ÉCOLE NATIONALE SUPÉRIEURE D'ARTS ET MÉTIERS
UNIVERSITÉ HASSAN II DE CASABLANCA

UNIVERSITÉ HASSAN II DE CASABLANCA

1ère année du cycle d'ingénieur

Intelligence Artificielle & Génie Informatique

Module : Recherche Opérationnelle

Rapport du mini-projet

Optimisation Des Ressources Pour Un Système De Chatbot Basé Sur IA

ID Projet : 193 **et ID Groupe :** IAGI-01

Réalisé par : EL MOUAFIK Fatima-Ezzahra **&** IDYOUSS Hafsa

Encadré Par : Pr. Abdessamad KAMOUSS

2024-2025

Table des matières

Résumé

Ce projet porte sur **l'optimisation des ressources pour un chatbot basé sur l'intelligence artificielle (IA)**, en appliquant les principes de la recherche opérationnelle (RO). Les chatbots, devenus essentiels dans divers secteurs comme le service client, l'e-commerce, et les services publics, présentent des défis liés à la gestion efficace de leurs ressources matérielles et logicielles. Ces ressources incluent notamment **les CPU, GPU, RAM et stockage**, dont une utilisation optimale est nécessaire pour garantir des performances élevées tout en minimisant les coûts opérationnels.

En utilisant **la programmation linéaire** comme méthode de modélisation, le projet vise à équilibrer les contraintes économiques et techniques en identifiant les configurations idéales pour les infrastructures utilisées par les chatbots. La méthode repose sur **une fonction multi-objectifs combinant la minimisation des coûts et la maximisation des performances**. À travers l'implémentation avec **Python (PuLP) et l'outil Solver d'Excel**, le modèle explore différentes allocations de ressources pour proposer une solution adaptée aux exigences d'un système complexe comme un chatbot IA. Ce projet ne se limite pas à une simple analyse technique, mais inclut également une réflexion sur les défis économiques associés à ces systèmes.

Les résultats montrent une configuration optimale qui permet de répondre aux besoins des utilisateurs avec une rapidité, tout en réduisant les coûts. Cependant, l'analyse critique des résultats souligne la nécessité d'un ajustement selon les besoins spécifiques et les contraintes des systèmes réels, ainsi qu'une attention particulière à la flexibilité des solutions proposées.

Abstract

This project focuses on **optimizing resources for an artificial intelligence (AI)-based chatbot** by applying the principles of operations research (OR). Chatbots, which have become essential in various sectors such as customer service, e-commerce, and public services, face challenges related to the efficient management of their hardware and software resources. These resources include **CPUs, GPUs, RAM, and storage**, which require optimal utilization to ensure high performance while minimizing operational costs.

By using **linear programming** as the modeling method, the project aims to balance economic and technical constraints by identifying the ideal configurations for the infrastructures used by chatbots. The method relies on **a multi-objective function combining cost minimization and performance maximization**. Through implementation with **Python (PuLP) and Excel Solver**, the model explores different resource allocations to propose a solution tailored to the requirements of a complex system like an AI chatbot. This project goes beyond a mere technical analysis by also addressing the economic challenges associated with such systems.

The results demonstrate an optimal configuration that meets user needs efficiently while reducing costs. However, critical analysis of the results highlights the need for adjustments based on specific requirements and real-world system constraints, as well as the importance of flexibility in the proposed solutions.

Liste des figures

Liste des tableaux

Introduction Générale

La recherche opérationnelle (RO) constitue une discipline essentielle en ingénierie, permettant d'améliorer la prise de décision et l'utilisation efficace des ressources. Elle repose sur des modèles mathématiques, des algorithmes, et des outils de simulation pour optimiser des systèmes complexes. Dans un contexte où les entreprises et les institutions cherchent à maximiser leur efficacité tout en minimisant les coûts, la RO s'avère être un levier stratégique pour concevoir et améliorer des systèmes technologiques. Elle permet notamment de modéliser des contraintes, de gérer les dépenses en ressources et de garantir une performance optimale.

Intelligence artificielle (IA) est une branche de l'informatique qui vise à développer des systèmes capables de reproduire ou de simuler des comportements intelligents humains. Elle permet d'effectuer des tâches comme le raisonnement, la résolution de problèmes, l'apprentissage, ou la compréhension du langage naturel. Ces systèmes s'appuient sur des algorithmes avancés, des modèles mathématiques et de grandes quantités de données pour offrir des solutions efficaces dans des domaines variés, allant de la santé à la robotique en passant par les services numériques.

Les chatbots sont des programmes informatiques capables de simuler une conversation humaine, apportant des réponses intelligentes et adaptées aux requêtes des utilisateurs. Leur utilisation s'étend sur des domaines variés, allant du service client à l'éducation en passant par les services publics. Cependant, à mesure que les chatbots deviennent plus complexes et plus demandés, la gestion efficace de leurs ressources — qu'il s'agisse de la puissance de calcul, de la mémoire ou des données — devient un enjeu crucial.

Durant ce mini-projet, nous allons d'abord identifier les ressources critiques dont dépend un chatbot, telles que la vitesse de réponse, la performance globale, et la consommation en mémoire. En nous appuyant sur des données spécifiques, nous explorerons différentes valeurs de ces ressources. Une modélisation détaillée par programmation linéaire sera réalisée pour formaliser cette problématique et établir des relations optimales entre les contraintes et les objectifs. Ensuite, nous implémenterons cette modélisation à l'aide de Python, permettant de résoudre le problème et de déterminer les valeurs idéales pour chaque ressource. Les résultats attendus incluent une réduction significative des coûts tout en maintenant une grande rapidité et une efficacité des performances.

Notre rapport contient les chapitres suivants :

1er Chapitre : Contexte et Problématique. Ce chapitre introduit le contexte général du projet, en présentant les concepts clés de l'intelligence artificielle et des chatbots. Il met en lumière les défis liés à leur efficacité et à leur coût, ainsi que les indicateurs de performance (CPU, GPU, RAM, etc.), essentiels pour évaluer et optimiser ces systèmes.

2eme Chapitre : Modélisation. Dans ce chapitre, la méthodologie adoptée pour résoudre la problématique est développée, et la programmation linéaire est choisie comme méthode principale. Ainsi, les variables de décision, les contraintes (positivité, seuils minimaux et maximaux, relations RAM-CPU/GPU) et la fonction multi-objectifs sont définies.

3eme Chapitre : Résolution et Analyse des Résultats. Ce chapitre présente les approches utilisées pour résoudre le modèle de programmation linéaire, comme la méthode du simplexe, Solver d'Excel, et PuLP en Python. Les résultats montrent une configuration optimale des ressources équilibrant coût et performance, accompagnée d'une analyse critique des limites et ajustements possibles pour des applications réelles.

Chapitre I :

