Université Abedelhafid Boussouf – Mila
Département d'Informatique
1ere année Master : Intelligence Artificielle et ses Applications
Année académique : 2025-2026
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 des urgences hospitalières est confronté à des difficultés dans la gestion des files d'attente des patients. Ce mini-projet vise à développer un simulateur permettant d'analyser les performances du système et d'optimiser la gestion des files d'attente afin de réduire les temps d'attente et d'améliorer l'efficacité du service.

II. Travail demandé

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

- Les clients sont les **patients**
- Les serveurs sont les **salles de soins**

1. Etape 1 : Simulateur M/M/S

La première étape consiste à développer et valider un simulateur de file d'attente M/M/s qui servira de base 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 les paramètres suivants :

- **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 calcule et retourne les indicateurs suivants :

- **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,

1.2. Validation

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 pouvez 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

2.1. Fonction simulate_mmsk()

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

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.

2.2. Validation

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

[1] https://qsa.inf.unideb.hu/prod/frontend/schemes/1

2.3. Programme principal

Écrire un programme principal qui permet à l'utilisateur :

1. Choisir le type de système (**M/M/s** ou **M/M/s/K**).
2. Saisir les paramètres :
   - Taux d'arrivée (λ),
   - Taux de service (μ),
   - Nombre de salles de soins (s),
   - Capacité maximale du système (K) **uniquement pour le système M/M/s/K**,
   - Durée de la simulation.
3. Afficher les indicateurs de performance du système simulé.

3. Etape 3 : 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.