# Utilisation des résaux de neurones pour le tuning des Algorithmes d'optimisation par colonies de fourmis

## Application aux chaines logistiques

Ahmed Haroun SABRY[1], Abdelkabir Bacha[2], Jamal BENHRA[3]
EAP Team, LISER laboratory, ENSEM
KM7, BP 8118 Route El Jadida
Casablanca, Morocco
harounsabry@gmail.com[1], bachaabdelkabir@gmail.com[2], jbenhra@hotmail.com[3]

***Résumé*— Le présent papier décrit une contribution au tuning des algorithmes d'optimisation par colonies de fourmis pour la résolution du problème du voyageur de commerce. Les algorithmes d'optimisation par colonies de fourmis sont très sensibles à la variation des paramètres. L'objectif de ce papier est l'utilisation des réseaux de neurones en Rétro-propagation pour l'estimation de la configuration idéale de ces paramètres. La nature du problème choisi est un TSP pour application logistique.**

**Mots-clefs — Optimisation par colonies de fourmis ; Paramétrage ; réseau de neurones ; TSP**

## I. INTRODUCTION

Une vaste majorité des problèmes d'ingénierie peuvent être exprimés sous la forme d'un problème d'optimisation où on définit une fonction cout. L'objectif étant de minimiser (parfois maximiser) cette fonction tout en satisfaisant les contraintes du problème.

Un des problèmes les plus étudiés en recherche opérationnelle et en mathématique en général est le problème du voyageur de commerce. Avec de nombreuses applications dans le monde réel principalement dans le domaine de la logistique et de la production. Le présent travail traite la version symétrique du TSP fortement utilisée en logistique et supply chains, dans le transport, l'entreposage ou tout simplement le choix des collaborateurs et des centres [1, 2].

Le problème du voyageur de commerce est un problème d'optimisation NP-difficile, et requiert l'utilisation d'efficientes approches algorithmiques pour sa résolution. La métaheuristique présentée dans ce papier est un exemple d'approche qui a prouvé son efficacité à résoudre le PVC ainsi qu'un grand nombre de problèmes similaires [3].

L'optimisation par colonies de fourmis est une métaheuristique bio inspirée et elle est très sensible à la variation des paramètres initiaux [4]. La qualité de la solution finale trouvée par l'algorithme dépend principalement de la variante algorithmique utilisée, de la vitesse de calcul et des préréglages de la métaheuristique. Ce travail de préréglage est souvent fastidieux et requiert une connaissance approfondie du comportement algorithmique par rapport au problème étudié [5].

L'objectif de ce travail est donc l'utilisation de l'apprentissage artificiel ici par réseaux de neurones pour le préréglage (Tuning) de la métaheuristique. Pour ce faire, le présent papier est structuré comme suit : une première partie dresse un état de l'art, la deuxième partie présente l'approche utilisée, la troisième partie discute les résultats trouvés, et la conclusion met fin à ce travail et ouvre la discussion sur d'autres opportunités de recherche.

## II. ETAT DE L'ART

### A. *Le problème du voyageur de commerce*

Le problème de voyageur de commerce (PVC) est un problème d'optimisation combinatoire, ou la fonction objectif est la minimisation du cout total du trajet (ici distance). Etant donné un ensemble de villes, il s'agit de trouver le chemin le plus court en passant par toutes les villes. Vu sa nature, ce problème se modélise facilement en utilisant les graphes [6] [7,8] .

Un cycle hamiltonien T sur un graphe complet $K\ n$ à $n$ sommets est un ensemble de $(n(n-1))/2$ arêtes, noté $E(T)$. Cet ensemble d'arêtes doit vérifier que le graphe partiel $G' = (V(K\ n), E(T))$ de $K\ n$ qu'il engendre consiste en un cycle unique qui couvre tous les sommets.

La longueur d'un circuit hamiltonien $\mu$ est la somme des longueurs des arcs qui le composent. Le PVC devient donc un problème consistant à trouver un circuit hamiltonien de longueur minimale sur le graphe complet valué G = (E, V)

Dorigo [9] a adopté le formalisme suivant :
– On définit $V$ un ensemble fini de $N$ noeuds représentant des villes
– On définit $E = [\ (i, j)\ |\ i, j \in V\ ]$ un ensemble fini d'arcs reliant les noeuds de $V$

Xème Conférence Internationale : Conception et Production Intégrées, CPI 2015, 2-4 Décembre 2015, Tanger - Maroc.

