

Arnaud LEGRAND

Laboratoire de l'Informatique du Parallélisme

Algorithmique parallèle hétérogène
et techniques d'ordonnancement :
approches statiques et dynamiques

Thèse réalisée au LIP sous la direction d'Olivier BEAUMONT et d'Yves ROBERT ○

Évolution des machines parallèles

Une forte demande en puissance de calcul.

Le parallélisme est une tentative de réponse.

L'union fait la force

L'algorithmique et l'ordonnancement étaient déjà difficiles dans le cas homogène. Sur une plate-forme hétérogène, c'est encore pire.

Évolution des machines parallèles

Une forte demande en puissance de calcul.

Le parallélisme est une tentative de réponse.

L'union fait la force

L'algorithmique et l'ordonnancement étaient déjà difficiles dans le cas homogène. Sur une plate-forme hétérogène, c'est encore pire.

Évolution des machines parallèles

Une forte demande en puissance de calcul.

Le parallélisme est une tentative de réponse.

L'union fait la force



L'algorithmique et l'ordonnancement étaient déjà difficiles dans le cas homogène. Sur une plate-forme hétérogène, c'est encore pire.

Évolution des machines parallèles

Une forte demande en puissance de calcul.

Le parallélisme est une tentative de réponse.

L'union fait la force



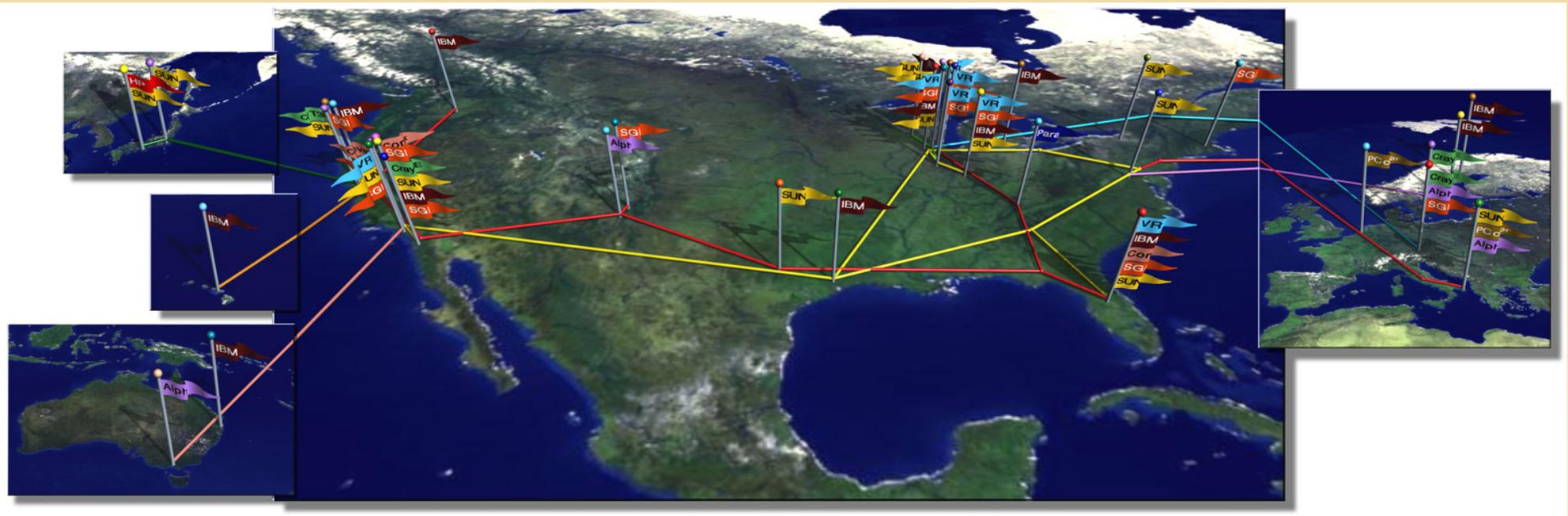
L'algorithmique et l'ordonnancement étaient déjà difficiles dans le cas homogène. Sur une plate-forme hétérogène, c'est encore pire.

Évolution des machines parallèles

Une forte demande en puissance de calcul.

Le parallélisme est une tentative de réponse.

L'union fait la force



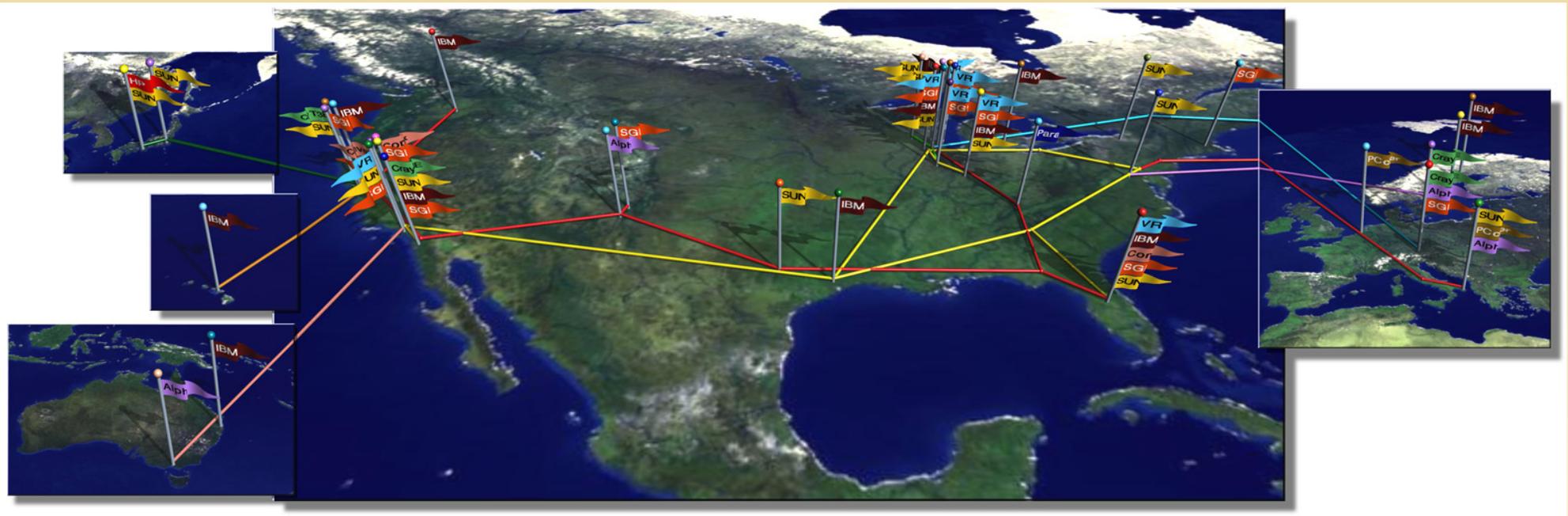
L'algorithmique et l'ordonnancement étaient déjà difficiles dans le cas homogène. Sur une plate-forme hétérogène, c'est encore pire.

Évolution des machines parallèles

Une forte demande en puissance de calcul.

Le parallélisme est une tentative de réponse.

L'union fait la force



L'algorithmique et l'ordonnancement étaient déjà difficiles dans le cas homogène. Sur une plate-forme hétérogène, c'est encore pire.

Contributions

Revisiter des problèmes classiques et prendre en compte l'hétérogénéité de la plate-forme.

Algorithmique parallèle : Principaux noyaux d'algèbre linéaire dense.

- distributions efficaces pour le produit de matrice et la décomposition LU ;
- techniques de rééquilibrage efficaces en cas de petites variations.

Modélisation et simulation : instabilité de la plate-forme \rightsquigarrow non reproductibilité des expériences : SIMGRID (ENS-Lyon + UCSD).

Ordonnancement : Répartir des tâches indépendantes plus ou moins complexes sur une plate-forme hétérogène :

- Ordonnancement classique
- Ordonnancement cyclique
- Tâches divisibles

Contributions

Revisiter des problèmes classiques et prendre en compte l'hétérogénéité de la plate-forme.

Algorithmique parallèle : Principaux noyaux d'algèbre linéaire dense.

- distributions efficaces pour le produit de matrice et la décomposition LU ;
- techniques de rééquilibrage efficaces en cas de petites variations.

Modélisation et simulation : instabilité de la plate-forme \rightsquigarrow non reproductibilité des expériences : SIMGRID (ENS-Lyon + UCSD).

Ordonnement : Répartir des tâches indépendantes plus ou moins complexes sur une plate-forme hétérogène :

- Ordonnement classique
- Ordonnement cyclique
- Tâches divisibles

Contributions

Revisiter des problèmes classiques et prendre en compte l'hétérogénéité de la plate-forme.

Algorithmique parallèle : Principaux noyaux d'algèbre linéaire dense.

- distributions efficaces pour le produit de matrice et la décomposition LU ;
- techniques de rééquilibrage efficaces en cas de petites variations.

Modélisation et simulation : instabilité de la plate-forme \rightsquigarrow non reproductibilité des expériences : SIMGRID (ENS-Lyon + UCSD).

Ordonnement : Répartir des tâches indépendantes plus ou moins complexes sur une plate-forme hétérogène :

- Ordonnement classique
- Ordonnement cyclique
- Tâches divisibles

Contributions

Revisiter des problèmes classiques et prendre en compte l'hétérogénéité de la plate-forme.

Algorithmique parallèle : Principaux noyaux d'algèbre linéaire dense.

- distributions efficaces pour le produit de matrice et la décomposition LU ;
- techniques de rééquilibrage efficaces en cas de petites variations.

Modélisation et simulation : instabilité de la plate-forme \rightsquigarrow non reproductibilité des expériences : SIMGRID (ENS-Lyon + UCSD).

Ordonnancement : Répartir des tâches indépendantes plus ou moins complexes sur une plate-forme hétérogène :

- Ordonnancement classique
- Ordonnancement cyclique
- Tâches divisibles

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.

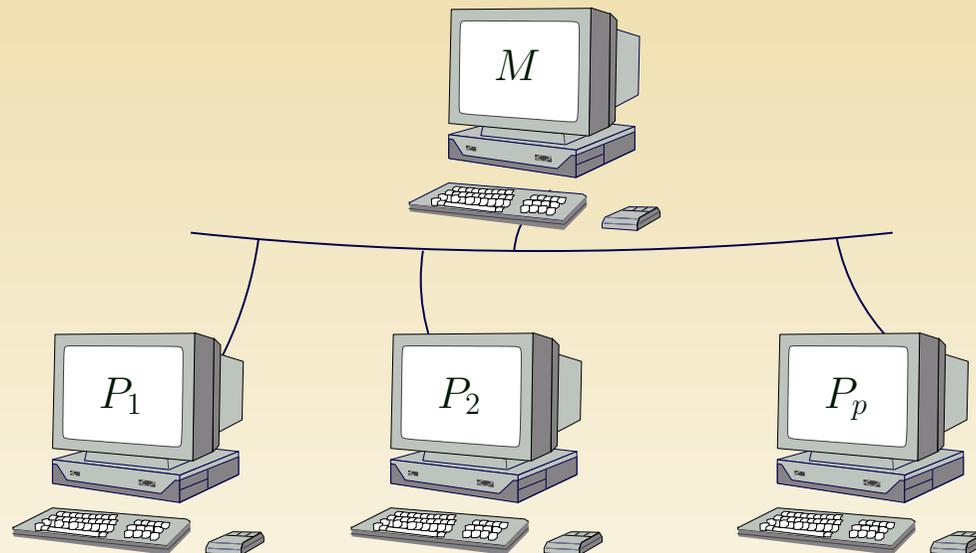
Mise en œuvre classique Un certain nombre de tâches indépendantes sont traitées par des processeurs identiques (les esclaves) sous la supervision d'un processeur particulier (le maître).

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

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).

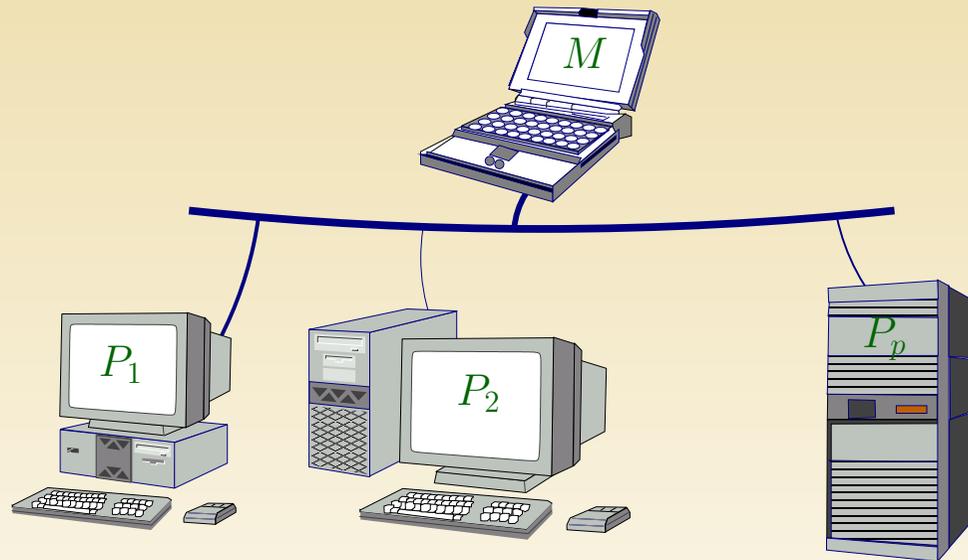


Version hétérogène Les temps de calcul et de communication des esclaves sont différents les uns des autres.

Architecture Maître-esclave

Architecture Maître/esclave Une technique simple mais efficace.

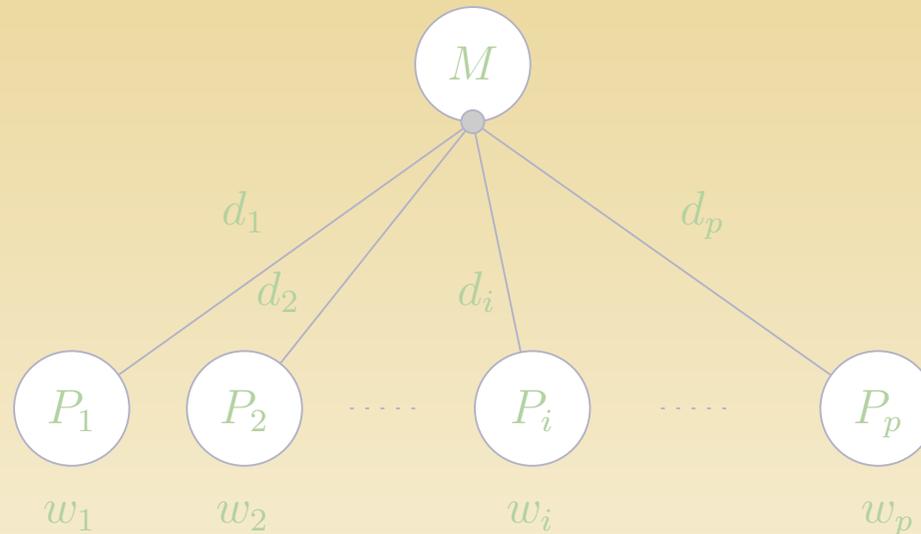
Mise en œuvre classique Un certain nombre de tâches indépendantes sont traitées par des processeurs identiques (les esclaves) sous la supervision d'un processeur particulier (le maître).



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

Modélisation

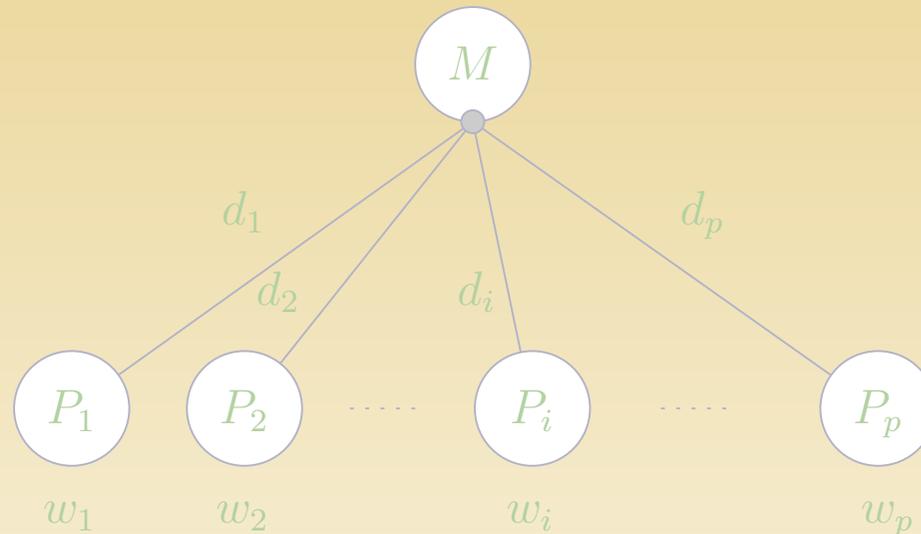
- Ensemble de tâches indépendantes à traiter par les p esclaves.
- Toutes les tâches sont identiques : elles représentent la même quantité de calcul



- 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

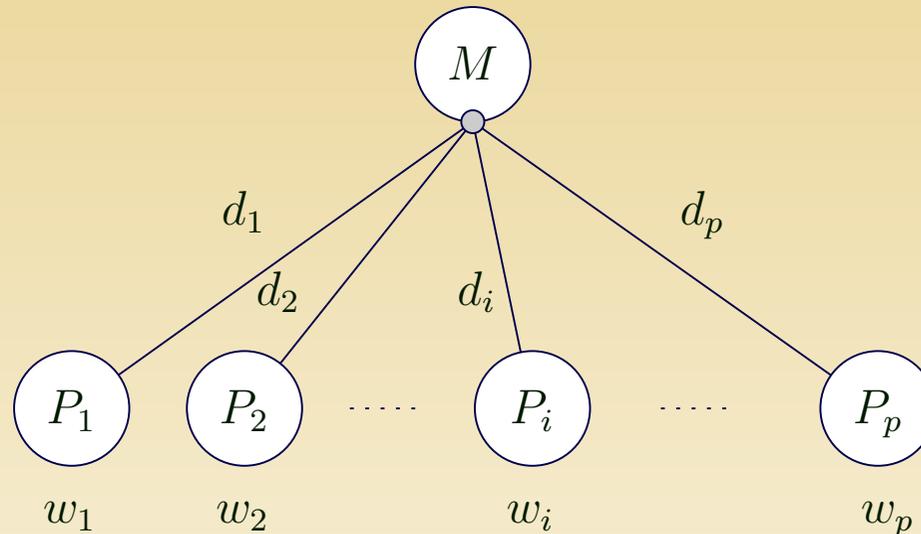
- Ensemble de tâches indépendantes à traiter par les p esclaves.
- Toutes les tâches sont identiques : elles représentent la même quantité de calcul



- 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

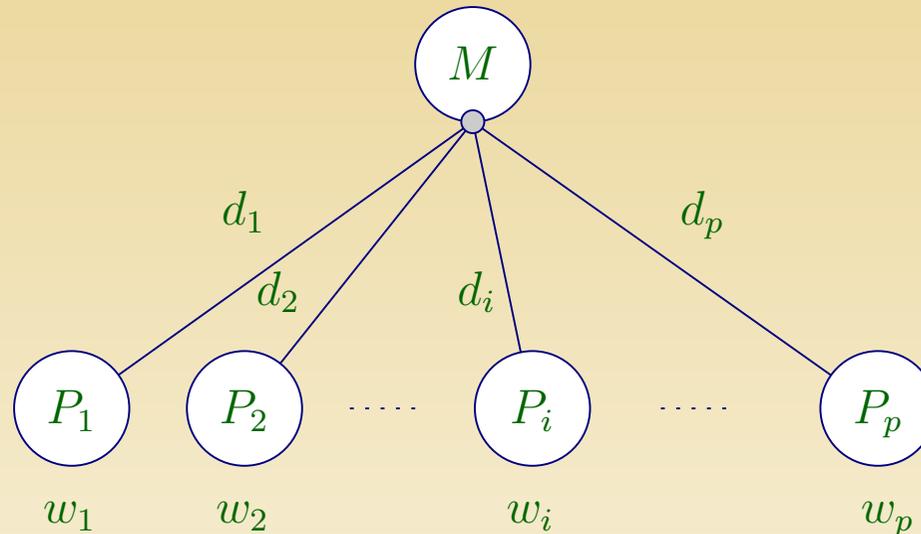
- Ensemble de tâches indépendantes à traiter par les p esclaves.
- Toutes les tâches sont identiques : elles représentent la même quantité de calcul



- 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

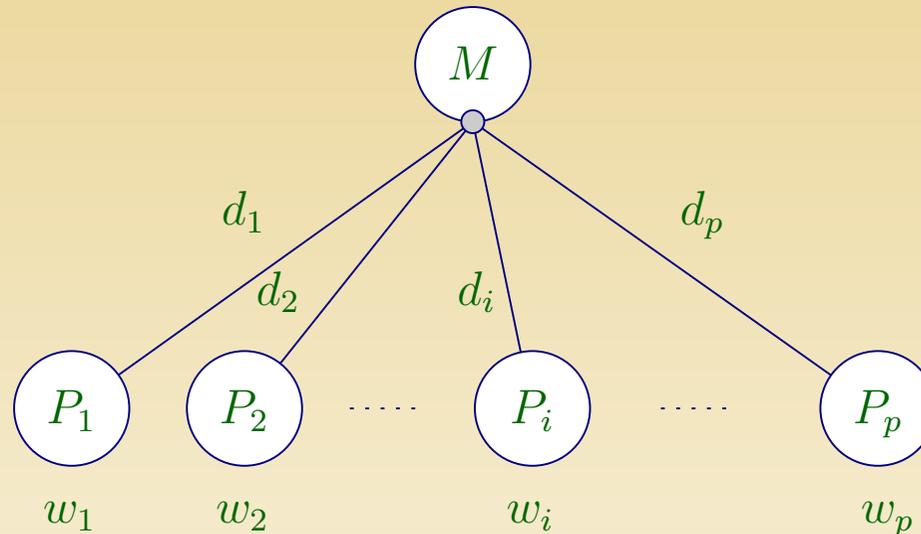
- Ensemble de tâches indépendantes à traiter par les p esclaves.
- Toutes les tâches sont identiques : elles représentent la même quantité de calcul



- 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

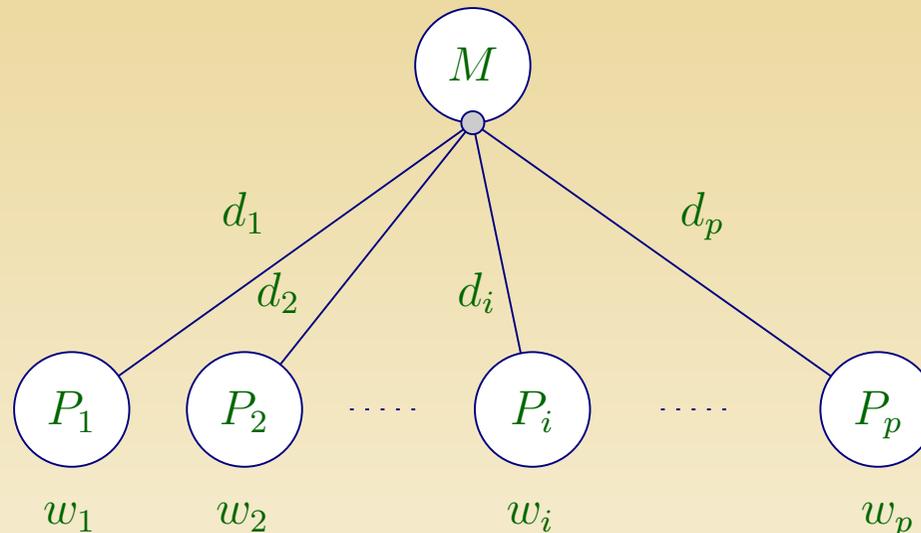
- Ensemble de tâches indépendantes à traiter par les p esclaves.
- Toutes les tâches sont identiques : elles représentent la même quantité de calcul



- 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 $(P_1(d_1, w_1), \dots, P_p(d_p, w_p), T^{(1)}, \dots, T^{(n)})$)).
Étant donné une plate-forme maître-esclave de caractéristique $(d_1, w_1), \dots, (d_p, w_p)$, quel est le temps minimal nécessaire au traitement de n tâches ?

MasterSlave $(P_1(d_1, w_1), \dots, P_p(d_p, w_p), T^{(1)}, \dots, T^{(n)})$ 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 $(P_1(d_1, w_1), \dots, P_p(d_p, w_p), T^{(1)}, \dots, T^{(n)})$)).
Étant donné une plate-forme maître-esclave de caractéristique $(d_1, w_1), \dots, (d_p, w_p)$, quel est le temps minimal nécessaire au traitement de n tâches ?

MasterSlave $(P_1(d_1, w_1), \dots, P_p(d_p, w_p), T^{(1)}, \dots, T^{(n)})$ 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].

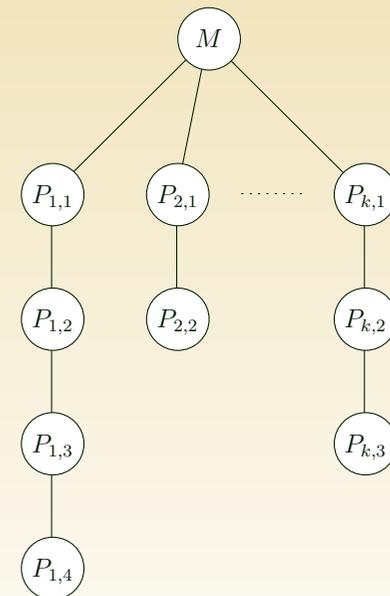
Résultats de complexité

Définition (MasterSlave $(P_1(d_1, w_1), \dots, P_p(d_p, w_p), T^{(1)}, \dots, T^{(n)})$)).
Étant donné une plate-forme maître-esclave de caractéristique $(d_1, w_1), \dots, (d_p, w_p)$, quel est le temps minimal nécessaire au traitement de n tâches ?

MasterSlave $(P_1(d_1, w_1), \dots, P_p(d_p, w_p), T^{(1)}, \dots, T^{(n)})$ 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].



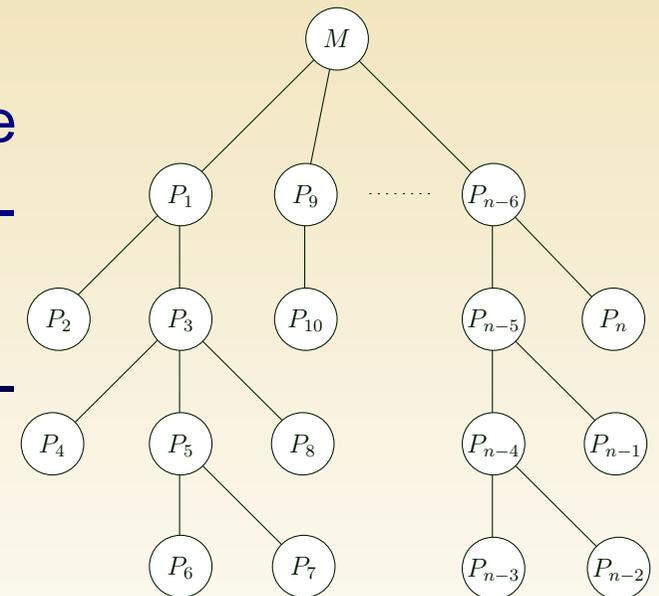
Résultats de complexité

Définition (MasterSlave $(P_1(d_1, w_1), \dots, P_p(d_p, w_p), T^{(1)}, \dots, T^{(n)})$)).
Étant donné une plate-forme maître-esclave de caractéristique $(d_1, w_1), \dots, (d_p, w_p)$, quel est le temps minimal nécessaire au traitement de n tâches ?

MasterSlave $(P_1(d_1, w_1), \dots, P_p(d_p, w_p), T^{(1)}, \dots, T^{(n)})$ 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].



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 plates-formes bien plus complexes (avec des chemins multiples...) 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 plates-formes bien plus complexes (avec des chemins multiples...) 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 plates-formes bien plus complexes (avec des chemins multiples...) 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 plates-formes bien plus complexes (avec des chemins multiples...) 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 :

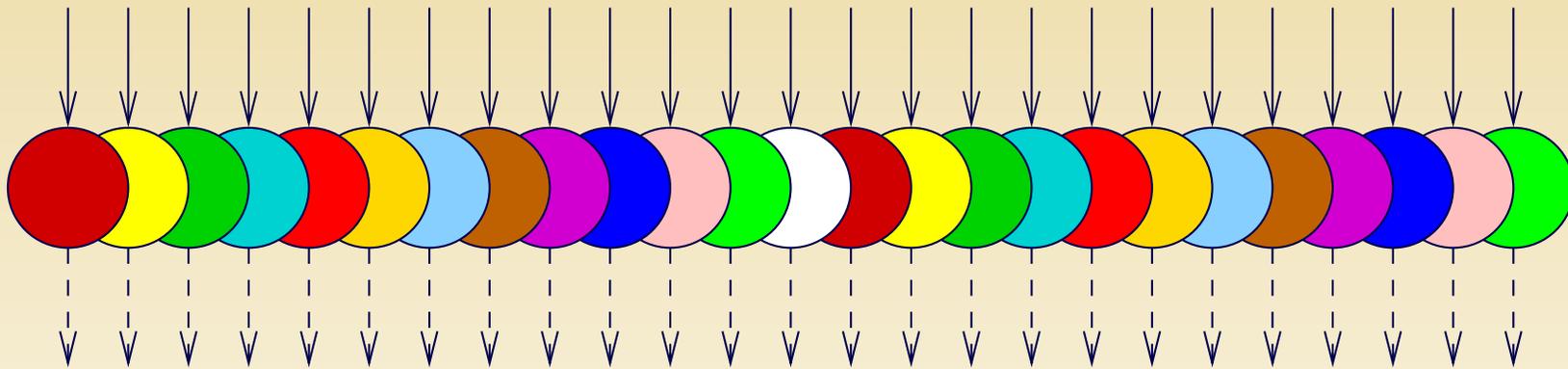
- la modélisation d'une grille de calcul ainsi que la mesure des différents paramètres est extrêmement délicate ;
- étant donné le temps de déploiement et la difficulté de mise en œuvre de telles plates-formes, le nombre de tâches à traiter est généralement très grand.

En se plaçant en régime permanent, on peut considérer des plates-formes bien plus complexes (avec des chemins multiples...) tout en construisant des ordonnancements efficaces.

Tâches indépendantes : **modélisation**

Graphe de tâches

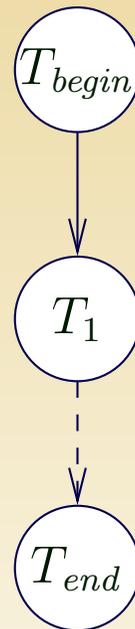
On dispose de n problèmes $\mathcal{P}^{(1)}, \mathcal{P}^{(2)}, \dots, \mathcal{P}^{(n)}$ à résoudre avec n grand.
Chaque problème correspond à une copie d'un même graphe de tâches $G_A = (V_A, E_A)$, le graphe d'application.



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

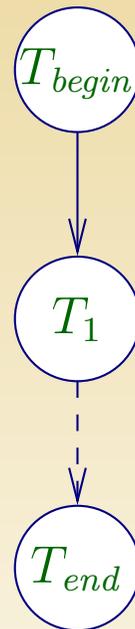
On dispose de n problèmes $\mathcal{P}^{(1)}, \mathcal{P}^{(2)}, \dots, \mathcal{P}^{(n)}$ à résoudre avec n grand. Chaque problème correspond à une copie d'un même graphe de tâches $G_A = (V_A, E_A)$, le graphe d'application.



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

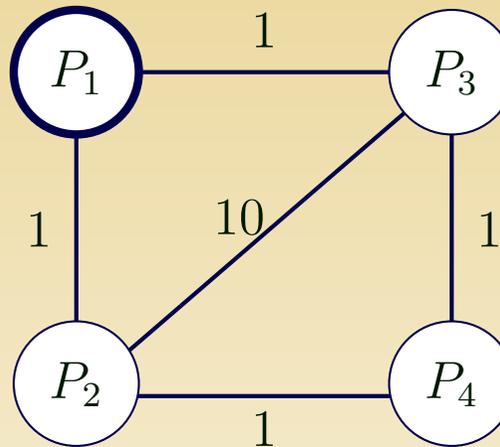
On dispose de n problèmes $\mathcal{P}^{(1)}, \mathcal{P}^{(2)}, \dots, \mathcal{P}^{(n)}$ à résoudre avec n grand. Chaque problème correspond à une copie d'un même graphe de tâches $G_A = (V_A, E_A)$, le graphe d'application.



