Faculté des sciences

Département d’informatique

**Mémoire de projet de fin d’études**

Pour l’obtention du titre

**D’Ingénieur d’Etat en Informatique**

**Filiere:** Master Big Data & cloud computing

**Sujet :**

How Machine Learning (or Deep Learning Models) and Neural Network Help in Routing in VRP Network

**Realisé par:** Doha SASSA **Encadrante:** Pr. Mme Hala KHANKHO

**Coordinateur** : Pr. JAAFAR ABOUCHABAKA

**Membre de jury :** Mr. Jaafar ABOUCHABAKA

Mme Najat Rafalia

Année universitaire : 2022-2023

***Remerciement***



Au terme de ce projet de fin d’études, il m’est très agréable de remercier toutes les personnes qui ont contribué, de près ou de loin, à la réalisation de ce modeste travail et je leur demande de bien vouloir m’excuser de ne pas pouvoir les citer toutes et qu’ils sachent que je leur dois ma sincère gratitude. Mes sincères remerciements vont au professeur Hala Khankhour, mon encadrante, qui n’a ménagé aucun effort pour m’apporter assistance durant toutes les étapes de mon travail. Ses remarques, ses qualités d’enseignant et sa rigueur m’ont été d’un apport considérable. Ainsi, il m’a offert la latitude de travailler très librement sur ce projet et m’a accordé, tout au long de la période de ce travail, le temps nécessaire pour me renseigner, me conseiller, m’orienter et m’acquitter les réponses nécessaires à mes interrogations multiples. Avec elle, j’ai pu goûter aux plaisirs de la mise en pratique de mes connaissances théoriques. Si j’ai pu réaliser ce travail dans les meilleures conditions, c’est grâce à sa disponibilité malgré sa grossesse. Que Dieu soit à ses rangs.



**Abstract**

The global conception of sustainable development includes a mechanism for achieving the economic goal by which all fundamental human needs are met by the use of technology with simultaneously taking care of our natural environment. To reduce air pollution and traffic we need to begin with optimizing the supply chain processes that occur on our globe every day and night. This purpose also serves every citizen by reducing everyday traffic congestions.

This project focuses on solving a variation of the vehicle routing problem (VRP) in which a time window is associated with each customer service and some services require simultaneous visits to different vehicles to be accomplished. The problem is therefore called VRP with time windows and synchronization constraints .We present a simulated annealing (SA) algorithm that integrates several local search techniques to address this problem. Experiments on literature examples show that our SA is fast and outperforms existing approaches. To our knowledge, this is the first time that dedicated local search. We present a CNN (conventional neural network) algorithm to detect traffic congestion patterns, customer groupings, low demand areas, this information can be used to generate routes and travel times and resource allocations. This model can be used to classify vehicle types according to their capacity constraints. We propose a multi-agent architecture to guarantee the autonomy of our program, flexibility and adaptability, parallelism and efficiency. To reduce program execution time by parallelizing operations, we will use CUDA Framework. A GPU can greatly improve performance by performing all of these operations simultanement.

**Keywords:** vehicle routing problem, machine learning, deep learning, multi-agent system, CNN, JAVA, JADE agent.

**Liste des abréviations**

|  |  |
| --- | --- |
| VRP | Vehicle Routinng Problem |
| TSP | Traveling System Problem |
| QOT | Quality of traveling |
| its | Intelligent transprtation system |
| SOC-VRP | Service oriented cooperative |
| CNN | Convolutional neural network |
| SA | Simulated anealing |
| JADE | Java agent development framwork |
| SMA | System multi agent |
| GPU | Graphics processing Unit |
| CUDA | Compute Unified Device Architecture |

**Table de matières**

**Remerciement**2

**Abstract**…………………………………………………………………………………………………………………………………………….3

**Liste des abréviations**……………………………………………………………………………………………………………………….4

**Introduction général**……………………………………………………………………………………………………………………..6-7

**I-Etat de l’art8**

Introduction8

* 1. C’est quoi le VRP ?.....................................................................................................................8

1.1.2. Problème de voyageur de commerce (TSP)…………………………………………………………………8

1.1.3. Pourquoi est-ce un tel défi le VRP……………………………………………………………………………….9

1.2. L’AI, l’apprentissage par machine et deep learning…………………………………………………………….10

1.2.1. Introduction au réseau de neurones à convolution………………………………………………10-11

1.2.2 Convolution Neural Network……………………………………………………………………………………11

* + - 1. Architecture CNN……………………………………………………………………………………………12
      2. Fonctionnement des couches convolutives……………………………………………….13-14
      3. Couches utilisées pour construire ConvNets………………………………………………14-15
      4. Avantage des réseaux de neurones convolutifs (CNN)…………………………………...15
      5. Inconvénients des réseaux de neurones convolutifs (CNN)…………………………….16

Conclusion………………………………………………………………………………………………………………………….....16

* 1. Algorithme recuit simulé (SA)………………………………………………………………………………………….16

1.3.1 C’est quoi l’algorithme recuit simulé ?......................................................................16

1.3.2 Définir le problème………………………………………………………………………………………………….17

1.3.3 Définir la fonction de perturbation………………………………………………………………………….17

1.3.4 Critère d'acceptation………………………………………………………………………………………….17-18

1.3.5 Calendrier des températures…………………………………………………………………………………..18

1.3.6 Exécuter l’algorithme SA…………………………………………………………………………………….18-19

1.3.7 Le diagramme de flux de SA…………………………………………………………………………………….19

1.3.8 Exemple d’application……………………………………………………………………………………………..20

Conclusion……………………………………………………………………………………………………………………………..20

**II-Expérimental21**

1. Choix d’algorithm :7

1.1. Convolutional neural network (CNN):8

1.2. Simulated annealing (SA)……………………………………………………………………………………………………..9

2. Environment de travail:…………………………………………………………………………………………………………….10

2.1. Python:………………………………………………………………………………………………………………………………11

2.2. JAVA…………………………………………………………………………………………………………………………………..12

2.3. JAVA AGENT DEVELOPMENT FRAMWORK (JADE)………………………………………………………………13

**Introduction général**

Grâce aux nouvelles technologies de l'information, le système de transport est devenu plus intelligent, intégrant des capacités puissantes telles que la perception approfondie de l'environnement de circulation, la conduite autonome, la communication V2X et la coordination. Avec l'évolution et le déploiement rapides de ces technologies, il passe d'un système de transport intelligent (ITS) à une nouvelle étape connue sous le nom d’ITS coopératif (C-ITS). Dans le contexte de cette tendance émergente, les défis dominants liés aux embouteillages, aux véhicules d'urgence bloqués et au gaspillage d'énergie dans les systèmes de transport conventionnels sont progressivement résolus avec des solutions de plus en plus intelligentes. Cela a suscité une attention considérable et suscité un intérêt considérable dans les domaines universitaire et industriel.

Comme chacun le sait, le problème d’acheminement des véhicules (VRP) est l’un de ces problèmes clés qui exerce un impact profond sur les systèmes de transport. Fondamentalement, de tels problèmes de planification d'itinéraire pour un seul véhicule peuvent toujours être assimilés à un problème spécial de voyageur de commerce (TSP) avec des contraintes de priorisation, couvrant la distance la plus courte, les autoroutes et les émissions minimales, etc. Cependant, ce problème deviendra plus compliqué lors de la planification des itinéraires. Pour plusieurs véhicules dans un réseau de circulation, en particulier si ces véhicules ont des attributs de service différents, par exemple, secours d'urgence, transports en commun et usage privé, comme le montre la figure 1. Dans ce contexte particulier, la résolution de ce nouveau VRP nécessite la prise en compte non seulement des contraintes imposées aux véhicules individuels, mais aussi des interdépendances découlant de la planification d'itinéraires multi véhicules et des besoins de conduite distinctifs associés aux différents types de véhicules de service. Il est évident que les fonctionnalités coordonnées et orientées services d’un tel nouveau VRP posent inévitablement de nouveaux défis aux approches de solutions conventionnelles. Heureusement, les progrès continus des technologies de l’information et de l’intelligence artificielle (IA) permettent de résoudre de plus en plus efficacement ce problème et en ont récemment fait un point chaud de la recherche.

