Amélioration de l'algorithme de chauve-souris par modification de règles d'évolution et introduction de mécanisme de croisement

Enhancing Bat Algorithm by Modifying Evolution Rules and Introducing Crossover Mechanism

Amina Bedboudi[1*] et Mohamed Cherif Bouras[2]

[1] *Laboratoire de recherché en informatique LRI, Université Badji Mokhtar-Annaba, BP 12, Annaba 23000, Algérie.*

[2]*Département de Mathématique Université Badji Mokhtar-Annaba, BP 12, Annaba, 23000, Algérie.*

Soumis le : 26/09/2017 *Révisé le : 09/02/2018* *Accepté le : 25/02/2018*

ملخص

خوارزمية الخفافيش هي واحدة من الوراثة الفوقية الواعدة المقترحة مؤخرا لحل مشاكل التحسين. لأنه يقوم على محاكاة سلوك تحديد الموقع بالصدفة من الخفافيش. في هذا العمل، نقدم تحسين هذه الخوارزمية من خلال تعديل مناسب لقواعد التطور وإدخال آلية كروس. وتبين لنا أن هذه الخوارزمية الجديدة تحسن النتائج التي حققتها خوارزمية الخفافيش الأصلي والخوارزميات المنشورة مؤخرا التي تعدلها. تطبيقه على خمس وظائف رياضية تستخدم على نطاق واسع يدل بوضوح وبدقة على أن النتائج التي تحققت هي أفضل. .

الكلمات المفتاحية: الاستدلالية الفوقية.خوارزمية*الخفافيش*.، مشاكل التحسين، *تبين* الهدف.

Résumé

L'algorithme de chauve-souris est l'une des méta-heuristiques prometteuses, récemment proposé pour la résolution de problèmes d'optimisation. Il se base sur la simulation du comportement d'écholocation des chauves-souris. Dans cet article, nous présentons une amélioration de cet algorithme par une modification appropriée des règles d'évolution et l'introduction de mécanisme de croisement. Nous montrons que ce nouvel algorithme améliore les résultats réalisés par l'algorithme de chauve-souris standard et les autres algorithmes le modifiant, publiés récemment. Son application sur cinq fonctions benchmarks largement utilisées dans la littérature, montre de façon claire et précise que les résultats réalisés sont nettement meilleurs.

Mots clés : Méta-heuristiques, algorithme de chauve-souris, optimisation, fonction objectif.

Abstract

The bat algorithm is one of the promising meta-heuristics recently proposed for solving optimization problems. It is based on the simulation of the echolocation behavior of bats. In this paper, we present an improvement of this algorithm by an appropriate modification of the rules of evolution and the introduction of a crossover mechanism. We show that this new algorithm improves the results achieved by the standard bat algorithm and the recently published algorithms that modify it. Its application on five benchmark functions widely used in the literature clearly and precisely shows that the achieved results are much better.

Key-words: Meta-Heuristics; Bat Algorithm, Optimization Problems, Objective Function.

[*] *Auteur correspondant : bedboudi.amina@hotmail.fr*

1. INTRODUCTION

Ces dernières années, de nouvelles approches de résolution de problèmes d'optimisation ont été développées, permettant d'obtenir des avantages par rapport aux techniques plus traditionnelles. La nécessité d'obtenir de nouvelles techniques d'optimisation découle du fait que les techniques traditionnelles telles que les approches de programmation mathématique sont devenues inefficaces pour résoudre les applications de la vie courante [13].

La complexité des problèmes d'optimisation à résoudre pour des systèmes de plus en plus complexes, les nombreuses variables de conception ainsi que leurs exigences pratiques rendent ces problèmes difficiles à gérer en utilisant des méthodes d'optimisation traditionnelles [1].

Récemment, les techniques méta-heuristiques sont utilisées pour obtenir une solution robuste pour résoudre des problèmes complexes. Ces algorithmes ont tendance à produire des solutions différentes même lorsque leurs conditions initiales restent constantes à chaque exécution en raison de leur nature aléatoire. Ils sont préférés pour ces fonctions qui ont plusieurs optimaux locaux, car ils peuvent échapper facilement aux minimums locaux en dépit de leur lenteur de convergence [2].

L'idée fondamentale derrière ces techniques est de simuler des systèmes biologiques et physiques dans la nature, tels que l'évolution naturelle, le système immunitaire, l'intelligence d'essaim, le processus de recuit, etc., dans un algorithme de traitement numérique. Ces algorithmes diffèrent dans la façon avec laquelle ils évoluent dans l'espace de recherche, en se basant sur une stratégie associée, inspirée de la nature [3].