Xth International Conference on Integrated Design and Production, CPI 2015, December 2-4, 2015, Tangier - Morocco.

Finalement un ensemble de constantes *d ij* représentant la longueur de chaque arc *(i,j) ∈ E* , c'est-à-dire la distance séparant la ville *i* de la ville *j (avec i,j∈ V)*.

### *B. L'optimisation par colonies de fourmis*

L'optimisation par colonies de fourmis est une métaheuristique utilisée pour trouver des solutions approchées à de nombreux problèmes combinatoires. Elle appartient à une famille de métaheuristiques bio inspirées.

L'optimisation par colonies de fourmis de l'anglais Ant colony optimization (ACO) est une approche distribuée, elle utilise des agents appelés fourmis artificielles. Et constitue avec d'autres algorithmes une classe distincte de métaheuristiques [10].

Les fourmis artificielles ressemblent aux fourmis naturelles, chaque fourmi est indépendante et communique avec les autres membres de la colonie à travers une substance chimique appelée phéromone.

Ces fourmis sont très sensibles aux traces de phéromones, elles l'utilisent pour marquer le chemin et pour communiquer avec les autres membres. Chaque nouvelle fourmi lorsqu'elle croise un chemin imprégné de phéromone à tendance à le suivre, ce dernier va l'emmener soit à une source de nourriture soit au nid.

Ce comportement collaboratif ainsi que la nature évaporatoire de la phéromone offre aux individus la capacité de trouver le chemin le plus court reliant le nid à la source de nourriture [11].

La fourmi artificielle est facile à programmer, *Ant system* est le premier algorithme à voir le jour, par Dorigo [12]. Dans ce dernier, une fourmi k positionnée dans la ville i, à tendance à choisir la ville j comme prochaine ville en fonction de la visibilité η de cette dernière et du taux de phéromone τ déposé sur le chemin menant à cette ville.

En utilisant ces deux paramètres η et τ, ainsi que deux autres paramètres α et β, définissant le poids de la visibilité et de la phéromone. Le choix de la prochaine ville à visiter est stochastique, avec une probabilité [11, 12] :

$$p(c_{ij}|s^p) = \frac{\tau_{ij}^{\alpha} \cdot \eta_{ij}^{\beta}}{\sum_{c_{il} \in N(s^p)} \tau_{il}^{\alpha} \cdot \eta_{il}^{\beta}}, \forall c_{ij} \in N(s^p),$$

Dans le *Ant system*, utilisé dans ce papier, la visibilité η dépends de la distance qui sépare *i* de *j*. le taux de phéromone τ dépends de la quantité déposée χ et de l'évaporation de ce dernier ρ. Les algorithmes d'optimisation par colonies de fourmis sont très sensibles à la variation de ces paramètres. Un tuning préliminaire est primordial pour atteindre des résultats satisfaisants pour un problème donné [13].

Dans la littérature on retrouve plusieurs approches de préréglage de l'algorithme [5], une première famille dite offline traite le choix des paramètres avant le démarrage de l'algorithme. Généralement ce choix se fait manuellement, et dépends de la habileté de l'auteur à choisir le bon set de paramètres par rapport au problème traité.

La deuxième famille de préréglages est dite online, la configuration choisie change au fur et à mesure que la recherche algorithmique continue [14].

La recherche de la bonne configuration est souvent laborieuse, et requiert une adaptation spécifique à l'instance étudiée. Des approches d'automatisation de cette tache ont été proposées, parmi elles l'utilisation de l'intelligence artificielle [5, 14]. Le paragraphe suivant détaille le modèle de calcul d'intelligence artificielle utilisé.

### *C. Les résaux de neurones artificiels*

Les réseaux neuronaux artificiels (RNA) sont construits sur un paradigme biologique, celui du neurone formel (bio inspirés, comme pour les algorithmes d'optimisation par colonies de fourmis décrits précédemment). Par analogie aux neurones biologiques, un réseau de neurones artificiels est capable d'apprendre et de reproduire des comportements « intelligents » d'une manière artificielle [15].