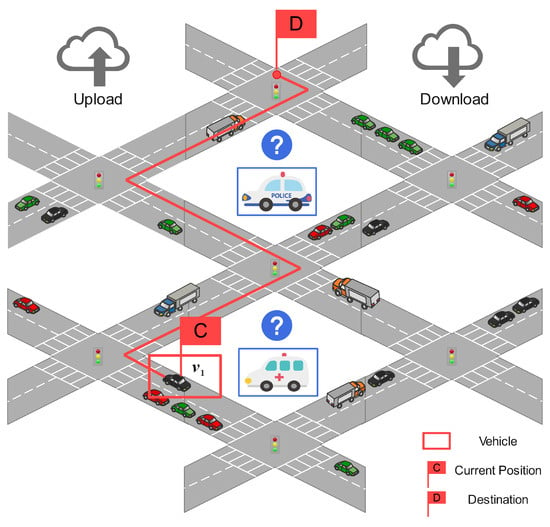


Figure 1: Service-oriented. Cooperative VRP in C-ITS.

En règle générale, le VRP est considéré comme une variation du problème des vendeurs de commerce, et les solutions pour le VRP sont presque entièrement dérivées de solutions pour le TSP. Par conséquent, plusieurs algorithmes statiques classiques, couvrant les algorithmes de Dijsktra, Floyd et Bellman–Ford, ont été largement adoptés en raison de leur bonne compréhensibilité. Bien que ces méthodes soient simples et efficaces, les lacunes de ces algorithmes deviennent assez évidentes, y compris les processus informatiques complexes et les résultats optimaux locaux. Pour améliorer l’efficacité des déplacements, au cours des dernières décennies, plusieurs mécanismes heuristiques et dynamiques de planification des itinéraires ont été proposés, explorant rapidement les arbres aléatoires, l’optimisation des colonies de fourmis et les algorithmes évolutifs. En outre, diverses méthodes d’IA ont également été introduites pour résoudre ce problème, y compris les réseaux de neurones, les arbres de décision flous, l’apprentissage par renforcement et l’apprentissage par renforcement profond.

Comme mentionné ci-dessus, les véhicules avec des propriétés de service différentes nécessiteront toujours différentes qualités de déplacement (QoT), ce qui est un facteur clé qui doit être pris en compte dans les systèmes de transport réels. Concrètement, les véhicules d’urgence, qui ont le QoT le plus élevé, devraient subir des retards de déplacement minimes dans la mesure du possible, tandis que les navettes et les véhicules privés nécessitent respectivement un QoT moyen et le QoT le plus bas. Dans de telles situations axées sur le service, nous avons introduit le terme ITS coopératif axé sur le service (SoC-ITS), et le VRP devient un VRP coopératif axé sur le service (SoC-VRP). D’après la documentation disponible, le EVRP a attiré l’attention au cours des dernières décennies. Pour planifier une voie d’urgence.

Dans ce contexte les travaux menés dans ce mémoire portent sur la présentation d'un algorithme de recuit simulé (SA) qui intègre plusieurs techniques de recherche locales pour résoudre ce problème. Les expériences sur des exemples de littérature montrent que notre (SA) est rapide et surpasse les approches existantes. À notre connaissance, c’est la première fois que la recherche locale dédiée. Nous présentons un algorithme CNN (réseau de neurones conventionnel) pour détecter les modèles de congestion du trafic, les groupements de clients, les zones à faible demande, ces informations peuvent être utilisées pour générer des itinéraires et des temps de déplacement et des allocations de ressources. ce modèle permet de classer les types de véhicules en fonction de leurs contraintes de capacité. Nous proposons une architecture multi-agents pour garantir l’autonomie de notre programme, la flexibilité et l’adaptabilité, le parallélisme et l’efficacité. Pour réduire le temps d’exécution du programme en parallélisant les opérations, nous utiliserons CUDA Framework. Un GPU peut grandement améliorer les performances en effectuant toutes ces opérations simultanément.

**L’état de l’art**

**Introduction**

Ce premier chapitre situe le travail sur comment l’apprentissage automatique (ou les modèles d’apprentissage profond) et le réseau neuronal aident au routage dans le réseau VRP. L’objectif de ce mémoire est la résolution d’une variante du problème de routage du véhicule (VRP) dans laquelle une fenêtre de temps est associée à chaque service client et certains services nécessitent des visites simultanées à différents véhicules pour être effectués. Le problème est donc appelé VRP avec des fenêtres de temps et des contraintes de synchronisation.

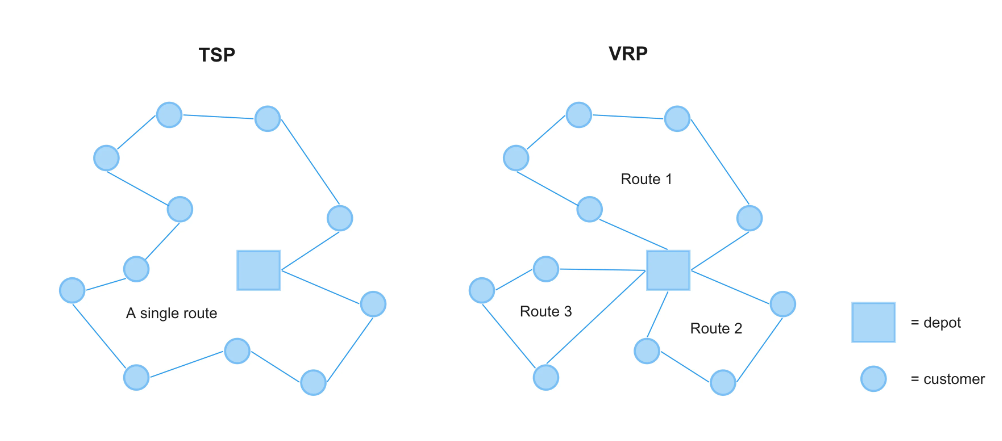
* 1. **C’est quoi le VRP ?**

La plupart des gens ont une compréhension de base du problème de routage du véhicule. Essentiellement, il s’agit de déterminer les itinéraires optimaux d’un dépôt à un ensemble de clients géographiquement dispersés afin de répondre aux exigences connues des clients. L’objectif principal du VRP est de minimiser les coûts de distribution, mais la complexité de la résolution de ce problème est souvent sous-estimée.

Le VRP n’est pas un phénomène nouveau puisqu’il remonte au milieu du XXe siècle. Il est apparu pour la première fois dans un article écrit par George Dantzig et John Ramser en 1959, dans lequel ce problème algorithmique a été appliqué pour la première fois dans le contexte de la livraison d’essence. À l’époque, les mathématiciens appliquaient des algorithmes d’optimisation d’itinéraires calculés manuellement pour trouver les itinéraires les plus optimaux pour un ensemble de véhicules. De nos jours, la technologie fournit un soutien considérable dans ce domaine sous la forme d’un logiciel d’optimisation des itinéraires. Grâce à des algorithmes à grande vitesse, les prestataires logistiques sont en mesure d’optimiser leur VRP en quelques minutes.

* + 1. **Problème de voyageur de commerce (TSP)**

Un exemple classique et bien connu d’un VRP est le problème des Voyageur de commerce (TSP). TSP se préoccupe de trouver le chemin le plus court qu’un vendeur ou un véhicule peut prendre, compte tenu d’une liste de destinations spécifiques qui ne peuvent être visitées qu’une seule fois. Remarquez que lorsqu’on parle de TSP, il n’y a qu’un seul véhicule dans l’équation. Dès que plusieurs véhicules entrent en jeu, cela devient un VRP. Ceci est illustré ci-dessous.



**Figure 2 :** TSP vs VRP



**Figure 3 :** TSP pose la question suivante : Étant donné une liste de villes et les distances entre chaque paire de villes, quel est le chemin le plus court possible qu’un vendeur peut prendre pour visiter chaque ville et retourner à la ville d’origine?

