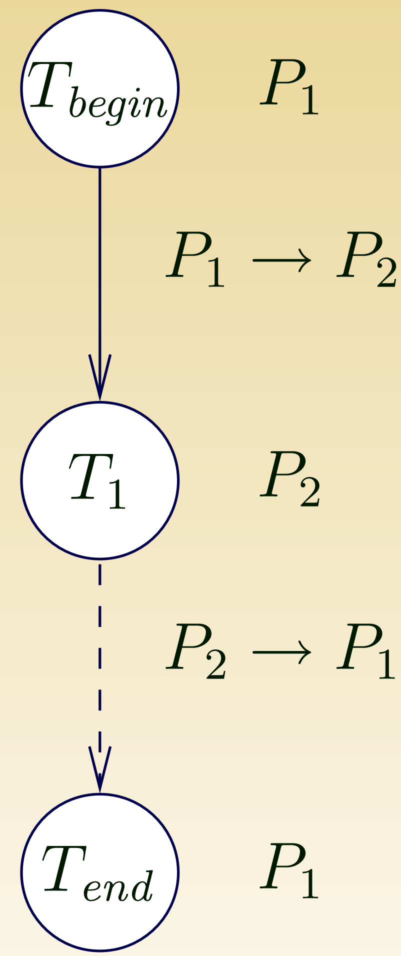
Régime permanent = superposition de plusieurs allocations.



Décomposition en allocations

Régime permanent = superposition de plusieurs allocations.

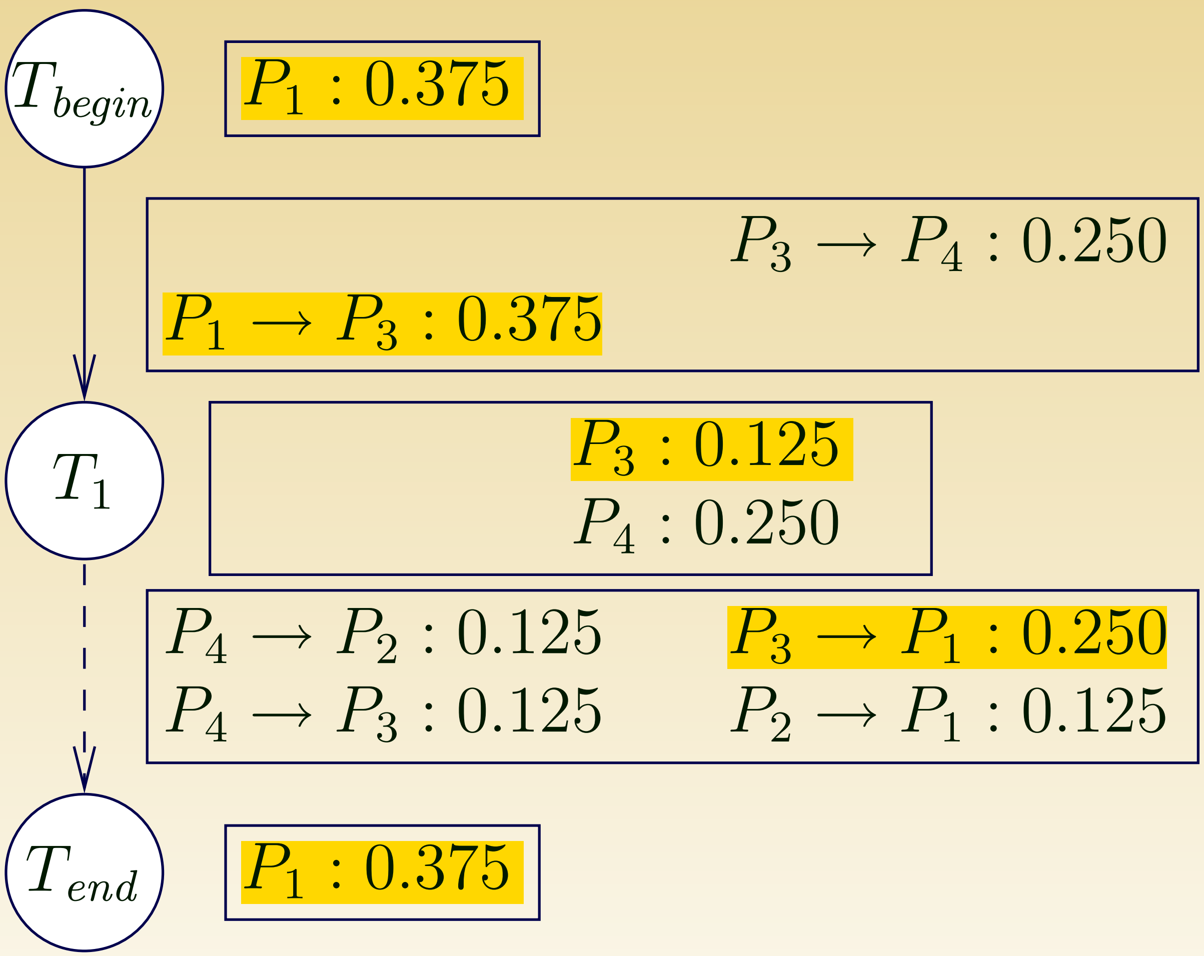


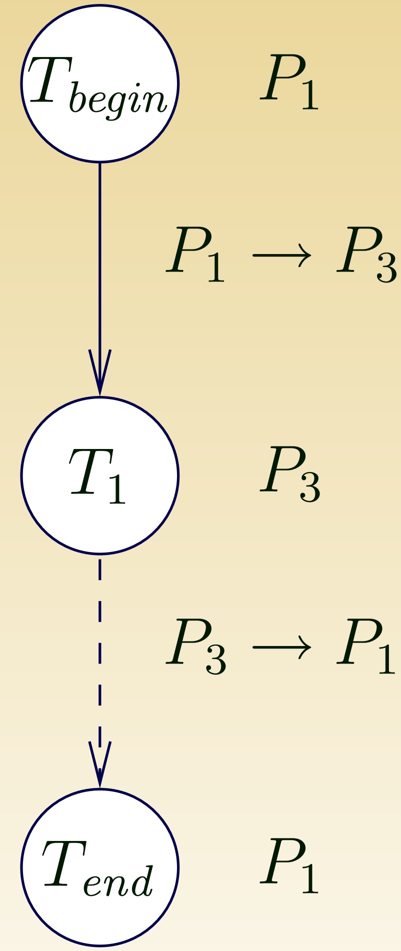


$\mathcal{A}_2 : 0.125$

Décomposition en allocations

Régime permanent = superposition de plusieurs allocations.

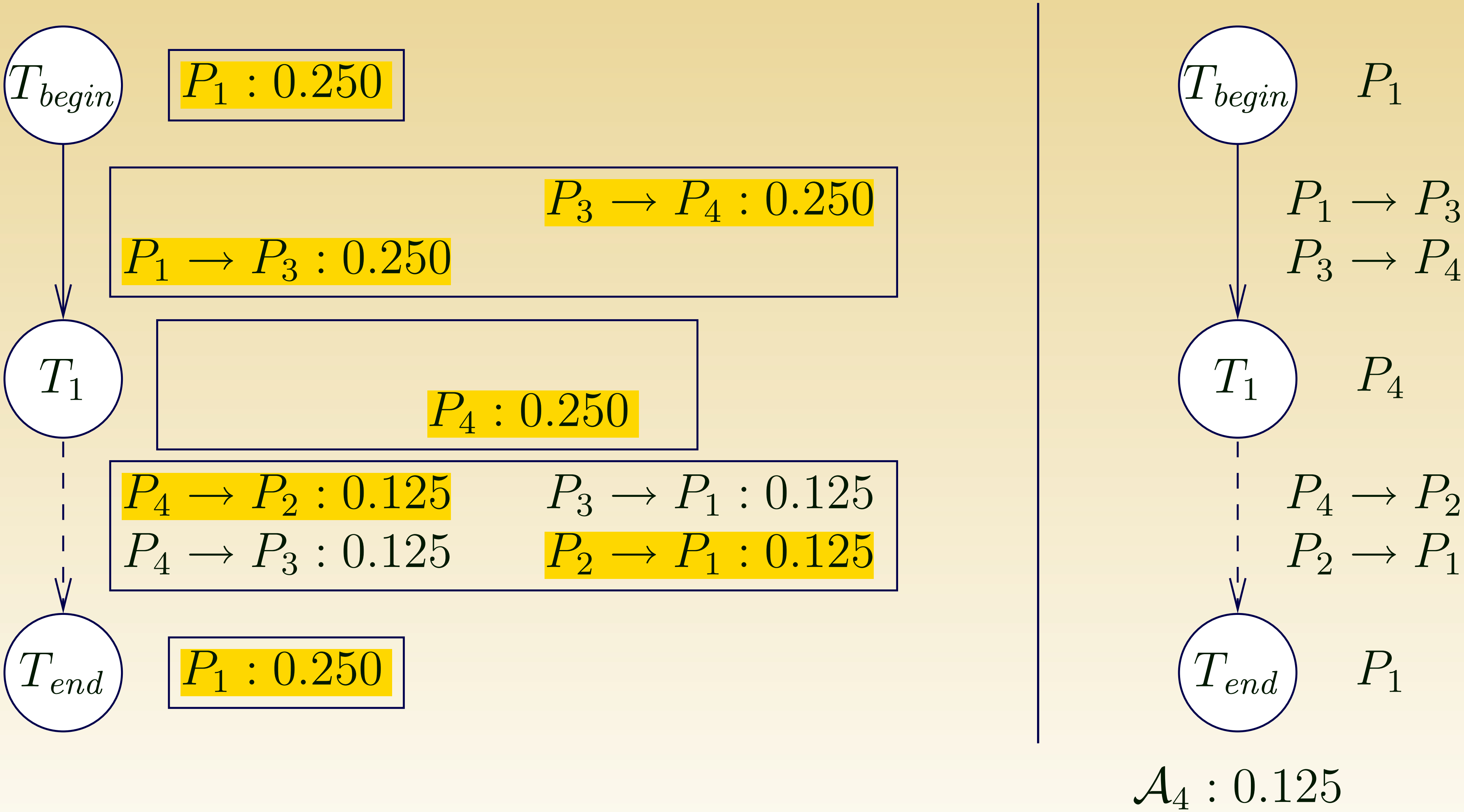




$\mathcal{A}_3 : 0.125$

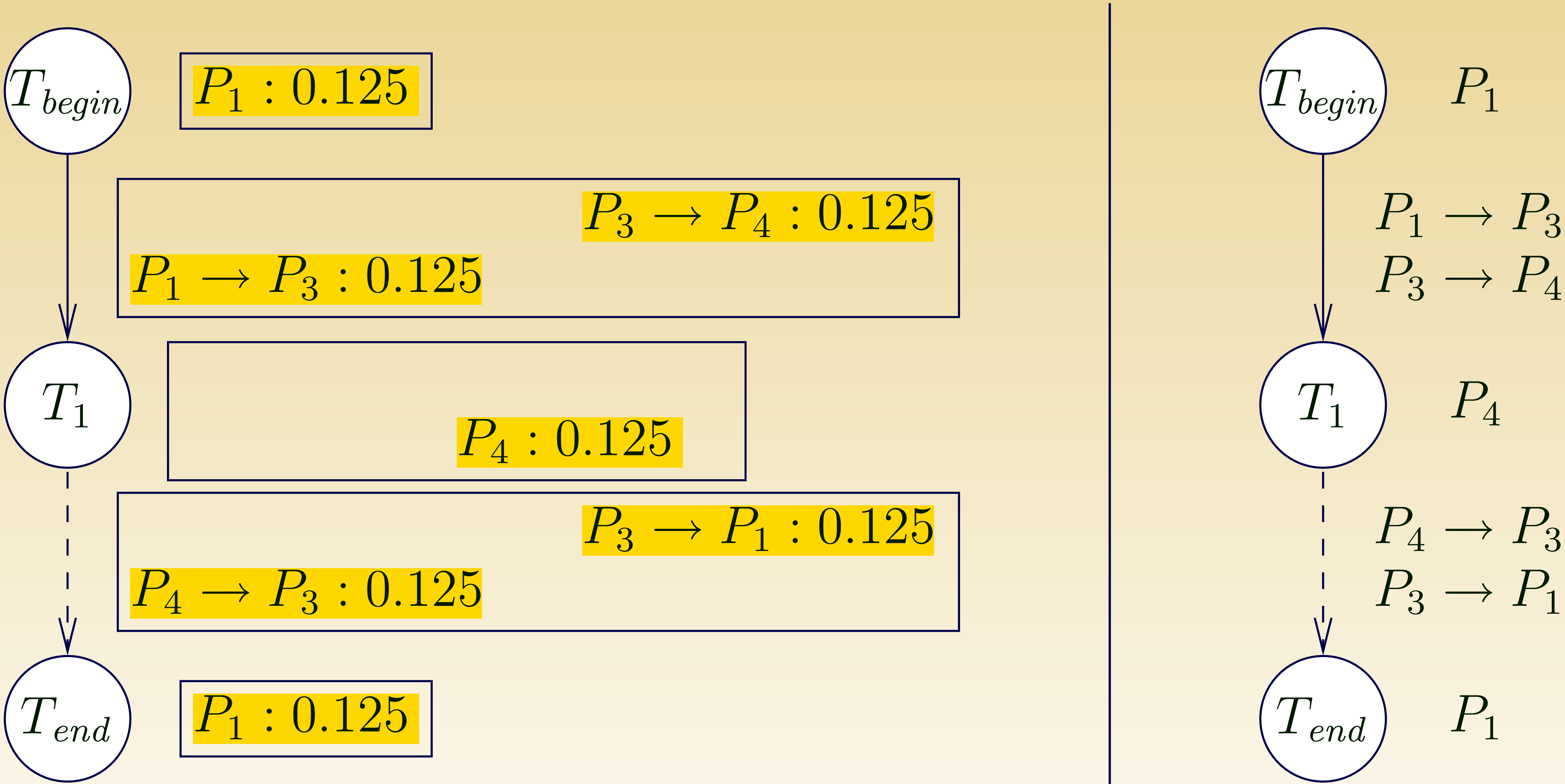
Décomposition en allocations

Régime permanent = superposition de plusieurs allocations.



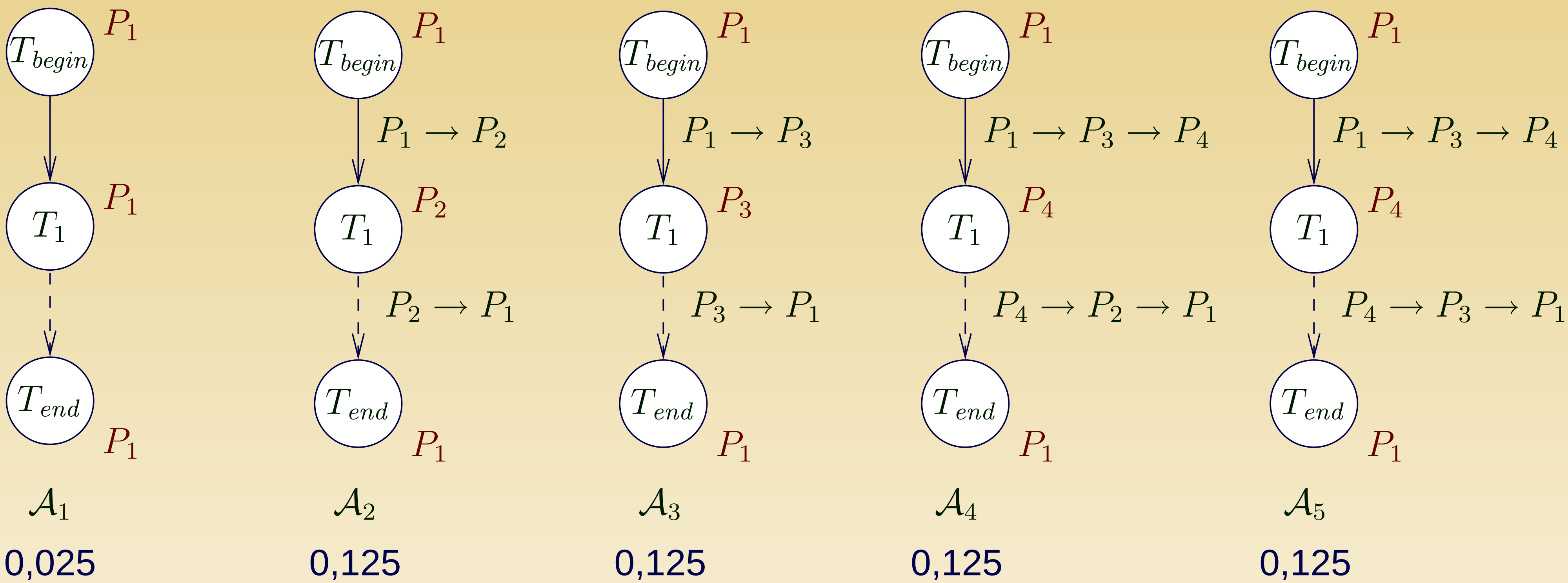
Décomposition en allocations

Régime permanent = superposition de plusieurs allocations.

