Centre universitaire Abedelhafid Boussouf – Mila
Institut des sciences et technologie
Département de mathématiques et informatique
1er année Master : Intelligence artificielle et ses applications – Année: 2023-2024
Matière : Modélisation et Simulation
Responsable de la matière : Dr. Sadek Benhammada

Mini Projet : Développement d'un Simulateur de File d'Attente pour le Service d'Urgence Hospitalier

Remise et consultation des projets :
La remise et la consultation du travail est prévue aprés les vacances de l'hiver.
Attention :
Toute ressemblance entre deux projets fera diviser la note par deux.
Tout étudiant qui ne saura pas expliquer son code lors de la consultation aura une note de zéro.

I. Introduction

Le service d'urgence hospitalier est souvent confronté à des défis liés à la gestion des files d'attente de patients. Un simulateur permettant d'analyser et d'optimiser ces files d'attente peut apporter des solutions significatives pour améliorer l'efficacité du service, réduire les temps d'attente et optimiser les ressources.

L'objectif principal de ce mini-projet est de développer un simulateur de file d'attente pour le service d'urgence d'un hôpital. Le simulateur à développer inclura des fonctionnalités permettant d'analyser les performances de la file d'attente des patients, et d'identifier des stratégies d'optimisation pour améliorer la gestion de la file d'attente.

II. Travail demandé

Dans le système de simulation de file d'attente du service des urgences de l'hôpital :

- Les patients sont les **patients**
- Les serveurs sont les **salles de soins** (pour une simplification du système)

1. Etape 1 : Simulateur M/M/S

La première étape consiste développer et valider un simulateur de file d'attente M/M/S qui servira de point de départ pour la réalisation du simulateur :

1.1. Fonction simulate_mms()

Créez une fonction nommée **`simulate_mms(lam, mu, s, end_time)`** qui simule une file d'attente de type M/M/S. Cette fonction prend en paramètres :

- **`lam`** : le taux d'arrivée des patients (nombre moyen de patients qui arrivent par heure),
- **`mu`** : le taux de service (nombre moyen de patients servis dans une salle de soins par heure),
- **`end_time`** : la durée de la simulation,
- **`s`** : le nombre de salles de soins.

La fonction effectue les calculs suivants et les renvoie :

- **`a`** : Taux d'utilisation des serveurs,
- **`P0:`** Probabilité que le système soit vide (aucun patients dans le système)
- **`Pw`** Probabilité d'attente d'un patient (Erlang_C),
- **`N`** : Nombre moyen de patients dans le système,

- **Nq** : Nombre moyen de patients en attente,
- **Ns** : Nombre moyen de patients en service (Nombre moyen de serveurs occupés),
- **T** : Temps moyen qu'un patient passe dans le système (attente + service),
- **Tq** : Temps moyen d'attente d'un patient,

Pour valider le simulateur M/M/S, comparer ses résultats avec ceux du modèle M/M/S analytique :

Données de test			**Résultats attendus**								
lam	**mu**	**s**	**a**	**P0**	**Pw**	**N**	**Nq**	**Ns**	**T**	**Tq**	**Ts**
10	4	3	0.833	0.045	0.702	6.011	3.511	2.5	0.601	0.351	0.25
60	18	4	0.833	0.021	0.658	6.622	3.289	3.333	0.11	0.055	0.056
30	20	2	0.75	0.143	0.643	3.429	1.929	1.5	0.114	0.064	0.05
60	25	2	1.2	Le système n'est pas stationnaire : a = 1.2 >= 1							
16	4	4	1.0	Le système n'est pas stationnaire : a = 1.0 >= 1							

Vous pouver faire plus de tests de comparaison avec le modèle analytique (TP 2 : Exercice 2), et/ou se servir des plateformes en ligne[1].

2. Etape 2: Simulateur M/M/S/K

Le service des urgences dispose d'une capacité limitée, une fois la capacité maximale est atteinte, les patients qui arrivent ne sont pas acceptés.

2.1. Fonction simulate_mmsk()

Créer une fonction **simulate_mmsk (`lam, mu, s, end_time, K`)** pour prendre en compte la capacité limitée du service des urgences (file d'attente M/M/S/K). La fonction **simulate_mmsk()**, prend en parametres :

- **lam** : le taux d'arrivée des patients (nombre moyen de patients qui arrivent par heure),
- **mu** : le taux de service (nombre moyen de patients servis dans une salle de soins par heure),
- **end_time** : la durée de la simulation,
- **K** : la capacité maximale du service des urgences (nombre maximal de patients)
- **s** : le nombre de salles de soins.

