

ECOLE POLYTECHNIQUE DE L'UNIVERSITÉ FRANÇOIS RABELAIS DE TOURS

Département Informatique

64 avenue Jean Portalis

37200 Tours, France

Tél. +33 (0)2 47 36 14 14

polytech.univ-tours.fr

Projet Recherche & Développement

2018-2019

Recherche Opérationnelle et Machine Learning : le cas d'une matheuristique pour un problème de flowshop

**POLYTECH[®]**
TOURS

Tuteurs académiques

Vincent T'KINDT

Romain RAVEAUX

Nicolas RAGOT

Étudiant

Alafate ABULIMITI (DI5)



Liste des intervenants

Nom	Email	Qualité
Alafate ABULIMITI	alafate.abulimiti@etu.univ-tours.fr	Étudiant DI5
Vincent T'KINDT	tkindt@univ-tours.fr	Tuteur académique, Département Informatique
Romain RAVEAUX	romain.raveaux@univ-tours.fr	Tuteur académique, Département Informatique
Nicolas RAGOT	nicolas.ragot@univ-tours.fr	Tuteur académique, Département Informatique



Avertissement

Ce document a été rédigé par Alafate ABULIMITI susnommé l'auteur.

L'Ecole Polytechnique de l'Université François Rabelais de Tours est représentée par Vincent T'kindt, Romain Raveaux et Nicolas Ragot susnommés les tuteurs académiques.

Par l'utilisation de ce modèle de document, l'ensemble des intervenants du projet acceptent les conditions définies ci-après.

L'auteur reconnaît assumer l'entière responsabilité du contenu du document ainsi que toutes suites judiciaires qui pourraient en découler du fait du non respect des lois ou des droits d'auteur.

L'auteur atteste que les propos du document sont sincères et assument l'entière responsabilité de la véracité des propos.

L'auteur atteste ne pas s'approprier le travail d'autrui et que le document ne contient aucun plagiat.

L'auteur atteste que le document ne contient aucun propos diffamatoire ou condamnable devant la loi.

L'auteur reconnaît qu'il ne peut diffuser ce document en partie ou en intégralité sous quelque forme que ce soit sans l'accord préalable des tuteurs académiques et de l'entreprise.

L'auteur autorise l'école polytechnique de l'université François Rabelais de Tours à diffuser tout ou partie de ce document, sous quelque forme que ce soit, y compris après transformation en citant la source. Cette diffusion devra se faire gracieusement et être accompagnée du présent avertissement.



Pour citer ce document

Alafate ABULIMITI, *Recherche Opérationnelle et Machine Learning : le cas d'une matheuristique pour un problème de flowshop*, Projet Recherche & Développement, Ecole Polytechnique de l'Université François Rabelais de Tours, Tours, France, 2018-2019.

```
@mastersthesis{
  author={ABULIMITI, Alafate},
  title={Recherche Opérationnelle et Machine Learning : le cas d'une matheuristique pour
    un problème de flowshop: },
  type={Projet Recherche \& Développement},
  school={Ecole Polytechnique de l'Université François Rabelais de Tours},
  address={Tours, France},
  year={2018-2019}
}
```

Table des matières

Liste des intervenants	a
Avertissement	b
Pour citer ce document	c
Table des matières	i
Introduction	1
1 Acteurs du projet	1
2 Contexte de réalisation	2
2.1 Contexte	2
2.2 Description de problème	2
2.2.1 Objectif	2
2.2.2 Hypothèse	3
2.2.3 Base méthodologique.....	3
Description générale	4
3 Environnement du projet	4
4 Caractéristiques des utilisateurs	4
5 Fonctionnalités et structure générale du système	5
6 Contraintes de développement, d'exploitation et de maintenance	6
Description des interfaces externes du logiciel	7
7 Interfaces homme/machine	7
8 Interfaces logiciel/logiciel	7

Spécifications fonctionnelles	9
9 Spécifications fonctionnelles pour le réseau de neurones	9
9.1 Définition de la fonction 1 : CreateInstance	9
9.2 Définition de la fonction 2 : CreateInitialSeq	9
9.3 Définition de la fonction 3 : GenerateBase	9
9.4 Définition de la fonction 4 : ParseCSV	10
9.5 Définition de la fonction 5 : NetConstructor	10
9.6 Définition de la fonction 6 : TrainNet	10
9.7 Définition de la fonction 7 : TestNet	11
10 Spécifications fonctionnelles pour la résolution du problème de flowshop	11
10.1 Définition de la fonction 8 : createTargetSeq	11
10.2 Définition de la fonction 8 : SolveFS	11
10.3 Définition de la fonction 9 : TestFS	11
Spécifications non fonctionnelles	12
11 Contraintes de développement et conception	12
11.1 Matériels	12
11.2 langage de programmation.....	12
11.3 logiciels et bibliothèques	12
12 Contraintes de fonctionnement et d'exploitation	13
12.1 Performances.....	13
12.2 Capacités.....	13
12.3 Contrôlabilité.....	13
12.4 Sécurité	13
État de l'art	14
12.5 Problème flowshop	14
12.5.1 Contexte.....	14
12.5.2 Algorithmes Heuristiques	15
12.6 Artificial Neural Network	17
12.7 Convolutional Neural Network.....	19
12.7.1 Contexte.....	19
12.7.2 Modèle	19
12.8 Recurrent Neural Network.....	21
12.9 Long Short-Term Memory(LSTM)	22
12.9.1 Contexte.....	22
12.9.2 Modèle	22
12.10 Pointer Network.....	24
12.10.1 Introduction.....	24
12.10.2 Modèle	24

12.11	Reinforcement Learning.....	25
12.11.1	Introduction.....	25
12.11.2	Modèle	25
Analyse et conception		27
13	Analyse du problème.....	27
14	Analyse d'entrée/sortie	27
15	Analyse de l'algorithme de génération de la base d'apprentissage	28
16	Modélisation des données	28
16.1	Entrée	29
16.2	Sortie	29
17	Modélisation de Modèle	30
17.1	Introduction	30
17.2	RNN	30
17.3	Encoder	31
17.4	Decoder	32
18	Loss Fonction	33
Mise en œuvre		34
19	Génération de la base d'apprentissage.....	34
20	Construction des réseaux de neurones	34
20.1	Preprocess	35
20.2	Seq2SeqModel.....	37
21	Test.....	37
21.1	Test scenario 1.....	38
21.2	Test scenario 2.....	39
21.3	Test scenario 3.....	40
22	Conclusion	41
Plan de développement		43
23	Découpage du projet en tâches	43
23.1	Tache Étude.....	43
23.1.1	Étude Problème flowshop	43
23.1.2	Étude problème d'apprentissage automatique	44
23.2	Réalisation.....	44
23.2.1	Réalisation d'écriture.....	44
23.2.2	Réalisation de programmation	46
23.2.3	Réalisation de Test	46
24	Planning.....	47
25	Bilan sur la qualité	47
26	Bilan auto-critique.....	47

Annexes	50
A Protocole pour générer la base d'apprentissage	51
1 Préparation	51
2 Définition de base d'apprentissage (Learning database).....	51
3 Exécution d'algorithme	52
B Protocole de test	54
1 Test pour le réseau de neurones	54
2 Test pour la résolution du problème de flowshop	55
C Guide Developement et utilisation	57
1 Téléchargement du projet.....	57
2 Libraires	57



Table des figures



Liste des Algorithmes

Introduction

Le Projet de Recherche & Développement (PR & D), réalisé en 5ème année, s'inscrit dans la formation dispensée à l'École Polytechnique de l'Université de Tours et constitue une réelle expérience en terme de conduite de projet. Le PR&D se déroule du 18 septembre 2018 au 30 mars 2018 à raison de 2 jours par semaine à temps plein. Il donne lieu à la rédaction d'un rapport avec le cahier de spécification et de deux présentations du travail effectué lors de deux soutenances.

Le PR & D est orienté vers une étude théorique et en même temps un développement algorithmique de résolution de problème « flowshop de temps total d'achèvement de deux machines » où les MOA étaient représentés par M. Vincent T'kindt, M. Romain Raveaux et M. Nicolas Ragot, membres du Laboratoire d'Informatique de Tours. Le projet s'inscrit dans un cadre théorique qui fait l'objet d'une résolution avec la technique de Deep Learning pour le problème de flowshop de temps total d'achèvement de deux machines.

Ce document est donc le cahier de spécification du projet « Recherche Opérationnelle et Machine Learning : le cas d'une matheuristique pour un problème de flowshop ». Il définit les besoins, l'environnement du projet et les objectifs à réaliser. Ce rapport présente aussi les différentes tâches à effectuer et le planning prévisionnel.

1 Acteurs du projet

Les acteurs du projet sont :

- La maîtrise d'œuvre (MOE) : Alafate ABULIMITI, étudiant en 5ème année de l'École Polytechnique de l'Université de Tours au sein du département Informatique, dans le cadre du PR & D ainsi que ses encadrants M. Vincent T'kindt, M. Romain Raveaux et M. Nicolas Ragot de l'équipe de recherche ROOT du Laboratoire d'Informatique de Tours. Les recherches menées au sein de cette équipe portent sur l'étude et la résolution des problèmes de recherche opérationnelle, que ce soit dans le cadre de problématiques industrielles ou de problèmes théoriques.
- La maîtrise d'ouvrage (MOA) : L'équipe de recherche ROOT du Laboratoire d'Informatique de Tours, représentée par M. Vincent T'kindt, M. Romain Raveaux et M. Nicolas Ragot.

Ce document a été rédigé par Alafate ABULIMITI et sera validé par les différents acteurs du projet.



Figure 1 – Le logo de l'école

2 Contexte de réalisation

2.1 Contexte

Un problème de planification typique consiste à organiser de nombreuses tâches tout en respectant les contraintes de temps et de ressources pour optimiser la fonction objectif. Il y a m machines en série. Chaque travail doit être traité sur chaque une des m machines. Tous les tâches(jobs) doivent suivre le même itinéraire, c'est à dire qu'ils doivent d'abord être traités sur la machine 1, puis sur la machine 2 et bientôt. Après l'achèvement sur une machine, un travail rejoint la file d'attente au machine suivante. Habituellement, toutes les files d'attente sont supposées fonctionner sous la discipline FIFO. L'objectif de l'algorithme est de minimiser le temps que la seconde machine termine la dernière tâche.

$$\min \sum_{j=1}^n C_{2,j} \quad (1)$$

un nouvel algorithme est proposé dans le papier « A matheuristic approach for the two-machine total completion time flowshop problem » de Federico Della Croce · Andrea Grosso Fabio Salassa. Il est basé sur l'algorithme d'origine.

Ils ont proposé d'ajouter une contrainte sur l'algorithme d'origine :

$x_{i,j}$: Si job i est en position j , $x_{i,j} \leftarrow 1$, sinon $x_{i,j} \leftarrow 0$ ($j = 1$ ou 2)

r représente une position spécifique dans la séquence et h représente la taille.

$$x_{i,j} = \bar{x}_{i,j} \quad \forall i \notin \bar{S}(r;h), j \notin \{r, \dots, r+h+1\} \quad (W)$$

2.2 Description de problème

Dans ce papier, il est mentionné qu'en choisissant h , ils pensent empiriquement que $h = 12$ est un meilleur paramètre d'optimisation. Notre question est de savoir si, pour ce problème typique de NP-Complete, nous pouvons donner les meilleurs r et h pour toute séquence S basée sur le réseau de neurones entraînés.

2.2.1 Objectif

L'objectif général du projet est démontrer la pertinence de coupler les techniques relatives à ces deux domaines. Pour cela nous allons nous intéresser à un problème d'optimisation (ordonnancement) pour lequel nous possédons un algorithme (une matheuristique) classique

dans la littérature Recherche Opérationnelle. Nous allons alors mettre au point un algorithme d'apprentissage à base de réseaux de neurones qui doit permettre de guider plus efficacement la matheuristique dans sa recherche d'une bonne solution au problème d'optimisation.

2.2.2 Hypothèse

Nous espérons pouvoir créer un système qui donne les meilleurs r et h lorsque nous entrons une séquence.

2.2.3 Base méthodologique

On rédige toutes les documentations à l'aide de la suite Office de Microsoft, LaTeX et Astah Professional. Pour les diagrammes d'analyse, on utilise le langage UML (Unified Modeling Language). Ce langage nous permet d'avoir une modélisation formelle et on l'utilise pour créer les diagrammes des cas d'utilisations.

Description générale

3 Environnement du projet

Le projet est réalisé en langage Python sous Macs. - Version : *Python 3.6.6* **Figure 2**ERROR.



Figure 2 – *Python*

- IDE : *JetBrains Pycharm* **Figure 3**ERROR



Figure 3 – *PyCharm*

4 Caractéristiques des utilisateurs

La MOA sera l'utilisatrice des logiciels. Celle-ci possède de bonnes connaissances en Deep Learning et en recherche opérationnelle.

5 Fonctionnalités et structure générale du système

Nous avons divisé le problème en trois parties, à savoir la résolution du problème de flowshop, la génération de la base d'apprentissage et la construction du réseau de neurones.

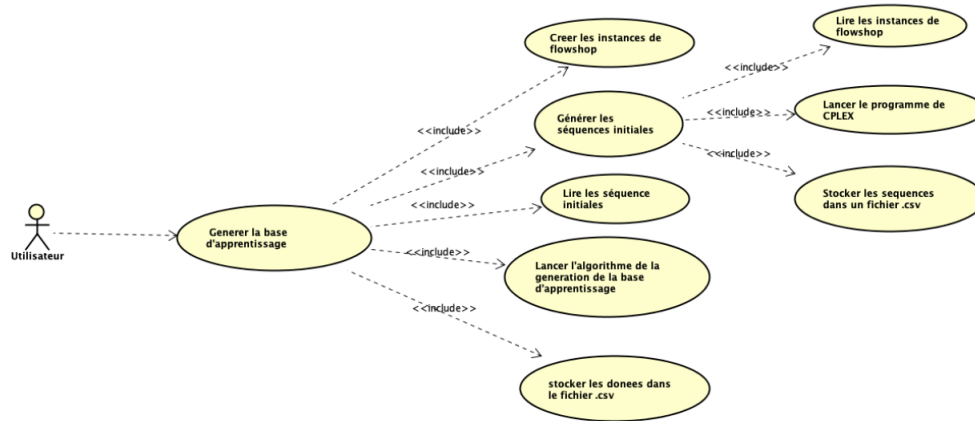


Figure 4 – Génération de la base d'apprentissage

Construire une excellente base d'apprentissage est un élément essentiel de la construction d'un réseau de neurones, car une bonne base d'apprentissage peut mieux entraîner notre réseau de neurones. Lors de la construction de la base d'apprentissage, nous devons d'abord créer les instances de flowshop, sur lesquelles nous exécutons notre programme CPLEX afin d'obtenir les séquences initiales, puis notre algorithme de génération de la base d'apprentissage en fonction des séquences initiales. Voir annexe A <la protocole de la génération de la base d'apprentissage> pour plus de détails.