* + 1. **Pourquoi est-ce un tel défi de résoudre le VRP ?**

**Capacité:**  
Chaque véhicule a une capacité de transport, ce qui signifie qu’il ne peut transporter qu’un certain nombre d’articles à la fois, sans dépasser le volume et le poids seuil. L’objectif est de créer un itinéraire qui permet à un véhicule de ramasser / livrer la quantité maximale au coût le plus bas dans la capacité disponible. Le niveau de complexité peut varier considérablement d’une entreprise à l’autre. Alors qu’un fournisseur de logistique générale pouvait transporter des marchandises allant de la nourriture à l’électronique et au mobilier, un détaillant de mode ne transporte généralement que des vêtements. Pour le fournisseur logistique, le même camion peut avoir des spécifications de capacité différentes selon le type de marchandise transportée. Le détaillant de mode, d’autre part, peut n’avoir qu’une seule catégorie, ce qui rend les spécifications de capacité beaucoup plus simplifiées.  
  
**Fenêtres temporelles:**  
Les fenêtres de temps indiquent quand chaque visite peut être effectuée et ajoutent donc une autre couche de complexité au puzzle. Chaque client et dépôt ont souvent leurs propres heures d’ouverture spécifiques qui varient tout au long de la semaine, qui doivent toutes être prises en compte lors de la résolution d’un VRP réel.  
  
**Ramassage et livraison**:  
Un VRP de ramassage et de livraison survient lorsqu’un véhicule doit ramasser des marchandises ou des personnes à divers endroits et les déposer à plusieurs points de destination. Le principal défi consiste à combiner de manière optimale le transport de toutes les marchandises (ou personnes) tout en minimisant la longueur totale de l’itinéraire. Dans la plupart des cas, les ramassages et les livraisons se font de façon séquentielle, ce qui signifie qu’il n’y a pas de poste de facteurs en cause.

**Contraintes de ressources:**

Les ressources sont limitées, mais très nécessaires pour la livraison des marchandises. Votre problème de routage de véhicule devra donc toujours prendre en compte le nombre de conducteurs / véhicules disponibles et les heures de travail, afin de parvenir à une solution réaliste.

**Conclusion :**

En conclusion, le VRP est un problème algorithmique extrêmement complexe qui a fait l’objet de nombreuses études depuis plus de 60 ans. Bien que ce ne soit pas un phénomène nouveau, trouver une solution optimale reste un défi. Les réseaux de distribution d’aujourd’hui sont très complexes et les prix du carburant montent en flèche, ce qui rend essentiel pour les entreprises de rechercher des moyens plus efficaces de résoudre le VRP. Le logiciel d’optimisation des itinéraires offre aux fournisseurs logistiques d’aujourd’hui la possibilité d’utiliser leurs ressources au maximum, ce qui leur permet d’économiser du temps, de l’argent et des émissions inutiles.

### **1.2 L'IA, l'apprentissage par machine et le Deep Learning**

Avant de vous plonger dans le fonctionnement des CNN, il est important de comprendre comment ces algorithmes de deep learning se rapportent au domaine plus large de l'IA et les distinctions entre les termes clés couramment utilisés dans ce domaine.

* **L'intelligence artificielle** **:** le domaine de l'informatique axé sur les programmes informatiques intelligents capables de sentir, de raisonner, d'agir et de s'adapter.
* **L'apprentissage par machine** **:** un sous-ensemble de l'IA dans lequel les algorithmes peuvent améliorer en performance au fil du temps lorsqu'ils sont mis en présence de plus de données.
* **Le réseau neuronal** **:** une série d'algorithmes utilisés comme processus dans l'apprentissage par machine qui peut reconnaître les signaux et des relations dans de grandes quantités de données. Les réseaux neuronaux utilisent une structure logique inspirée du cerveau humain et constituent la base des algorithmes pour le deep learning.
* **Le Deep learning :**un sous-domaine de l'apprentissage par machine dans lequel les réseaux neuronaux multicouches apprennent à partir de grandes quantités de données.

**1.2.1 Introduction au réseau de neurones à convolution**

Un réseau de neurones convolutionnel (CNN) est un type d’architecture de réseau de neurones à apprentissage profond couramment utilisé en vision par ordinateur. La vision par ordinateur est un domaine de l’intelligence artificielle qui permet à un ordinateur de comprendre et d’interpréter les données visuelles ou d’image.

En ce qui concerne le Machine Learning, les réseaux de neurones artificiels fonctionnent très bien. Les réseaux neuronaux sont utilisés dans divers ensembles de données comme les images, l’audio et le texte. Différents types de réseaux de neurones sont utilisés à différentes fins, par exemple pour prédire la séquence de mots que nous utilisons Réseaux de neurones récurrents plus précisément un LSTM, de même pour la classification des images, nous utilisons des réseaux de neurones de convolution. Dans ce blog, nous allons construire un bloc de base pour CNN.

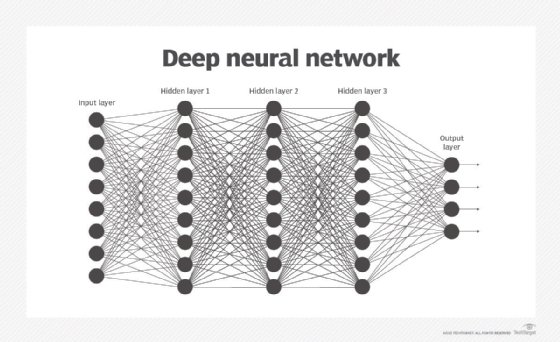
Dans un réseau neuronal ordinaire, il existe trois types de couches :

**1-Couches d’entrée :** C’est la couche dans laquelle nous donnons une entrée à notre modèle. Le nombre de neurones dans cette couche est égal au nombre total de caractéristiques dans nos données (nombre de pixels dans le cas d’une image).

**2-Couche cachée:** L’entrée de la couche d’entrée est ensuite introduite dans la couche cachée. Il peut y avoir de nombreuses couches cachées en fonction de notre modèle et de la taille des données. Chaque couche cachée peut avoir des nombres différents de neurones qui sont généralement plus grands que le nombre de caractéristiques. La sortie de chaque couche est calculée par multiplication matricielle de la sortie de la couche précédente avec des poids pouvant être définis pour cette couche, puis par l’ajout de biais exploitables suivis d’une fonction d’activation qui rend le réseau non linéaire.

**3-Couche de sortie:** La sortie de la couche cachée est ensuite introduite dans une fonction logistique comme sigmoid ou softmax qui convertit la sortie de chaque classe dans le score de probabilité de chaque classe.

Les données sont introduites dans le modèle et la sortie de chaque couche est obtenue à partir de l’étape ci-dessus est appelée **feedforward**, nous calculons ensuite l’erreur en utilisant une fonction d’erreur, certaines fonctions d’erreur courantes sont **cross-entropy**, erreur de perte carrée, etc. La fonction d’erreur mesure la performance du réseau. Après cela, nous nous propageons dans le modèle en calculant les dérivés. Cette étape est appelée **Backpropagation** qui est essentiellement utilisée pour minimiser la perte.



**Figure 4 :** Diagramme illustrant comment un réseau de neurones traite les données.

## 1.2.2 Convolution Neural Network

## Le réseau de neurones convolutionnel (CNN) est la version étendue des réseaux de neurones artificiels (ANN) qui est principalement utilisée pour extraire la caractéristique de l’ensemble de données matricielles de type grille. Par exemple, des jeux de données visuels comme des images ou des vidéos où les modèles de données jouent un rôle important.

## 1.2.2.1 Architecture CNN

## CNN architecture -Geeksforgeeks

## Figure 5 : Architecture CNN simple

## Le calque convolutionnel applique des filtres à l’image d’entrée pour extraire des fonctionnalités, le calque de regroupement échantillonne l’image pour réduire le calcul, et le calque entièrement connecté effectue la prédiction finale. Le réseau apprend les filtres optimaux par rétropropagation et descente de gradient.

## convolutional neural network vs. recurrent neural network

## Figure 6: Tableau illustrant les différences entre un réseau de neurones convolutifs et un réseau de neurones récurrents.

## Fonctionnement des couches convolutives

## Les réseaux de neurones à convolution ou convNets sont des réseaux de neurones qui partagent leurs paramètres. Imaginez que vous avez une image. Il peut être représenté comme un cuboïde ayant sa longueur, sa largeur (dimension de l’image) et sa hauteur (c’est-à-dire le canal car les images ont généralement des canaux rouges, verts et bleus).