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 Plate-forme

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

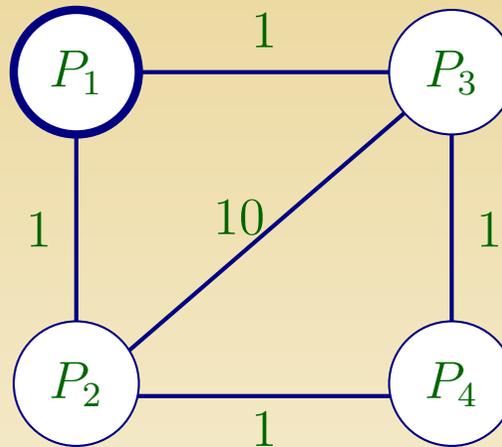


Chaque arête $P_i \rightarrow P_j$ est étiquetée par $c_{i,j}$: le temps nécessaire à l'envoi d'un message de taille unitaire entre P_i et P_j .

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

Graphe de Plate-forme

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

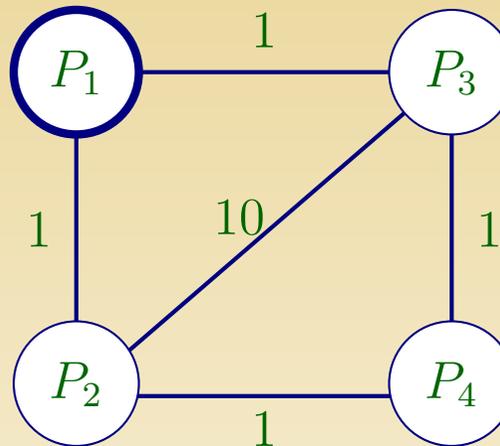


Chaque arête $P_i \rightarrow P_j$ est étiquetée par $c_{i,j}$: le temps nécessaire à l'envoi d'un message de taille unitaire entre P_i et P_j .

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

Graphe de Plate-forme

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

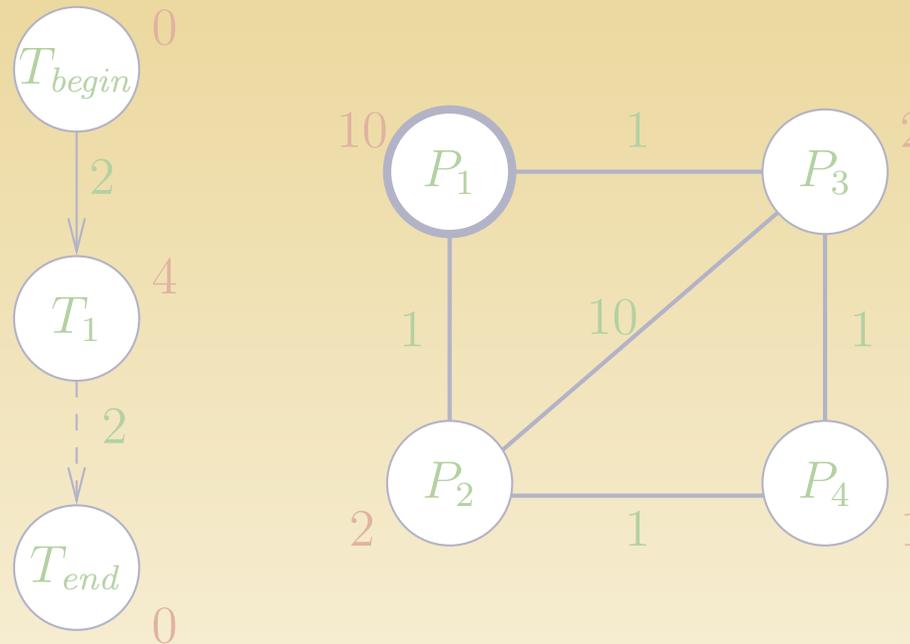


Chaque arête $P_i \rightarrow P_j$ est étiquetée par $c_{i,j}$: le temps nécessaire à l'envoi d'un message de taille unitaire entre P_i et P_j .

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 ($k \in \{begin, 1, end\}$).

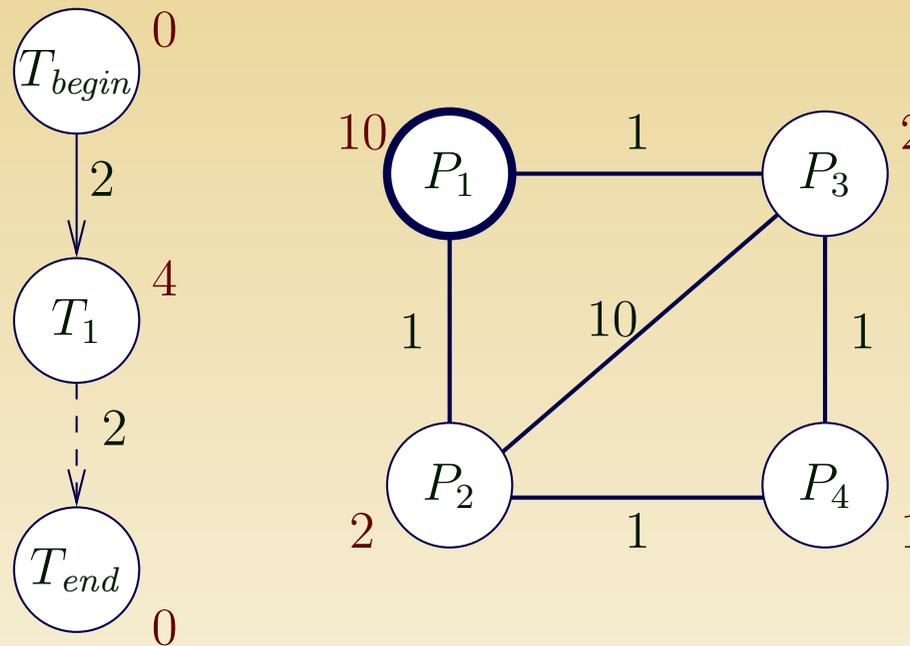


Chaque arête $e_{k,l} : T_k \rightarrow T_l$ de G_A est étiquetée par un coût de communication $data_{k,l}$ représentant la quantité de données créées par T_k et utilisées par T_l .

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 ($k \in \{begin, 1, end\}$).

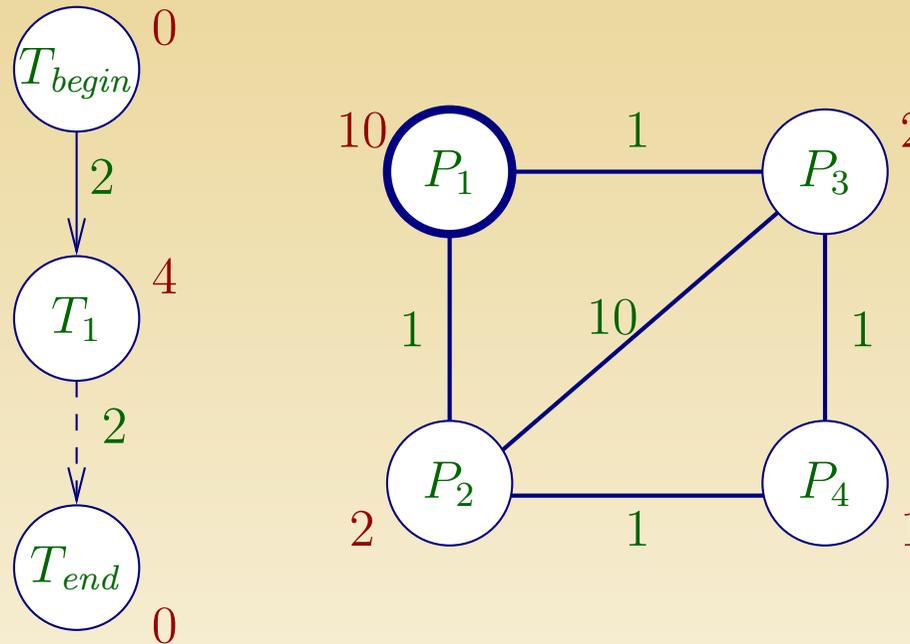


Chaque arête $e_{k,l} : T_k \rightarrow T_l$ de G_A est étiquetée par un coût de communication $data_{k,l}$ représentant la quantité de données créées par T_k et utilisées par T_l .

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 ($k \in \{begin, 1, end\}$).

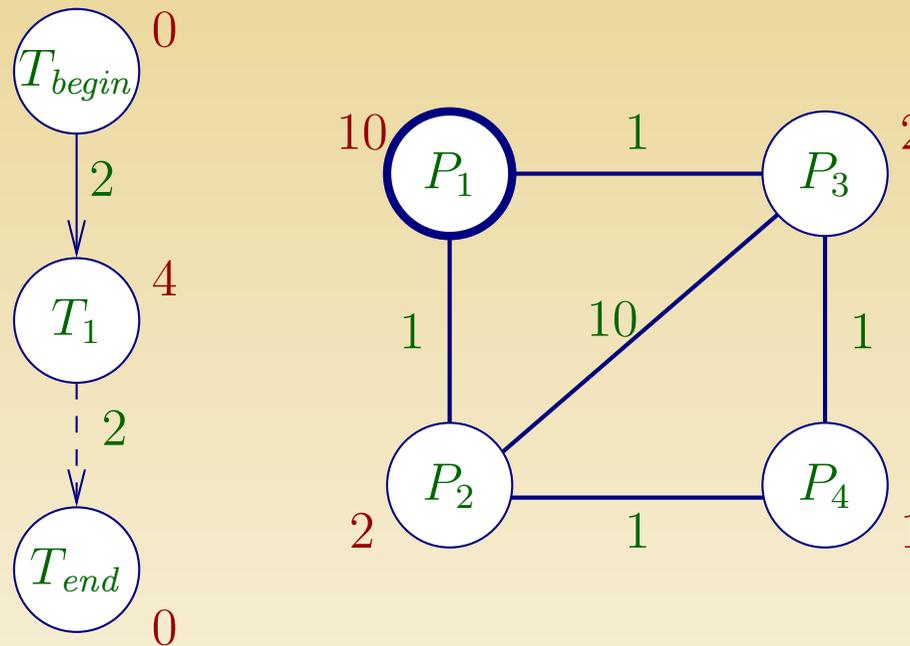


Chaque arête $e_{k,l} : T_k \rightarrow T_l$ de G_A est étiquetée par un coût de communication $data_{k,l}$ représentant la quantité de données créées par T_k et utilisées par T_l .

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 ($k \in \{begin, 1, end\}$).



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 précédence
- les contraintes d'utilisation des processeurs
- les contraintes d'utilisation du réseau
- les contraintes 1-port

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 précédence
- les contraintes d'utilisation des processeurs
- les contraintes d'utilisation du réseau
- les contraintes 1-port

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 précédence
- les contraintes d'utilisation des processeurs
- les contraintes d'utilisation du réseau
- les contraintes 1-port

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 précédence
- les contraintes d'utilisation des processeurs
- les contraintes d'utilisation du réseau
- les contraintes 1-port

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 précédence
- les contraintes d'utilisation des processeurs
- les contraintes d'utilisation du réseau
- les contraintes 1-port

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 précédence
- les contraintes d'utilisation des processeurs
- les contraintes d'utilisation du réseau
- les contraintes 1-port

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 .

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

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

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

Définitions

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

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

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

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

Définitions

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

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

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

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

Définitions

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

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

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

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

Équations du régime permanent

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

$$\forall P_i, \sum_{P_i \rightarrow P_j} \sum_{e_{k,l} \in E_A} (\text{sent}(P_i \rightarrow P_j, e_{k,l}) \times \text{data}_{k,l} \times c_{i,j}) \leq 1 \quad (3)$$

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

$$\forall P_i, \sum_{P_j \rightarrow P_i} \sum_{e_{k,l} \in E_A} (\text{sent}(P_j \rightarrow P_i, e_{k,l}) \times \text{data}_{k,l} \times c_{j,i}) \leq 1 \quad (4)$$

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

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

Équations du régime permanent

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

$$\forall P_i, \sum_{P_i \rightarrow P_j} \sum_{e_{k,l} \in E_A} (\text{sent}(P_i \rightarrow P_j, e_{k,l}) \times \text{data}_{k,l} \times c_{i,j}) \leq 1 \quad (3)$$

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

$$\forall P_i, \sum_{P_j \rightarrow P_i} \sum_{e_{k,l} \in E_A} (\text{sent}(P_j \rightarrow P_i, e_{k,l}) \times \text{data}_{k,l} \times c_{j,i}) \leq 1 \quad (4)$$

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

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

Équations du régime permanent

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

$$\forall P_i, \sum_{P_i \rightarrow P_j} \sum_{e_{k,l} \in E_A} (\text{sent}(P_i \rightarrow P_j, e_{k,l}) \times \text{data}_{k,l} \times c_{i,j}) \leq 1 \quad (3)$$

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

$$\forall P_i, \sum_{P_j \rightarrow P_i} \sum_{e_{k,l} \in E_A} (\text{sent}(P_j \rightarrow P_i, e_{k,l}) \times \text{data}_{k,l} \times c_{j,i}) \leq 1 \quad (4)$$

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

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

Équations du régime permanent

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

$$\forall P_i, \sum_{P_i \rightarrow P_j} \sum_{e_{k,l} \in E_A} (\text{sent}(P_i \rightarrow P_j, e_{k,l}) \times \text{data}_{k,l} \times c_{i,j}) \leq 1 \quad (3)$$

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

$$\forall P_i, \sum_{P_j \rightarrow P_i} \sum_{e_{k,l} \in E_A} (\text{sent}(P_j \rightarrow P_i, e_{k,l}) \times \text{data}_{k,l} \times c_{j,i}) \leq 1 \quad (4)$$

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

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

Équations du régime permanent

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

$$\forall P_i, \sum_{P_i \rightarrow P_j} \sum_{e_{k,l} \in E_A} (\text{sent}(P_i \rightarrow P_j, e_{k,l}) \times \text{data}_{k,l} \times c_{i,j}) \leq 1 \quad (3)$$

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

$$\forall P_i, \sum_{P_j \rightarrow P_i} \sum_{e_{k,l} \in E_A} (\text{sent}(P_j \rightarrow P_i, e_{k,l}) \times \text{data}_{k,l} \times c_{j,i}) \leq 1 \quad (4)$$

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

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

Équations du régime permanent

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

$$\forall P_i, \sum_{P_i \rightarrow P_j} \sum_{e_{k,l} \in E_A} (\text{sent}(P_i \rightarrow P_j, e_{k,l}) \times \text{data}_{k,l} \times c_{i,j}) \leq 1 \quad (3)$$

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

$$\forall P_i, \sum_{P_j \rightarrow P_i} \sum_{e_{k,l} \in E_A} (\text{sent}(P_j \rightarrow P_i, e_{k,l}) \times \text{data}_{k,l} \times c_{j,i}) \leq 1 \quad (4)$$

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

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

Loi de conservation

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

Fichiers de type $e_{k,l}$ reçus : $\sum_{P_j \rightarrow P_i} sent(P_j \rightarrow P_i, e_{k,l})$

Fichiers de type $e_{k,l}$ créés : $cons(P_i, T_k)$

Fichiers de type $e_{k,l}$ consommés : $cons(P_i, T_l)$

Fichiers de type $e_{k,l}$ émis : $\sum_{P_i \rightarrow P_j} sent(P_i \rightarrow P_j, e_{k,l})$

En régime permanent :

$$\forall P_i, \forall e_{k,l} : T_k \rightarrow T_l \in E_A,$$

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

$$\sum_{P_i \rightarrow P_j} sent(P_i \rightarrow P_j, e_{k,l}) + cons(P_i, T_l) \quad (6)$$

Loi de conservation

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

Fichiers de type $e_{k,l}$ reçus : $\sum_{P_j \rightarrow P_i} sent(P_j \rightarrow P_i, e_{k,l})$

Fichiers de type $e_{k,l}$ créés : $cons(P_i, T_k)$

Fichiers de type $e_{k,l}$ consommés : $cons(P_i, T_l)$

Fichiers de type $e_{k,l}$ émis : $\sum_{P_i \rightarrow P_j} sent(P_i \rightarrow P_j, e_{k,l})$

En régime permanent :

$$\forall P_i, \forall e_{k,l} : T_k \rightarrow T_l \in E_A,$$

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

$$\sum_{P_i \rightarrow P_j} sent(P_i \rightarrow P_j, e_{k,l}) + cons(P_i, T_l) \quad (6)$$

Loi de conservation

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

Fichiers de type $e_{k,l}$ reçus : $\sum_{P_j \rightarrow P_i} sent(P_j \rightarrow P_i, e_{k,l})$

Fichiers de type $e_{k,l}$ créés : $cons(P_i, T_k)$

Fichiers de type $e_{k,l}$ consommés : $cons(P_i, T_l)$

Fichiers de type $e_{k,l}$ émis : $\sum_{P_i \rightarrow P_j} sent(P_i \rightarrow P_j, e_{k,l})$

En régime permanent :

$$\forall P_i, \forall e_{k,l} : T_k \rightarrow T_l \in E_A,$$

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

$$\sum_{P_i \rightarrow P_j} sent(P_i \rightarrow P_j, e_{k,l}) + cons(P_i, T_l) \quad (6)$$

Loi de conservation

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

Fichiers de type $e_{k,l}$ reçus : $\sum_{P_j \rightarrow P_i} sent(P_j \rightarrow P_i, e_{k,l})$

Fichiers de type $e_{k,l}$ créés : $cons(P_i, T_k)$

Fichiers de type $e_{k,l}$ consommés : $cons(P_i, T_l)$

Fichiers de type $e_{k,l}$ émis : $\sum_{P_i \rightarrow P_j} sent(P_i \rightarrow P_j, e_{k,l})$

En régime permanent :

$$\forall P_i, \forall e_{k,l} : T_k \rightarrow T_l \in E_A,$$

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

$$\sum_{P_i \rightarrow P_j} sent(P_i \rightarrow P_j, e_{k,l}) + cons(P_i, T_l) \quad (6)$$

Loi de conservation

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

Fichiers de type $e_{k,l}$ reçus : $\sum_{P_j \rightarrow P_i} sent(P_j \rightarrow P_i, e_{k,l})$

Fichiers de type $e_{k,l}$ créés : $cons(P_i, T_k)$

Fichiers de type $e_{k,l}$ consommés : $cons(P_i, T_l)$

Fichiers de type $e_{k,l}$ émis : $\sum_{P_i \rightarrow P_j} sent(P_i \rightarrow P_j, e_{k,l})$

En régime permanent :

$$\forall P_i, \forall e_{k,l} : T_k \rightarrow T_l \in E_A,$$

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

$$\sum_{P_i \rightarrow P_j} sent(P_i \rightarrow P_j, e_{k,l}) + cons(P_i, T_l) \quad (6)$$

Loi de conservation

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

Fichiers de type $e_{k,l}$ reçus : $\sum_{P_j \rightarrow P_i} sent(P_j \rightarrow P_i, e_{k,l})$

Fichiers de type $e_{k,l}$ créés : $cons(P_i, T_k)$

Fichiers de type $e_{k,l}$ consommés : $cons(P_i, T_l)$

Fichiers de type $e_{k,l}$ émis : $\sum_{P_i \rightarrow P_j} sent(P_i \rightarrow P_j, e_{k,l})$

En régime permanent :

$$\forall P_i, \forall e_{k,l} : T_k \rightarrow T_l \in E_A,$$

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

$$\sum_{P_i \rightarrow P_j} sent(P_i \rightarrow P_j, e_{k,l}) + cons(P_i, T_l) \quad (6)$$

Borne supérieure du débit

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

SOUS LES CONTRAINTES

$$\left\{ \begin{array}{l} (7a) \quad \forall P_i, \forall T_k \in V_A, 0 \leq \text{cons}(P_i, T_k) \times w_{i,k} \leq 1 \\ (7b) \quad \forall P_i, P_j, 0 \leq \text{sent}(P_i \rightarrow P_j, e_{k,l}) \times (\text{data}_{k,l} \times c_{i,j}) \leq 1 \\ (7c) \quad \forall P_i, \sum_{P_i \rightarrow P_j} \sum_{e_{k,l} \in E_A} (\text{sent}(P_i \rightarrow P_j, e_{k,l}) \times \text{data}_{k,l} \times c_{i,j}) \leq 1 \\ (7d) \quad \forall P_i, \sum_{P_j \rightarrow P_i} \sum_{e_{k,l} \in E_A} (\text{sent}(P_j \rightarrow P_i, e_{k,l}) \times \text{data}_{k,l} \times c_{j,i}) \leq 1 \\ (7e) \quad \forall P_i, \sum_{T_k \in V_A} \text{cons}(P_i, T_k) \times w_{i,k} \leq 1 \\ (7f) \quad \forall P_i, \forall e_{k,l} \in E_A : T_k \rightarrow T_l, \\ \qquad \qquad \qquad \sum_{P_j \rightarrow P_i} \text{sent}(P_j \rightarrow P_i, e_{k,l}) + \text{cons}(P_i, T_k) = \\ \qquad \qquad \qquad \sum_{P_i \rightarrow P_j} \text{sent}(P_i \rightarrow P_j, e_{k,l}) + \text{cons}(P_i, T_l) \end{array} \right.$$

Comment construire un ordonnancement qui atteigne ce débit ?

Borne supérieure du débit

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

SOUS LES CONTRAINTES

$$\left\{ \begin{array}{l} (7a) \quad \forall P_i, \forall T_k \in V_A, 0 \leq \text{cons}(P_i, T_k) \times w_{i,k} \leq 1 \\ (7b) \quad \forall P_i, P_j, 0 \leq \text{sent}(P_i \rightarrow P_j, e_{k,l}) \times (\text{data}_{k,l} \times c_{i,j}) \leq 1 \\ (7c) \quad \forall P_i, \sum_{P_i \rightarrow P_j} \sum_{e_{k,l} \in E_A} (\text{sent}(P_i \rightarrow P_j, e_{k,l}) \times \text{data}_{k,l} \times c_{i,j}) \leq 1 \\ (7d) \quad \forall P_i, \sum_{P_j \rightarrow P_i} \sum_{e_{k,l} \in E_A} (\text{sent}(P_j \rightarrow P_i, e_{k,l}) \times \text{data}_{k,l} \times c_{j,i}) \leq 1 \\ (7e) \quad \forall P_i, \sum_{T_k \in V_A} \text{cons}(P_i, T_k) \times w_{i,k} \leq 1 \\ (7f) \quad \forall P_i, \forall e_{k,l} \in E_A : T_k \rightarrow T_l, \\ \qquad \qquad \qquad \sum_{P_j \rightarrow P_i} \text{sent}(P_j \rightarrow P_i, e_{k,l}) + \text{cons}(P_i, T_k) = \\ \qquad \qquad \qquad \qquad \qquad \qquad \qquad \qquad \qquad \sum_{P_i \rightarrow P_j} \text{sent}(P_i \rightarrow P_j, e_{k,l}) + \text{cons}(P_i, T_l) \end{array} \right.$$

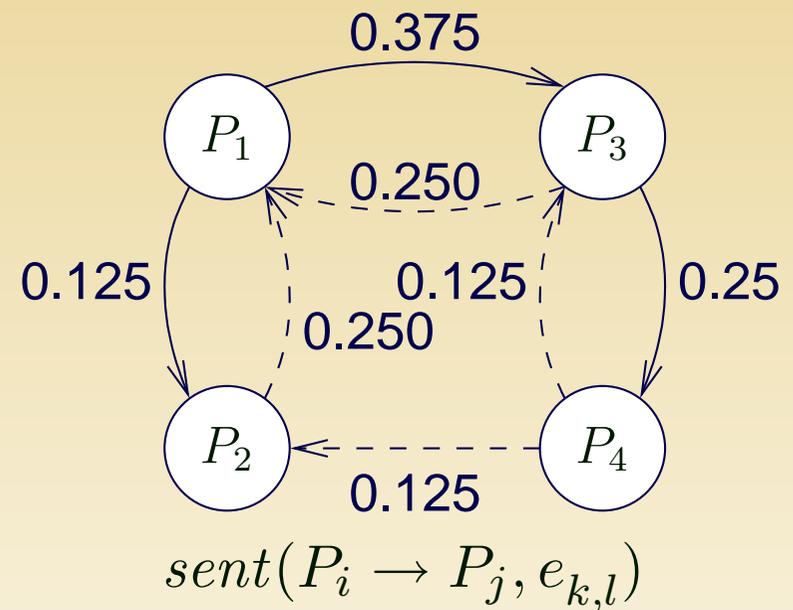
Comment construire un ordonnancement qui atteigne ce débit ?

Retour sur notre exemple

Répartition des calculs

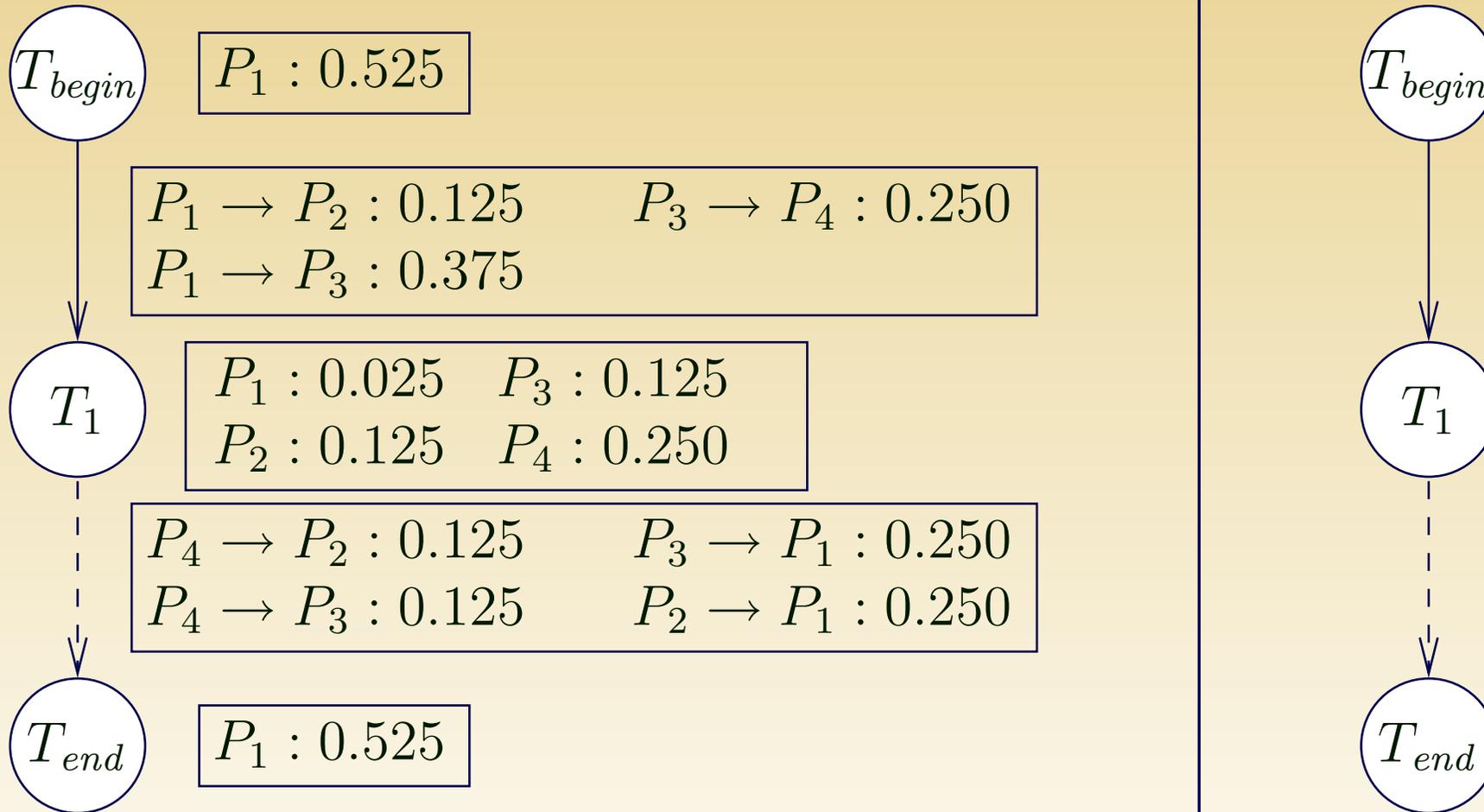
	$cons(P_i, T_1)$
P_1	0.025
P_2	0.125
P_3	0.125
P_4	0.250
Total	21 tâches/ 40 secondes

Communications



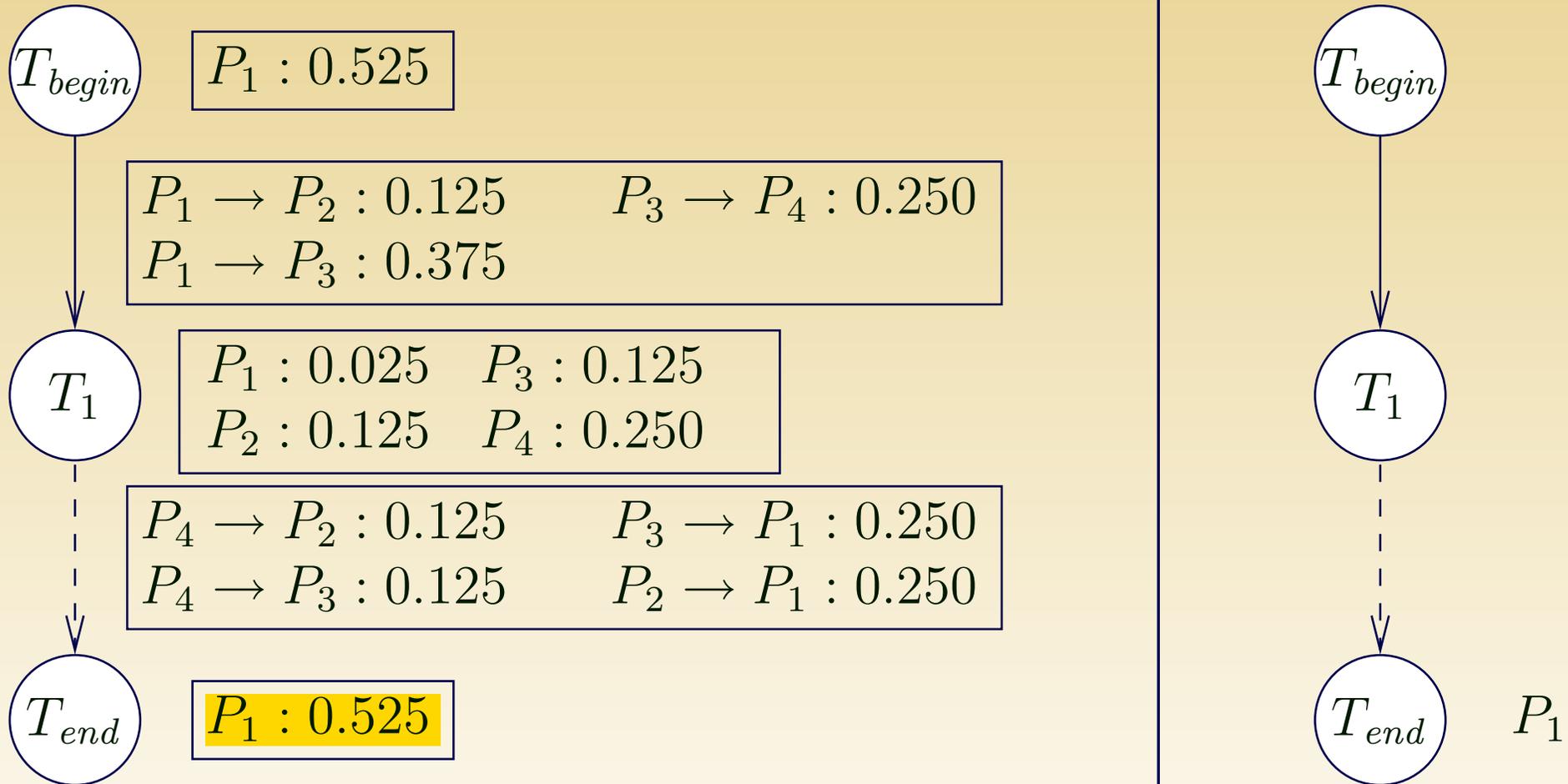
Décomposition en allocations

Régime permanent = superposition de plusieurs allocations.



