**Problème d’Optimisation pour**

**le Transport Ambulancier**

Noura YOUSFI#1, Najoua Elfethi\*2

#1 Etudiante Master Sciences des Données et Systèmes Intelligents,

Université Mohamed Premier, FP Nador ;

[yous.noura@gmail.com](mailto:yous.noura@gmail.com)

\*2 Etudiante Master Sciences des Données et Systèmes Intelligents,

Université Mohamed Premier, FP Nador ;

najouaelfethi@gmail.com

Abstract— Ce document élabore le problème de minimisation d’un chemin qui parcours un transport ambulancier, afin de réduire le temps pour arriver à l’hôpital cible dans la ville de Nador, afin de sauver des vies, en utilisant deux algorithmes : A\* et PSO (Particle Swarm Optimization).

I. Introduction

Notre projet sert à faire une minimisation des chemins (trouver le chemin optimal) que les ambulances peuvent circuler en faisant le départ du Centre-Ville pour atteindre l’hôpital Laari Cheikh le plutôt possible afin de sauver des vies, pour ce faire on va travailler avec deux algorithmes d’optimisation et comparer entre leurs résultats pour savoir les forces et les faiblesses de chaque approche.

1. Modélisation du Problème

Le réseau routier ça peut être représenté sous forme de graphe G=(V,E) avec V est l’ensemble des intersections (là où il y a la possibilité de changer la route), représentés par des nœuds, E est l’ensemble des routes représentées par des arêtes, ces arêtes se caractérise par la distance (d) qui est la longueur de la route (en kilomètres) et le Trafic (y) est nombre de véhicules sur la route.

On a choisi de travailler sur un problème plus spécifique, c’est de savoir le chemin le plus court (pour arriver à l’hôpital Laari Cheikh du point de départ qui est le Centre-ville)

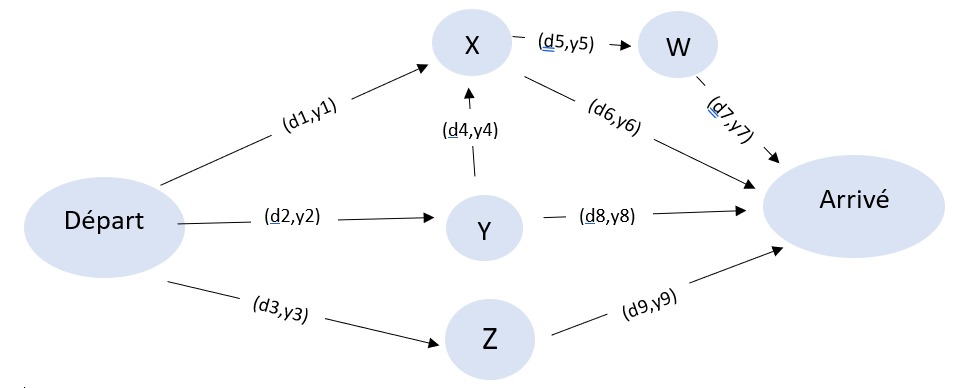


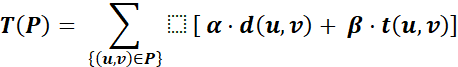
Fig. 1 Graphe représentant le problème

La fonction de coût du chemin optimal prend en compte la distance et le trafic pour calculer le temps de trajet entre deux points :

**Temps (u, v) = α ・ d(u, v) + β ・ t(u, v)**

* **d(u,v)** : la distance entre deux intersections (en km).
* **t(u,v)** : le trafic sur la route (nombre de véhicules).
* **α** : coefficient de pondération représente l’impact de la distance sur le temps de trajet
* **β**: coefficient de pondération représente l'impact du trafic sur le temps de trajet
* Les valeurs deα et βdoivent être positives : α>0 et β>0
* α + β = 1

L’objectif est de minimiser le temps total estimé sur un chemin d’un point de départ à un point d’arrivé :



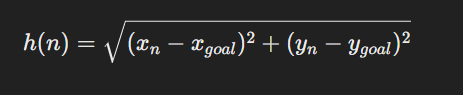
III.Présentation des Algorithmes

1. Algorithme A\*

L’algorithme A\* est une approche basée sur la recherche heuristique, qui utilise une fonction d’évaluation pour arriver à une solution optimale.

**Fonction Heuristique :**

On utilise la **distance euclidienne** entre un nœud de départ n et la destination goal:



**Fonction Coût :**



* : La fonction de coût total pour un état « n » est la somme de ces deux éléments ( et ).