## Image Channel - Geeksforgeeks

## Figure 7: dimension d’image (width & height)

## Maintenant, imaginez prendre un petit patch de cette image et exécuter un petit réseau de neurones, appelé un filtre ou un noyau, avec, disons, des sorties K et les représenter verticalement. Maintenant, faites glisser ce réseau de neurones sur l’ensemble de l’image, par conséquent, nous obtiendrons une autre image avec des largeurs, des hauteurs et des profondeurs différentes. Au lieu des canaux R, G et B, nous avons maintenant plus de canaux, mais moins de largeur et de hauteur. Cette opération s’appelle Convolution. Si la taille du patch est la même que celle de l’image, ce sera un réseau de neurones régulier. En raison de ce petit patch, nous avons moins de poids.

## Convolution operation in CNN - Geeksforgeeks

## Maintenant, parlons un peu de mathématiques qui sont impliqué dans tout le processus de convolution.

## Les couches de convolution se composent d’un ensemble de filtres (ou de noyaux) pouvant être mémorisés ayant de petites largeurs et hauteurs et la même profondeur que celle du volume d’entrée (3 si la couche d’entrée est entrée image).

## Par exemple, si nous devons exécuter la convolution sur une image avec des dimensions 34x34x3. La taille possible des filtres peut être axax3, où « a » peut être n’importe quoi comme 3, 5 ou 7 mais plus petit par rapport à la dimension de l’image.

## Au cours de la passe avant, nous faisons glisser chaque filtre sur l’ensemble du volume d’entrée étape par étape où chaque étape est appelée foulée (qui peut avoir une valeur de 2, 3 ou même 4 pour les images de grande dimension) et calculons le produit de points entre les poids du noyau et le patch du volume d’entrée.

## En glissant nos filtres, nous obtiendrons une sortie 2-D pour chaque filtre et nous les empilerons en conséquence, nous obtiendrons un volume de sortie ayant une profondeur égale au nombre de filtres. Le réseau apprendra tous les filtres.

* + - 1. **Couches utilisées pour construire ConvNets**

Une architecture complète de réseaux de neurones à convolution est également connue sous le nom de covnets. Un covnets est une séquence de couches, et chaque couche transforme un volume à un autre grâce à une fonction différenciable.

Types de couches : jeux de données

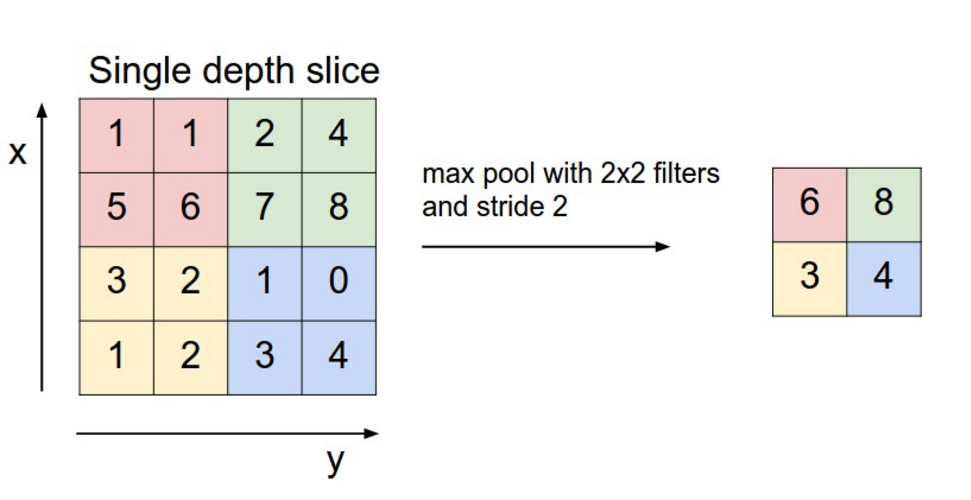
Prenons un exemple en exécutant un covnets sur une image de dimension 32 x 32 x 3.

**Couches d’entrée :** C’est la couche dans laquelle nous donnons l’entrée à notre modèle. Dans CNN, généralement, l’entrée sera une image ou une séquence d’images. Ce calque contient l’entrée brute de l’image avec la largeur 32, la hauteur 32 et la profondeur 3.

**Couches convolutionnelles** : Il s’agit de la couche utilisée pour extraire la fonctionnalité du jeu de données d’entrée. Il applique aux images d’entrée un ensemble de filtres lisibles appelés noyaux. Les filtres / noyaux sont des matrices plus petites, généralement de forme 2 2, 3 3 ou 5 5. Il glisse sur les données d’image d’entrée et calcule le produit de points entre le poids du noyau et le patch d’image d’entrée correspondant. La sortie de cette couche est référée aux cartes d’entités publicitaires. Supposons que nous utilisions un total de 12 filtres pour cette couche, nous obtiendrons un volume de sortie de dimension 32 x 32 x 12.

**Couche d’activation :** En ajoutant une fonction d’activation à la sortie de la couche précédente, les couches d’activation ajoutent une non-linéarité au réseau. il appliquera une fonction d’activation par élément à la sortie de la couche de convolution. Certaines fonctions d’activation courantes sont RELU : max (0, x), **Tanh, Leaky** **RELU,** etc. Le volume reste inchangé, donc le volume de sortie aura des dimensions 32 x 32 x 12.

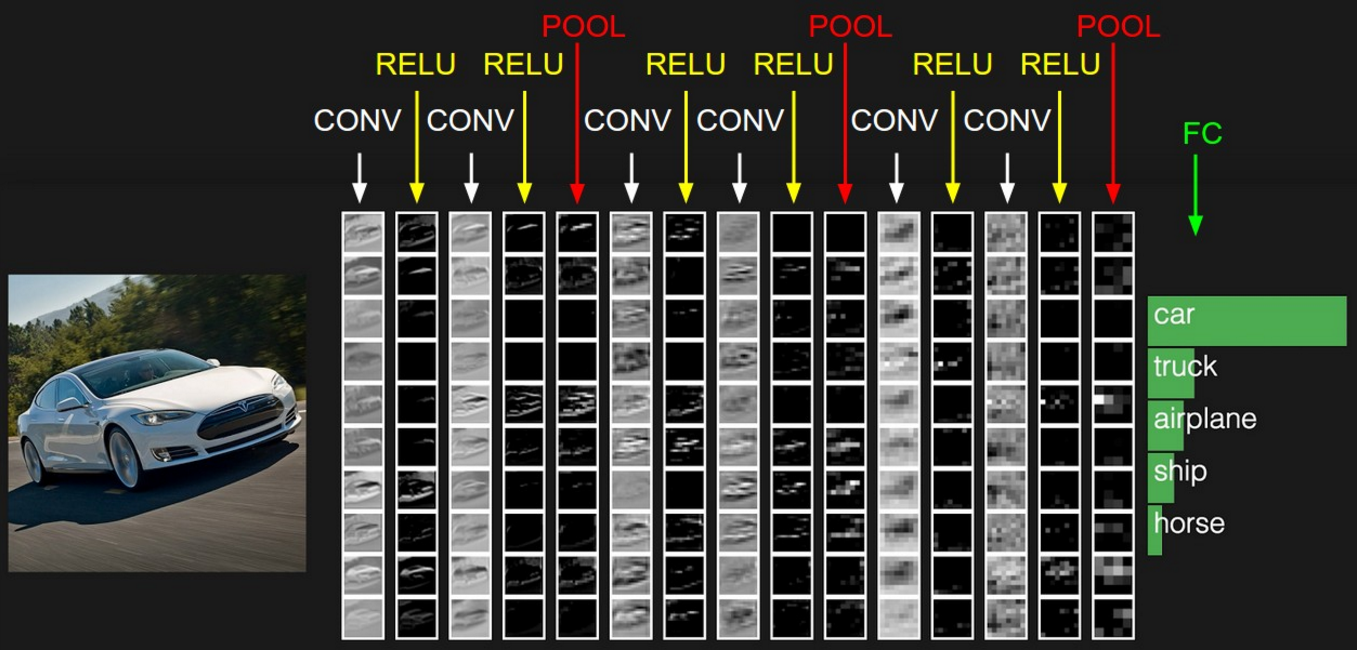
**Couche de mise en commun (Pooling layer) :** Cette couche est périodiquement insérée dans les **covnets** et sa fonction principale est de réduire la taille du volume, ce qui rend le calcul rapide réduit la mémoire et empêche également le sur-ajustement. Deux types courants de couches de mise en commun sont la mise en commun maximale et la mise en commun moyenne. Si nous utilisons une piscine max avec 2 x 2 filtres et stride 2, le volume résultant sera de dimension 16x16x12.