L'une des méthodes méta-heuristiques les plus prometteuses est l'algorithme de chauve-souris qui se base sur la simulation du comportement d'écholocation des chauves-souris [4]. La capacité de l'écholocation est fascinante car ces chauves-souris peuvent trouver leurs proies et discriminer différents types d'insectes, même dans l'obscurité complète. Ils y parviennent en émettant des signaux sonores vers l'environnement et en écoutant les échos qui leur renvoient. Ils peuvent identifier la localisation d'autres objets et mesurer instinctivement la distance à laquelle ils sont éloignés en retardant le retour du son. Dans la littérature, des travaux de recherche ont proposé un ensemble de modifications ou d'hybridations sur l'algorithme de chauve-souris dans le but d'améliorer ses performances.

Iztok F. a fait une hybridation entre les stratégies d'évolution différentielle et l'algorithme de chauve-souris standard aboutissant à un algorithme de chauve-souris hybride. Il a montré que cet algorithme améliore significativement la version originale de cet algorithme [5].

Yilmaz S. a amélioré les mécanismes d'exploration et d'exploitation de l'algorithme de chauve-souris par trois modifications, la première a analysé la structure de la vitesse avec le processus de mise à jour pondérée par l'inertie. Le second prend en compte la différence entre la solution courante et la meilleure solution globale pour obtenir la plus proche et la plus grande dimension de la solution. La troisième modification consiste à faire l'hybridation entre la colonie d'abeilles artificielles et l'algorithme de chauve-souris pour améliorer la capacité d'exploration de l'algorithme de chauve-souris [6].

Chen Z. a supprimé le paramètre de vitesse et a ajouté le poids d'inertie de l'emplacement dans l'algorithme de chauve-souris qui est déterminé à l'aide de la distribution normale, puis la fréquence des impulsions émises ajoute au changement de position aléatoire et à la position optimale des chauve-souris [7].

Amir H. a introduit le concept du chaos dans l'algorithme de chauve-souris pour accroître la mobilité de recherche globale pour une optimisation globale robuste. Cette méthode utilise des cartes chaotiques pour remplacer les variables aléatoires, améliorant l'efficacité de la recherche de l'algorithme de chauve-souris standard [8].

Yilmaz S. a amélioré le mécanisme d'exploration en égalisant le volume et le taux d'émission d'impulsions à la dimension du problème en les attribuant à chaque dimension de la solution séparément, ce qui permet de réaliser simultanément différentes capacités d'exploration et d'exploitation [9].

Wasi M. a présenté un algorithme de chauve-souris auto-adaptatif et amélioré pour le problème de l'optimisation numérique globale dans les domaines continus. Il a introduit deux équations de recherche de solution améliorées. Il a également utilisé une probabilité de sélection pour contrôler la

fréquence d'emploi qui conduit à un nouveau mécanisme de recherche auto-adaptable pour l'algorithme de chauve-souris [10].

Ali A. a accéléré le processus de recherche en invoquant la méthode Nelder-Mead comme méthode de recherche locale afin d'affiner la solution la mieux obtenue à chaque itération [11].

En plus de ces modifications, des chercheurs ont étendu l'algorithme de chauve-souris afin de pouvoir gérer un ensemble d'applications de nature à obtenir de meilleurs résultats, tels que les problèmes multi-objectifs, les problèmes de clustering des données, et d'autres applications d'optimisation [12] [13] [14] [15].

Dans cet article, nous présentons une amélioration de l'algorithme de chauve-souris standard, modifiant certaines règles d'évolution et en introduisant la règle génétique de croisement. Ces modifications visent à améliorer la qualité des résultats, la consommation en temps d'exécution, ainsi que les caractéristiques d'exploration et de convergence de l'algorithme de chauve-souris.

La structure de cet article est la suivante. Dans la section II, l'algorithme standard de chauve-souris est introduit. Notre algorithme modifié est présenté dans la section III. La section IV illustre les expérimentations et la comparaison des résultats. À la fin de cet article, une conclusion et des perspectives sont présentée à la section V.

2. L'ALGORITHME DE CHAUVE-SOURIS STANDARD

Parmi plusieurs méta-heuristiques biologiquement inspirées, l'algorithme des chauves souris (Bat Algorithm: BA) proposé par Xin She Yang [4] ont attiré l'attention des chercheurs qui travaillent dans le domaine des algorithmes d'optimisation à inspiration biologique.

