ATRSC : PROJET
GESTION OPTIMISÉE D'ATELIER

Ce projet est à faire en binôme et à rendre par mail pour le 23 mai minuit au plus tard. Il faut fournir

— un document rassemblant vos réponses aux différentes questions,

— différents bouts de codes, comme expliqué plus en détail dans l'énoncé ci-dessous.

Un petit code écrit en Python, disponible sur Educnet, permet la simulation de l'atelier. Il est demandé de l'utiliser pour toutes les questions requérant de l'implémentation.

1. Fonctionnement de l'atelier

On considère un atelier dans lequel arrivent des produits à usiner. Dans cet atelier, il y a m machines et q employés. Un employé ne peut travailler sur une machine que s'il possède la qualification de travail sur cette machine.

Il y a k types de produits. À chaque type correspond une séquence de machines à visiter. Le temps de passage d'un produit sur une machine est tiré uniformément au hasard dans un intervalle dont les bornes dépendent uniquement du type. Les produits arrivent dans l'atelier selon un processus de Poisson dont l'intensité dépend également uniquement du type.

On voit chaque machine comme une file d'attente avec un *espace d'attente* et un *espace de service*. Lorsqu'un produit arrive sur une machine, deux situations sont possibles. Soit la machine est complètement vide, auquel cas le produit se met dans l'espace de service ; soit il y a déjà un ou plusieurs produits dans la machine, auquel cas le produit qui vient d'arriver se met dans l'espace d'attente de la machine.

Lorsque les employés sont inactifs, ils sont dans une *salle d'attente*. Lorsqu'un produit arrive dans l'espace de service d'une machine, l'employé possédant la qualification pour travailler sur cette machine se rend sur la machine et réalise la tâche attendue. Si plusieurs employés possèdent la qualification, c'est celui qui est dans la salle d'attente depuis le plus longtemps qui est choisi. Si aucun employé ne possède la qualification, le produit se met *en attente*.

Lorsqu'un employé termine une tâche, on regarde s'il y a des produits en attente sur des machines dont il possède la qualification. L'employé peut alors se rendre sur une de ces machines et réaliser la tâche attendue. Lorsqu'il y a plusieurs telles machines, un algorithme ALGO détermine sur laquelle l'employé doit se rendre. L'algorithme peut toujours aussi décider d'envoyer l'employé dans la salle d'attente. S'il n'y a pas de produit en attente sur une machine dont il possède la qualification, l'employé se rend dans tous les cas dans la salle d'attente.

Noter que l'ordre des produits dans un espace d'attente ne peut être modifié.

2. Instances

On considère quatre instances différentes. Toutes les instances ont le même nombre de machines, à savoir 8, et le même nombre de type de produits, à savoir 4. On a donc $m = 8$ et $k = 4$.

De plus elles ont les mêmes paramètres de processus d'arrivées (voir table 1), les mêmes parcours (voir table 2) et les mêmes distributions des temps de traitement. Les temps de traitement sont tous tirés selon des lois uniformes sur des intervalles donnés dans la table 3.

Date: 4 avril 2023.

Types	T1	T2	T3	T4
Intensités	0.29	0.32	0.47	0.38

TABLE 1. Intensités des processus d'arrivées des différents types de produits

Machines		M1	M2	M3	M4	M5	M6	M7	M8
Types	T1	[0.58, 0.78]	[0.23, 0.56]	[0.81, 0.93]	[0.12, 0.39]				[0.82, 1.04]
	T2		[0.59, 0.68]		[0.74, 0.77]			[0.30, 0.55]	
	T3	[0.57, 0.64]		[0.37, 0.54]		[0.35, 0.63]			
	T4					[0.36, 0.51]	[0.61, 0.70]	[0.78, 0.85]	[0.18, 0.37]

TABLE 2. Temps de traitement des différents produits

Machines		M1	M2	M3	M4	M5	M6	M7	M8
Types	T1	1	2	3	4				5
	T2		1		2			3	
	T3	3		1		2			
	T4					1	2	3	4