**Source image:** *cs231n.stanford.edu*

**Flattening:** The resulting feature maps are flattened into a one-dimensional vector after the convolution and pooling layers so they can be passed into a completely linked layer for categorization or regression.

**Fully Connected Layers:** It takes the input from the previous layer and computes the final classification or regression task.



**Couche de sortie :** La sortie des couches entièrement connectées est ensuite introduite dans une fonction logistique pour les tâches de classification comme sigmoïde ou softmax qui convertit la sortie de chaque classe dans le score de probabilité de chaque classe.

* + - 1. **Avantages des réseaux de neurones convolutifs (CNN) :**

1. Bon pour détecter les motifs et les fonctionnalités dans les images, les vidéos et les signaux audio.

2. Robuste à l’invariance de translation, de rotation et d’échelle.

3. Formation de bout en bout, pas besoin d’extraction manuelle des fonctionnalités.

4. Peut gérer de grandes quantités de données et atteindre une grande précision.

**1.2.2.5 Inconvénients des réseaux de neurones convolutifs (CNN) :**

1. Computationally cher à former et nécessite beaucoup de mémoire.
2. Peut être sujette au sur-ajustement si pas assez de données ou une régularisation appropriée est utilisée.
3. Nécessite de grandes quantités de données étiquetées.
4. L’interprétabilité est limitée, il est difficile de comprendre ce que le réseau a appris.

**Conclusion**

En conclusionLe Deep Learning et les CNN continueront à faire partie des outils d'IA les plus puissants pour les développeurs et les entreprises à l'avenir. Les entreprises seront toujours amenées à trouver des solutions nouvelles et innovantes pour les aider à relever leurs uniques défis. Nombreux sont celles qui se tourneront vers des technologies reposant sur le deep learning et les CNN, comme la vision par ordinateur, l'IA, la réalité augmentée et la réalité virtuelle, pour trouver des solutions. À mesure que l'IA progresse, Intel s'engage à faire en sorte que les développeurs, les scientifiques des données, les chercheurs et les ingénieurs des données puissent préparer, construire, déployer et mettre à l'échelle leurs solutions d'IA de la manière la plus transparente possible.

* 1. **Algorithme recuit simulé (SA)**

**Introduction**

Trouver une solution optimale à certains problèmes d’optimisation peut être une tâche incroyablement difficile, souvent pratiquement impossible. En effet, lorsqu’un problème devient suffisamment important, nous devons rechercher un grand nombre de solutions possibles pour trouver la solution optimale. Même avec la puissance de calcul moderne, il y a souvent encore trop de solutions possibles à envisager. Dans ce cas, parce que nous ne pouvons pas raisonnablement nous attendre à trouver le meilleur dans un délai raisonnable, nous devons nous contenter de quelque chose qui est assez proche.

**1.3.1. C’est quoi l’algorithme recuit simulé ?**

SA est une technique d’optimisation méta heuristique introduite par **Kirkpatrick** et **al**. en 1983 pour résoudre le problème des vendeurs itinérants (TSP).

L’algorithme SA est basé sur le procédé de recuit utilisé en métallurgie, où un métal est chauffé à haute température rapidement puis refroidi progressivement. À des températures élevées, les atomes se déplacent rapidement, et lorsque la température est réduite, leur énergie cinétique diminue également. À la fin du processus de recuit, les atomes tombent dans un état plus ordonné et le matériau est plus ductile et plus facile à travailler.

De même, en SA, un processus de recherche commence par un état de haute énergie (une solution initiale) et abaisse progressivement la température (un paramètre de contrôle) jusqu’à atteindre un état d’énergie minimale (la solution optimale).

La SA a été appliquée avec succès à un large éventail de problèmes d’optimisation, tels que le TSP, le pliage de protéines, le partitionnement de graphes et la planification d’atelier. Le principal avantage de la SA est sa capacité à s’échapper des minima locaux et à converger vers un minimum global. SA est également relativement facile à mettre en œuvre et ne nécessite pas de connaissance a priori de l’espace de recherche.

Le processus de recuit simulé commence par une solution initiale, puis améliore itérativement la solution actuelle en la perturbant aléatoirement et en acceptant la perturbation avec une certaine probabilité. La probabilité d’accepter une solution pire est initialement élevée et diminue progressivement à mesure que le nombre d’itérations augmente.

L’algorithme SA est assez simple, et il peut être implémenté directement comme décrit ci-dessous.

**1.3.2. Définir le problème**

Premièrement, il faut définir le problème pour l’optimiser, c’est-à-dire définir la fonction énergétique, c’est-à-dire la fonction pour minimiser ou maximiser. Par exemple, si nous voulons minimiser une fonction à valeur réelle de deux variables, par exemple, f(x,y) = x 2 + y 2, l’énergie correspond à la fonction f(x,y) elle-même. Dans le cas du PTS, l’énergie liée à une séquence de villes est représentée par la longueur totale du trajet.

Une fois la fonction énergétique définie, nous devons définir la valeur de température initiale et la solution candidate initiale. Ce dernier peut être généré aléatoirement ou en utilisant une autre méthode heuristique. Ensuite, nous calculons l’énergie de la solution candidate initiale.

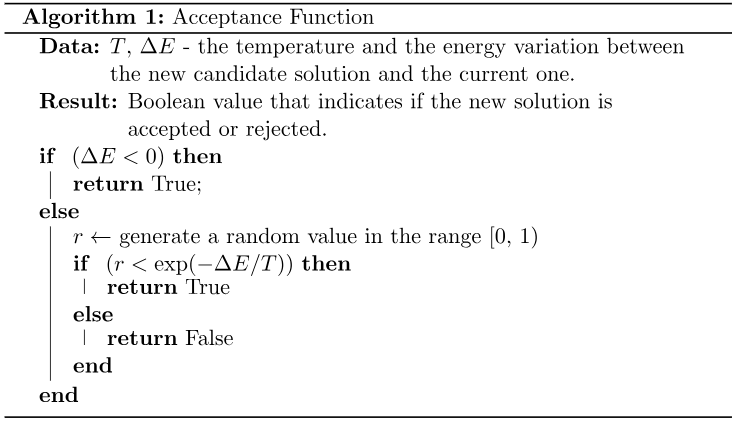
**1.3.3. Définir la fonction de perturbation**

Une fonction de perturbation est définie pour générer de nouvelles solutions candidates. Cette fonction devrait générer des solutions proches de la solution actuelle mais pas trop similaires. Par exemple, si nous voulons minimiser une fonction f(x,y), nous pouvons perturber aléatoirement la solution actuelle en ajoutant une valeur aléatoire entre -0,1 et 0,1 à x et y. Dans le cas du TSP, une nouvelle solution candidate peut être générée en échangeant deux villes dans l’ordre de déplacement de la solution actuelle.

**1.3.4. Critère d'acceptation**

Le critère d’acceptation détermine si une nouvelle solution est acceptée ou rejetée. L’acceptation dépend de la différence d’énergie entre la nouvelle solution et la solution actuelle, ainsi que de la température actuelle. Le critère d’acceptation classique de SA vient de la mécanique statistique, et il est basé sur la distribution de probabilité de Boltzmann. Un système en équilibre thermique à température T peut être trouvé dans un état à énergie E avec une probabilité proportionnelle à exp (-E / k T) (1) commencer {équation\*} operatorname {Prob}(E) sim exp (-E / k T) fin{équation\*} où k est la constante de Boltzmann. Par conséquent, à basse température, il y a une petite chance que le système soit dans un état de haute énergie. Cela joue un rôle crucial dans l’AS car une augmentation de l’énergie permet d’échapper aux minima locaux et de trouver le minimum global.