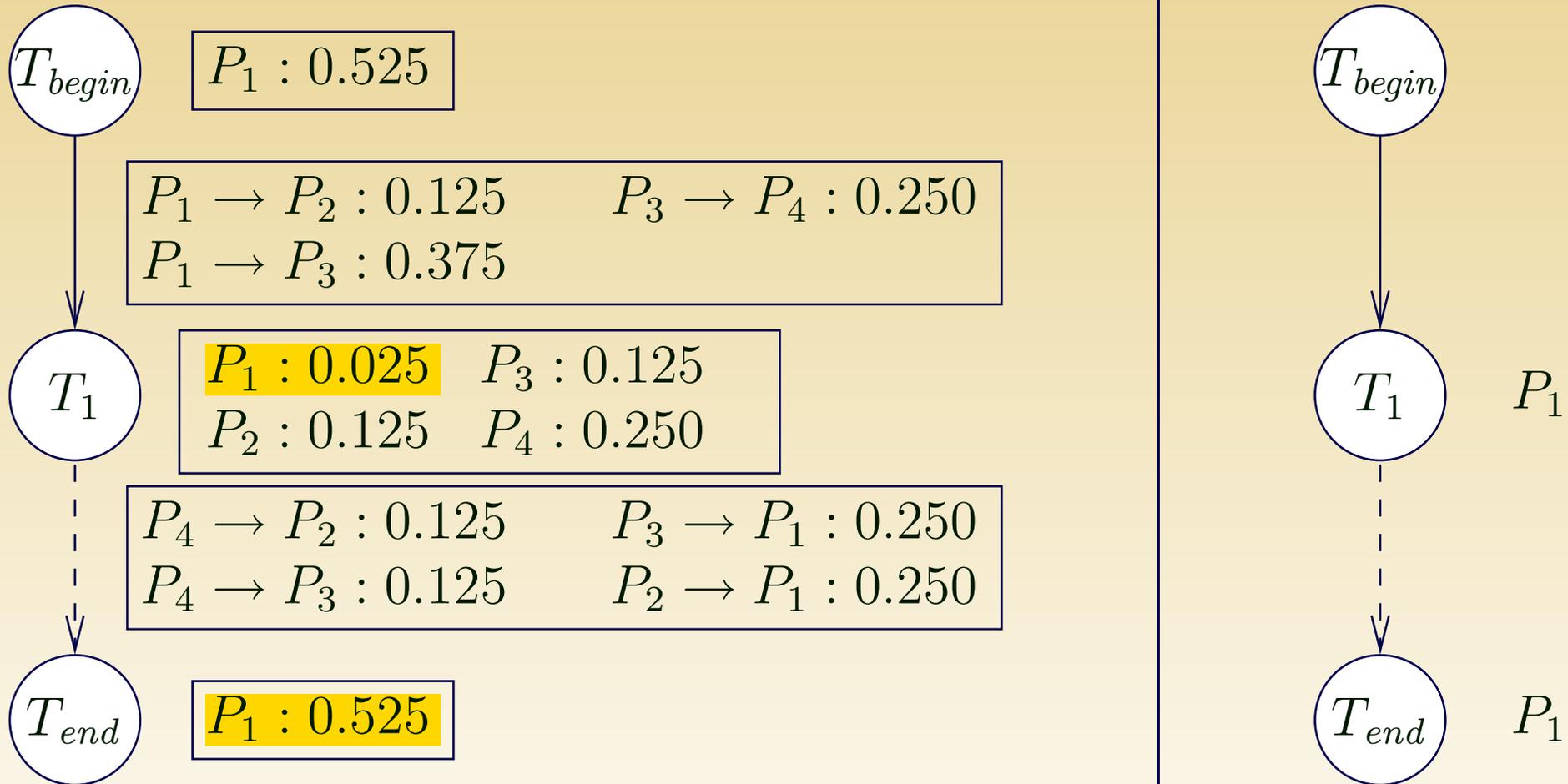
Décomposition en allocations

Régime permanent = superposition de plusieurs allocations.



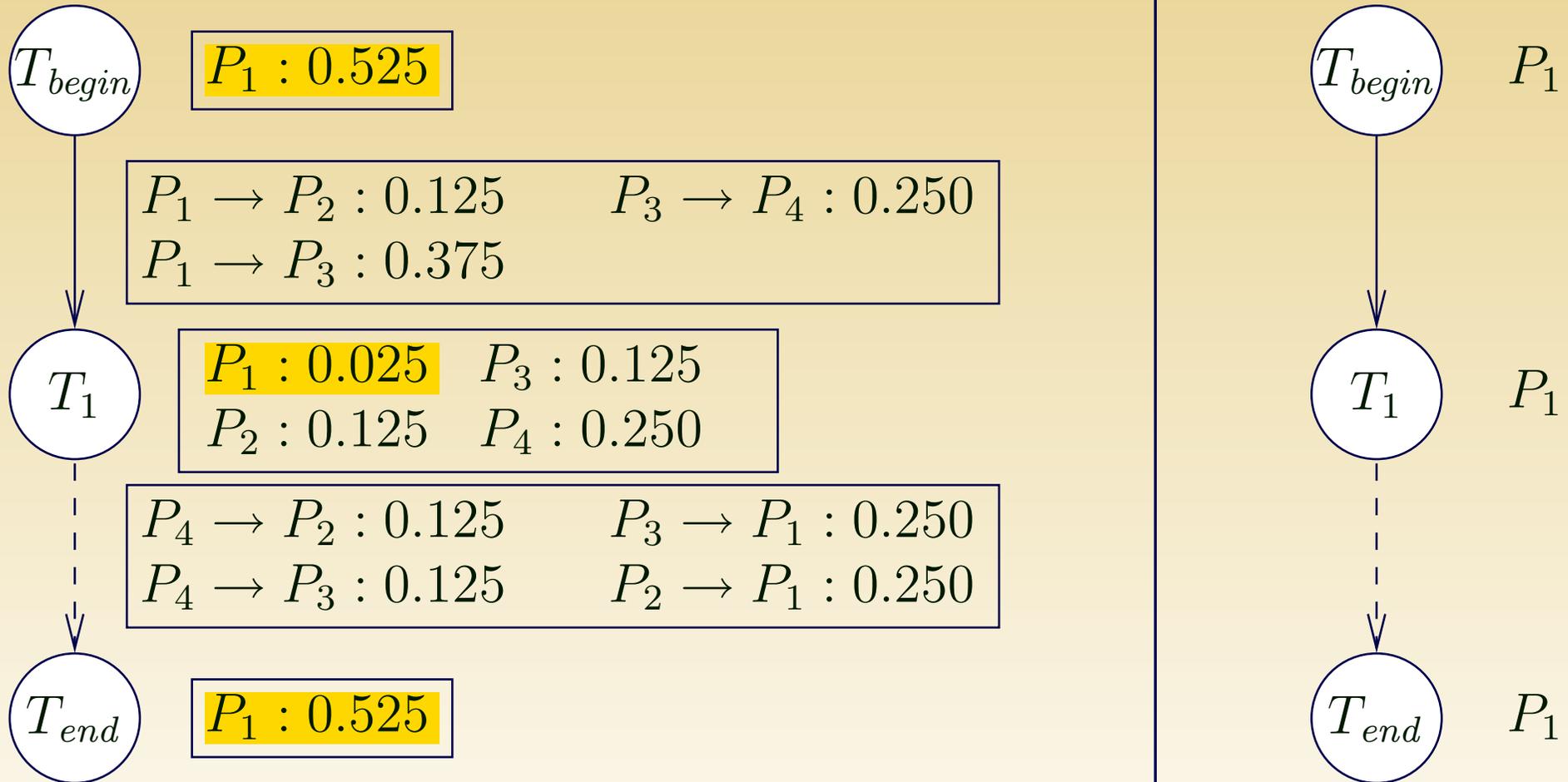
Décomposition en allocations

Régime permanent = superposition de plusieurs allocations.



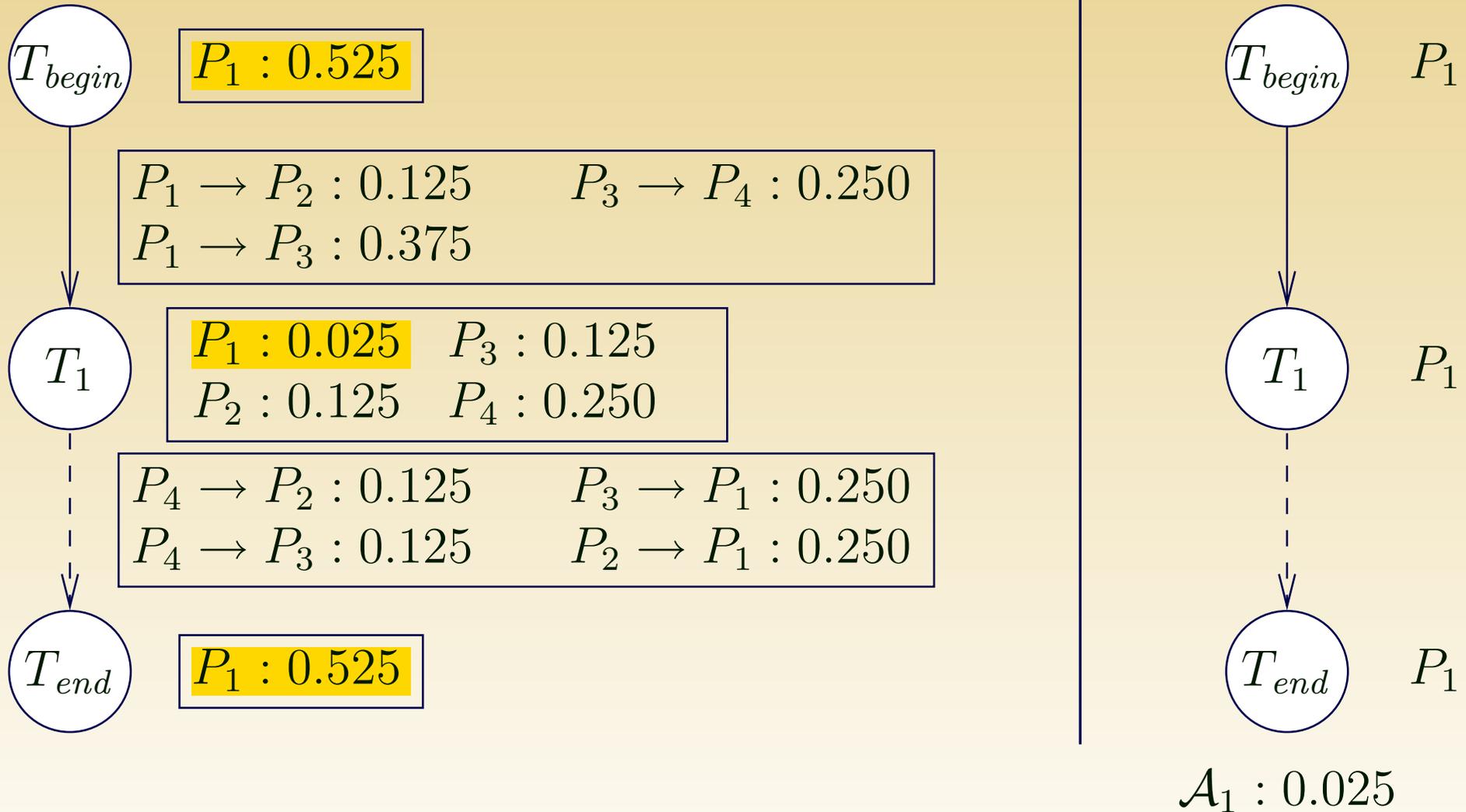
Décomposition en allocations

Régime permanent = superposition de plusieurs allocations.



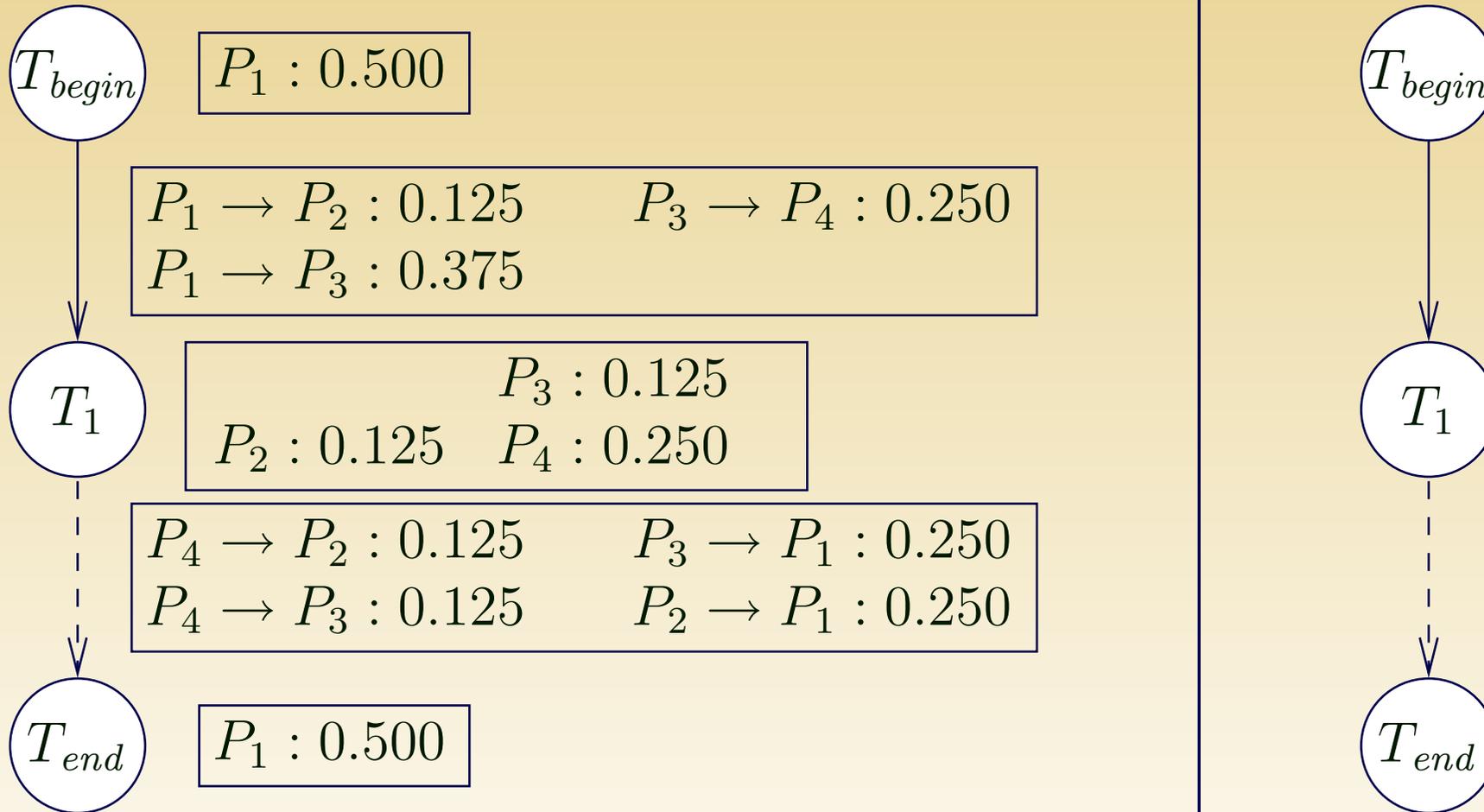
Décomposition en allocations

Régime permanent = superposition de plusieurs allocations.



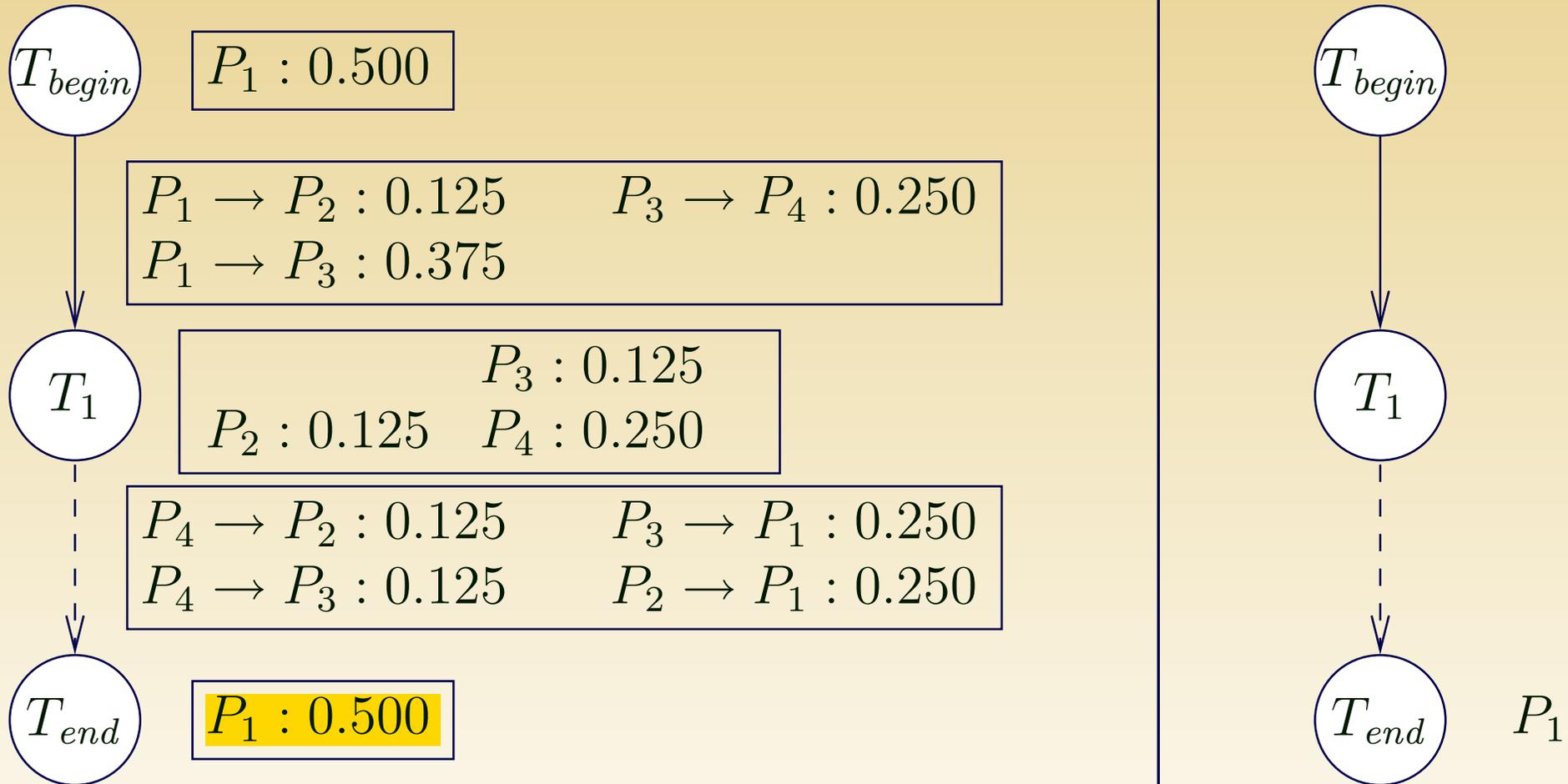
Décomposition en allocations

Régime permanent = superposition de plusieurs allocations.



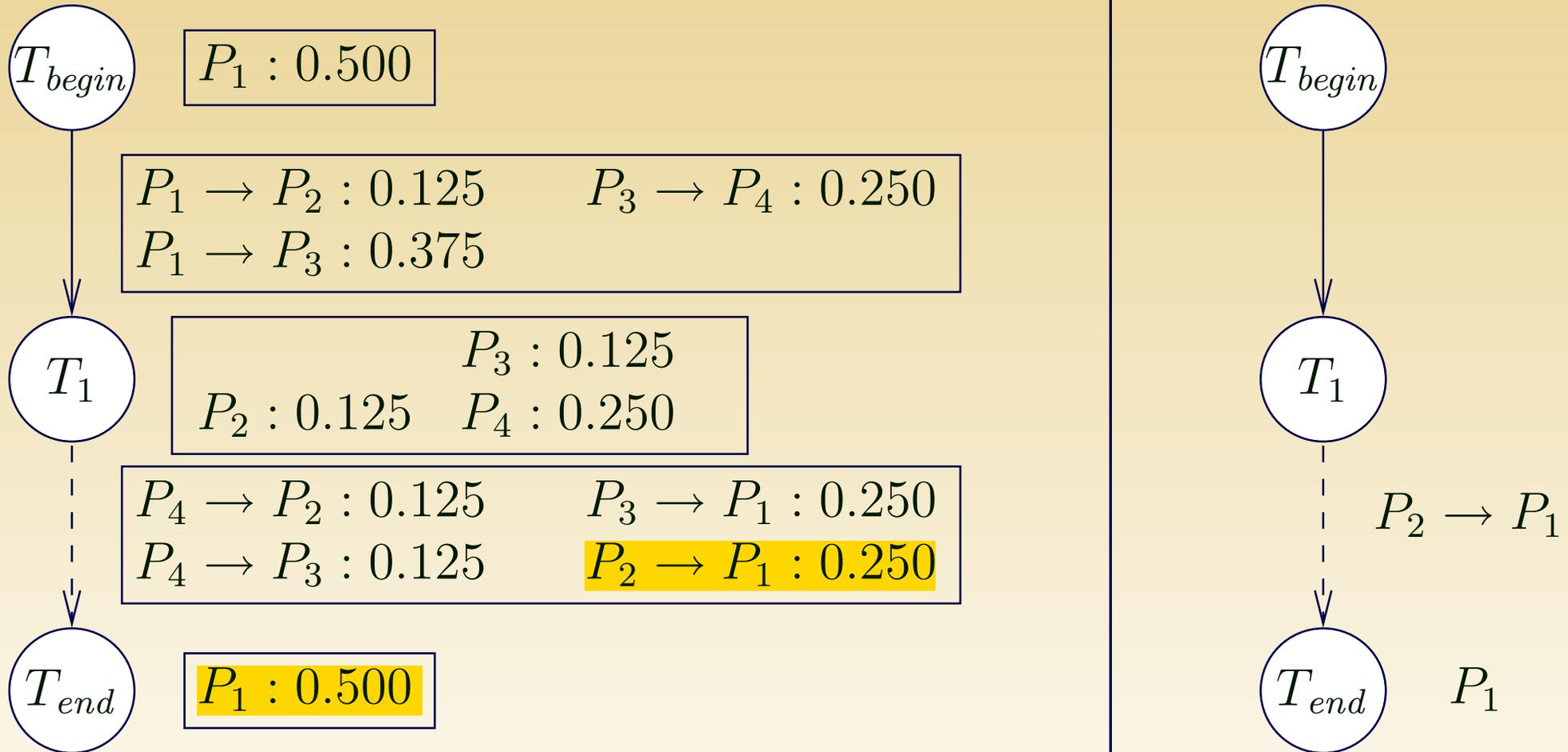
Décomposition en allocations

Régime permanent = superposition de plusieurs allocations.



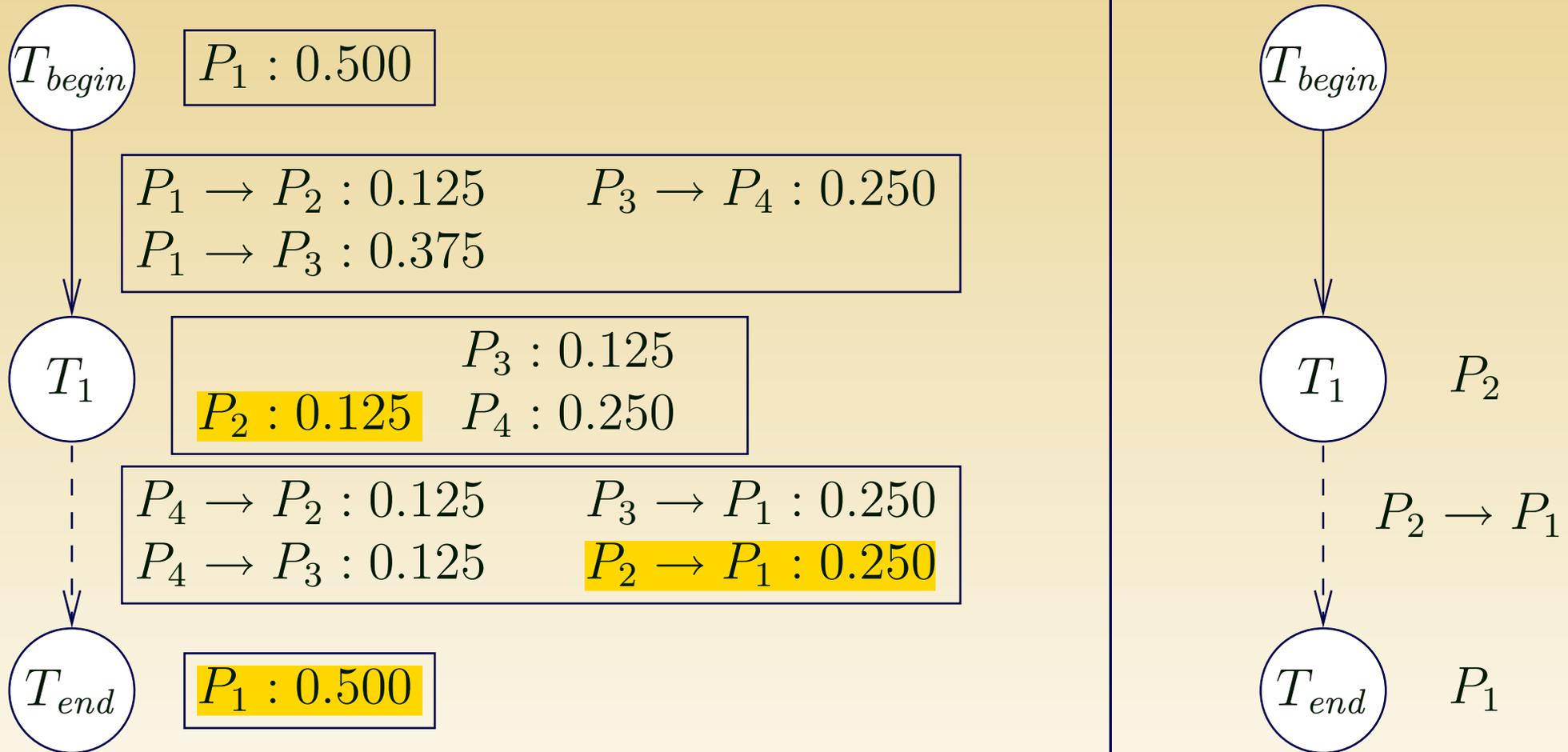
Décomposition en allocations

Régime permanent = superposition de plusieurs allocations.



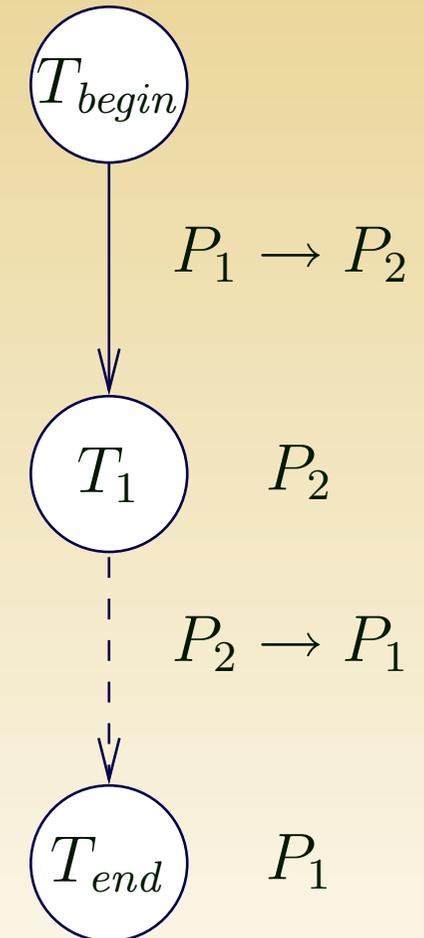
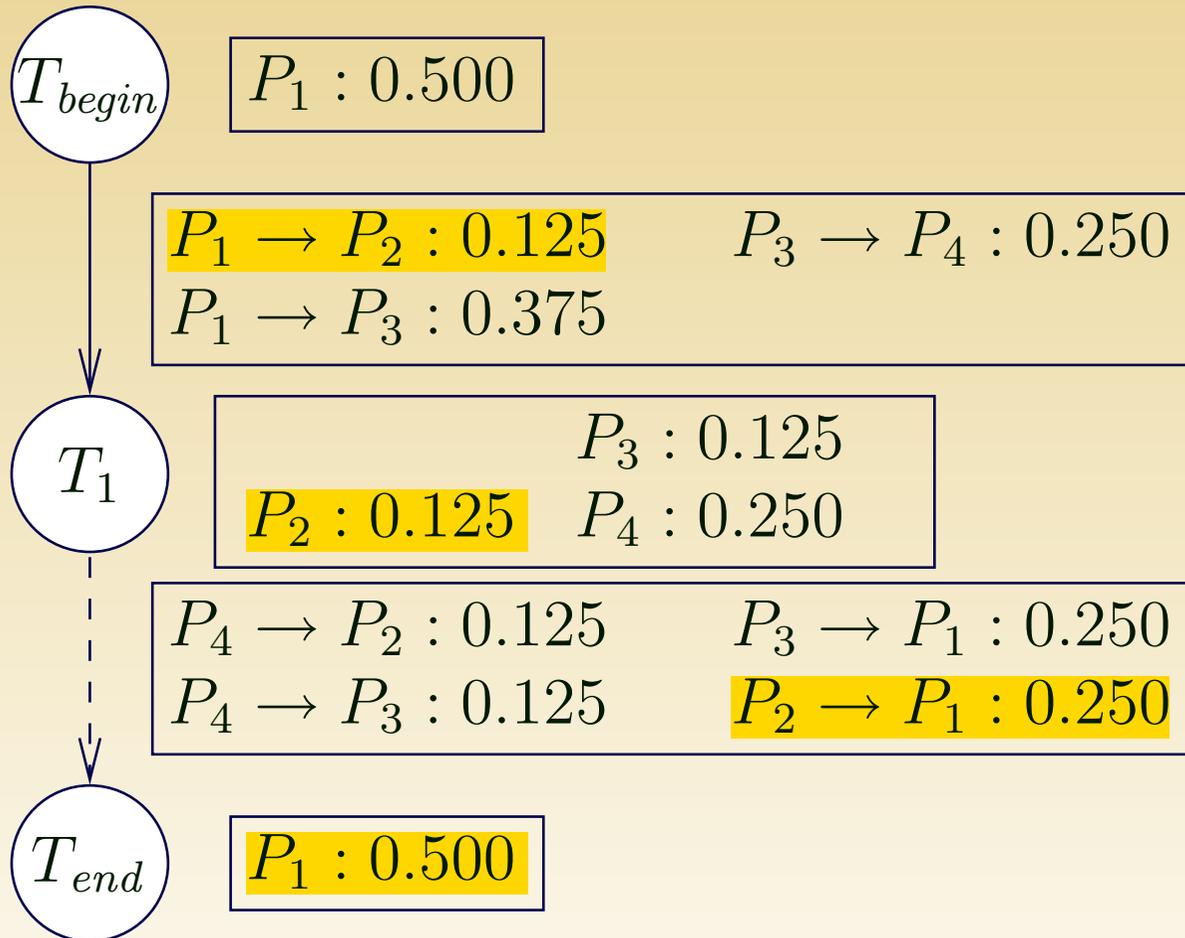
Décomposition en allocations

Régime permanent = superposition de plusieurs allocations.