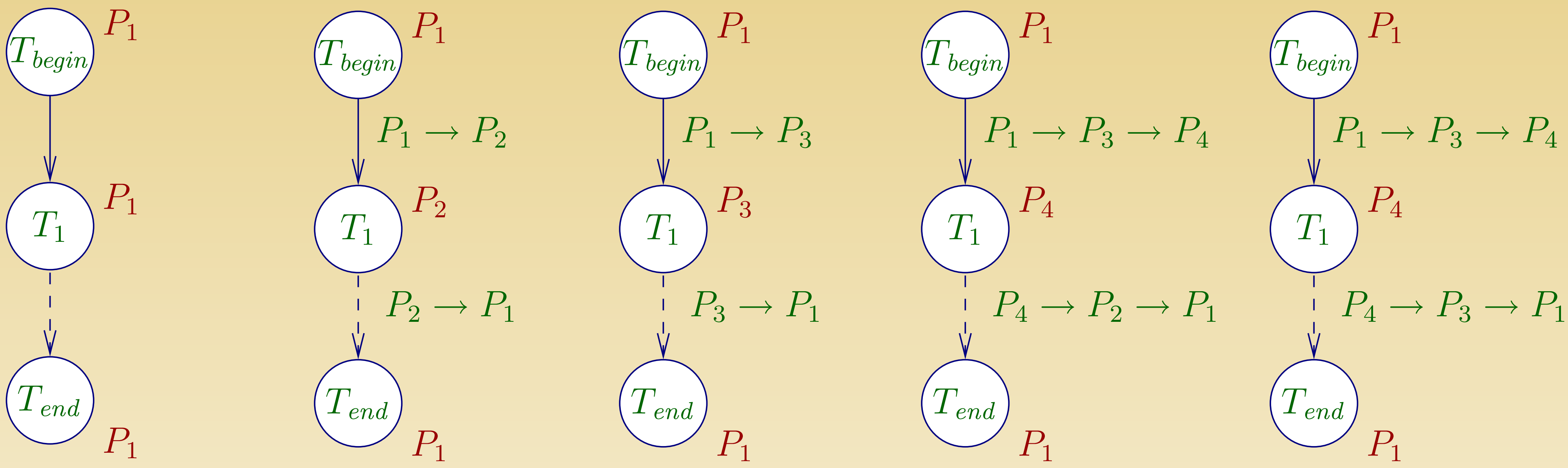


$\mathcal{A}_5 : 0.125$

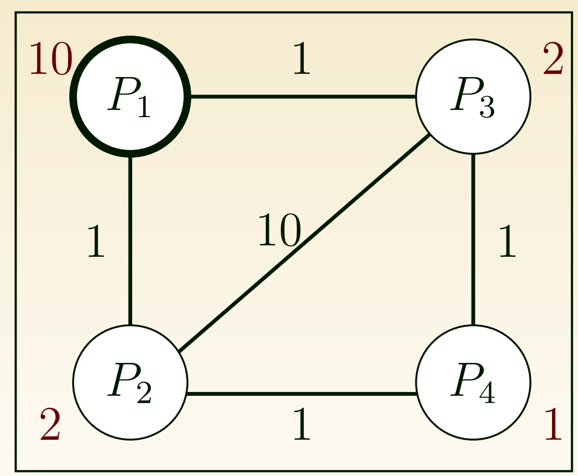
Décomposition en allocations



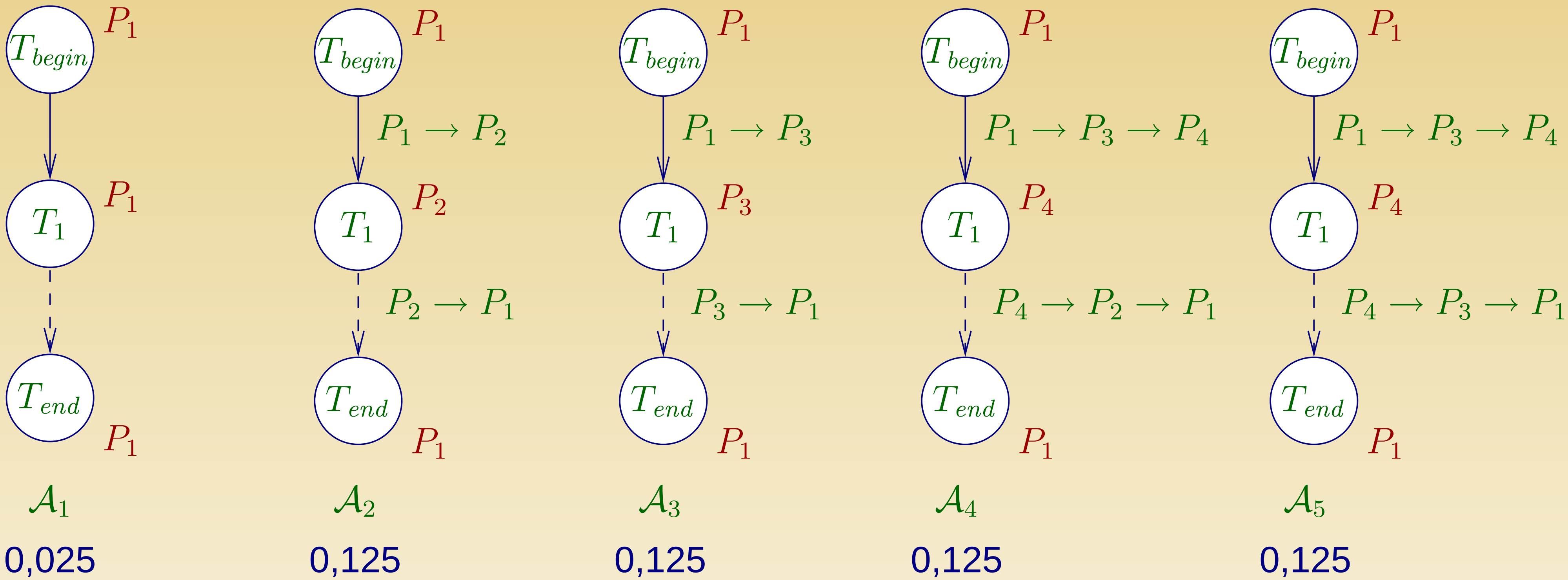
Décomposition en allocations



$\mathcal{A}_1$	$\mathcal{A}_2$	$\mathcal{A}_3$	$\mathcal{A}_4$	$\mathcal{A}_5$
0,025	0,125	0,125	0,125	0,125

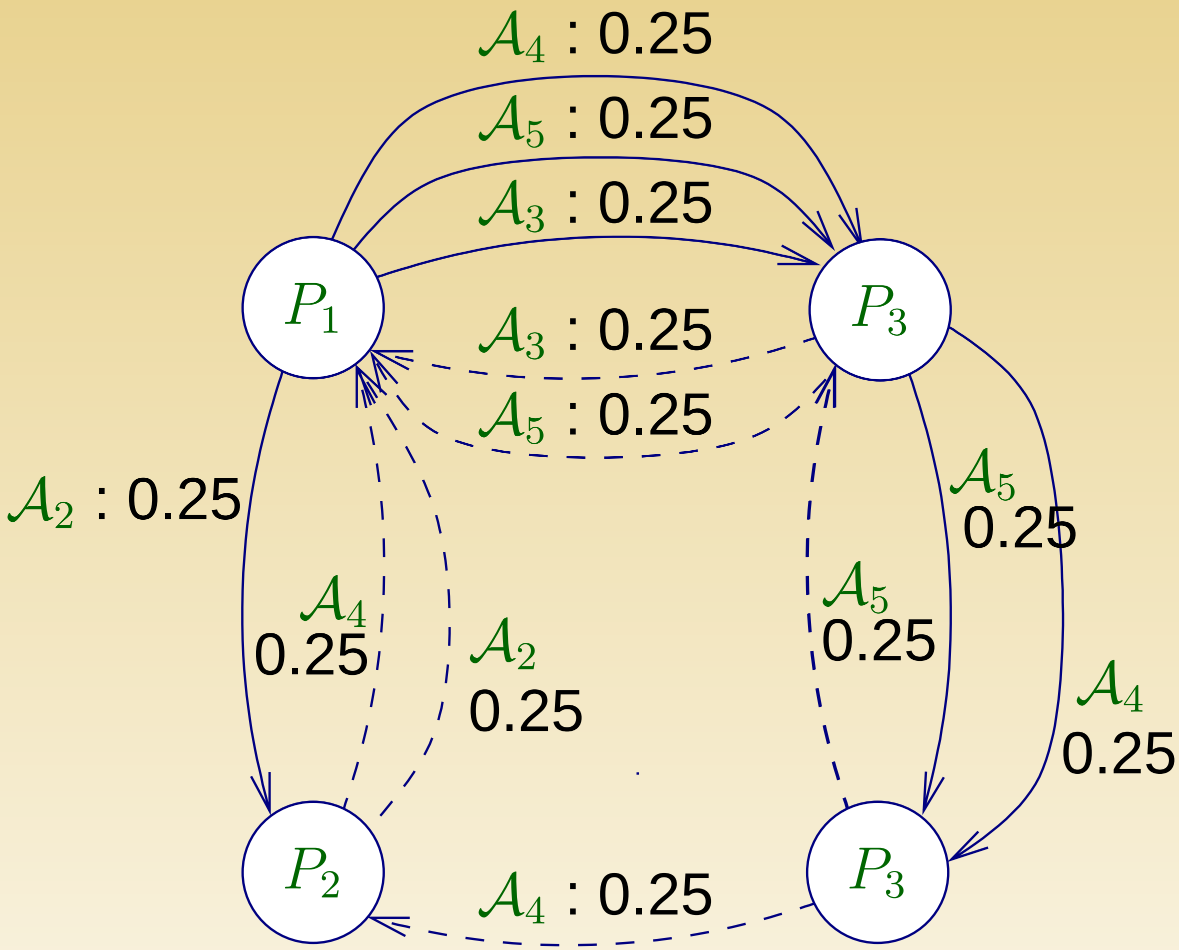


Décomposition en allocations



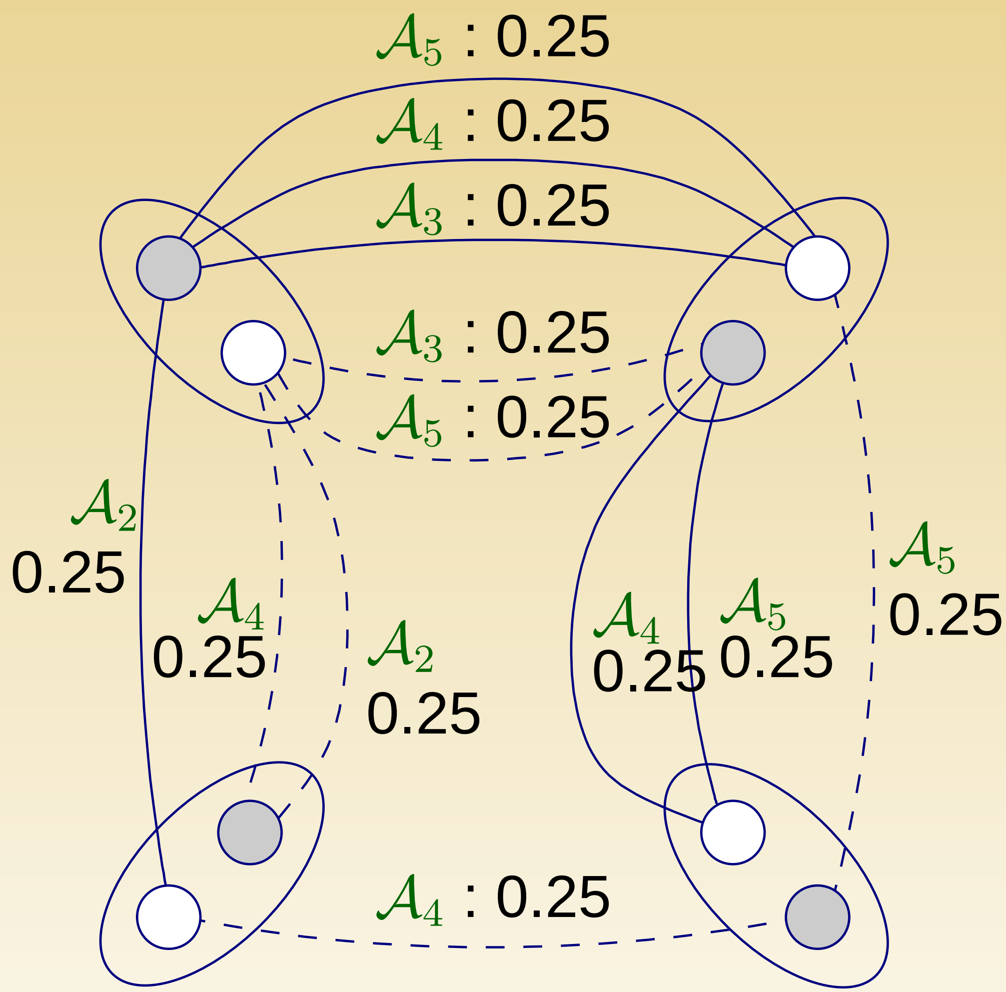
Comment organiser ces allocations ?

Graphe de communications

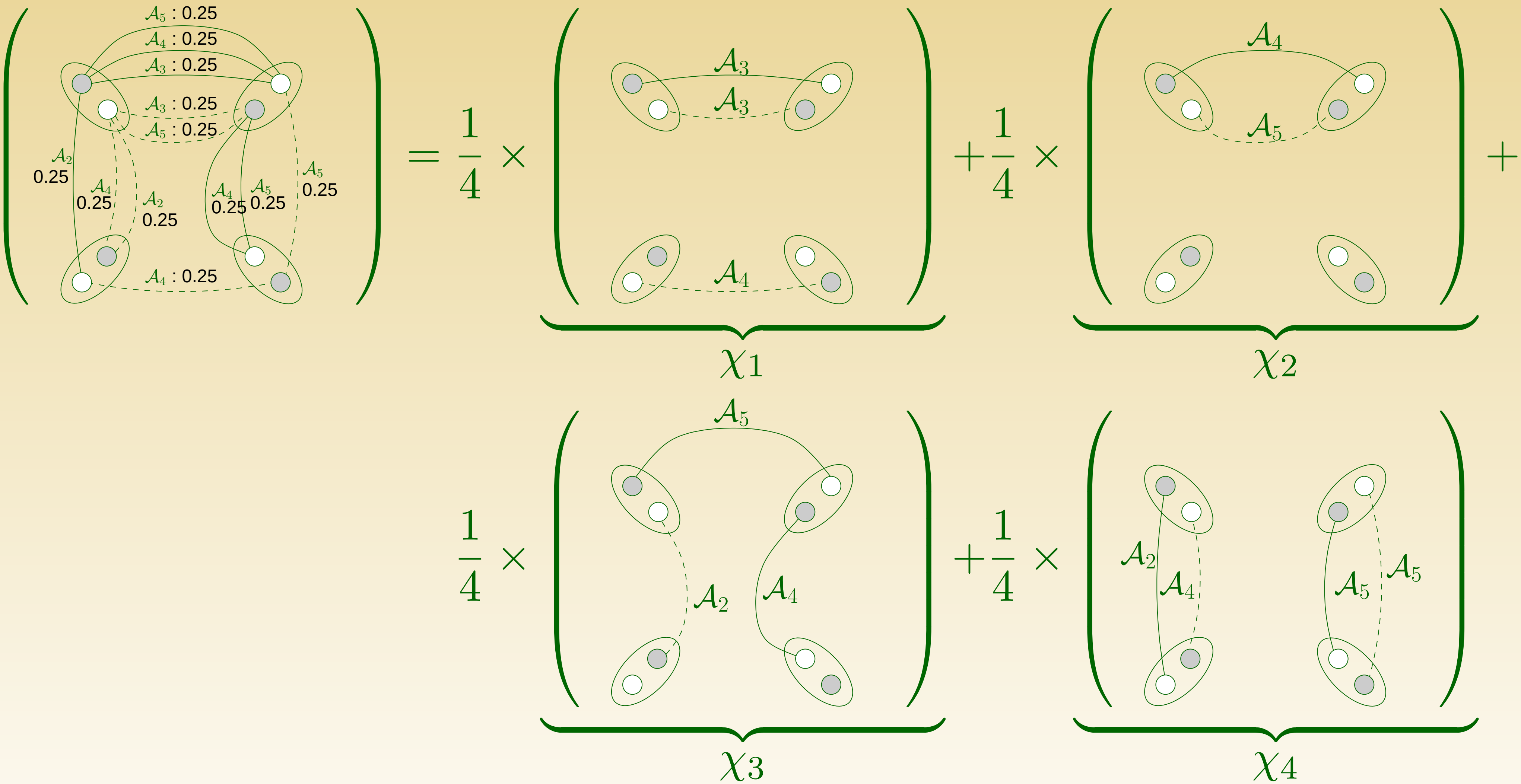


Fraction de temps passée au transfert d'un $e_{k,l}$ de P_i vers P_j pour une allocation donnée.

