Tuning manuel de l'algorithme d'optimisation par essaims de particules appliqué au problème du voyageur de commerce

Abdelhakim GHARIB
LISER laboratory, ENSEM
KM7, BP 8118 Route El Jadida
Casablanca, Morocco
gharib.abdelhakim@gmail.com

Jamal BENHRA
LISER laboratory, ENSEM
KM7, BP 8118 Route El Jadida
Casablanca, Morocco
jbenhra@hotmail.com

Abstract— **L'optimisation par essaim de particules (PSO) est une méta-heuristique basée sur la collaboration des individus entre eux. Elle a été conçue pour trouver de bonnes solutions aux problèmes d'optimisation. Le comportement de l'algorithme d'optimisation par essaim de particules dépend fortement des valeurs définies pour ses paramètres. La bonne configuration des paramètres de l'algorithme PSO est une tâche difficile et prend beaucoup de temps puisqu'elle nécessite l'évaluation d'un grand nombre de combinaisons des paramètres pour trouver le réglage le plus approprié. Le but du présent article est de faire un réglage manuel pour l'algorithme de PSO, appliqué au problème de voyageur de commerce, afin de trouver la meilleure combinaison possible entre les différents paramètres et d'étudier l'influence de ces derniers sur la solution finale. (*Abstract*)**

Keywords—Optimisation; heuristique; essaim de particules; tuning; problème du voyageur de commerce (key words)

I. INTRODUCTION

Dans de nombreuses applications dans le monde réel, il y a toujours un besoin de trouver des configurations optimales à partir d'un ensemble discret d'objets. Ceci est connu en tant que problème d'optimisation combinatoire. Alors que beaucoup de ces problèmes d'optimisation combinatoire peuvent être résolus en un temps polynomial, une majorité appartient à la classe des problèmes NP-difficiles [1]. Pour faire face à ces problèmes d'optimisation, des approximations et des algorithmes heuristiques ont été utilisés comme un compromis entre la qualité de la solution et le temps de calcul [2]. Une classe d'algorithmes heuristiques, les algorithmes méta-heuristiques, a été développée et fournie des résultats prometteurs dans le domaine de l'optimisation combinatoire. Cette classe comprend: le recuit simulé (SA), la recherche tabou, les algorithmes génétiques (GA) [3], L'optimisation par colonies de fourmis (ACO) [4]. Etc.

Une nouvelle famille d'algorithmes méta-heuristiques, appelée : Optimisation par essaim de particules (PSO) [5], a été mise au point et a prouvé son efficacité. Comme d'autres algorithmes méta-heuristiques qui sont inspirés de la nature, l'algorithme PSO est considéré comme étant une technique de recherche adaptative basée sur la collaboration entre les individus. L'idée est qu'un groupe d'individus, peu intelligents, peut avoir une organisation globale complexe.

Nous allons utiliser l'optimisation par essaim de particules (PSO) pour résoudre un problème NP-Difficile et qui est le problème du voyageur de commerce (TSP) tout en effectuant un tunning manuel à notre algorithme PSO. Ceci, en utilisant des instances de référence tirées de la bibliothèque TSPLib [6].

Le reste de l'article est organisé comme suit: La section 2 décrit l'état de l'art. La section 3 présente le problème traité et sa modélisation. La section 4 porte sur l'analyse des résultats et enfin, la section 5 conclut ce travail.

II. ETAT DE L'ART

A. Présentation de l'optimisation par essaim de particules

L'optimisation par essaim de particules (PSO) est une méthode d'optimisation stochastique développée par Eberhart et Kennedy en 1995 [5]. Elle a été inspirée du comportement social des animaux vivant dans des essaims, notamment des bancs de poissons et des vols groupés d'oiseaux.

Dans son application à des problèmes d'optimisation, cette méthode repose sur un ensemble d'individus, à l'origine disposés de façon aléatoire, appelées particules et qui se déplacent dans l'espace de recherche. Chacune des particules est considérée comme une solution du problème et possède une position X_{id} et une vitesse V_{id}. En outre, chaque particule a une mémoire de sa meilleure position visitée P_{id} et aussi de celle de son voisinage P_{gd}. L'évolution des équations de l'algorithme dans le cas continu est déterminée comme suit:

$$\begin{cases} V_{id}^{t+1} = \omega\, V_{id}^{t} + C_1R_1\,(P_{id} - X_{id}) + C_2R_2\,(P_{gd} - X_{id}) & (1) \\ X_{id}^{t+1} = X_{id}^{t} + V_{id}^{t+1} & (2) \end{cases}$$

On notera que ω désigne le coefficient d'inertie, les coefficients C_1 et C_2 sont des constantes déterminées de façon empirique en fonction de la relation $C_1 + C_2 \leq 4$ et enfin, R_1 et R_2 sont des nombres positifs aléatoires qui suivent une distribution uniforme sur [0,1] [5].

