**Notions job-shop, Group-shop, Open-shop**

En résumé, le GSPPR nécessite de planifier toutes les opérations efficacement sans dépasser le seuil de puissance. Notez que le GSP généralise le JSP et le OSP. En effet, une instance du GSP avec une seule opération par groupe est une instance du JSP, et une instance du GSP avec un seul groupe par tâche est une instance du OSP.

**Enoncé formelle du Problème d'Ordonnancement de Groupe-Shop avec Exigences en Puissance (GSPPR)**

Le GSPPR consiste à ordonnancer un ensemble de n jobs (tâches), où chaque tâche j se compose d'un ensemble = { **. . .** } d'opérations à effectuer sur les machines  **. . .** , respectivement. Chaque opération de l'ensemble des opérations **O** est subdivisée en sous-opérations , et chacune de ces sous-opérations de l’ensemble des sous-opérations nécessite un temps déterminé pour son exécution ainsi qu'une puissance requise pour sa réalisation .

L'objectif du problème considéré est de minimiser le temps d'achèvement (makespan) , c'est-à-dire le temps d'achèvement de la dernière tâche effectuée. Cependant, l'ordonnancement doit respecter certaines contraintes de précédence entre les opérations. Plus précisément, l'ensemble des opérations d'une tâche **j** est partitionné en groupes, et désigne le k-ème groupe de la tâche **j**. Les contraintes de précédence exigent de terminer toutes les opérations du groupe avant le début de toute opération du groupe si **k** ≤ **l**. Cependant, les opérations d'un groupe peuvent être planifiées dans n'importe quel ordre. De plus, l'ordonnancement doit respecter le seuil d'énergie, c'est-à-dire que la consommation totale d'énergie des opérations effectuées simultanément doit être inférieure au seuil . Enfin, les opérations ne peuvent pas être interrompues et sont disponibles dès le temps t=0.

**Formulation mathématique du problème.**

Le modèle de la formulation mathématique avec la représentation des flux d’énergie pour le GSPPR est basé sur les variables suivantes :

- : Variable binaire indiquant si l’opération **i** est traitée avant l’opération **j**

(1 si l'opération **i** est traitée avant l’opération **j**, 0 sinon).

- : la date de début de la sous opération **k** de l’opération **i**

-  : la date de fin de la sous opération **k** de l’opération **i**

- : la durée de la sous opération **k** de l’opération **i**

-  : le besoin en puissance de la sous opération **k** de l’opération **i**

-  : représentant le flux de sous-opération entre les opérations **i** et **j**, et entre les sous-opérations **k** et **k′**.

-  : variable binaire représentant l’attente de flux entre les sous opérations

(1 s’il y a un flux entre sous opération **k** de l'opération **i** et la sous opération **k'** de l'opération **j**, 0 sinon)

**Minimiser**

**Sujet à**

- Entre 2 opérations, soit l'une précède l'autre, soit c'est l’inverse.

- Une sous-opération k+1 commence directement lorsque la sous-opération k se termine :

- La somme des puissances injectées dans une sous-opération doit être égale à son besoin :

* Ce qui sort d'une sous-opération ne peut pas excéder ce qui a été consommé :

- La somme des flux entrants ne peut pas excéder la capacité maximale :

🡪Prenons pour la suite **H** un nombre positif grand (H)

- Identification de flux entre les opérations (attente):

- mise à jour des dates de débuts en fonction des précédences :

- Gérer le flux entre opération effectuer sur une même machine :

Avec  : la dernière sous-opération de l’opération **i**

- réglez le makespan sur le temps de réalisation de la dernière opération :