Présentation du contexte du mini-projet

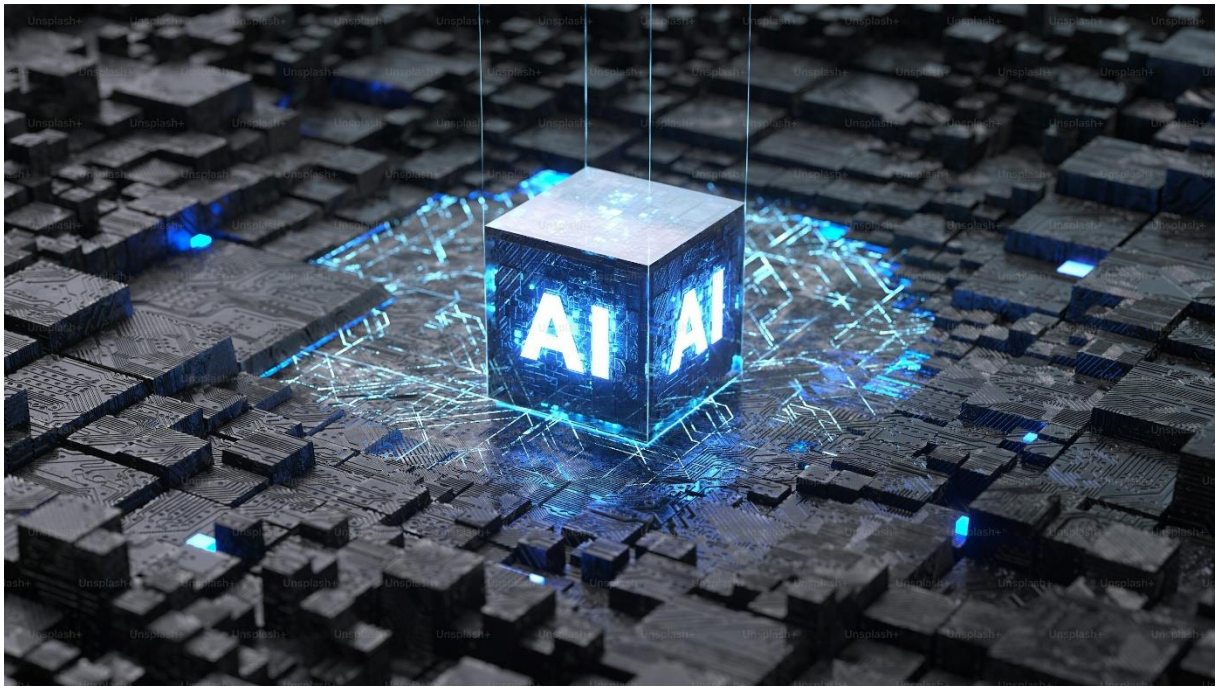


Figure 1 : Illustration du domaine de l'intelligence artificielle

Introduction

Ce chapitre introduit le contexte général dans lequel s'inscrit ce mini-projet, en mettant en lumière les enjeux technologiques et économiques liés aux chatbots basés sur l'IA. Il détaille également les problématiques rencontrées dans ce domaine, ainsi que l'importance de la recherche opérationnelle pour y répondre.

1.Présentation du contexte

Les chatbots, depuis leur émergence, ont transformé la manière dont les entreprises interagissent avec leurs clients. Ils jouent un rôle clé dans l'automatisation des services, permettant une disponibilité 24/7 et une gestion efficace des requêtes des utilisateurs. Selon une étude récente, le marché mondial des chatbots devrait atteindre une valeur de 10,5 milliards de dollars d'ici 2026, avec un taux de croissance annuel composé (CAGR) de 23 %.

Cependant, ces systèmes reposent sur des infrastructures techniques exigeantes, avec des besoins croissants en termes de puissance de calcul, de bande passante, et de stockage. Ces exigences entraînent des coûts élevés, notamment en énergie, et posent des questions sur la durabilité environnementale de ces solutions. Par exemple, un chatbot utilisé à grande échelle peut consommer plusieurs milliers de kilowattheures par an pour maintenir ses performances.

1.1. Processus de fonctionnement :

Les chatbots alimentés par l'intelligence artificielle (IA) sont des programmes capables de simuler une conversation humaine à travers des interfaces textuelles ou vocales. Leur fonctionnement repose principalement sur la compréhension du langage naturel (NLP) et l'apprentissage automatique. Lorsqu'un utilisateur soumet une requête, le chatbot utilise ces technologies pour analyser et comprendre la demande avant de générer une réponse appropriée. Ce processus d'interaction peut être amélioré par l'intégration continue de nouvelles données, permettant au chatbot de répondre de manière de plus en plus précise aux demandes des utilisateurs. Ce système est appliqué dans divers domaines, tels que le service client, le e-commerce et même la gestion des ressources humaines.

1.2. Intérêt et les enjeux économiques :

Les chatbots IA permettent aux entreprises de réduire les coûts liés au support client et aux services en automatisant les réponses à des demandes fréquentes, tout en offrant une disponibilité 24/7. Cette technologie aide également à augmenter l'efficacité opérationnelle en permettant de traiter un grand nombre de demandes simultanément. Les enjeux économiques résident dans l'automatisation de processus récurrents, la réduction des coûts de main-d'œuvre, et l'amélioration de l'expérience client. De plus, avec l'adoption croissante des chatbots dans des secteurs tels que la vente en ligne et le service à la clientèle, les entreprises voient une opportunité pour optimiser leurs ressources tout en offrant une interaction de qualité.

1.3. Quelques chiffres clés :

En 2022, la taille du marché mondial des chatbots était estimée à près de 44 milliards de dollars et devrait atteindre 75 milliards de dollars d'ici 2030, avec un taux de croissance annuel d'environ 13,2 % sur la période de prévision (2023-2030).

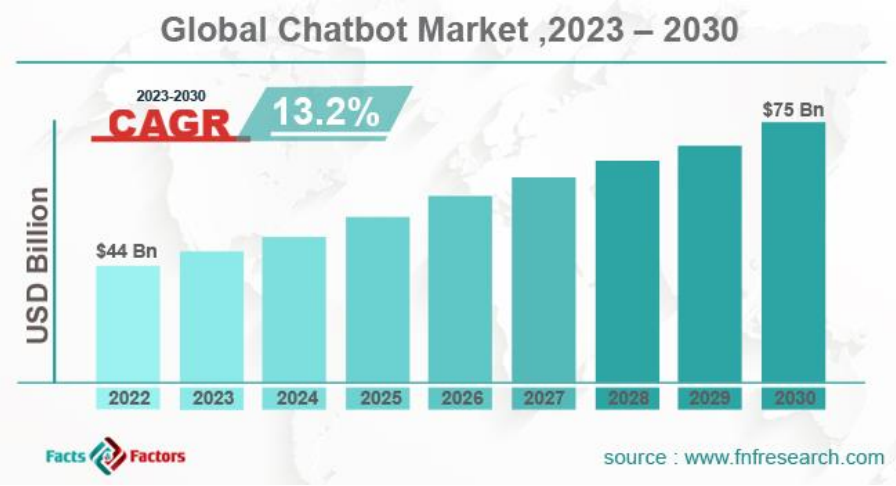


Figure 2 : Graphique du marché mondial des chatbots 2023 – 2030

En 2023, 88 % des consommateurs ont interagi avec au moins un chatbot, et 62 % préfèrent utiliser un chatbot plutôt que d'attendre un agent du service client humain.

Des experts prévoient que le marché des chatbots IA continuera de croître, avec une adoption accrue dans divers secteurs tels que le commerce de détail, la santé et les services financiers.

1.4. Quelques problématiques :

Malgré leur efficacité, les chatbots IA présentent certaines limitations. L'un des défis majeurs est leur capacité à comprendre le contexte et les nuances des conversations humaines. De plus, les chatbots peuvent rencontrer des difficultés face à des requêtes complexes ou des demandes non prévues, nécessitant parfois une intervention humaine. Un autre obstacle est l'intégration des chatbots avec les systèmes existants, qui peut être complexe et coûteuse pour les entreprises. Enfin, bien que les chatbots soient de plus en plus sophistiqués, la gestion des attentes des utilisateurs reste une problématique importante.

1.5. Intérêt de RO dans ce domaine :

La recherche opérationnelle (RO) est un outil puissant pour l'optimisation des systèmes de chatbots basés sur l'IA. En appliquant des techniques de modélisation et d'optimisation, la RO permet de résoudre des problèmes complexes tels que la gestion des ressources informatiques, l'amélioration de l'efficacité du traitement des demandes et l'optimisation des coûts d'exploitation des chatbots. Elle aide également à concevoir des stratégies permettant d'améliorer la gestion des volumes élevés de demandes en période de forte affluence, tout en maximisant la satisfaction client.

2.Présentation de la problématique

L'optimisation des ressources pour un système de chatbot basé sur l'intelligence artificielle (IA) constitue un défi majeur pour les entreprises souhaitant offrir une interaction fluide, rapide et efficace avec leurs utilisateurs tout en minimisant les coûts opérationnels. En effet, le succès de ces systèmes repose non seulement sur leur capacité à comprendre et à répondre aux requêtes des utilisateurs, mais aussi sur l'utilisation optimale des ressources matérielles et logicielles. Les chatbots IA, bien qu'efficaces, exigent des infrastructures puissantes et évolutives pour traiter les demandes en temps réel, ce qui soulève des questions sur la gestion des ressources serveur, la répartition des charges de travail et l'optimisation des algorithmes d'IA.