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.

La stratégie de déplacement d'une particule, comme elle est représentée sur la Figure 1, est influencée par les trois composants suivants:
1. Un élément d'inertie ($\omega\ V_{id}^t$): la particule tend à suivre sa direction actuelle de mouvement;
2. Une composante cognitive ($C_1R_1\ (P_{id} - X_{id})$): la particule tend à se déplacer vers la meilleure position par laquelle elle est déjà passée;
3. Une composante sociale ($C_2R_2\ (P_{gd} - X_{id})$): la particule a tendance à compter sur l'expérience de son voisinage et donc elle se dirige vers la meilleure position déjà atteinte par ses voisins.

Kennedy et Eberhart [7] ont aussi proposé une version discrète et binaire de l'algorithme PSO en définissant les trajectoires et les vitesses des particules en termes de changement de probabilité qu'un bit est positionné à 0 ou à 1 [8]. Les particules se déplacent dans un espace d'état restreint de 0 et 1 avec une certaine probabilité qui est une fonction de facteurs individuels et sociaux. La probabilité de $X_{id}^t = 1$, $Pr(X_{id} = 1)$, est une fonction de X_{id}^{t-1}, V_{id}^{t-1}, P_{id}^{t-1} et P_{gd}^{t-1}. La probabilité de $X_{id}^t = 0$ est égale à $1 - Pr(X_{id} = 1)$. Ainsi l'équation (2), est remplacée par l'équation (3) où R_3 est un nombre aléatoire. $\psi(v_{id}^t)$ est une transformation logistique qui peut contraindre V_{id}^t à l'intervalle [0,1] et peut être considérée comme une probabilité.

$$X_{id}^t = \begin{cases} 1, \text{ si } R_3 < \psi\ (V_{id}^t) \\ 0, \text{ autrement} \end{cases}$$

Dans notre papier, nous appliquons la première définition de l'algorithme PSO.

B. *Optimisation à l'aide du PSO*

L'optimisation par essaim de particules a été appliquée dans de nombreux problèmes. Cette section mentionne brièvement certaines de ces applications.

H. Yoshida et al. [9] ont appliqué le PSO dans le domaine des systèmes d'alimentation pour minimiser la perte de puissance active d'un réseau électrique. Ce fut la première application du PSO dans le domaine de l'énergie électrique. El - Gallad et al. [10] ont adapté le PSO pour résoudre le problème traditionnel de la distribution économique. Van der Merwe et Engelhrecht [11] ont introduit deux méthodes en utilisant le PSO pour résoudre le problème de partitionnement des données. Zhu et al. [12] ont développé une stratégie de recherche à base du PSO pour le problème de temps du VRP et qui peut être utilisée pour la réduction de la pollution de l'air dans les logistiques inverse et amont. Nan et al. [13], quant à eux, ont utilisé le PSO pour résoudre l'emplacement des usines de fabrication dans la logistique inverse. La distribution des biens est aussi déterminée par l'algorithme PSO, pour minimiser les coûts. Abdelhalim et al [14] ont utilisé le PSO pour le problème de partitionnement Hardware / Software. Băutu et al. [15] ont discuté comment le PSO peut être utilisé pour minimiser l'énergie d'un système composé de points de charges répulsifs confinés dans une sphère. Chiang et al. [16] et Tu et al. [17] ont démontré l'application du PSO au problème du Job Shop.

C. *Application du PSO au TSP*

Au cours des dernières années, et puisque le TSP est un bon terrain d'essai des techniques d'optimisation, de nombreux chercheurs dans divers domaines ont consacré leur temps pour trouver des méthodes efficaces de résolution du TSP.

Parmi ces techniques d'optimisation, nous trouvons l'optimisation par essaim de particules.

Hendtlass et al. [18] ont proposé d'introduire une mémoire à chaque particule dans le but d'améliorer la diversité de l'algorithme.

Clerc et al. [19] ont développé plusieurs variantes de l'algorithme PSO et les ont appliqué au problème du TSP asymétrique (br17.atsp instance). Dans leur algorithme, les positions sont définies comme des tournées du TSP et sont représentées dans des vecteurs de permutation de |N| nœuds du graphe correspondant à l'instance considérée.

Wang et al. [20] ont proposé un opérateur de swap et une séquence de swap. Ce papier a conçu un PSO spécial, mais n'a tout de même pas amélioré la formule de mise à jour.