Cet algorithme est basé sur le comportement de l'écholocation de microchiroptères. L'écholocation est un sonar biologique (à cause de manque de vision) qui permet détecter la distance, et ils ont aussi la capacité de faire la différence entre la nourriture/proie et les obstacles. Il s'appuye sur une technique de régulation de fréquence pour augmenter la diversité des solutions dans la population, et par là-même, il tente d'équilibrer l'exploration et l'exploitation pendant le processus de recherche, en imitant les variations des taux d'émission d'impulsions et la vitesse des chauves-souris lors de la recherche de proies [12].

L'algorithme standard de chauve-souris présente de nombreux avantages; L'un d'entre eux est qu'il peut obtenir une convergence rapide aux étapes initiales en passant de l'exploration à l'exploitation. Cela en fait un algorithme efficace lorsqu'une solution rapide est nécessaire. Afin d'améliorer les performances, de nombreuses modifications ont été ajoutées pour augmenter la diversité de la solution et pour améliorer les performances de l'algorithme de chauve-souris standard comme mentionné précédemment [12] [14].

A. Initialisation de l'Algorithme de chauve-souris

La population initiale est générée de façon aléatoire pour n nombre de chauve-souris. Chaque individu de la population est décrit par un vecteur à valeurs réelles avec une dimension d. L'équation suivante est utilisée pour générer la population initiale.

$$\mathbf{X_{ij} = X_{min\,j} + rand(0,1)(X_{max\,j} - X_{min\,j})} \qquad \mathbf{(1)}$$

où i = 1,2, ..., n; j = 1,2, ..., d; Xmaxj et Xminj sont les limites supérieures et inférieures pour la dimension j.

B. Solution, fréquence et vitesse

Dans les simulations, nous utilisons naturellement des chauves-souris virtuelles. Nous devons définir les règles de mise à jour de leurs positions x_i et les vitesses v_i , dans un espace de recherche bidimensionnel, à chaque itération t. Parmi toutes les solutions, il existe une meilleure solution courante x*. Les règles précédentes peuvent être traduites pour obtenir les nouvelles solutions x_i^t et vitesses v_i^t à l'étape t, par application des équations suivantes de mise à jour.

$$\mathbf{f_i = f_{min} + (f_{max} - f_{min})\beta,} \qquad \mathbf{(2)}$$

$$\mathbf{V_i^t = V_i^{t-1} + (X_i^t - X^*)f_i,} \qquad \mathbf{(3)}$$

$$\mathbf{X_i^t = X_i^{t-1} + V_i^t} \qquad \mathbf{(4)}$$

Où $\beta \in [0,1]$ est un vecteur aléatoire issu d'une distribution uniforme. X* est la meilleure solution globale courante qui est déterminée en comparant toutes les solutions parmi tous les n chauves-souris. Alors que $\lambda_i f_i$ est l'augmentation de vitesse, nous utilisons f_i pour régler la vitesse tout en fixant l'autre facteur λ_i. La plage de valeurs de f_i diffère d'un problème à l'autre en fonction du domaine, de la taille du problème, etc. Initialement, chaque chauve-souris reçoit de manière aléatoire une fréquence qui est dérivée uniformément de $[f_{min}, f_{max}]$. Lorsqu'une solution est sélectionnée parmi les meilleures solutions courantes, une nouvelle solution pour chaque chauve-souris est générée localement à l'aide d'une transformation intégrant un facteur aléatoire.

$$\mathbf{X_{new} = X_{old} + \varepsilon A^t}, \qquad \textbf{(5)}$$

Où $\varepsilon \in [-1,1]$ est un nombre aléatoire, alors que A_t est le volume moyen de toutes les chauves-souris à cette étape de traitement. La mise à jour des vitesses et des positions des chauves-souris est similaire à la procédure d'optimisation standard des essaims de particules [14] [15]. Étant donné que le contrôle de la portée du mouvement des particules envahissantes, l'algorithme de chauve-souris peut être considéré comme étant une combinaison équilibrée de l'optimisation standard des essaims de particules et de la recherche locale intensive, contrôlée par le volume et le taux de pulsation.

C. Mise à jour du volume et du taux de pulsation

Le volume A_i et le taux de pulsation r*i* doivent être mis à jour à chaque itération. Au fur et à mesure que l'intensité diminue une fois que la chauve-souris a trouvé sa proie, alors que le taux de pulsation ri augmente, le volume peut être choisi comme valeur de commodité. Lorsque le volume atteint le minimum A_{min}, cela signifie que la chauve-souris a trouvé la proie et a cessé d'émettre un son. Les équations suivantes montrent comment le volume A_i et le rythme r sont mis à jour pendant les itérations.