L'un des principaux défis est l'optimisation des performances tout en maintenant des coûts raisonnables. Par exemple, l'utilisation de processeurs hautes performances, comme ceux proposés par NVIDIA avec ses GPU A100, peut offrir des gains considérables en termes de vitesse de traitement et d'efficacité énergétique, mais ces technologies représentent également un investissement significatif. De plus, l'adaptation du chatbot à une demande en constante évolution nécessite une gestion dynamique des ressources afin de garantir une expérience utilisateur optimale. La mise en place de telles infrastructures doit donc être accompagnée d'une analyse approfondie des coûts, des bénéfices et des contraintes logistiques.

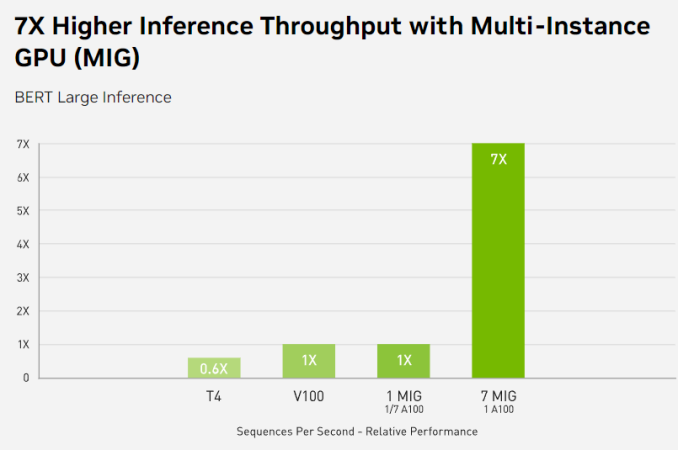


Figure 3 : Performances d'Inférence avec MIG (GPU A100)

Pour relever ce défi, il est impératif de trouver un équilibre entre performance et maîtrise des ressources. Les stratégies d'optimisation doivent inclure des solutions innovantes, telles que l'utilisation de modèles d'IA allégés pour réduire la complexité du traitement, ou encore l'adoption d'infrastructures cloud scalables pour ajuster les ressources en fonction des besoins.

Ainsi, la problématique centrale de ce projet est de répondre à cette question complexe : **Comment peut-on optimiser l'utilisation des ressources dans un système de chatbot basé sur l'intelligence artificielle, de manière à réduire les coûts de traitement tout en maintenant une rapidité de réponse qui répondent aux attentes des utilisateurs ?**

3.Les indicateurs clés de performance d'un chatbot (KPIs)

Les premiers chatbots étaient essentiellement des programmes de FAQ interactifs, qui s'appuyaient sur un ensemble limité de questions courantes avec des réponses pré-écrites. Au fil du temps, les algorithmes de chatbot sont devenus capables d'une programmation plus complexe basée sur des règles et même d'un traitement automatique du langage naturel, permettant aux clients d'exprimer leurs requêtes sous forme de conversation. Cela a donné naissance à un nouveau type de chatbot qui s'adapte au contexte et qui utilise le machine learning pour optimiser en permanence sa capacité à traiter et à prévoir correctement les requêtes.

Afin d'évaluer l'efficacité d'un chatbot. On a un ensemble d'indicateurs clés de performance (KPIs) essentiels est utilisé pour analyser et optimiser sa performance.

- **CPU (Central Processing Unit) :** est un composant matériel qui constitue l'unité centrale de traitement d'un serveur. Il gère toutes les tâches informatiques nécessaires comme le calcul, contrôle des périphériques et l'exécution des tâches au fonctionnement du système d'exploitation et des applications.
- **GPU (Graphics Processing Unit) :** un circuit électronique conçu pour accélérer l'affichage et le traitement d'images sur divers appareils. Il traite plus efficacement qu'un CPU général les opérations mathématiques complexes grâce à son architecture parallèle qu'il le permet de traiter

simultanément de nombreuses opérations similaires comme l'apprentissage automatique et les simulations complexes.

- **RAM (Random Access Memory) :** est la mémoire vive qui sert à stocker des données de façon temporaire quand elles sont utilisées par le processeur pour exécuter des programmes. Lorsque l'ordinateur est éteint, les données contenues dans la RAM sont perdues
- **Stockage (Cloud) :** est un service qui permet de stocker des données en les transférant via Internet ou via un autre réseau vers un système de stockage hors site géré par une tierce partie. Il existe des centaines de systèmes de stockage cloud : du stockage personnel destiné à accueillir des fichiers privés d'une personne, au stockage professionnel que les entreprises utilisent comme solution de sauvegarde à distance où elles peuvent transférer, stocker et partager en toute sécurité des fichiers de données.
- **Vitesse :** se réfère au temps de réponse qu'il met pour traiter une requête utilisateur et fournir une réponse appropriée. Cette rapidité dépend de plusieurs facteurs, notamment la performance des CPU et GPU utilisés, l'efficacité des algorithmes d'intelligence artificielle, et l'optimisation de l'infrastructure serveur.

Conclusion

Après avoir mis l'accent sur le fonctionnement du chatbot, ses enjeux économiques, ainsi que les problématiques rencontrées dans sa gestion, nous avons également exploré l'importance de la recherche opérationnelle pour optimiser ses performances. En analysant les indicateurs clés de performance, nous avons souligné les défis liés à son efficacité. Ainsi, après avoir posé les bases essentielles de ce projet, nous nous orientons désormais vers la modélisation afin de définir des solutions concrètes et adaptées.

Chapitre II :

Modélisation de la problématique

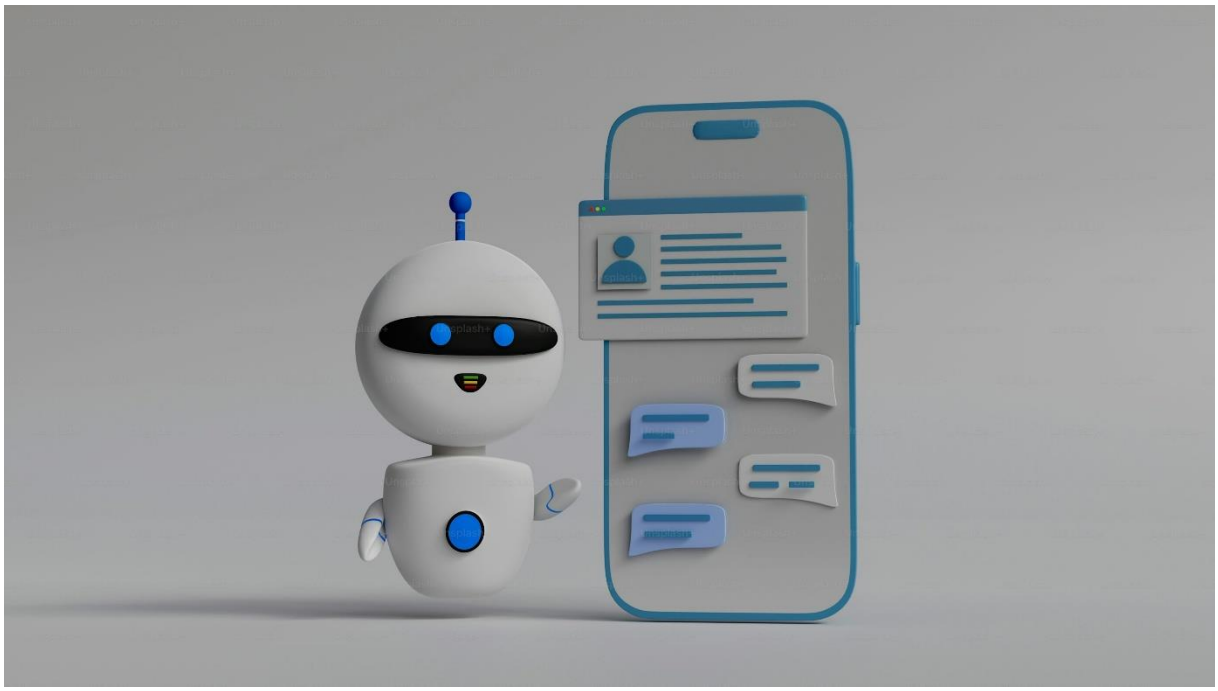


Figure 4 : Illustration d'un chatbot

Introduction

Dans ce chapitre, nous allons aborder la phase de modélisation du projet, qui consiste à définir et formaliser les éléments clés nécessaires pour optimiser les performances du chatbot. Cette étape inclut le choix de la modélisation appropriée, la définition des variables de décision, l'identification des contraintes, ainsi que la formulation de la fonction objectif. Nous verrons comment ces éléments interagissent pour créer un modèle opérationnel permettant de répondre aux enjeux identifiés dans le chapitre précédent.