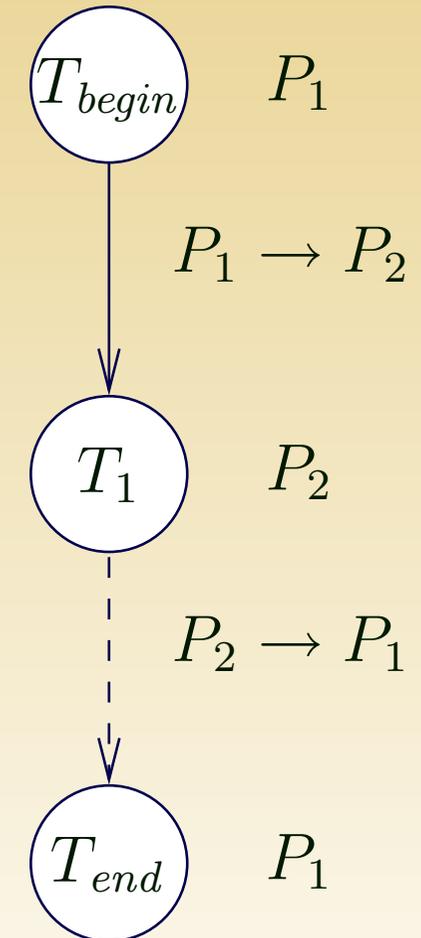
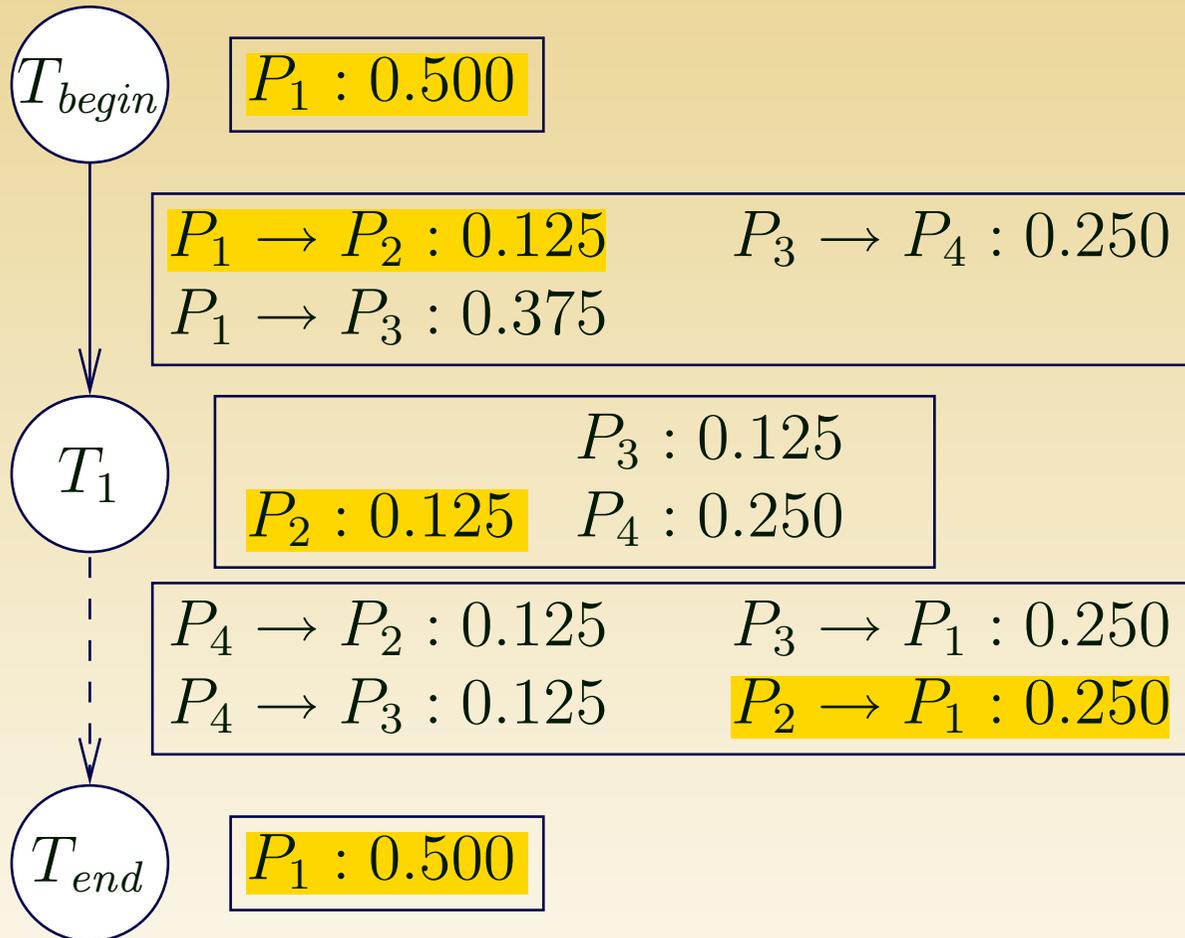
Décomposition en allocations

Régime permanent = superposition de plusieurs allocations.



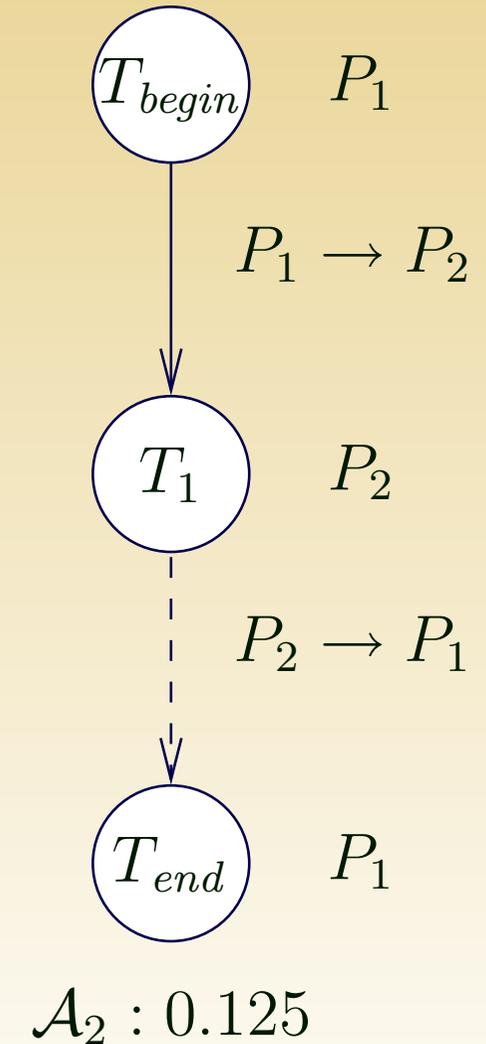
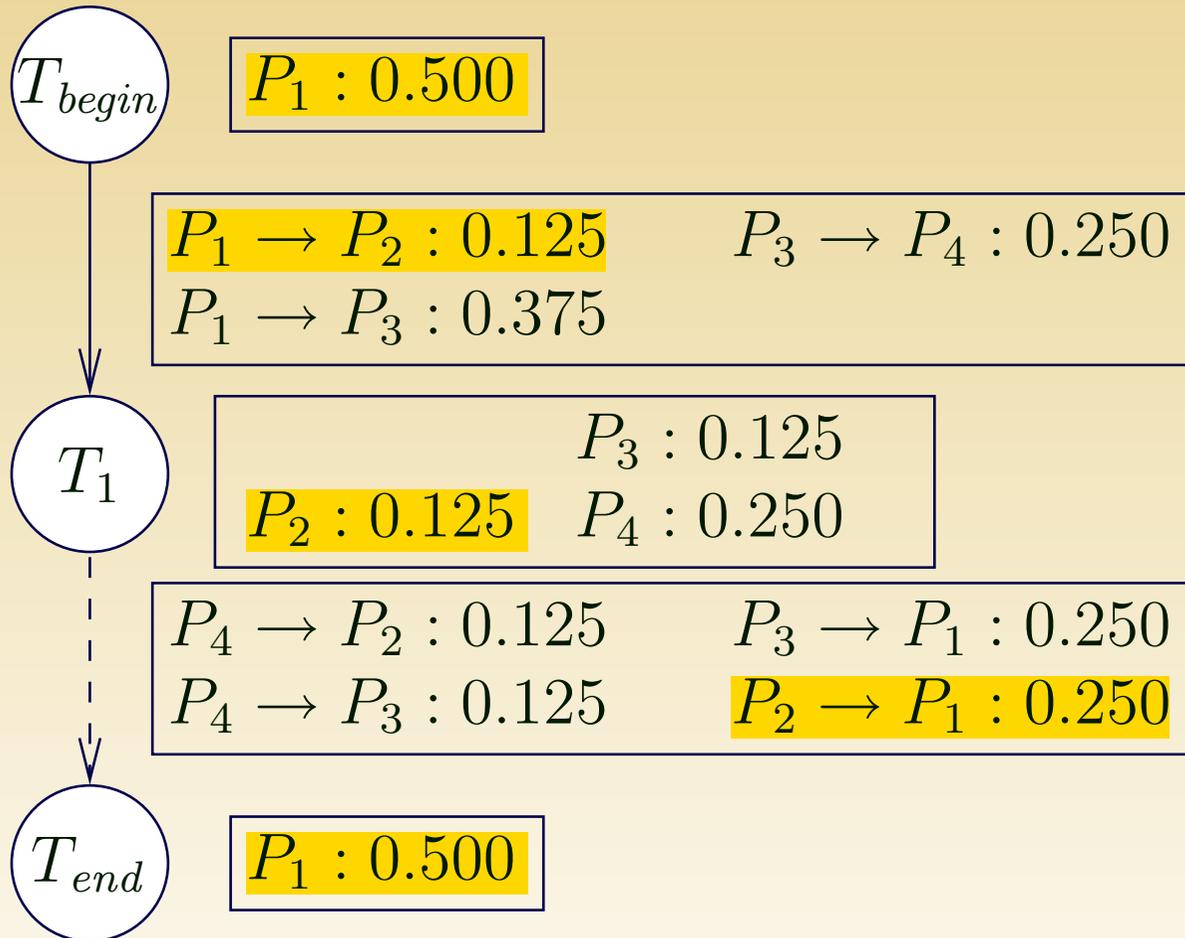
Décomposition en allocations

Régime permanent = superposition de plusieurs allocations.



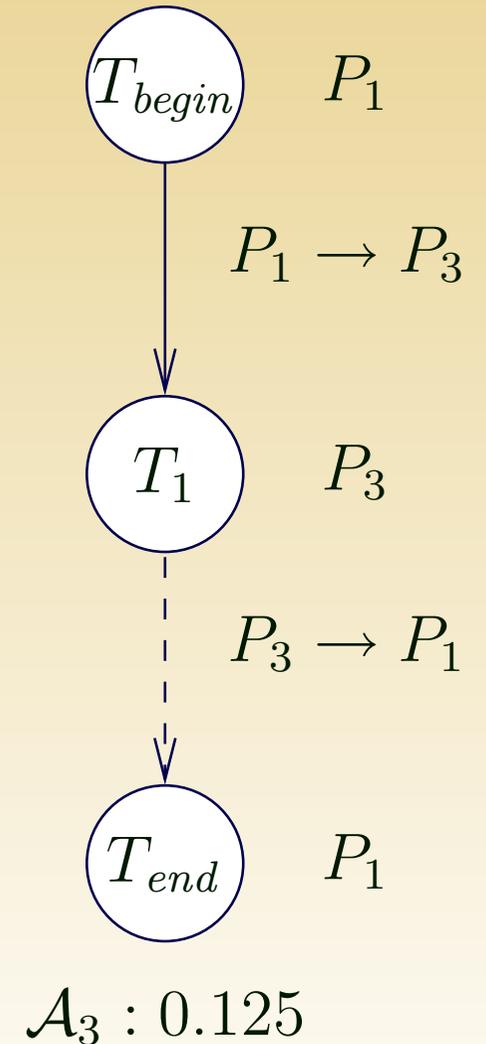
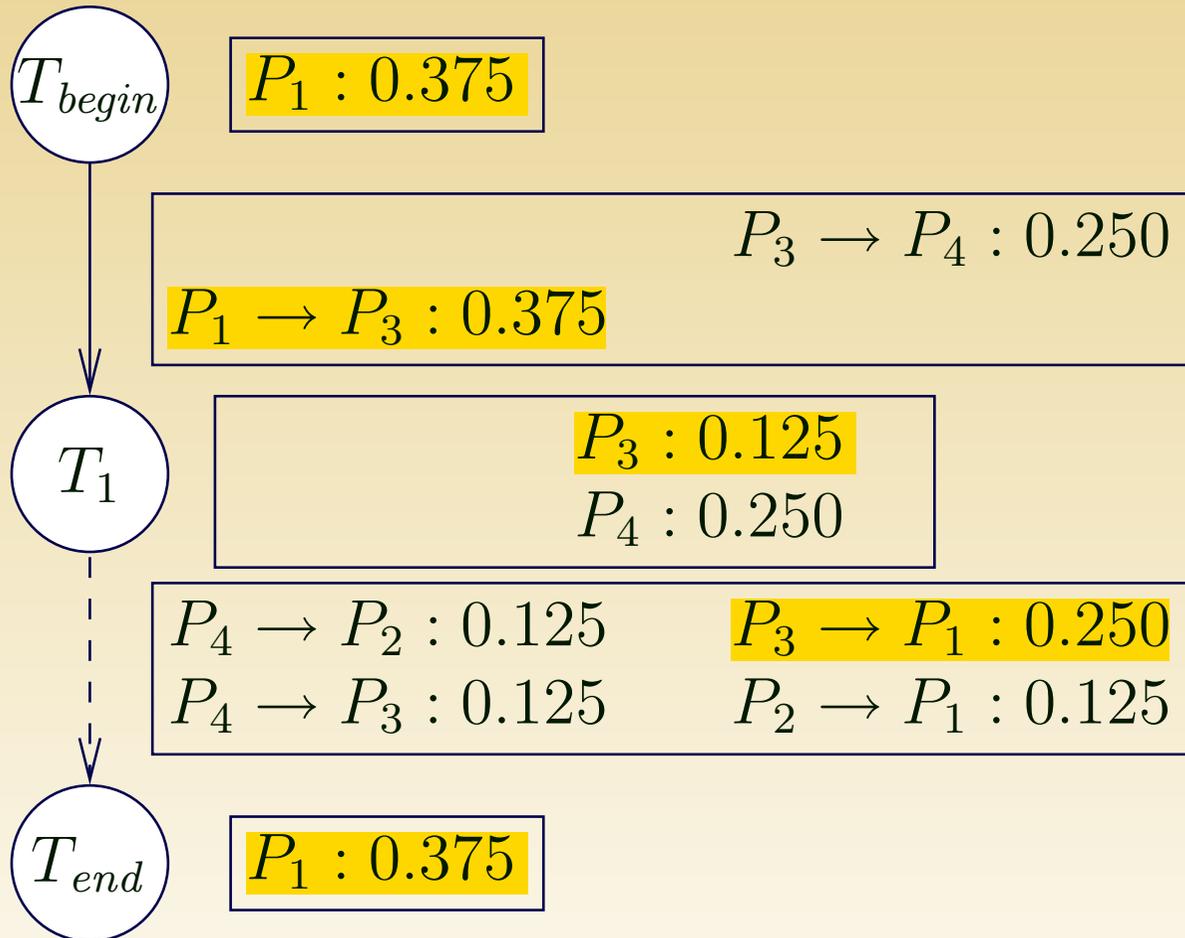
Décomposition en allocations

Régime permanent = superposition de plusieurs allocations.



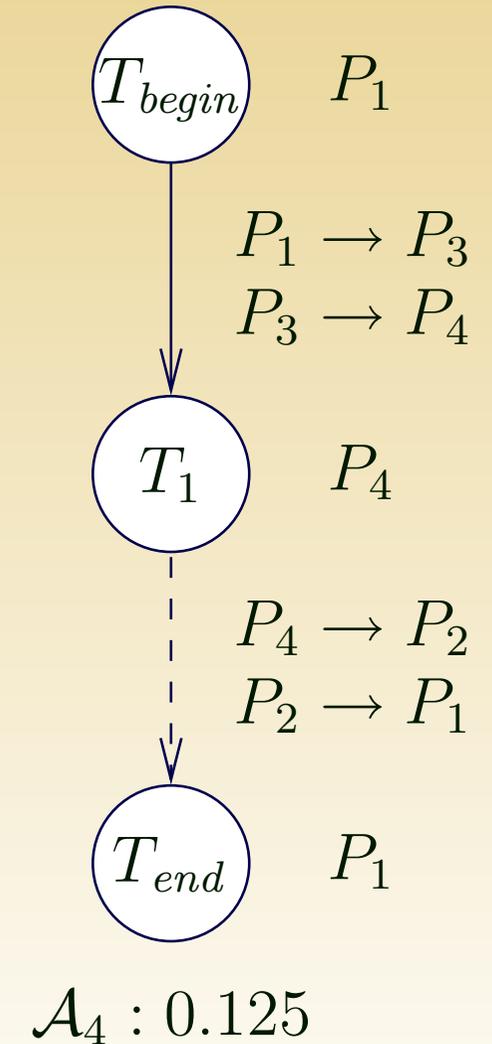
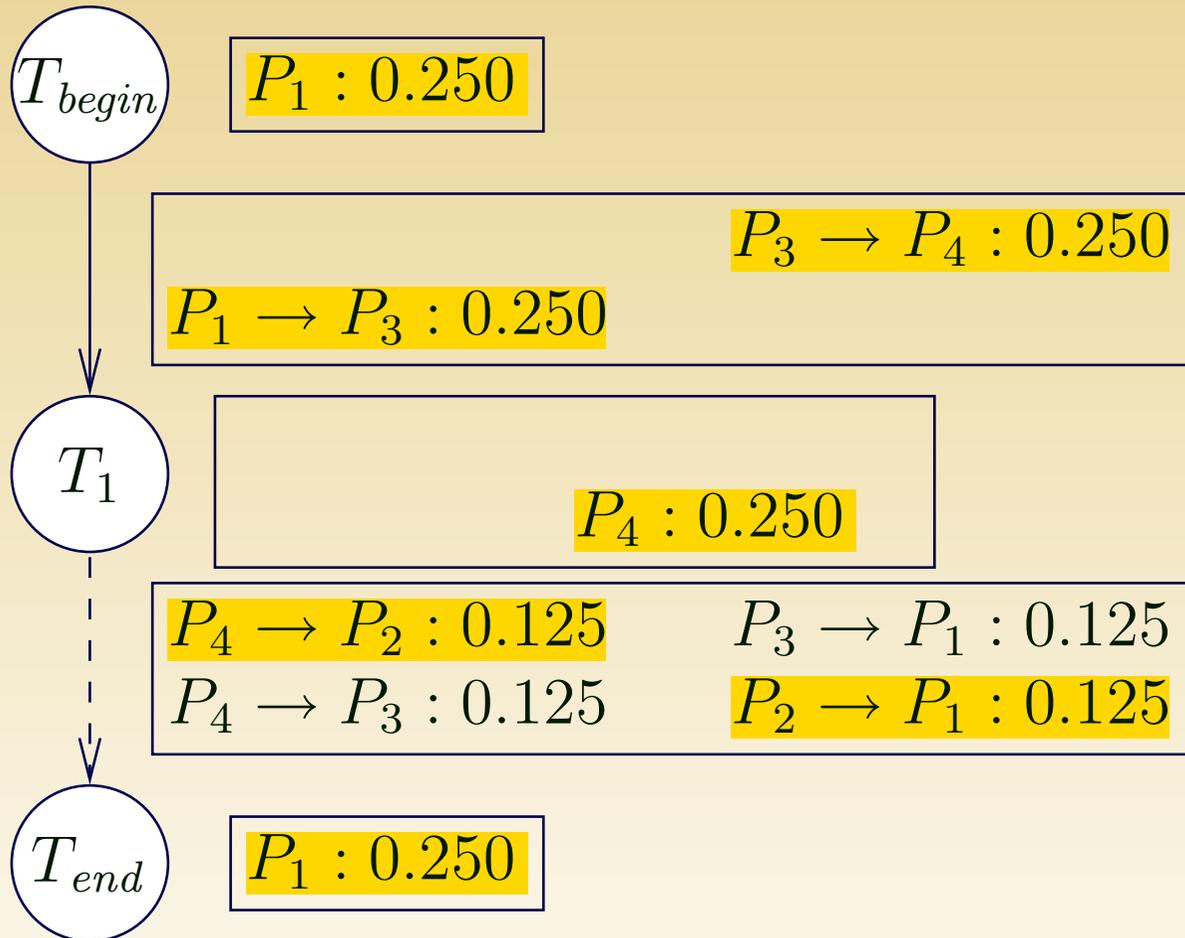
Décomposition en allocations

Régime permanent = superposition de plusieurs allocations.



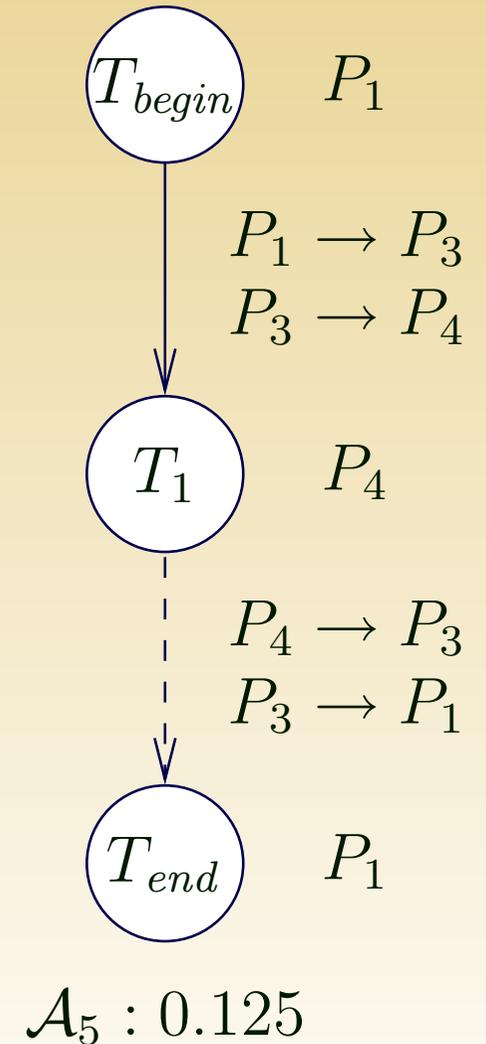
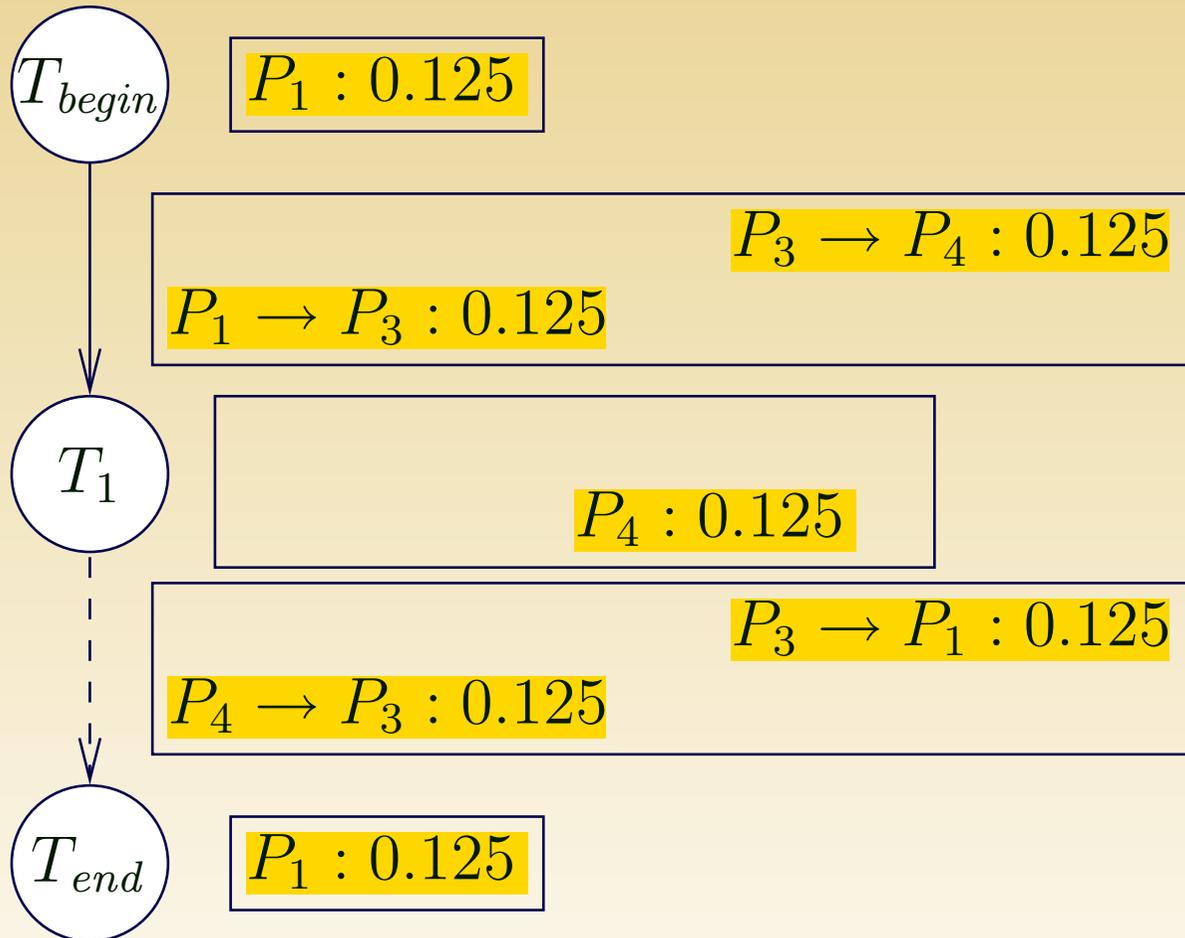
Décomposition en allocations

Régime permanent = superposition de plusieurs allocations.

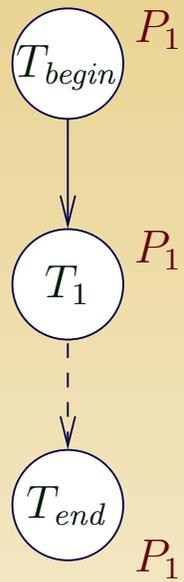


Décomposition en allocations

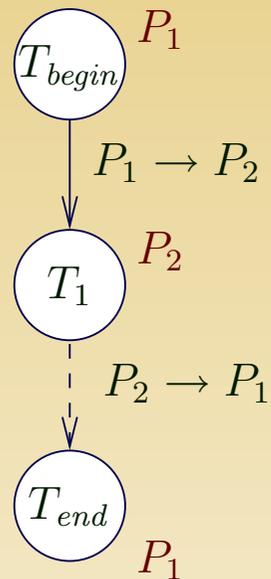
Régime permanent = superposition de plusieurs allocations.



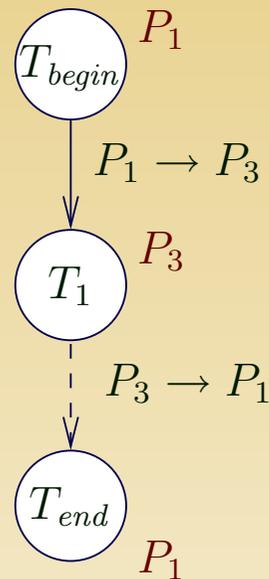
Décomposition en allocations



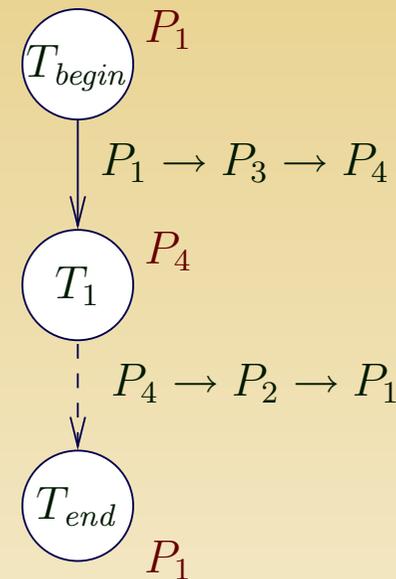
\mathcal{A}_1
0,025



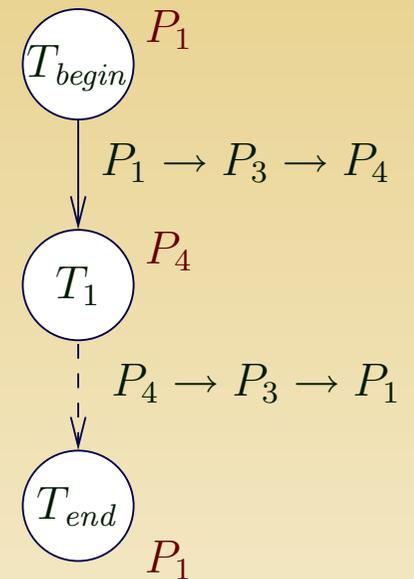
\mathcal{A}_2
0,125



\mathcal{A}_3
0,125

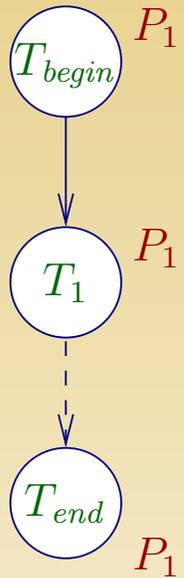


\mathcal{A}_4
0,125

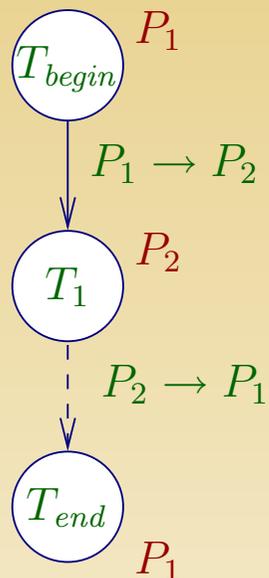


\mathcal{A}_5
0,125

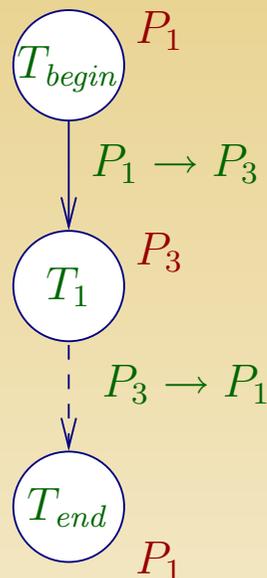
Décomposition en allocations



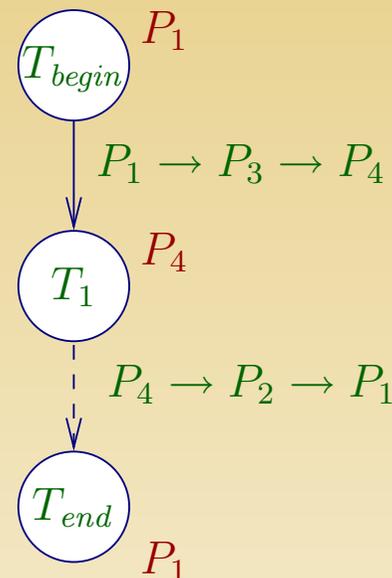
\mathcal{A}_1
0,025



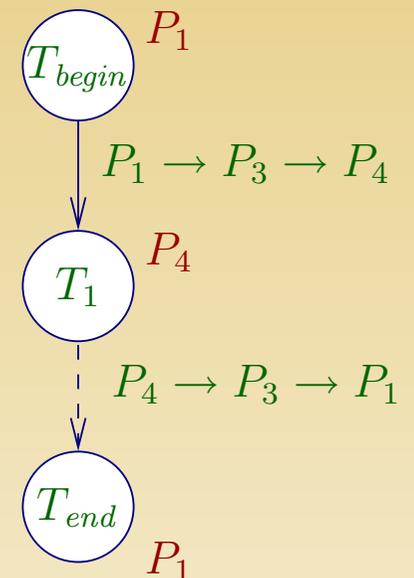
\mathcal{A}_2
0,125



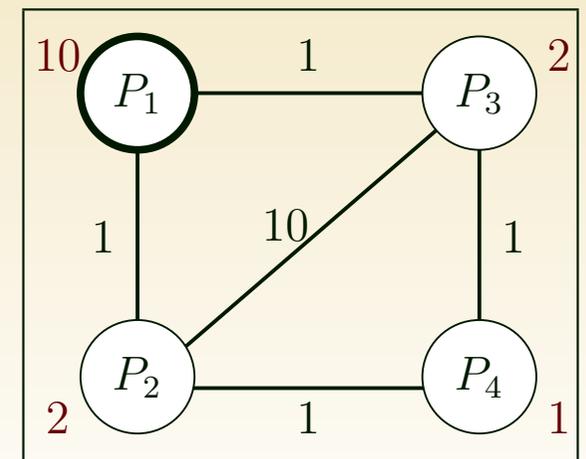
\mathcal{A}_3
0,125



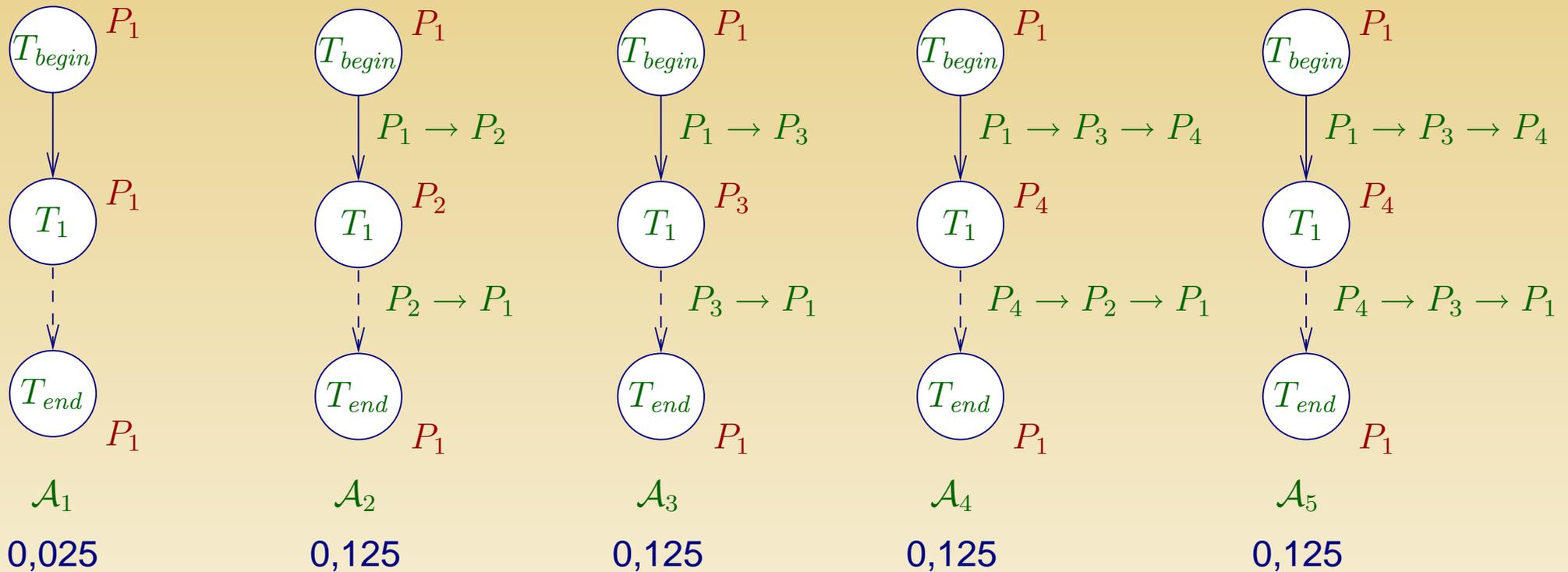
\mathcal{A}_4
0,125



\mathcal{A}_5
0,125

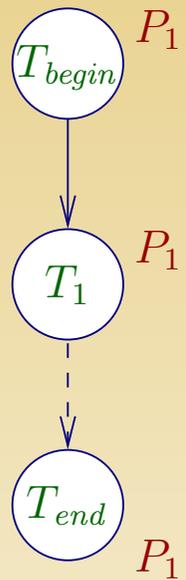


Décomposition en allocations

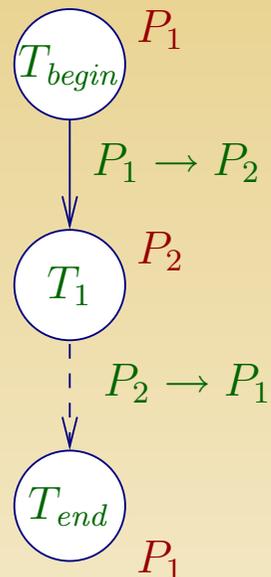


Cette décomposition est toujours possible.