Pang et al. [21] ont étendu le travail de Wang et al. [20]. Leurs algorithme alterne entre l'espace continu et l'espace discret (permutation). Pour éviter une convergence prématurée, ils ont utilisé un opérateur chaotique. Cet opérateur change de façon aléatoire la position et la vitesse dans l'espace continu tout en multipliant ces vecteurs par un nombre aléatoire. Pang et al. [22] ont également présenté un fuzzy-based PSO pour le TSP. Ils ont appliqué leur algorithme aux instances burma14 et berlin52.

Dans le travail de Shi et al. [8], des algorithmes basés sur le PSO sont présentés pour le TSP et le TSP généralisé où une stratégie de recherche aléatoire et des techniques de passage sont utilisées pour accélérer la vitesse de convergence.
Comparé aux algorithmes existants basés sur l'intelligence des essaims pour résoudre le TSP, il a été montré que la taille des problèmes étudiés peut être augmentée en utilisant l'algorithme proposé.

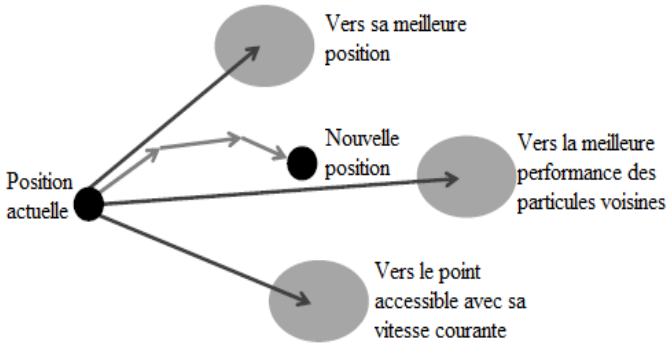

Figure1: Mouvement des particules

III. PROBLÉMATIQUE ET MODÉLISATION

A. *Définition du problème*

Le voyageur de commerce doit visiter n villes et doit passer une seule fois par chacune d'elles. Généralement, on commence par choisir n'importe quelle ville et on termine par un retour à cette ville (voir la figure 2). Nous devons minimiser la distance totale parcourue sachant que les distances entre les villes sont connues. La notion de distance peut être remplacée par le temps ou le coût. Le problème de voyageur de commerce est un problème NP-difficile. Cela signifie que le temps nécessaire pour trouver une solution optimale augmente de façon exponentielle en fonction de l'ampleur du problème.

Selon Kemighan [23], la définition du problème du voyageur de commerce est la suivante:
Etant donné un graphe G = (N, A), où N = {v_0, v_1 ... .. v_n} est l'ensemble des nœuds (villes). Soit A = {(v_i, v_j) / v_i, v_j Є N, i <j} l'ensemble des arrêts qui relient les nœuds si les distances sont symétriques et A = {(v_i, v_j) / v_i, v_j Є N i ≠ j} représente les arcs si les distances sont asymétriques. En d'autres termes, le problème est symétrique si la distance du voyage de la ville A à la ville B est la même que celle du voyage inverse (de la ville B à la ville A).

Dans la littérature, nous constatons que la recherche pour le cycle hamiltonien dans un graphe peut être faite par l'un des deux types d'algorithmes [24]:

a) un algorithme qui génère tous les cycles donc il est exponentiel;

b) un algorithme qui teste des solutions (en un temps polynomial).

B. *Compléxité du problème de voyageur de commerce*

Dans le cas symétrique, qui sera le cas de notre étude, un calcul de la complexité du problème montre que le nombre de solutions possibles est donné par la formule :
(n-1) ! / 2; où n est le nombre de villes.

Considérant que le temps de calcul pour un voyage est égal à 1 µs, nous constatons effectivement que le temps de calcul est assez grand comme indiqué dans le tableau 1.

TABLE I. TEMPS NECESSAIRE AU CALCUL ET NOMBRE DE POSSIBILITES

Nombre de villes	Nombre de possibilités	Temps nécessaire au calcul
4	3	3 µs
14	3113510400	52 min
20	60E+15	1928 ans
52	7.8E+65	2.5E+43 millard d'années
280	4.66E+157	2.5E+43 milliard d'années

C. *Formulation mathématique du problème de voyageur de commerce*

Les variables nécessaires pour modéliser le problème du voyageur de commerce mathématiquement sont définies comme suit:
d_{ab} = distance entre la ville a et la ville b;
n = nombre de villes;
x_{ab} = variable binaire qui prend la valeur 1 si la ville a est visitée immédiatement avant la ville b. Sinon, cette variable est mise à 0.
Ainsi, nous devons réduire la longueur du cycle hamiltonien, donc la fonction objectif sera:

$$Z = \sum_{a=1}^{n} \sum_{b=1}^{n} d_{ab}\, x_{ab}$$

Les contraintes sont comme suit :

$$\sum_{a=1}^{n} x_{ab} = 1 \quad \text{Pour chaque nœud a (4)}$$

$$\sum_{b=1}^{n} x_{ab} = 1 \quad \text{Pour chaque nœud b (5)}$$

$$\sum_{a \in E} \sum_{b \in E} x_{ab} \leq |E| - 1 \quad \text{Pour E} \in \text{N avec } 2 \leq |E| \leq \text{n-1 (6)}$$