1.Choix de la modélisation appropriée : Programmation linéaire

Le choix de la programmation linéaire comme méthode de modélisation pour notre problématique est motivé par sa capacité à optimiser efficacement des problèmes impliquant des contraintes linéaires et une fonction objectif claire. Dans le cadre de l'optimisation des ressources pour un chatbot, cette approche permet de modéliser de manière précise et systématique la relation entre les différentes variables, telles que les coûts de traitement, la vitesse de réponse, et les ressources nécessaires (CPU, GPU, serveurs). La programmation linéaire offre également l'avantage de fournir des solutions optimales dans des situations complexes, où plusieurs contraintes doivent être respectées simultanément. En intégrant cette méthode, nous pourrons déterminer les meilleures configurations pour minimiser les coûts tout en maximisant les performances du chatbot, ce qui répond directement à la problématique soulevée dans le premier chapitre.

La modélisation du problème d'allocation des ressources en maintenant une réponse rapide implique deux objectifs principaux :

• **Minimiser les coûts** associés à l'utilisation des ressources

• **Maximiser la performance** du chatbot en termes de requêtes traitées par seconde.

$$\begin{cases} \text{Maximiser ou Minimiser } z(x_1,x_2,\ldots,x_n) = \sum_{j=1}^{n} c_j x_j \\ \text{Sous les contraintes} \begin{cases} \sum_{i=1}^{n} a_{1i}x_i \leq,=,\geq b_1 \\ \sum_{i=1}^{n} a_{2i}x_i \leq,=,\geq b_2 \\ \ldots\ldots\ldots\ldots\ldots\ldots \\ \ldots\ldots\ldots\ldots\ldots\ldots \\ \sum_{i=1}^{n} a_{mi}x_i \leq,=,\geq b_m \\ x_1,x_2,\ldots,x_n \in \mathbb{R} \end{cases} \end{cases}$$

Figure 5 : Formule générale d'un programme linéaire

2.Variables de décisions

- x_1 : Nombre de **CPU** alloués **(en unités de CPU)**
- x_2 : Nombre de **GPU** alloués **(en unités de GPU)**
- x_3 : Quantité de **RAM** allouée **(en Go de RAM)**
- x_4 : Quantité de **Stockage** allouée **(en Go de stockage)**

3.Contraintes

3.1. Contraintes de positivité :

Les ressources allouées doivent être strictement positives, ce qui signifie que chaque variable de décision doit être non négatives :

$$x_1 \geq 0 \quad , \quad x_2 \geq 0 \quad , \quad x_3 \geq 0 \quad , \quad x_4 \geq 0$$

3.2. Contraintes minimales :

Pour assurer le fonctionnement du système, certaines ressources doivent avoir une allocation minimale même à faible charge :

$$x_1 \geq CPU_{min} \quad , \quad x_2 \geq GPU_{min} \, , \quad x_3 \geq RAM_{min} \, , \quad x_4 \geq Stockage_{min}$$

CPU_{min}, GPUmin ,RAM_{min} ,$Stockage_{min}$ sont les valeurs minimales respectives pour chaque ressource.

3.3. Contraintes maximales :

Certaines ressources ne doivent pas dépasser un certain seuil pour éviter une sur-allocation. Ces limites maximales peuvent être définies en fonction des capacités du serveur ou de l'infrastructure :

$$x_1 \leq CPU_{max} \quad , \quad x_2 \leq GPU_{max} \quad , \quad x_3 \leq RAM_{max} \quad , \quad x_4 \leq Stockage_{max}$$

CPU_{max} ,GPU_{max} ,RAM_{max} ,$Stockage_{max}$ sont les valeurs maximales respectives pour chaque ressource.

3.3. Relation RAM-CPU :

Chaque unité de CPU nécessite une quantité minimale de RAM : $x_3 \geq k_1 \cdot x_1$

k_1 est la quantité minimale de RAM (en Go) requise par unité de CPU.

3.4. Relation RAM-GPU :

Chaque unité de GPU nécessite une quantité minimale de RAM : $x_3 \geq k_2 \cdot x_2$

k_2 est la quantité minimale de RAM (en Go) requise par unité de GPU.

3.5. Tableaux des valeurs :

Pour réaliser l'étude, nous allons utiliser des **valeurs approximatives**, basées sur des estimations réalistes sur des plateformes de référence comme **AWS** et **Google Cloud**. Ces valeurs sont des approximations basées sur des configurations couramment utilisées pour les systèmes déployant des Chatbots sur des serveurs cloud. Les contraintes avec des estimations sont alors citées dans les tableaux ci-dessous:

Variable	Valeur Minimale	Valeur Maximale
Nombre de CPU alloués	16	32
Nombre de GPU alloués	04	08
Quantité de RAM allouée (en Go)	64	256
Quantité de Stockage allouée (en Go)	100	500

Tableau 1 : Estimation des valeurs minimales et maximales

Variables	Relation	Description
CPU-RAM	$x_3 \geq 4 \cdot x_1$	Chaque unité de CPU nécessite au moins 4Go de RAM
GPU-RAM	$x_3 \geq 8 \cdot x_2$	Chaque unité de GPU nécessite au moins 8Go de RAM

Tableau 2 : Estimation des coefficients RAM-CPU et RAM-GPU

Ces contraintes assurent que les unités CPU et GPU ont suffisamment de mémoire pour fonctionner efficacement. Les applications modernes, notamment les modèles NLP (Natural Language Processing), nécessitent souvent beaucoup de mémoire RAM pour charger des modèles volumineux et traiter plusieurs requêtes simultanément.

4.Fonction multi-objectifs :

La fonction *multi-objectifs* à optimiser **combine la minimisation des coûts et la maximisation de la performance dans une seule fonction**, en intégrant un facteur de pondération pour chaque objectif. Cette fonction est exprimée comme suit :

$$Z = \alpha \cdot C - \beta \cdot P$$

Avec : $C = c_1 \cdot x_1 + c_2 \cdot x_2 + c_3 \cdot x_3 + c_4 \cdot x_4$ en USD/seconde

C	**Le coût total** à minimiser	**Valeur estimée**
c_1	Coût par **CPU** (en USD/secondes par unité de CPU)	0,046 USD
c_2	Coût par **GPU** (en USD/secondes par unité de GPU)	1,5 USD
c_3	Coût par Go de **RAM** (en USD/secondes par Go de RAM)	0,0055 USD
c_4	Coût par Go de **stockage** (en USD/secondes par Go de stockage)	0,0001 USD

Tableau 3 : Estimation des coûts (USD/seconde)

Et : $P = v_1 \cdot x_1 + v_2 \cdot x_2$ en requêtes/seconde

P	**La performance totale** à maximiser	**Valeur estimée**
v_1	Performance par **CPU** (en requêtes/secondes par CPU)	100000req/s CPU
v_2	Performance par **GPU** (en requêtes/secondes par GPU)	500req/s GPU

Tableau 4 : Estimation des performances (requêtes/seconde)

Où :

α	Un facteur de pondération pour **le coût** (secondes/USD)
β	Un facteur de pondération pour **la performance** (secondes/requêtes)

Tableau 5 : Définition des facteurs

L'objectif est de **minimiser Z**, ce qui revient à réduire les coûts tout en augmentant la performance.

Les valeurs utilisées pour les coûts et les performances des ressources dans notre modèle proviennent de benchmarks et de données publiques disponibles sur les services cloud, tels qu'**AWS EC2**, ainsi que sur les spécifications matérielles des composants modernes.