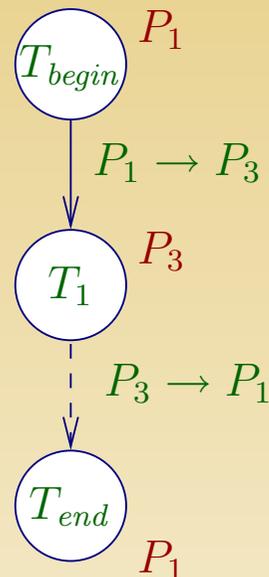
Décomposition en allocations



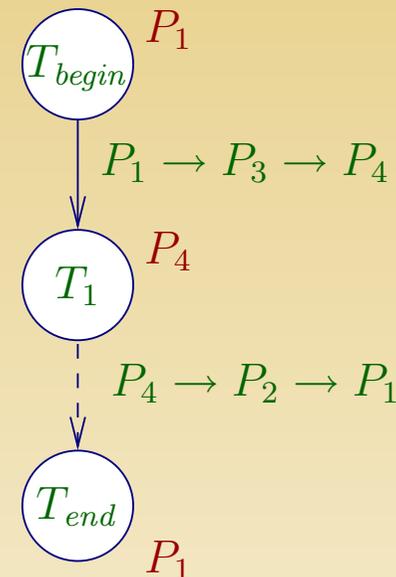
\mathcal{A}_1
0,025



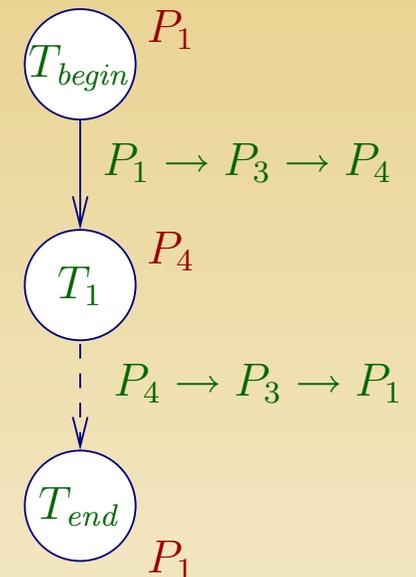
\mathcal{A}_2
0,125



\mathcal{A}_3
0,125



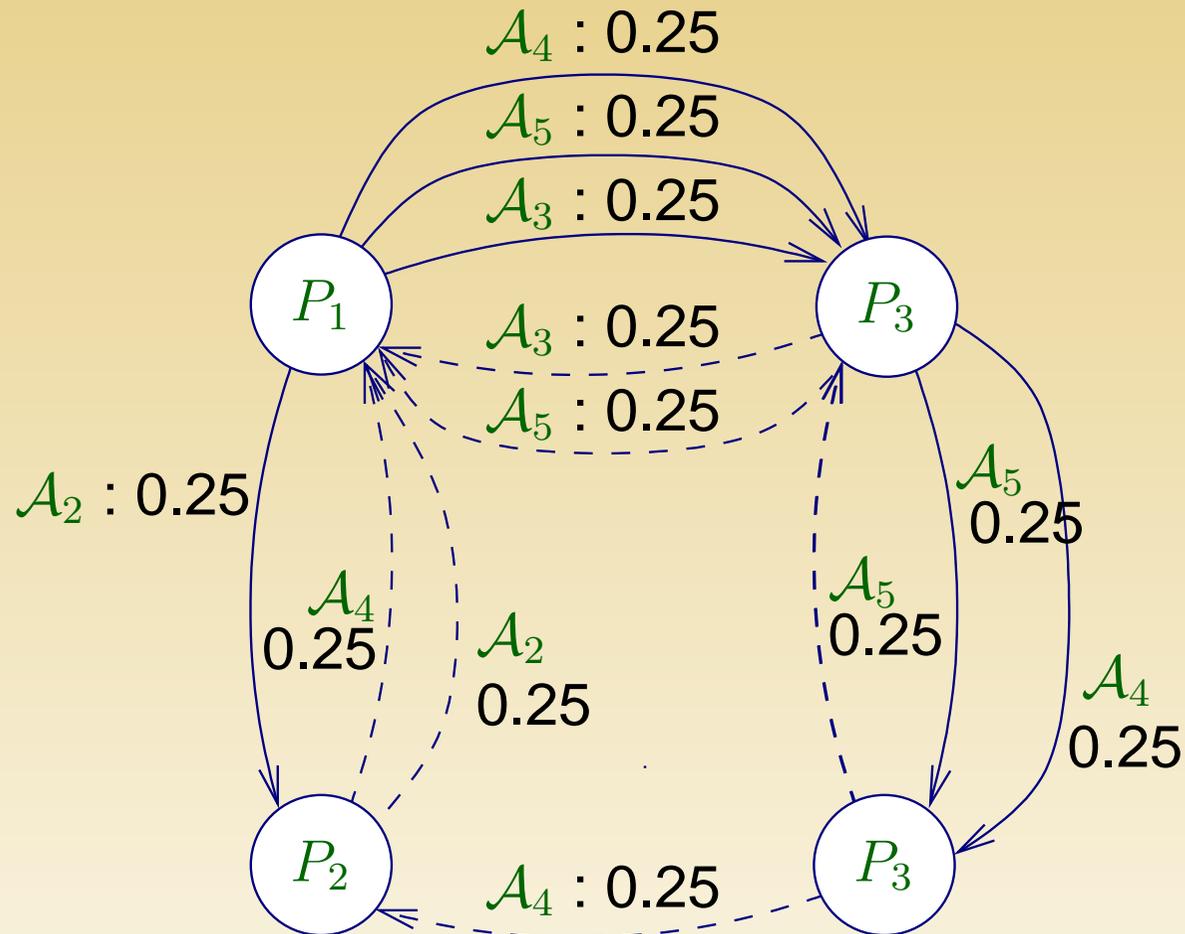
\mathcal{A}_4
0,125



\mathcal{A}_5
0,125

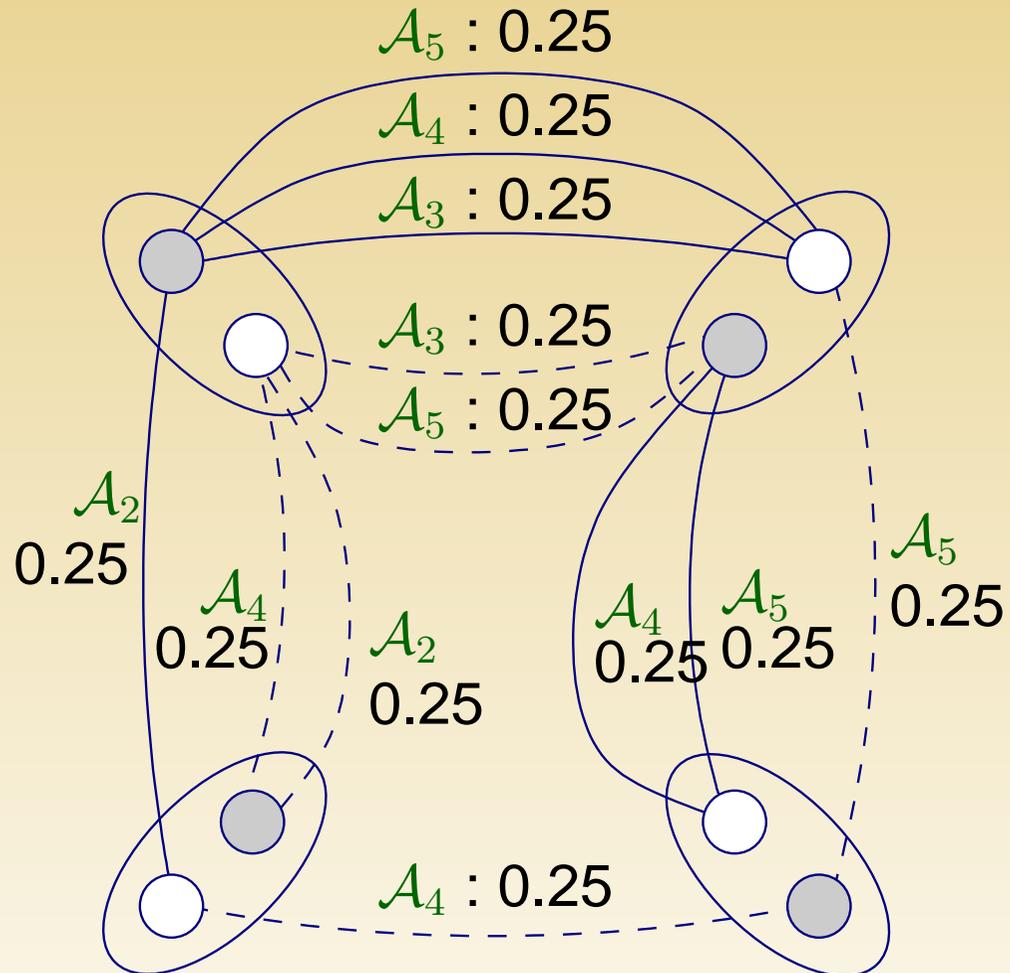
Comment organiser ces allocations ?

Graphe de communications

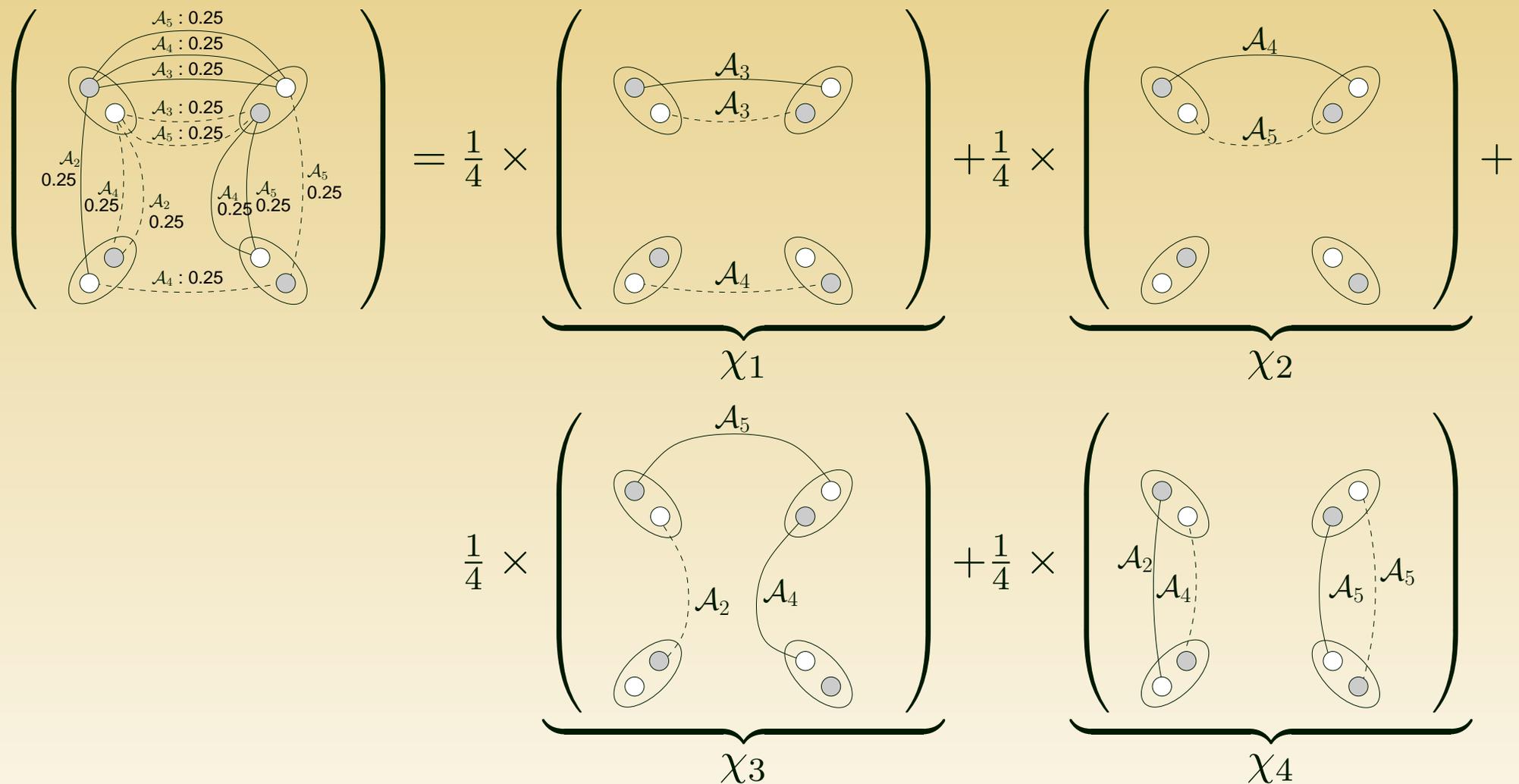


Fraction de temps passée au transfert d'un $e_{k,l}$ de P_i vers P_j pour une allocation donnée.

Contraintes 1-port = couplage

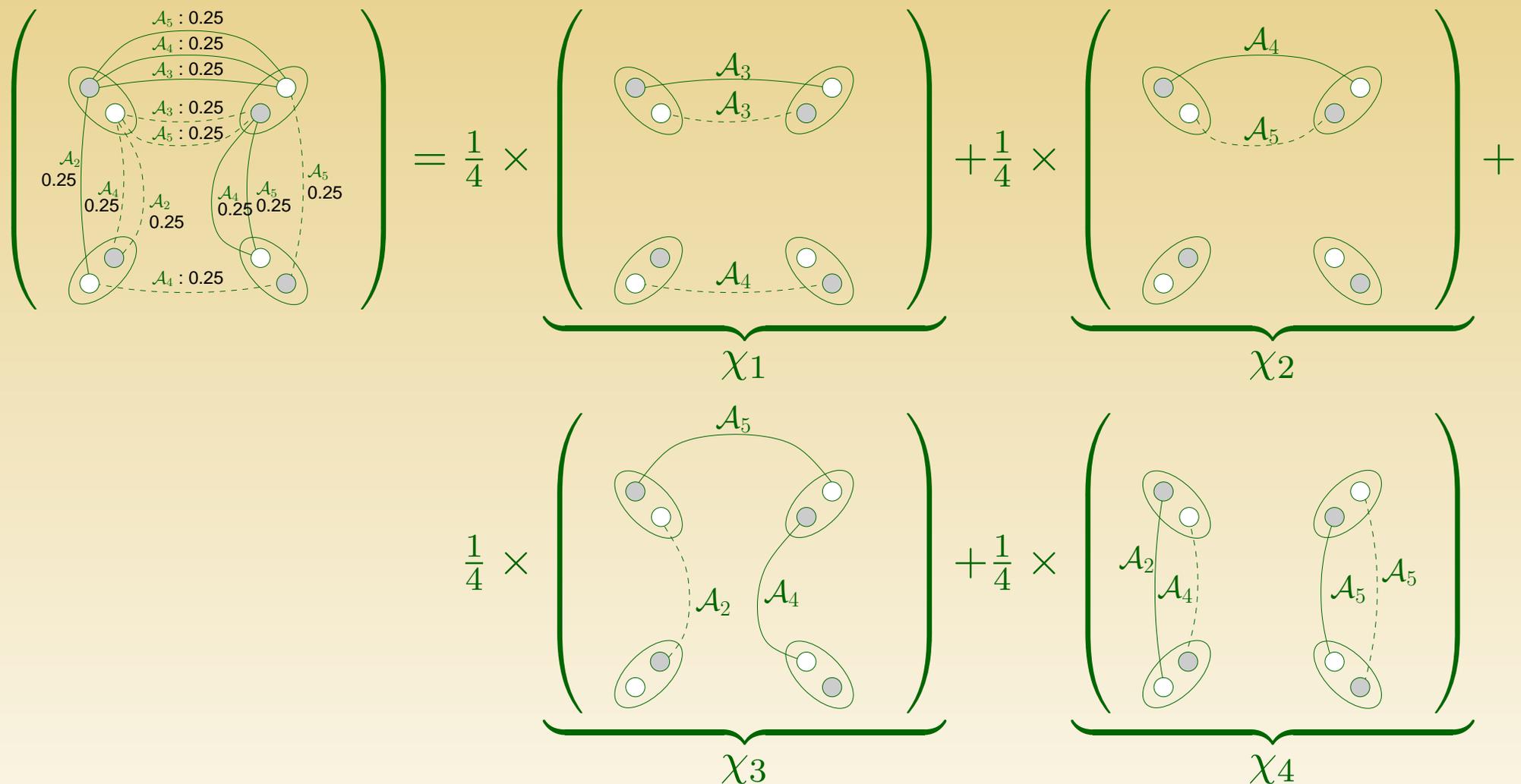


Décomposition en couplages



Cette décomposition est toujours possible

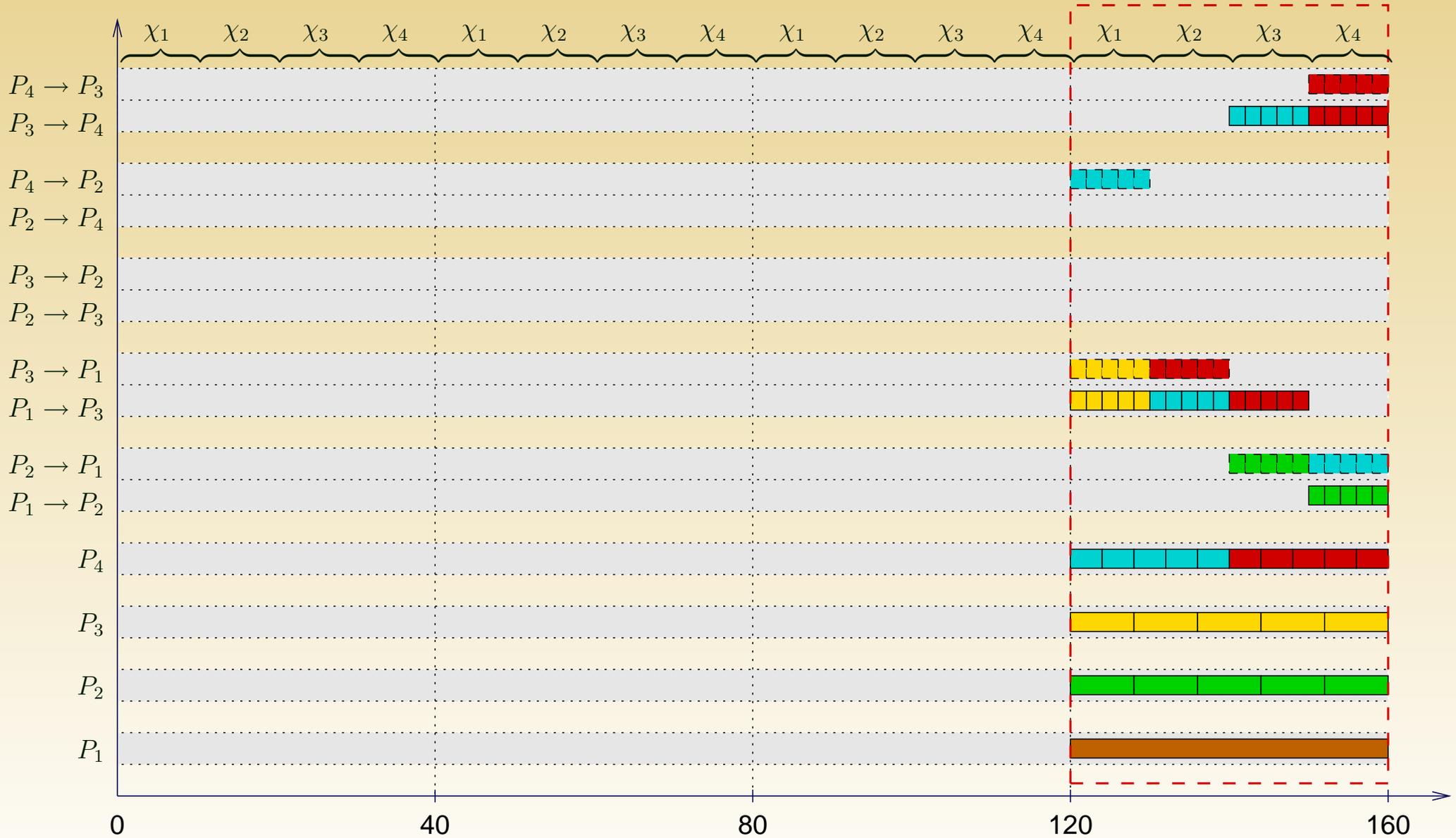
Décomposition en couplages



Cette décomposition est toujours possible

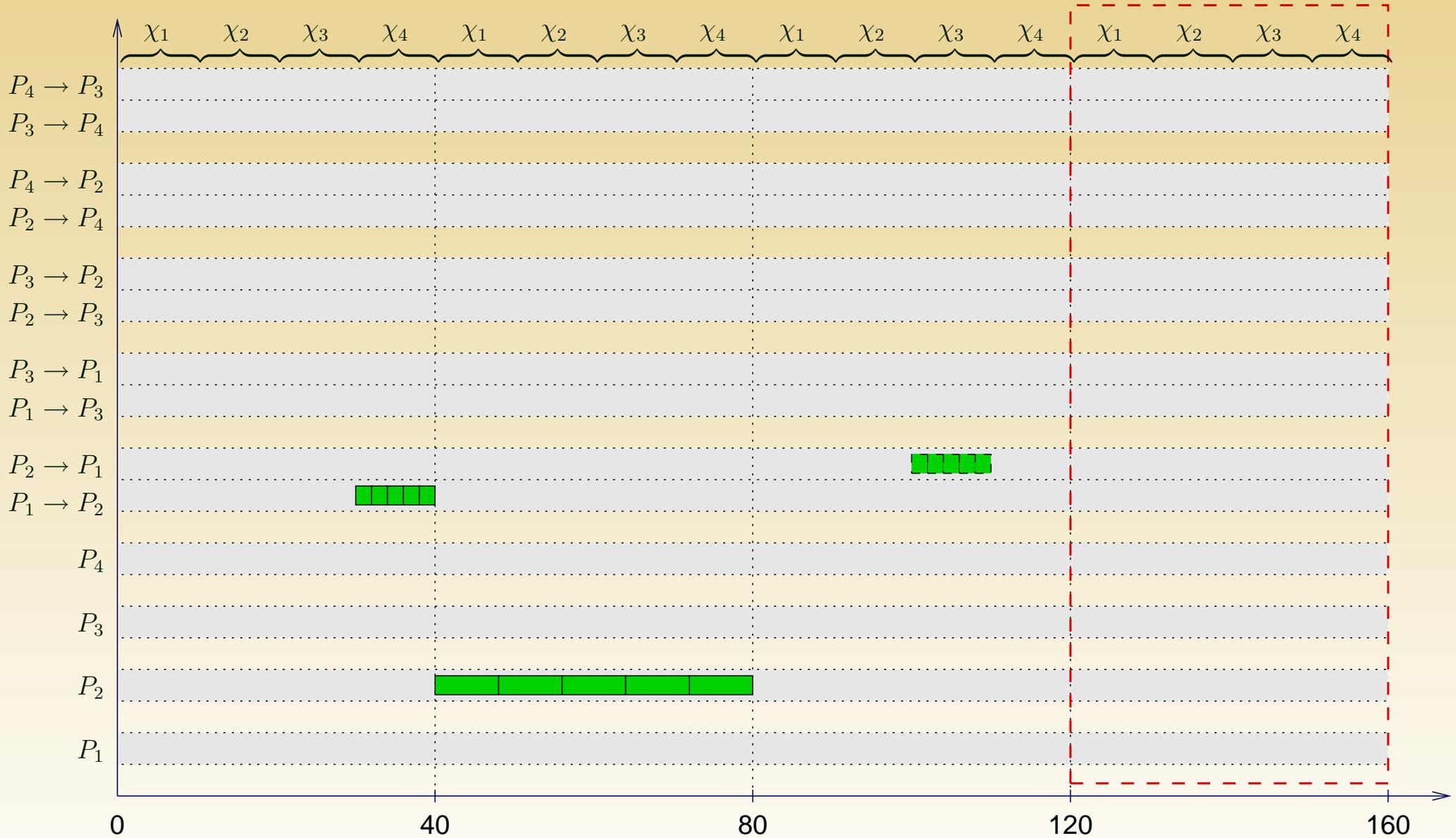
Ordonnancement cyclique de débit optimal

\mathcal{A}_1
 \mathcal{A}_2
 \mathcal{A}_3
 \mathcal{A}_4
 \mathcal{A}_5



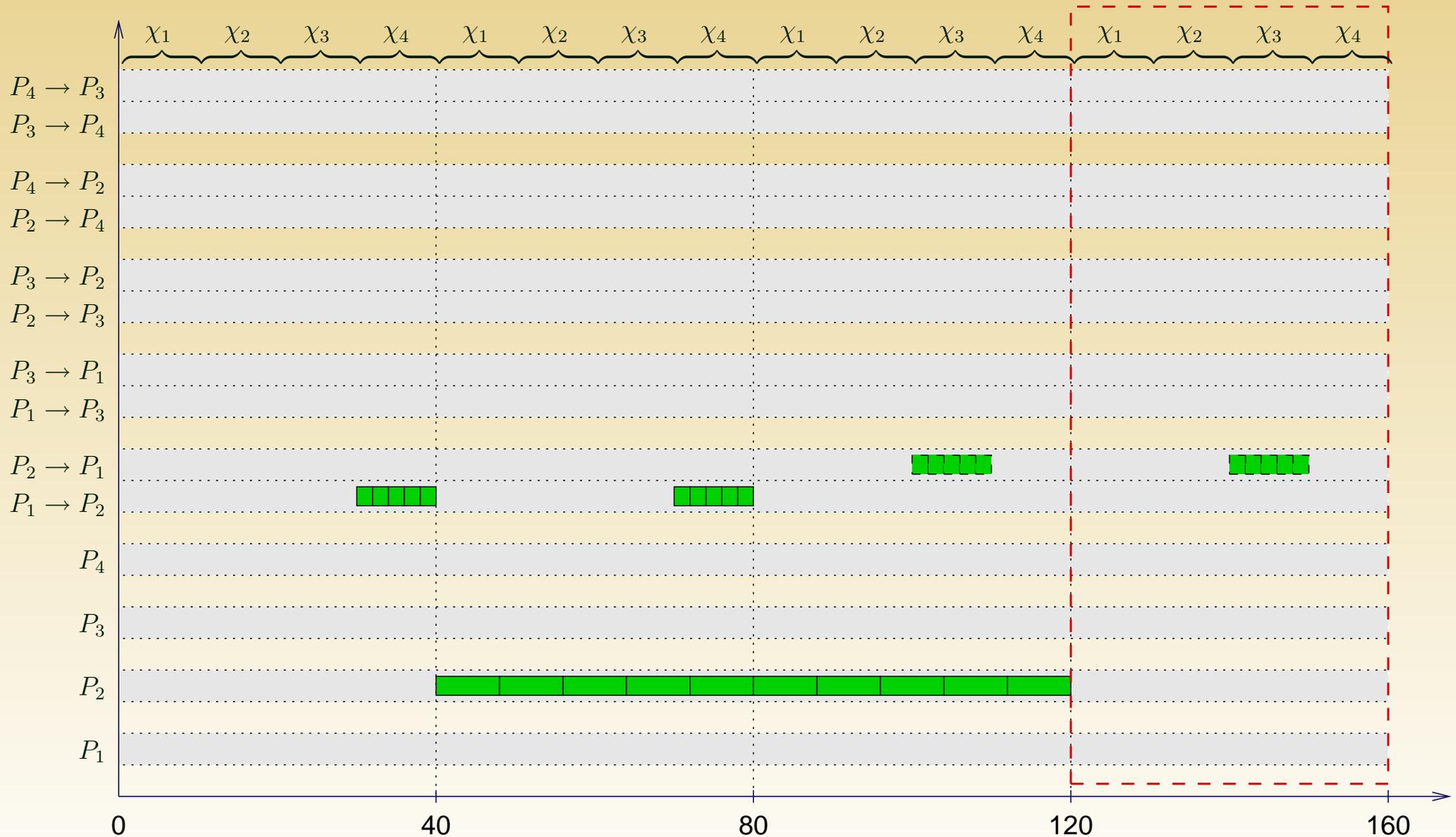
Ordonnancement cyclique de débit optimal

\mathcal{A}_1
 \mathcal{A}_2
 \mathcal{A}_3
 \mathcal{A}_4
 \mathcal{A}_5



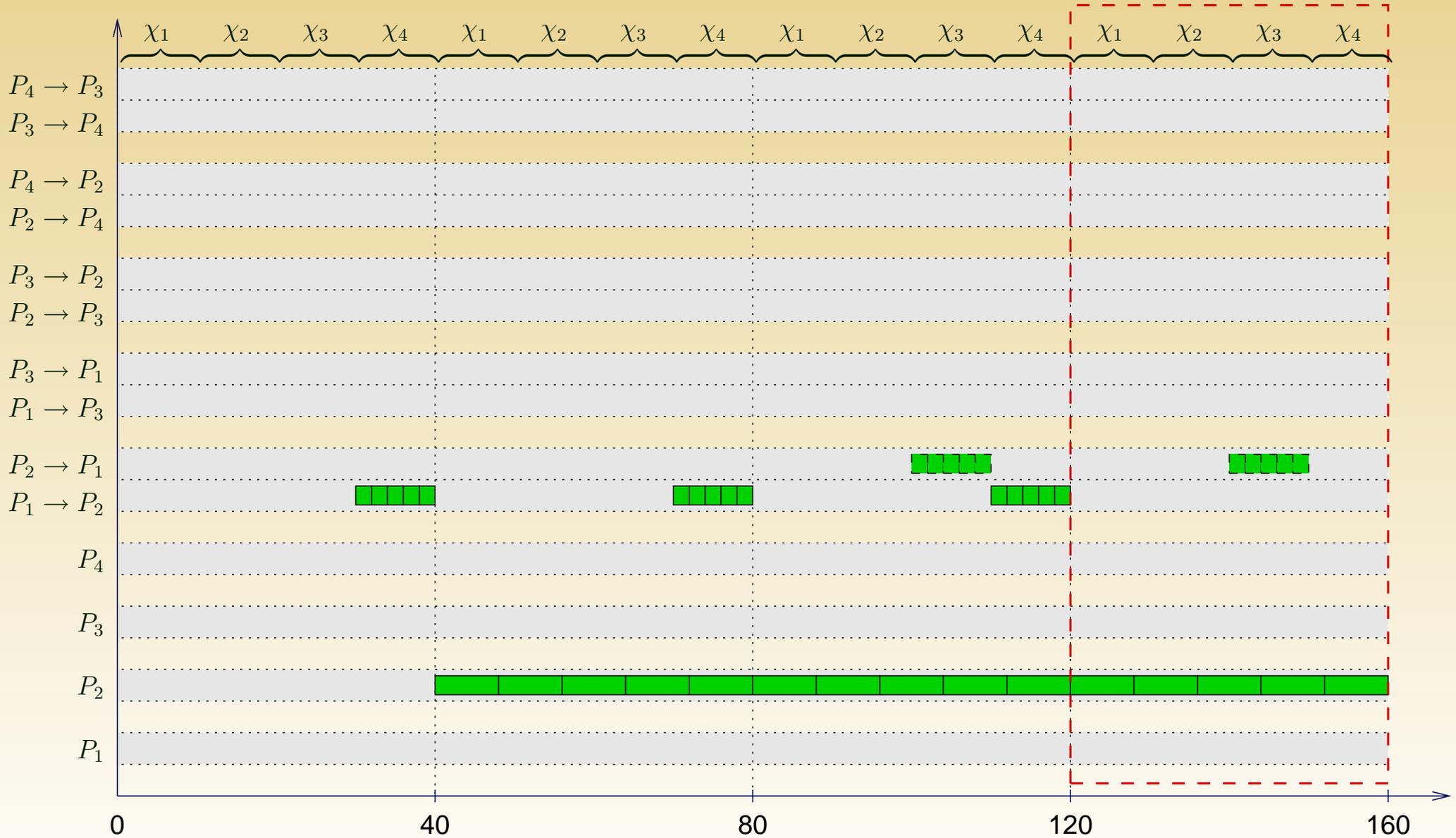
Ordonnancement cyclique de débit optimal