Basé sur la distribution de Boltzmann, l’algorithme suivant définit le critère d’acceptation d’une variation d’énergie Delta E à température T.



**Figure 8 :** L’algorithme de la fonction d’acceptation

Une solution candidate avec moins d’énergie est toujours acceptée. À l’inverse, une solution candidate avec une énergie plus élevée est acceptée aléatoirement avec une probabilité exp (- Delta E / T) (pour notre part, nous pouvons définir k = 1). Ce dernier cas peut être mis en œuvre en comparant la probabilité avec une valeur aléatoire générée dans la plage [0, 1).

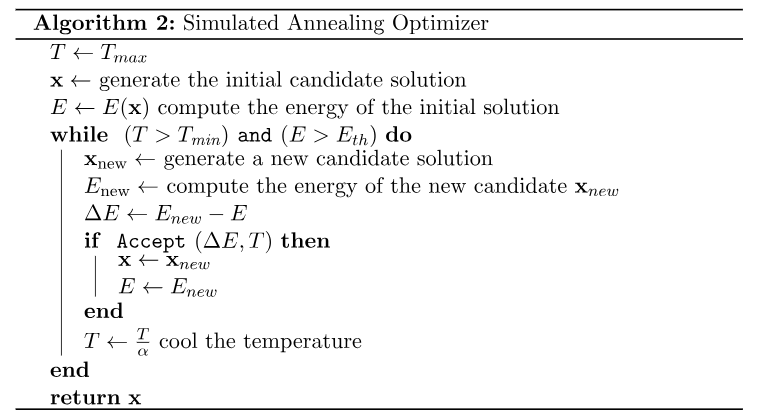
**1.3.5. Calendrier des températures**

Le programme de température détermine la façon dont la température du système change au fil du temps. Au début, la température est élevée afin que l’algorithme puisse explorer un large éventail de solutions, même si elles sont pires que la solution actuelle. À mesure que les itérations augmentent, la température diminue progressivement, de sorte que l’algorithme devient plus sélectif et accepte de meilleures solutions avec une probabilité plus élevée. Un ordonnancement simple peut être obtenu en divisant la température actuelle par un facteur alpha, qui est inférieur à 1.

**1.3.6. Exécuter l’algorithme SA**

Enfin, exécutez l’algorithme en appliquant itérativement la fonction de perturbation et le critère d’acceptation à la solution actuelle. L’algorithme se termine lorsque la température a refroidi à un certain niveau T\_{min} ou lorsque l’énergie de la solution actuelle est inférieure à un seuil fixe E\_{th}.

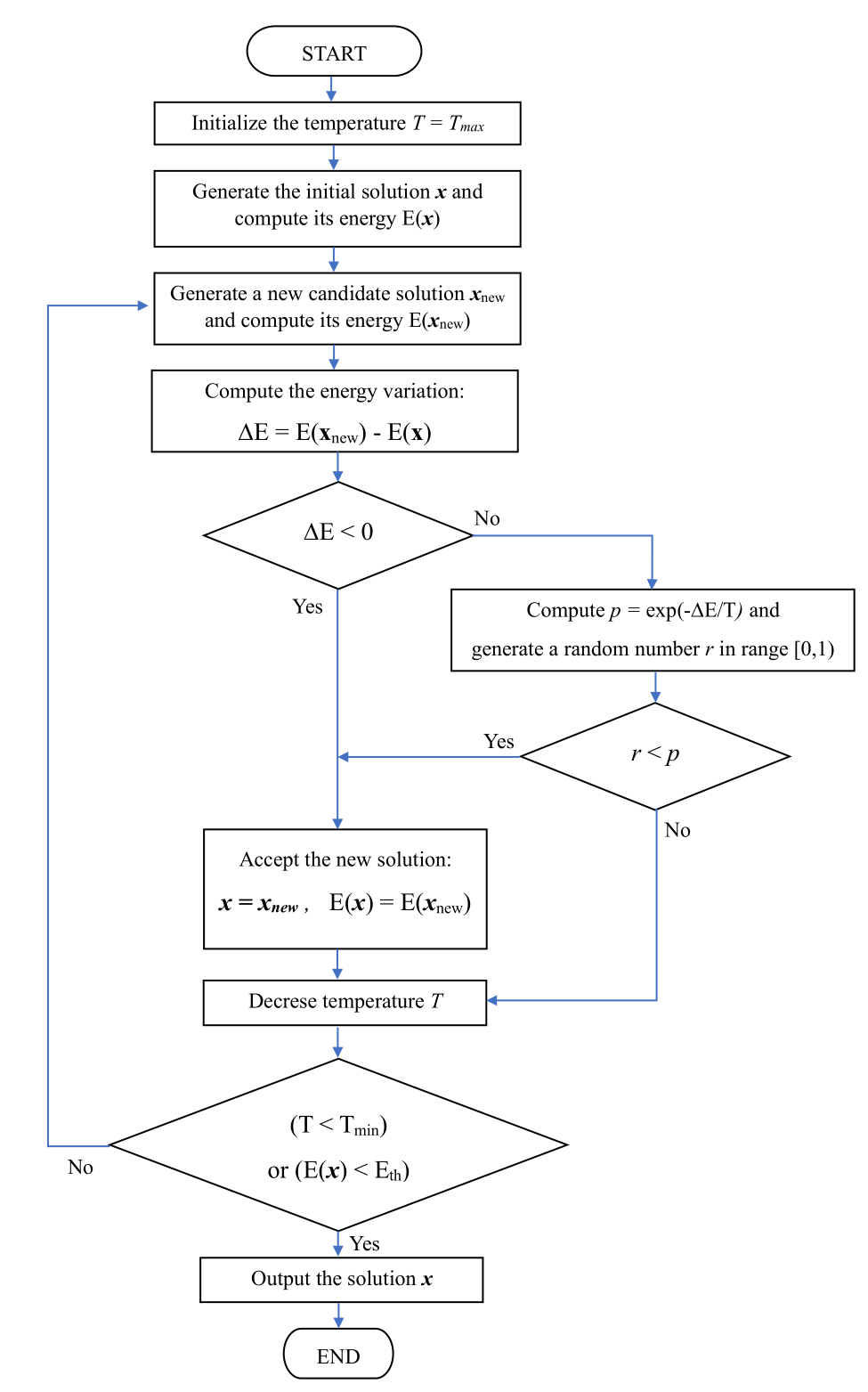
Voici le pseudocode de SA :



**Figure 9 :** l’optimiseur de l’algorithme SA

**1.3.7. Le diagramme de flux de SA**

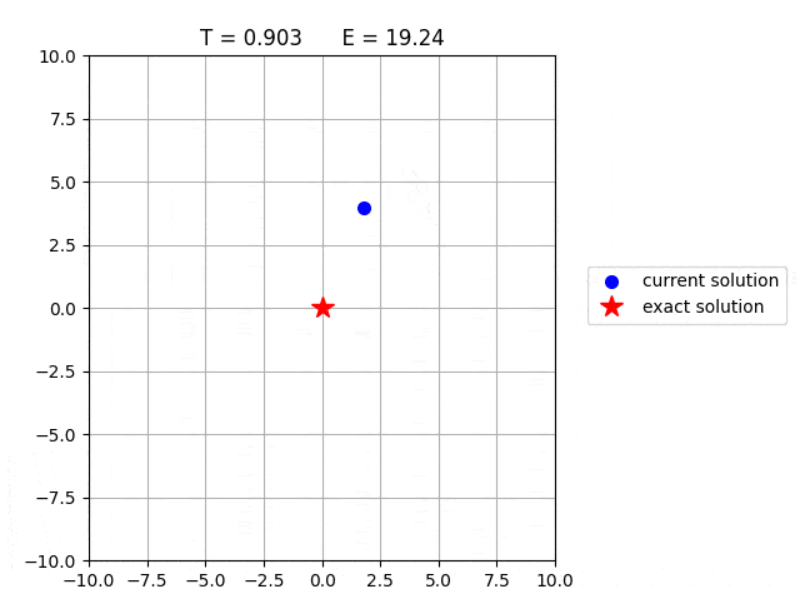
Ici, nous fournissons un organigramme détaillé représentant toutes les étapes de l’AS :



**Figure 10 :** le diagramme de flux de l’algorithme recuit simulé

**1.3.8. Exemple d’application**

Pour mieux comprendre l’algorithme, nous utilisons SA pour illustrer la minimisation de la fonction f(x,y) = x 2 + y 2. Nous avons utilisé comme espace de recherche une grille de taille 101 fois 101 placée dans la zone carrée définie par (x,y) en [-10, 10] fois [-10, 10]. Nous fixons la vitesse de refroidissement alpha=0,84 et la solution initiale (x,y)=(4,4). À chaque étape, une nouvelle solution est générée en déplaçant aléatoirement la solution actuelle par pm 0,2 dans x et y direction.