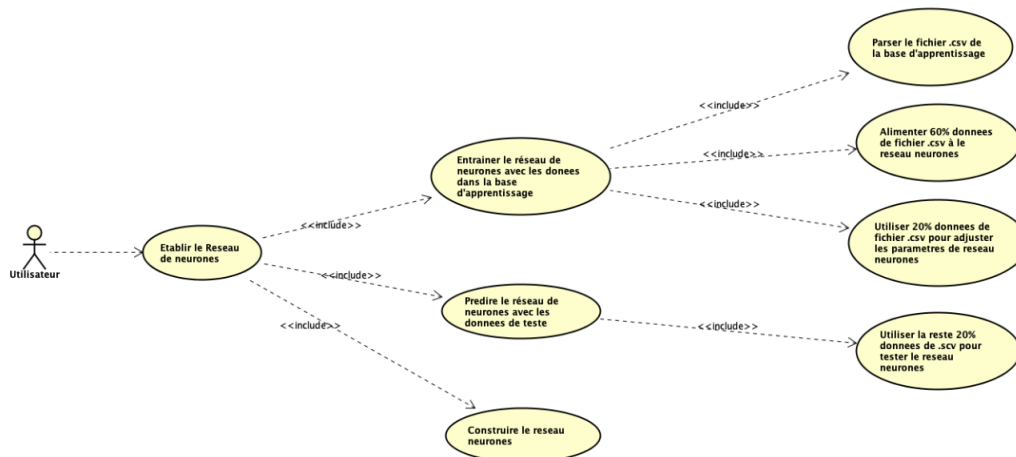


Figure 5 – Construction du réseau de neurones

Nous établissons d'abord un réseau de neurones correspondant sur la base du modèle que nous avons sélectionné (le type spécifique sera déterminé dans la section pratique ultérieure), puis nous alimenterons 60% des données de la base d'apprentissage vers notre réseau de neurones, nous utiliserons 20% des données de la base d'apprentissage pour ajuster les paramètres et notre réseau de neurones a été testé avec les 20% restants des données.

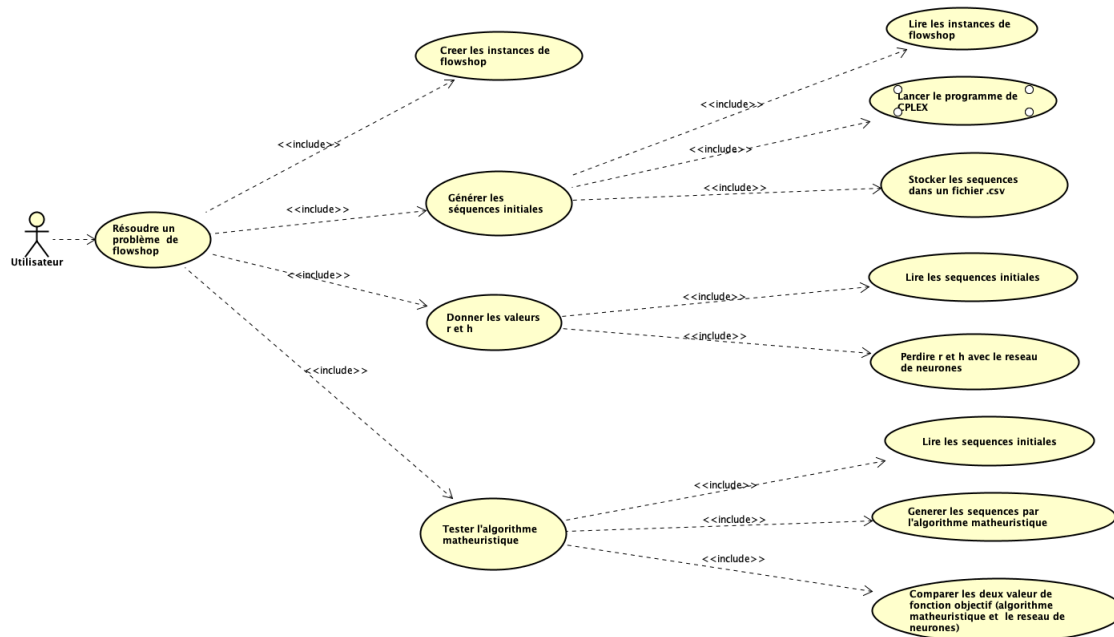


Figure 6 – Résolution du problème de flowshop

Notre dernière partie consiste à résoudre notre problème de flowshop : nous générons d'abord l'instance initiale, puis nous obtenons la séquence initiale conformément à l'exemple, puis, selon la séquence initiale, nous utilisons la méthode d'optimisation mentionnée dans le papier pour obtenir la séquence optimisée. Le réseau de neurones entraîné obtient une autre séquence ou r , h . Nous comparons les valeurs de la fonction objectif de ces deux méthodes de test.

6 Contraintes de développement, d'exploitation et de maintenance

Nous n'avons pas de contraintes matérielles, mais il convient de noter que nous devons économiser le coût de notre processeur et essayer de résoudre ce problème dans un certain délai. Pour le programme de résolution, nous pouvons utiliser Python pour le langage de programmation et Pycharm pour l'IDE. Le résultat peut être écrit dans un fichier .csv.

Description des interfaces externes du logiciel

7 Interfaces homme/machine

Dans ce projet, notre programme se lancera en ligne de commande. Nous sommes composés de trois étapes, premièrement, nous devons d'abord générer la base d'apprentissage et stocker dans le fichier .csv. Nous définissons l'entrée et la sortie comme suit :

- ✓ Entrée : Le chemin pour stocker le fichier .csv
- ✓ Sortie : Fichier .csv qui stocke les données.

Deuxièmement, nous devons lire le fichier csv et former notre réseau de neurones. Nous définissons l'entrée et la sortie comme suit :

- ✓ Entrée : Le chemin de fichier .csv
- ✓ Sortie : Le réseau neurone entraîné

Troisièmement, nous devons utiliser le réseau de neurone pour résoudre le problème de flow-shop. Nous définissons l'entrée et la sortie comme suit :

- ✓ Entrée : le chemin .csv qui contient la séquence pour résoudre
- ✓ Sortie : les valeurs r et h .

8 Interfaces logiciel/logiciel

Afin de générer la séquence initiale et de générer la base d'apprentissage, nous allons utiliser le programme CPLEX. Nous appellerons ce programme comme un fichier exe. L'appel est le suivant :

Il n'y aura donc pas d'autre interaction de notre programme avec d'autre.

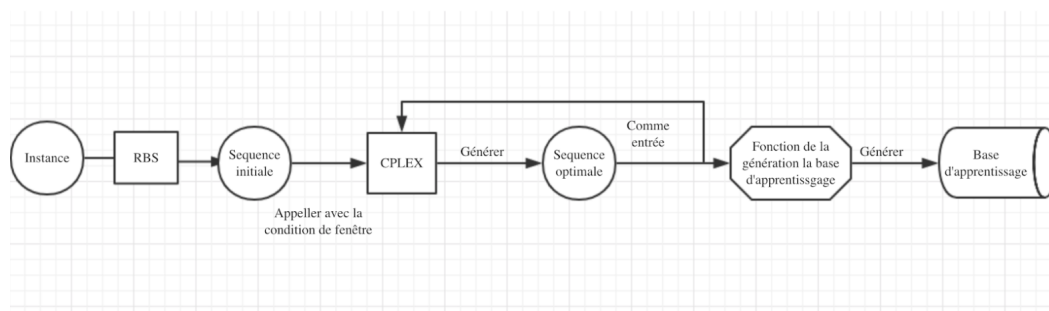


Figure 7 – *Interaction avec CPLEX*

Spécifications fonctionnelles

Dans ce projet, nous construirons deux programmes, l'un pour construire et entraîner un réseau de neurones, et l'autre pour résoudre le problème de flowshop.

9 Spécifications fonctionnelles pour le réseau de neurones

9.1 Définition de la fonction 1 : CreateInstance

Identification de la fonction 1

Cette fonction permet de générer l'instance du problème de flowshop pour générer la séquence initiale.

Description de la fonction 1

- ✓ Entrée : NULL
- ✓ Sortie : un fichier de .csv qui stocke la table d'instance

9.2 Définition de la fonction 2 : CreateInitialSeq

Identification de la fonction 2

Cette fonction permet de appeler le programme CPLEX et générer la séquence initiale.

Description de la fonction 2

- ✓ Entrée : le chemin de fichier .csv qui stocke la table d'instance.
- ✓ Sortie : un fichier de .csv qui stocke la séquence initiale et le temps d'achèvement.

9.3 Définition de la fonction 3 : GenerateBase

Identification de la fonction 3

Cette fonction permet de générer la base d'apprentissage selon la protocole pour générer la base d'apprentissage.

Description de la fonction 3

- ✓ Entrée : le chemin de fichier .csv qui stocke la séquence initiale et le temps d'achèvement.
- ✓ Sortie : un fichier csv qui stocke la base d'apprentissage, voyez Annexe A : <Protocole pour la base de l'apprentissage- 2. Définition de la base d'apprentissage> pour détailler la structure.

9.4 Définition de la fonction 4 : ParseCSV

Identification de la fonction 4

Nous devons analyser le fichier .csv et en extraire les données et les diviser en 3 parties :

- 1.ensemble d'entraînement(60%).
- 2.ensemble de validation(20%).
- 3.ensemble de test(20%).

Description de la fonction 4

- ✓ Entrée : un fichier csv, il contient les données générées par la fonction GenerateBase.
- ✓ Sortie :
 1. Forme de matrice d'ensemble de entraînement selon la base d'apprentissage
 2. Forme de matrice d'ensemble de validation selon la base d'apprentissage
 3. Forme de matrice d'ensemble de test selon la base d'apprentissage

9.5 Définition de la fonction 5 : NetConstructor

Identification de la fonction 5

C'est le constructeur du réseau de neurones que nous utilisons.

Description de la fonction 5

- ✓ Entrée : Entrée : NULL
- ✓ Sortie : Sortie : un système de réseau de neurones.

9.6 Définition de la fonction 6 : TrainNet

Identification de la fonction 6

Cette fonction consiste à entraîner notre réseau de neurones.

Description de la fonction 6

- ✓ Entrée : le premier retour et le deuxième retour de fonction ParseCSV (ensemble d'entraînement et ensemble de validation)
- ✓ Sortie : Le Réseau de neurones qui est entraîné.

9.7 Définition de la fonction 7 : TestNet

Identification de la fonction 7

Cette fonction consiste à tester notre réseau de neurones.

Description de la fonction 7

- ✓ Entrée : le troisième retour de fonction ParseCSV (ensemble de test)
- ✓ Sortie : Le Réseau de neurones qui est testé.

10 Spécifications fonctionnelles pour la résolution du problème de flowshop

10.1 Définition de la fonction 8 : createTargetSeq

Identification de la fonction 8

Cette fonction appellera CreateInstance et CreateInitialSeq pour générer une séquence, et nous appellerons la fonction suivante pour trouver les paramètres optimaux pour cette séquence.

Description de la fonction 8

- ✓ Entrée : NULL
- ✓ Sortie : un fichier .csv pour stocker la séquence et le temps d'achevenement.

10.2 Définition de la fonction 8 : SolveFS

Identification de la fonction 8

Cette fonction utilisera le réseau de neurones que nous avons entraîné pour donner les meilleurs paramètres r et h d'optimisation pour une séquence de flowshop. Pour une séquence de flowshop.

Description de la fonction 8

- ✓ Entrée : le chemin de fichier .csv qui stocke la séquence et le temps d'achèvement.
- ✓ Sortie : les meilleurs paramètres r et h .

10.3 Définition de la fonction 9 : TestFS

Identification de la fonction 9

Cette fonction consiste à tester l'efficacité de notre programme, nous utilisons la méthode d'optimisation mentionnée dans le papier pour obtenir la séquence optimisée. Le réseau de neurones entraîné obtient une autre séquence ou r , h . Nous comparons les valeurs de la fonction objectif de ces deux méthodes de test.

Description de la fonction 9

- ✓ Entrée : le chemin de fichier .csv qui stocke la séquence initiale et le temps d'achèvement.
- ✓ Sortie : le fichier csv qui stocke les résultat de fonction objective de deux méthodes différentes.

Spécifications non fonctionnelles

11 Contraintes de développement et conception

11.1 Matériels

Notre projet étant de résoudre un problème de recherche opérationnelle de manière efficace à l'aide de l'apprentissage automatique, il utilise un réseau de neurones pour résoudre ce problème, donc le coût CPU est donc important. Nous utiliserons le CPU pour entraîner au lieu du GPU.

11.2 langage de programmation

Nous utiliserons python comme langage de programmation pour les raisons suivantes :

- ✓ Python a de nombreuses bibliothèques et frameworks utiles pour la construction de réseaux de neurones que nous pouvons utiliser.
- ✓ La langue est concise et largement utilisée.

11.3 logiciels et bibliothèques

- ✓ Nous utiliserons pyCharm comme IDE en raison de sa praticabilité et de son efficacité.
- ✓ Nous allons utiliser le framework Keras. Keras est une bibliothèque de réseaux de neurones écrite en Python pur qui se concentre sur l'apprentissage en profondeur et fonctionne sous TensorFlow ou Theano. TensorFlow et Theano sont deux des bibliothèques d'apprentissage en profondeur les plus populaires, mais elles sont relativement compliquées pour les débutants. Keras est simple à utiliser et possède une structure claire. La plate-forme est puissante et est basée sur TensorFlow ou Theano. Keras s'exécute dans un environnement Python 2.7 ou 3.6+.

- ✓ Dans notre projet, nous utiliserons des opérations matricielles, et le NumPy est une bibliothèque ouverte de Python. Cet outil peut être utilisé pour stocker et traiter des matrices volumineuses, bien plus efficace que la structure de liste imbriquée de Python. Il est donc également utilisé dans notre programme.

12 Contraintes de fonctionnement et d'exploitation

12.1 Performances

La performance de ce programme dépend de :

- ✓ La taille du taux correct dans notre ensemble de test.
- ✓ Étant donné que ce projet concentre sur l'utilisation de la CPU, le temps nécessaire pour établir une prévision est également un indicateur.

12.2 Capacités

- ✓ Le logiciel de résolution de problème de flowshop peut accepter une seule recherche à la fois, si on lance plusieurs recherches en même temps, notre logiciel sera bloqué.
- ✓ Pendant le déroulement du programme, nous devons contrôler le temps d'exécution maximum. En d'autres termes, nous devons terminer notre projet dans les délais impartis.

12.3 Contrôlabilité

Afin de suivre l'exécution du programme, des messages seront affichés en console.

12.4 Sécurité

Aucune demande particulière n'a été fait en sécurité.

État de l'art

Dans cette partie, je vais présenter des états de l'art de Problème flowshop en même temps le mécanisme de CNN et RNN. Celles-ci ont été guidés par M. Vincent T'kindt ,M. Romain Raveaux et M. Nicolas Ragot. Le but de ces recherches est de bien comprendre les différentes idées par des articles récents qui présentent le développement de ce domaine. A la fin de l'état de l'art, je vais proposer une idée réalisable afin de répondre aux objectifs de ce PR & D.

12.5 Problème flowshop

12.5.1 Contexte

Définition de problème flowshop

Il y a m machines en série. Chaque travail doit être traité sur chaque une des m machines. Tous les emplois doivent suivre le même itinéraire, c'est à dire qu'ils doivent d'abord être traités sur la machine 1, puis sur la machine 2 et bientôt. Après l'achèvement sur une machine, un travail rejoint la file d'attente au machine suivante. Habituellement, toutes les files d'attente sont supposées fonctionner sous la discipline FIFO, ce qui est un travail ne peut pas passer un autre en attente dans une file d'attente. On peut montrer qu'il existe toujours un emploi du temps optimal sans séquence de travail change entre les deux premières machines et les deux dernières Machines.

Définition du temps d'achèvement

Heure à laquelle les travaux terminent leur traitement sur la m -ème machine, qui est la dernière. Nous aimerions trouver un calendrier qui minimise une fonction des délais d'exécution.

$$C_{max} = \max(C_1, \dots, C_n)$$

$$\sum_{i=1}^n C_{i,m}$$

12.5.2 Algorithmes Heuristiques

Afin de minimiser le temps d'achèvement, un algorithme heuristique est proposé. L'algorithme est le suivant :

Définitions de paramètres

- * $P_{i,j}$: Le temps de terminer le job i sur la seule machine j (Processing time) ($j = 1$ ou 2)
- * $x_{i,j}$: Si job i est en position j , $x_{i,j} \leftarrow 1$, sinon $x_{i,j} \leftarrow 0$ ($j = 1$ ou 2)
- * $C_{i,j}$: Le temps de terminer le job i sur la machine j (Completion time) ($j = 1$ ou 2)

Objectif

$$\min \sum_{j=1}^n C_{2,j} \quad (1)$$

L'objectif de l'algorithme est de minimiser le temps que la seconde machine termine la dernière tâche.