Dans la littérature, les RN ont connus trois grandes phases de développement, leur premier succès est venu en 1940 avec les travaux de McCulloch et Pitts [16].

La deuxième phase vient dans les années 60 avec le theorème de convergence de Rosenblatt [17] et le travail de Minsky et Papert démontrant les limitations d'un perceptron [18]. Ceci a conduit à une démotivation de la communauté scientifique qui a conduit à une stagnation dans le domaine qui a duré 20 ans.

Depuis les années 1980, les RN ont reçu un intérêt vif. Les avancées en résultantes sont nombreuses parmi elles on cite l'approche de Hopfield en 1982 [19] et l'apprentissage par l'algorithme de Back-propagation (rétropropagation du gradient) pour les perceptrons multicouches MLP [20].

Le modèle de calcul basé sur les perceptrons multicouches (multilayer perceptrons) est le type des réseaux de neurones le plus utilisé dans les problèmes d'approximation et d'optimisation [21].

Comme leurs nom l'indique les MLP sont organisés en couches, un MLP permet un flux unidirectionnel des données à partir des entrées jusqu'aux sorties du réseau. Le perceptron multicouche a souvent une couche d'entrée, une couche de sortie et une ou plusieurs couches cachées. La figure 1 montre un exemple de ce perceptron avec deux couches cachées, une couche d'entrée à trois entrées et une couche de sortie à sortie unique.

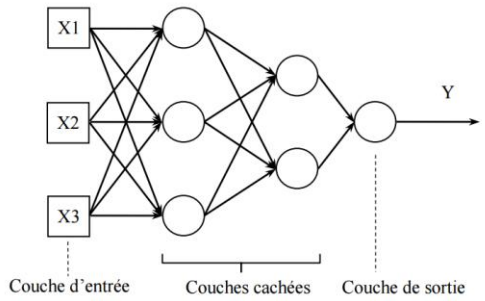


**Fig 1 – Le modèle de perceptrons multicouches (MLP)**

Un neurone biologique a la propriété de produire et de conduire un influx nerveux, le neurone artificiel utilise les fonctions de transfert pour imiter ce comportement biologique. Deux fonctions permettent de faire cela :

- La fonction d'activation détermine le signal total que reçoit le neurone Il est souvent donné par le produit scalaire des entrées pondérées.
- La fonction de sortie est f(I). La combinaison des deux fonctions constitue la fonction de transfert qui permet de calculer la sortie du neurone à partir des données d'entrées [22].

La figure 2 montre un neurone artificiel à plusieurs entrées, et une fonction de transfert f.

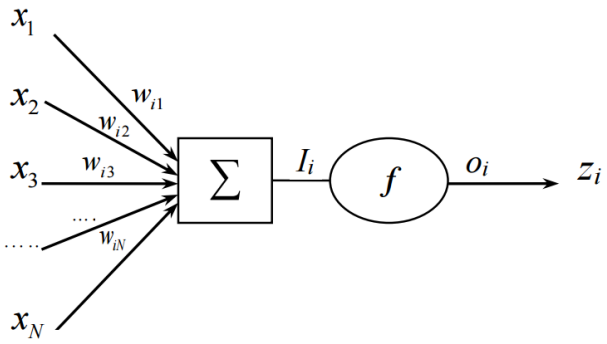


**Fig 2 – Modèle d'un neurone artificiel**

En fonction du problème étudié et l'architecture du réseau de neurone choisi, plusieurs fonctions d'activation peuvent être utilisées, la fonction de transfert la plus utilisé par la littérature est la fonction logistique.

L'algorithme d'apprentissage joue un rôle pour définir les poids des arcs dans un RN. Un bon algorithme d'apprentissage permet une meilleure utilisation du réseau.