La contrainte (4) assure qu'on quitte chaque point une seule fois et la contrainte (5) vérifie que nous entrons une seule fois dans chaque point. Donc, l'unicité de visite de chaque point est respectée. Ensuite, on ajoute la contrainte (6) pour qu'il n'y ait pas de sous tours. Dans la contrainte (6), | E | est le cardinal de l'ensemble E.

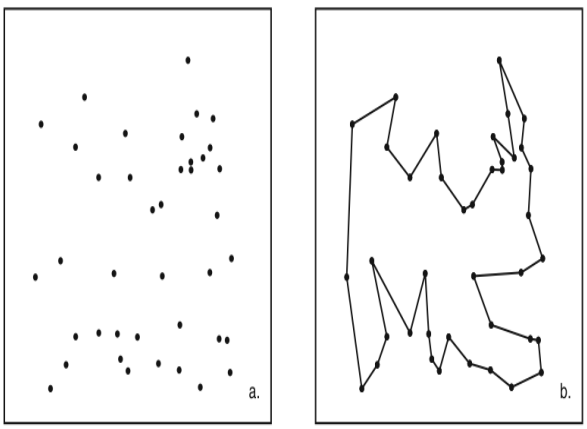

Figure 2 : Exemple de 40 nœuds (a) et du chemin qui les relie (b)

IV. RÉSULTATS ET DISCUSSION

A. *Simulation du problème*

Cette section présente les étapes suivies pour développer notre solution afin de résoudre le problème du voyageur de commerce en utilisant l'optimisation par essaim de particules.
Le lancement de l'algorithme PSO est fait selon le schéma de la figure 3. La logique des opérations effectuées dans notre code est la suivante :
Après la lecture des coordonnées des villes et le calcul des distances entre ces dernières, vient la partie de l'initialisation des particules. Cette étape est très importante avant le lancement de l'algorithme qui consiste à:
1. Initialisation des positions des particules par l'affectation d'une séquence aléatoire de villes pour chaque particule.
2. Initialisation des vitesses des particules avec des valeurs nulles.
3. Initialisation d'un tableau « X_Pbest » - pointeur vers un tableau de taille particles_count*cities_count permettant de stocker au niveau de chacune de ses lignes dont l'indice fait référence à la particule, la meilleure solution trouvée par cette particule depuis le début de l'exécution - qui est lié aux meilleures solutions trouvées par chacune des particules suivant les positions aléatoires de départ.
4. Initialisation d'un tableau « X_Gbest » - pointeur vers un tableau à une dimension de taille cities_count permettant de stocker la meilleure solution trouvée par toutes les particules et ceci depuis le début de l'exécution - en calculant la meilleure solution trouvée parmi les positions aléatoires de départ.
Nous utilisons également une fonction qui renvoie les particules à l'espace de solution:
Cette fonction permet de faire retourner les particules à l'espace des solutions une fois que les mouvements ont été appliqués aux positions actuelles. Elle prend comme argument la table des positions obtenue après déplacement et agit sur la même table en changeant les villes déjà visitées par les villes les plus proches qui n'ont jamais été visitées.

Pour tester notre algorithme, nous avons décidé de lancer une expérience avec variation de la complexité du TSP.