* : Le coût réel pour atteindre l'état actuel à partir de l'état initial.



* : La fonction heuristique qui estime le coût restant pour atteindre l'objectif depuis l'état actuel.



1. Algorithme PSO (Particle Swarm Optimization)

L'Optimisation par Essaim de Particules (PSO) est une métaheuristique tirée du comportement collectif des essaims d’oiseaux, pour résoudre des problèmes d'optimisation avec une amélioration itérative de la solution selon une mesure de qualité. Chaque solution est représentée par une particule qui se déplace dans l’espace de recherche en fonction de sa propre expérience et de celle des autres.

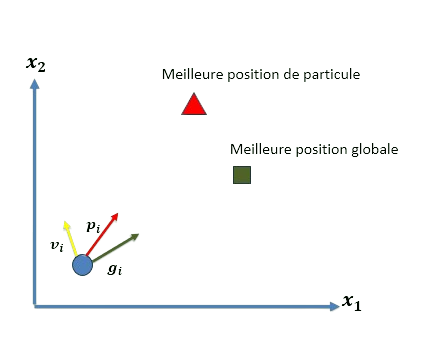


Fig. 2 Le schéma de génération des particules PSO

Chaque particule représente une solution dans un espace, et chaque particule met à jour sa position en fonction de sa position actuelle, de sa meilleure position (), de la meilleure position de l'essaim () et de sa Vélocité (). À chaque itération, la position des particules est mise à jour pour converger vers la solution optimale.



On était base sur le chemin optimal générer par A\* comme chemin initial pour l’algorithme PSO.

IV. Implémentation des Algorithmes

1. Environnement de Développement

OSMNX : bibliothèque python pour télécharger des graphes comme, afin d’analyser les réseaux routiers.

MATH : bibliothèque python pour les calculs mathématiques et géométriques.

HEAPQ : bibliothèque python utilisée dans les algorithmes de graphes.

MATPLOTLIB : bibliothèque python a pour objectif de visualisations graphiques en Python.

NETWORKX : bibliothèque python pour la manipulation des réseaux complexes.

1. Implémentation de l’Algorithme A\*

Pour implémenter l’algorithme A\* sur notre problème, tout d’abord on a récupéré les données spatiales en utilisant la bibliothèque OSMNX, puis définir les coordonnées GPS du nœud de départ le Centre-Ville et le nœud d’arrivée l’Hôpital Laari Cheikh dans la ville Nador, et trouver les nœuds les plus proches à ces coordonnées GPS, ensuite on a calculer l’heuristique entre un nœud n et le nœud d’arrivée en utilisant la distance Euclidienne, aussi on a calculé le temps estimé entre deux nœuds (prenant en considération la distance et le trafic).

Après l’implémentation de l’algorithme A\*, on affiche une représentation visuelle sur la carte du chemin optimal trouvé entre nœud de départ et nœud d’arrivée et aussi les différentes intersections du trajet (représenter en vert). On a utilisé le graph entier du ville Nador avec 5602 nœuds pour appliquer l’algorithme A\*.