La fonction **simulate_mmsk()** donnent en sortie

- **a** : Taux d'utilisation des serveurs,
- **P0 :** Probabilité que le système soit vide (aucun patients dans le système)
- **Pw** Probabilité d'attente d'un patient (Erlang_C),
- **N** : Nombre moyen de patients dans le système,
- **Nq** : Nombre moyen de patients en attente,
- **Ns** : Nombre moyen de patients en service (Nombre moyen de serveurs occupés),
- **T** : Temps moyen qu'un patient passe dans le système (attente + service),
- **Tq** : Temps moyen d'attente d'un patient,
- **P_k** : Probabilité de rejet d'un patient,
- **TabN** : Historique des nombres de patients dans le service des urgences

[1] https://qsa.inf.unideb.hu/prod/frontend/schemes/1 (Sélectionner le modèle M/M/C)

- **TabUpN**: Historique des temps de mise à jour du nombre de patients dans le service des urgences.

Données de test				Résultats attendus								
lam	**mu**	**s**	**k**	**a**	**P0**	**Pw**	**Pk**	**N**	**Nq**	**Ns / c**	**T**	**Tq**
15	5	3	10	0.899	0.0225	0.787	0.101	5.53	2.83	2.7	0.41	0.21
60	18	4	100	0.833	0.0213	0.658	0	6.62	0.0548	3.33	0.11	0.0548
30	4	2	80	1	0	1	0.733	79.6	77.6	2	9.95	9.7
60	25	3	10	0.776	0.063	0.592	0.0305	3.77	1.44	2.33	0.0648	0.0248
5	40	4	50	0.0313	0.882	0	0	0.125	0	0.125	0.025	0

2.2. Fonction plot_perform_servers ()

Créer une fonction **plot_perform_servers(lam, mu, K, end_time, num_max_servers)**, qui prend en paramètres :

- **`lam:`** taux d'arrivée des patients
- **`mu`** : taux de service
- **K** : La capacité maximale du service des urgences
- **`s_min:`** nombre minimal de serveurs
- **`s_max:`** nombre maximal de serveurs
- **`end_time`**: durée de simulation

La fonction **plot_perform_servers ()** calcule et affiche à l'aide de graphiques :

- Le nombre moyen de patients aux urgences en fonction du temps, pour chaque nombre de serveurs entre **s_min et s_max**
- Les mesures performances du système en fonction du nombre de salles de soins :
  - Un graphique pour les nombres moyens : **`N`**, **`Nq`**, et **`Ns`**
  - Un graphique pour les temps moyens : **`T`** et **`Tq`**
  - Un graphique pour le taux d'utilisations des salles de soins « **a** », la probabilité d'attente « **`Pw`** », et la probabilité de rejet d'un patient **`Pk`**.

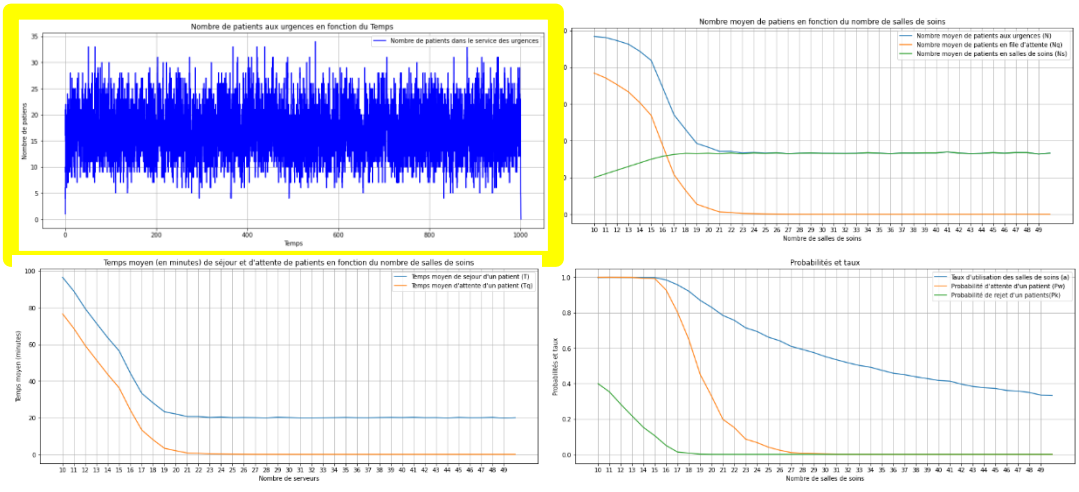

Figure 1. Exemple de graphiques affiches par la fonctions ***plot_perform_servers ()***