Contraintes

$$\sum_{i=1}^n x_{i,j} = 1 \quad \forall j = 1, \dots, n \quad (2)$$

$$\sum_{j=1}^n x_{i,j} = 1 \quad \forall i = 1, \dots, n \quad (3)$$

$$C_{1,1} = \sum_{i=1}^n p_{1,i} x_{i,1} \quad (4)$$

$$C_{2,1} = C_{1,1} + \sum_{i=1}^n p_{2,i} x_{i,1} \quad (5)$$

$$C_{1,j} = C_{1,j-1} + \sum_{i=1}^n p_{1,i} x_{i,j} \quad \forall j = 2, \dots, n \quad (6)$$

$$C_{2,j} \geq C_{1,j} + \sum_{i=1}^n p_{2,i} x_{i,j} \quad \forall j = 2, \dots, n \quad (7)$$

$$C_{2,j} \geq C_{1,j-1} + \sum_{i=1}^n p_{2,i} x_{i,j} \quad \forall j = 2, \dots, n \quad (8)$$

$$x_{i,j} \in \{0, 1\} \quad (9)$$

1. Les contraintes (2) à (3) indiquent qu'un travail est choisi pour chaque position dans la séquence et que chaque travail est traité exactement une fois.
2. Les contraintes (4) - (5) définissent l'heure de fin du premier travail sur les deux machines.
3. La contrainte (6) indique que chaque tâche ne peut démarrer sur la machine qu'une fois celle-ci terminée.
4. La contrainte (7) interdit pour chaque tâche le début de la deuxième opération sur la machine deux avant que l'opération précédente sur la machine une ne soit terminée.
5. La contrainte (8) indique que chaque tâche ne peut démarrer sur la machine deux qu'après la fin de la tâche la précédant immédiatement sur la même machine.

Dans le papier « A matheuristic approach for the two-machine total completion time flowshop problem » de Federico Della Croce · Andrea Grosso Fabio Salassa, un nouvel algorithme est proposé. Il est basé sur l'algorithme d'origine.

Ils ont proposé d'ajouter une contrainte sur l'algorithme d'origine :

$$x_{i,j} = \bar{x}_{i,j} \quad \forall i \notin \bar{S}(r;h), j \notin \{r, \dots, r+h+1\} \quad (W)$$

r représente une position spécifique dans la séquence et h représente la taille.

L'idée principale de cet algorithme est de faire une optimisation sur la séquence générée par l'algorithme d'origine. Pour ce faire, il exécute l'algorithme d'origine sur la partie locale de la séquence d'origine.

La contrainte (W) ici est d'ouvrir une "fenêtre" pour la séquence d'origine. Exemple :

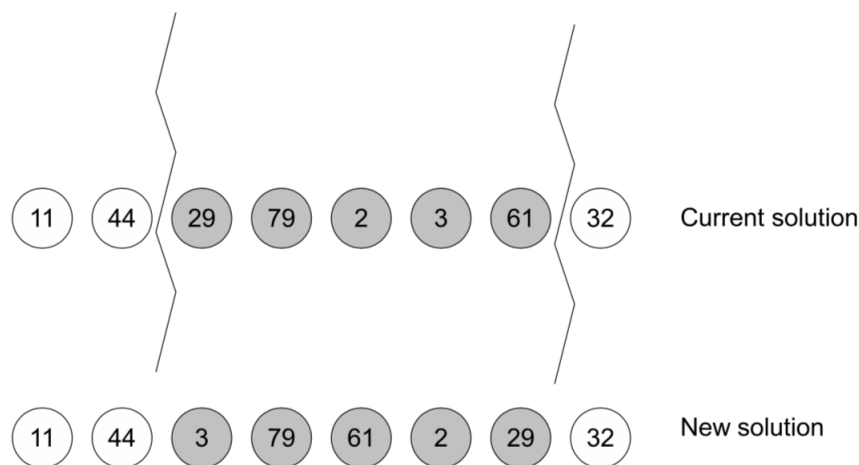


Figure 8 – Exemple de l'algorithme

Dans cet exemple, nous sélectionnons $r = 2, h = 5$, plus la contrainte W et exécutons l'algorithme d'origine. On peut avoir une nouvelle séquence. Le temps d'achèvement de la nouvelle séquence est plus petite que la séquence précédente.

Problématique

Le but de ce projet est d'utiliser les réseaux de neurones pour trouver les valeurs les plus appropriées pour r et h .

Compte tenu de nos coûts de CPU, nous rencontrerons deux problèmes : 1. Quel réseau de neurones utilisons-nous pour ce projet ? 2. Comment construire une bonne base d'apprentissage pour entraîner notre réseau de neurones ?

12.6 Artificial Neural Network

Le réseau de neurones artificiels (ANN) est un point chaud de recherche dans le domaine de l'intelligence artificielle depuis les années 1980. Il résume le réseau de neurones du cerveau humain du point de vue du traitement de l'information, établit un modèle simple et forme différents réseaux selon différentes méthodes de connexion. Il est également souvent appelé réseau de neurones ou réseau de neurones. Un réseau de neurones est un modèle opérationnel constitué d'un grand nombre de nœuds (ou neurones) connectés les uns aux autres. Chaque nœud représente une fonction de sortie spécifique appelée **fonction d'activation**. La connexion entre tous les deux nœuds représente une valeur de pondération permettant de transmettre le signal de connexion, appelée pondération, équivalente à la mémoire du réseau de neurones artificiels. La sortie du réseau varie en fonction de la méthode de connexion du réseau, de la valeur de poids et de la **fonction d'activation**. Le réseau lui-même est généralement une approximation d'un algorithme ou d'une fonction dans la nature, ou peut être une expression d'une stratégie logique.

Neurone

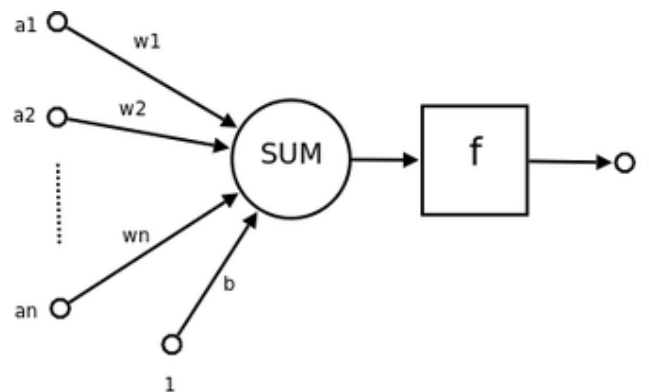


Figure 9 – une neurone

- . $a_1 \dots a_n$ sont les Composants d'entrée du vecteur.
- . $w_1 \dots w_n$ sont les poids des neurones
- . b est le biais
- . f est une fonction de transfert, généralement est une fonction non linéaire.
- . t est la sortie du neurone

Structure de ANN

Un réseau de neurones multicouches commun se compose de trois parties.

- 1 La couche d'entrée, un certain nombre de neurones, accepte un grand nombre de messages d'entrée non linéaires. Le message d'entrée est appelé le vecteur d'entrée.
- 2 La couche de sortie, le message est transmis, analysé et pesé dans le lien de neurone pour former une sortie. Le message de sortie est appelé le vecteur de sortie.
- 3 La couche cachée est la couche de nombreux neurones et liens entre la couche d'entrée et la couche de sortie. La couche cachée peut avoir une ou plusieurs couches. Le nombre de nœuds (neurones) dans la couche masquée est variable, mais plus le nombre est élevé, plus la non-linéarité du réseau de neurones est significative. Le mieux la robustesse.

Fonction de perte

La fonction de perte sert à estimer le degré d'incohérence entre la valeur prédite $f(x)$ de votre modèle et la valeur vraie Y . Il s'agit d'une fonction à valeur réelle non négative, généralement

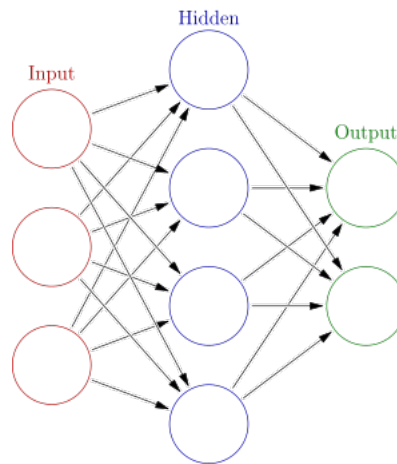


Figure 10 – Structure de ANN

exprimée par $L(Y, f(x))$. Plus la fonction de perte est petite, meilleure est la robustesse du modèle. Généralement il est exprimé comme suit :

$$\theta^* = \arg \min_{\theta} \frac{1}{N} \sum_{i=1}^N L(y_i, f(x_i, \theta)) + \lambda \Phi(\theta)$$

Algorithme de la Backpropagation

Afin de choisir les paramètres optimaux, nous devons utiliser l'Algorithme de Backpropagation. C'est une méthode couramment utilisée pour former des réseaux de neurones artificiels en association avec des méthodes d'optimisation telles que la descente de gradient. Cette méthode calcule le gradient de la fonction de perte pour les poids dans le réseau. Ce gradient est renvoyé à la méthode d'optimisation pour mettre à jour les poids afin de minimiser la fonction de perte.

Overfitting et Underfitting

Lorsque le modèle ne donne pas de bons résultats pour les données d'apprentissage, le modèle est en underfitting. Cela est dû au fait que le modèle ne peut pas capturer la relation entre l'exemple en entrée (appelé X) et la valeur cible (appelée Y). Lorsque le modèle fonctionne bien sur les données d'apprentissage, mais pas sur les données d'évaluation, il indique que le modèle est en overfitting. En effet, le modèle garde en mémoire les données qui ont été vues, mais elles ne peuvent pas être résumées comme un exemple qui n'a pas été vu.

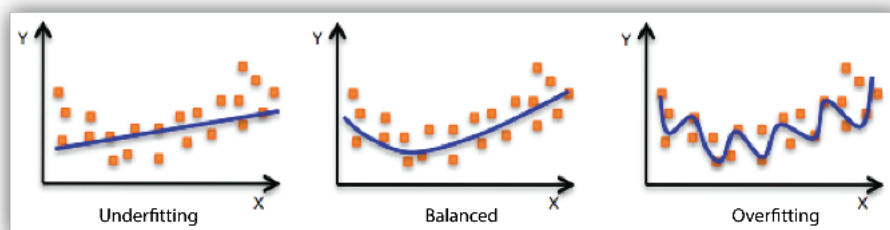


Figure 11 – Overfitting et Underfitting

12.7 Convolutional Neural Network

12.7.1 Contexte

Les réseaux de neurones convolutifs sont une méthode d'identification efficace qui s'est développée au cours des dernières années et a suscité une grande attention. Dans les années 1960, quand Hubel et Wiesel étudièrent la sélection sensible et directionnelle locale des neurones dans le cortex cérébral du chat, ils découvrirent que leur structure de réseau unique pouvait réduire efficacement la complexité du réseau de neurones de rétroaction, puis proposèrent un réseau de neurones convolutionnel (Réseaux de neurones convolutionnels - CNN en abrégé). De nos jours, CNN est devenu l'un des points chauds de la recherche dans de nombreux domaines scientifiques, en particulier dans le domaine de la classification des formes, car il évite le prétraitement complexe des images, il peut donc saisir directement l'image d'origine, ce qui lui a permis d'être largement utilisé. La nouvelle machine de reconnaissance proposée par K. Fukushima en 1980 est le premier réseau de mise en œuvre de réseaux de neurones à convolution. Par la suite, davantage de chercheurs ont amélioré le réseau. Parmi ceux-ci, les résultats de recherche représentatifs sont «l'amélioration des machines cognitives» proposée par Alexander et Taylor, qui combine les avantages de diverses méthodes améliorées et évite une longue propagation de la propagation des erreurs.

Généralement, la structure de base du CNN comprend deux couches, dont une couche d'extraction de caractéristiques, et l'entrée de chaque neurone est connectée au domaine d'acceptation local de la couche précédente et les caractéristiques locales sont extraites. Une fois la caractéristique locale extraite, sa relation de position avec les autres caractéristiques est également déterminée : la seconde est la couche de mappage des caractéristiques, chaque couche de calcul du réseau est composée de plusieurs cartes de caractéristiques et chaque carte de caractéristiques est un plan. Tous les neurones du plan ont des poids égaux. La structure de mappage des fonctionnalités utilise une petite fonction sigmoïde qui affecte le noyau de la fonction en tant que fonction d'activation du réseau de convolution, de sorte que la mappe des fonctionnalités présente une invariance de déplacement. De plus, étant donné que les neurones d'une surface de cartographie partagent des poids, le nombre de paramètres de réseau libre est réduit. Chaque couche de convolution du réseau de neurones de convolution est suivie d'une couche de calcul pour la moyenne locale et l'extraction quadratique. Cette structure d'extraction unique à deux caractéristiques réduit la résolution des entités.

12.7.2 Modèle

En traitement d'image, une image est souvent représentée sous forme de vecteur de pixels, pour une image, nous avons beaucoup de pixels. Avec trop de paramètres, nous ne pouvons pas entraîner de réseaux de neurones. Donc, on doit réduire les paramètres pour accélérer l'entraînement.

Couche de convolution

1. Perception locale

Les réseaux de neurones convolutifs comportent deux façons pouvant réduire le nombre de paramètres, le premier étant appelé champ de perception locale. Il est généralement admis que la perception du monde extérieur par les gens va du local au global et que la connexion spatiale des images est également une connexion proche des pixels locaux, tandis que la corrélation des pixels éloignés est faible. Par conséquent, il n'est pas nécessaire que chaque neurone perçoive

l'image globale : il suffit de percevoir la partie locale, puis d'intégrer les informations locales à un niveau supérieur pour obtenir des informations globales. Comme le montre la figure ci-dessous : l'image de gauche correspond à une connexion complète, l'image de droite correspond à une connexion locale.

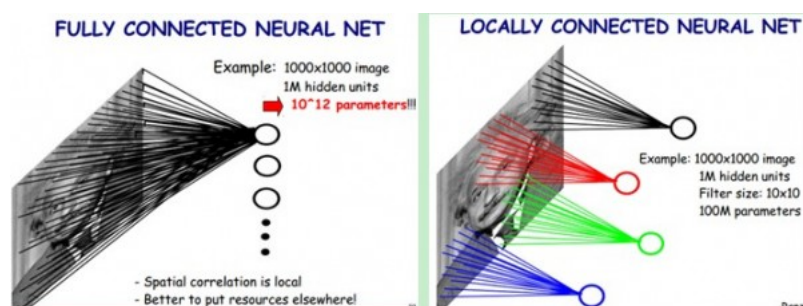


Figure 12 – Perception locale

2. Partage de paramètres

Mais en fait, si les paramètres sont encore trop nombreux, nous pouvons utiliser la deuxième méthode, à savoir le partage des paramètres. Le principe implicite est que les propriétés statistiques d'une partie de l'image sont identiques à celles des autres parties. Cela signifie également que les fonctionnalités que nous avons apprises dans cette partie peuvent également être utilisées dans une autre partie. Nous pouvons donc utiliser les mêmes caractéristiques d'apprentissage pour toutes les positions sur cette image.

Comme le montre la figure ci-dessous, un processus de convolution d'un noyau à 3x3 convolutive sur une image de 5x5 est présenté. Chaque convolution est une méthode d'extraction de caractéristiques qui filtre les parties de l'image qui répondent aux critères (plus la valeur d'activation est grande, plus il est éligible).