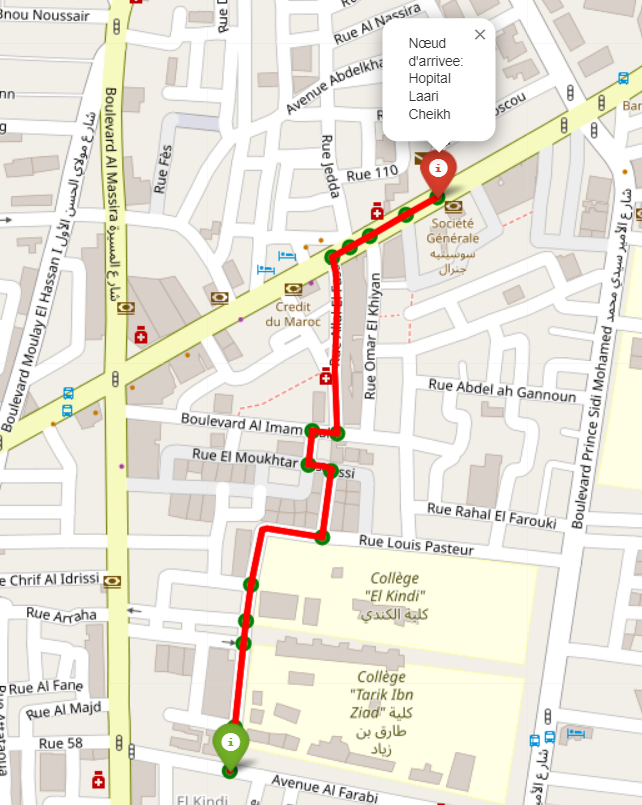


Fig. 3 visualisation du parcours optimal avec A\*

En se basant sur la Figure 3 ; le chemin optimal trouvée pour l’ambulance entre Centre-ville et l’Hôpital Laari Cheikh est le suivant : [359187007, 5720434617, 5620534295, 1892664265, 5720454258, 5720462285, 5720462284, 5720462280, 5720474735, 359191092, 359190659, 1784918407, 1785470206, 1785468821, 1784918008]. Le temps total estimée pour parcourir ce trajet est de 84.41071263611946 unités (combinaison de l’impact de la distance et du trafic).

Dans la représentation ci-dessous on a une visualisation d’un sous-graphe qui représente les différents chemins possibles pour arriver à l’Hôpital Laari Cheikh ainsi que le chemin optimal. Le nombre de ces chemins potentiels est liée au paramètre cutoff de 17 qui représente le nombre maximum de routes(edges) qu’un chemin potentiel peut avoir.



Fig. 4 chemins possibles entre nœud de départ et le nœud goal avec A\*

Pour tester la performance de A\* on a pensé de faire ça avec des sous-graphes de tailles différentes, et mesuré le temps d’exécution et enfin visualiser cette performance.

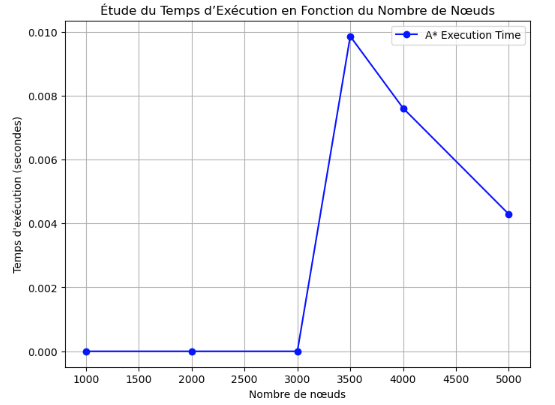


Fig. 5 Visualisation du temps d’exécution en fonction du nombre de nœuds

En se basant sur la Figure 5 ; les temps d’exécution sont très faibles, allant de 0.000 à 0.010 secondes. Cela montre que l’algorithme A\* est très efficace pour les graphes de taille modérée (jusqu’à 5000 nœuds).

1. Implémentation de l’Algorithme PSO

Pour ce faire en premier lieu, on a fait extraire un sous-graphe de 2891 nœuds avec 2km de distance, et retourner le coût total du chemin en utilisant la fonction de Fitness pour évaluer la qualité de chaque chemin potentiel, puis initialiser les particules (chemins potentiels) et faire leur mise à jour, après implémenter cet algorithme, et chercher les chemins possibles, et enfin afficher les résultats (le chemin optimal trouvé et son coût total) et les visualiser.