La méthode de rétro propagation du gradient (Backpropagation en anglais) est une méthode qui permet de calculer le gradient de l'erreur pour chaque nœud (ici neurone) d'un graphe [20], la retro propagation calcul cette erreur en partant de la dernière couche ( dite de sortie) en montant vers la première couche ( dite d'entrée)

De façon plus précise, les poids synaptiques (d'arcs) qui contribuent à générer une erreur importante se verront changés de manière plus significative que les poids qui ont engendré une erreur peu importante. Ceci rend l'apprentissage plus efficace en termes de temps et ressources computationnelles.

## III. APPROCHE ADOPTEE

Cette section présente notre contribution pour aider à paramétrer les ACOs en utilisant les RN en back propagation. Pour ce faire nous avons généré une base d'apprentissage pour les RN en se basant sur les simulations de l'ACO sur un problème donné. Le problème choisi est le fameux BERLIN52 de la TSPLib [23]. Ce problème contient 52 villes et la solution optimale donnée par la TSPLib est 7542. Ce chiffre va nous aider à comparer les performances de l'algorithme en utilisant les configurations proposées par les RN.

La base d'apprentissage contient 20 simulations avec des données aléatoires des paramètres α β ρ et χ. Chaque simulation est faite sur une machine personnelle équipée d'un processeur Intel(R) Core(TM) i3-3217U CPU @1.80GHz et 3Gb de RAM. Le nombre maximal d'itérations allouées est 1000, ceci permet de profiter au maximum des ressources du processeur avec les configurations proposées. Le résultat de l'ACO en utilisant les 4 paramètres proposés est ensuite inclus dans la base d'apprentissage.

Le réseau de neurones utilisé est illustré dans la figure 3, nous optons pour le temps de convergence et la fonction objectif comme paramètres d'entrée. Et les paramètres α β ρ et χ comme éléments de sortie.

La fonction de transfert choisie est une sigmoïde pour la couche cachée, et linéaire pour la couche de sortie. Le nombre de neurones dans la couche cachée est 10. Le nombre de neurones de sortie est 4.

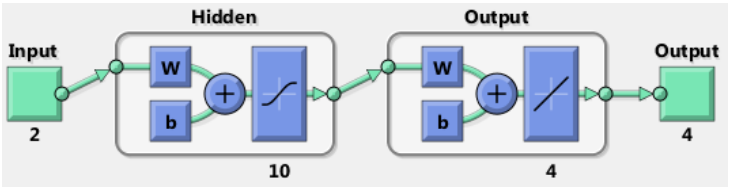


**Fig 3 – Modèle du réseau de neurones utilisé**

Parmi les données d'apprentissage 20% seront utilisées pour la validation et 5% pour le test de fitness de notre réseau. L'algorithme d'apprentissage utilisé est Levenberg-Marquardt en rétro propagation. Ceci permet d'obtenir une solution numérique au problème de minimisation du gradient d'erreur.

Après validation de l'apprentissage de notre réseau, nous lançons l'algorithme de la figure 4 combinant des itérations ACO/RN pour enfin converger vers une proposition de paramètres que nous jugeons satisfaisante.

L'approche est la suivante : nous démarrons l'algorithme avec la création d'un RN multicouches en respectant les contraintes E/S. Nous crayons ensuite une base d'apprentissage pour le RN à partir de paramètres choisis aléatoirement.

L'étape cruciale est le lancement de l'apprentissage du RN et la validation de cet apprentissage en utilisant l'algorithme en back propagation de Levenberg-Marquardt [24]. Vient ensuite la boucle principale de notre approche dans laquelle une itération RN/ACO est bouclée jusqu'à l'atteinte de l'objectif souhaité.

La boucle principale permet d'évaluer les paramètres proposés par le RN en les introduisant dans une nouvelle instance d'ACO. Ces paramètres ainsi que leur résultats sont ajoutés dans la base d'apprentissage et l'algorithme de Levenberg-Marquardt est lancé une nouvelle fois pour adapter les RN a cette nouvelle information.