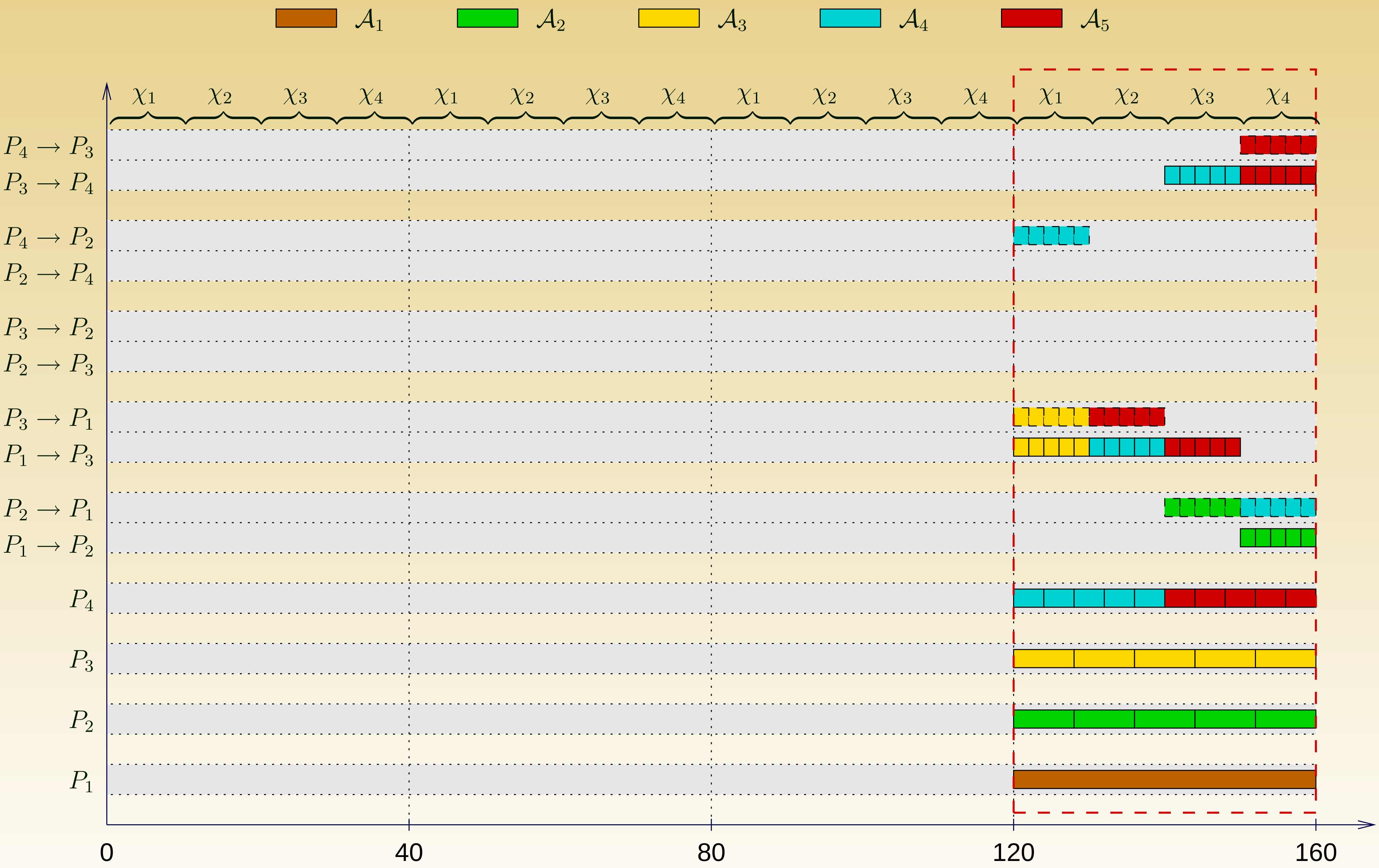
Contraintes 1-port = couplage



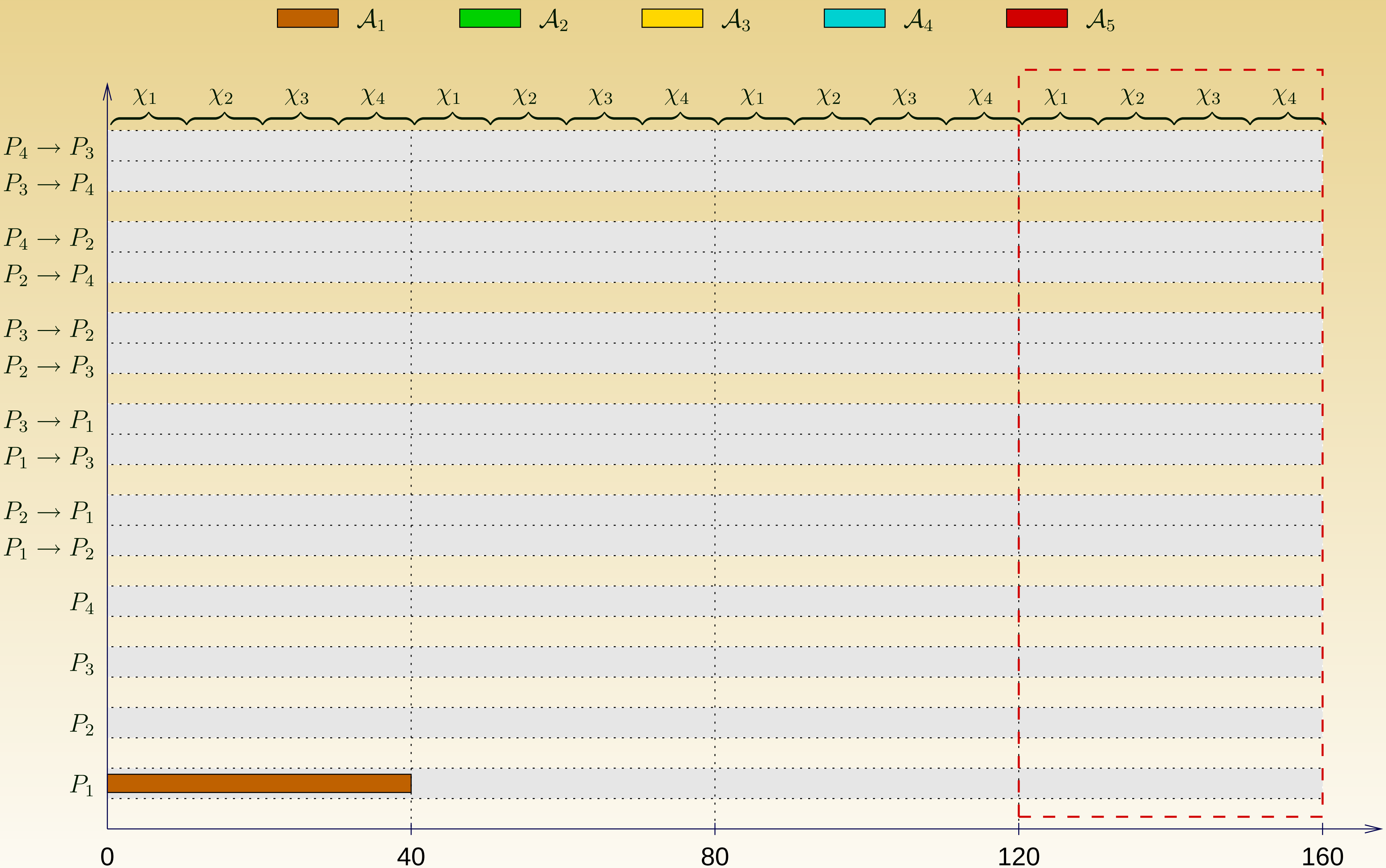
Décomposition en couplages



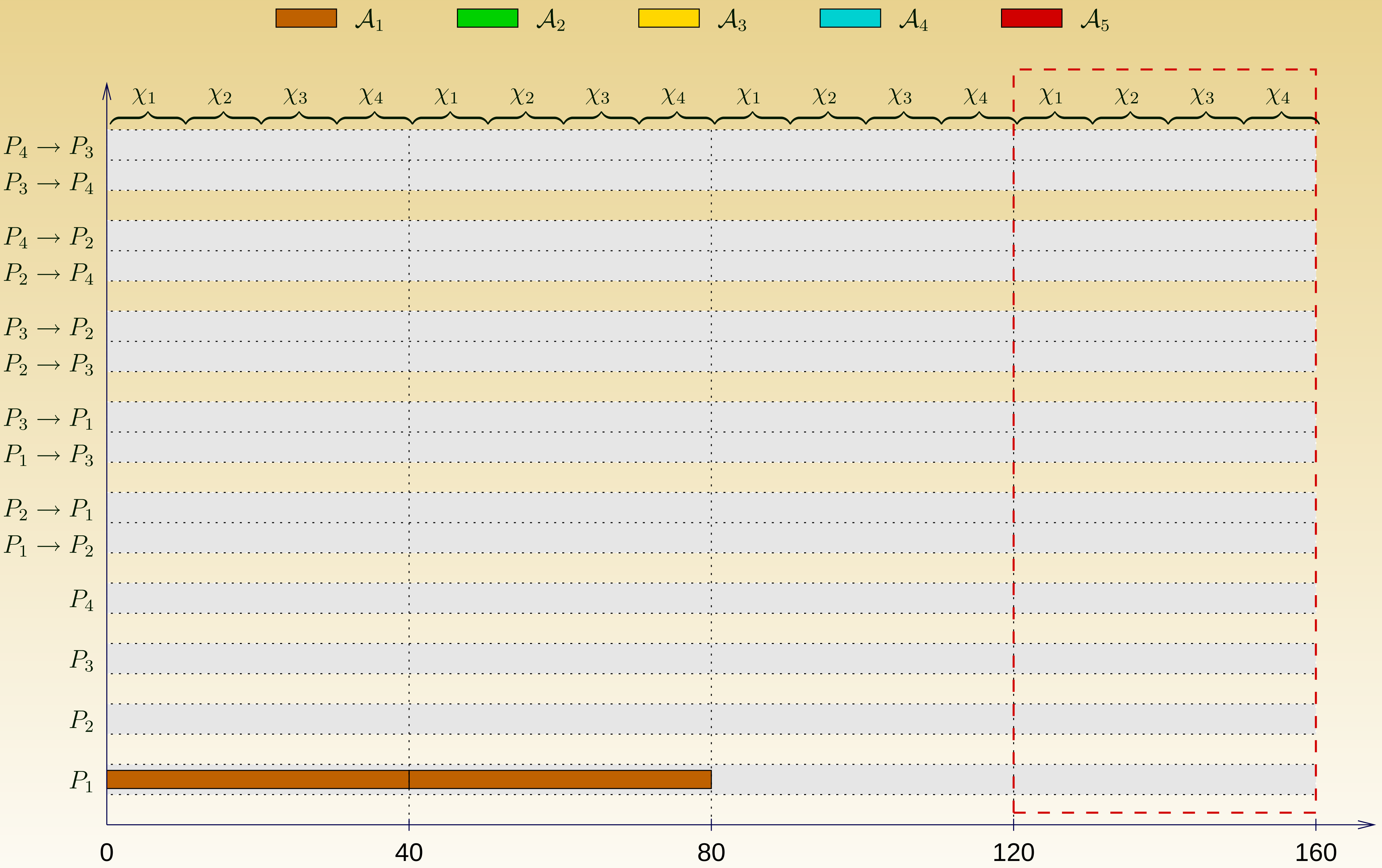
Ordonnancement cyclique de débit optimal



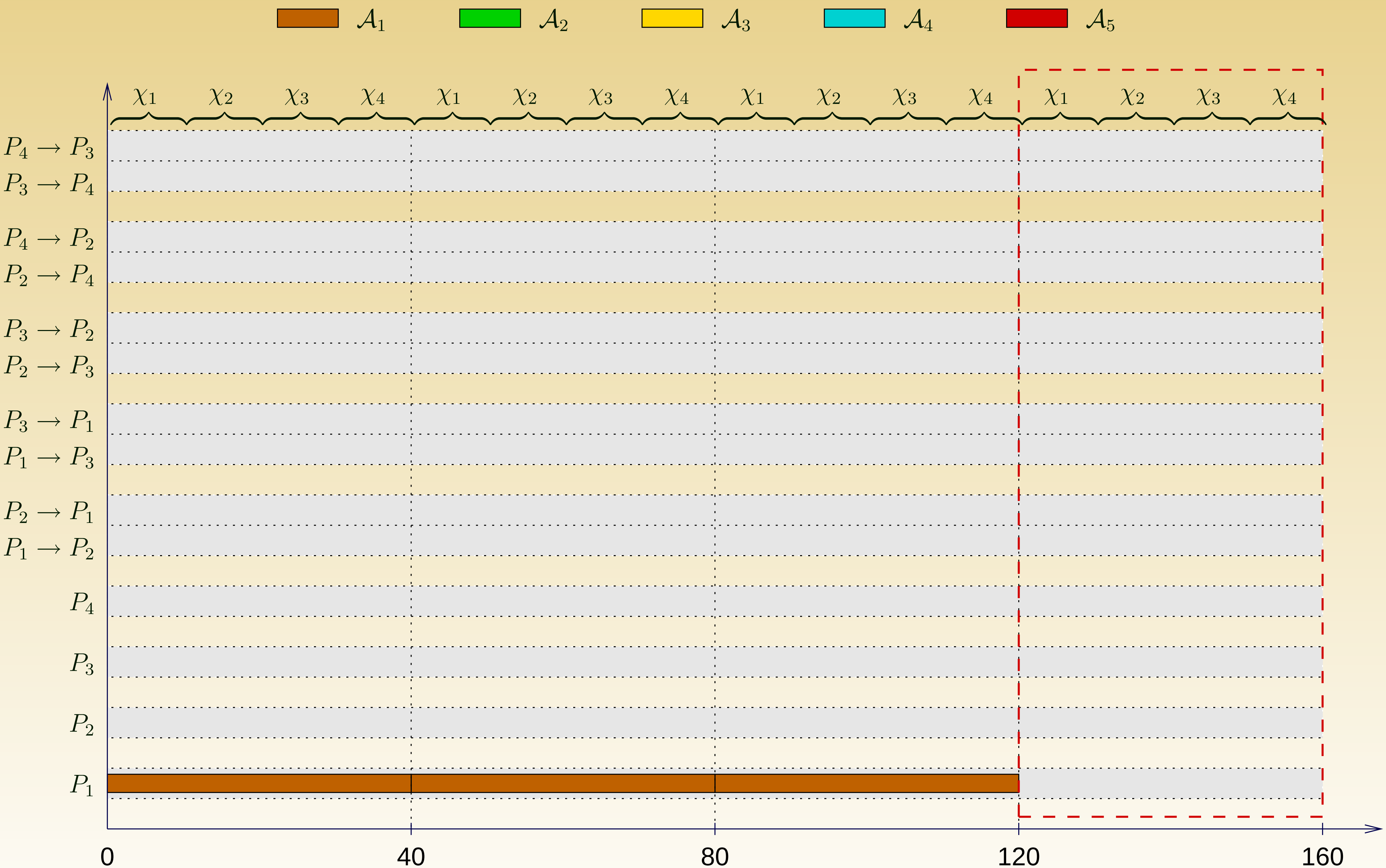
Ordonnancement cyclique de débit optimal



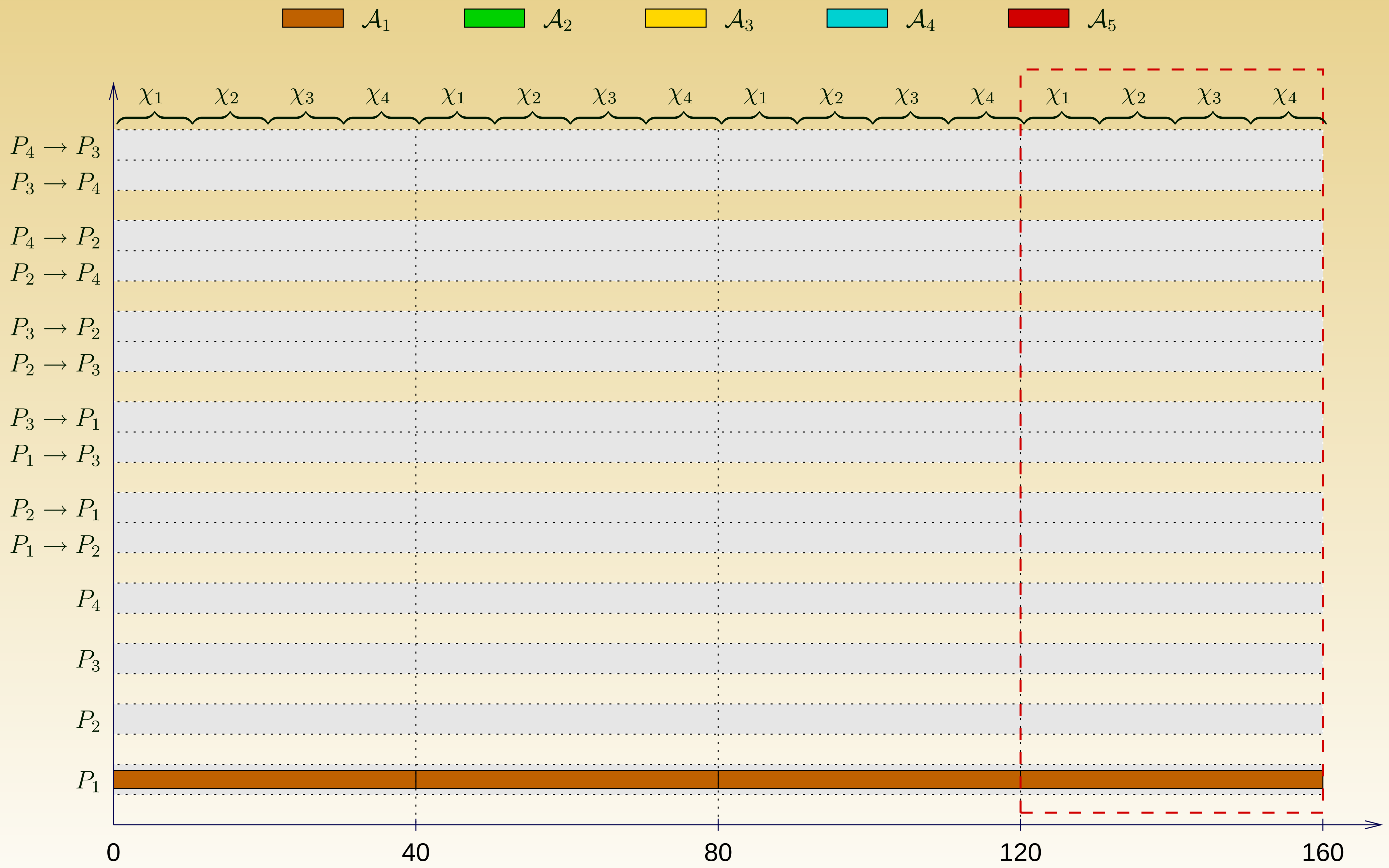
Ordonnancement cyclique de débit optimal



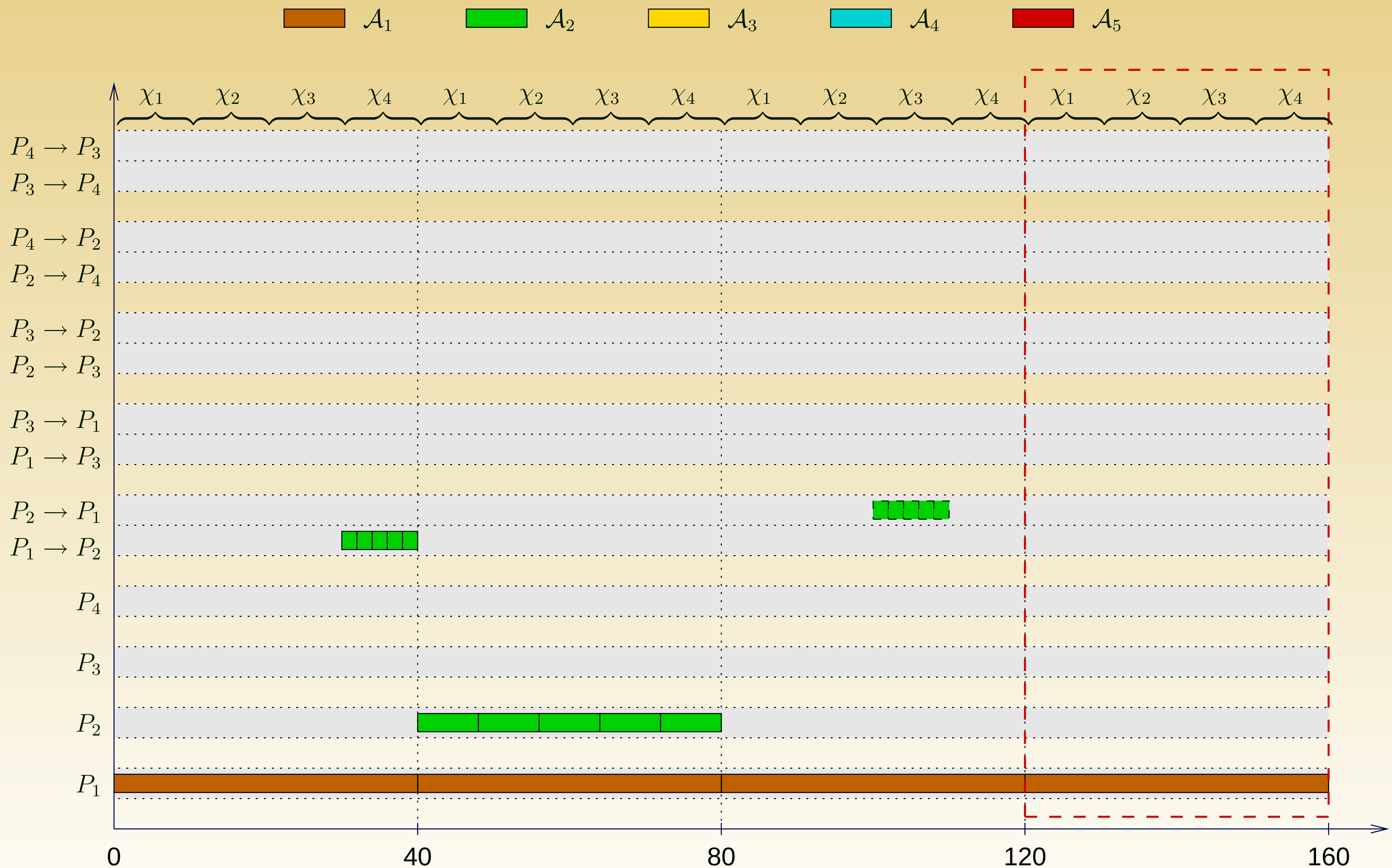
Ordonnancement cyclique de débit optimal