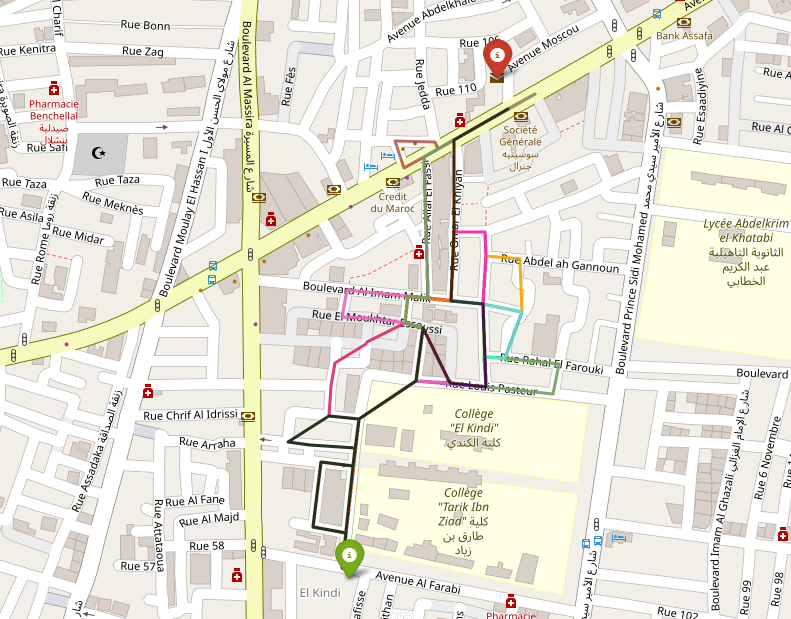


Fig. 6 Visualisation des chemins possibles avec PSO

En se basant sur la Figure 6 ; les différents chemins potentiels pour arriver à l’Hôpital sont représentés par différents couleurs, avec un cutoff de 17 on a eu 46 chemins potentiels, mais on va sélectionner juste 10 particules pour éviter des calcules trop lourds. Apres 50 itérations on a eu le chemin optimal suivant : [359187007, 5720434617, 5620534295, 1892664265, 5720454258, 5720462285, 5720462284, 5720462280, 5720474735, 359191092, 1892664513, 1611237316, 1591274567, 359190268, 5620534170], avec un coût total de 84.35743284152878 unités.

**Etude du Cout en fonction du Nombre d’Itérations avec PSO :**

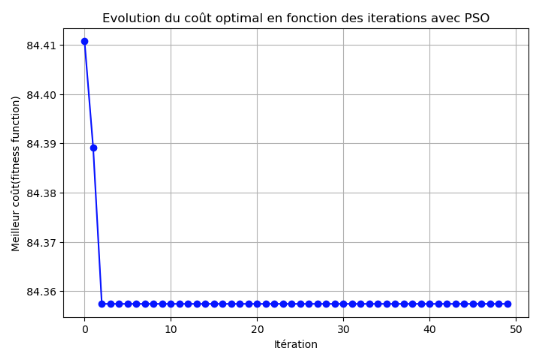


Fig. 7 Visualisation du cout optimal en fonctions des itérations avec PSO

En se basant sur la Figure 7 ; le cout optimal diminue légèrement avec les itérations, passant de 84,41 a 84.35. Cela indique que l’algorithme PSO améliore progressivement la qualité des solutions avec chaque itération. Au départ plusieurs chemins sont générés et à chaque itération le PSO fait la mise à jour des différents chemins et fait choisir les meilleures solutions actuelles en explorant d’autres alternatifs et si un nouveau chemin avec un cout plus bas est trouvé, il devient le meilleur, si aucun meilleur chemin n’est trouvé, la valeur reste constante.

**Etude du Temps d’Exécution en fonction du Nombre de Nœuds avec PSO :**

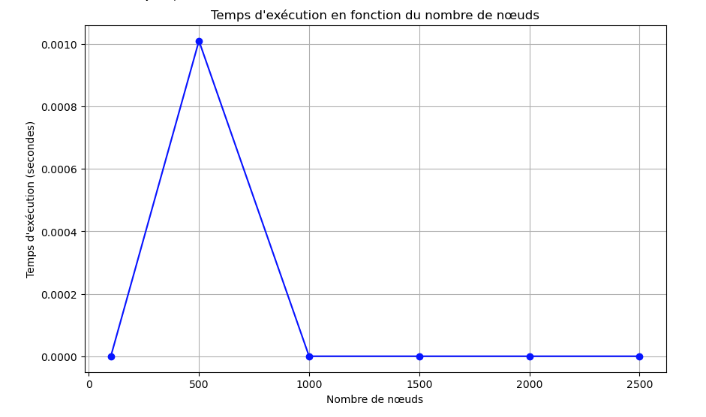


Fig. 8 Visualisation du temps d’exécution en fonctions du nombre de nœuds

En se basant sur la Figure 8 ; Le temps d’exécution de recherche du chemin optimal était mesuré pour chaque sous-graphe, les temps d'exécution sont extrêmement faibles, variant de 0,0000 à 0,0010 secondes. Cela indique que l'algorithme PSO est très efficace pour les graphes de taille modérée (jusqu'à 2500 nœuds).

**Etude du Temps d’Exécution en fonction du Nombre des Itérations avec PSO :**

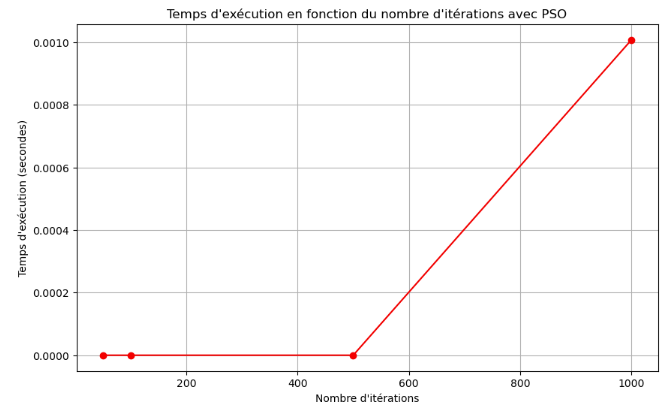


Fig. 9 Visualisation du temps d’exécution en fonctions du nombre d’itérations

En se basant sur la Figure 9 ; les temps d'exécution sont très faibles, variant de 0,0000 à 0,0010 secondes. Cela indique que l'algorithme PSO est très rapide pour le problème, même avec un nombre élevé d'itérations (jusqu'à 1000).