\mathcal{A}_1
 \mathcal{A}_2
 \mathcal{A}_3
 \mathcal{A}_4
 \mathcal{A}_5



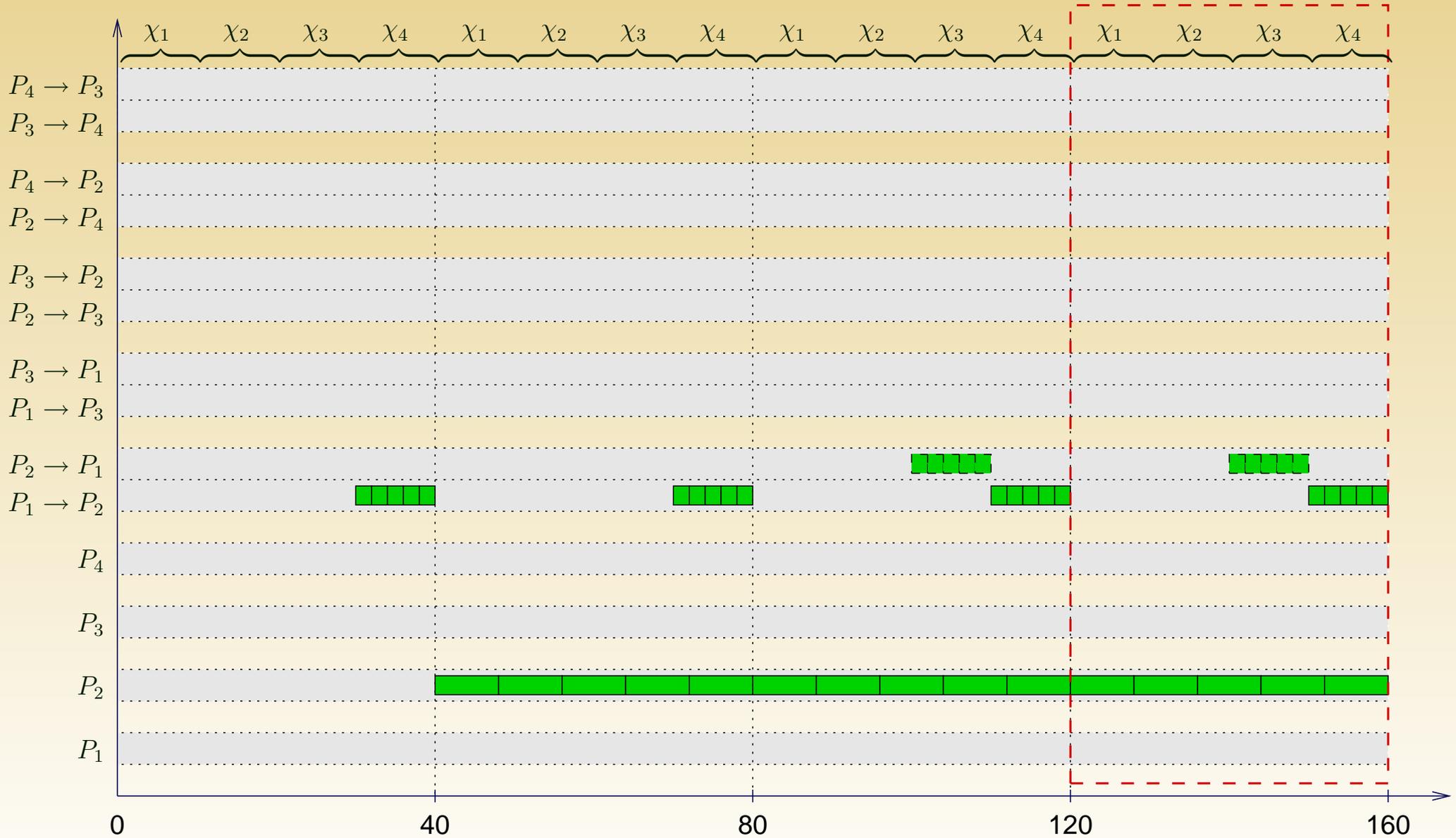
Ordonnancement cyclique de débit optimal

\mathcal{A}_1
 \mathcal{A}_2
 \mathcal{A}_3
 \mathcal{A}_4
 \mathcal{A}_5



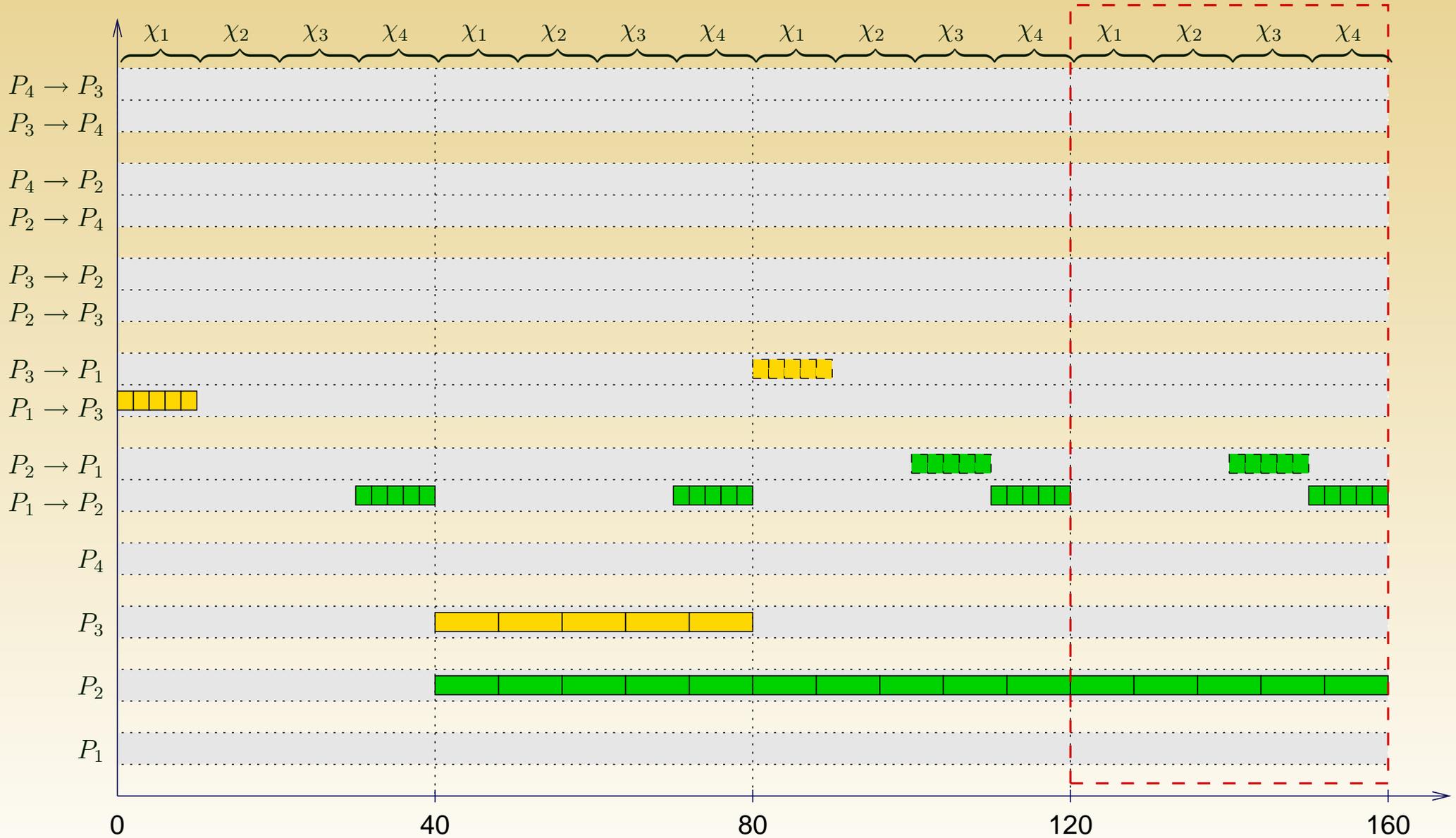
Ordonnancement cyclique de débit optimal

\mathcal{A}_1
 \mathcal{A}_2
 \mathcal{A}_3
 \mathcal{A}_4
 \mathcal{A}_5



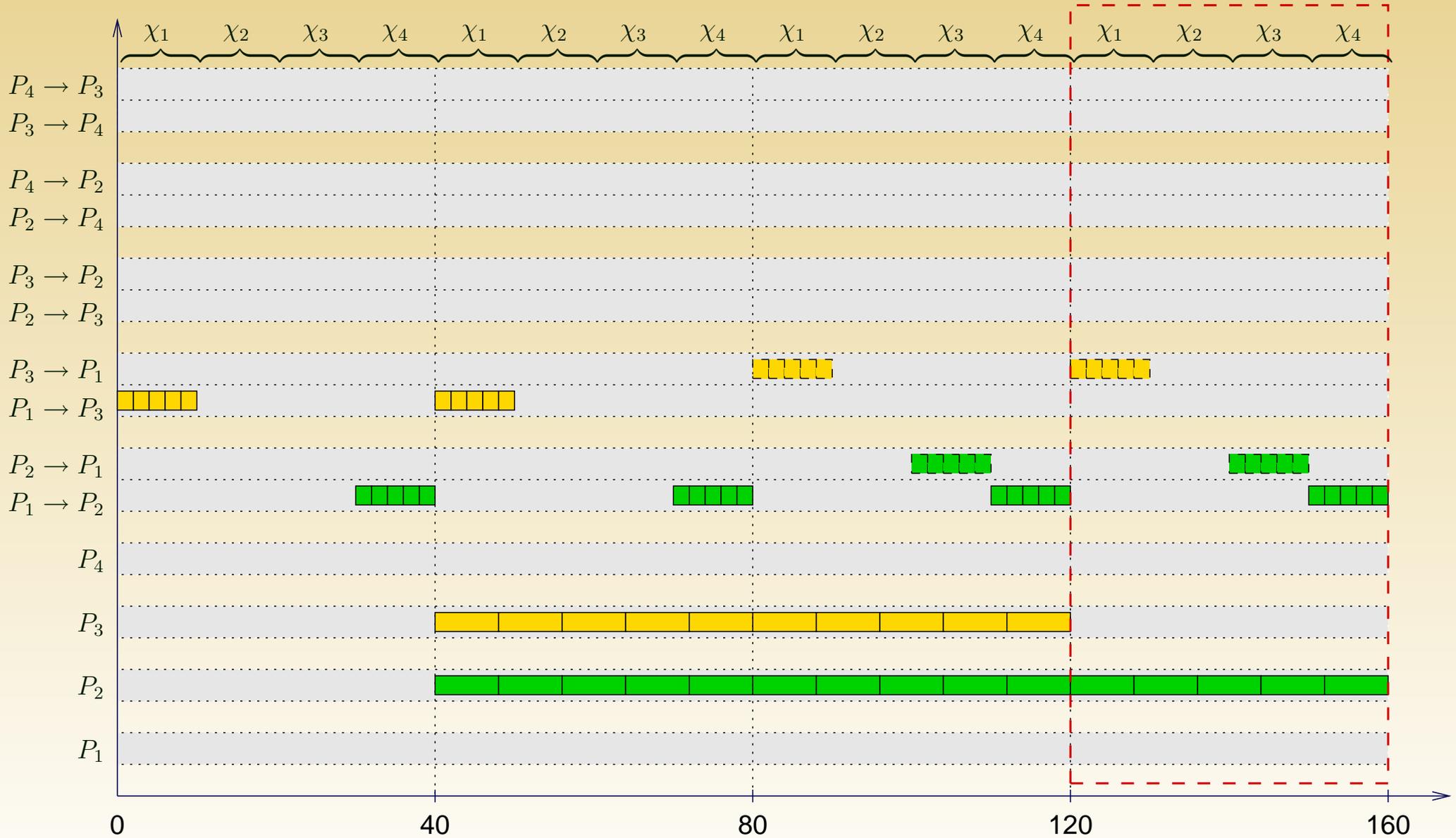
Ordonnancement cyclique de débit optimal

\mathcal{A}_1
 \mathcal{A}_2
 \mathcal{A}_3
 \mathcal{A}_4
 \mathcal{A}_5



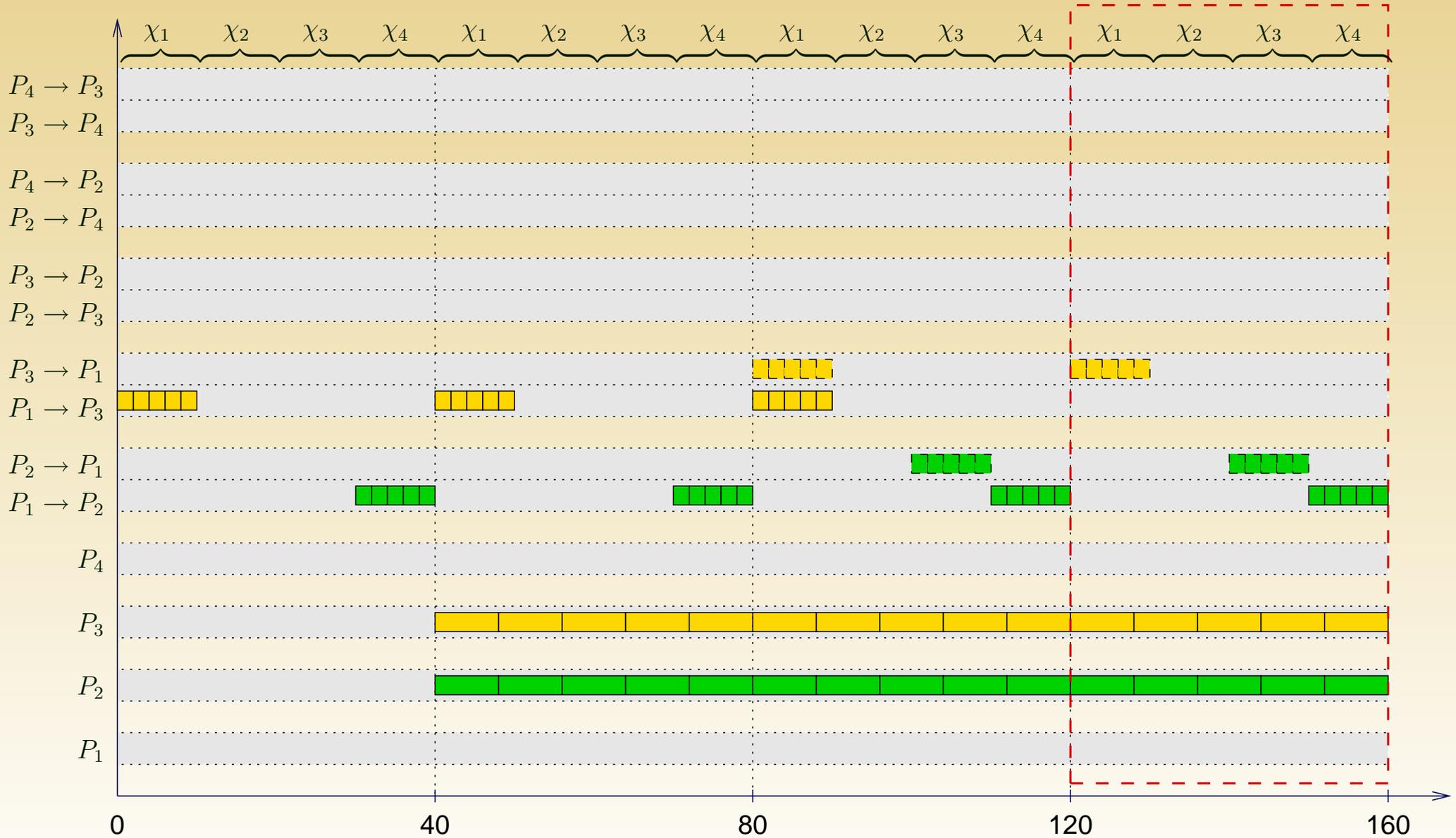
Ordonnancement cyclique de débit optimal

\mathcal{A}_1
 \mathcal{A}_2
 \mathcal{A}_3
 \mathcal{A}_4
 \mathcal{A}_5



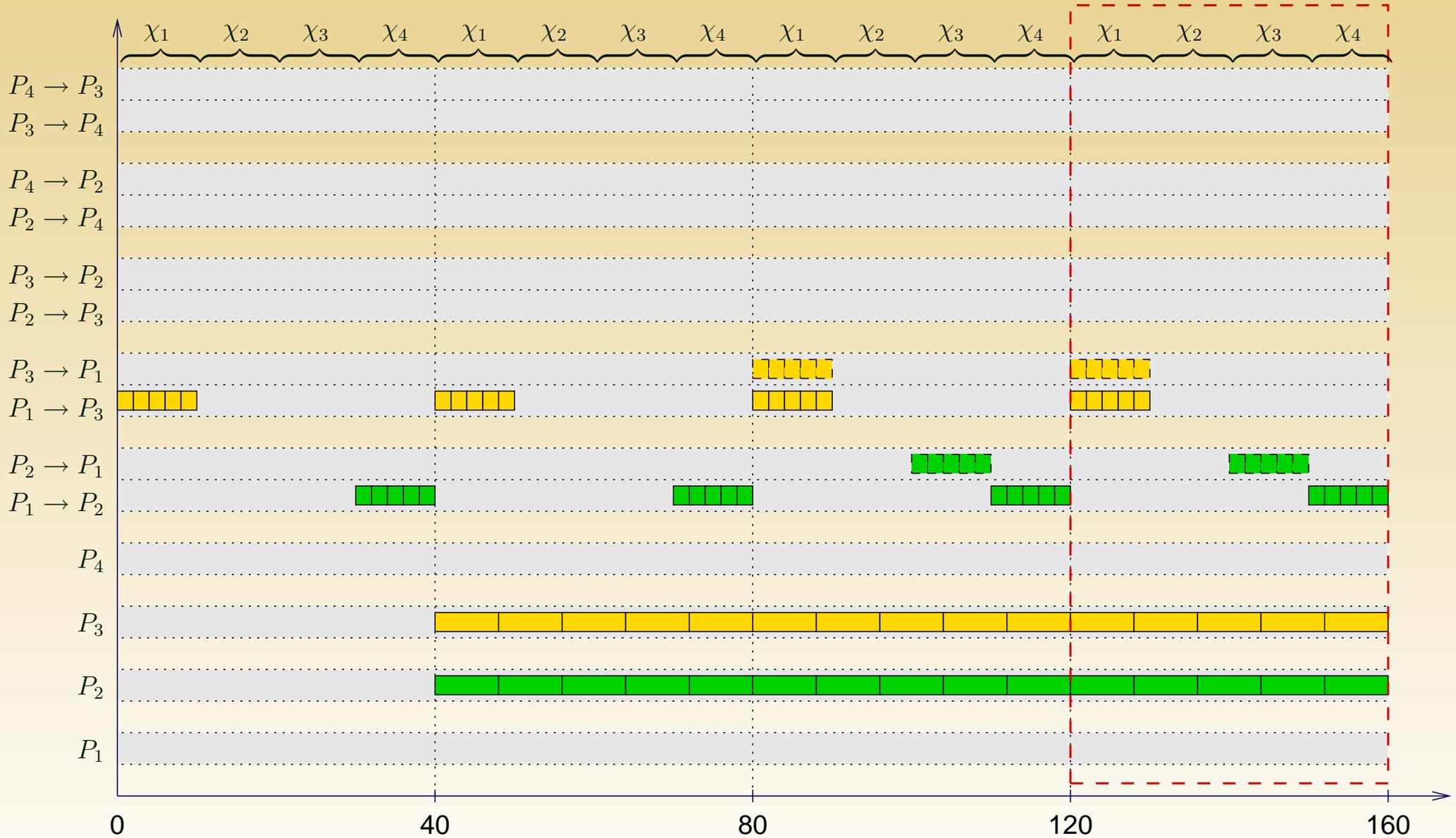
Ordonnancement cyclique de débit optimal

\mathcal{A}_1
 \mathcal{A}_2
 \mathcal{A}_3
 \mathcal{A}_4
 \mathcal{A}_5



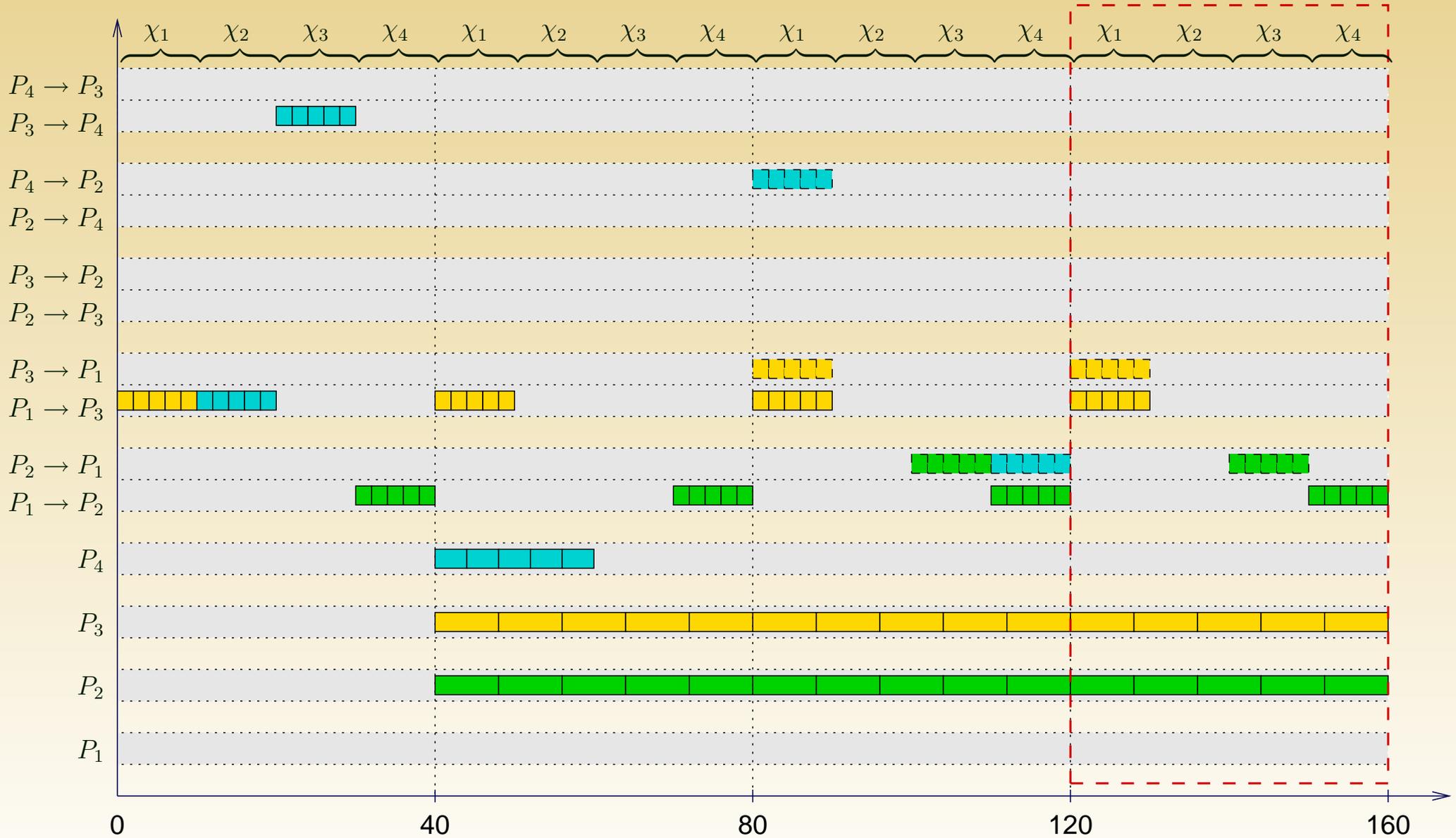
Ordonnancement cyclique de débit optimal

\mathcal{A}_1
 \mathcal{A}_2
 \mathcal{A}_3
 \mathcal{A}_4
 \mathcal{A}_5



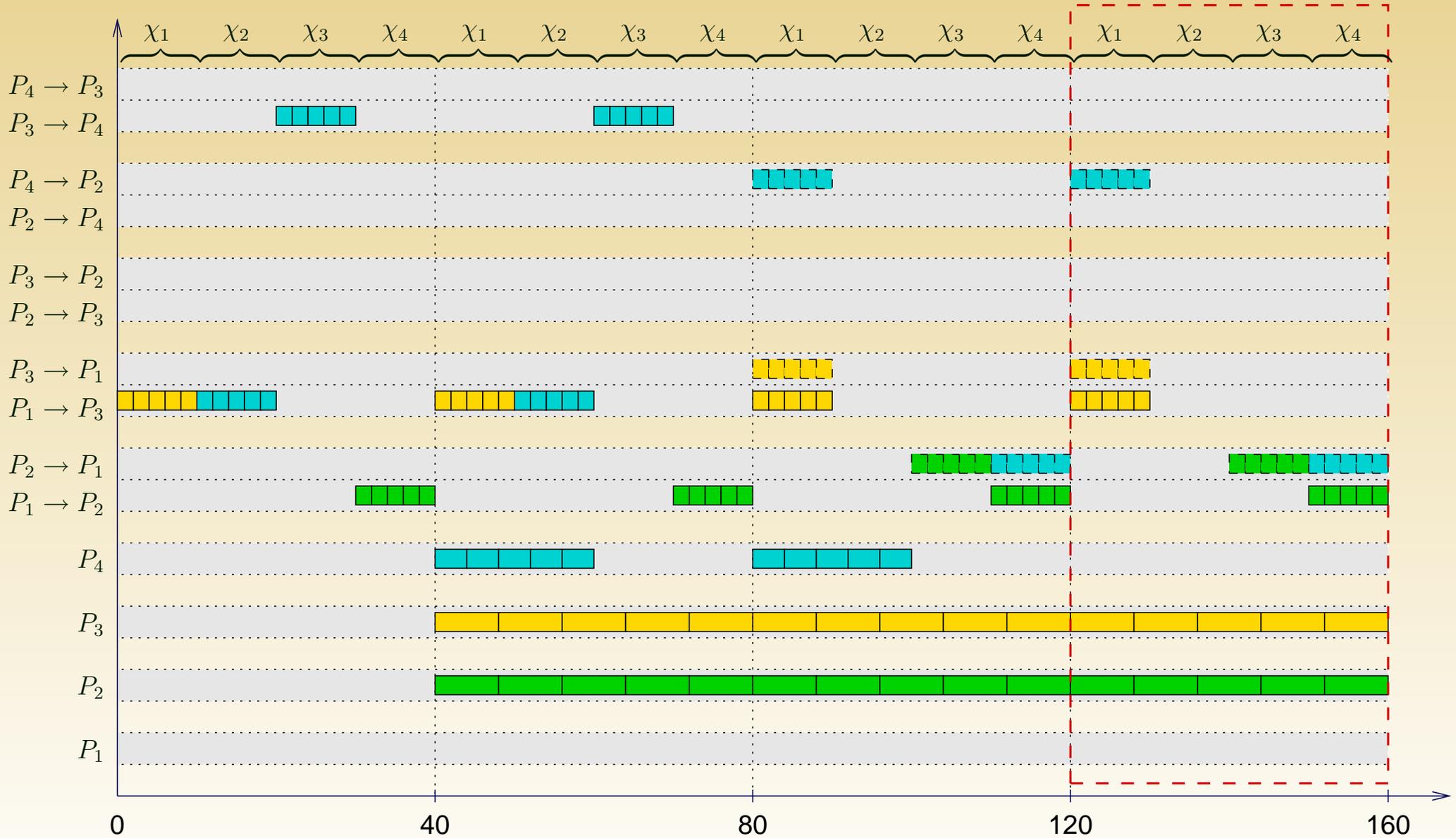
Ordonnancement cyclique de débit optimal

\mathcal{A}_1
 \mathcal{A}_2
 \mathcal{A}_3
 \mathcal{A}_4
 \mathcal{A}_5



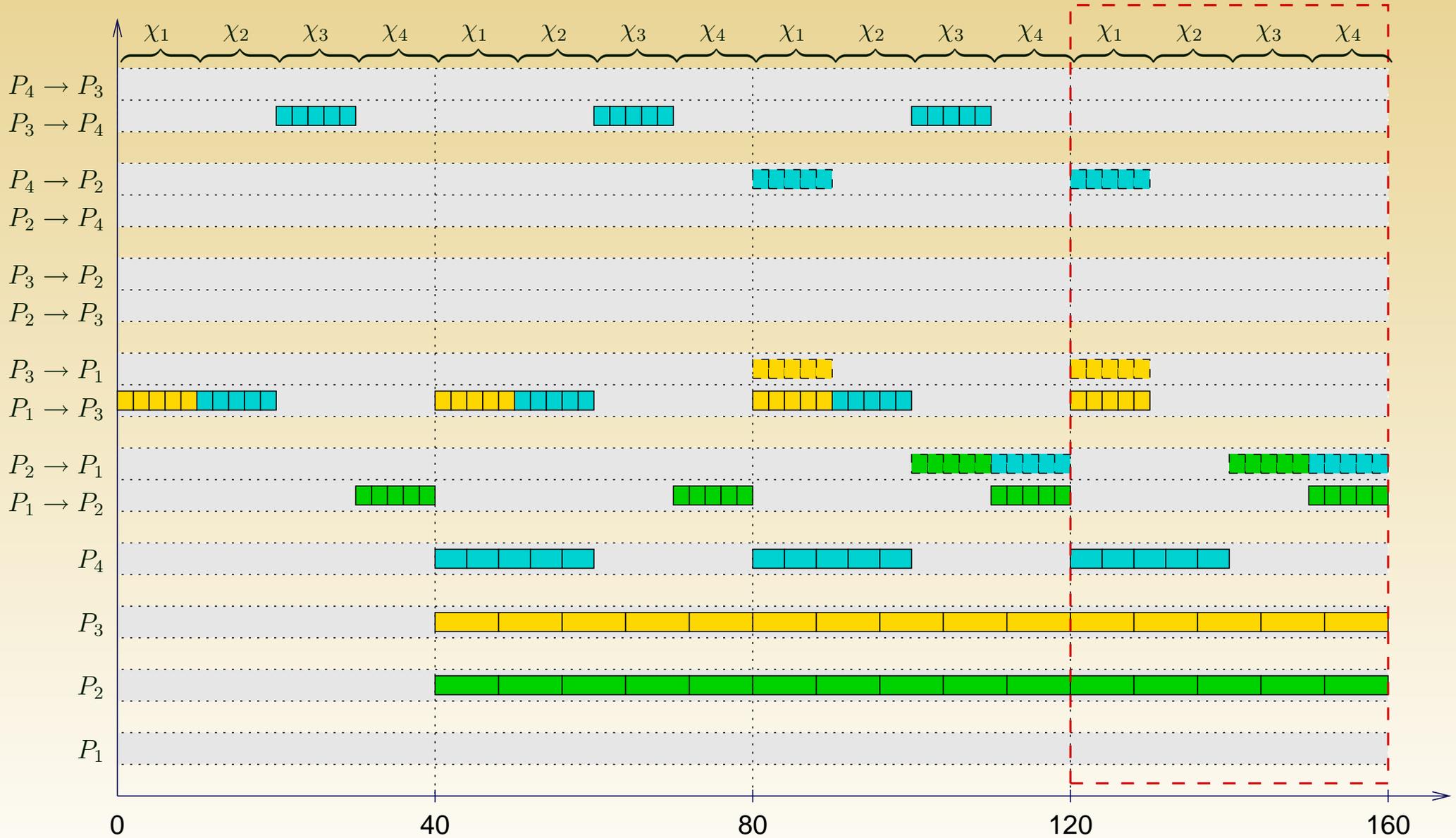
Ordonnancement cyclique de débit optimal