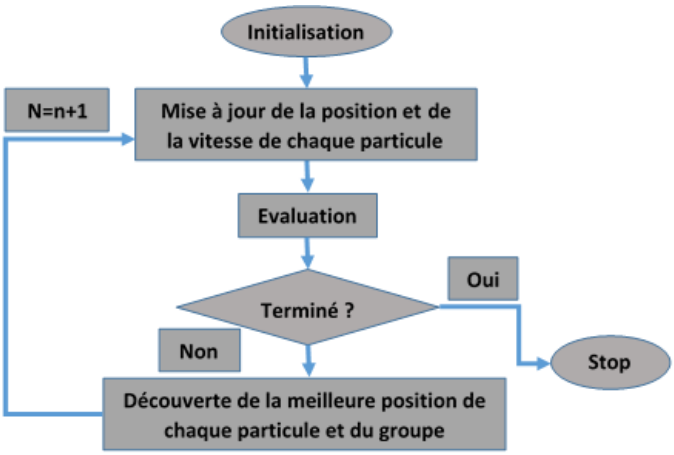

Figure3: Diagramme de lancement de l'algorithme PSO

L'expérience est basée sur des instances de benchmark, Berlin52, Eil101 et A280 [6] et qui sont tirées de la bibliothèque TSPLIB.
La plupart des problèmes utilisés comme benchmark peuvent être trouvés dans la TSPLIB :
http://elib.zib.de/pub/mptestdata/tsp/tsplib/tsp/index.html.
La TSPLIB offre les coordonnées des problèmes, le meilleur parcours et les solutions optimales pour les trois instances test que nous utilisons [6].
Nous utilisons également une instance (instance1) de 20 points que nous avons posé et dont les cordonnées sont respectivement les suivantes:
X {30, 50, 43, 40, 40, 36, 21, 8, 16, 7, 3, 80, 55, 60, 45, 1, 30, 45, 25, 3};
Y {5, 9, 10, 30, 20, 43, 48, 40, 36, 21, 18, 16, 27, 17, 80, 1, 12, 21, 65, 4};
Pour cette instance nous avons utilisé notre algorithme PSO, l'algorithme ACO et l'algorithme génétique (AG) utilisés par A. H. Sabry et al. [3]. Ces deux derniers donnent un résultat égal à 301.

Toutes les simulations ont été faites sur Windows 64 bits avec un ordinateur personnel dont la configuration est la suivante : Processeur i3-2370 cadencé à 2.40GHz, et ayant 4 GB de Ram.
Notre algorithme PSO a été développé sous JAVA.

B. *Tunning manuel du PSO appliqué au TSP*

Dans le but de trouver la meilleure solution pour notre problème, et puisque la solution dépend des quatre paramètres de l'algorithme PSO, nous avons décidé de faire un tunning manuel.

Ceci s'est effectué en faisant varier les valeurs des quatre paramètres (ω, C_1, C_2 et le nombre de particules) avec un maximum d'itérations égal à 1000.

La variation de ces paramètres s'est effectuée comme suit :

- Nombre de particules : 10/ 50/ 100/ 500/ 1000 (5 valeurs).
- ω : initialisé à partir de 0.11 avec un pas de 0.1 (10 valeurs)
- C_1 : initialisé à partir de 0.2 avec un pas de 0.2 (11 valeurs)
- C_2 : initialisé à partir de 0.2 avec un pas de 0.2 (11 valeurs)

Le total des simulations pour chaque instance est égal à :
5*10*11*11 = 6050.
Mais, avec la contrainte $C_1+C_2 <= 4$, ce nombre a diminué et est devenu 5900 simulations.
Le tableau 2 regroupe les meilleures solutions trouvées pour chacune des instances considérées et pour chaque cas aussi.
La figure 4 montre l'évolution de la solution en fonction du nombre de particules utilisé et la figure 5 trace les chemins obtenus pour les meilleures solutions (obtenues avec 1000 particules).

TABLE II RESULTATS DU TUNING

Instance	Nombre de particules	W	C1	C2	Meilleure solution trouvée
Instance1	10	0,11	1,6	2,0	329,9562
	50	0,41	1,2	1,6	328,04
	100	0,31	2,2	1,4	322,5443
	500	0,11	1,2	2,2	322,6882
	1000	0,31	1,4	2,2	302,71
Berlin52	10	0,31	1,6	1,2	17161,4289
	50	0,51	0,8	0,6	15490,8137
	100	0,21	1,2	2,2	14803,3198
	500	0,31	0,8	1,6	13657,6362
	1000	0,11	0,8	2,2	12568,8868
eil101	10	0,41	1,0	0,8	2171,2398
	50	0,21	1,2	1,4	1867,7627
	100	0,11	1,2	1,6	1619,8959
	500	0,21	1,2	1,8	1622,2073
	1000	0,11	1,4	1,6	1576,2973
a280	10	0,41	0,6	0,2	10578,3041
	50	0,51	0,6	0,2	8645,4
	100	0,21	0,4	0,2	8609,707
	500	0,21	0,4	0,2	7437,0231
	1000	0,31	0,3	0,4	7254,89