5.Modélisation complète

En posant **α = β = 0.5** (on peut les modifier selon la priorité de C et P). On obtient le modèle suivant :

$$\begin{cases} [Min]\ Z = \ 0.00275 * x3 + 0.00005 * x4 - 50000 * x1 - 249.25 * x2 \\ \begin{cases} x1 \leq 32 \\ x2 \leq 8 \\ x3 \leq 256 \\ x4 \leq 500 \\ x1 \geq 16 \\ x2 \geq 4 \\ x3 \geq 64 \\ x4 \geq 100 \\ x3 \geq 4.x1 \\ x3 \geq 8.x2 \end{cases} \\ x1 \ \geq \ 0, x2 \ \geq \ 0, x3 \ \geq \ 0, x4 \ \geq \ 0 \end{cases}$$

Conclusion

En conclusion, ce chapitre a permis de poser les bases solides pour la modélisation du problème d'optimisation des ressources dans un système de chatbot basé sur IA, en adoptant une approche de programmation linéaire. Cette modélisation offre une vision claire et structurée de la problématique, préparant ainsi le terrain pour le chapitre suivant, qui se concentrera sur la résolution de ce modèle et l'analyse des résultats obtenus.

Chapitre III :

Résolution du programme linéaire

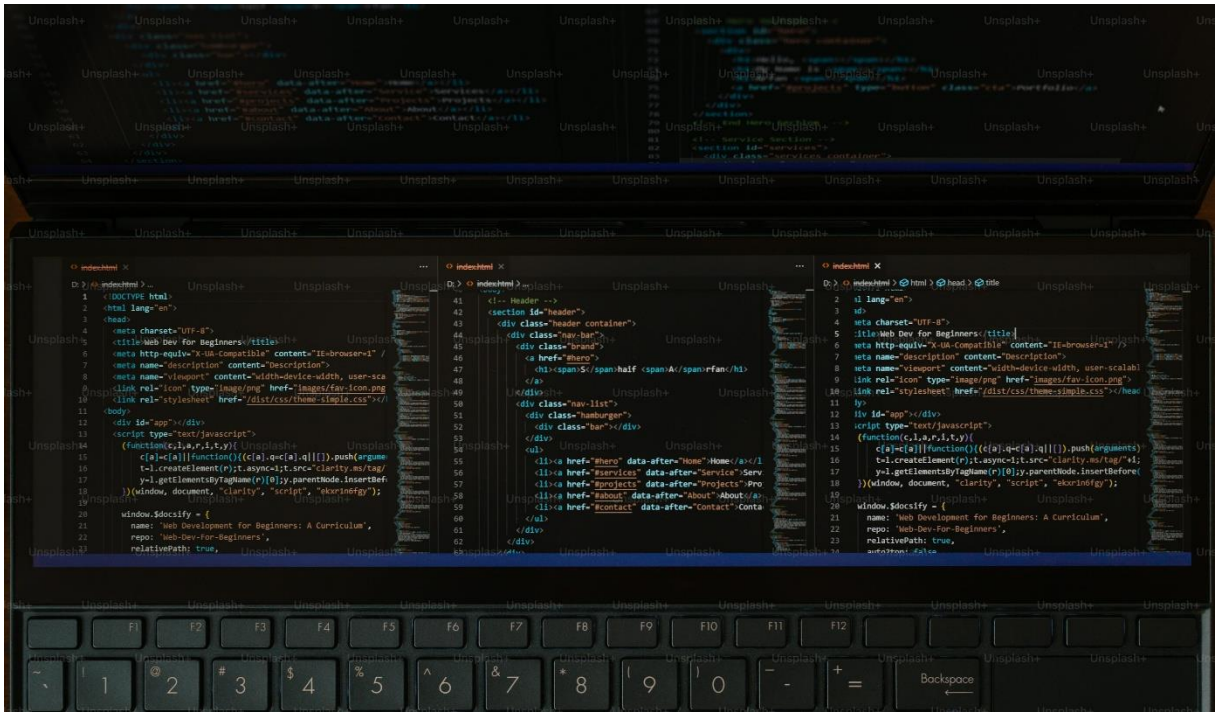


Figure 6 : illustration de programmation avec python

Introduction

Dans ce chapitre, nous allons traiter la phase de résolution de notre problématique en utilisant la méthode du simplexe vue en cours en s'appuiant sur 2 méthodes differentes : La première est la méthodes solveur et la deuxième est un code Python afin de trouver les solutions optimales permettant d'optimiser les coûts de traitement tout en maintenant une grande vitesse pour le chatbot.

1.Outils utilisés

1.1. Fonction Solveur :

Le Solveur est un complément **Microsoft Excel** qui permet de trouver une valeur optimale (maximale ou minimale) pour une formule dans une seule cellule, appelée cellule objectif, en fonction de contraintes appliquées aux valeurs d'autres cellules et un groupe de cellules, appelées variables de décisions. Le Solveur affine les valeurs de ces derniers pour satisfaire aux limites appliquées aux cellules de contraintes et produire le résultat souhaité pour la cellule objectif.

- **Etape 1 : Définir les cellules**

Exemple : analyse des ventes de trois produits.

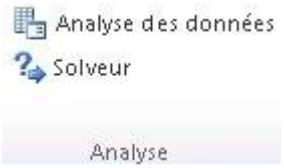
- Colonne B : le nombre d'articles
- Colonne C : le prix des articles
- Colonne D : des formules pour calculer le bénéfice de chaque produit

	A	B	C	D
2		Article	Prix de l'article	Profit
3	Produit A	50	10,99 €	549,50 €
4	Produit B	50	14,99 €	749,50 €
5	Produit C	50	24,99 €	1 249,50 €
6	Total :	150		2 548,50 €

Tableau 6 : Tableau 6 : exemple des cellules de données

Le défi consiste à maximiser les bénéfices totaux, tout en tenant compte de leurs limitations

- **Etape 2 : Ouvrir Solveur**



Sous l'onglet *Données*, dans le groupe *Analyse*, cliquez sur *Solveur*.

- **Etape 3 : Définir les paramètres du Solveur**

On commence par spécifier la cellule objectif dans le champ *Objectif à définir*. (Dans notre exemple, la cellule cible est ***D6***)

Comme l'objectif est de maximiser cette cellule, on choisit l'option *Max*.

On spécifie ensuite les variables dans la zone *Cellules variables*. (Dans notre exemple, c'est la plage B3:B5)

Pour les contraintes, on peut les définir avec la boîte de dialogue (figure 9) qui contient trois parties : une *Référence de cellule*, un *Opérateur* et une valeur de *Contrainte*.

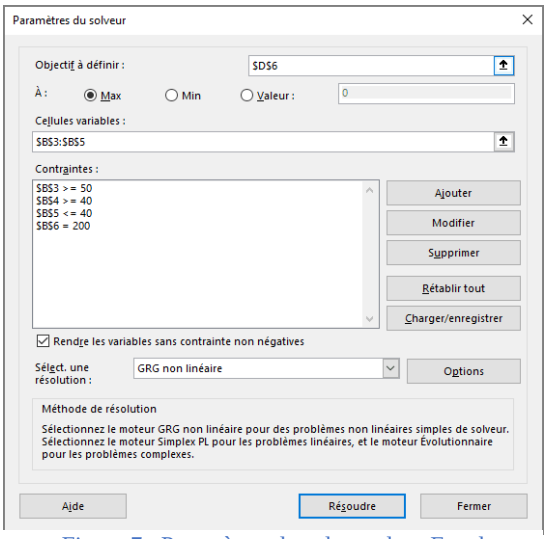


Figure 7 : Paramètres du solveur dans Excel

Pour définir la première contrainte, par exemple la capacité de production totale est de 200 unités, on sélectionne B6 comme référence de cellule, (=) dans la liste déroulante des opérateurs et 200 comme valeur de contrainte.