**Figure 11 :** montrant la solution candidate, son énergie et la température à chaque étape

Nous pouvons observer que des solutions pires sont fréquemment acceptées lorsque la température est élevée. Inversement, lorsque la température est basse (par exemple, T<1), l’algorithme est plus sélectif et de meilleures solutions sont acceptées avec une probabilité plus élevée.

**Conclusion**

Dans cette partie, nous avons donné un aperçu de l’algorithme SA. Nous avons illustré la procédure d’optimisation et fourni un exemple pratique de son application.

* 1. **Système multi agent**

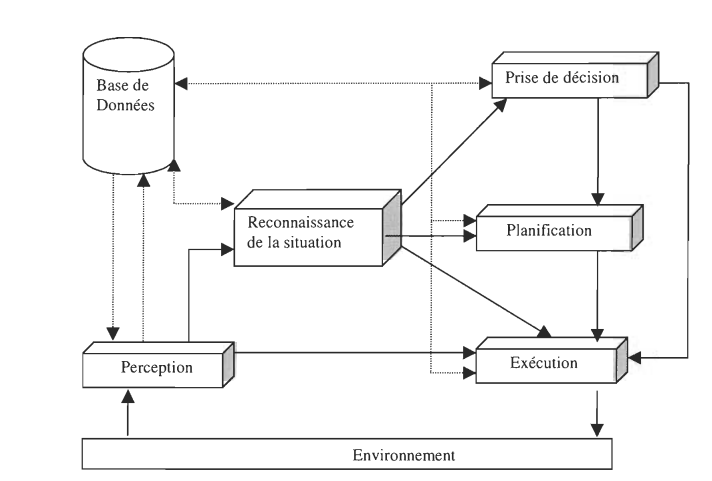
**Introduction**

Le thème des systèmes multi agents (SMAs), s’il n’est pas récent, est actuellement un champ de recherche très actif. Cette discipline est à la connexion de plusieurs domaines en particulier de l’intelligence artificielle, des systèmes informatique distribués et du génie logiciel. C’est une discipline qui s’intéresse aux comportements collectifs produits par les interactions de plusieurs entités autonomes et flexibles appelées agents, que ces interactions tournent autour de la coopération, de la concurrence ou de la coexistence entre ces agents. Ce chapitre introduit, tout d’abord, les notions d’agents et de systèmes multi agents (SMAs), et détaille par la suite les différentes questions que soulèvent la problématique des SMAs, en particulier : les interactions et la coopération, la coordination, la planification et la communication. À la fin, un aperçu des différentes applications des SMAs est donné.

**1.4.1. Agent**

Un agent est un système informatique encapsulé situé dans un environnement dans lequel il est capable d'effectuer une action flexible et autonome, compatible aux objectifs de la conception. Les agents sont: des entités clairement identifiables de résolution de problèmes avec des bornes et des interfaces bien définies; situés dans un environnement particulier; ils reçoivent des entrées liées aux états de cet environnement par des capteurs et agissent sur cet environnement par des émetteurs; destinés à atteindre un objectif spécifique; autonomes et responsables de leur comportement; capables d'adopter un comportement flexible pour résoudre des problèmes selon les objectifs de la conception; ils sont réactifs (capables de s'adapter aux changements d'état de leur environnement) et proactifs (capables d'adopter un nouvel objectif); capables dans un univers multi-agents, de communiquer, coopérer, se coordonner, négocier les uns avec les autres.

La figure 1.1 donne, de façon générale, l'architecture interne d'un agent.

****

**Figure 12 :** Architecture d’agent

Lorsqu'un agent perçoit une situation dans l'environnement, il essaie de la reconnaître. Si la situation lui est familière, il peut enclencher un processus de planification afin de résoudre le problème. il peut aussi reconnaître la situation en terme d'action et donc, passe à l'exécution de la tâche (Reconnaissance- Exécution). Lorsque l'agent perçoit des situations qu'il connaît très bien, il peut faire intervenir son comportement réactif en passant directement à l'action (Perception-Exécution). S'il ne peut pas résoudre un problème (situation non-familière), il engage un processus de coopération pour demander de l'aide aux autres agents (Reconnaissance Prise de décisions).

**1.4.2. Systèmes Multi-agent**

Les systèmes multi-agents mettent en œuvre des agents homogènes et hétérogènes ayant des buts communs ou distincts. Ils sont dynamiques. Un système multi-agent est un système distribué composé d'un ensemble d'agents qui interagissent le plus souvent, selon des modes de coopération, de concurrence ou de coexistence.

Un SMA est généralement caractérisé par :

1. chaque agent a des informations ou des capacités de résolution de problèmes limitées, ainsi chaque agent à un point de vue partiel.

2. il n'y aucun contrôle global du système multi-agents.

3. les données sont décentralisées.

4. le calcul est asynchrone.

**1.4.3. Des Objets aux Agents**

D’un point de vue informatique, l’approche multi-agent peut être considérée comme une évolution du paradigme orienté-objet. Du point de vue conceptuel, un objet est simplement une structure de données à laquelle sont associées des fonctions. Les agents sont des entités autonomes, ce qui signifie que leur comportement ne dépend pas d’une pression extérieure, contrairement aux objets. -Agent: entité autonome interagissant avec son environnement -Objet: entité passive possédant un état et sur lequel on peut effectuer des opérations. (Chaib-draa, 2010) -Un agent est à un degré d’abstraction plus élevé qu’un objet. Un agent peut être constitué de plusieurs objets. -C’est un paradigme de programmation mettant en évidence l’autonomie et les interactions. (Programmation orientée-agent)

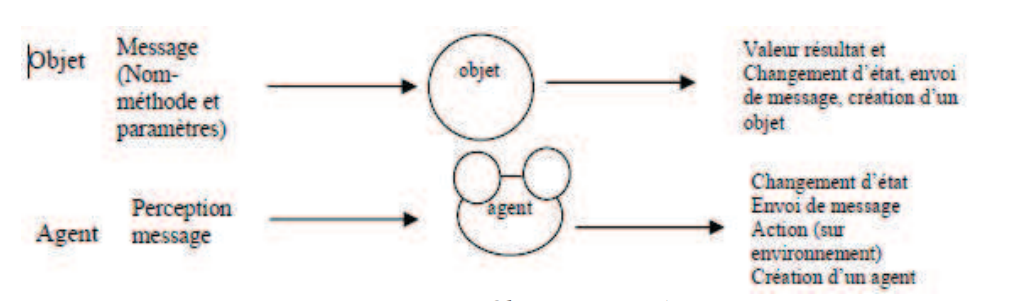
**Similarités** :

-Possèdent un «état interne»,

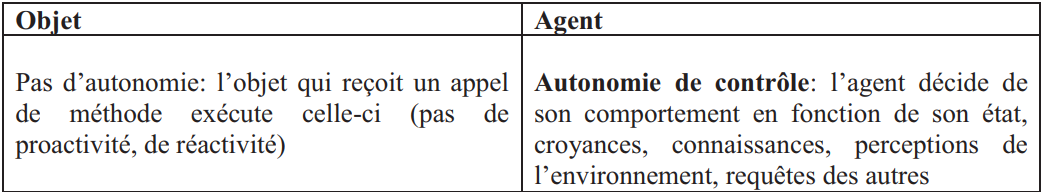
-Des unités de comportement modulaires (méthodes/compétences).