\mathcal{A}_1
 \mathcal{A}_2
 \mathcal{A}_3
 \mathcal{A}_4
 \mathcal{A}_5



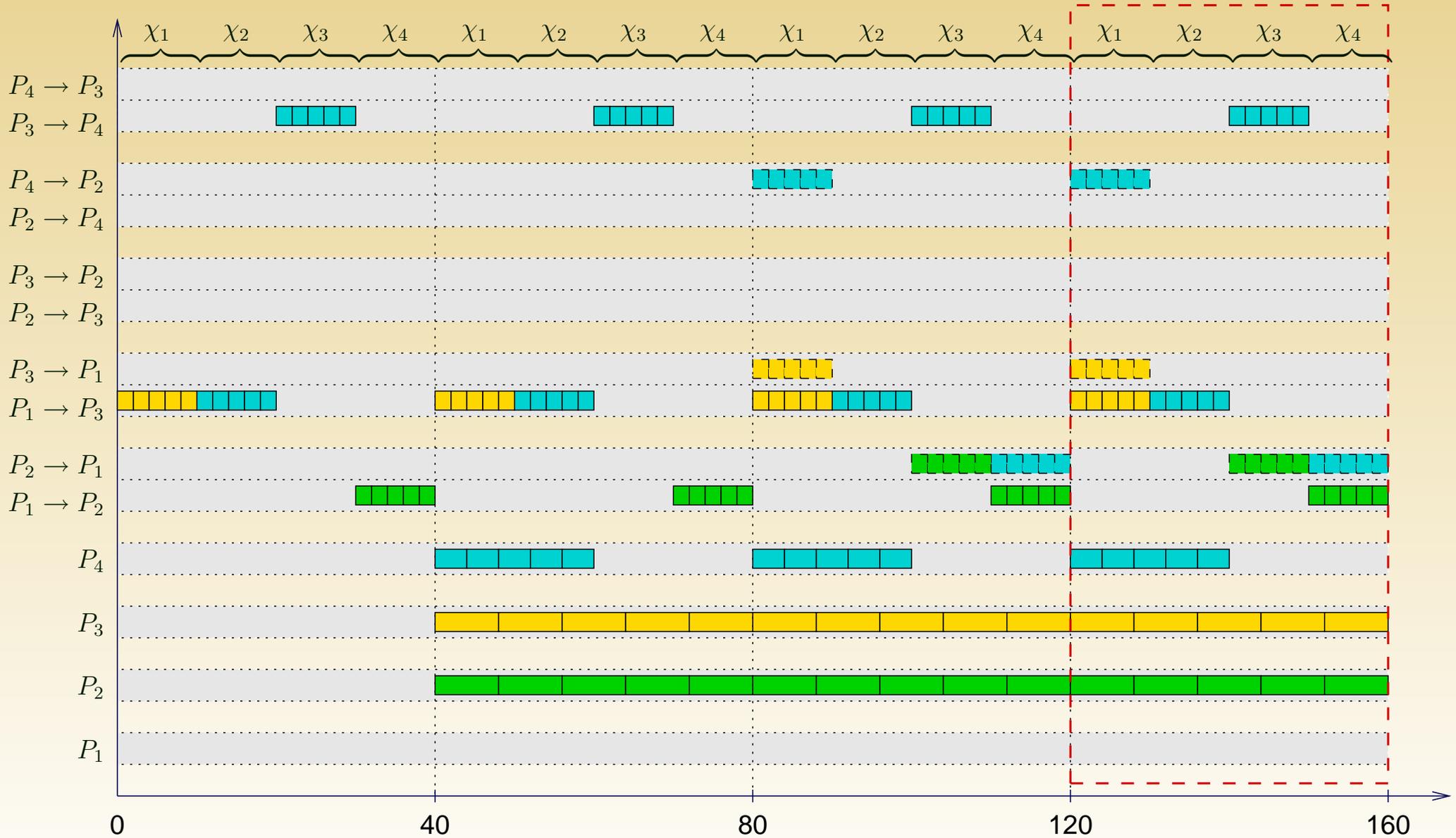
Ordonnancement cyclique de débit optimal

\mathcal{A}_1
 \mathcal{A}_2
 \mathcal{A}_3
 \mathcal{A}_4
 \mathcal{A}_5



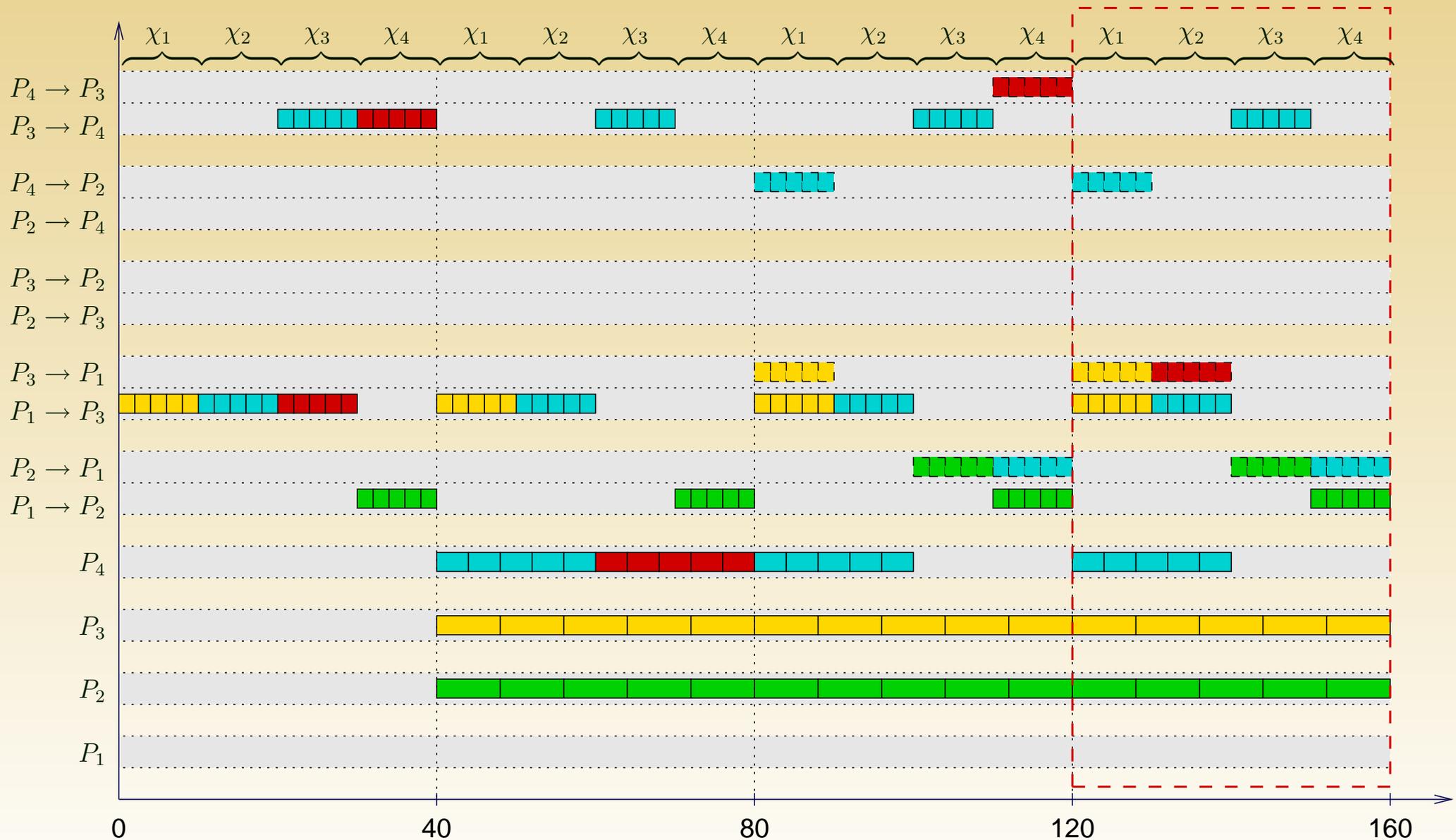
Ordonnancement cyclique de débit optimal

\mathcal{A}_1
 \mathcal{A}_2
 \mathcal{A}_3
 \mathcal{A}_4
 \mathcal{A}_5



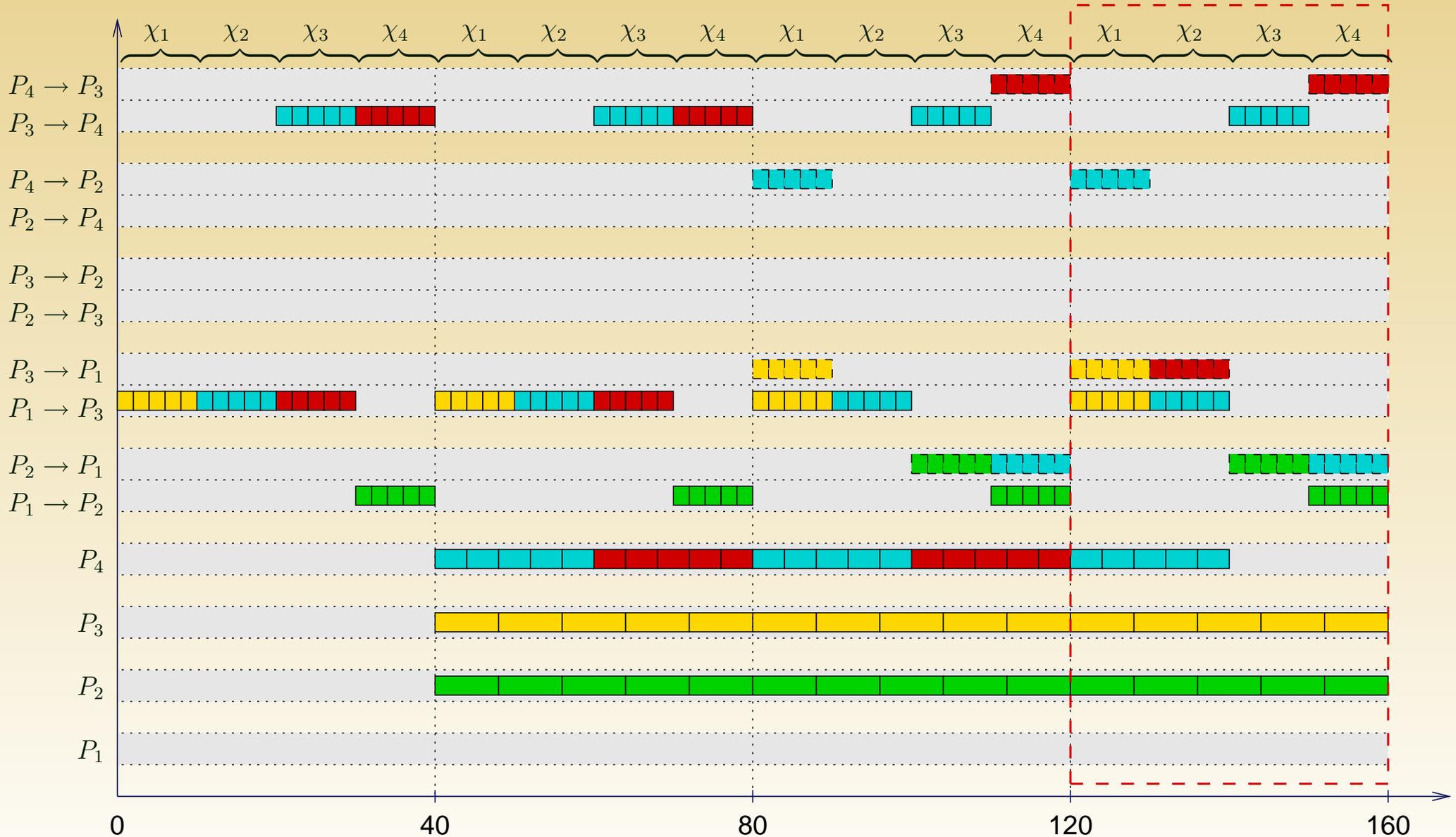
Ordonnancement cyclique de débit optimal

\mathcal{A}_1
 \mathcal{A}_2
 \mathcal{A}_3
 \mathcal{A}_4
 \mathcal{A}_5



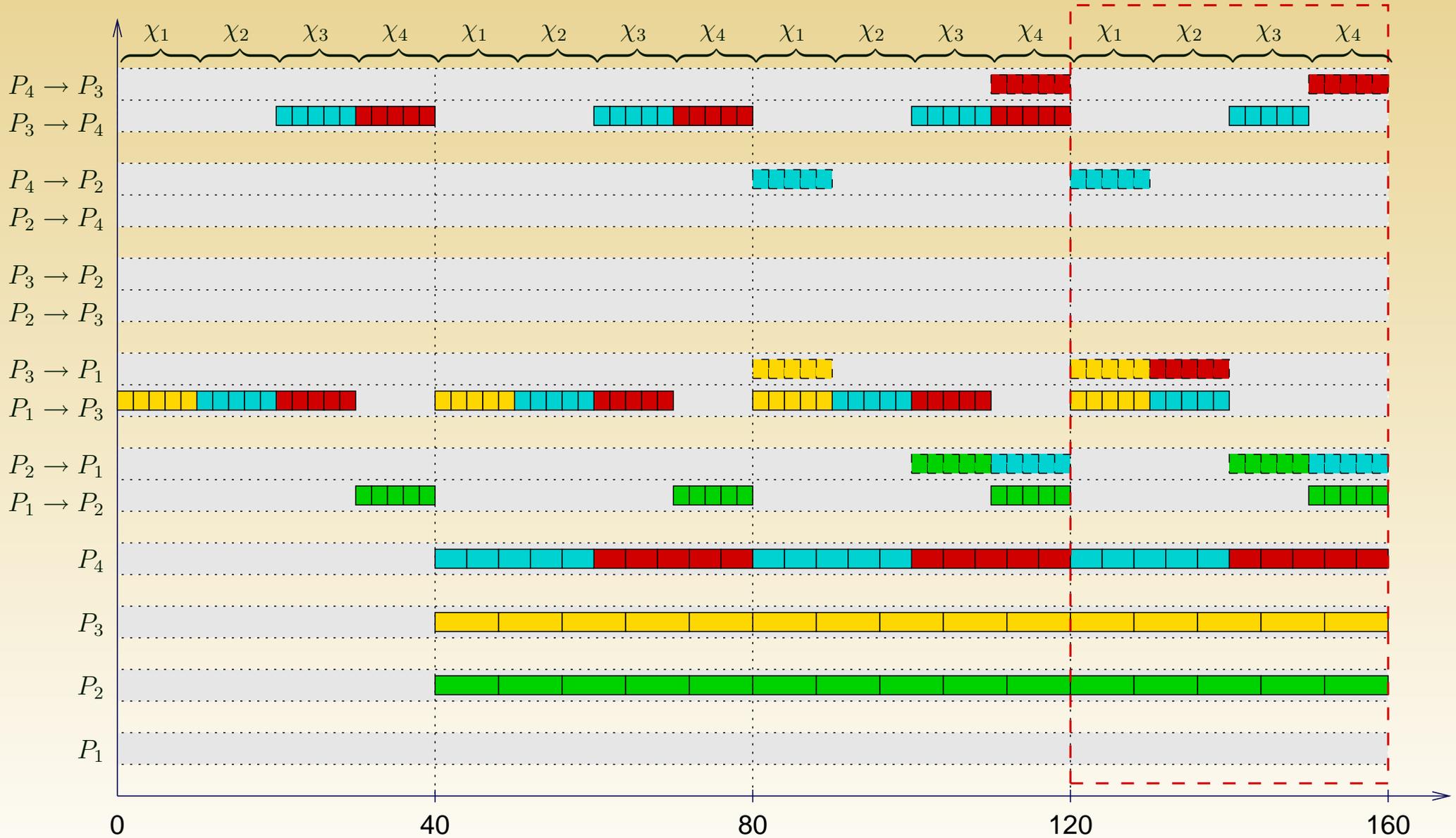
Ordonnancement cyclique de débit optimal

\mathcal{A}_1
 \mathcal{A}_2
 \mathcal{A}_3
 \mathcal{A}_4
 \mathcal{A}_5



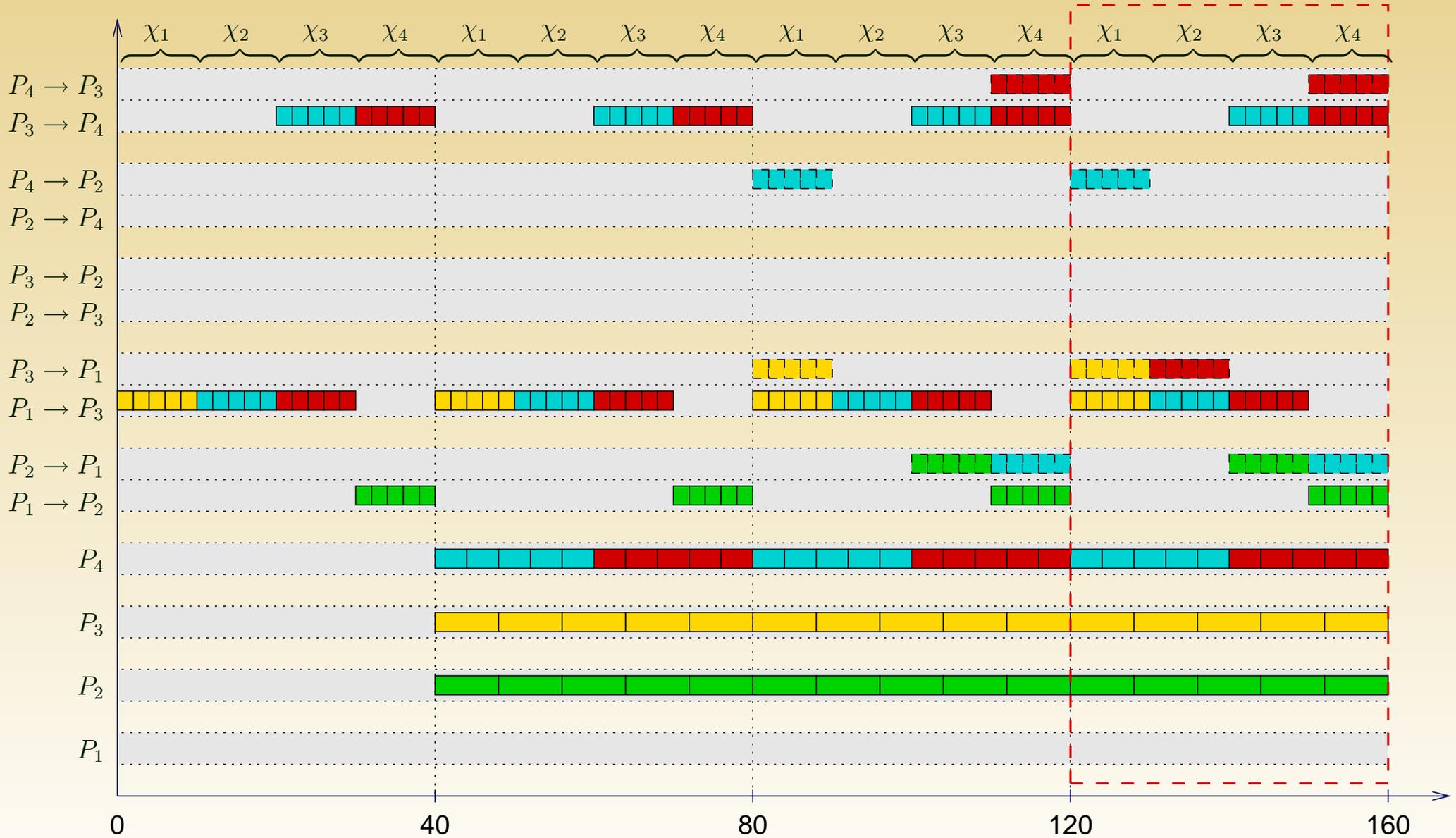
Ordonnancement cyclique de débit optimal

\mathcal{A}_1
 \mathcal{A}_2
 \mathcal{A}_3
 \mathcal{A}_4
 \mathcal{A}_5



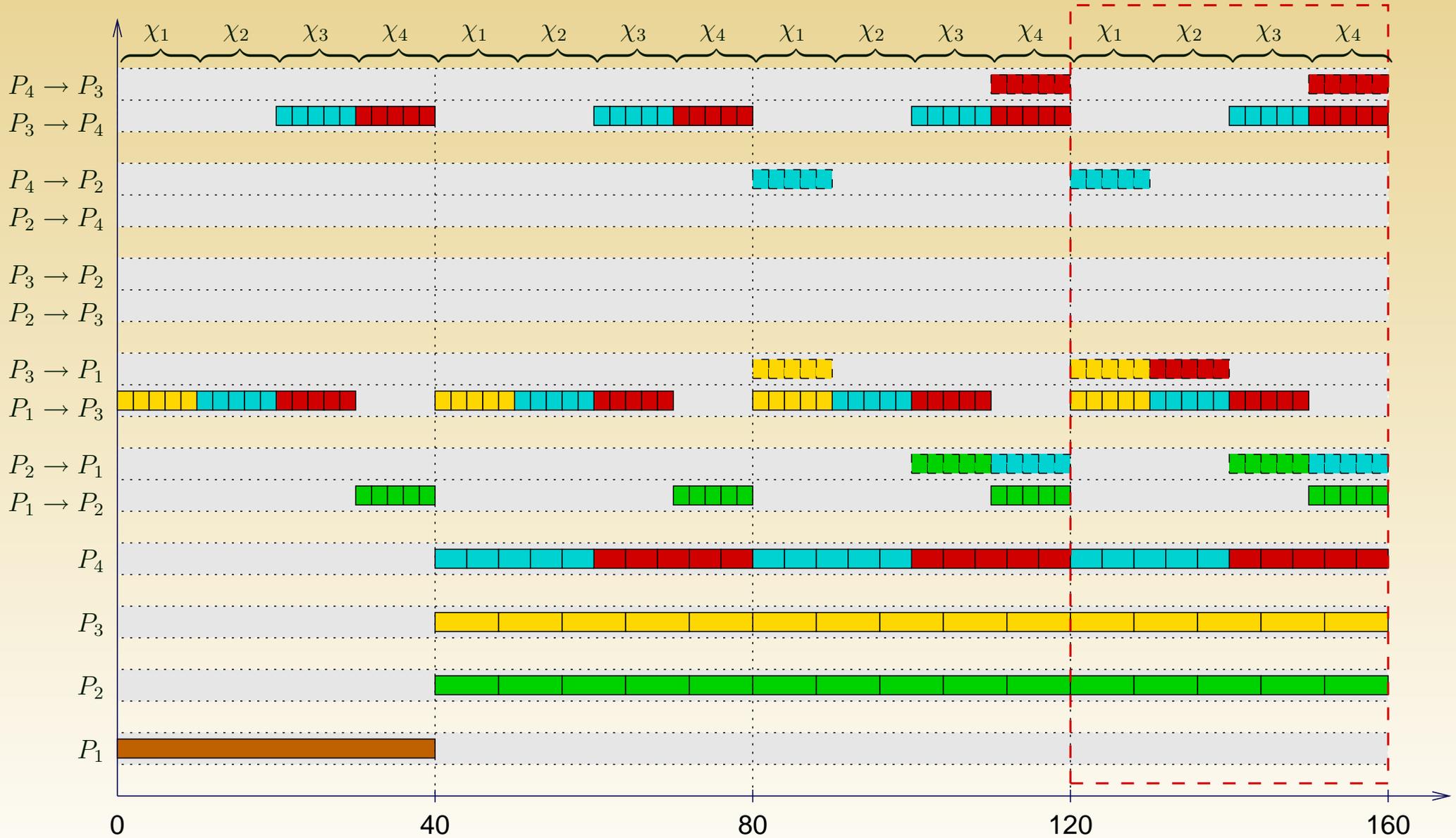
Ordonnancement cyclique de débit optimal

\mathcal{A}_1
 \mathcal{A}_2
 \mathcal{A}_3
 \mathcal{A}_4
 \mathcal{A}_5



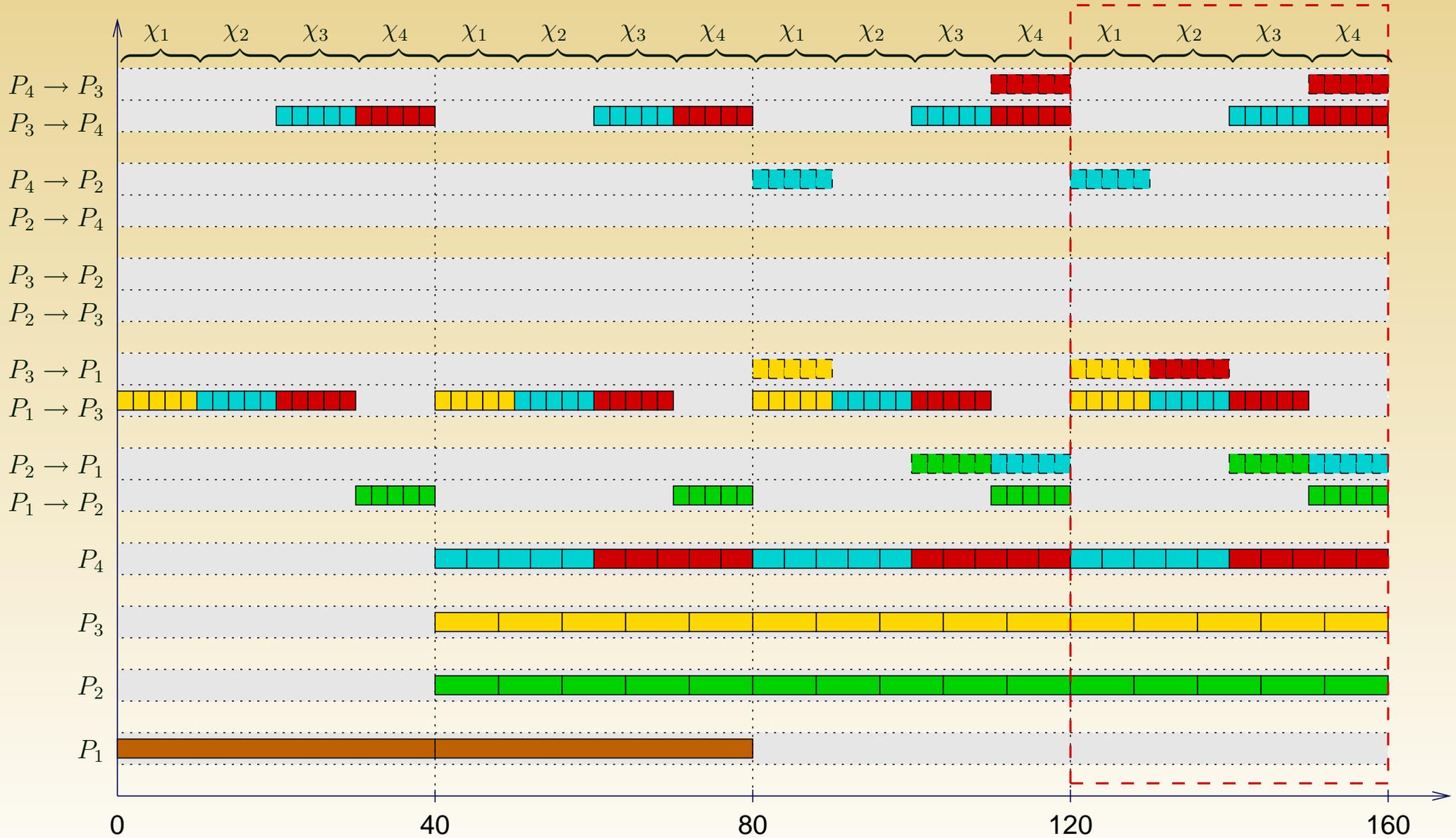
Ordonnancement cyclique de débit optimal

\mathcal{A}_1
 \mathcal{A}_2
 \mathcal{A}_3
 \mathcal{A}_4
 \mathcal{A}_5



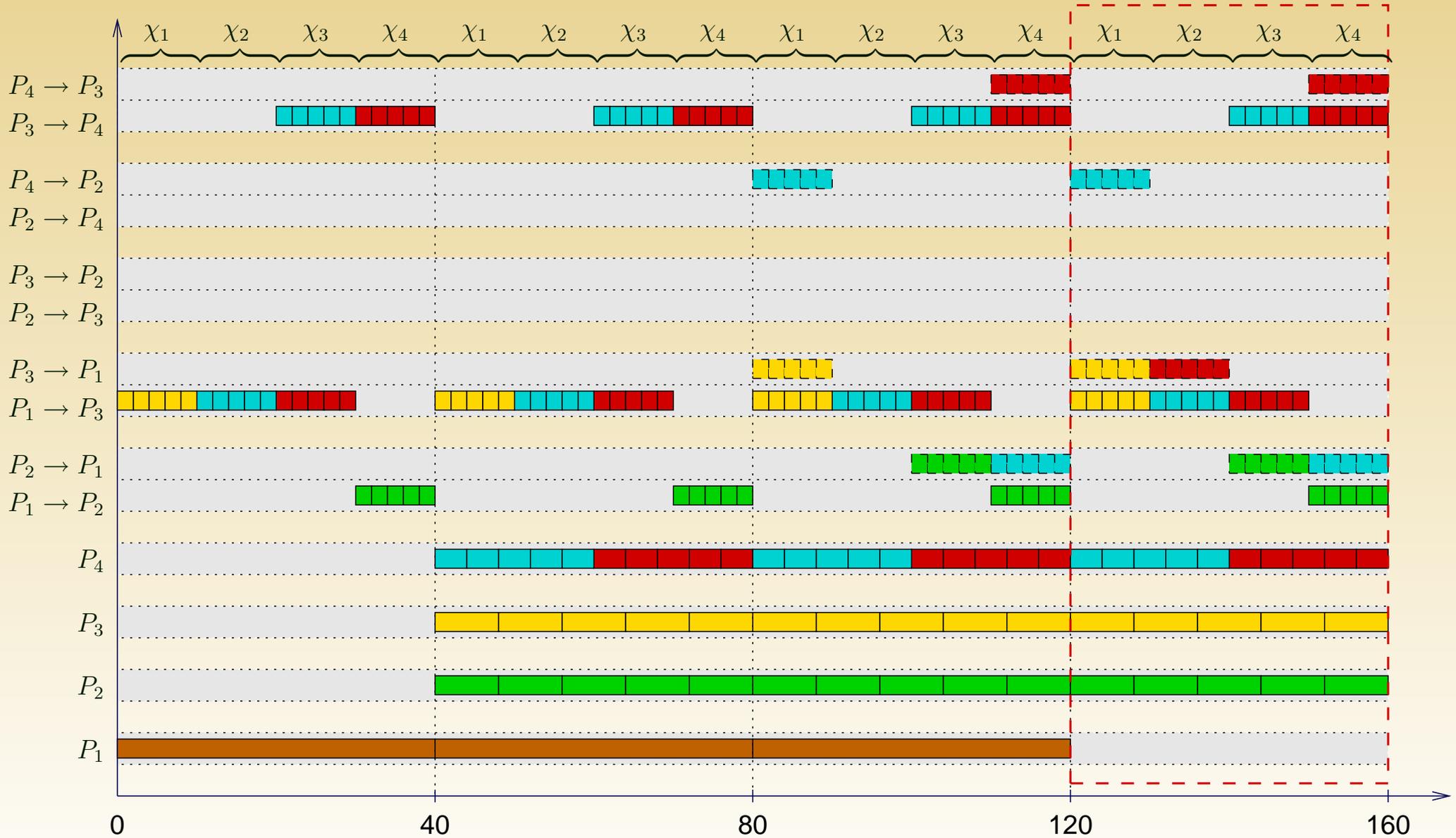
Ordonnancement cyclique de débit optimal

\mathcal{A}_1
 \mathcal{A}_2
 \mathcal{A}_3
 \mathcal{A}_4
 \mathcal{A}_5



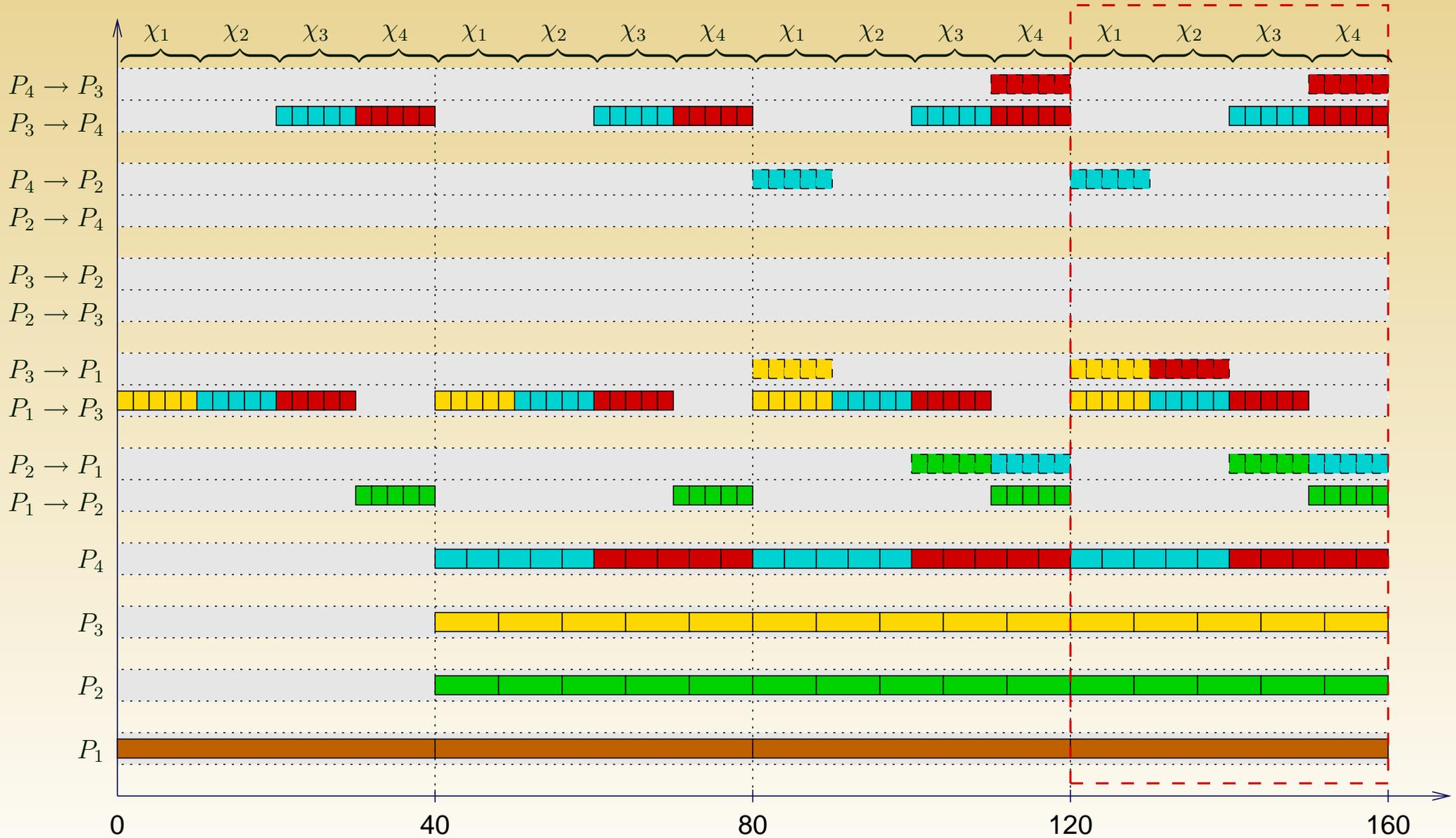
Ordonnancement cyclique de débit optimal

\mathcal{A}_1
 \mathcal{A}_2
 \mathcal{A}_3
 \mathcal{A}_4
 \mathcal{A}_5



Ordonnancement cyclique de débit optimal

\mathcal{A}_1
 \mathcal{A}_2
 \mathcal{A}_3
 \mathcal{A}_4
 \mathcal{A}_5



Ordonnancement asymptotiquement optimal

La technique employée pour résoudre cet exemple simple se généralise et 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.