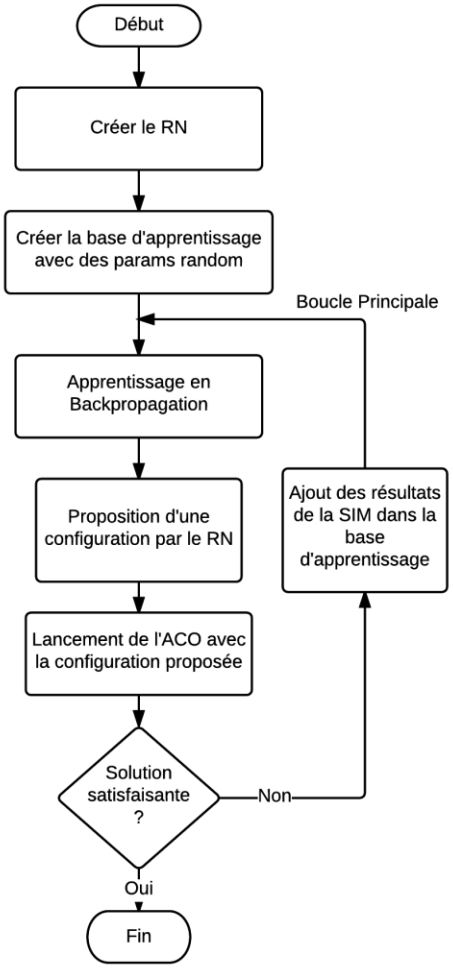


**Fig 4 – L'algorithme de Tuning implémenté (ACO/RN)**

Les RNs vont donc profiter d'un retour d'expérience, ils seront guidés dans leur recherche de la bonne configuration en incluant d'une façon dynamique de nouvelles données d'apprentissage. Il est prévu ainsi que l'hybride RN/ACO va converger rapidement pour trouver la configuration idéale, et que pendant les itérations les RNs ne vont pas retourner deux sets de paramètres identiques.

Une solution est jugé être satisfaisante lorsqu'elle s'approche de l'optimum global proposé par la librairie de benchmark TSPLib. Dans le cas général des solutions au-dessous de 10000 sont jugées acceptables pour BERLIN52 [3][23].

## IV. RESULTATS ET DISCUSSION

Notre algorithme converge après 5 itérations retournant ainsi une configuration qui a permis d'atteindre un cout de fonction objectif de 7574. Le taux d'erreur calculé de cette solution finale est 0.42%, ce qui est plus que satisfaisant pour un problème combinatoire de la taille de BERLIN52. Avec $7.755*10^{65}$ combinaisons possibles BERLIN52 est considérée une instance moyenne à grande [3, 23].

Le tableau 1 montre l'ensemble des configurations proposées par l'approche à travers les itérations ACO-RN.

**Tab 1 – Liste des configurations proposées par le RN**

| Itération | α | β | ρ | χ |
|---|---|---|---|---|
| 1 | 2,8004 | 6,1138 | 0,4129 | 6,9562 |
| 2 | 2,0248 | 2,5005 | 0,2796 | 5,7632 |
| 3 | 1,0455 | 2,0521 | 0,2086 | 3,9383 |
| 4 | 1,6882 | 2,3847 | 0,3051 | 2,9141 |
| 5 | 0,9627 | 1,9676 | 0,1752 | 3,9102 |

La différence entre les configurations proposées est intéressante, dans la mesure où les RNs adaptent les paramètres au fur et à mesure que de nouvelles données sont présentes. On remarque que les RNs ne proposent pas la même configuration deux fois ce qui confirme l'apprentissage sur la base actualisée.

La fonction objectif converge rapidement, la figure 5 montre cette évolution. On remarque que les premiers paramètres proposés donnent des résultats semblables aux paramètres aléatoires de la base d'apprentissage : un cout de 23343 avec une erreur de 67 %.

La deuxième itération propose des paramètres qui ont permis d'atteindre un cout de 10095 avec une erreur de 25%. Les trois dernières itérations proposent des configurations plus précises permettant d'atteindre des résultats proches de l'optimum global avec des taux d'erreurs inferieurs à 1%.