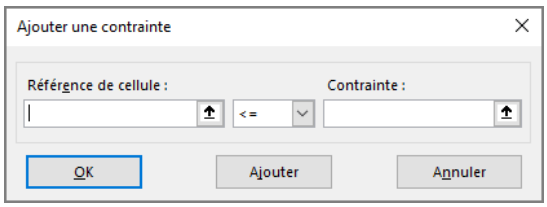


Figure 8 : Définir les contraintes

- **Etape 4 : Les options du Solveur**

On peut choisir n'importe lequel des trois algorithmes ou méthodes de résolution suivante dans la même boîte de dialogue *Paramètres du solveur* :

GRG non linéaire (problèmes non linéaires simples)

Simplex PL (problèmes linéaires)

Evolutionary (problèmes complexes)

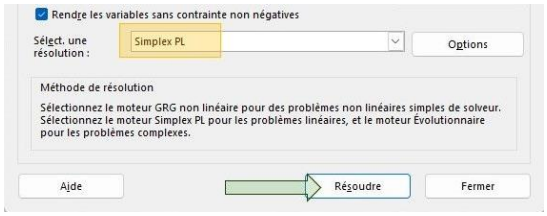


Figure 9 : La méthode de résolution

- **Etape 5 : Résolution**

On clique sur le bouton *Résoudre* pour lancer le processus de solution. Excel annonce bientôt qu'il a trouvé une solution et ouvre la boîte de dialogue *Résultat du solveur*. Dans la première zone, on a deux choix à cocher :

Remplacez les valeurs de cellule d'origine par les valeurs trouvées par Solveur, ou *Restaurez les valeurs* de cellule d'origine.

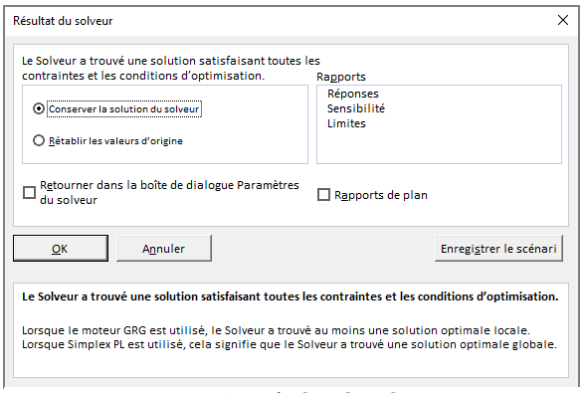
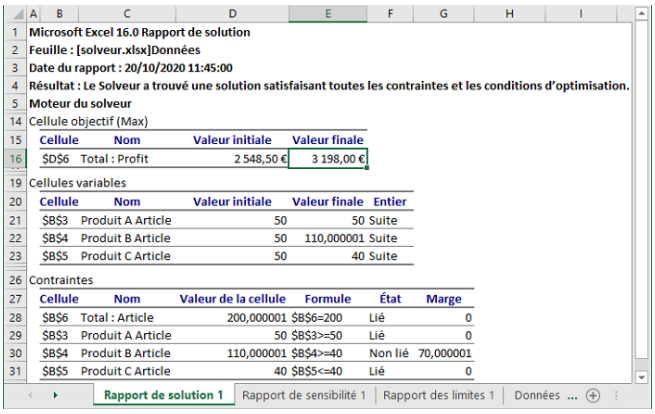


Figure 10 : Résultat du solveur

- **Etape 6 : Rapport des résultats**

Dans la zone *Rapport*, on sélectionne tout ou une partie des trois rapports décrivant ce que *Solveur* a fait. Excel crée chaque rapport sur une nouvelle feuille de calcul, avec un nom approprié.

Pour voir les rapports, on coche la case *Rapports de plan.*



1	Microsoft Excel 16.0 Rapport de solution							
2	Feuille : [solveur.xlsx]Données							
3	Date du rapport : 20/10/2020 11:45:00							
4	Résultat : Le Solveur a trouvé une solution satisfaisant toutes les contraintes et les conditions d'optimisation.							
5	Moteur du solveur							
14	Cellule objectif (Max)							
15	Cellule	Nom	Valeur initiale	Valeur finale				
16	D6	Total : Profit	2 548,50 €	3 198,00 €				
19	Cellules variables							
20	Cellule	Nom	Valeur initiale	Valeur finale	Entier			
21	B3	Produit A Article	50	50	Suite			
22	B4	Produit B Article	50	110,000001	Suite			
23	B5	Produit C Article	50	40	Suite			
26	Contraintes							
27	Cellule	Nom	Valeur de la cellule	Formule	État	Marge		
28	B6	Total : Article	200,000001	B6=200	Lié	0		
29	B3	Produit A Article	50	B3>=50	Lié	0		
30	B4	Produit B Article	110,000001	B4>=40	Non lié	70,000001		
31	B5	Produit C Article	40	B5<=40	Lié	0		

Figure 11 : Rapport des résultats

1.2. Python en utilisant PuLP :

PuLP est une bibliothèque open-source dédiée à la programmation linéaire en Python. Elle offre une gamme complète d'outils pour modéliser des problèmes d'optimisation et les résoudre à l'aide de différents solveurs standards basés sur divers algorithmes de résolution. En tant que wrapper, PuLP facilite la formulation des problèmes directement dans un environnement Python convivial.

On initialise le problème en définissant les variables de décisions pour pouvoir lui ajouter nos contraintes et fonction objectif. Si on veut maximiser cette dernière, on utilise *LpMaximize*, sinon *LpMinimize* pour la minimiser.

```
# Nombre de grosses voitures produites
x = LpVariable('x', lowBound=0, cat=LpInteger)
# Nombre de petites voitures produites
y = LpVariable('y', lowBound=0, cat=LpInteger)
```

Figure 12 : Définition des variables

```
probleme = LpProblem(name='chiffre_affaires', sense=LpMaximize)
```

Figure 13 : Définition l'objectif de l'optimisation

```
probleme += (x+y <= n_caoutchouc)
probleme += (2*x+y <= n_acier)
```

Figure 14 : Définition des contraintes

```
probleme += prix_grosse_voiture*x + prix_petite_voiture*y
```

Figure 15 : Définition de la fonction objectif

Enfin, on choisit le solveur à utiliser, dans ce cas, c'est CBC, ce dernier est le solveur par défaut si on n'a rien préciser. En plus, on peut limiter le temps que le solveur a pour trouver une solution, par exemple à 20 secondes.

```
solver = PULP_CBC_CMD(timeLimit=20, msg=True)
probleme.solve(solver=solver)
```

Figure 16 : Choix du solveur

Une fois que le problème a été résolu, on peut afficher les valeurs trouvées.

```
print(f'x = {x.value()}')
print(f'y = {y.value()}')
```

Figure 17 : Affichage des résultats

2.Résolution et résultats

2.1. Méthode Solveur :

Après modéliser le programme linéaire en définissant les variables de décision, les contraintes, ainsi que la fonction multi-objectif, nous avons utilisé la méthode solveur pour résoudre le problème d'optimisation. Le processus de calcul s'est déroulé en les étapes suivantes :

1- **Entrer les données du Solveur :**

	A	B	C	D	E	F	G	H
1	**Ressource**	**Coût (USD/unité/s)**	**Performance (requêtes/unité/s)**	**Quantité allouée**	**Limite minimale**	**Limite maximale**	**C =**	0
2	CPU	0.046	100000	0	16	32	**P =**	0
3	GPU	1.5	500	0	4	8	**Z =**	0
4	RAM (Go)	0.0055	-	0	64	256		
5	Stockage (Go	0.0001	-	0	100	500		

Figure 18 : Définition des cellules des données sur Excel

Colonnes B et C indiquent *les coûts*, la colonne D correspond aux *variables de décision*, tandis que les colonnes E et F représentent *les contraintes minimales et maximales*. Enfin, la colonne H regroupe *les fonctions C, P et Z.*

1.1-Expression de la fonction du coût total C :

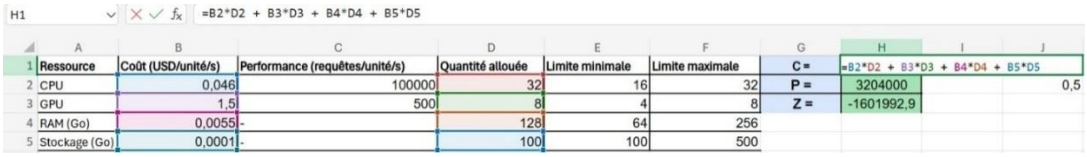


Figure 19 : Fonction du coût total

1.2-Expression de la fonction de la performance totale P :