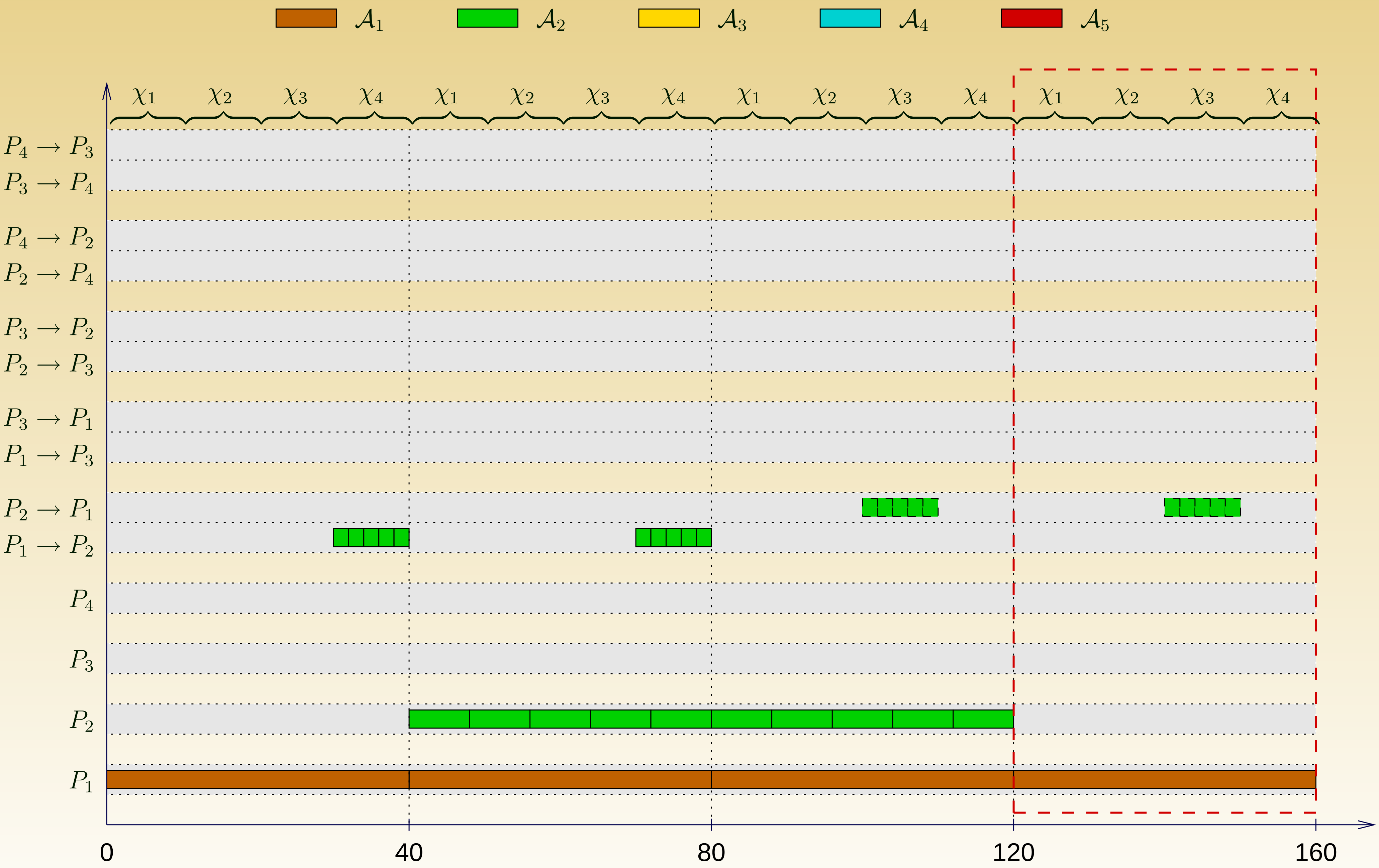
Ordonnancement cyclique de débit optimal



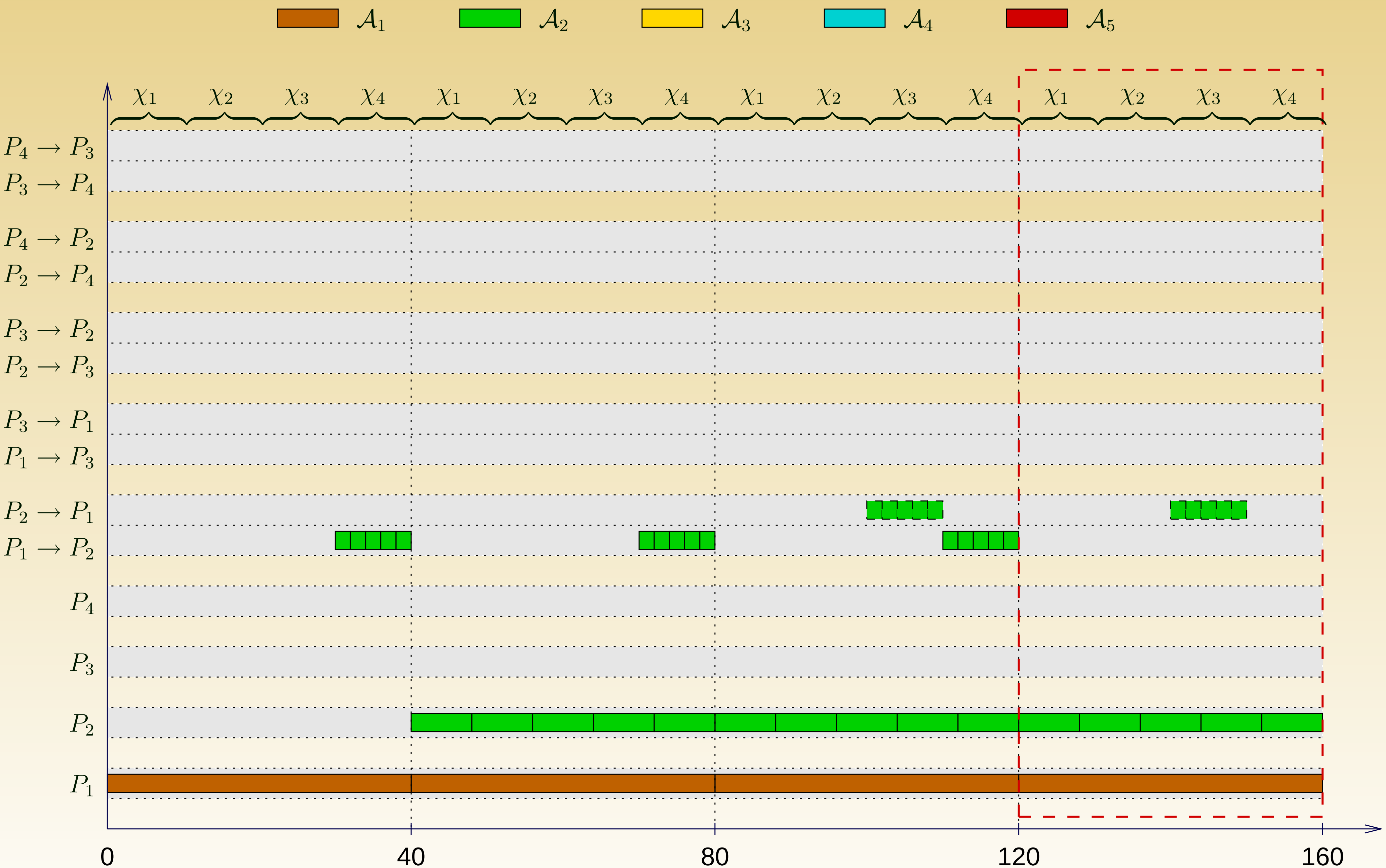
Ordonnancement cyclique de débit optimal



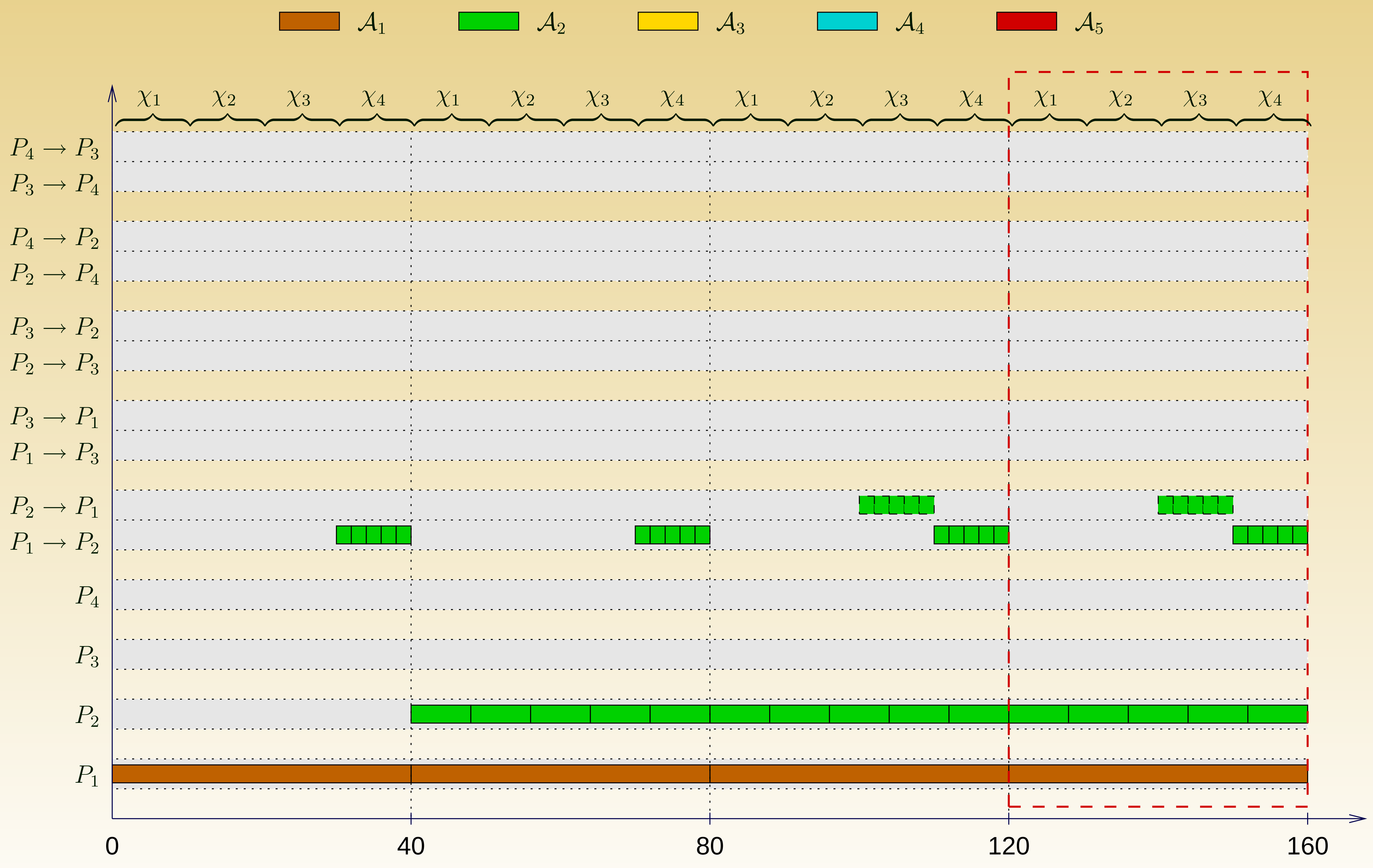
Ordonnancement cyclique de débit optimal



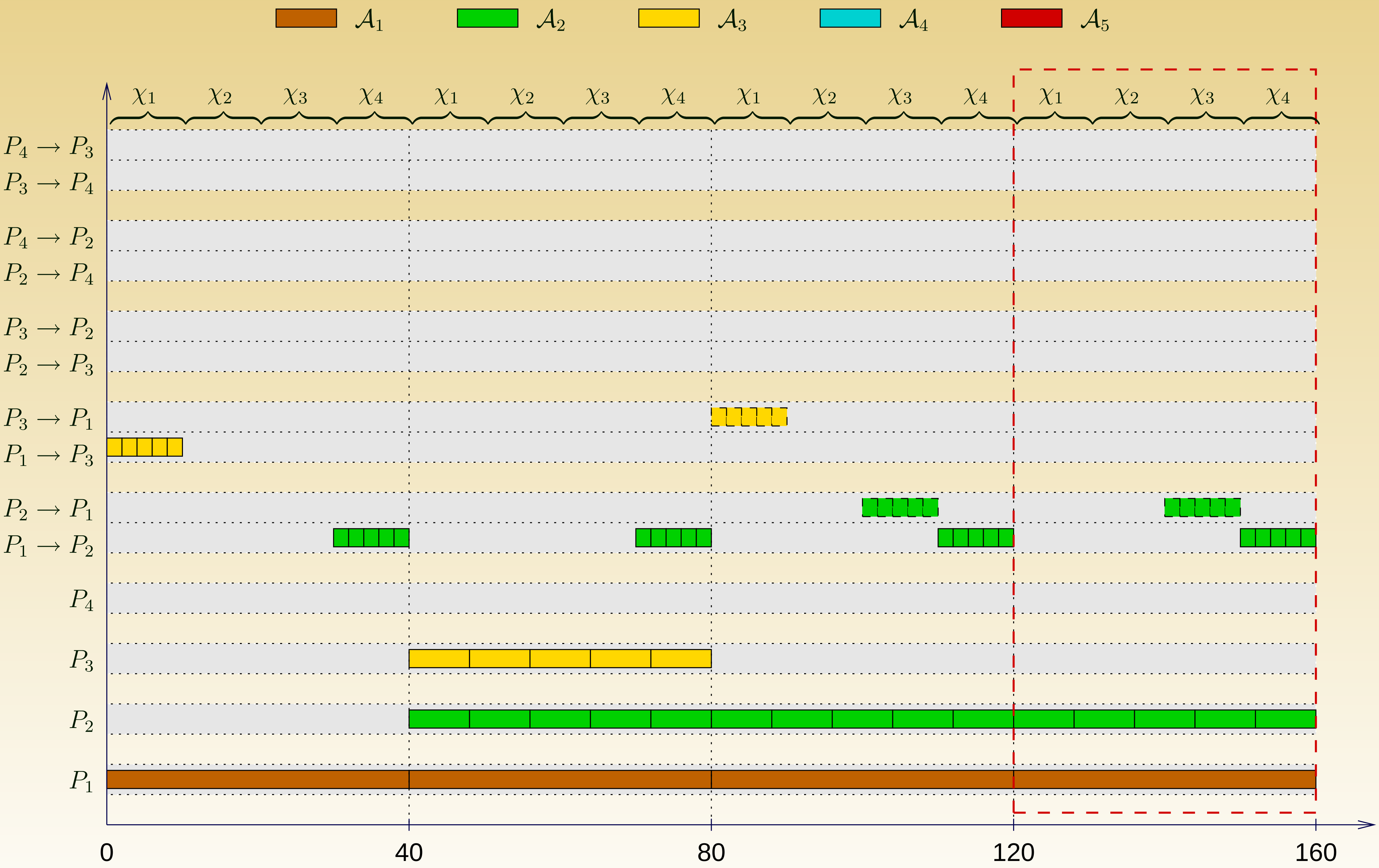
Ordonnancement cyclique de débit optimal



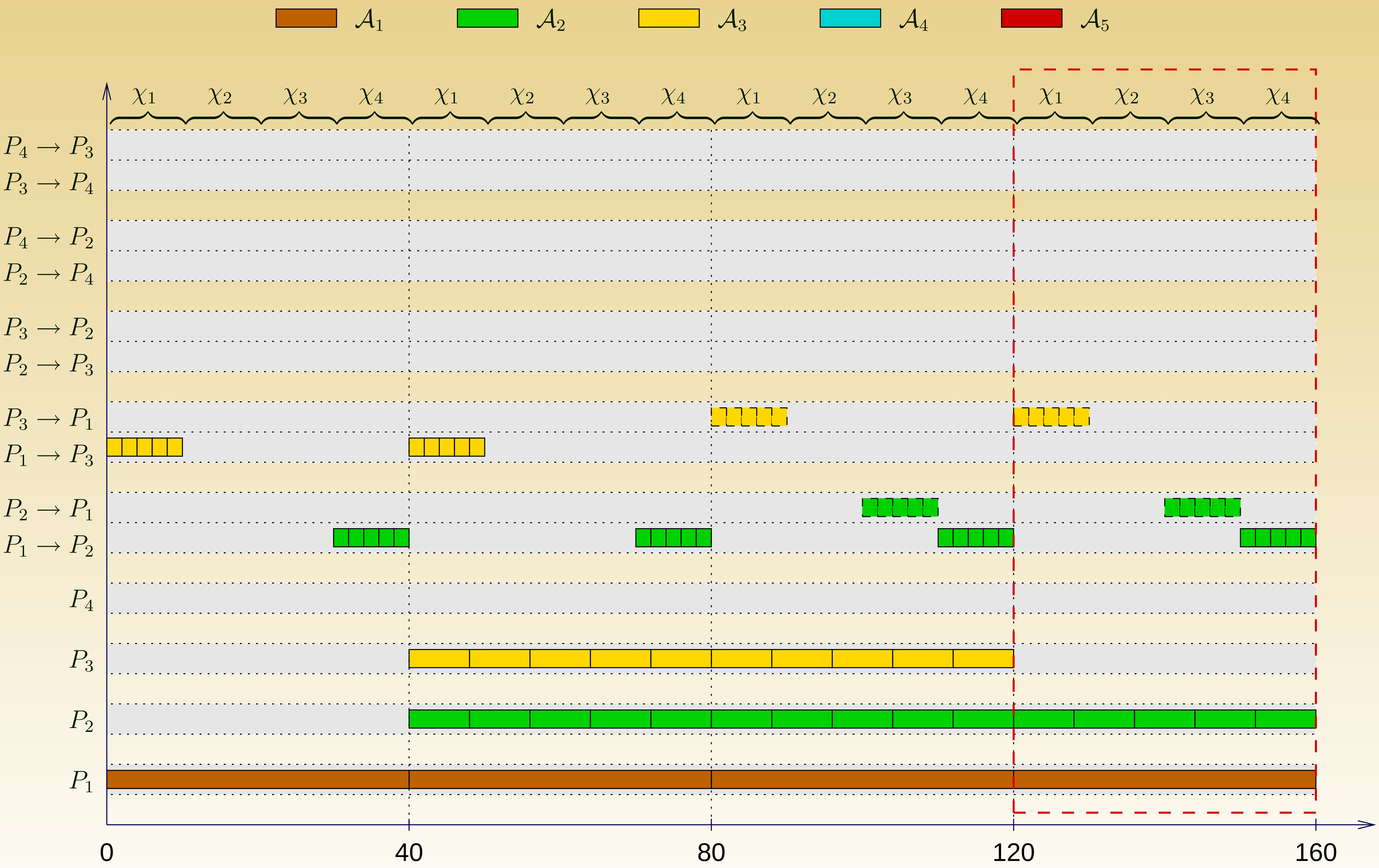
Ordonnancement cyclique de débit optimal