V. Expérimentations et Résultats

1. Analyse des Performances des Algorithmes sur des Graphes de petite taille

L’algorithme le plus rapide pour des graphes de petite taille en fonctions du temps d'exécution c’est A\* qui montre des temps d'exécution extrêmement faibles pour des graphes de petite à moyenne taille (jusqu'à 5000 nœuds). Bien que PSO est également rapide, il est légèrement plus lent que A\* pour des graphes de petite taille.

La cause de l’efficacité de A\* sur des graphes de petites tailles est l’utilisation d’une heuristique pour guider la recherche, ce qui lui permet de trouver rapidement le chemin optimal sans explorer inutilement l'espace de recherche. En revanche PSO est un algorithme d'optimisation basé sur les particules qui explore l'espace de recherche de manière itérative.

#### L’algorithme A\* devient inefficace lorsque la taille du graphe augmente en fonction du temps d'exécution, car son heuristique lui permet de réduire considérablement l'espace de recherche à explorer. Même si PSO est rapide pour des graphes de petite taille, son temps d'exécution augmente de manière significative lorsque la taille du graphe augmente.

1. Analyse de l’Impact du Nombre d’Itérations sur le Temps d’Exécution

Pour l'algorithme PSO, d'après l'analyse graphique on observe une augmentation significative du temps d'exécution à partir de 600-800 itérations, mais elle reste relativement faible. Cela montre que PSO est capable de gérer un grand nombre d'itérations sans dégradation majeure de ses performances.

1. Analyse des Paramètres des Algorithmes qui ont Impact sur le Temps d’Exécution

Pour l’algorithme A\* on a deux paramètres critiques qui influence le temps d’exécution ; dans un premier temps on a l'heuristique (par exemple, la distance euclidienne dans notre cas), s’il est bien choisi on peut avoir une réduction de l'espace de recherche en guidant l'algorithme vers la solution optimale plus rapidement. Ensuite on a la taille du graphe c’est-à-dire le nombre de nœuds et d'arêtes dans le graphe influence directement le temps d'exécution. Plus le graphe est grand, plus A\* doit explorer de nœuds, ce qui augmente le temps de calcul.

Pour PSO on distingue 3 paramètres importants qui peuvent influencer le temps d’exécution ; plus le nombre d’itérations est élevé, plus l'algorithme explore l'espace de recherche, ce qui augmente le temps de calcul. Ensuite on a la taille de la population (nombre de particules) s’il est grand cela permet une exploration plus approfondie de l'espace de recherche, mais cela augmente également le temps d'exécution, car chaque particule doit être mise à jour à chaque itération. Enfin on a la fonction de coût (Fitness).

1. Le choix de bons paramètres pour optimiser chaque méthode

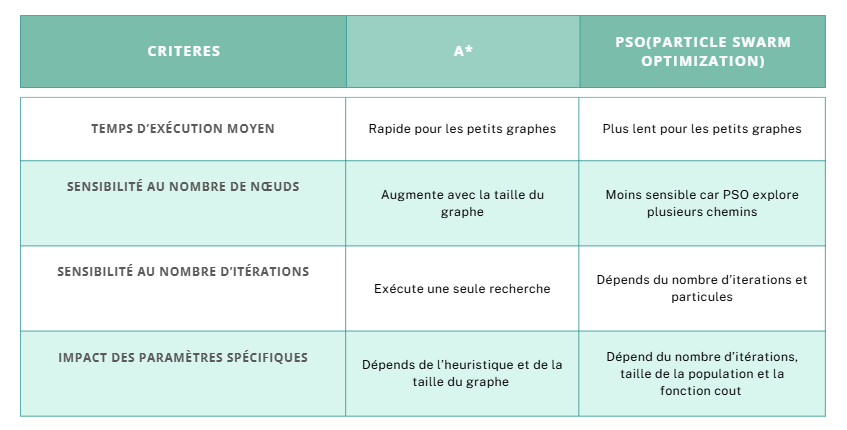
Pour l’algorithme A\*, le choix d'une heuristique adaptée par exemple, la distance euclidienne est souvent un bon choix pour des graphes spatiaux. Aussi la réduction de la taille du graphe en supprimant les nœuds et arêtes inutiles ou en utilisant des techniques de prétraitement.