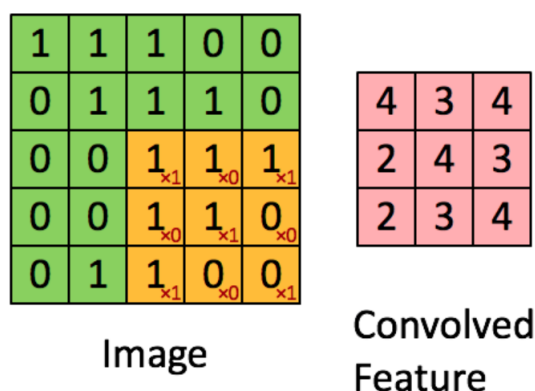


Figure 13 – Partage de paramètres

3. Multi-Noyau convolutif

Lorsqu'il n'y a que 100 paramètres mentionnés ci-dessus, cela indique qu'il n'y a qu'un seul 100 * 100 nœuds convolutif. Évidemment, l'extraction de fonctionnalités n'est pas suffisante. Nous pouvons ajouter plusieurs noyaux convolutifs tels que 32 noyaux convolutifs, et en apprendre 32 caractéristiques. Lorsqu'il y a plusieurs filtres, comme indiqué ci-dessous :

Couche de Pooling

Après avoir obtenu les caractéristiques par convolution, nous souhaitons utiliser l'étape suivante pour effectuer la classification. En théorie, il est possible d'utiliser toutes les fonctionnalités

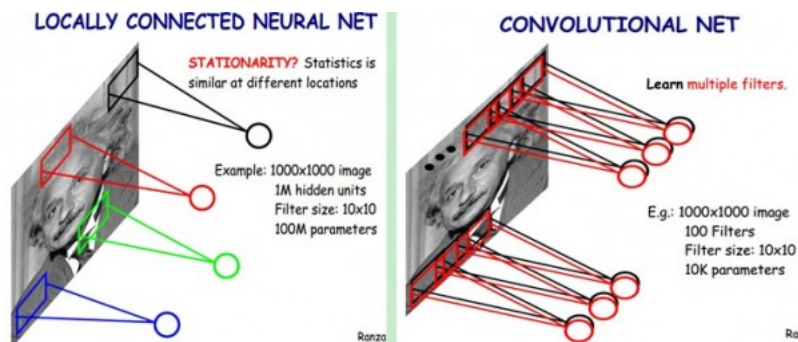


Figure 14 – Multi-Noyau convolutif

extraites pour former le classificateur, tel que le classificateur softmax, mais il s'agit d'un défi de calcul.

Pour résoudre ce problème, rappelons tout d'abord que nous avons décidé d'utiliser la fonctionnalité de convolution car l'image a une propriété «statique», ce qui signifie que les fonctionnalités utiles dans une région d'image ont toutes les chances d'être dans une autre. La même chose s'applique à la région. Par conséquent, afin de décrire de grandes images, une idée naturelle consiste à agréger des statistiques sur des entités situées à différents endroits, par exemple en calculant la moyenne (ou le maximum) d'une entité particulière sur une région d'une image. Ces caractéristiques statistiques récapitulatives ont non seulement des dimensions beaucoup plus basses (par rapport à l'utilisation de toutes les fonctionnalités extraites), mais améliorent également les résultats (pas facile à surapprentissage). Ce type d'agrégation est appelé pooling, parfois appelé pooling moyen ou pooling maximum (selon la méthode de calcul du pooling).

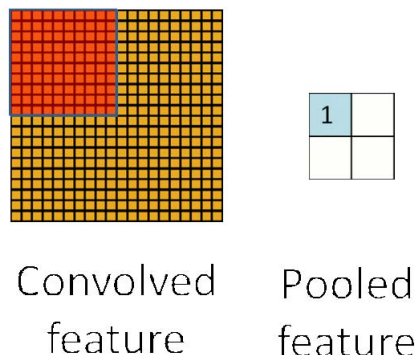


Figure 15 – Pooling

Couche entièrement connectée

Les neurones dans une couche entièrement connectée ont des connexions avec toutes les activations de la couche précédente, comme on le voit dans les réseaux neuronaux réguliers. Leurs activations peuvent donc être calculées avec une multiplication matricielle suivie d'un décalage de polarisation.

12.8 Recurrent Neural Network

Réseau de neurones récurrent, un réseau de neurones est un réseau de neurones artificiel dans lequel des nœuds sont connectés dans une boucle. L'état interne d'un tel réseau peut présenter un comportement de synchronisation dynamique. Contrairement au réseau neuronal à réaction, le RNN peut utiliser sa mémoire interne pour traiter des séquences d'entrée à minutage arbitraire, ce qui facilite le traitement de la reconnaissance de l'écriture manuscrite, de la reconnaissance vocale, etc., sans segmentation.

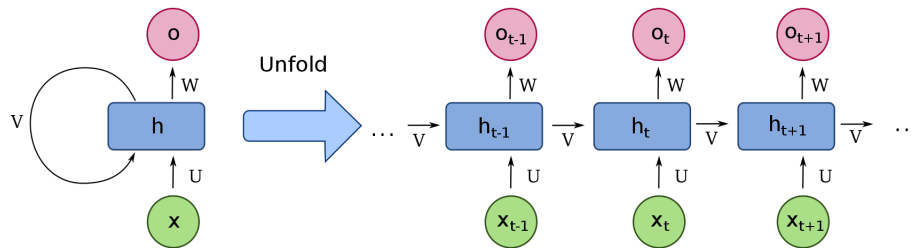


Figure 16 – Réseau de neurones récurrent

12.9 Long Short-Term Memory(LSTM)

12.9.1 Contexte

RNN n'a aucun moyen de résoudre le problème de la mémoire à long terme. Par exemple, en essayant de prédire le dernier mot de "J'ai 25 ans ... je parle le français". Des informations récentes montrent que le mot suivant peut être le nom d'une langue, mais si nous voulons restreindre le choix, nous devons inclure le contexte "France" et déduire les mots de l'information précédente. Il est tout à fait possible que l'intervalle entre les informations pertinentes et la position prédite soit grand. Cependant, RNN n'a aucun moyen de résoudre ce problème. Mais LSTM offre une bonne solution aux problèmes de séries chronologiques à long terme.

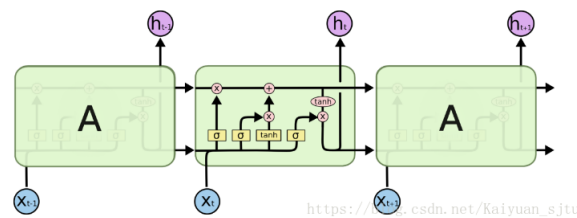


Figure 17 – Structure de LSTM

12.9.2 Modèle

Porte d'oubli

La porte d'oubli est utilisée pour contrôler si elle est oubliée. En LSTM, elle est contrôlée avec une certaine probabilité d'oubli ou non de l'état de cellule cachée de la couche supérieure. La structure de la porte d'oubli est montrée dans la figure suivante : L'état caché $h(t-1)$ de la séquence précédente et les données de séquence $x(t)$ entrées dans la figure sont obtenus par une fonction d'activation, généralement sigmoïde, pour obtenir la sortie $f(t)$ de la porte d'oubli. Puisque la sortie $f(t)$ de sigmoïde est comprise entre $[0, 1]$, la sortie $f^{(t)}$ représente ici la probabilité d'oubli de l'état de la cellule cachée. C'est :

Porte d'entrée

Après que le RNN a passé la "porte d'oubli", il doit également compléter la mémoire la plus récente de l'entrée actuelle, ce qui nécessite la création d'une "porte d'entrée".

La porte d'entrée est responsable du traitement de l'entrée de la position actuelle de la séquence et de sa sous-structure : La figure montre que la porte d'entrée est composée de deux parties : la

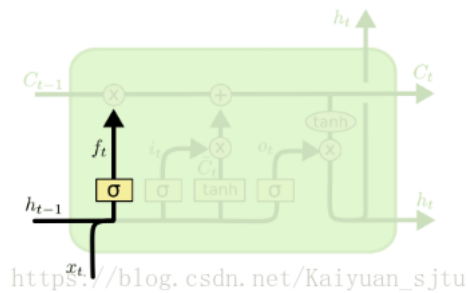


Figure 18 – Structure de porte d'oubli

$$f^{(t)} = \sigma(W_f h^{(t-1)} + U_f x^{(t)} + b_f)$$

Figure 19 – Formule de porte d'oubli

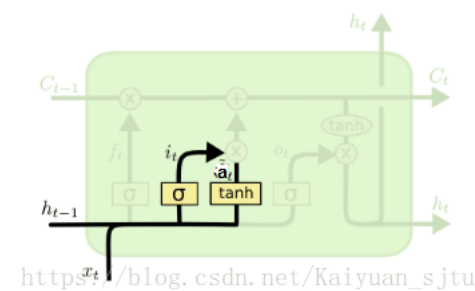


Figure 20 – Structure de porte d'entrée

première utilise la fonction d'activation sigmoid, la sortie est $i(t)$, la deuxième partie utilise la fonction d'activation tanh et la sortie est $a(t)$. Multiplier puis mettez à jour l'état de la cellule. Utiliser une expression mathématique, c'est :

$$\begin{aligned} \hat{i}^{(t)} &= \sigma(W_i h^{(t-1)} + U_i x^{(t)} + b_i) \\ \hat{a}^{(t)} &= \tanh(W_a h^{(t-1)} + U_a x^{(t)} + b_a) \end{aligned}$$

Figure 21 – Formule de porte d'entrée

Mise à jour de l'état de la cellule Avant d'étudier la porte de sortie LSTM, nous devons d'abord examiner l'état de la cellule du LSTM. La porte d'oubli précédente et le résultat de la porte d'entrée agissent tous deux sur l'état de la cellule $C(t)$. Comme indiqué ci-dessous :

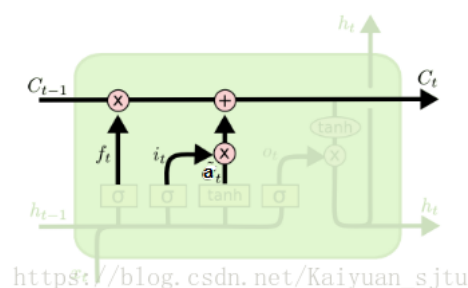


Figure 22 – Mise à jour de l'état de la cellule

L'état de la cellule $C(t)$ est constitué de deux parties, la première partie est le produit de $C(t-1)$ et la sortie de la porte d'oubli $f(t)$ et la seconde partie est le produit des portes $i(t)$ et $a(t)$ de la porte d'entrée. C'est :

$$C^{(t)} = C^{(t-1)} \odot f^{(t)} + i^{(t)} \odot a^{(t)}$$

Figure 23 – Formule de la mise à jour de l'état de la cellule

Porte de sortie

Avec le nouvel état de cellule masqué $C(t)$, nous avons la porte de sortie avec la structure suivante :

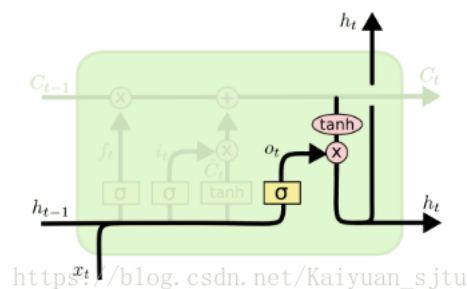


Figure 24 – Structure de porte de sortie

On voit sur la figure que la mise à jour de l'état caché $h(t)$ comprend deux parties, la première partie est $o(t)$, qui est l'état caché $h(t-1)$ de la séquence précédente et les données de séquence $x(t)$. Et la fonction d'activation sigmoïde est obtenue, la deuxième partie est constituée de l'état caché $C(t)$ et de la fonction d'activation tanh, c'est :

$$o^{(t)} = \sigma(W_o h^{(t-1)} + U_o x^{(t)} + b_o)$$

$$h^{(t)} = o^{(t)} \odot \tanh(C^{(t)})$$

Figure 25 – Formule de porte de sortie

12.10 Pointer Network

12.10.1 Introduction

Le réseau de pointeurs est une variante du modèle Seq2Seq. Seq2Seq est basé sur un encodeur LSTM et un décodeur LSTM. Dans le contexte de la traduction automatique, le plus souvent entendu est : Phrase dans une langue, le codeur le transforme en une déclaration de taille fixe. Le décodeur le convertit en une phrase dont la longueur peut être différente de celle de la phrase précédente. Par exemple, "Comment ça va ?" - deux mots - sera traduit par "comment allez-vous ?" - trois mots. Au lieu de convertir une séquence en une autre séquence, le réseau de pointeurs génère une série de pointeurs sur les éléments de la séquence en entrée. L'utilisation la plus élémentaire consiste à trier les éléments d'une séquence ou d'une collection de longueur variable.

12.10.2 Modèle

(a) Seq2Seq - Un RNN (bleu) traite la séquence d'entrée pour créer un vecteur de code utilisé pour générer la séquence de sortie (violet) à l'aide de la règle de chaîne de probabilité et d'un

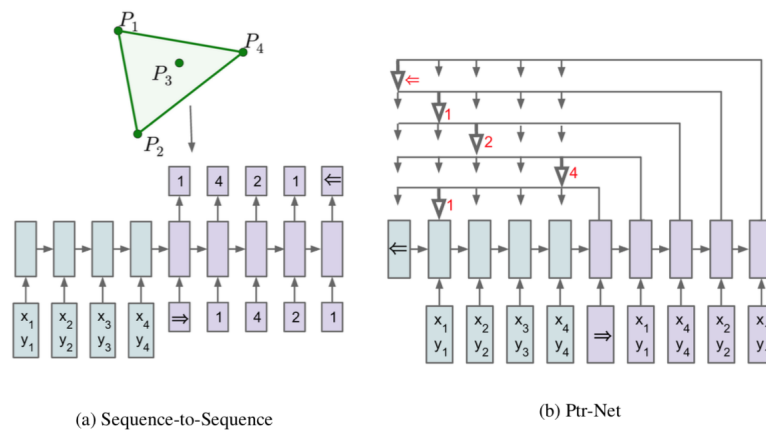


Figure 26 – PtrNet et Seq2Seq

autre RNN. La dimensionnalité de sortie est fixée par la dimensionnalité du problème et il en est de même lors de la formation et de l'inférence [1]. (b) Ptr-Net - Un RNN codant convertit la séquence d'entrée en un code (bleu) qui est transmis au réseau générateur (en violet). À chaque étape, le réseau générateur produit un vecteur qui module un mécanisme d'attention basé sur le contenu sur les entrées ([5, 2]). Le mécanisme d'attention produit en sortie une distribution softmax dont la taille du dictionnaire est égale à la longueur de l'entrée.

12.11 Reinforcement Learning

12.11.1 Introduction

L'apprentissage par renforcement est un comportement guidé par une récompense, dans lequel l'agent apprend par essais et erreurs et dont le but est de maximiser la récompense de l'agent par son interaction avec l'environnement. Le signal amélioré fourni par l'environnement dans l'apprentissage par renforcement consiste à évaluer la qualité de l'action (généralement un signal scalaire) au lieu d'indiquer au système d'apprentissage par renforcement RLS (système d'apprentissage par renforcement) comment générer l'action correcte. Parce que l'environnement externe fournit très peu d'informations, RLS doit apprendre de sa propre expérience. De cette manière, RLS acquiert des connaissances dans un environnement d'évaluation des actions et améliore le plan d'action pour s'adapter à l'environnement.