TABLE 3. Parcours des différents types de produits

$$Q(\text{I1}) = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix} \qquad Q(\text{I2}) = \begin{pmatrix} 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \end{pmatrix}$$

FIGURE 1. Matrices des qualifications pour les instances I1 et I2

En revanche, elles diffèrent par leurs nombres d'employés et par les qualifications de ces derniers. Deux instances, I1 et I2, ont $q = 4$ employés (voir figure 1) et deux instances, I3 et I4, en ont $q = 6$ (voir figure 2). Les lignes des matrices Q des qualifications représentent les employés et les colonnes les machines. L'employé i peut travailler sur la machine j uniquement si l'entrée à la ligne i, colonne j, vaut 1.

3. Minimiser le temps de séjour moyen

L'objectif de cette première partie est de minimiser le temps de séjour moyen d'un produit en régime permanent. Plus précisément, on considère le système en régime permanent sur une fenêtre de temps ; on fait la moyenne des temps de séjours des produits sur cette fenêtre de temps (pour tous les produits étant arrivés puis partis sur cette fenêtre) ; on regarde cette quantité lorsque la fenêtre de temps est arbitrairement grande.

Question 1. *Proposer en la justifiant une borne inférieure heuristique sur le temps de séjour moyen en régime permanent. On pourra faire un raisonnement indépendant pour chaque instance.*

Question 2. *Proposer un algorithme* ALGO *qui minimise le temps de séjour moyen en régime permanent, en justifiant autant que possible son choix.*

Question 3. *Fournir le code de l'algorithme. (Il faut que le code puisse être inclus à l'endroit prévu dans le simulateur*[1] *sans avoir à modifier d'autres parties du simulateur.) Indiquer les résultats obtenus avec des intervalles de confiance et commenter.*

1. C'est dans la méthode `algo` de la `class Worker`.

$$Q(\text{I3}) = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \qquad Q(\text{I4}) = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \end{pmatrix}$$

FIGURE 2. Matrices des qualifications pour les instances I3 et I4

4. ÉQUILIBRER LA CHARGE DES EMPLOYÉS

Quelque soit l'instance, on ne s'attend pas a priori à ce que la charge soit répartie équitablement entre les employés. Cela peut d'ailleurs se vérifier avec le simulateur, qui renvoie l'utilisation moyenne des employés. L'*utilisation* d'un employé sur une réalisation est la valeur asymptotique (quand t tend vers $+\infty$) de $\frac{\text{temps travaillé sur } [0,t]}{t}$. Le simulateur fait la moyenne sur les réplications des utilisations empiriques mesurées en régime permanent.

Question 4. *Expliquer pourquoi on s'attend en effet à ce qu'il y ait de tels écarts.*

On souhaite maintenant minimiser les écarts d'utilisation entre employés. Concrètement, on veut que l'écart d'utilisation entre l'employé qui travaille le plus et celui qui travaille le moins soit le plus petit possible. (Mais on souhaite toujours maintenir le système dans un état stable : ne demander à personne de travailler n'est pas une solution acceptable par exemple.)

Pour cela, on peut proposer une nouvelle version d'ALGO, mais on peut aussi changer la politique du choix de l'employé lorsqu'il y en a plusieurs dans la salle d'attente avec la compétence nécessaire. (Dans la version originale, c'est l'employé avec la compétence qui attend le plus longtemps qui est choisi.)

Question 5. *Proposer une nouvelle version d'*ALGO *et/ou de la politique du choix de l'employé dans la salle d'attente pour que la charge de travail soit mieux répartie entre les employés lorsque cela est possible.*

Question 6. *Fournir le code de l'algorithme. (À nouveau, il faut que le code puisse être inclus aux endroits prévus dans le simulateur*[2] *sans avoir à modifier d'autres parties du simulateur.) Indiquer les résultats obtenus avec des intervalles de confiance et commenter.*

Question 7. *Discuter le choix de formalisation du critère d'équité proposé ci-dessus.*

2. Pour la gestion de la salle d'attente, c'est dans la méthode `worker_available` de la `class System`.