-Communiquent par envoi de messages,.

-Peuvent agir pour modifier leur état.

****

**Figure 13 :** Objet «versus» Agent

****

**Figure 14 :** Différence entre un agent et objet

**1.4.4. Utilité des systèmes multi agents**

Les systèmes multi-agents sont extrêmement adaptés à des environnements nécessitant l'interaction de multiples entités, notamment dans des contextes distribués tels que la coordination entre des frégates, la gestion du trafic aérien ou les bases de données coopératives. Ces domaines, intrinsèquement distribués, trouvent une solution aisée et efficace à leur modélisation à travers les systèmes multi-agents.

Une autre situation propice à l'utilisation des systèmes multi-agents se présente lorsque diverses entités et leurs données appartiennent à des organisations distinctes désirant préserver la confidentialité de leurs informations pour des raisons concurrentielles. Par exemple, dans les missions maritimes impliquant plusieurs pays, plusieurs bateaux de nations différentes doivent collaborer. Cependant, chaque nation souhaite garder confidentielles certaines informations, considérées comme des secrets militaires. De ce fait, la conception d'un système coordonnant ces bateaux exigerait une représentation précise des objectifs et des intérêts de chaque pays à travers la création d'agents propres à chacun. Ces agents pourraient ensuite communiquer entre eux pour coordonner la mission, tout en échangeant uniquement les informations nécessaires à cette coordination, préservant ainsi la confidentialité des données sensibles.

Même dans des domaines où l'utilisation de systèmes multi-agents n'est pas une nécessité absolue, leur utilisation comporte des avantages indéniables. Ces systèmes sont particulièrement adaptés à la résolution de problèmes offrant diverses méthodes, perspectives ou résolutions. Ils trouvent une application efficace dans la modélisation du raisonnement humain au sein de simulations complexes, comme celles des combats aériens. En outre, ils présentent des avantages inhérents à la résolution distribuée et concurrente de problèmes, notamment :

La modularité, simplifiant la programmation et permettant l'extensibilité du système en facilitant l'ajout de nouveaux agents par rapport à l'ajout de nouvelles capacités à un système monolithique.

La vitesse, grâce au parallélisme offert par la possibilité de travailler simultanément sur un problème par plusieurs agents.

La fiabilité, résultant du partage de contrôle et de responsabilités entre les différents agents, permettant au système de tolérer la défaillance d'une ou plusieurs entités sans compromettre l'ensemble du système.

**1.4.5. Propriétés d’un agent intelligent**

Un agent peut être caractérisé par plusieurs propriétés, nous pouvons citer parmi elles :

1. **Autonomies :**

Un agent autonome possède la capacité d'opérer de manière indépendante, sans une intervention directe d'humains ou d'autres agents, tout en contrôlant ses actions et son état interne. Il est caractérisé par la poursuite de ses propres objectifs et la capacité à prendre des décisions relatives à ces objectifs, y compris la résolution de conflits internes.

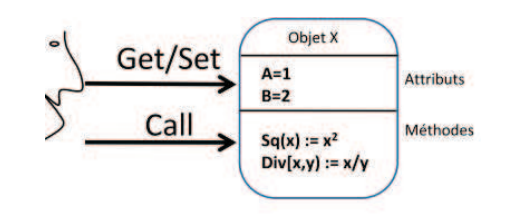
En tant qu'entité autonome, un tel agent est capable de :

1. Agir conformément à ses intentions, agissant de manière délibérée vers des objectifs spécifiques.
2. Adopter des objectifs qu'il estime réalisables tout en abandonnant ceux qu'il perçoit comme impossibles à atteindre.
3. Planifier ses propres actions tout en tenant compte des actions des autres agents ou entités.
4. Raisonner sur les différentes possibilités pour atteindre ses objectifs et résoudre les éventuels conflits internes concernant les objectifs sélectionnés.
5. **Réactivité :**

Les agents sont capables de percevoir leur environnement et d'ajuster leurs actions en réponse aux changements survenant dans cet environnement.

La réactivité des agents implique également leur capacité à adapter leur comportement en fonction des modifications des conditions environnementales.

En revanche, un objet est une entité passive, voire réactive, dans le sens où si aucune demande n'est faite pour obtenir la valeur d'un attribut ou pour activer une méthode spécifique de cet objet, celui-ci demeure inerte et ne réalise aucune action. Autrement dit, il ne réagit pas ou n'agit pas de manière autonome sans qu'une demande extérieure ne soit formulée.

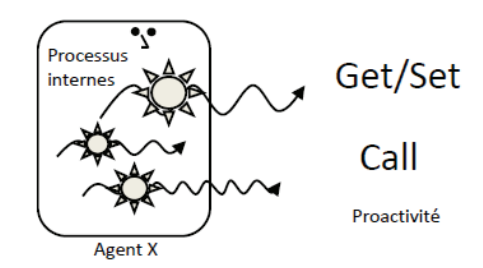


**Figure 14 :** Réactivité

1. **Proactivité**

Les agents ne se contentent pas de réagir à leur environnement ; ils démontrent la capacité de manifester un comportement orienté par des objectifs et d'initier des actions. La proactivité se définit par la faculté d'un agent à anticiper des situations et à modifier son comportement pour les prévenir.

En plus des attributs et des méthodes, un agent possède des processus internes qui demeurent actifs même en l'absence de sollicitations externes. Par conséquent, un agent est capable d'agir de manière autonome et proactive, sans qu'une demande externe ne soit nécessaire pour déclencher son action.



**Figure 15** : Proactivité

1. **Adaptabilité**

Face à un environnement constamment changeant, un agent doit constamment ajuster son plan d'action pour atteindre ses objectifs. À cette fin :

1. L'agent doit procéder à une évaluation continue et à une perception continue de la situation (le contexte) dans lequel il agit.
2. Il doit être capable de construire des représentations en cours d'exécution (c'est-à-dire apprendre en cours de fonctionnement).
3. Il doit développer des plans dynamiques qui initient ou arrêtent des processus internes en réponse aux changements de l'environnement.

L'adaptabilité se définit comme la capacité d'un agent à s'ajuster à l'environnement dans lequel il opère. Un agent adaptatif est donc capable de moduler ses compétences (communicationnelles et comportementales) en fonction de l'environnement dans lequel il évolue et des autres agents avec lesquels il interagit.

Dans un environnement dynamique, l'adaptabilité représente une propriété cruciale pour un agent, lui permettant de s'ajuster et de réagir de manière appropriée aux changements continus de son environnement.

1. **Sociabilité**

La sociabilité d'un agent désigne sa capacité à s'intégrer efficacement dans un environnement où plusieurs agents interagissent en échangeant des messages dans le but de réaliser un objectif commun.

Cette caractéristique sera approfondie ultérieurement lorsque nous aborderons les propriétés d'un Système Multi-Agent (SMA), car elle englobe d'autres propriétés telles que la communication, la coopération et la délégation.

1. **Apprentissage**

L'apprentissage est une propriété singulière, car tous les agents ne sont pas nécessairement dotés de cette capacité. Les agents peuvent avoir besoin d'apprendre lorsqu'ils interagissent ou réagissent à leur environnement externe.

Cette caractéristique confère aux systèmes la faculté d'acquérir une compréhension des comportements au fil du temps, sans nécessiter une programmation manuelle de ces comportements.

La capacité d'apprentissage d'un agent se manifeste par sa capacité à utiliser de nouvelles connaissances pour ajuster son comportement.

Bien que l'apprentissage soit un élément essentiel de l'intelligence, il est rare de trouver des agents dotés de cette capacité.

1. **Sécurité**

La sécurité revêt une importance capitale, particulièrement dans le cadre de cette étude, car elle assure qu'en interagissant avec un agent, celui-ci n'a pas été altéré par un virus, des informations erronées, ou des connaissances dépourvues de sens

**1.4.6. Propriétés des systèmes multi-agents**

**1. interaction agents**

1. **Coopération**
2. **Coordination**
3. **Organisation**
4. **Planification**
5. **Synchronisation**
6. **Délégation**
7. **Communication**