12.11.2 Modèle

L'apprentissage par renforcement est développé à partir des théories de l'apprentissage des animaux, du contrôle adaptatif des perturbations de paramètres, etc. Les principes de base sont les suivants : Si une certaine stratégie comportementale de l'agent entraîne une récompense positive (signal amélioré), la tendance de l'agent à générer cette stratégie comportementale sera alors renforcée. L'objectif de l'agent est de trouver la stratégie optimale dans chaque état discret pour maximiser et récompenser la remise souhaitée. L'apprentissage par renforcement considère l'apprentissage comme un processus d'évaluation provisoire. L'agent sélectionne une action pour l'environnement. Une fois que l'environnement a accepté l'action, l'état change et, en même temps, un signal amélioré (récompense ou punition) est envoyé à l'agent. L'agent intensifie à nouveau le signal et l'état actuel de l'environnement. En choisissant l'action suivante, le principe

de choix consiste à augmenter la probabilité d'être renforcé positivement (attribution). L'action sélectionnée affecte non seulement la valeur d'amélioration immédiate, mais également l'état de l'environnement au moment suivant et la valeur d'amélioration finale.

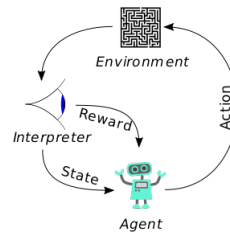


Figure 27 – *Structure de Reinforcement Learning*

Analyse et conception

13 Analyse du problème

Dans ce projet, nous allons utiliser l'approche du réseau de neurones pour trouver les paramètres optimaux r et h dans l'algorithme heuristique. Le réseau de neurones présente les caractéristiques suivantes :

- ✓ Il peut approximer complètement les relations non linéaires arbitrairement complexes.
- ✓ Toutes les informations quantitatives ou qualitatives sont distribuées de manière équipotentielle à chaque neurone du réseau, de sorte qu'elles sont extrêmement robustes.
- ✓ L'adoption d'un procédé de traitement distribué parallèle permettant d'effectuer rapidement un grand nombre d'opérations.
- ✓ Il peut apprendre et s'adapter à des systèmes inconnus ou incertains.
- ✓ Capacité à traiter simultanément des connaissances quantitatives et qualitatives.

Dans l'algorithme heuristique original, nous pouvons rencontrer le fait que la solution que nous avons obtenue est optimale localement et non la solution optimale. Sous cette prémisse, nous espérons obtenir la solution optimale plus rapidement en combinant les caractéristiques des réseaux de neurones.

14 Analyse d'entrée/sortie

Entrée

Nous avons deux programmes, l'un pour la construction d'un réseau de neurones et l'autre pour la résolution du problème de flowshop.

En général, l'entrée du réseau de neurones est une base d'apprentissage et la construction de la base d'apprentissage est basée sur un algorithme permettant de construire une table d'apprentissage. Pour la construction de la table d'apprentissage du réseau de neurones, les importances sont :

1. Assez de données. C'est-à-dire que pour construire un réseau de neurones avec une très bonne robustesse et qu'il n'est pas facile de sur-adapter, il suffit de disposer de suffisamment

de données. Dans le processus de résolution de ce problème, le temps de génération peut être limité, ce qui peut poser le problème du manque de données.

2. Données de bonne qualité. Les données de bonne qualité signifient que les données ne sont pas redondantes et qu'il n'y a pas de données erronées. Dans le processus de génération de données, nous allons rencontrer ces problèmes, je vais analyser dans la partie de analyse de l'algorithme.

Pour la résolution du problème de flowshop, l'entrée est une séquence initiale.

Sortie

Globalement, pour un réseau de neurones, la sortie est un système qui fonctionne bien et qui est bien entraîné. Comment évaluer les performances d'un réseau de neurones? Nous avons ici deux méthodes, l'une est la courbe de la fonction de perte et l'autre est la validation croisée. voir Annexe B <Protocole de test> pour plus de détails.

Pour la résolution du problème de flowshop. La sortie ici est le r et h optimaux de l'entrée après que la séquence initiale a traversé le réseau de neurones.

15 Analyse de l'algorithme de génération de la base d'apprentissage

Dans l'annexe A, nous voyons que nous fournissons un algorithme pour générer une table d'apprentissage. Après analyse, nous avons constaté que nous pouvions rencontrer les problèmes suivants pour cet algorithme :

1. Pour une séquence surdimensionnée, parcourir tous les r et h sera une tâche fastidieuse. Ainsi, selon le papier mentionnés, ils ont choisi une taille de fenêtre h égale à 12. En train de construire la base d'apprentissage, nous pouvons partir du nombre autour de 12, tel que 6 à 18. Il y a deux raisons :

- ✓ Si h est trop petit, l'effet d'optimisation n'est pas évident. 6 est donc notre valeur la moins expérimentée.
- ✓ Si h est trop grand, nous avons des limites de temps et il est donc possible que nous n'obtenions pas de solution satisfaisante.

2. Lors de la composition de la base d'apprentissage, le problème que nous avons rencontré était le suivant : lorsque nous avons le même S , nous avons un $\%I$ différent, parce que pour un S , différents r et h peuvent correspondre au différent $\%I$. Cela fera en sorte que le réseau de neurones ne fonctionne pas correctement. Quand nous sommes en train d'importer la base d'apprentissage, nous faisons un filtre, pour le même S , nous choisissons le plus haut $\%I$.

3. Lorsque nous générons S' , il peut y avoir deux S' dans le même ordre mais correspondant à différents r et h . Donc, quand nous devons écrire une petite méthode pour déterminer si c'est le cas, s'il y a S' , nous choisirons une fenêtre plus petite (un h plus petit)

16 Modélisation des données

La tâche principale de notre projet est de savoir comment former notre réseau de neurones pour prédire notre r et h . Mais dans ce processus, la construction des données de entraînement deviendra très importante.

Nos données seront générées à partir du programme CPLEX et sauvegardées dans un fichier txt, construit comme suit :

n	r	h	PerImp	PerImpSolInit	(pi1,pi2)S	(pi1,pi2)BestS
...

- ✓ n : nombre de jobs
- ✓ r : position de départ de la meilleure fenêtre dans la séquence
- ✓ h : taille de la meilleure fenêtre
- ✓ PerImp : Pourcentage d'amélioration par rapport à la séquence devant
- ✓ PerImpSolInit : Pourcentage d'amélioration par rapport à la séquence initiale
- ✓ (pi1,pi2)S : Séquence entrée dans le programme CPLEX avec la forme de processing time.
- ✓ (pi1,pi2)S : Séquence sortie améliorée par le programme CPLEX avec la forme de processing time.

Il est difficile pour nous d'utiliser ces données directement pour former nos réseaux de neurones. Nous devons utiliser ces données pour transformer les données en une structure de données que nous pouvons utiliser.

Nous utilisons un réseau de neurones de modèles supervisés pour prédire r et h . Nous allons donc être divisés en deux parties pour construire le modèle des données.

16.1 Entrée

L'entrée utilisée pour former le réseau de neurones répond aux exigences suivantes :

1. Il peut représenter la structure de la séquence.
2. Assez de informations aident le réseau de neurones à comprendre la structure de la séquence.

Selon les exigences ci-dessus, notre conception est la suivante :

Vecteur du [Processing Time Completion Time] = un Job

Nous utilisons processing time plus completion time pour représenter les informations d'un travail. Completion time est l'heure à laquelle sont terminés les machines 1 et 2 à chaque fois qu'un travail est passé.

Exemple : [4 10 4 10] Job 2 [5 17 9 27] Job 4...

Une partie de cette séquence, dans le vecteur de job2, Le premier chiffre est processing time dans la machine 1 Le deuxième chiffre est processing time dans la machine 2, le troisième représente completion time dans la machine 1 et le quatrième représente completion time dans la machine2.

16.2 Sortie

L'Sortie utilisée pour former le réseau de neurones répond aux exigences suivantes :

1. Il peut représenter la structure de la fenêtre.
2. Assez de informations aident le réseau de neurones à comprendre la structure de la fenêtre.

Selon les exigences ci-dessus, notre conception est la suivante :

Nous utilisons une séquence de travaux pour représenter la structure et la taille de la fenêtre

optimisée par CPLEX. Nous utilisons 0 pour les travaux en dehors de la fenêtre et 1 pour les travaux en dehors de la fenêtre.

Exemple : 00000011111000

Dans cet exemple, $r = 6$ et $h = 5$

17 Modélisation de Modèle

17.1 Introduction

Nous devons examiner de plus près ce modèle seq2seq, puis définissons l'équation. Le modèle seq2seq est divisé en deux parties distinctes, une pour le Encoder et une pour le Decoder.

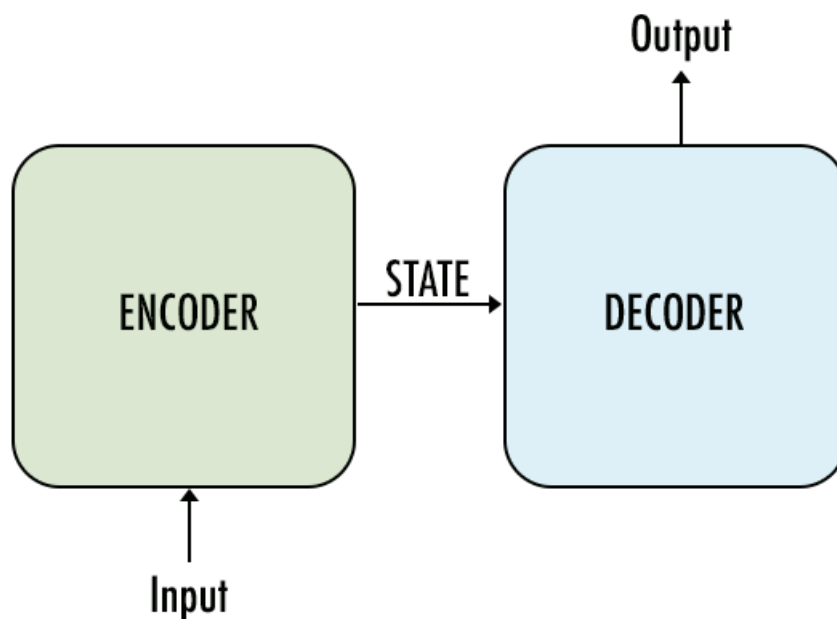


Figure 28 – Seq2Seq Modèle

17.2 RNN

Avant d'introduire le modèle seq2seq, nous introduisons d'abord le modèle RNN classique.

$$h^{<t>} = g_1(W_{aa}a^{<t-1>} + W_{ax}X^{<t>} + b_a)$$

$$\hat{y}^{<t>} = g_2(W_{ya}a^{<t>} + b_y)$$

J'utiliserai ces conventions symboliques pour représenter ces indices matriciels, par exemple W_{ax} , le deuxième indice signifie que W_{ax} doit multiplier Avec un x , le premier indice a indique qu'il est utilisé pour calculer une variable a . Le même, vous pouvez voir W_{ya} ici est multiplié par a , utilisé pour calculer y .

Comme indiqué ci-dessus :

✓ n est la taille de séquence.

We can process a sequence of vectors \mathbf{x} by applying a **recurrence formula** at every time step:

$$\boxed{h_t} = \boxed{f_W}(\boxed{h_{t-1}}, \boxed{x_t})$$

new state old state input vector at some time step
some function with parameters W

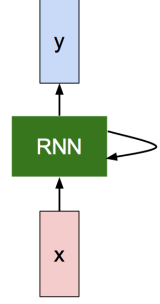


Figure 29 – RNN classique

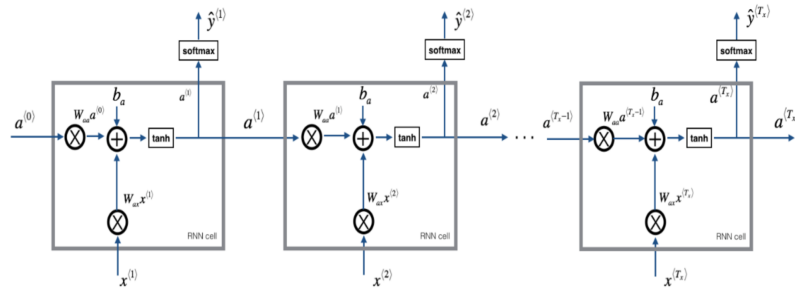


Figure 30 – Structure de RNN classique

- ✓ m est la nombre de neurones.
- ✓ $h^{<t>}$ est un état caché au moment t , $h \in \mathbb{R}^{n \times 1}$.
- ✓ $\hat{y}^{<t>}$ est le nouvel état au moment t , $\hat{y} \in \{0, 1\}^{1 \times n}$.
- ✓ $W_{aa} \in \mathbb{R}^{m \times m}$.
- ✓ $W_{ax} \in \mathbb{R}^{n \times n}$.
- ✓ $W_{ya} \in \mathbb{R}^{n \times n}$.
- ✓ $b_a \in \mathbb{R}^{n \times 1}$ est la valeur de déviation pour calculer $a^{<t>}$.
- ✓ $b_y \in \mathbb{R}^{n \times 1}$ est la valeur de déviation pour calculer $\hat{y}^{<t>}$.
- ✓ X_t est l'entrée au moment t , $X \in \mathbb{R}^{n \times 2}$.

17.3 Encoder

Ici, nous utilisons le bi-direction encoder le plus courant. Une séquence de vecteur $X = (x_1, \dots, x_{T_x})$. nous définissons :

- ✓ h_t est un état caché au moment t , $h \in \mathbb{R}^{n \times 1}$.
- ✓ $c \in \mathbb{R}^{n \times 1}$ est un vecteur généré à partir de la séquence des états caché.
- ✓ f et q sont des fonctions non linéaires.

Nous avons :

$$h_t = f(x_t, h_{t-1})$$

$$c = q(h_1, \dots, h_{T_x})$$

Cependant, étant donné que RNN standard traitent des séquences dans une série chronologique, ils ont tendance à ignorer les informations contextuelles futures. Une solution très évidente consiste à ajouter un délai entre l'entrée et la cible, ce qui donne au réseau le temps d'ajouter des informations de contexte futures, c'est-à-dire d'ajouter des informations futures sur la période M afin de prédire la sortie ensemble. L'idée de base du RNN bidirectionnel est de proposer

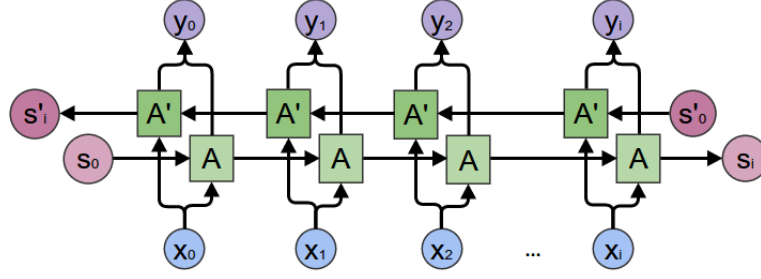


Figure 31 – Bi-directional Encoder

que chaque séquence d'entraînement constitue deux RNN aller-retour, et que les deux soient connectés à une couche de sortie. Cette structure fournit des informations complètes sur le contexte passé et futur pour chaque point de la séquence d'entrée de la couche en sortie.

17.4 Decoder

Ici, nous utilisons le Decoder en fonction du contenu de [cet article](#). nous définissons :

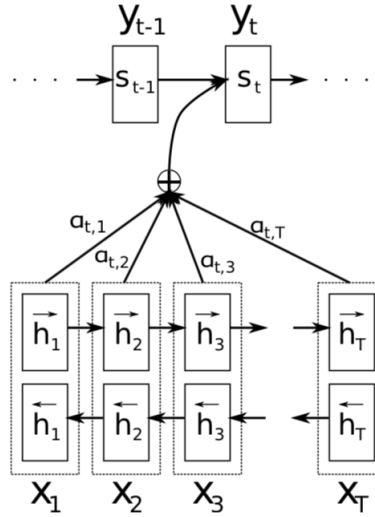


Figure 32 – Decoder

- ✓ s_t est un état caché RNN pour le moment t , $s \in \mathbb{R}^{1 \times n}$.
- ✓ c_t est un vecteur généré à partir de la séquence des états caché. Le vecteur de contexte c_t dépend d'une séquence d'annotations (h_1, \dots, h_{T_x}) à laquelle un encodeur mappe la phrase d'entrée, $c \in \mathbb{R}^{n \times 1}$.
- ✓ $\alpha_{tj} \in \mathbb{R}^{n \times n}$ est les poids de chaque h_j .
- ✓ $e_{tj} \in \mathbb{R}^{n \times n}$ est un modèle d'alignement qui indique dans quelle mesure les entrées autour de la position j et les sorties à la position i correspondent. Le score est basé sur l'état caché RNN s_{t-1} et la j -ème h_j de la phrase d'entrée.
- ✓ Nous paramétrons le modèle d'alignement a en tant que réseau de neurones à action directe formé conjointement avec tous les autres composants du système proposé.
- ✓ y_t est la sortie au moment de t $y \in \{0, 1\}^{n \times 1}$.