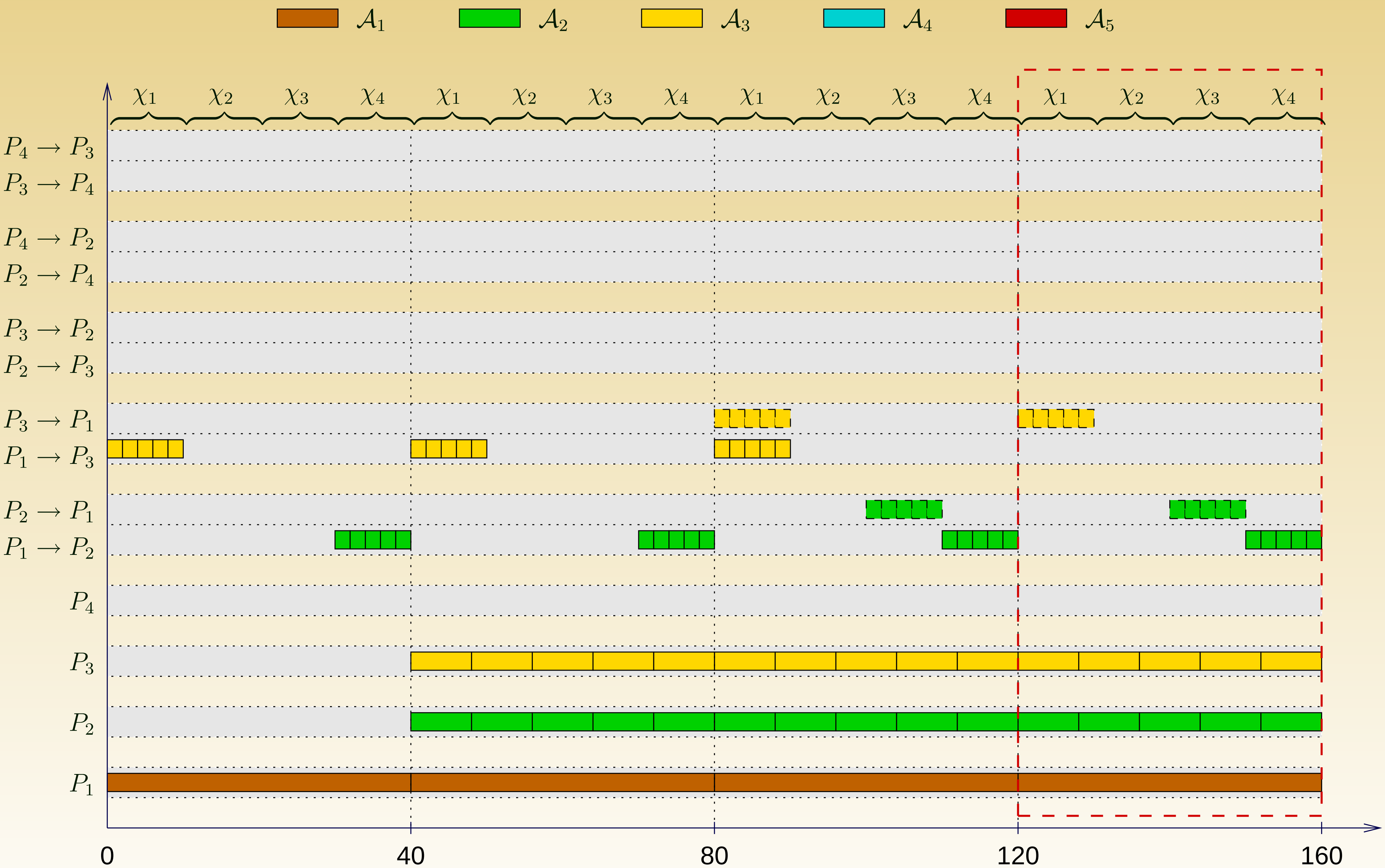
Ordonnancement cyclique de débit optimal



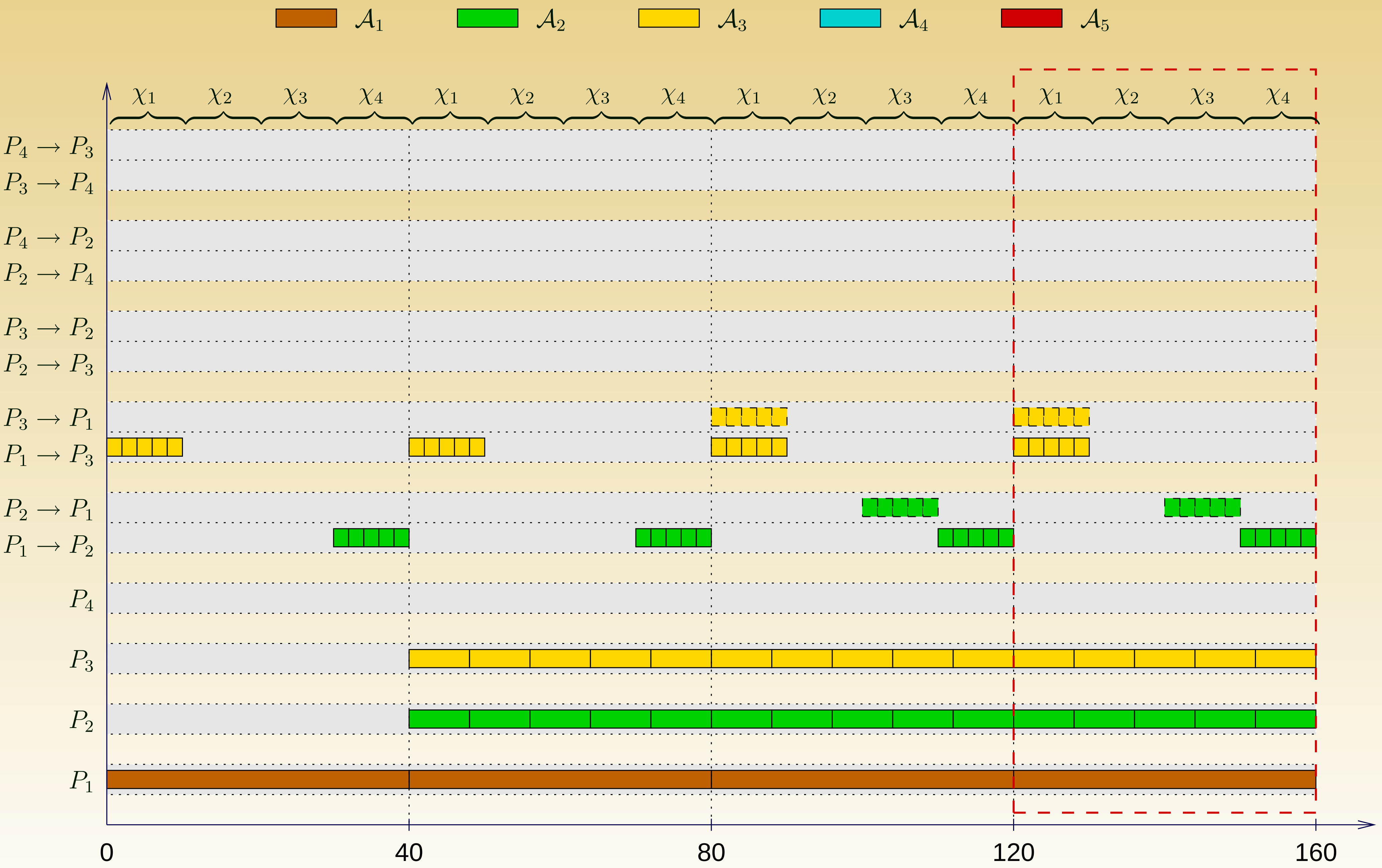
Ordonnancement cyclique de débit optimal



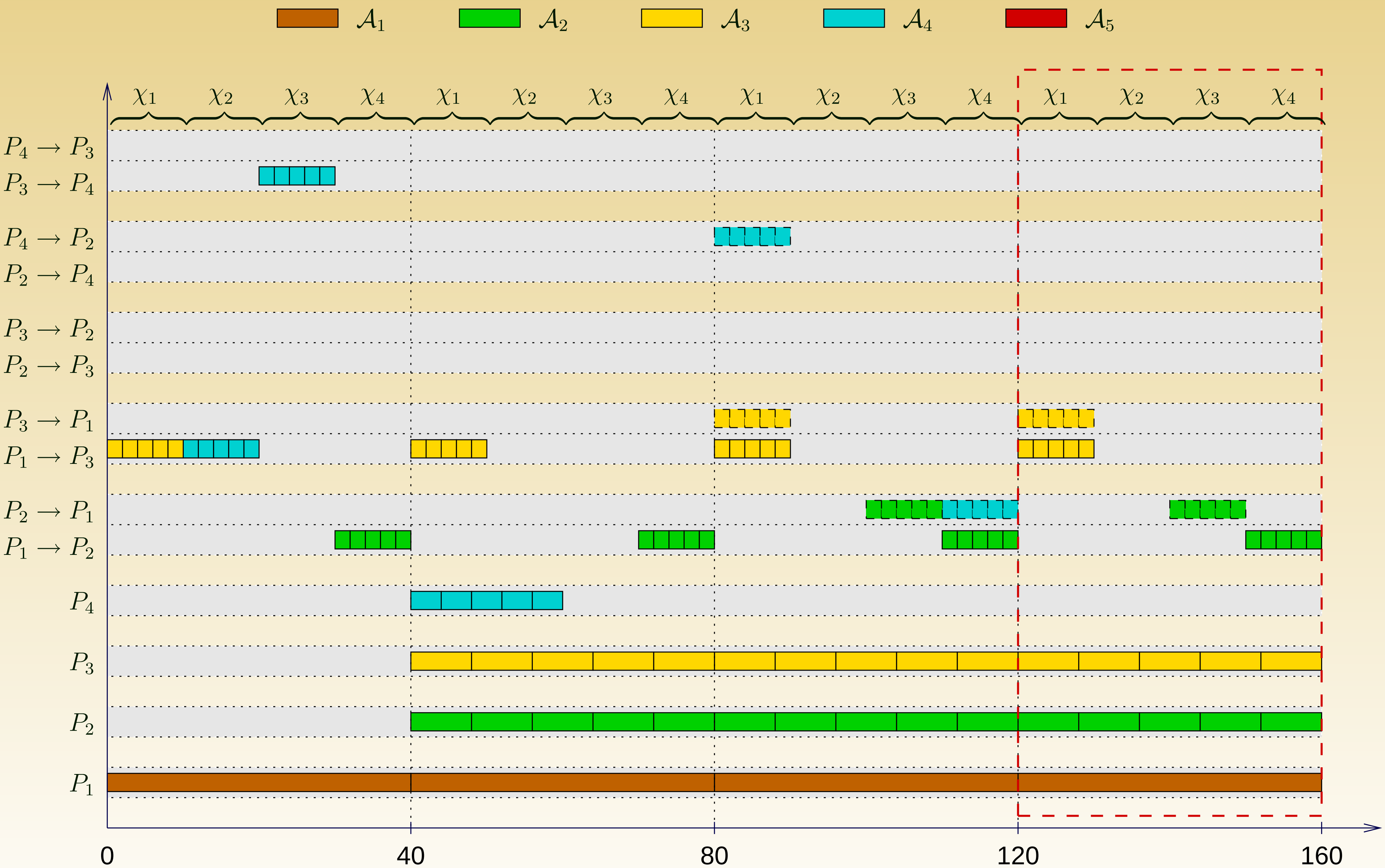
Ordonnancement cyclique de débit optimal



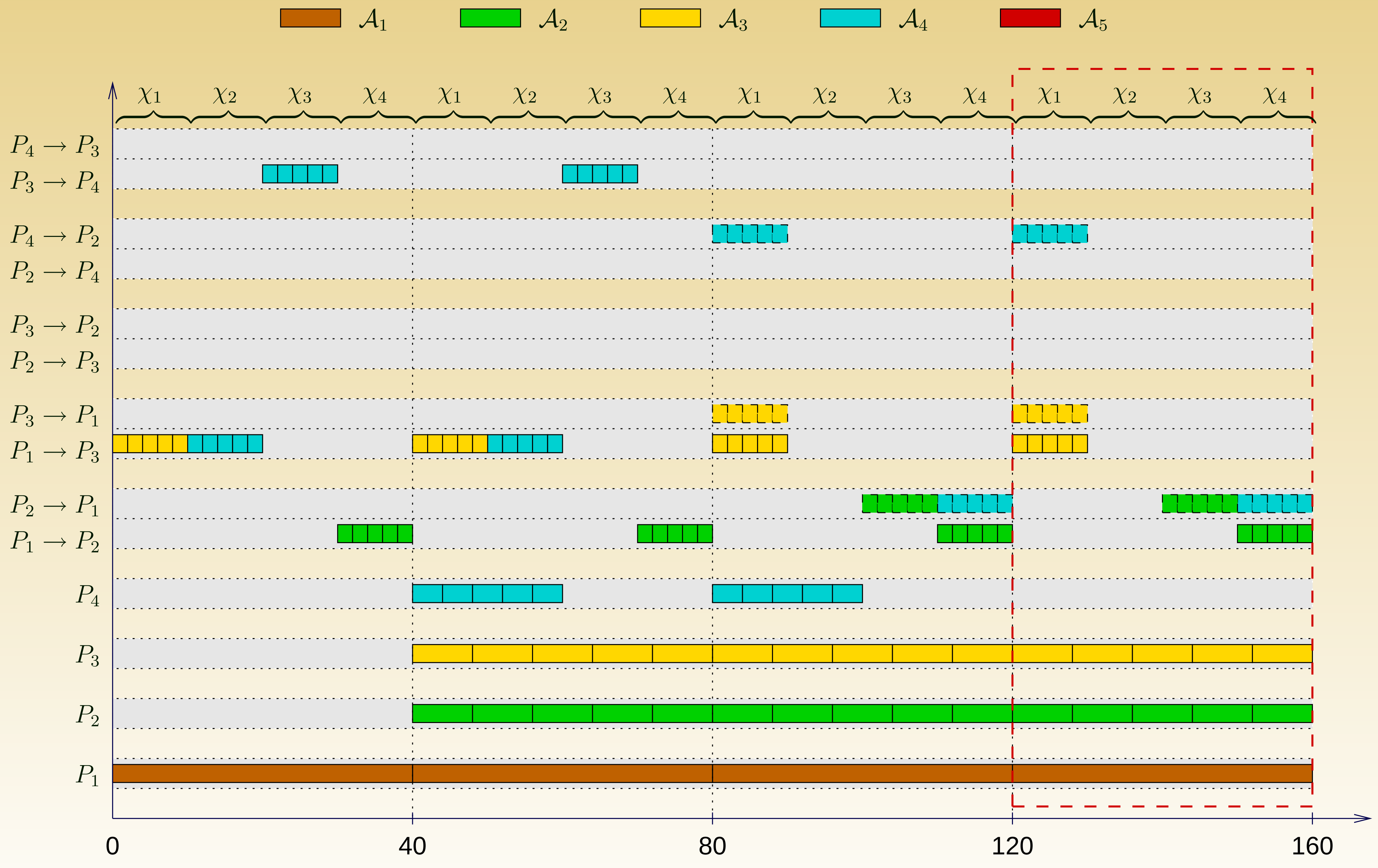
Ordonnancement cyclique de débit optimal



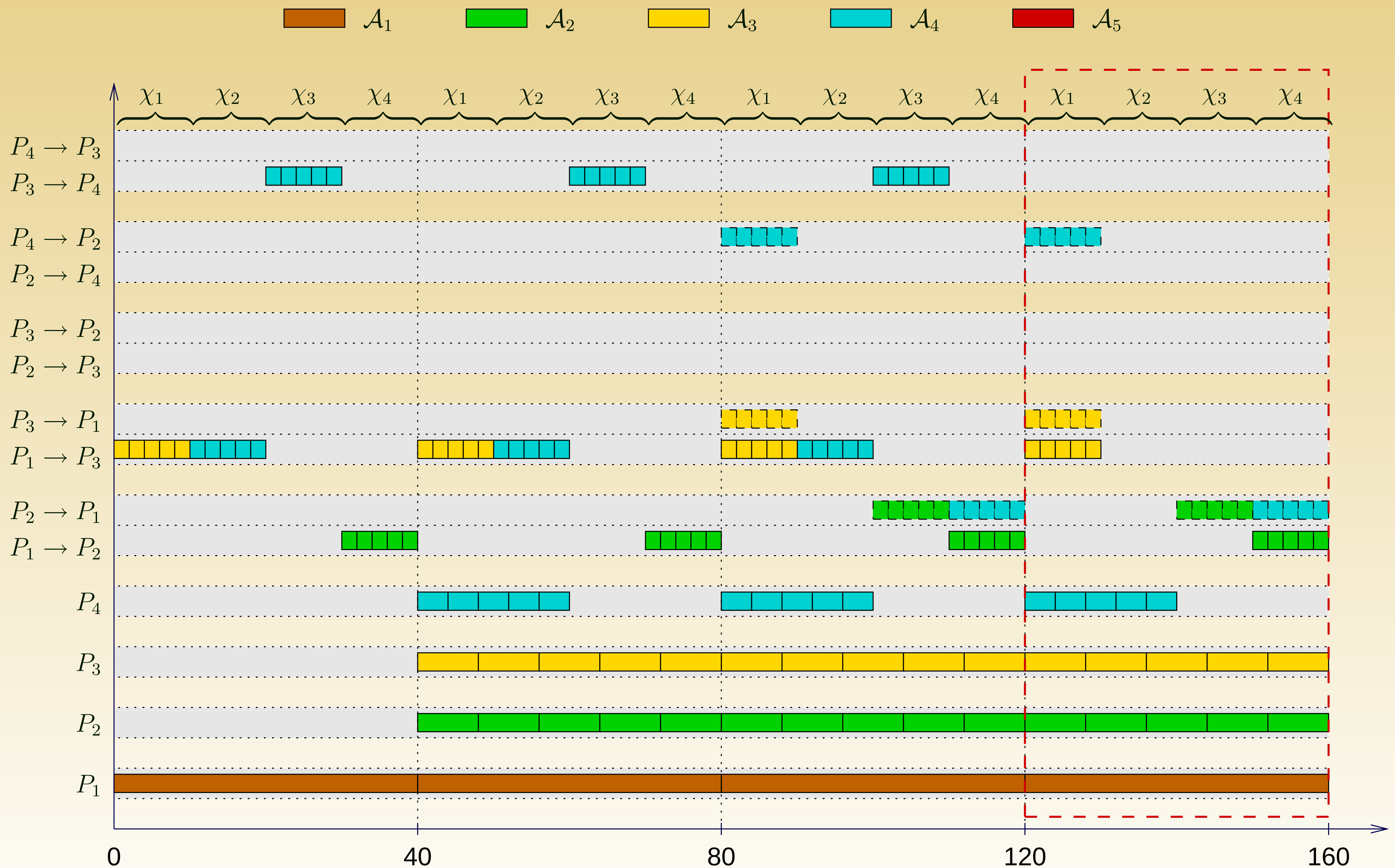
Ordonnancement cyclique de débit optimal