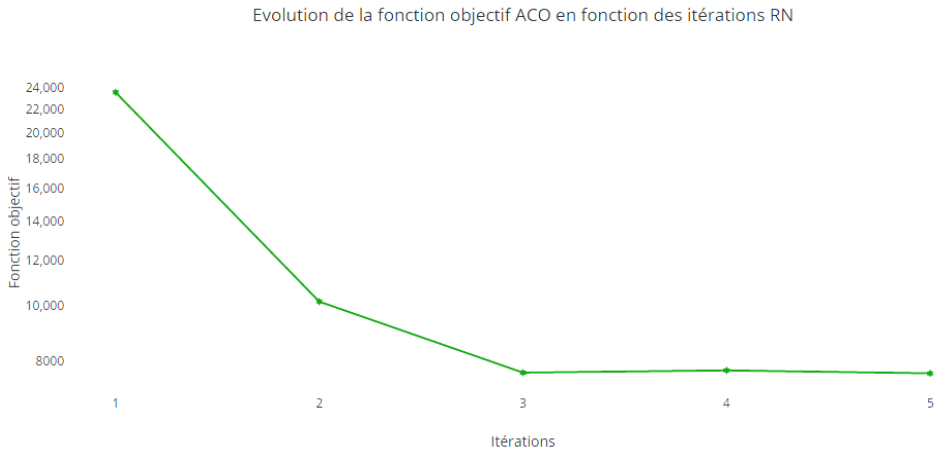


**Fig 5 – Evolution de la qualité de la fonction objectif dans les itérations ACO/RN**

La meilleure configuration est proposée dans la dernière itération avec un cout de 7574 et une erreur de 0.42%. A noter que continuer les itérations au-delà de la cinquième, ne conduit pas à proposer la configuration permettant de trouver l'optimum global avec 0.0% d'erreur.

## V. CONCLUSION

Le travail présenté dans ce papier avait pour objectif d'assister le pré réglage (tuning) des ACO en utilisant l'intelligence artificielle. Nous implémentons pour cela un perceptron multicouches utilisant l'algorithme de backpropagation Levenberg-Marquardt.

Le réglage se fait en mode offline, on alterne des instances ACO/RN pour guider les RNs dans leur recherche d'une configuration idéale des paramètres ACO.

Les RNs ont réussi à retourner un set de paramètres permettant d'atteindre des résultats de recherche plus que satisfaisants.

Ceci dit le comportement de l'hybride ACO-RN peut être amélioré en utilisant une base d'apprentissage plus riche ou contenant des lignes plus significatives pour l'algorithme d'apprentissage.

La base actuelle a été générée d'une façon aléatoire. Nous pensons qu'une base contenant des paramètres choisis soigneusement pourrait conduire à de meilleurs résultats (dans ce cas réduire l'erreur de 0.42% à 0.0%) pour atteindre l'optimum global du TSP.

Le modèle de RN implémenté dans l'hybride est configuré en utilisant les recommandations présentes dans la littérature, Ceci ne garantit en aucun que le modèle adopté est le mieux adapté pour l'hybridation. D'autres pistes peuvent donc s'illustrer pour trouver la bonne configuration RN adaptée à l'hybridation. Ceci requiert des expériences numériques plus approfondies et avec des bases plus riches.

La finalité du travail et de combiner les capacités de recherche des deux intelligences afin de résoudre un des problèmes les plus pertinents en logistique et production. Sans un bon tuning l'ACO ne peut résoudre efficacement le problème traité.

D'une façon plus précise, il s'agissait de prouver que la combinaison des deux familles d'intelligence artificielle peut conduire à trouver l'optimum global. A travers les itérations l'ACO éclaircit le MLP dans son apprentissage, et le MLP configure l'ACO pour résoudre la présente tache de recherche combinatoire.

Pour conclure, des opportunités de recherche peuvent se montrer pour avoir un autre modèle d'hybridation prenant en charge l'aspect online du tuning. D'autres pistes peuvent être explorées pour l'utilisation d'une autre approche d'intelligence artificielle comme les réseaux bayésiens.

## REFERENCES :


[1] VERGARA, F. Elizabeth, KHOUJA, Moutaz, et MICHALEWICZ, Zbigniew. An evolutionary algorithm for optimizing material flow in supply chains. Computers & Industrial Engineering, 2002, vol. 43, no 3, p. 407-421.

[2] AGHEZZAF, El-Houssaine, RAA, Birger, et VAN LANDEGHEM, Hendrik. Modeling inventory routing problems in supply chains of high consumption products. European Journal of Operational Research, 2006, vol. 169, no 3, p. 1048-1063.