Nous avons :

$$c_t = \sum_{j=1}^{T_x} \alpha_{tj} h_{tj}$$

$$\begin{aligned}
 e_{tk} &= a(s_{t-1}, h_j) \\
 \alpha_{tj} &= \frac{\exp(e_{tj})}{\sum_{k=1}^{T_x} \exp(e_{tk})} \\
 s_t &= f(s_{t-1}, y_{t-1}, c_t) \\
 p(y_t | y_1, \dots, y_{t-1}, X) &= g(y_{t_1}, s_t, c_t)
 \end{aligned}$$

18 Loss Function

Dans notre projet, nous utilisons cette fonction de loss : categorical crossentropy.

$$L_i = - \sum_j \hat{y}_{i,j} \log(y_{i,j})$$

\hat{y} sont les prédictions, y sont les vraies valeurs, i désigne le point de données et j désigne la classe.

Mise en œuvre

19 Génération de la base d'apprentissage

Pour cette partie, nous avons déjà les outils :

- ✓ RBSflowshop.exe : lorsque nous entrons dans une instance, nous pouvons générer la première séquence d'origine.
- ✓ Programme f2SumCj : Ce programme implémente entièrement l'algorithme mentionné dans [cet article](#). Une séquence optimisée est donnée en fonction d'une séquence d'origine, mais où $h = 12$. Et la position de r est sélectionnée de manière aléatoire et le processus d'optimisation est limité par le temps CPU spécifié.

Tâches :

- ✓ Générer différentes instances.
- ✓ Modifier le deuxième programme pour implémenter l'algorithme de l'annexe A
- ✓ Exécuter le programme modifié dans le laboratoire ROOT et exécuter le fichier txt sur chacun des cinq ordinateurs
- ✓ Consolider les fichiers dans un fichier base.txt.

Configuration :

- ✓ Nombre de jobs : 100
- ✓ Durée de la génération : 1 mois
- ✓ Nombre d'ordinateurs en marche : 4
- ✓ CPU : Core i7

Résultats :

- ✓ Nous générons 4448 instances.
- ✓ Nous avons un total de 22 260 données.

20 Construction des réseaux de neurones

Afin de mettre en œuvre le modèle que nous avons construit précédemment, j'ai créé le diagramme UML suivant en fonction des besoins.

Je l'ai divisé en deux packages, l'un pour le prétraitement et l'autre pour Seq2SeqModel.

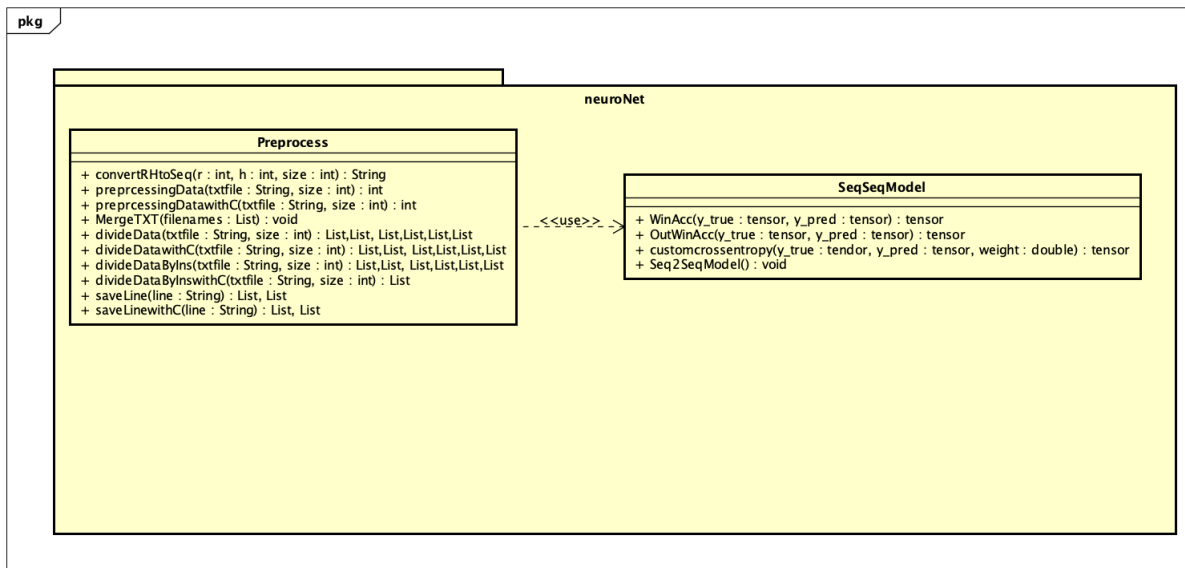


Figure 33 – UML des réseaux de neurones

20.1 Preprocess

Preprocess a les fonctions suivantes :

convertRHtoSeq

La fonction peut utiliser les nombres : r et h. et créer une séquence comme (0000011111).

- ✓ 0 : l'empoisonnement par la fenêtre
- ✓ 1 : la position dans la fenêtre
- ✓ param r : position de départ
- ✓ param h : taille des fenêtres
- ✓ param size : taille de la séquence
- ✓ return : la séquence binaire qui peut donner la position de la fenêtre (comme 00000111000).

preprocessingData

La fonction peut extraire les données du txt généré à partir du programme CPLEX et transformer en formulaire de données utilisable par le modèle Seq2Seq.

- ✓ param fichier txt : le chemin du fichier txt de la base de données.
- ✓ param size : taille de la séquence
- ✓ return : le nombre d'instances et le fichier csv pour le seq2seq.

preprocessingDatawithC

Function peut extraire les données du txt généré à partir du programme CPLEX, calculer le temps d'exécution de chaque travail et l'ajouter à la base de données, et transformer en formulaire de données utilisable par le modèle Seq2Seq.

- ✓ param txtfile : le chemin du fichier txt de la base de données.
- ✓ param size : taille de la séquence
- ✓ le nombre d'instances et le fichier csv pour le seq2seq

MergeTXT

La fonction peut fusionner différents fichiers texte en un seul fichier texte.

- ✓ param filenames : the list of different txt files generated by the CPLEX program.

divideData

La fonction peut diviser les données en 3 parties : ensemble d'entraînement, ensemble de test et ensemble de validation.

- ✓ param txtfile : le chemin du fichier txt de la base de données.
- ✓ param size : taille de la séquence
- ✓ return : X_train, y_train, X_test, y_test, X_validation, y_validation
- ✓ X_train : Ensemble d'apprentissage qui a la séquence initiale avec processing time.
- ✓ y_train : Ensemble d'apprentissage qui a la séquence binaire pouvant indiquer la fenêtre.
- ✓ X_test : Ensemble de test ayant la séquence initiale avec processing time.
- ✓ y_test : Ensemble de test ayant la séquence binaire pouvant indiquer la fenêtre.
- ✓ X_validation : Ensemble de validation contenant la séquence initiale avec processing time.
- ✓ y_validation : ensemble de validation ayant la séquence binaire pouvant indiquer la fenêtre.

divideDatawithC

La fonction peut diviser les données en 3 parties : ensemble d'entraînement, ensemble de test et ensemble de validation.

- ✓ param txtfile : le chemin du fichier txt de la base de données.
- ✓ param size : taille de la séquence
- ✓ return : X_train, y_train, X_test, y_test, X_validation, y_validation
- ✓ X_train : Ensemble d'apprentissage qui a la séquence initiale avec processing time et completion time.
- ✓ y_train : Ensemble d'apprentissage qui a la séquence binaire pouvant indiquer la fenêtre.
- ✓ X_test : Ensemble de test ayant la séquence initiale avec processing time et completion time
- ✓ y_test : Ensemble de test ayant la séquence binaire pouvant indiquer la fenêtre.
- ✓ X_validation : Ensemble de validation contenant la séquence initiale avec processing time et completion time
- ✓ y_validation : ensemble de validation ayant la séquence binaire pouvant indiquer la fenêtre.

saveLine

La fonction de complément pour les différents ensembles du modèle seq2seq

- ✓ param line : une ligne du fichier csv

- ✓ return : ptimes_list, solved_list.
- ✓ ptimes_list : la séquence avec processing time.
- ✓ solved_list : la séquence binaire pouvant indiquer la fenêtre.

saveLinewithC

La fonction de complément pour les différents ensembles du modèle seq2seq.

- ✓ param line : une ligne du fichier csv
- ✓ return : ptimes_list, solved_list.
- ✓ ptimes_list : la séquence avec processing time et completion time.
- ✓ solved_list : la séquence binaire pouvant indiquer la fenêtre.

20.2 Seq2SeqModel

Seq2SeqModel a les fonctions suivantes :

WinAcc

métrique personnalisée pour le modèle seq2seq, la prédiction pour la fenêtre.

- ✓ param y_true : la vraie cible.
- ✓ param y_pred : la prédiction.
- ✓ return : acc : l'exactitude de la prédiction pour la fenêtre.

OutWinAcc

métrique personnalisée pour le modèle seq2seq, la prédiction hors de la fenêtre. la valeur est entre 0 et 1, plus la valeur est basse, plus la précision est grande pour la fenêtre.

- ✓ param y_true : la vraie cible.
- ✓ param y_pred : la prédiction.
- ✓ return : acc : l'exactitude de la prédiction hors de la fenêtre.

customcrossentropy

Fonction de perte personnalisée pour le modèle seq2seq, c'est pour punir la prédiction de 1.

- ✓ param y_true : la vraie cible.
- ✓ param y_pred : la prédiction.
- ✓ param weight : poids de la punition.
- ✓ return : la valeur de la fonction de perte.

Seq2SeqModel

Function can input the data and train the model seq2seq.

21 Test

Nous ne savons pas si le modèle que nous avons construit est valide ou si notre problème peut être résolu avec ce modèle. Pour la robustesse et la convivialité du modèle seq2seq, nous devons tester que le modèle que nous avons construit peut prédire la meilleure fenêtre pour une séquence. Nous avons donc fait le test suivant.

Paramètres :

- ✓ nombre de neurones : 100
- ✓ profondeur : 1
- ✓ nombre de epoch par default : 5000
- ✓ loss fonction : categorical crossentropy
- ✓ adam : Cet algorithme est un moyen de calculer le taux d'apprentissage adaptatif pour chaque paramètre.

Nous avons fait beaucoup de tests pour tester notre modèle et nous analysons ici les trois tests les plus représentatifs.

21.1 Test scenario 1

Nous avons testé 16 données. Objectif : Le système peut bien apprendre 16 données.

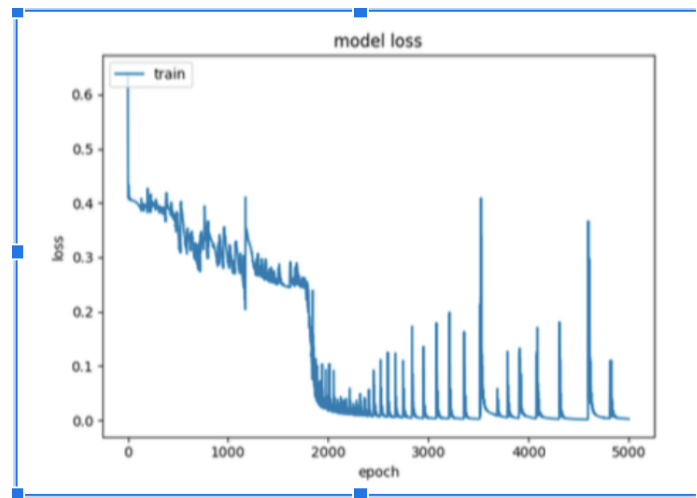


Figure 34 – Model loss pour 16 données.

Comme le montre la figure, notre fonction de perte a atteint 0 après 1000 epochs, ce qui indique que le modèle peut bien prédire ces 16 données. Comme on peut le voir sur la figure ci-dessus, la

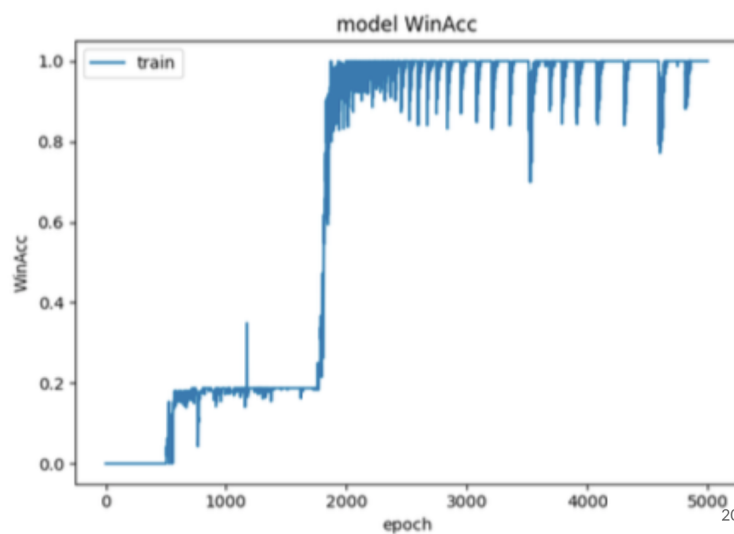


Figure 35 – WinAcc pour 16 données.

précision du modèle dans la fenêtre atteint 1, après 1800 epochs, ce qui indique que la structure de la fenêtre des 16 données peut être bien prédite.

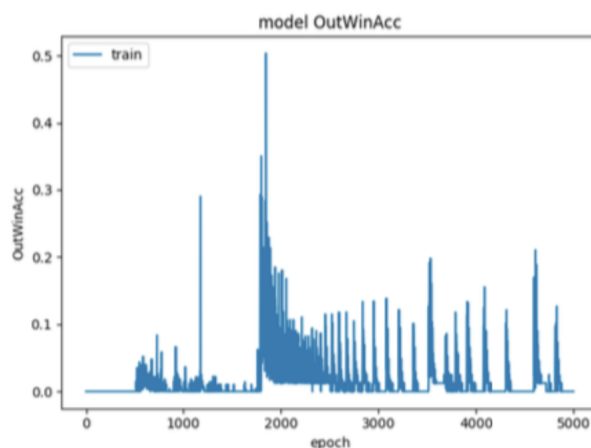


Figure 36 – *OutWinAcc pour 16 données.*

Comme le montre la figure ci-dessus, lorsque la valeur du modèle hors fenêtre augmente de 1 800 epochs, elle est nulle à 2 500 epochs, ce qui indique que la structure hors de la fenêtre des 16 données peut être bien prédite.

21.2 Test scenario 2

Selon notre protocole de test, nous savons que la base de validation est une norme permettant d'indiquer si un système peut fonctionner normalement. Il est donc primordial qu'un système formé puisse prédire avec précision des données inconnues, nous ajoutons donc 6 données de validation aux 16 ensembles de données.