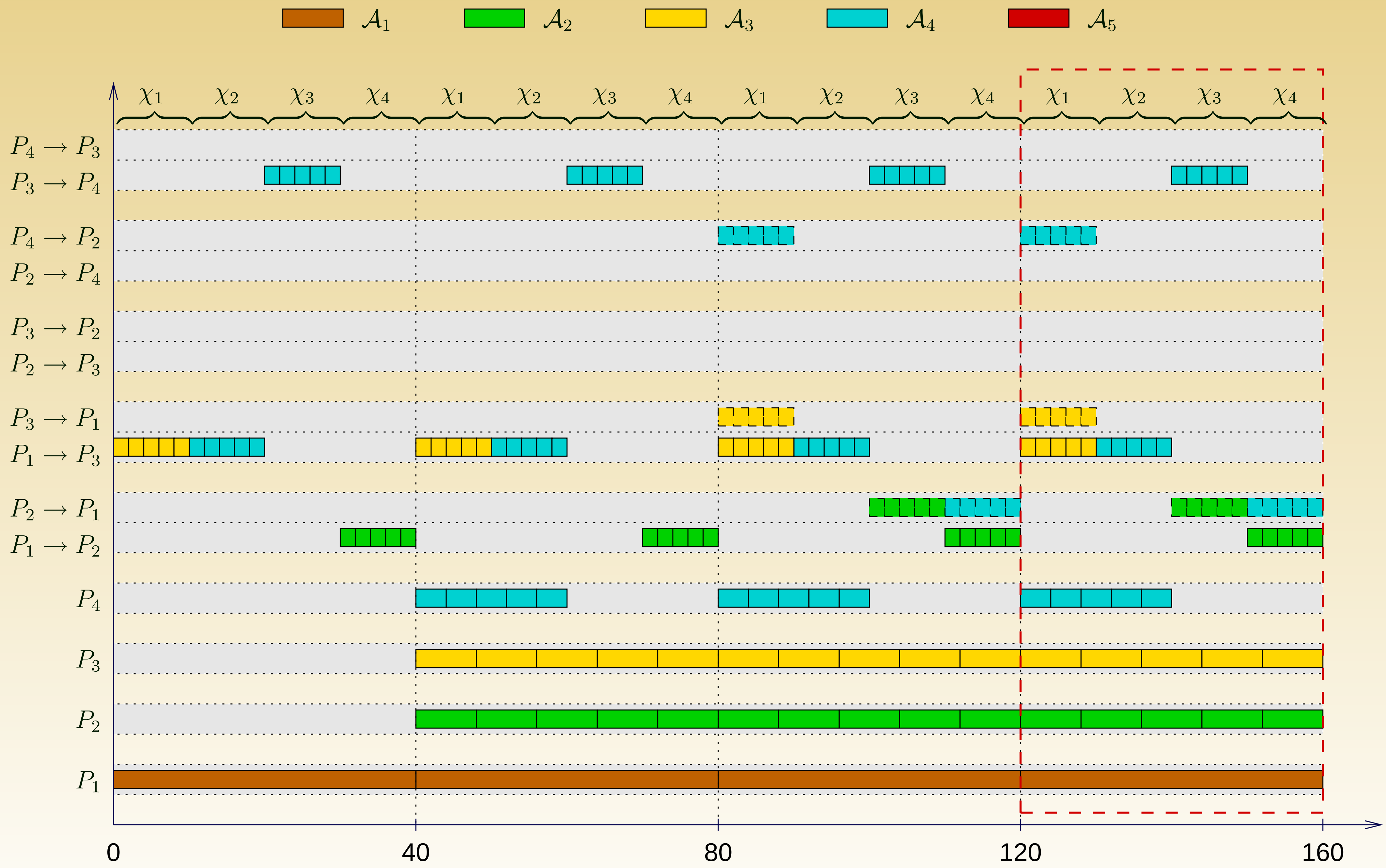
Ordonnancement cyclique de débit optimal



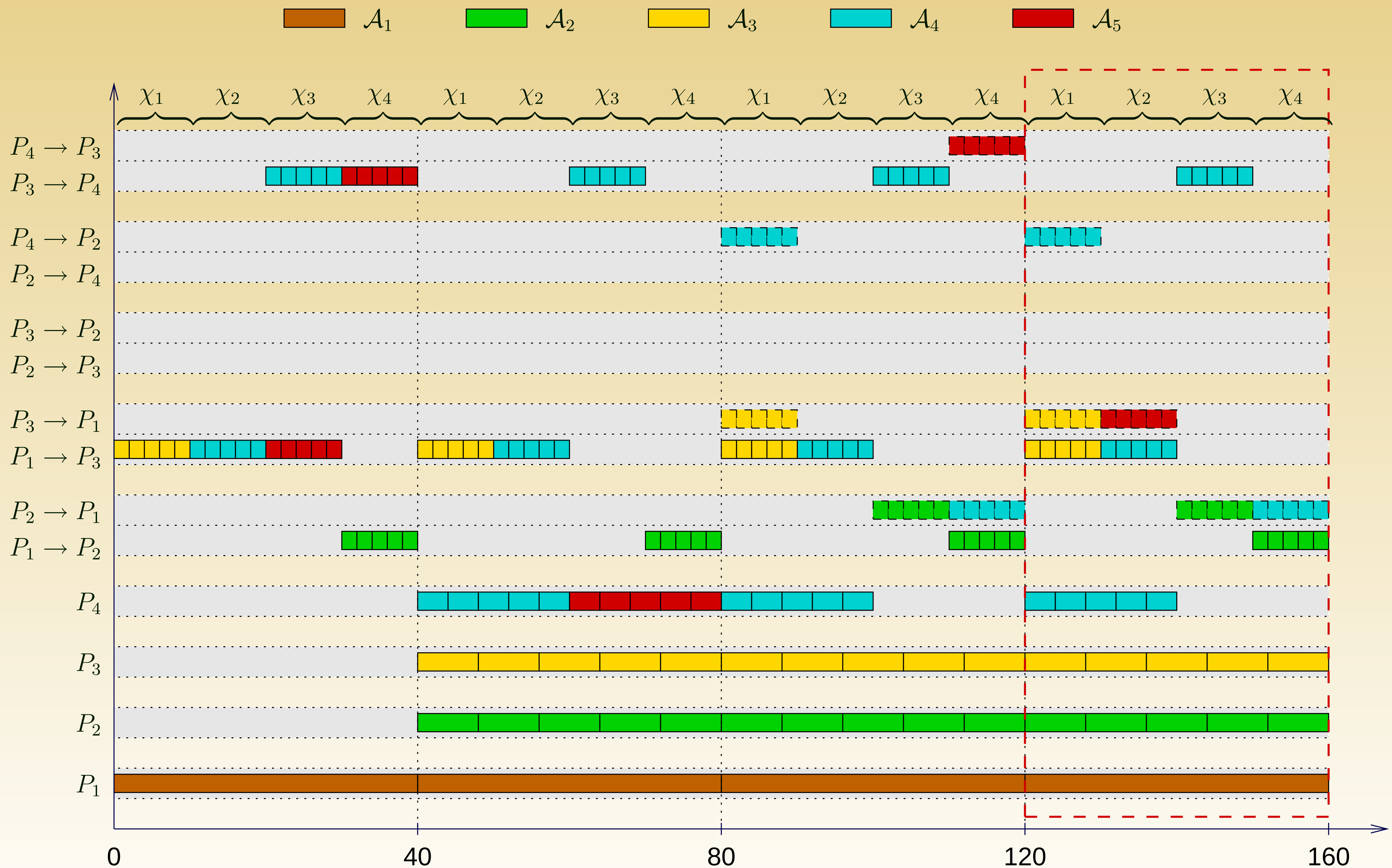
Ordonnancement cyclique de débit optimal



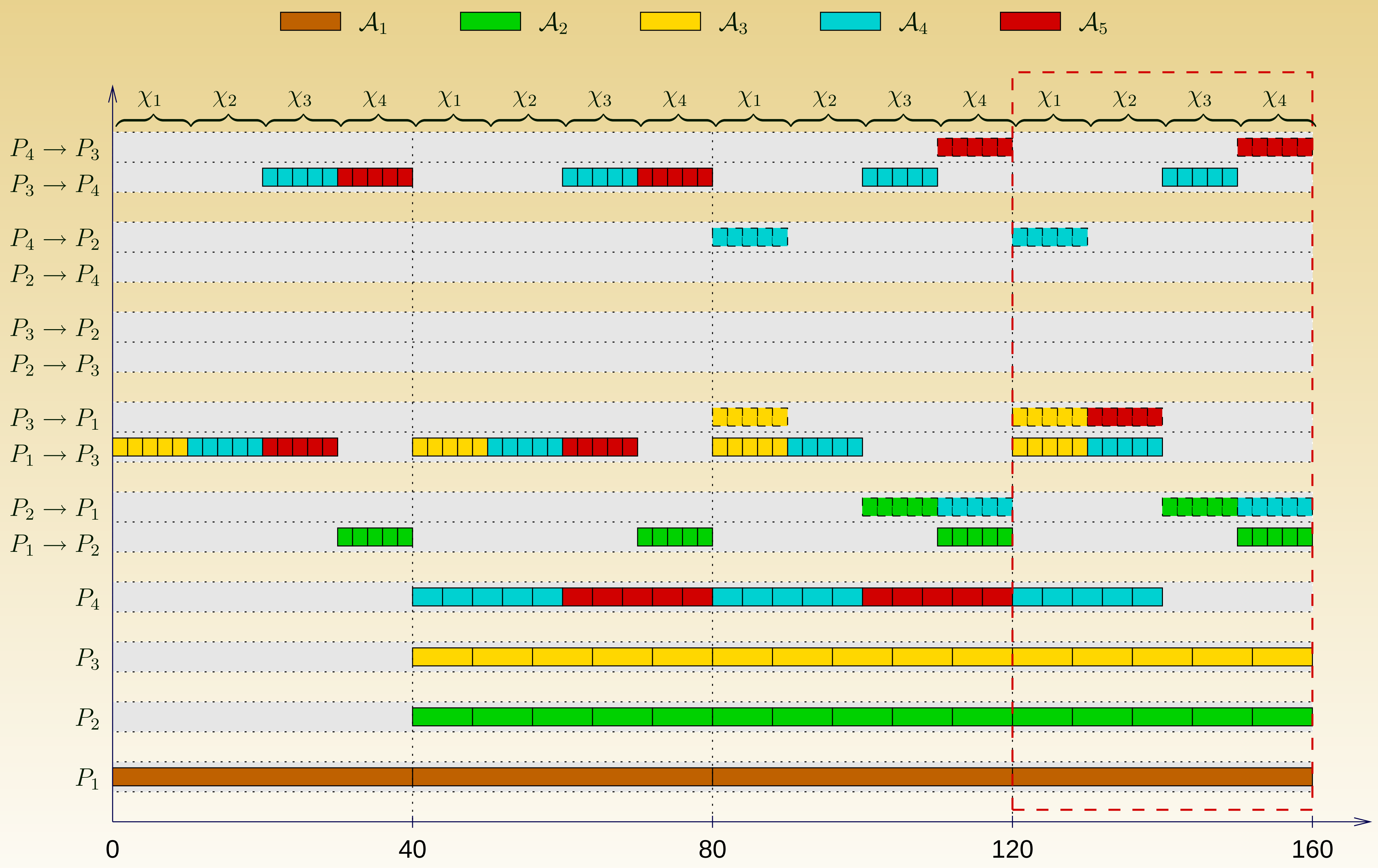
Ordonnancement cyclique de débit optimal



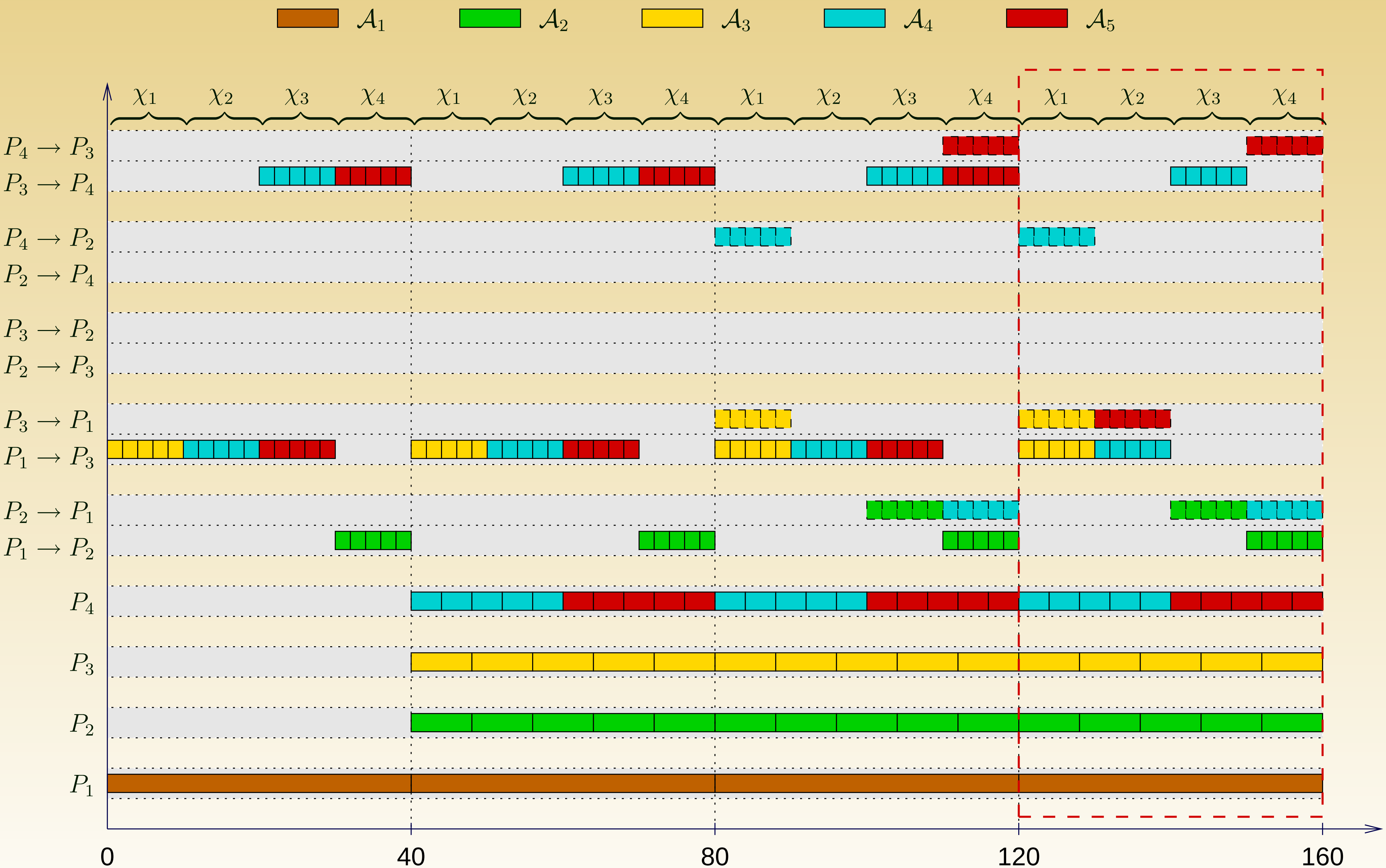
Ordonnancement cyclique de débit optimal



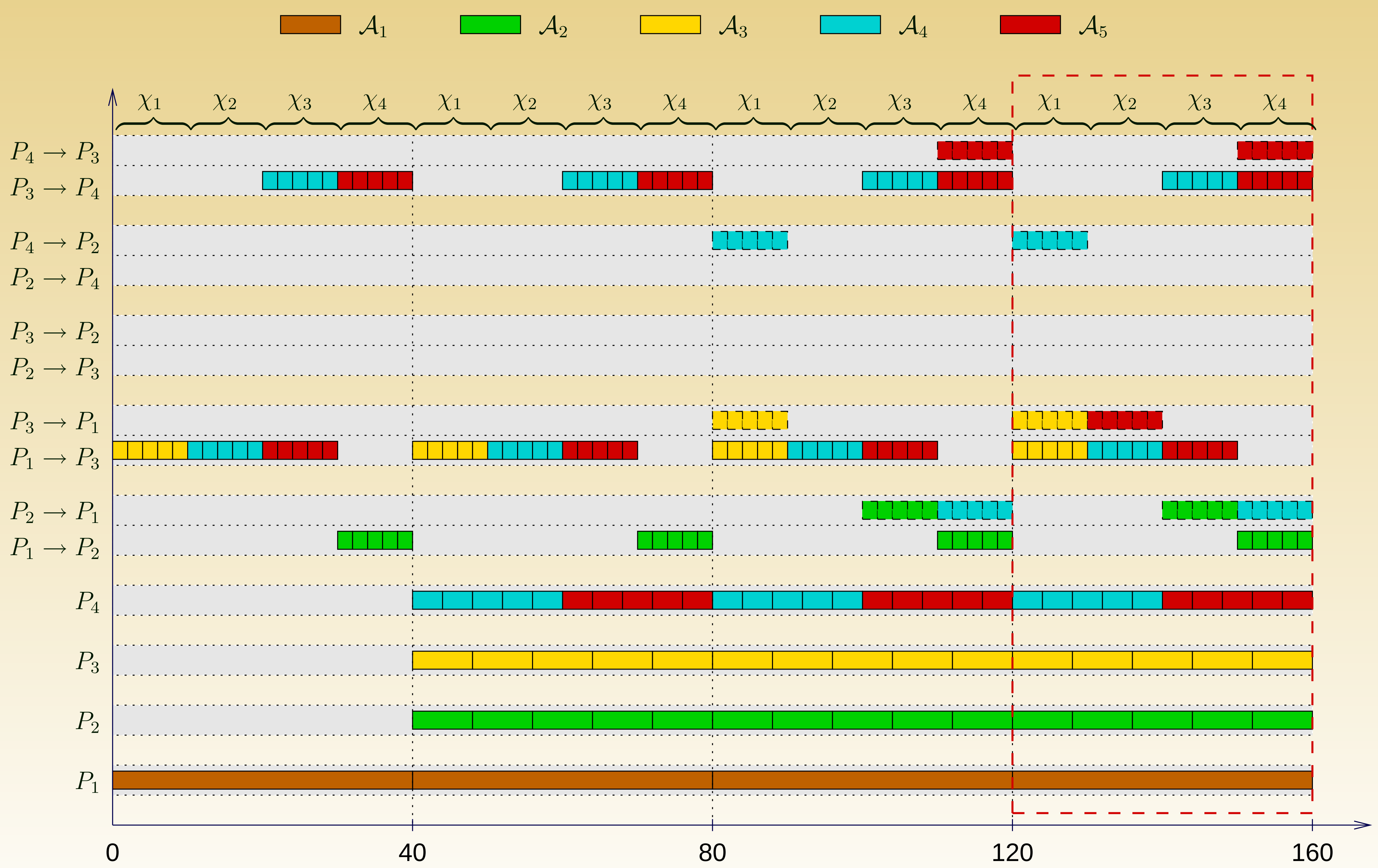
Ordonnancement cyclique de débit optimal



Ordonnancement cyclique de débit optimal



Ordonnancement cyclique de débit optimal



Ordonnancement asymptotiquement optimal

La construction précédente est bien polynomiale.