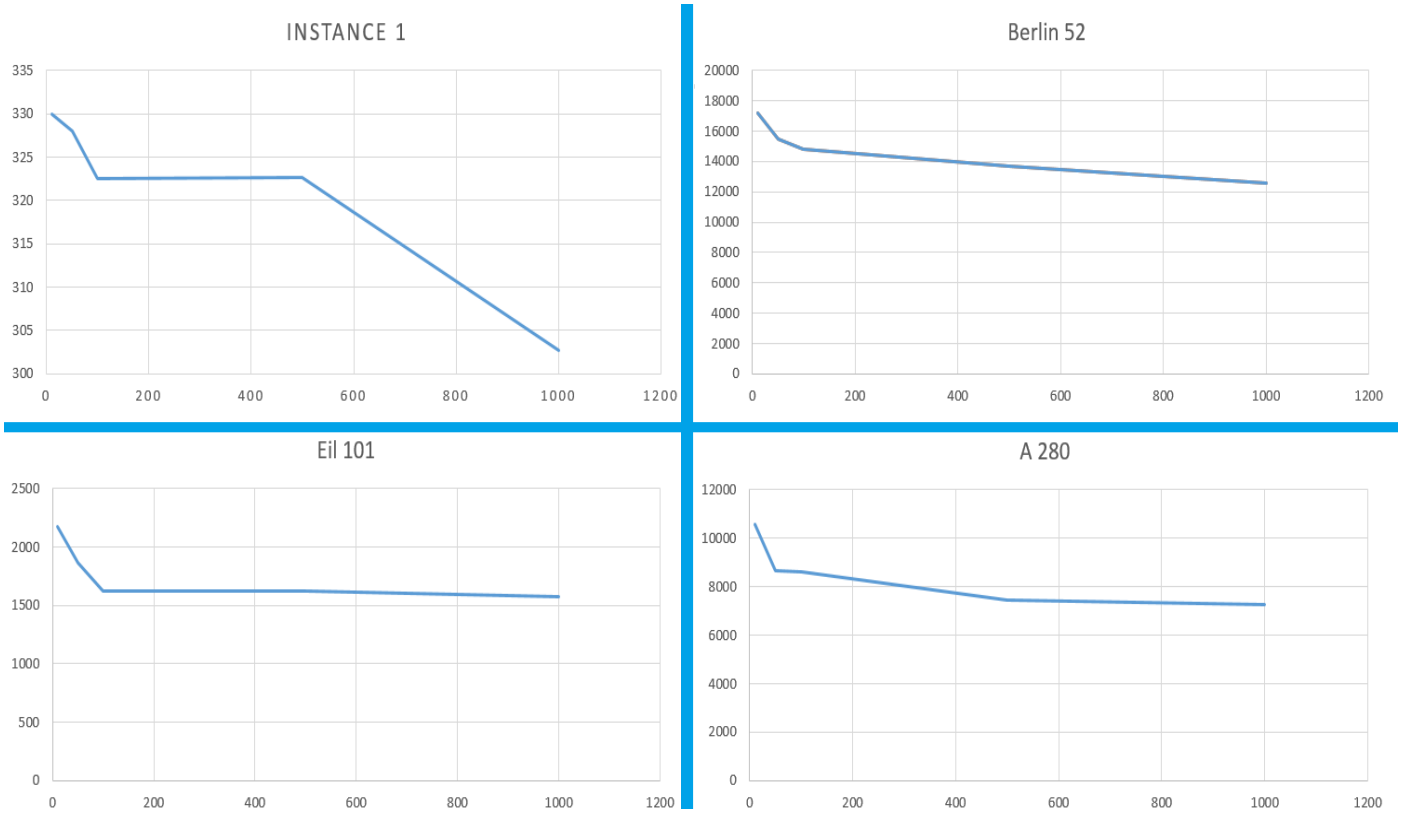

Figure 4 : Variation de la solution en fonction du nombre des particules

C. *Discussion des résultats*

Après avoir effectué notre tunning manuel, nous avons trouvé que l'augmentation du nombre de particules affine la qualité de la solution et ceci quelque soit les valeurs des trois autres paramètres.

Ceci s'explique par le fait que plus il y a de particules et plus l'espace de recherche est balayé.

Pourtant, et vu que l'espace de recherche varie selon « n^n » où « n » est le nombre de nœuds, alors peu importe le nombre de particules, ces dernières risquent de sortir de l'espace de solutions. Pour cela et afin de limiter l'espace de recherche, des méthodes ont été développées pour contenir et faire bouger les particules autour pour qu'elles ne sortent pas de l'espace des solutions [22].

Concernant les autres paramètres C_1, C_2 et ω, leurs variations dépendent de l'instance et nous ne pouvons pas faire de conclusion puisque la solution dépend des trois paramètres à la fois et ils ne peuvent pas être étudiés séparément.

V. CONCLUSION

Ce papier a présenté notre contribution dans la résolution du problème de voyageur de commerce en utilisant l'optimisation par essaims de particules dans son état standard et en effectuant un tuning manuel à ce dernier. Notre tuning a démontré que plus on augmente le nombre de particules et plus l'algorithme converge.

Mais, l'augmentation du nombre de particules augmente aussi le temps de traitement de ce dernier.