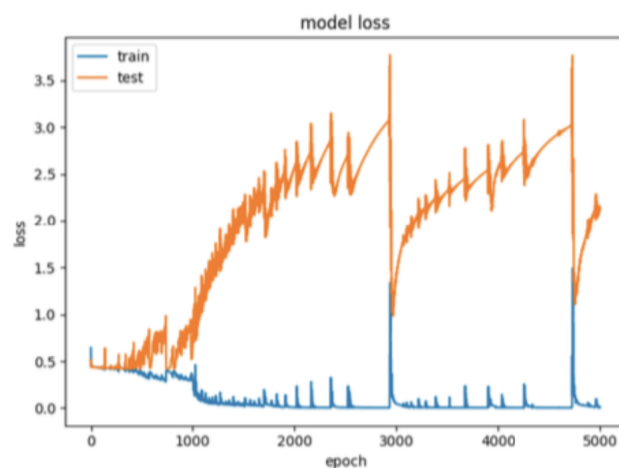


Figure 37 – *Model loss pour 16 données avec 6 données de validation.*

Nous pouvons voir que bien que la loss du ensemble d'entraînement soit déjà égale à 0 pour 1000 epochs, mais il y a beaucoup de flottement sur le ensemble de validation.

Nous pouvons voir que pour la précision dans la fenêtre, l'ensemble d'entraînement est 1 et l'ensemble de validation est d'environ 0,2.

Nous pouvons voir que pour la précision hors de la fenêtre, l'ensemble d'entraînement est 0 et l'ensemble de validation est d'environ 0,8.

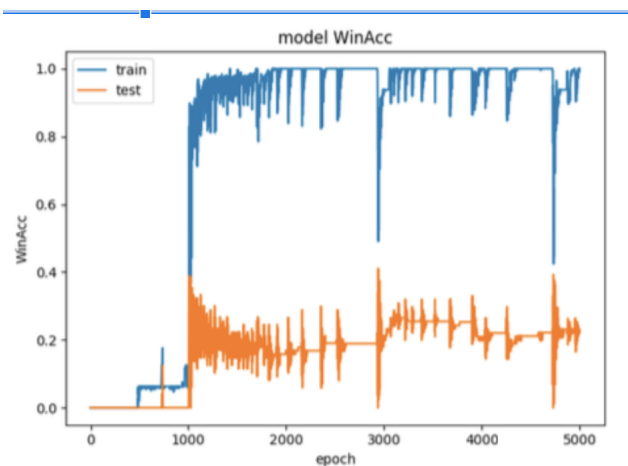


Figure 38 – WinAcc pour 16 données avec 6 données de validation.

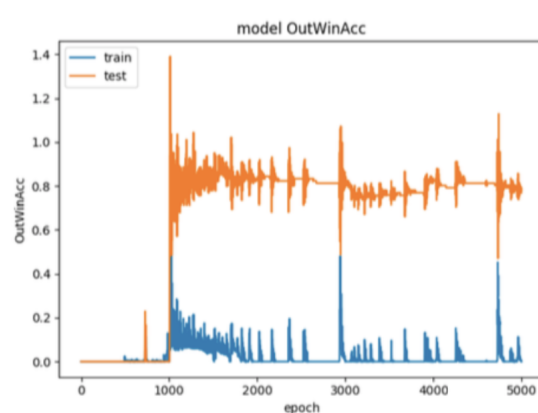


Figure 39 – OutWinAcc pour 16 données avec 6 données de validation.

21.3 Test scenario 3

Lorsque nous ajoutons le ensemble de validation, nous pouvons également faire des prédictions, après quoi nous voulons savoir si nous nous entraînerons mieux si nous ajoutons plus de données.

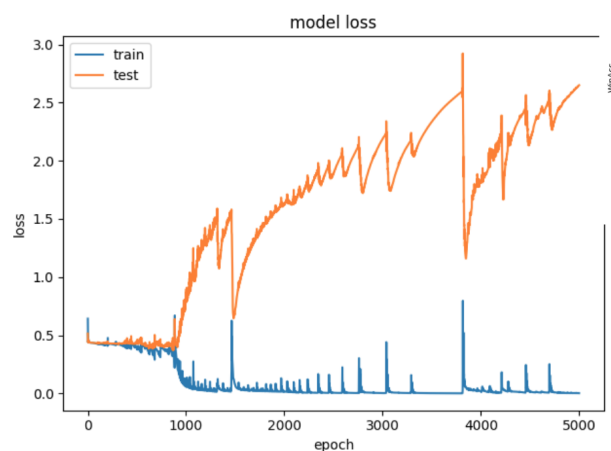


Figure 40 – Model loss pour 32 données avec 12 données de validation.

Nous pouvons voir que bien que la loss du ensemble d'entraînement soit déjà égale à 0 pour

1000 epochs, mais il y a beaucoup de flottement sur le ensemble de validation.

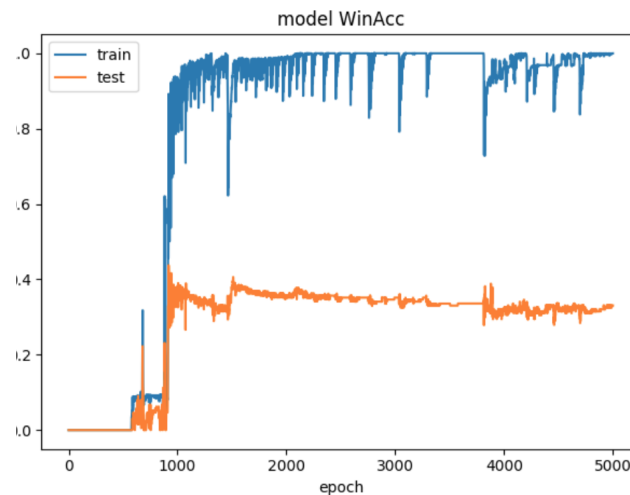


Figure 41 – WinAcc pour 32 données avec 12 données de validation.

Nous pouvons voir que pour la précision dans la fenêtre, l'ensemble d'entraînement est 1 et l'ensemble de validation est d'environ 0,3. Plus de 0,1 précision en comparant avec 16 données.

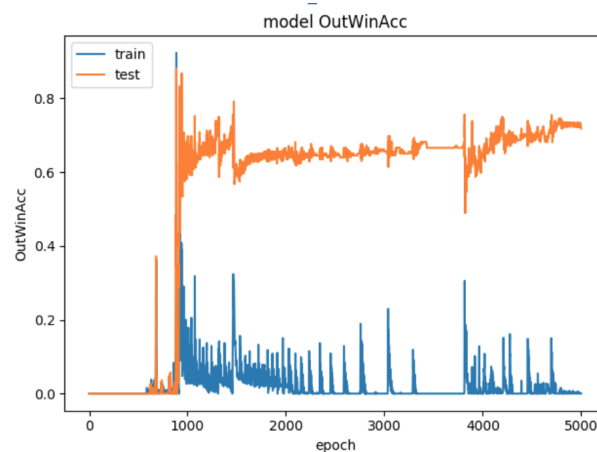


Figure 42 – OutWinAcc pour 32 données avec 12 données de validation.

Nous pouvons voir que pour la précision hors de la fenêtre, l'ensemble d'entraînement est 0 et l'ensemble de validation est d'environ 0,6. Moins de 0,2 précision en comparant avec 16 données.

22 Conclusion

Tâches terminées

1. Créer un formulaire d'étude
2. Établir un réseau de neurones et tester

Tâches inachevées

1. Entraîner le modèle avec une grande quantité de données : nous avons commencé la formation au début de mars, car la quantité de données est énorme et le nombre de compétences est de 50 000. Il n'est donc pas encore terminé. La durée de formation est estimée à 6 mois.
2. Construire une nouvelle application peut résoudre les problèmes de flowshop avec ce modèle
3. Tester la fonctionnalité de l'application

Hypothèse

1. La formation de gros volumes de données peut améliorer la précision du modèle à un certain niveau.
2. Le modèle peut résoudre efficacement le problème de la recherche des meilleurs r et h .

Plan de développement

23 Découpage du projet en tâches

23.1 Tache Étude

23.1.1 Étude Problème flowshop

Tâche 1 : Comprendre les problèmes de flowshop et les problèmes à résoudre

Description de la tâche

Selon le document donné par le professeur, comprendre le problème de flowshop et le problème que nous devons résoudre.

Estimation de charge

Cette tâche est estimée à 2 jour/homme.

Livrable

Un diaporama de compréhension du problème.

Tâche 2 : Présentation de compréhension

Description de la tâche

Après avoir terminé le travail du diaporama, prendre rendez-vous avec les professeurs et montrer et échanger des idées.

Estimation de charge

Cette tâche est estimée à 1 jour/homme.

23.1.2 Étude problème d'apprentissage automatique

Tâche 3 :Lecture et compréhension du réseau de neurones

Description de la tâche

Cette tâche vous aide à comprendre le projet et à définir sa portée. Lire les articles correspondants et découvrir les technologies correspondantes, telles que CNN, LSTM, Seq2Seq, etc.

Estimation de charge

Cette tâche est estimée à 10 jours/homme.

23.2 Réalisation

23.2.1 Réalisation d'écriture

Tâche 4 : Écrire la protocole pour générer la base d'apprentissage

Description de la tâche

Pour Entraîner le réseau de neurones, écrire une protocole pour générer la base d'apprentissage.

Estimation de charge

Cette tâche est estimée à 4 jour/homme.

Livrable

Une protocole pour générer la base d'apprentissage.

Tâche 5 : Rédaction la protocole pour générer la base d'apprentissage

Description de la tâche

L'importance de cette protocole étant très importante pour ce projet, il est nécessaire d'échanger des opinions avec l'enseignant et de modifier la protocole.

Estimation de charge

Cette tâche est estimée à 10 jour/homme.

Tâche 6 : Rédaction du cahier des spécifications

Description de la tâche

Cette tâche comprend la rédaction d'un cahier de spécification. Cela permet à la planification de définir le contexte, le périmètre, le système et ses fonctions, ainsi que l'avancement du projet.

Livrables

Un cahier de spécification sera à rendre.

Estimation de charge

Cette tâche est estimée à 6 jours/homme.

Tâche 7 : Échange du cahier des spécifications entre la MOA et la MOE*Description de la tâche*

Une fois le premier projet de spécification terminé, nous en discutons le contenu afin que les parties puissent parvenir à un accord.

Livrables

Un cahier de spécification final sera à rendre avant 9/12/2018. À cette livraison, la MOA et la MOE auront signé ce cahier.

Estimation de charge

Cette tâche est estimée à 1 jour/homme.

Contraintes temporelles

Le livrable devra être rendu au plus tard le 9/12/2018.

Tâche 8 : Préparation soutenance mi-parcours*Description de la tâche*

Préparation de la soutenance de mi-parcours

Livrable

Soutenance avec PowerPoint.

Estimation de charge

Cette tâche est estimée à 2 jour/homme.

Tâche 9 : Rédaction du rapport final*Description de la tâche*

Cette tâche consiste à la rédaction du rapport final afin de rendre compte du projet. Celui-ci pourra être réalisé tout au long du projet.

Estimation de charge

Cette tâche est estimée à 6 jours/hommes.

Livrable

Le livrable de cette tâche est le rapport.

Tâche 10 : Échange du rapport final entre la MOA et la MOE*Description de la tâche*

L'échange de vues sur le rapport final a lieu entre les professeurs.

Estimation de charge

Cette tâche est estimée à 1 jours/hommes

Tâche 11 : Préparation de la soutenance finale*Description de la tâche*

Cette tâche consiste à préparer la soutenance de ce PRD avec la réalisation d'un diaporama sur le projet et les résultats obtenus, ainsi qu'une éventuelle manipulation.

Estimation de charge

Cette tâche est estimée à 2 jours/hommes.

Livrable

Le livrable de cette tâche est le code source, les exécutables du projet et le PowerPoint de la soutenance.

23.2.2 Réalisation de programmation**Tâche 12 : Implémentation de la génération de la base d'apprentissage***Description de la tâche*

Construire une table d'apprentissage pour le réseau de neurones conformément au "Protocole pour la base d'apprentissage"

Estimation de charge

Cette tâche est estimée à 2 jour/homme.

Livrable

Un fichier .csv de la base apprentissage.

Tâche 13 : Construire un réseau de neurones.*Description de la tâche*

Afin de résoudre ce problème, un réseau de neurones basé sur RNN est construit avec Keras.

Estimation de charge

Cette tâche est estimée à 4 jour/homme.

Livrable

Le livrable de cette tâche sera un programme.

23.2.3 Réalisation de Test**Tâche 14 : Tests et correction du programme du réseau de neurones.***Description de la tâche*

Cette tâche consiste à effectuer de nombreux tests sur notre implémentation afin de vérifier le bon fonctionnement du programme.

Estimation de charge

Cette tâche est estimée à 4 jours/homme.

Livrable

Un guide d'utilisateur sera à fournir.

24 Planning**25 Bilan sur la qualité**

Je pense qu'il y a encore des perspectives pour la qualité du modèle, mais nous n'avons pas terminé l'entraînement du modèle, nous ne pouvons donc pas directement tirer de conclusions. Avec l'aide de M. Raveaux, nous avons beaucoup testé le modèle. Nous pouvons penser que ce modèle peut atteindre une prévision de qualité donnée, et cela peut également nous aider à intégrer le modèle à la prochaine étape pour résoudre le problème de flowshop. Utiliser le modèle expérimenté pour prédire la structure de la fenêtre et accélérer la solution en temps CPU.

Pour la qualité du code, j'ai eu une erreur dans le code qui a généré la base d'apprentissage. Je suis très reconnaissant à M.T'kint de l'avoir signalé et corrigé. Sans son aide, je ne pourrais pas continuer les étapes de l'entraînement.

26 Bilan auto-critique

Au cours de l'année, je suis très reconnaissant de l'aide et de la patience des trois enseignants. Avec l'aide de M. Raveaux, M.T'kint et M.Ragot, j'ai résolu beaucoup de problèmes. Chaque fois que nous avons une réunion, nous pouvons créer une collision de nouvelles idées, ce qui me donne confiance dans mon projet. En plus j'ai aussi beaucoup appris beaucoup de connaissance sur la Recherche Opérationnelle et le Deep Learning.

Je suis également très reconnaissant à M.Ramel pour ses précieux conseils et recommandations concernant mon cahier de spécifications et ma modélisation.

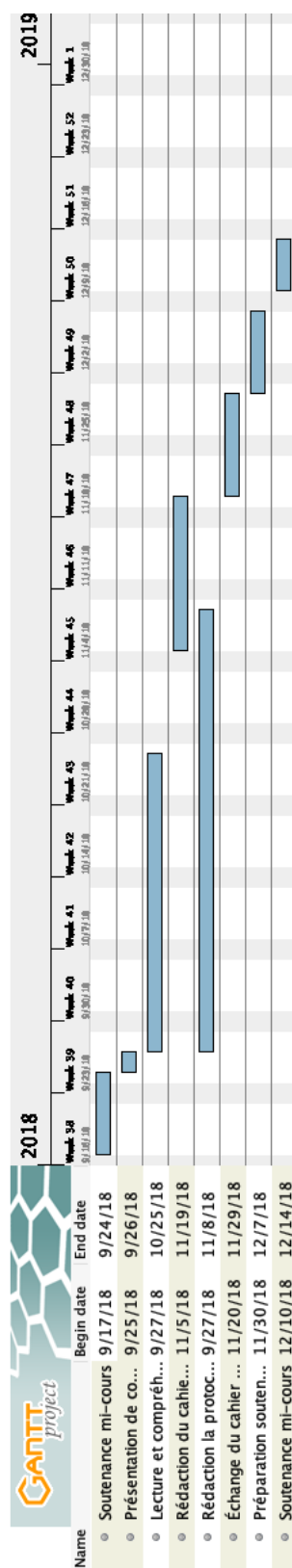


Figure 43 – Plan de développement pour S9

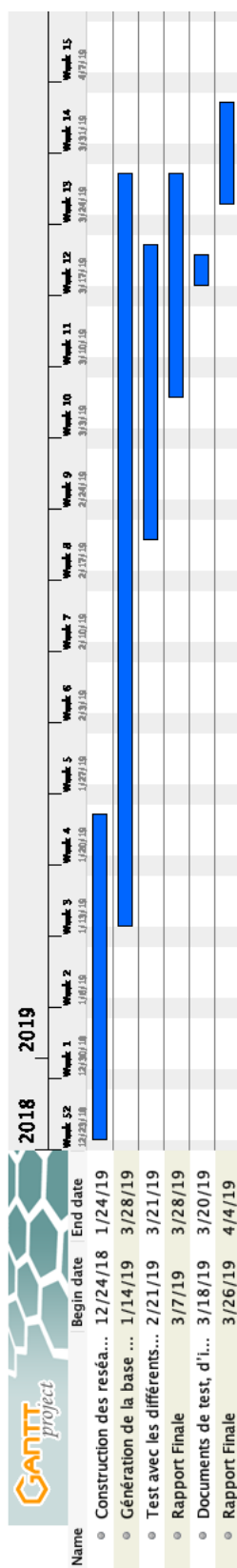


Figure 44 – Plan de développement pour S10

Annexes

A

Protocole pour générer la base d'apprentissage

L'extraction des données est divisée en deux phases :

- La table de séquences initiales générée par la méthode RBS (Recovering Beam Search).
- Effectuer plusieurs fois l'algorithme d'optimisation (avec contrainte W) avec la table de séquences initiales de RBS pour former une base d'apprentissage.