Cet ordonnancement est asymptotiquement optimal : en temps T, il n'est possible d'exécuter qu'un nombre constant (indépendant de T) de tâches de plus.

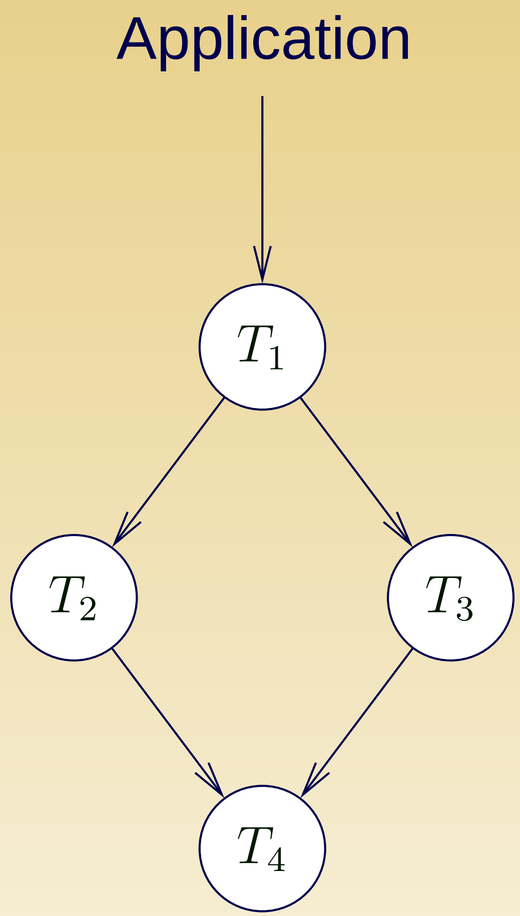
Ordonnancement asymptotiquement optimal

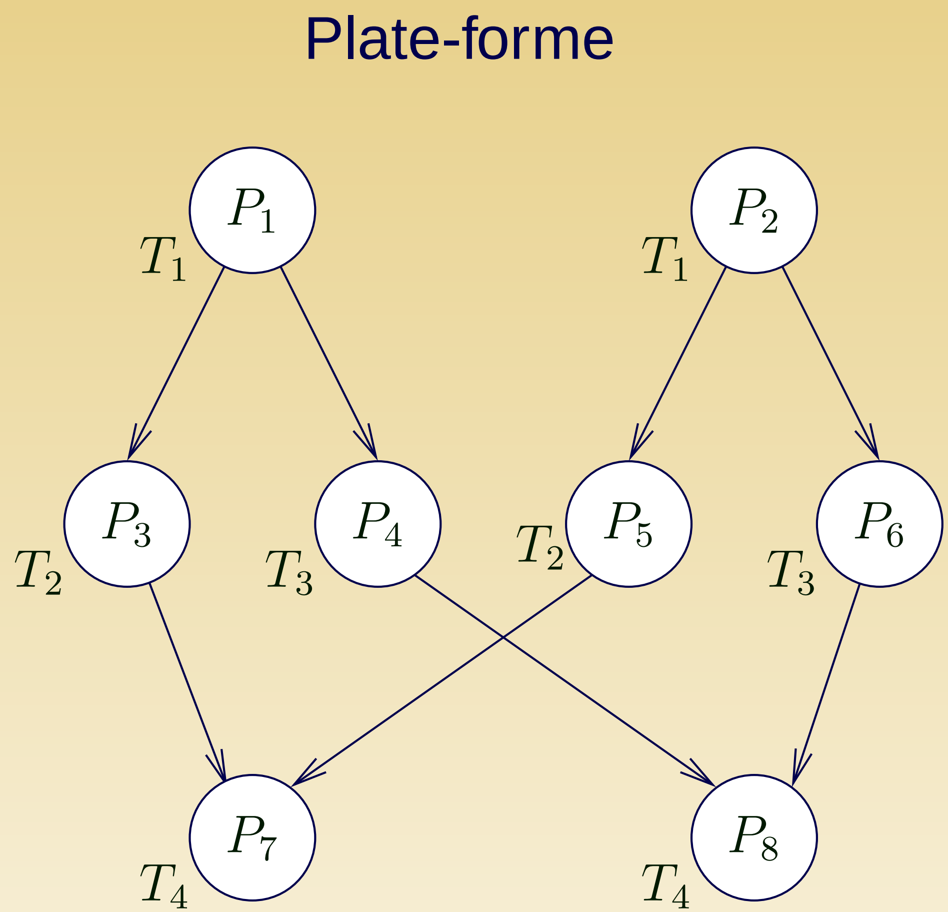
La construction précédente est bien polynomiale.

Cet ordonnancement est asymptotiquement optimal : en temps T, il n'est possible d'exécuter qu'un nombre constant (indépendant de T) de tâches de plus.

Graphe de tâches quelconque

Les équations précédentes ne marchent plus...

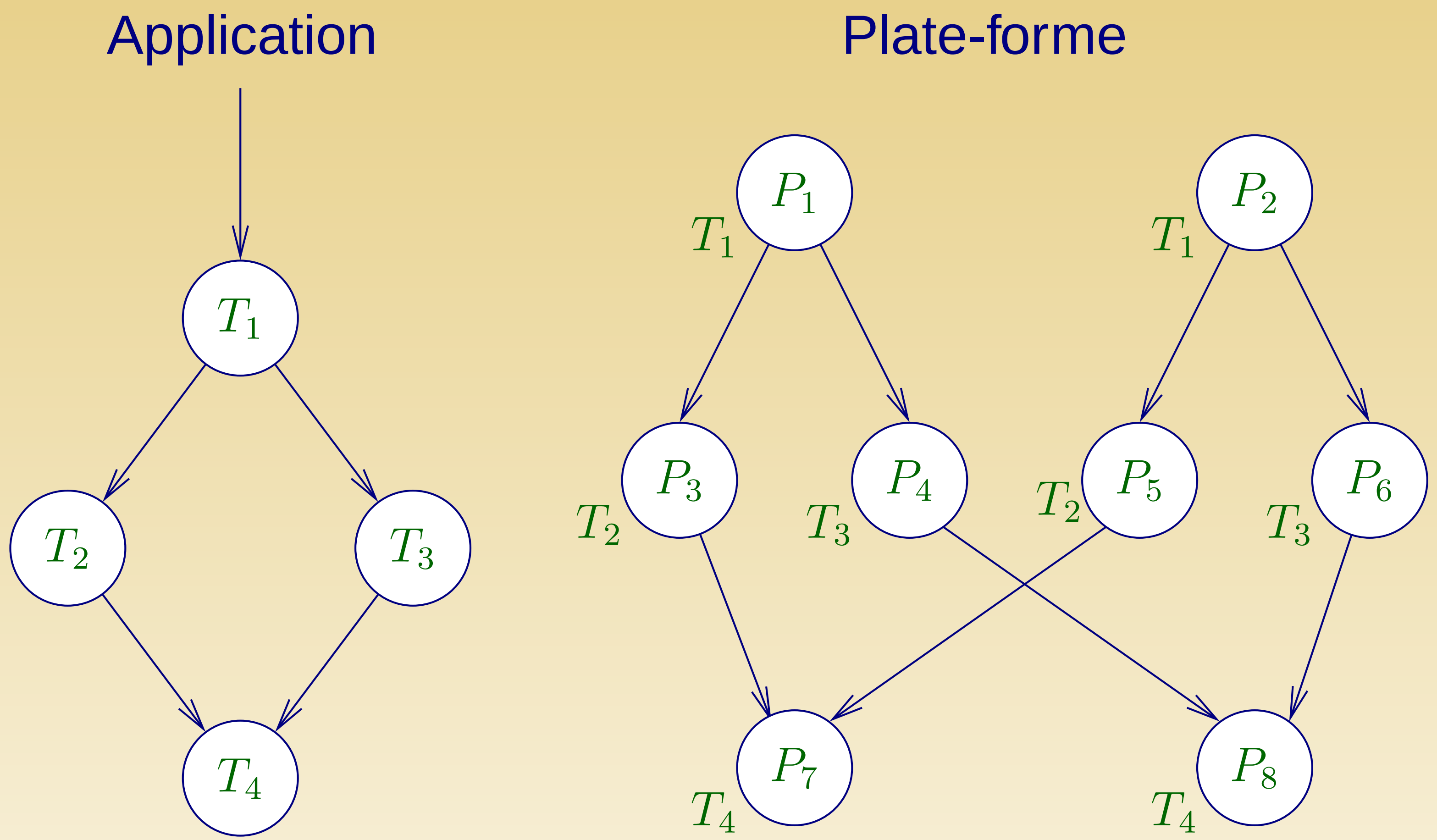




Il n'est plus possible de décomposer la solution du programme linéaire en une somme pondérée d'allocations.

$\rightsquigarrow$ Introduire une partie de l'ordonnancement de certains ancêtres dans le programme linéaire.

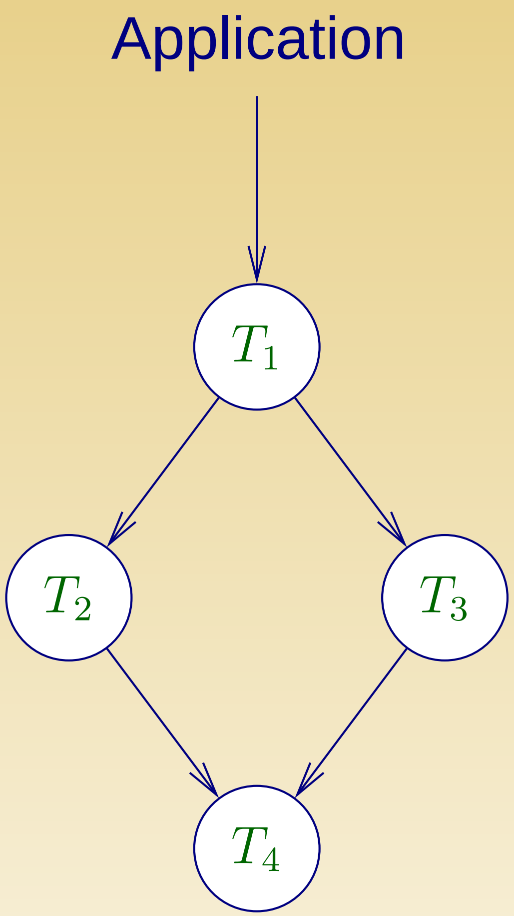
Les équations précédentes ne marchent plus. . .

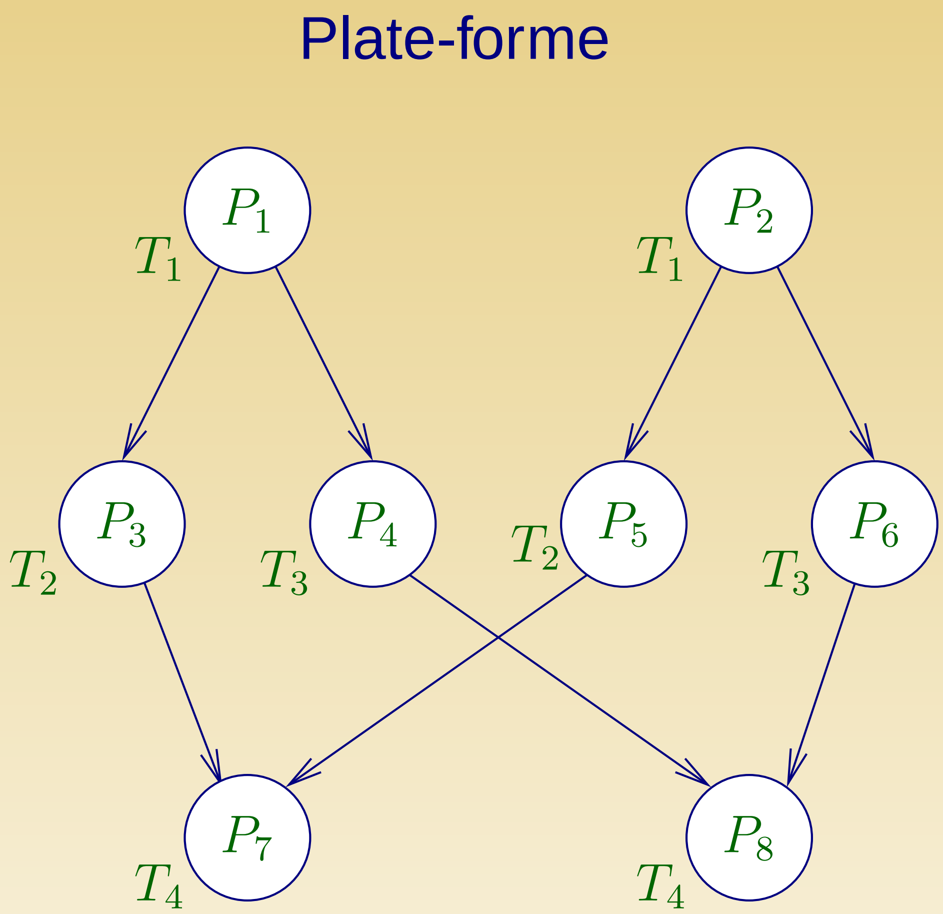


Il n'est plus possible de décomposer la solution du programme linéaire en une somme pondérée d'allocations.

$\leadsto$ Introduire une partie de l'ordonnancement de certains ancêtres dans le programme linéaire.

Les équations précédentes ne marchent plus…





Il n'est plus possible de décomposer la solution du programme linéaire en une somme pondérée d'allocations.

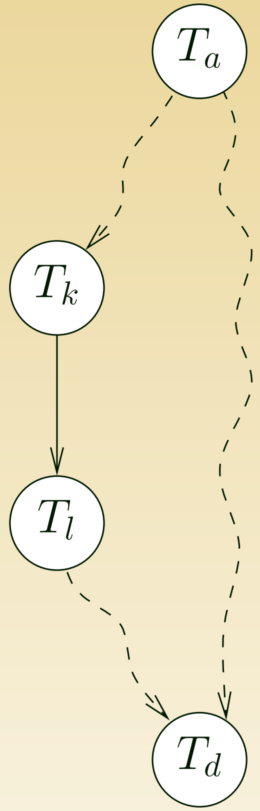
$\rightsquigarrow$ Introduire une partie de l'ordonnancement de certains ancêtres dans le programme linéaire.

Tâche contraignante

Définition. Pour toute dépendance $e_{k,l}$, une tâche T_a est dite contraignante pour $e_{k,l}$ si T_a est un ancêtre de T_l et s'il existe un T_d tel que :

– T_d est un descendant de T_l,

– il y a un chemin de T_a à T_d sommet-disjoint (excepté T_a et T_d) de celui allant de T_a à T_d en passant par T_l.



Une liste de contraintes représente l'allocation de certains ancêtres (par exemple $\{T_{begin} \mapsto P_1, T_1 \mapsto P_2, T_2 \mapsto P_2\}$).

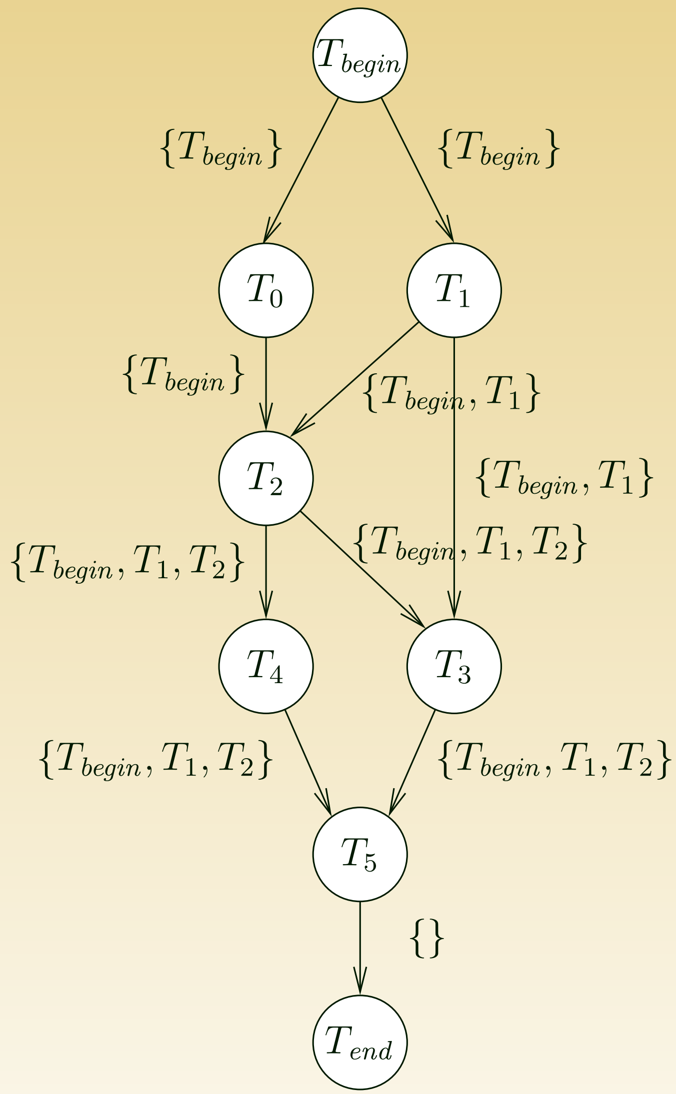
Tâche contraignante

Définition. Pour toute dépendance $e_{k,l}$, une tâche T_a est dite contraignante pour $e_{k,l}$ si T_a est un ancêtre de T_l et s'il existe un T_d tel que :

– T_d est un descendant de T_l,

– il y a un chemin de T_a à T_d sommet-disjoint (excepté T_a et T_d) de celui allant de T_a à T_d en passant par T_l.

Une liste de contraintes représente l'allocation de certains ancêtres (par exemple $\{T_{begin} \mapsto P_1, T_1 \mapsto P_2, T_2 \mapsto P_2\}$).