Pour l’algorithme PSO, le choix d'un nombre maximal d'itérations raisonnable en fonction de la complexité du problème reste une bonne solution. Ensuite on peut faire utiliser un nombre de particules suffisant pour explorer l'espace de recherche, mais pas trop grand pour éviter des calculs inutiles. Aussi l’optimisation de la fonction de si possible pour réduire le temps de calcul.

VI. Comparaison et Analyse Générale

1. Comparaison des deux algorithmes A\* et PSO :

Fig. 10 Comparaison de A\* et PSO



1. Forces et Faiblesses de A\* et PSO :

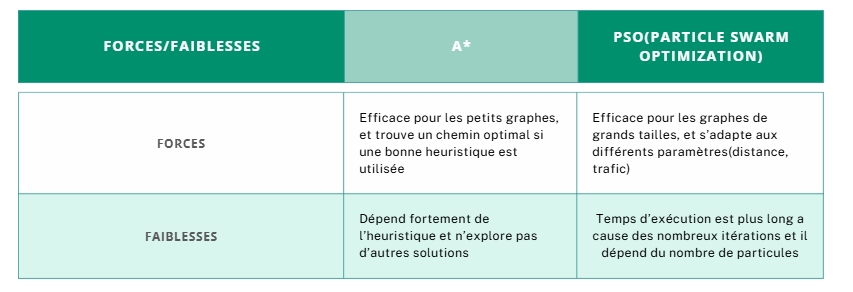


Fig. 11 Forces et Faiblesses de A\* et PSO

1. Algorithme efficace pour notre cas

Dans notre cas d’étude, PSO est le mieux adapte pour un réseau routier réel (graphe de grand taille) où il est crucial d’explorer plusieurs chemins possibles. En revanche, A\* est efficace pour les petits graphes ou le nombre de nœuds est limité.

VII. Conclusion

Dans ce travail, nous avons étudié le problème d’optimisation du transport ambulancier en utilisant deux algorithmes : A\* et PSO. A\* s’est révélé très efficace pour les graphiques de petite taille. En revanche, PSO a montré sa robustesse pour des graphes de grande taille.

L’analyse des performances a montré que A\* est plus rapide pour les petits graphes, tandis que PSO est mieux adapté aux grands réseaux routiers où l’exploration de plusieurs alternatives est nécessaire. Cependant, le temps d’exécution de PSO augmente avec le nombre d’itérations et la taille du graphe.

References

# [1] [Mohammad Haziq Ibrahim](https://ieeexplore.ieee.org/author/37089685972); [Swee Peng Ang](https://ieeexplore.ieee.org/author/38543081200); [Muhammad Norfauzi Dani](https://ieeexplore.ieee.org/author/37086292644); [Mohammad Ishlah Rahman](https://ieeexplore.ieee.org/author/37089724835); [Rafidah Petra](https://ieeexplore.ieee.org/author/37085711957); [Sheik Mohammed Sulthan](https://ieeexplore.ieee.org/author/37089567363),“*Optimizing Step-Size of Perturb & Observe and Incremental Conductance MPPT Techniques Using PSO for Grid-Tied PV System*.”

# [2] [Federico Marini](https://www.sciencedirect.com/author/7005080189/federico-marini), “*Particle swarm optimization (PSO). A tutorial.*”

# [3] [Shang Erke](https://journals.sagepub.com/doi/full/10.1177/1729881420962263#con1) <https://orcid.org/0000-0002-6669-3933>, [Dai Bin](https://journals.sagepub.com/doi/full/10.1177/1729881420962263#con2), […], and [Zhao Dawei](https://journals.sagepub.com/doi/full/10.1177/1729881420962263#con6) , “*An improved A-Star based path planning algorithm for autonomous land vehicles*”

# [4] Najiba Sbihi,” *Algorithme de recherche d'un stable de cardinalite maximum dans un graphe sans etoile*.”

**[5]** Serge Bouchardon ,“*La valeur heuristique de la littérature numérique*.”

**[6]** Jacques Martinet and Henri Cohen , “*Étude heuristique des groupes de classes des corps de nombres. ”*