Pour cette raison, une extension de ce travail sera effectuée pour trouver un meilleur tunning basé sur les différents paramètres du PSO (ω, C_1, C_2 et le nombre de particules) et qui sera réalisé automatiquement en se basant sur les réseaux de neurones.

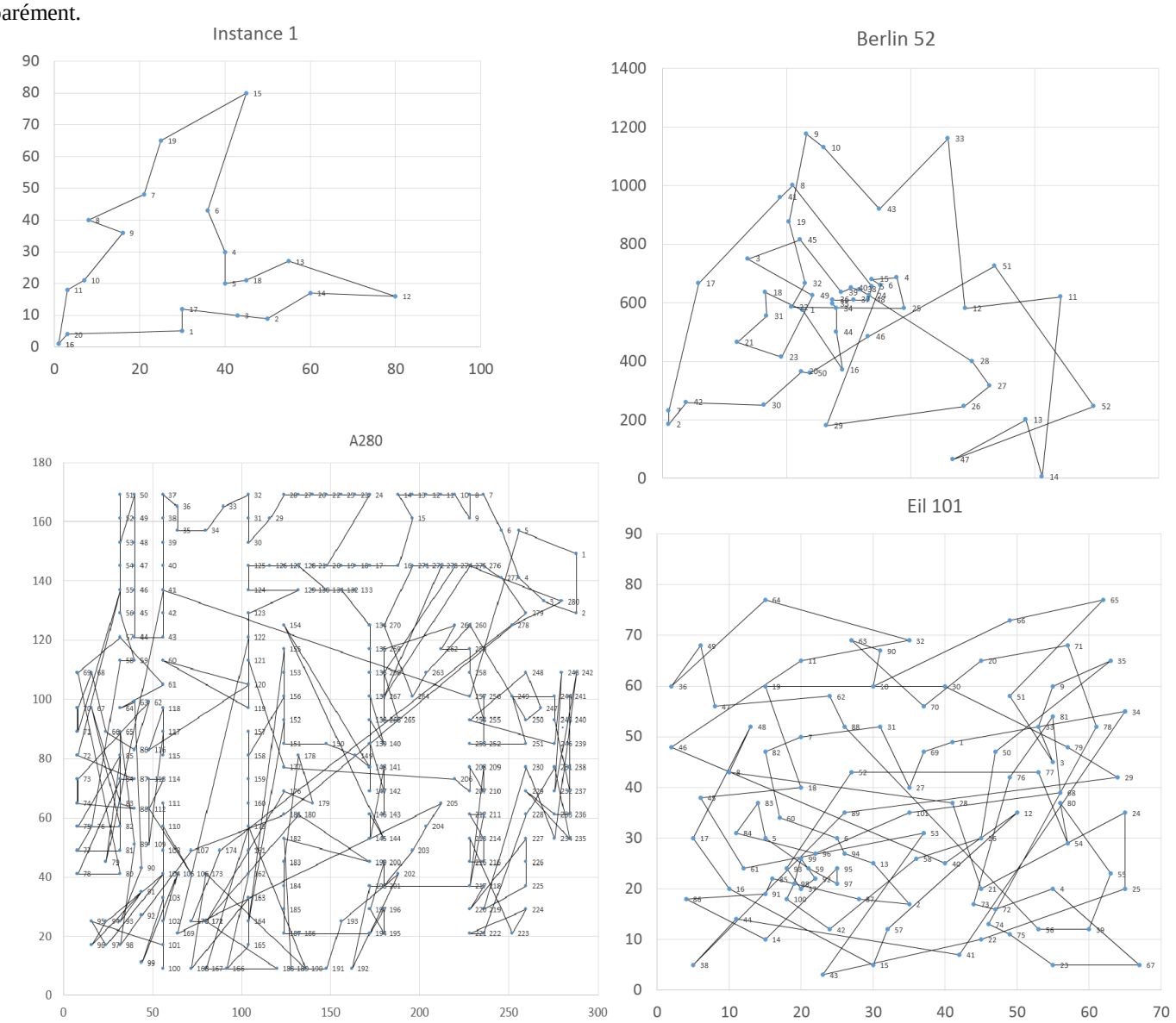

Figure 5 : Chemins obtenus pour les meilleures solutions trouvées

Références

[1] Aardal, K., Hoesel, S. v., Lenstra, J. K. and Stougie, L. (1997). A Decade of Combinatorial Optimization. Department of Information and Computing Sciences, Utrecht University, UU-CS-1997-12,

[2] Festa, P. and Resende, M. G. C. (2008). Hybrid Grasp Heuristics. AT&T Labs Research, Florham Park, July

[3] Sabry Ahmed Haroun, Benhra Jamal and El Hassani Hicham. Article: A Performance Comparison of GA and ACO Applied to TSP. International Journal of Computer Applications 117(20):28-35, May 2015

[4] A. H. Sabry, A. Bacha, and J. Benhra, "A contribution to solving the traveling salesman problem using ant colony optimization and web mapping platforms Application to logistics in a urban context," in Codit'14, Metz, France, 2014.