2.3. Fonction plot_perform_capacity ()

Créer une fonction **plot_perform_ capacity (lam, mu, K, end_time, num_max_servers)**, qui prend en paramètres :

- **`lam:`** taux d'arrivée des patients
- **`mu`** : taux de service
- **K** : La capacité maximale du service des urgences
- **`K_min:`** Valeur minimale de ***K*** (Capacité maximale du système)
- **`K_max:`** Valeur maximale de ***K***
- **`end_time`**: durée de simulation

La fonction **plot_perform_capacity ()** calcule et affiche à l'aide de graphiques :

- Le nombre moyen de patients aux urgences en fonction du temps, pour chaque valeur de **K** dans [**`K_min , K_max`**]
- Les mesures performances du système en fonction de la capacité du système **K** :
  - Un graphique pour les nombres moyens : **`N`**, **`Nq`**, et **`Ns`**
  - Un graphique pour les temps moyens : **`T`** et **`Tq`**
  - Un graphique pour le taux d'utilisations des salles de soins « **a** », la probabilité d'attente « **`Pw`** », et la probabilité de rejet d'un patient **`Pk`**.

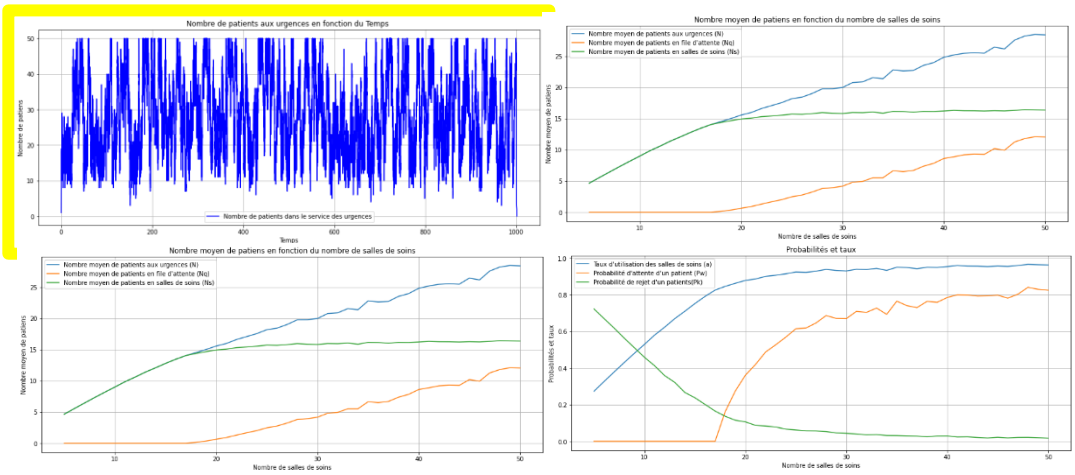

Figure 2. Exemple de graphiques affiches par la fonctions ***plot_perform_capacity ()***

2.4. Programme principal

Ecrire le programme principal qui donne à l'utilisateur le choix entre :

(1) Exécuter une simulation,

(2) Afficher les performances en fonction des nombres de salles de soins

(3) Afficher les performances en fonction des capacités des urgences

3. Etape 3 : Assignation de priorités aux patients

Les patients qui se présentent au service des urgences sont affectés à l'une des deux files d'attente en fonction de la gravité de leur état :

(1) La file d'attente des patients prioritaires
(2) La file d'attente des patients non prioritaires.

Les patients non prioritaires ne pourront accéder aux soins que +lorsque tous les patients prioritaires seront pris en charge.

Sachant que 10% des patients qui arrivent aux urgences sont prioritaires, modifiez le code précédent afin d'intégrer la prise en compte de l'assignation de priorité aux patients. Le simulateur doit calculer et visualiser graphiquement les performances distinctes des deux files d'attente, à savoir celle des patients prioritaires et celle des patients non prioritaires.

4. Etape 4 : Ajustement de données

Des données de temps inter-arrivée de patients, et de durées de service sont collectées sur un service des urgences. Dans l'objectif de modéliser de manière précise les temps inter-arrivée et les durées de service des patients :

1. Utiliser l'environnement R pour ajuster les distributions pour les deux processus. Cela implique la sélection d'une loi théorique appropriée et l'estimation des paramètres associés.
2. Modification du code précédent afin de générer des temps d'arrivée et de service en se basant sur les résultats de l'ajustement des distributions théoriques obtenues dans l'étape précédente.