[3] PAPADIMITRIOU, Christos H. The Euclidean travelling salesman problem is NP-complete. Theoretical Computer Science, 1977, vol. 4, no 3, p. 237-244.

[4] A. Colorni, M. Dorigo, and V. Maniezzo, "Distributed optimization by ant colonies," in Proceedings of the first European conference on artificial life, 1991, pp. 134-142.

[5] PELLEGRINI, Paola, STÜTZLE, Thomas, et BIRATTARI, Mauro. Off-line vs. On-line Tuning: A Study on\ mathcal {MAX--MIN} Ant System for the TSP. In : Swarm Intelligence. Springer Berlin Heidelberg, 2010. p. 239-250.

[6] G. Gutin and D. Karapetyan, "A memetic algorithm for the generalized traveling salesman problem," *Natural Computing*, vol. 9, pp. 47-60, 2010.

[7] A. Punnen, "The Traveling Salesman Problem: Applications, Formulations and Variations," in *The Traveling Salesman Problem and Its Variations*. vol. 12, G. Gutin and A. Punnen, Eds., ed: Springer US, 2007, pp. 1-28.

[8] G. Gutin and A. P. Punnen, *The traveling salesman problem and its variations* vol. 12: Springer Science & Business Media, 2002.

[9] M. Dorigo and L. M. Gambardella, "Ant colony system: a cooperative learning approach to the traveling salesman problem," *IEEE Transactions on Evolutionary Computation*, vol. 1, pp. 53-66, 1997.

[10] M. Dorigo, V. Maniezzo, and A. Colorni, "Ant system: optimization by a colony of cooperating agents," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 26, pp. 29-41, 1996.

[11] J.-L. Deneubourg, S. Aron, S. Goss, and J. M. Pasteels, "The self-organizing exploratory pattern of the argentine ant," *Journal of insect behavior*, vol. 3, pp. 159-168, 1990.

[12] DORIGO, Marco, MANIEZZO, Vittorio, et COLORNI, Alberto. The ant system: An autocatalytic optimizing process. Technical report, 1991.

[13] B. Bullnheimer, R. F. Hartl, and C. Strauss, "A new rank based version of the Ant System. A computational study," 1997.

[14] STÜTZLE, Thomas, LÓPEZ-IBÁNEZ, Manuel, PELLEGRINI, Paola, et al. Parameter adaptation in ant colony optimization. In : Autonomous search. Springer Berlin Heidelberg, 2012. p. 191-215.

[15] HOPFIELD, John J. Neural networks and physical systems with emergent collective computational abilities. Proceedings of the national academy of sciences, 1982, vol. 79, no 8, p. 2554-2558.

[16] MCCULLOCH, Warren S. et PITTS, Walter. A logical calculus of the ideas immanent in nervous activity. The bulletin of mathematical biophysics, 1943, vol. 5, no 4, p. 115-133.

[17] ROSENBLATT, Frank. The perceptron: a probabilistic model for information storage and organization in the brain. Psychological review, 1958, vol. 65, no 6, p. 386.

[18] MINSKY, Marvin et SEYMOUR, Papert. Perceptrons. 1969.

[19] HOPFIELD, John J. Neural networks and physical systems with emergent collective computational abilities. Proceedings of the national academy of sciences, 1982, vol. 79, no 8, p. 2554-2558.

[20] HECHT-NIELSEN, Robert. Theory of the backpropagation neural network. In : Neural Networks, 1989. IJCNN., International Joint Conference on. IEEE, 1989. p. 593-605.

[21] GARDNER, M. W. et DORLING, S. R. Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences. Atmospheric environment, 1998, vol. 32, no 14, p. 2627-2636.

[22] ABRAHAM, Ajith. Artificial neural networks. handbook of measuring system design, 2005.

[23] G. Reinelt, "TSPLIB—A Traveling Salesman Problem Library," ORSA Journal of Computing, vol. 3, pp. 376-384, 1991

[24] MORÉ, Jorge J. The Levenberg-Marquardt algorithm: implementation and theory. In : Numerical analysis. Springer Berlin Heidelberg, 1978. p. 105-116.