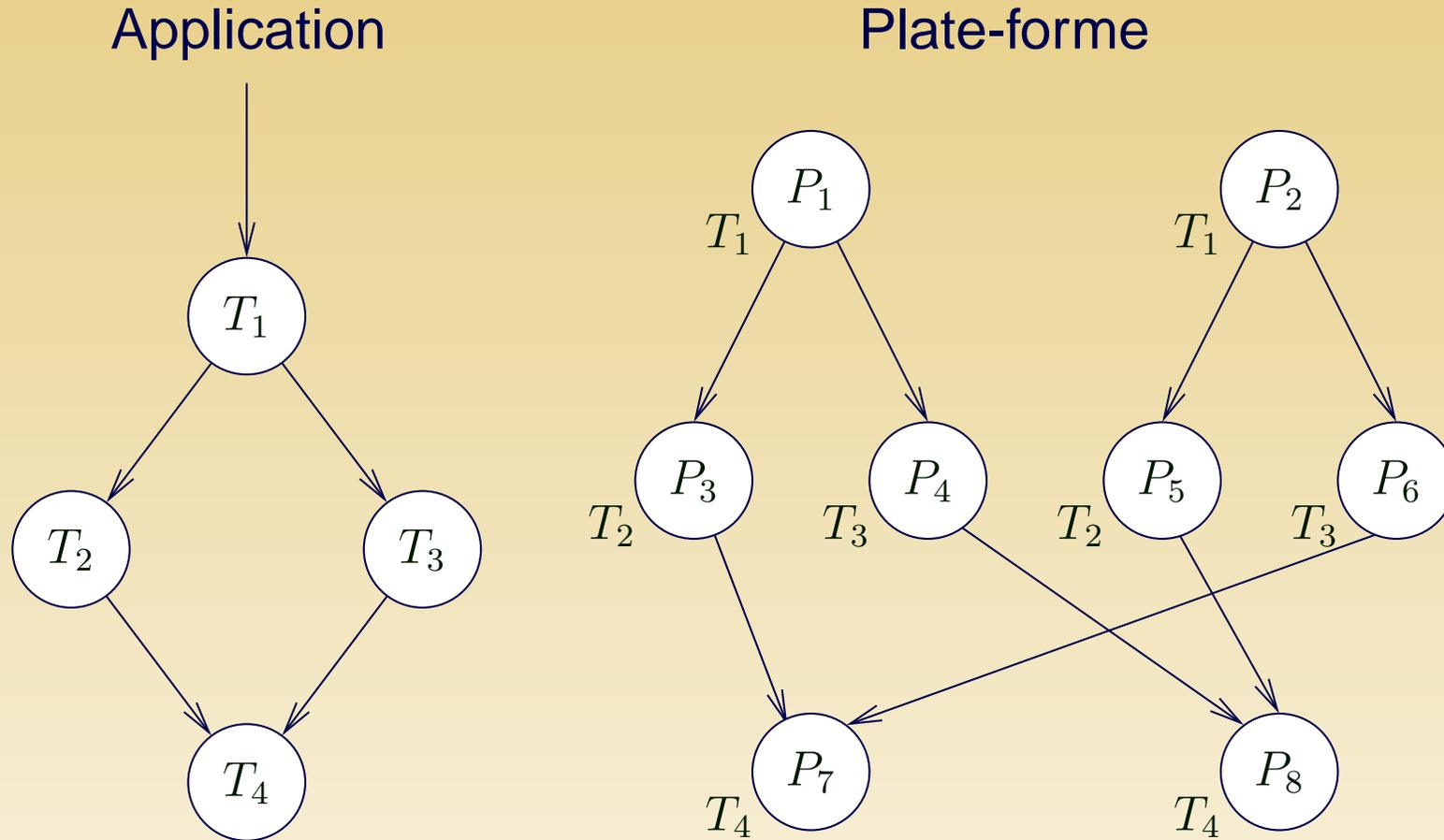
Ordonnancement asymptotiquement optimal

La technique employée pour résoudre cet exemple simple se généralise et 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

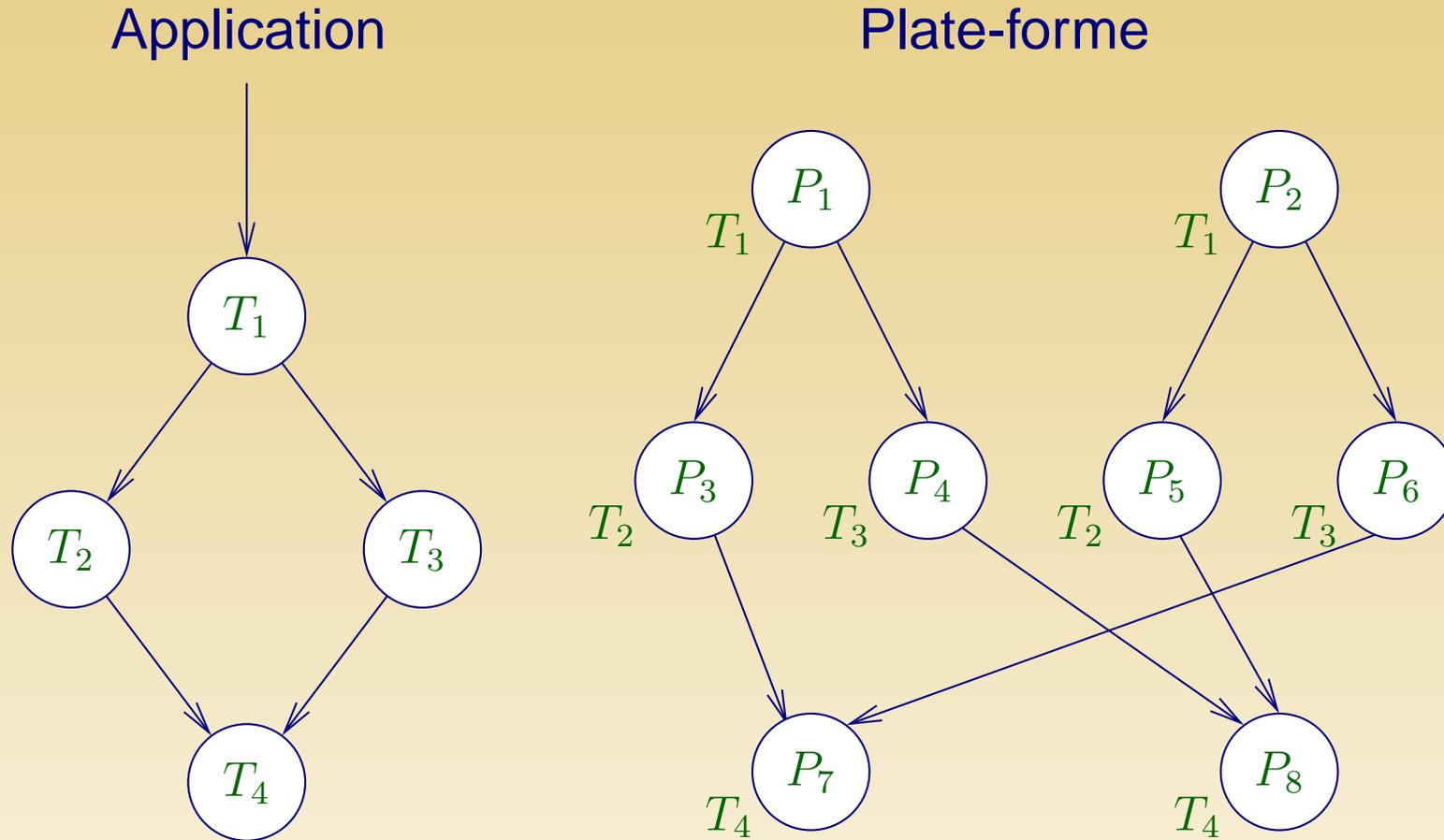
Les équations précédentes ne marchent plus...



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

⇒ Introduire une partie de l'ordonnancement de certains ancêtres dans le programme linéaire.

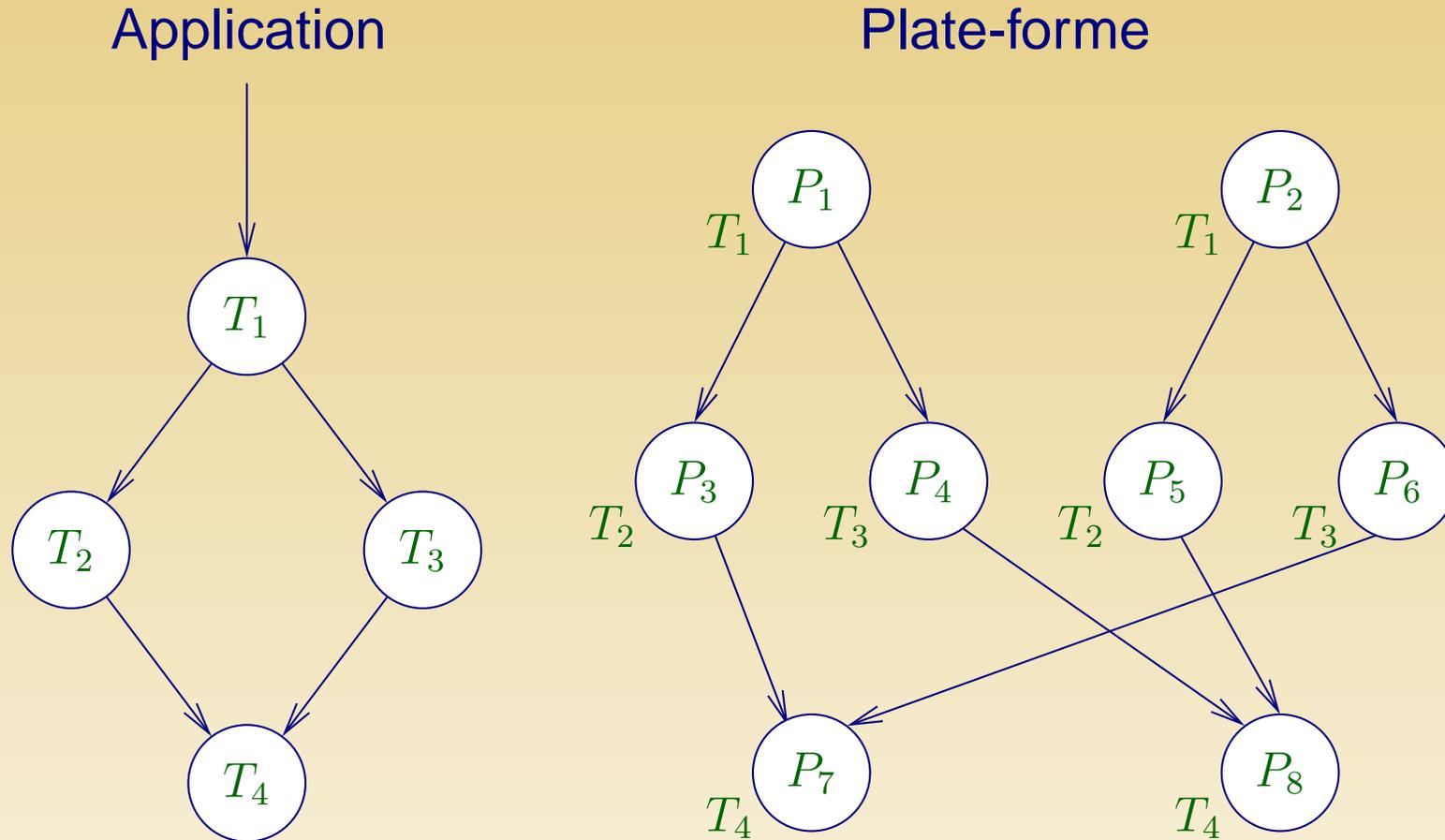
Les équations précédentes ne marchent plus...



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

⇒ 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 : la loi de conservation n'est plus valide.

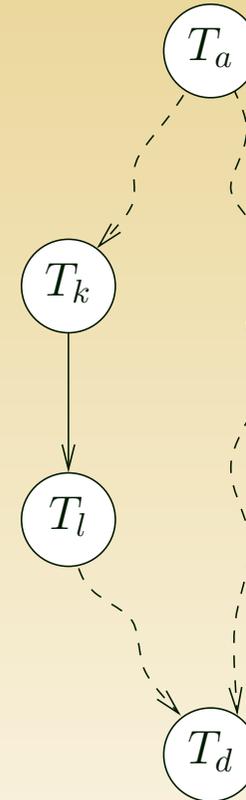
⇒ Introduire une partie de l'ordonnancement de certains ancêtres dans le programme linéaire.

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 $\{T_{begin} \mapsto P_1, T_1 \mapsto P_2, T_2 \mapsto P_2\}$).

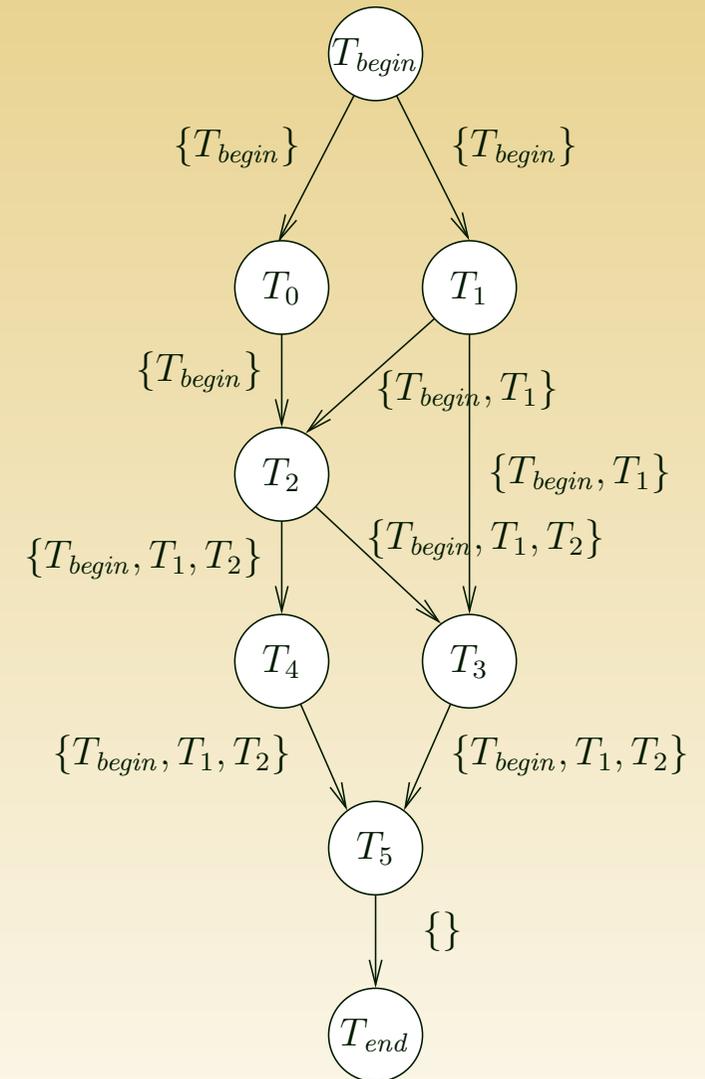


Tâche contraignante

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 $\{T_{begin} \mapsto P_1, T_1 \mapsto P_2, T_2 \mapsto P_2\}$).

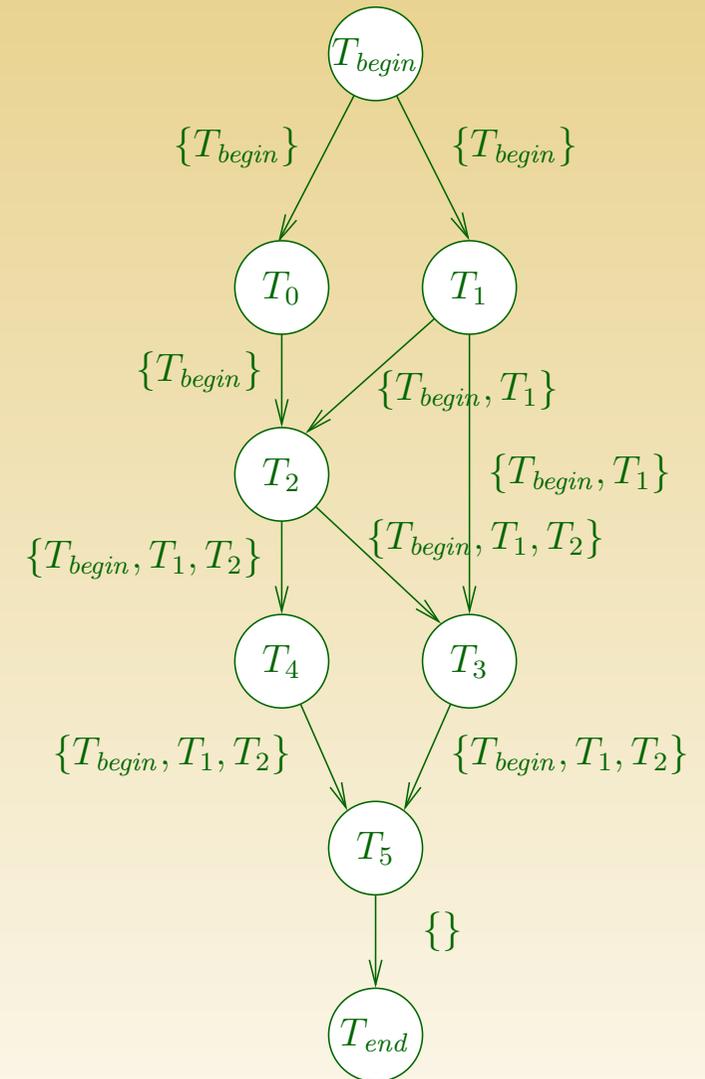


Tâche contraignante

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 $sent(P_i \rightarrow P_j, e_{k,l}^L)$ et $cons(P_i, T_k^L)$.

Une loi de conservation un peu plus compliquée :

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

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

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 $sent(P_i \rightarrow P_j, e_{k,l}^L)$ et $cons(P_i, T_k^L)$.

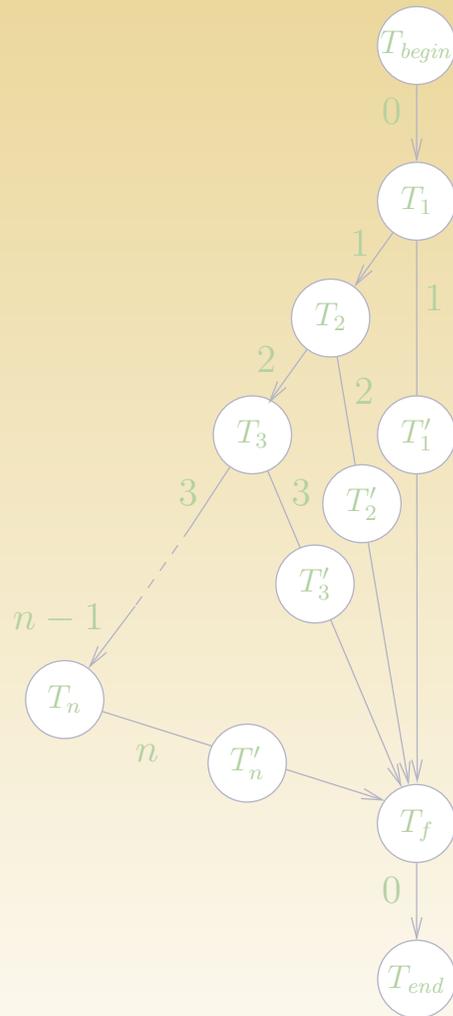
Une loi de conservation un peu plus compliquée :

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

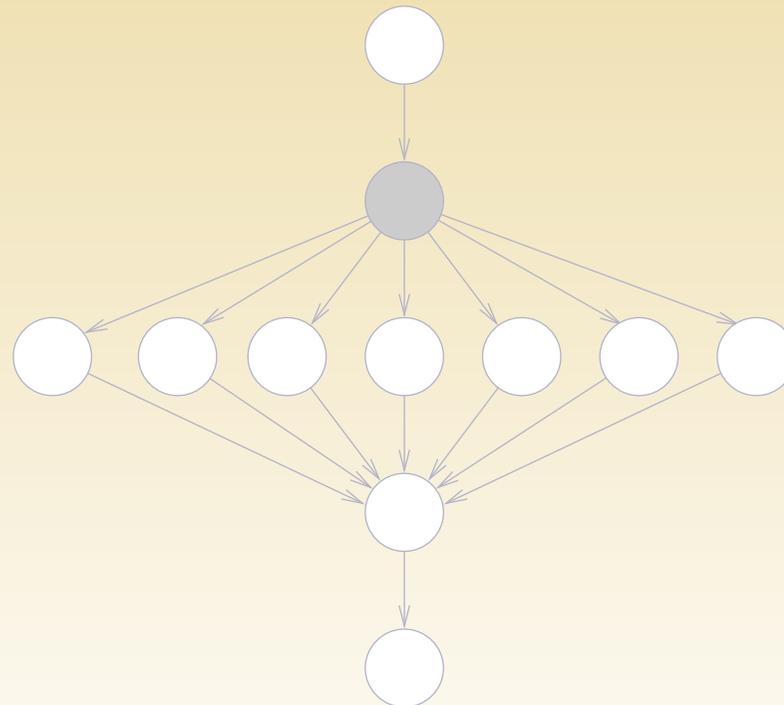
$$\sum_{P_j \rightarrow P_i} sent(P_j \rightarrow P_i, e_{k,l}^L) + cons(P_i, T_k^L) =$$
$$\sum_{P_i \rightarrow P_j} sent(P_i \rightarrow P_j, e_{k,l}^L) + \sum_{\substack{L_2 \in CnstsIn(T_l) \\ L \text{ et } L_2 \text{ compatibles}}} prod(P_i, T_l^{L_2})$$

Profondeur de dépendance

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



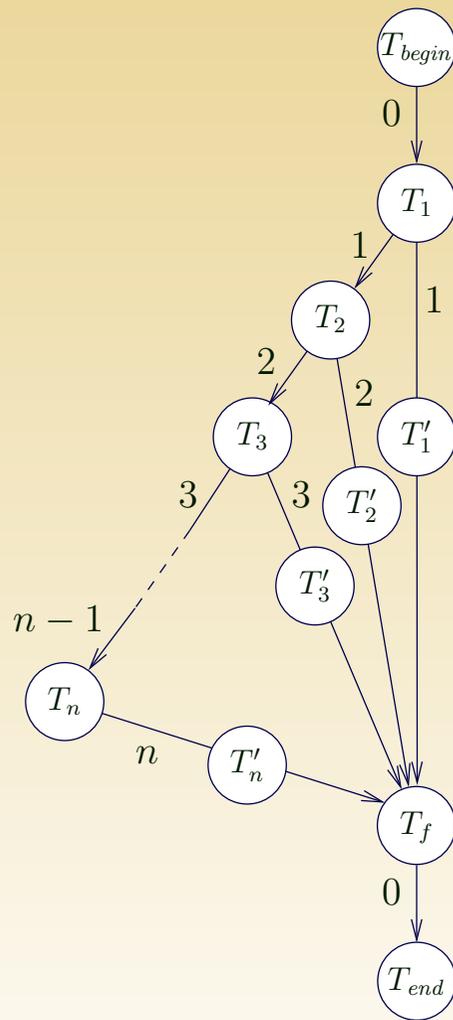
Profondeur : n



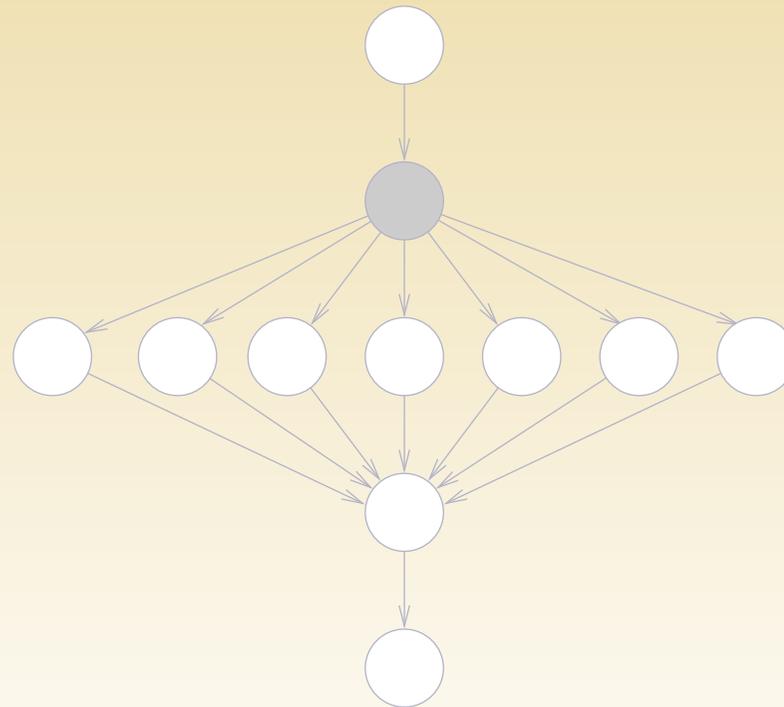
Profondeur : 1

Profondeur de dépendance

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



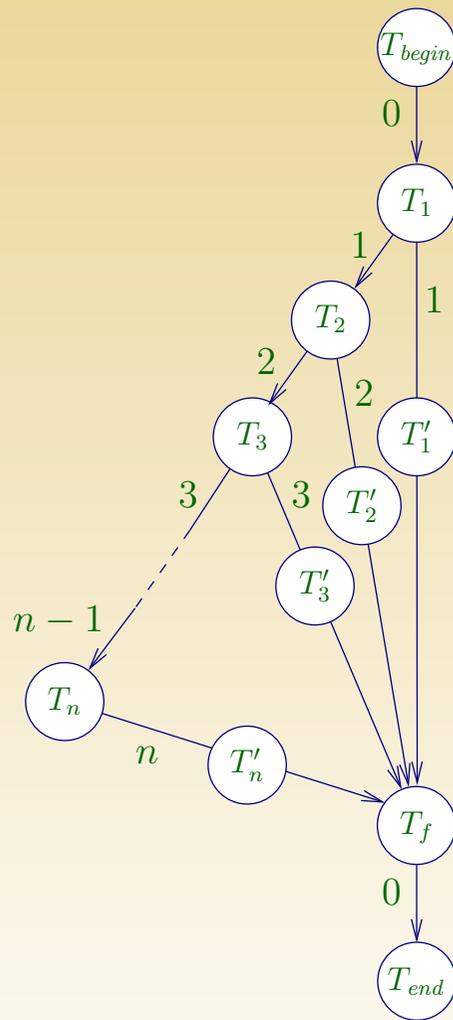
Profondeur : n



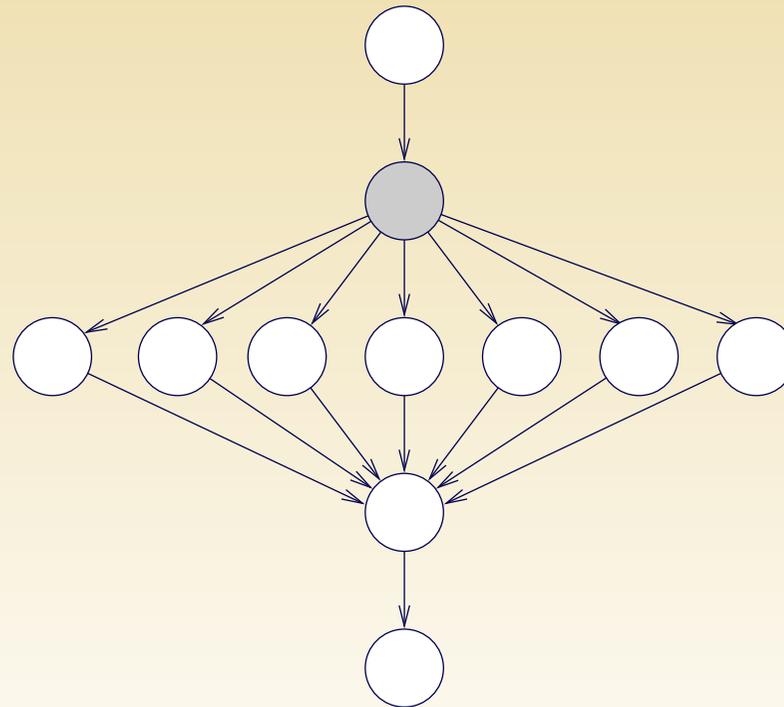
Profondeur : 1

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 ($\text{Reg-Perm}_d(G_a, G_p)$). Étant donné un graphe de tâches G_a dont la profondeur de dépendance est bornée par d et un graphe de plateforme 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 ($\text{Reg-Perm}_d(G_a, G_p)$). Étant donné un graphe de tâches G_a dont la profondeur de dépendance est bornée par d et un graphe de plateforme 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 ($\text{Reg-Perm}_d(G_a, G_p)$). Étant donné un graphe de tâches G_a dont la profondeur de dépendance est bornée par d et un graphe de plateforme 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.
- 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.
- 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.
- 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.

Perspectives

- Les idées présentées ici s'étendent à d'autres situations :
 - Tâches divisibles
 - Macro-communications (scatter, gather, reduce, broadcast, ...)

Un certain nombre de problèmes restent cependant ouverts :

- taille de la période, approximation du motif ;
 - dans quelles situations la programmation linéaire est-elle suffisamment expressive ?
 - stabilité, résistance aux variations de charge :  SIMGRID.
- De la simulation à la mise en œuvre réelle : APST, DIET, OAR, ...

Perspectives

- Les idées présentées ici s'étendent à d'autres situations :
 - Tâches divisibles
 - Macro-communications (scatter, gather, reduce, broadcast, ...)

Un certain nombre de problèmes restent cependant ouverts :

- taille de la période, approximation du motif ;
 - dans quelles situations la programmation linéaire est-elle suffisamment expressive ?
 - stabilité, résistance aux variations de charge : \rightsquigarrow SIMGRID.
- De la simulation à la mise en œuvre réelle : APST, DIET, OAR, ...

Perspectives

- Les idées présentées ici s'étendent à d'autres situations :
 - Tâches divisibles
 - Macro-communications (scatter, gather, reduce, broadcast, ...)

Un certain nombre de problèmes restent cependant ouverts :

- taille de la période, approximation du motif ;
 - dans quelles situations la programmation linéaire est-elle suffisamment expressive ?
 - stabilité, résistance aux variations de charge :  SIMGRID.
- De la simulation à la mise en œuvre réelle : APST, DIET, OAR, ...

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.