1 Préparation

Nous avons d'abord trois bases d'instances de n jobs(J_i) avec $n = 100, 300, 500$:

	M_1	M_2
J_1
...		
J_i	$P_{i,1}$	$P_{i,2}$
...		
J_n	$P_{n,1}$	$P_{n,2}$

$10 < P_{i,j} < 100, P_{i,j} \in F \quad j = 1, 2$ (nombre de machine)

Lorsque nous appliquons RBS à une instance, nous obtenons une séquence : S^i (séquence initiale)

2 Définition de base d'apprentissage (Learning database)

Notre objectif est de définir un algorithme pour obtenir une base d'apprentissage à partir de S^i de la préparation. Nous définissons donc d'abord le contenu de cette base :

Id	r	h	%I	%I'	S	S'
0

Id : Id de chaque ligne.

r : La position de l'exécution de l'algorithme d'optimisation sur une séquence S ($r \in [0; n - h]$)

h : La taille de la fenêtre d'optimisation ($h < n; r + h < n$)

%I : Le pourcentage d'amélioration par rapport à la séquence d'origine S^i .

%I' : Le pourcentage d'amélioration par rapport à S généré par l'algorithme Heuristique

S : Séquence avant d'optimisation(S^i ou départ)

S' : Séquence générée par l'algorithme Heuristique

3 Exécution d'algorithme

Name : Learning database generation

Inputs : Instance I, %I^e (the percentage of improvement that we expected), T(the limited working time), F(the number of iterations that we wanted), H(Heuristic Algorithm), S^i .

Pre-condition : 1. S^i generated by RBS in the first step

Output : Learning database

Post-condition : 1. The number of iterations reaches F 2. The working time reaches T

```

S ← Si;
f ← 0(the nb of execution);
t ← 0(the time that the algorithm spent);
tend ← 0;
tbegin ← current time;
while t ≤ T and f ≤ F do
  Sbest ← ∅;
  %Ibest ← 0;
  for r = 1 to |S| - 1 do
    for h = 1 to |S| - r do
      S' ← H(S, r, h)
      %I ← CalculateI(Si, S')
      %I' ← CalculateI(S, S')
      if %Ibest ≤ %I' then
        %Sbest ← S'
        %Ibest ← %I'
        Write (r, h, %I, %I', S, Sbest) to data table
      end if
    end for
  end for
  f ← f + 1
  tend ← current time
  t ← tend - tbegin
end while

```

Algorithm 1 : Learning database generation

Name : CalculateI

Input : S^{base} (Calculate the percentage based on S^{base}) S^{obj} (Calculate the percentage for S^{obj})

Output : %I (the percentage of improvement)

Calculate the completion time(for the last machine) S^{base}, S^{obj} correspond to S^{base}, S^{obj} respectively.
 $\%I \leftarrow 1 - (S^{obj}/S^{base})$

Algorithme 2 : CalculateI

B

Protocole de test

Notre test est divisé en deux parties, la première est le test de notre réseau de neurones et la deuxième partie, le test de notre programme de résolution du problème de flowshop.

1 Test pour le réseau de neurones

Introduction

Lors de la formation et de l'apprentissage de réseaux de neurones, il est possible que l'apprentissage de réseaux de neurones ne soit pas efficace en raison de divers problèmes, ou qu'il y ait trop d'interférences, il existe de nombreuses raisons à ces problèmes, qui peuvent être des problèmes de données et une efficacité d'apprentissage. Problèmes avec les paramètres. Pour résoudre ces problèmes, nous divisons généralement les données en données d'apprentissage (60%) , données de test (20%) et de données de validation(20%).

Courbe d'erreur

Nous pouvons tester notre réseau de neurones en traçant une courbe d'erreur. L'apprentissage automatique peut partir de la valeur de l'erreur. À mesure que le temps d'entraînement s'allonge, l'excellent réseau de neurones permet de prédire une réponse plus précise et l'erreur de prédiction sera plus grande. Moins : au final, l'espace pouvant être amélioré devient plus petit et la courbe tend à être horizontale.

Validation croisée

La validation croisée est utilisée pour évaluer les performances prédictives du modèle, en particulier les performances du modèle formé sur les nouvelles données, ce qui peut réduire dans une certaine mesure les surajustements.

Méthode

Dans la tâche d'apprentissage automatique, après avoir obtenu les données, nous divisons d'abord le jeu de données d'origine en trois parties : l'ensemble d'apprentissage, l'ensemble de vérification et l'ensemble de test. L'ensemble d'apprentissage est utilisé pour entraîner le modèle, et l'ensemble de validation est utilisé pour la configuration de sélection des paramètres

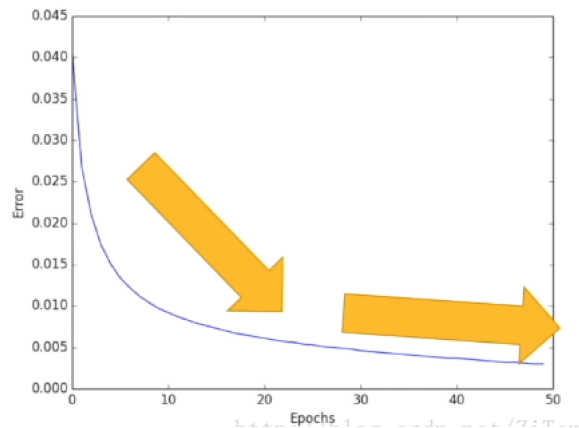


Figure 1 – Courbe d'erreur

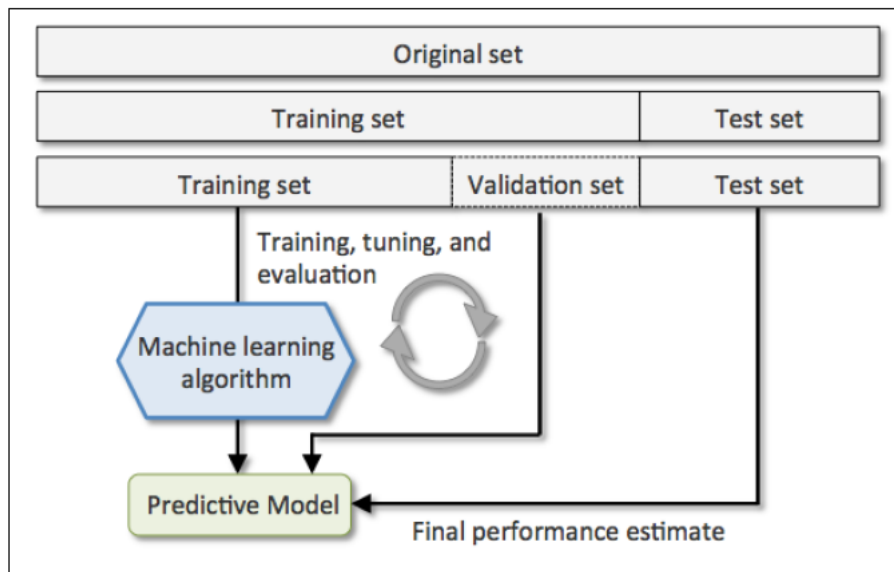


Figure 2 – Validation croisée

du modèle, l'ensemble de test est une donnée inconnue pour le modèle et permet d'évaluer la capacité de généralisation du modèle.

Ici, nous utiliserons 20% des données de base d'apprentissage comme ensemble de validation. Nous pouvons utiliser ces deux méthodes pour tester notre réseau de neurones afin de tester ses performances.

2 Test pour la résolution du problème de flowshop

Cette section est le critère le plus important pour l'éligibilité de notre projet. Notre idée de base est de comparer les deux résultats. Les deux résultats sont les suivants : 1. Calculer la fonction objective de la séquence de résultat final en utilisant la méthode mentionnée dans l'article. 2 Utilisez notre réseau de neurones pour dériver une séquence ou r , h , ainsi que la valeur de la fonction objective de cette séquence. Le processus de test est le suivant :

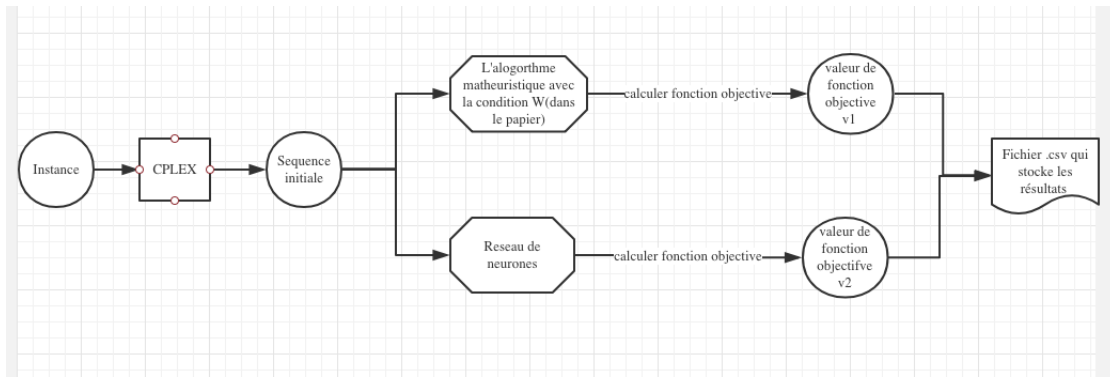


Figure 3 – Le processus de test

Ici, nous stockons les résultats des deux méthodes dans un fichier csv. Les documents sont organisés comme suit :

Ins	MATHEUR	NN
0

Ins : le nombre d'instances MATHEUR : la valeur de fonction objective par la méthode d'algorithme matheuristique dans le papier.

NN : la valeur de fonction objective par le réseau de neurones.

Nous devons écrire différents fichiers .csv pour les différents nombres de jobs.

En comparant cela, nous pouvons obtenir les performances spécifiques de notre programme. Les deux méthodes peuvent se comporter différemment selon le nombre de jobs et le nombre d'instances.



Guide Developement et utilisation

1 Téléchargement du projet

Nous pouvons télécharger le projet en github : <https://github.com/AlafateABULIMITI/PRD>.
Nous utilisons la commande *git clone* pour télécharger le projet dans un dossier local.

2 Libraires

1. Ici, nous utilisons beaucoup de libraires, voici une liste :

numpy

pandas

matplotlib

keras

tensorflow

recurrentshop

seq2seq

sklearn

2. Pour installer les libraires suivantes, nous utilisons l'outil pip. Voici les commandes d'installation pour chaque libraire.

numpy : pip install numpy

pandas : pip install pandas

matplotlib : pip install matplotlib

keras : pip install keras

tensorflow : pip install tensorflow

sklearn : pip install sklearn

recurrentshop :

“ git clone https://www.github.com/farizrahman4u/recurrentshop.git cd recurrentshop python setup.py install “

seq2seq : sudo pip install git + https://github.com/farizrahman4u/seq2seq.git

Important : pour utiliser ce modele dans notre projet, il faut modifier le source code dans le fichier python3.6/site-packages/seq2seq/cell.py. line 45 et 105 à

“python y = Activation('softmax')(W2(h)) “

WEBOGRAPHIE

- (1) Artificial_neural_network : https://en.wikipedia.org/wiki/Artificial_neural_network
- (2) Convolutional_neural_network : https://en.wikipedia.org/wiki/Convolutional_neural_network
- (3) Recurrent_neural_network : https://en.wikipedia.org/wiki/Recurrent_neural_network
- (4) Long short-term memory (LSTM): https://en.wikipedia.org/wiki/Long_short-term_memory

BIBLIOGRAPHIE

- (1) A matheuristic approach for the two-machine total completion time flow shop problem (Federico Della Croce · Andrea Grosso · Fabio Salassa) Published online: 6 July 2011 ©Springer Science+Business Media, LLC 2011
- (2) NEURAL COMBINATORIAL OPTIMIZATION WITH REINFORCEMENT LEARNING Irwan Bello*, Hieu (Pham*, Quoc V. Le, Mohammad Norouzi, Samy Bengio Google Brain) ICLR 2017
- (3) Pointer Network Google Brain (<https://arxiv.org/abs/1506.03134>)
- (4) Mastering the game of Go without human knowledge (David Silver1*, Julian Schrittwieser1*, Karen Simonyan1*, Ioannis Antonoglou1, Aja Huang1, Arthur Guez1, Thomas Hubert1, Lucas Baker1, Matthew Lai1, Adrian Bolton1, Yutian Chen1, Timothy Lillicrap1, Fan Hui1, Laurent Sifre1, George van den Driessche1, Thore Graepel1 & Demis Hassabis1) (https://deepmind.com/documents/119/agz_unformatted_nature.pdf)

Recherche Opérationnelle et Machine Learning : le cas d'une matheuristique pour un problème de flowshop

Résumé

Ce projet est le projet de fin d'études de la dernière année de la cycle d'ingénieur, qui a une signification très importante: ce projet intègre toutes les connaissances et la technologie acquises grâce à la formation en cinq ans et met toutes les réalisations en pratique.

Le projet associe recherche opérationnelle et apprentissage automatique. Plus précisément, il utilise le modèle d'apprentissage en profondeur pour résoudre le problème très classique de FlowShop dans la recherche opérationnelle. Dans [cet article](#), une optimisation de l'algorithme d'origine est proposée, mais les paramètres de cet algorithme d'optimisation sont des paramètres empiriques. Notre objectif est de créer une base d'apprentissage à l'aide des outils existants et d'utiliser cet base pour entraîner le modèle Seq2Seq, conçu pour améliorer l'algorithme en prévoyant avec précision la combinaison optimale de paramètres mentionnés dans l'article.

Mots-clés

Deep Learning, Recherche Opérationnelle, Seq2Seq, Flowshop

Abstract

This project is the end of studies project of the last year of the engineering cycle, which has a very important meaning: this project integrates all the knowledge and technology acquired through the training in five years and puts all the achievements in practice.

The project combines operational research and machine learning. Specifically, he uses the deep learning model to solve the classic FlowShop problem in operations research. In [this article](#), an optimization of the original algorithm is proposed, but the parameters of this algorithm d optimization are empirical parameters. Our goal is to create a learning base using existing tools and to use this base to train the Seq2Seq model, designed to improve the algorithm by accurately predicting the optimal combination of parameters mentioned in the article.

Keywords

Deep Learning, Operational research, Seq2Seq, Flowshop

Tuteurs académiques

Vincent T'KINDT
Romain RAVEAUX
Nicolas RAGOT

Étudiant

Alafate ABULIMITI (DI5)