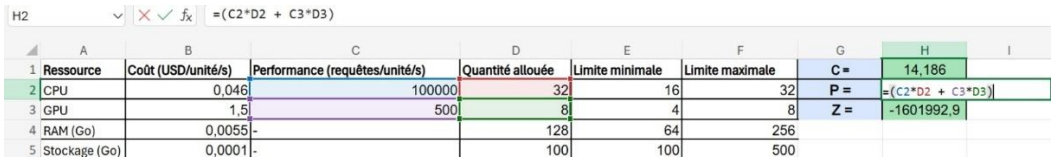


Figure 20 : Fonction de la performance totale

1.3-Expression de la fonction multi-objectifs Z :

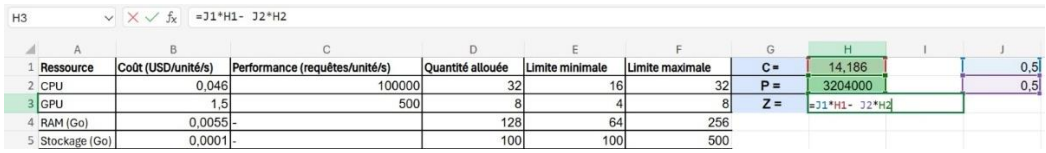


Figure 21 : Fonction multi-objectifs

2- Choix de la méthode de résolution :

Pour résoudre ce programme linéaire, nous avons opté pour la méthode simplexe. Cette méthode itérative a permis de trouver la solution optimale en explorant les différentes possibilités offertes par les contraintes du modèle

3- Exécution du calcul :

Une fois le modèle correctement configuré dans le solveur, nous avons lancé le calcul. Le solveur a alors procédé à une série d'itérations pour explorer les solutions possibles et déterminer la configuration des ressources la plus efficace, tout en respectant les contraintes imposées.

4- Résultats obtenus :

La répartition des ressources entre les différentes fonctions du chatbot a été déterminée. Ces valeurs optimales nous permettent de savoir combien de ressources allouer à chaque partie du système pour obtenir une performance optimale tout en minimisant les coûts.

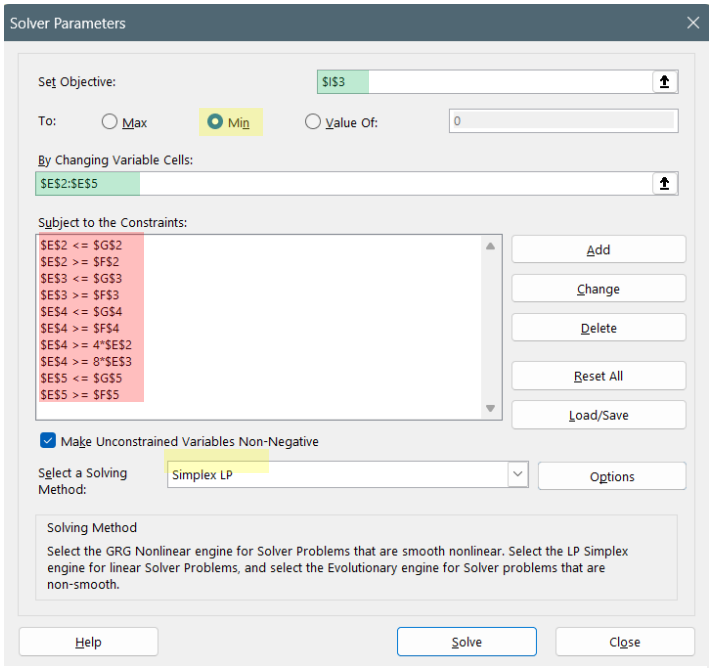


Figure 22 : Paramètres Solveur (Résolution)

	A	B	C	D	E	F	G	H
1	**Ressource**	**Coût (USD/unité/s)**	**Performance (requêtes/unité/s)**	**Quantité allouée**	**Limite minimale**	**Limite maximale**	**C =**	14.186
2	CPU	0.046	100000	32	16	32	**P =**	3204000
3	GPU	1.5	500	8	4	8	**Z =**	-1601992.91
4	RAM (Go)	0.0055	-	128	64	256		
5	Stockage (Go	0.0001	-	100	100	500		

Figure 23 : Résultats du Solveur

D'où on obtient une solution optimale avec **32 unités de CPU, 8 unités de GPU, 128 GO de RAM et 100 GO de stockage**. L'utilisation de ces valeurs permet d'assurer un minimum de coûts avec une vitesse de réponse plus rapide.

2.2. Méthode Python PuLP :

De la même manière, on résout programme linéaire à l'aide de la bibliothèque Python PuLP

```
from pulp import LpMinimize, LpProblem, LpVariable

# Définition du problème d'optimisation avec l'objectif de minimisation
problem = LpProblem("Allocation_Optimisation", LpMinimize)

# Définition des variables de décision représentant les ressources allouées
x1 = LpVariable("x1_CPU", lowBound=16, upBound=32, cat="Continuous")
x2 = LpVariable("x2_GPU", lowBound=4, upBound=8, cat="Continuous")
x3 = LpVariable("x3_RAM", lowBound=64, upBound=256, cat="Continuous")
x4 = LpVariable("x4_Storage", lowBound=100, upBound=500, cat="Continuous")

# Coefficients de coût associés à CPU, GPU, RAM, Stockage respectivement
c1, c2, c3, c4 = 0.046, 1.5, 0.0055, 0.0001
```

```
# Coefficients de performance associés à CPU et GPU respectivement
v1, v2 = 100000, 500

# Définition de la fonction de coût total
C = c1 * x1 + c2 * x2 + c3 * x3 + c4 * x4

# Définition de la fonction de performance totale
P = v1 * x1 + v2 * x2

# Définition des pondérations pour la fonction objectif
alpha, beta = 0.5, 0.5

# Ajout de la fonction objectif à minimiser
problem += alpha * C - beta * P

# Ajout des contraintes du problème
problem += x3 >= 4 * x1
problem += x3 >= 8 * x2

# Résolution du problème d'optimisation
problem.solve()

# Affichage des résultats optimaux
print("\nRésultats optimaux :")
print(f"x1 (CPU) : {x1.value()} unités")
print(f"x2 (GPU) : {x2.value()} unités")
print(f"x3 (RAM) : {x3.value()} Go")
print(f"x4 (Stockage) : {x4.value()} Go")

# Afficher la valeur de la fonction objectif Z
print(f"Valeur de Z : {problem.objective.value()}")
```

Et on obtient les mêmes résultats suivants :

```
Résultats optimaux :
x1 (CPU) : 32.0 unités
x2 (GPU) : 8.0 unités
x3 (RAM) : 128.0 Go
x4 (Stockage) : 100.0 Go
Valeur de Z : -1601992.9070000001
```

Figure 24 : Résultats du calcul avec PuLP

3.Interprétations et critiques

3.1. Interprétations des résultats :

Les résultats obtenus indiquent une allocation optimisée des ressources, répondant aux exigences d'un Chatbot complexe conçu pour gérer des charges utilisateur élevées tout en équilibrant les contraintes de coût et de performance. La configuration optimale comprenant **32 unités de CPU**, **8 GPU**, **128 Go de RAM** et **100 Go de stockage** reflète une solution fiable et adaptée aux besoins d'un système avancé de traitement de langage naturel, tel qu'un modèle de type GPT-4 ou similaire.

Les **32 unités de CPU** offrent une puissance de calcul suffisante pour gérer des tâches moins parallélisées, telles que la gestion des requêtes entrantes, la coordination des processus et le traitement des données.

Les **8 unités de GPU**, bien que coûteuses, sont essentielles pour des modèles volumineux comme GPT-4, où les calculs en parallèle sont nécessaires pour une inférence rapide et efficace. Les GPU permettent également de maintenir des temps de réponse faibles, ce qui est crucial pour garantir une expérience utilisateur fluide.

Les **128 Go de RAM** respectent les contraintes liées à la dépendance entre CPU et GPU. Cette quantité est idéale pour garantir que les données nécessaires aux calculs GPU et CPU sont disponibles en mémoire. Elle permet également de gérer de grandes quantités de données utilisateurs et de charger des modèles volumineux en mémoire.