$$\mathbf{A_i^{t+1} = \alpha A_i^t}, \qquad \textbf{(6)}$$

$$\mathbf{r^{t+1} = r_i^0[1 - \exp(-\gamma t)]}, \qquad \textbf{(7)}$$

Où α et γ sont des constantes. L'α constant est similaire au facteur de refroidissement dans le recuit simulé. Pour tout $0 < \alpha < 1$ et $\gamma > 0$ tel que

$$\mathbf{A_i^t \to 0, r_i^t \to r_i^0, as\ t \to \infty} \qquad \textbf{(8)}$$

Le choix des paramètres nécessite une certaine expérimentation. Chaque chauve-souris devrait avoir différentes valeurs de sonorité et taux d'émission d'impulsion. Et cela peut être réalisé par randomisation. Le volume et les taux d'émission ne sont mis à jour que si les nouvelles solutions sont améliorées, ce qui signifie que ces chauves-souris se déplacent vers la solution optimale.

Algorithme 1. Pseudo code de l'algorithme de chauve-souris standard.

1. Fonction objective: f(x), x = (x_1, ..., x_d)
2. Initialiser la population de chauve-souris x_i et la vitesse v_i, i = 1,2, ..., n
3. Définir la fréquence d'impulsions f_i de chaque position x_i
4. Initialiser le taux de pulsation r_i et le volume A_i
5. Tant que (t <nombre maximum d'itérations)

5.1. Générer de nouvelles solutions en ajustant la fréquence et en actualisant les vitesses et les positions / solutions. (équations 2, 3; 4).

5.2. Si (rand> r_i)

5.2.1. Sélectionnez une solution parmi les meilleures solutions

5.2.2. Générer une solution locale autour de la meilleure solution sélectionnée x*
(équation 5)

5.3. Fin si

5.4. Si (rand <A_i et $f(x_i) < f(x^*)$)

5.4.1. Accepter de nouvelles solutions

5.4.2. Augmenter ri et réduire Ai (équations 6, 7)

5.5. Fin si

5.6. Trouver la meilleure solution x*
6. Fin tant que
7. Afficher les résultats donnés par la meilleure solution x*
8. Fin de l'algorithme

3. ALGORITHME PROPOSÉ

Dans cette section nous développons notre proposition de modification de l'algorithme de chauve-souris standard (Bat algorithme), que nous appelons BATA (Bat Accéléré). En effet, des études montrent que les chauves-souris utilisent le retard de l'émission et la détection de l'écho, la différence de temps entre leurs deux oreilles et les variations de volume des échos pour construire un scénario tridimensionnel de l'environnement. Ils peuvent détecter la distance et l'orientation de la cible, le type de proie et même la vitesse de déplacement de la proie, telle que celle des petits insectes.

Pour simplifier, on suppose que $f \in [0, f_{max}]$. On sait que les fréquences plus élevées ont des longueurs d'ondes courtes et parcourent une distance plus courte. Pour les chauves-souris, les plages typiques sont à quelques mètres. La fréquence est donc un facteur efficace important qui peut affecter la recherche globale de l'algorithme. Afin d'améliorer l'algorithme de chauve-souris, une modification est apportée pour améliorer les capacités d'exploration. La nouvelle étape de chaque chauve-souris est contrôlée par un vecteur de position, la meilleure position globale et la fréquence. Dans le but de réduire le temps d'exécution de la recherche et étant donné que l'équation de la vitesse est doublement intégrée, nous proposons de la supprimer des règles (équation 3). Cette modification accélère la convergence de l'algorithme global. L'équation 3 est annulée et l'équation de la nouvelle étape (équation 4) est modifiée comme suit:

$$\mathbf{x_i^t = f_i x_i^{t-1} + (1 - f_i) r_i x^*} \qquad \textbf{(9)}$$

Ainsi, l'équation (9) est modifiée de façon à intégrer un effet multiplicatif de la solution courante par la fréquence au premier terme et un deuxième effet multiplicatif de la meilleure solution globale au deuxième terme.

De plus, si la nouvelle solution obtenue x_i^t n'est pas meilleure que la solution x_i^{t-1}, un croisement est effectué entre les deux solutions. Si le résultat obtenu de ce croisement ($S^{(t)}$) est meilleur que x_i^{t-1}, il la remplacera. La règle de croisement applique les équations (10) et (11) et sélectionne la meilleure des deux qui va devenir la nouvelle solution à intégrer à la population.

$$\mathbf{S^{(t1)} = x_i^t + \alpha x_i^{t-1}} \qquad \textbf{(10)}$$

$$\mathbf{S^{(t2)} = x_i^t + (1 - \alpha) x_i^{t-1}} \qquad \textbf{(11)}$$

Où $\alpha \in [0, 1]$.