[5] Kennedy, J. and Eberhart, R. (1995). Particle Swarm Optimization, Proceedings of IEEE International Conference on Neural Networks, pp. 1942-1948, 27th November – 1 December, 1995

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

[7]Kennedy, J. & Eberhart, R.C. (1997) A discrete binary version of the particle swarm algorithm, Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics, Vol. 5, No. 2, pp. 4104 - 4109, ISBN: 0780340531, Orlando, Florida, October 1997, IEEE

[8] Shi XH, Liang YC, Lee HP, Lu C, Wang QX, "Particle swarm optimization-based algorithms for TSP and generalized TSP", Inf Process Lett 103(5), 2007, pp.169–176.

[9] H. Yoshida, Y. Fukuyama, S. Takayama, and Y. Nakanishi, "A particle swarm optimization for reactive power and voltage control in electric power systems considering voltage security assessment," in Proc. IEEE Int. Conf. Syst., Man, Cybern., 1999, vol. 6, pp. 497–502.

[10] A. I. El-Gallad, M. El-Hawary, A. A. Sallam, and A. Kalas, "Swarm intelligence for hybrid cost dispatch problem," in Proc. Canadian Conf. Elect. Comput. Eng., 2001, vol. 2, pp. 753–757

[11] Van der Merwe DW, Engelhrecht AP. Data Clustering using Particle Swarm Optimization. The 2003 Congress on Evolutionary Computation, 2003, p 215 -220.

[12] Q. Zhu, L. Qian, Y. Li, and S. Zhu, "An improved particle swarm optimization algorithm for vehicle routing problem with time windows," 2006, pp. 1386-1390.

[13] L. Nan, "Research on location of remanufacturing factory based on particle swarm optimization," in Proc. 2011 International Conference on Management Science and Industrial Engineering (MSIE), 2011, pp. 1016 -1019.

[14] M. B. Abdelhalim, A. E. Salama and S. E.-D.Habib, "Hardware Software Partitioning using Particle Swarm Optimization Technique", The 6th International Workshop on System on Chip for Real Time Applications, pp. 186 -194, 2006.

[15] Băutu, A. & Băutu, E. (2007). Energy Minimization Of Point Charges On A Sphere With Particle Swarms, 7th International Balkan Workshop on Applied Physics, Constantza, Jul. 2007

[16] Chiang, T. C., Chang, P. Y., & Huang, Y. M. (2006). Multi-processor tasks with resource and timing constraints using particle swarm optimization. International Journal of Computer Science and Network Security, 6 (4), 71–77

[17] Tu, K., Hao, Z., & Chen, M. (2006). PSO with improved strategy and topology for job shop scheduling. Lecture Notes in Computer Science, 4222, 146–155.

[18] T. Hendtlass. Preserving Diversity in Particle Swarm Optimization. Lecture Notes in Computer Science, vol. 2718: 4104–4108, Springer, 2003

[19] Clerc, M, "Discrete particle swarm optimization, illustrated by the traveling salesman problem", In: Studies in Fuzziness and Soft Computing New optimization techniques in engineering, Babu, B.V. & Onwubolu, G.C. (Eds.), Vol. 141, 2004, pp.219-239.

[20] Wang KP, Huang L, Zhou CG, Pang W, "Particle swarm optimization for traveling salesman problem", In: International conference on machine learning and cybernetics, 2003, pp.1583–1585.

[21] Pang, W.; Wang, K.; Zhou, C.; Dong, L.; Liu, M.; Zhang, H. & Wang, J, "Modified particle swarm optimization based on space transformation for solving traveling salesman problem", In: Proceedings of the Third International Conference on Machine Learning and Cybernetics, Shangai, China, August 2004, pp.2342-2346.

[22] Pang, W.; Wang, K.; Zhou, C. & Dong, L, "Fuzzy discrete particle swarm optimization for solving traveling salesman problem", In: Proceedings of the Fourth International Conference on Computer and Information Technology, Wuhan, China, September 2004, pp.796-800.

[23] Kemighan, L. S. (1973). "An Effective Heuristic Algorithm for the Traveling Salesman Problem." Opé rations Research 21: 2245-2269

[24] Laporte, G. (1992). "The Vehicle Routing Problem: An Overview of Exact and Approximate Algorithms." European Joumal of Operational Research 59: 345-358.