Les **100 Go de stockage** alloués, bien que modestes, sont suffisants pour contenir les fichiers du modèle, les données temporaires, et les journaux nécessaires au bon fonctionnement du chatbot. Dans le contexte d'un système distribué ou d'un stockage externe, cette allocation peut être augmentée en cas de besoin.

La solution obtenue repose sur une **pondération équilibrée entre le coût (α)** et la **performance (β)** dans l'expression de la fonction multi-objectif. Dans ce cas, les coûts et les performances sont considérés avec la même priorité, ce qui garantit une allocation de ressources optimisée du coté économique et du coté performance. Cependant, cette priorité peut être ajustée :

- **Si la réduction des coûts devient prioritaire**, la valeur de α peut être augmentée, ce qui réduirait le nombre de GPU alloués ou d'autres ressources coûteuses.
- **Si la performance devient cruciale**, la valeur de β pourrait être augmentée, conduisant à une configuration avec davantage de GPU ou une RAM supérieure, au prix d'un coût plus élevé.

3.2. Critiques des résultats :

La valeur de Z n'est pas un problème en soi, tant que le sens de l'optimisation est respecté. La valeur négative est survenue vu que le terme associé à la performance est dominant par rapport au coût, ce qui indique que les performances du système (en requêtes par seconde) sont très élevées, entraînant un résultat global négatif.

Bien que les résultats montrent une solution robuste, la configuration semble surdimensionnée pour un chatbot standard. L'allocation de 8 GPU et 128 Go de RAM pourrait être excessive, surtout si le chatbot ne nécessite pas de modèles d'une telle capacité ou s'il ne réalise pas d'entraînement continu. Cela pourrait conduire à un coût total élevé avec une valeur de Z assez importante, ce qui suggère un compromis excessif vers la performance sans tenir compte suffisamment des contraintes de coût. Les ressources CPU (32 unités) et GPU (8 unités) pourraient être optimisées, et la quantité de RAM pourrait être réduite, notamment si les GPU sont déjà bien utilisés pour l'inférence. La fonction de coût pourrait être réajustée pour mieux équilibrer les priorités entre performance et coût. De plus, les résultats ne

prennent pas suffisamment en compte des scénarios d'utilisation plus réalistes, et des tests supplémentaires sur des configurations plus légères pourraient être nécessaires pour affiner cette solution.

Vu que les valeurs utilisées tout au long de ce traitement sont juste des approximations, la solution optimale peut ne pas fonctionner pour n'importe quel chatbot et elle sera alors fiable juste pour les cas qui respectent les estimations utilisées dans la fonction multi-objectf et les contraintes.

Conclusion

Dans cette phase d'optimisation, nous avons cherché à déterminer les ressources matérielles (CPU, GPU, RAM, stockage) nécessaires pour un chatbot tout en équilibrant les coûts et la performance. Les résultats obtenus indiquent une configuration robuste, mais qui pourrait ne pas être entièrement optimisée dans le contexte d'une solution chatbot typique.

Conclusion Générale

Ce projet a exploré l'optimisation des ressources pour un chatbot basé sur l'intelligence artificielle, mettant en lumière l'importance de concilier performance et coûts dans un environnement technologique exigeant. En adoptant une méthodologie rigoureuse, nous avons défini les enjeux liés à la gestion des infrastructures nécessaires pour faire fonctionner un chatbot de grande complexité. Le projet a débuté par une analyse des défis économiques et techniques, suivie d'une phase de modélisation mathématique utilisant une programmation linéaire multi-objectifs. Cette étape clé a permis de formuler une fonction objectif intégrant les coûts et les performances, tout en respectant des contraintes réalistes basées sur les caractéristiques des infrastructures modernes, comme les dépendances entre CPU, GPU, RAM et stockage. Les résultats obtenus, avec une allocation optimale de 32 unités de CPU, 8 GPU, 128 Go de RAM et 100 Go de stockage, démontrent la capacité du modèle à fournir une configuration robuste et adaptée pour répondre efficacement à des charges utilisateurs élevées. Ces ressources permettent de minimiser les coûts tout en garantissant une réponse rapide et fiable, ce qui est essentiel pour les modèles de chatbot volumineux tels que GPT-4. Par ailleurs, ce projet a montré la flexibilité du modèle grâce à l'utilisation de pondérations pour ajuster les priorités entre coûts et performances, offrant ainsi des solutions adaptées à divers cas d'utilisation.

À travers ce travail, nous avons pu découvrir la méthode Solveur et apprendre à l'utiliser pour résoudre un problème linéaire. De plus, nous avons appris à utiliser une fonction multi-objectifs, une approche permettant d'optimiser un vecteur de fonctions simultanément, chacune ayant une priorité, ce qui nous a ouvert de nouvelles perspectives dans la résolution de problèmes complexes. Nous avons également surmonté plusieurs difficultés, notamment le fait que nous avions l'habitude d'appliquer la programmation linéaire sur des problèmes ayant des variables de décision plus concrètes (une somme d'argent, un nombre de litres d'un liquide, le nombre d'heures de la disponibilité d'une machine...) , alors que dans ce cas, les paramètres sont plus discrets et difficile à manipuler directement. Cette situation nous a obligés à adapter notre approche et à développer des compétences en matière de modélisation et d'analyse.

Ce projet peut certainement être étendue à des recherches plus larges ou à des projets industriels car ce travail illustre non seulement l'importance de la recherche opérationnelle pour résoudre des problèmes concrets, mais également la nécessité de penser à l'avenir en intégrant des solutions adaptatives qui répondent aux variations dynamiques de la demande, tout en explorant des technologies émergentes pour améliorer encore l'efficacité des systèmes d'intelligence artificielle. En termes de travail en équipe, l'entraide et la collaboration ont été essentielles, car notre curiosité commune nous a amenés à travailler étroitement ensemble sur toutes les tâches, assurant ainsi qu'aucune information cruciale ne sera ratée. Cela a renforcé notre compréhension et notre efficacité.

En ce qui concerne l'organisation du projet, il n'y a pas de modifications majeures à apporter, car les sujets étaient riches et bien structurés, offrant ainsi une base solide pour une exploration approfondie. Ce projet a été une expérience d'apprentissage précieuse qui, nous l'espérons, pourra servir de modèle pour des futures projets dans notre carrière professionnelle.

Bibliographie

- Cours recherche opérationnelle (Programmation linéaire) – Pr. Abdessamad Kamouss
- Qu'est-ce qu'un chatbot ? | Oracle Morocco
- Optimiser les performances du chatbot IA avec le monitoring de l'IA de New Relic | New Relic
- Conception d'un « chatbot » pour soutenir les services d'information dans les bibliothèques universitaires - François Malick DIOUF1 , Reine Marie Ndela MARONE
- At a CAGR of 23.5% - The Chatbot Market Size is Estimated
- La consommation électrique des chatbots reposant sur l'IA est très importante - IT SOCIAL
- Qu'est-ce qu'un chatbot ? | IBM
- Playbook Chatbot : Tout savoir sur les chatbots
- Taille du marché des chatbots, part, croissance, tendances mondiales, prévisions 2030
- Statistiques clés sur les chatbots pour 2025 : Perceptions, croissance du marché, tendances
- Mes 20 prédictions pour 2025 – FredCavazza.net
- Chatbots IA : tout savoir sur ces assistants virtuels
- NVIDIA A100 | NVIDIA
- Comment mesurer l'efficacité de votre chatbot IA service client ?
- GPU et CPU : différence entre les unités de traitement – AWS
- Qu'est-ce qu'un GPU ? | IBM
- RAM d'un PC ou mémoire vive : définition simple, quantité nécessaire, différence avec le disque dur
- Stockage cloud - Définition et utilisation | Microsoft Azure
- Définir et résoudre un problème à l'aide du Solveur - Support Microsoft
- Utiliser le Solveur - Microsoft Excel 365
- Tarification d'instance à la demande EC2 – Amazon Web Services
- Instances P4d Amazon EC2 – AWS
- Introduction à la programmation linéaire avec PuLP