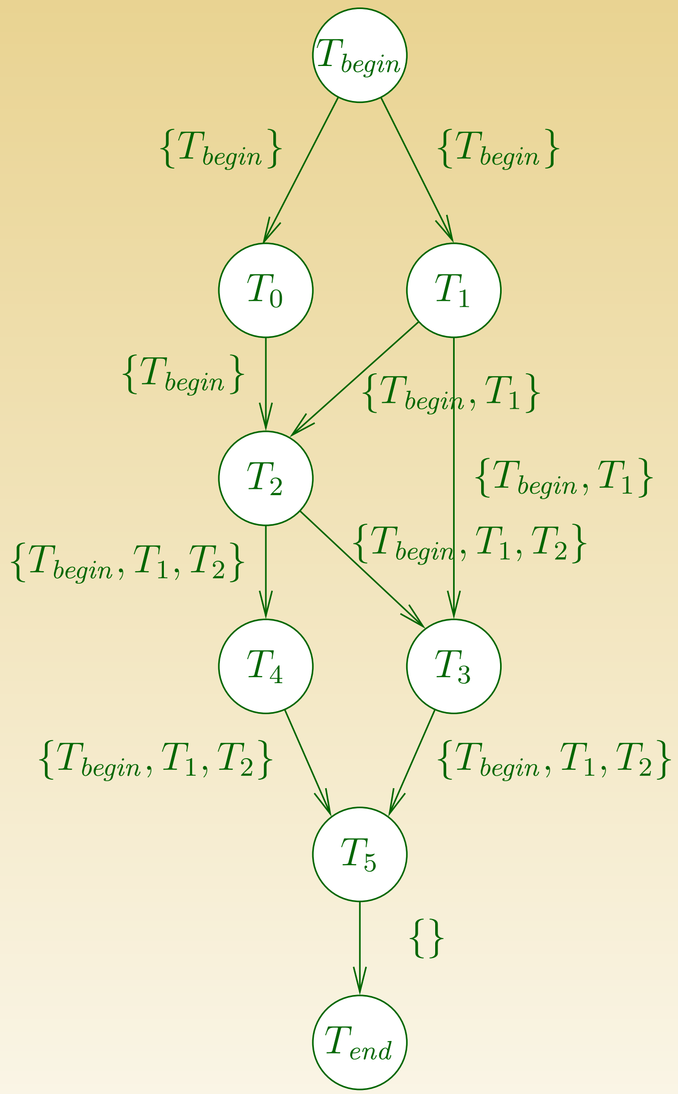
Tâche contraignante

Définition. Pour toute dépendance $e_{k,l}$, une tâche T_a est dite contraignante pour $e_{k,l}$ si T_a est un ancêtre de T_l et s'il existe un T_d tel que :

– T_d est un descendant de T_l,

– il y a un chemin de T_a à T_d sommet-disjoint (excepté T_a et T_d) de celui allant de T_a à T_d en passant par T_l.

Une liste de contraintes représente l'allocation de certains ancêtres (par exemple $\{T_{begin} \mapsto P_1, T_1 \mapsto P_2, T_2 \mapsto P_2\}$).



Listes de contraintes

Les variables $sent(P_i \rightarrow P_j, e_{k,l})$ et $cons(P_i, T_k)$ sont annotées par une liste de contraintes L et s'écrivent donc $sent(P_i \rightarrow P_j, e^L_{k,l})$ et $cons(P_i, T^L_k)$.

Une loi de conservation un peu plus compliquée :

$$\forall P_i, \forall e_{k,l} \in E_A, \forall L \in Cnsts(e_{k,l})$$

$$\sum_{P_j \rightarrow P_i} sent(P_j \rightarrow P_i, e^L_{k,l}) + cons(P_i, T^L_k) =$$

$$\sum_{P_i \rightarrow P_j} sent(P_i \rightarrow P_j, e^L_{k,l}) + \sum_{\substack{L_2 \in CnstsIn(T_l) \\ L \text{ et } L_2 \text{ compatibles}}} prod(P_i, T^{L_2}_l)$$

Listes de contraintes

Les variables $sent(P_i \to P_j, e_{k,l})$ et $cons(P_i, T_k)$ sont annotées par une liste de contraintes L et s'écrivent donc $sent(P_i \to P_j, e_{k,l}^L)$ et $cons(P_i, T_k^L)$.

Une loi de conservation un peu plus compliquée :

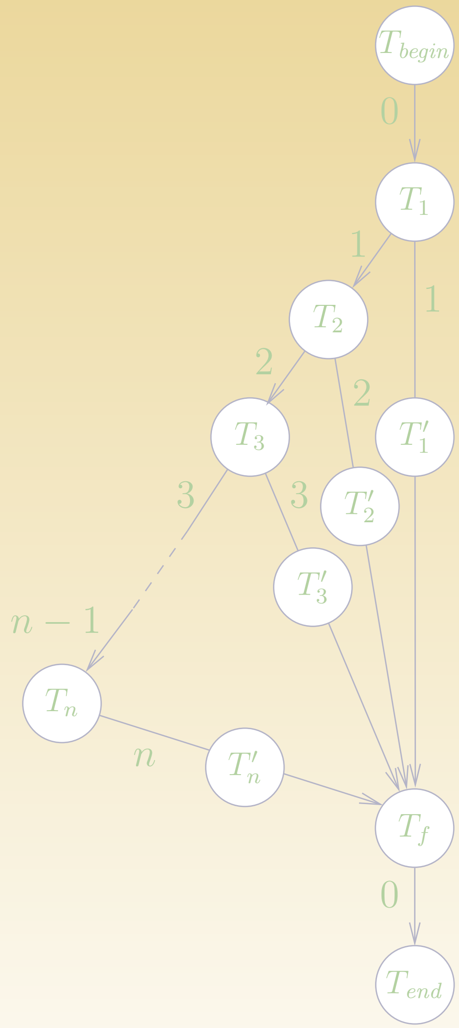
$$\forall P_i, \forall e_{k,l} \in E_A, \forall L \in Cnsts(e_{k,l})$$

$$\sum_{P_j \to P_i} sent(P_j \to P_i, e_{k,l}^L) + cons(P_i, T_k^L) =$$

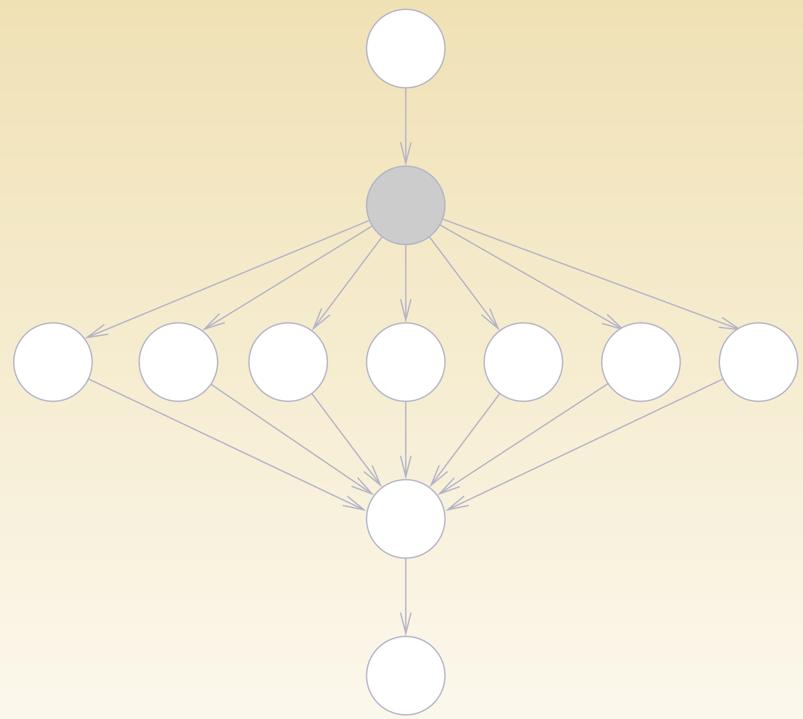
$$\sum_{P_i \to P_j} sent(P_i \to P_j, e_{k,l}^L) + \sum_{\substack{L_2 \in CnstsIn(T_l) \\ L \text{ et } L_2 \text{ compatibles}}} prod(P_i, T_l^{L_2})$$

Profondeur de dépendance

Définition. La profondeur de dépendance est le nombre maximal de tâches contraignantes d'un $e_{k,l}$, sur l'ensemble des $e_{k,l}$.



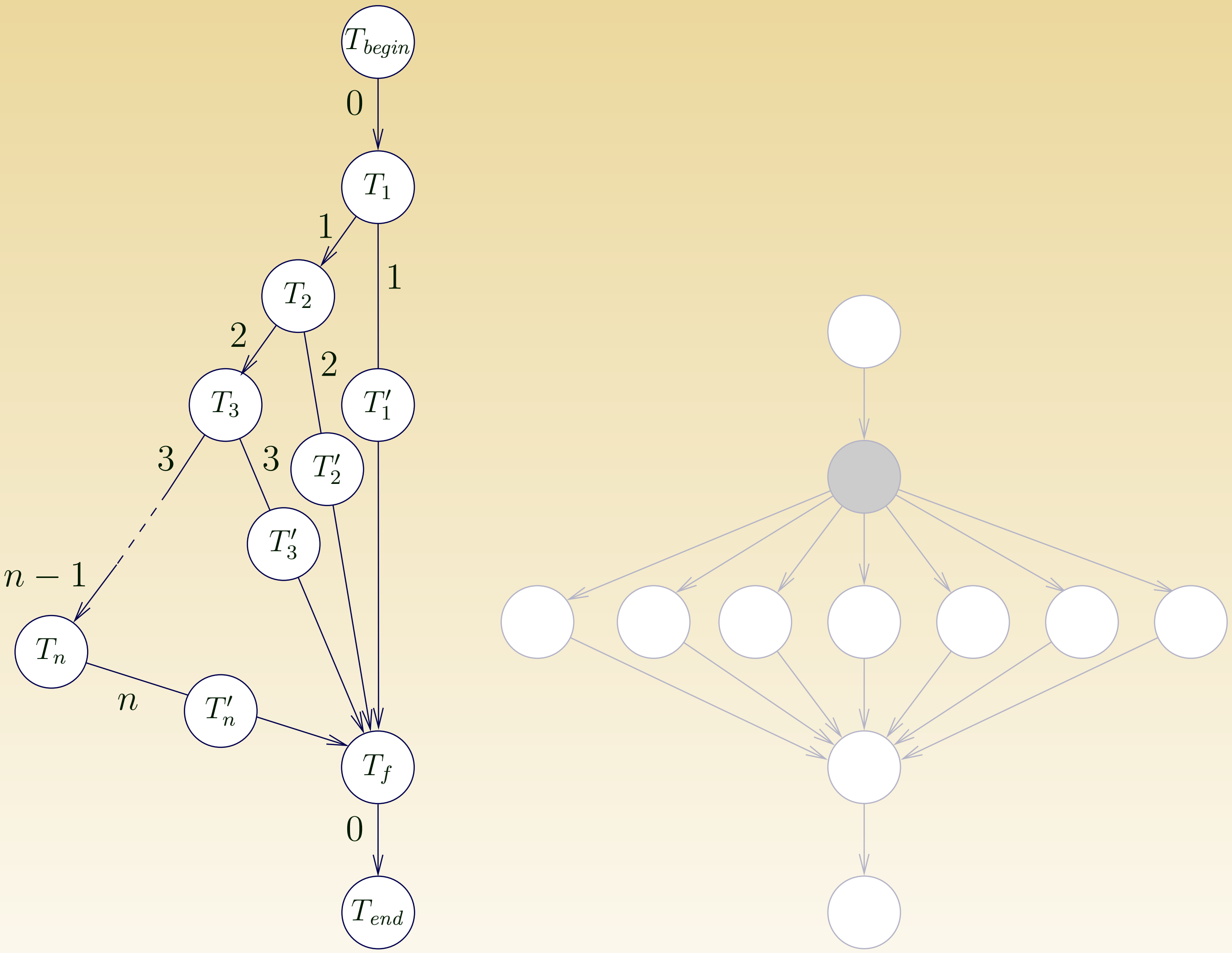
Profondeur : n



Profondeur : 1

Profondeur de dépendance

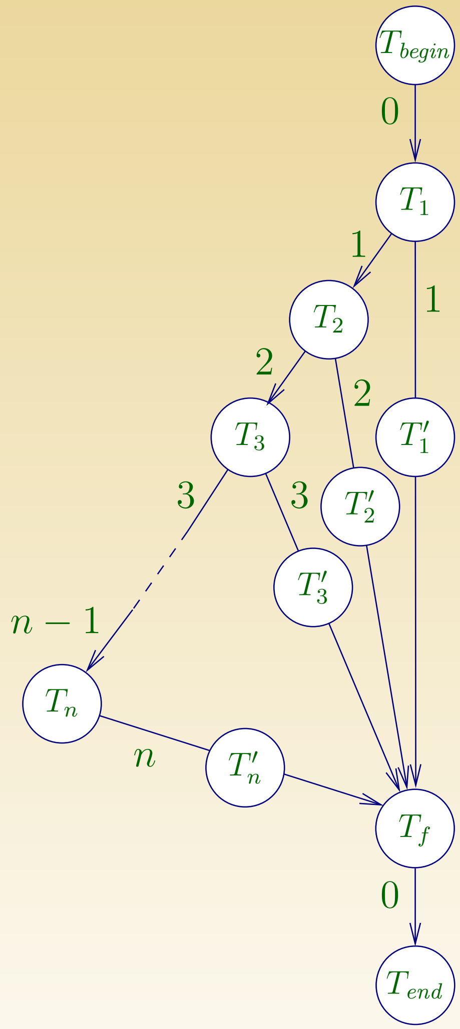
Définition. La profondeur de dépendance est le nombre maximal de tâches contraignantes d'un $e_{k,l}$, sur l'ensemble des $e_{k,l}$.

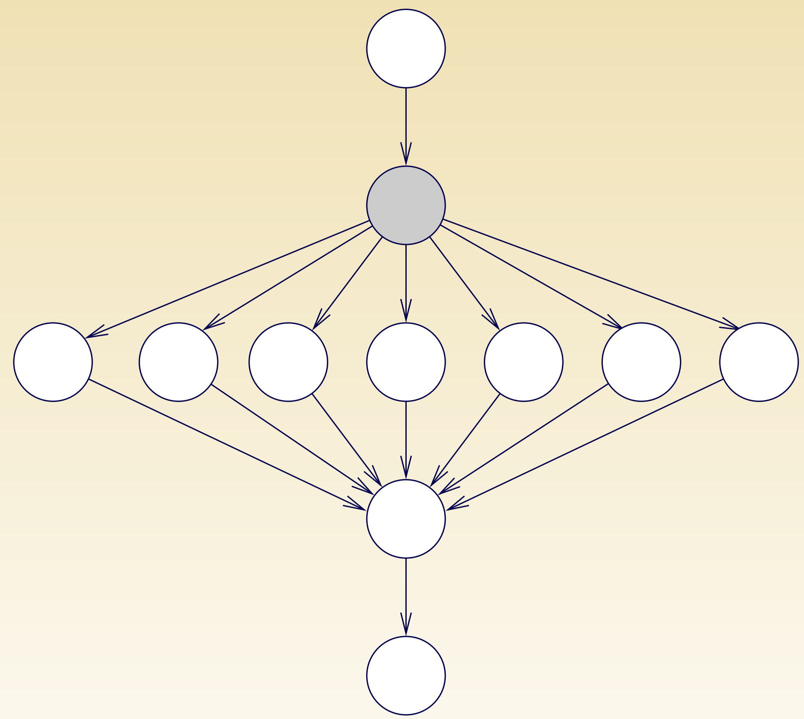


Profondeur : n Profondeur : 1

Profondeur de dépendance

Définition. La profondeur de dépendance est le nombre maximal de tâches contraignantes d'un $e_{k,l}$, sur l'ensemble des $e_{k,l}$.





Profondeur : n Profondeur : 1

Profondeur de dépendance

Définition (Reg-Perm$_{\boldsymbol{d}}(\boldsymbol{G_a}, \boldsymbol{G_p})$). Étant donné un graphe de tâches G_a dont la profondeur de dépendance est bornée par d et un graphe de plate-forme G_p, trouver un ordonnancement périodique réalisant le régime permanent optimal.

Pour tout $d \in \mathbb{N}$, Reg-Perm$_d$ est polynomial.

Question ouverte : Reg-Perm est-il NP-complet ?

Profondeur de dépendance

Définition (Reg-Perm$_d(G_a, G_p)$). Étant donné un graphe de tâches G_a dont la profondeur de dépendance est bornée par d et un graphe de plate-forme G_p, trouver un ordonnancement périodique réalisant le régime permanent optimal.

Pour tout $d \in \mathbb{N}$, Reg-Perm$_d$ est polynomial.

Question ouverte : Reg-Perm est-il NP-complet ?

Profondeur de dépendance

Définition (Reg-Perm$_{d}(G_a, G_p)$). Étant donné un graphe de tâches G_a dont la profondeur de dépendance est bornée par d et un graphe de plate-forme G_p, trouver un ordonnancement périodique réalisant le régime permanent optimal.

Pour tout $d \in \mathbb{N}$, Reg-Perm$_d$ est polynomial.

Question ouverte : Reg-Perm est-il NP-complet ?

Conclusion

- La prise en compte de l'hétérogénéité et de modèles de communications réalistes complique considérablement les problèmes les plus simples.
- Se placer en régime permanent permet de proposer des solutions efficaces, même pour les métriques usuelles, tout en utilisant des modélisations de la plate-forme plus fines.
- Les distributions périodiques peuvent être remises en cause régulièrement pour prendre en compte les variations de charge de la plate-forme.
- Dans le cas où l'on se restreint à des plates-formes en arbre, les équations se simplifient et se résolvent de façon gloutonne, ce qui conduit à un ordonnancement local et dynamique.

Conclusion

- La prise en compte de l'hétérogénéité et de modèles de communications réalistes complique considérablement les problèmes les plus simples.
- Se placer en régime permanent permet de proposer des solutions efficaces, même pour les métriques usuelles, tout en utilisant des modélisations de la plate-forme plus fines.
- Les distributions périodiques peuvent être remises en cause régulièrement pour prendre en compte les variations de charge de la plate-forme.
- Dans le cas où l'on se restreint à des plates-formes en arbre, les équations se simplifient et se résolvent de façon gloutonne, ce qui conduit à un ordonnancement local et dynamique.

Conclusion

- La prise en compte de l'hétérogénéité et de modèles de communications réalistes complique considérablement les problèmes les plus simples.
- Se placer en régime permanent permet de proposer des solutions efficaces, même pour les métriques usuelles, tout en utilisant des modélisations de la plate-forme plus fines.
- Les distributions périodiques peuvent être remises en cause régulièrement pour prendre en compte les variations de charge de la plate-forme.
- Dans le cas où l'on se restreint à des plates-formes en arbre, les équations se simplifient et se résolvent de façon gloutonne, ce qui conduit à un ordonnancement local et dynamique.

Conclusion

- La prise en compte de l'hétérogénéité et de modèles de communications réalistes complique considérablement les problèmes les plus simples.
- Se placer en régime permanent permet de proposer des solutions efficaces, même pour les métriques usuelles, tout en utilisant des modélisations de la plate-forme plus fines.
- Les distributions périodiques peuvent être remises en cause régulièrement pour prendre en compte les variations de charge de la plate-forme.
- Dans le cas où l'on se restreint à des plates-formes en arbre, les équations se simplifient et se résolvent de façon gloutonne, ce qui conduit à un ordonnancement local et dynamique.

Bibliographie

[BLR02] Olivier Beaumont, Arnaud Legrand, and Yves Robert. A polynomial-time algorithm for allocating independent tasks on heterogeneous fork-graphs. In ISCIS XVII, Seventeenth International Symposium On Computer and Information Sciences, pages 115–119. CRC Press, 2002.

[Dut03a] Pierre-François Dutot. Complexity of master-slave tasking on heterogeneous trees. European Journal of Operational Research, 2003. Special issue on the Dagstuhl meeting on Scheduling for Computing and Manufacturing systems (to appear).

[Dut03b] Pierre-François Dutot. Master-slave tasking on heterogeneous processors. In International Parallel and Distributed Processing Symposium IPDPS'2003. IEEE Computer Society Press, 2003.