L'effet de cette modification est illustré par les résultats de test sur un ensemble de benchmarks présentés dans la section suivante.

4. RÉSULTATS EXPÉRIMENTAUX

A. Fonctions Benchmarks

Afin de vérifier l'efficacité de modification proposée, notre algorithme est testé en utilisant quatre fonctions benchmark de minimisation avec la valeur zéro comme solution optimale à différentes dimensions comme indiqué dans le tableau I. Les valeurs de "meilleure, mauvaise, et moyenne, ainsi que l'écart-type" sont montrées dans le tableau II.

Tableau I Fonctions Benchmark de Test

N°	nom de la fonction	dimension	plage	fonction
1	Sphère	10, 50	[-5.12,5.12]	$\mathbf{f_1(x) = \sum_{i=1}^{n} x_i^2}$
2	Griewangk	10, 50	[-600,600]	$\mathbf{f_2(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos(\frac{x_i}{\sqrt{i}}) + 1}$
3	Ackley Function	10, 50	[-100,100]	$\mathbf{f_3(x) = -20\exp(-0.2\sqrt{(\frac{1}{n}\sum_{i=1}^{n} x_i^2) - \exp(\frac{1}{n}\sum_{i=1}^{N} \cos(2\pi X_i))}}$
4	Rastrigin	10, 50	[-15,15]	$\mathbf{f_4(x) = \sum_{i=1}^{d} [x_i^2 - 10\cos(2\pi x_i) + 10]}$
5	Rosenbrock	10, 50	[-15,15]	$\mathbf{f_5(x) = \sum_{i-1}^{d-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2}$

B. Paramètres

Notre algorithme, appelé BATA (BAT Accéléré), l'algorithme chauve-souris standard (BA) et les autres modifications les plus notables sont testés avec 25 exécutions indépendantes pour chaque fonction de test. La taille de la population (nombre de chauves-souris) est fixée à 30. Le nombre de générations est fixé à 1000, et 2000 pour la dimension = 10 et 50 respectivement pour chaque exécution. La valeur de la fréquence minimale est fixée à 0 alors que la valeur maximale est fixée à 2. La mise en œuvre s'effectue à l'aide du langage Java sur un PC à processeur Intel® I3, CPU 3 GHz et RAM 4Go.

C. Comparaison

En comparant les résultats avec BA et les modifications précédentes comme Hybrid Bat Algorithm (HBA) [5], Modified Bat Algorithm (MBA) [9], Novel Adaptive Bat Algorithm (NABA) et "Bat Algorithm with Auto-Adaptive Mutation" (BA-SAM) [10], nous montrons que notre algorithme est meilleure en terme de fonction objectif(fitness) et déviation standard (qualité du résultat), tel qu'illustré par le tableau II.

Tableau II BA, HBA, MBA, NABA, BA-SAM et BAT appliqués sur les 4 fonctions Benchmark.

Fonction	nom de la fonction	Dimension	algorithme	La meilleure fitness	+ mauvaise fitness	Déviation standard
f1	Sphère	10	BA	1.42	1.03e+0.1	2.11
			HBA	4.31e-09	2.24e-03	1.41e-07
			MBA	3.73e -03	1.42e -02	3.31e- 03
			NABA	1.42e-04	1.32	2.81e-01
			BA-SAM	1.74e-06	3.41e-01	1.15e-01
			BATA	**2.44e-13**	**5.40e-11**	**2.42e-11**
		50	BA	7.17e+02	1.27e+03	9.42e+01
			HBA	4.17e-09	2.89e-03	1.46e-07
			MBA	1.87e+02	5.51e+02	1.14e+02
			NABA	2.24e+02	5.20e+02	8.21e+01
			BA-SAM	2.01e+02	5.49e+02	8.77e+01
			BATA	**0**	**8.49e-07**	**3.10e-07**
f2	Griewangk	10	BA	5.81	4.15e+01	9.19
			HBA	2.25e-09	3.97e-05	1.14e-07
			MBA	2.45	2.46e+01	5.19

			NABA	2.68	5.16e+01	1.21e+01
			BA-SAM	1.31e-01	1.14e+01	2.91
			BATA	**0**	**1.24e-12**	**4.43e-13**
		50	BA	7.15e+02	1.11e+03	9.91e+01
			HBA	4.80e-09	2.82e-03	1.61e-07
			MBA	1.84e+02	5.51e+02	1.21e+02
			NABA	2.26e+02	5.21e+02	8.27e+01
			BA-SAM	2.22e+02	5.19e+02	8.17e+01
			BATA	**0**	**8.19e-07**	**3.80e-07**
f3	Ackley	10	BA	1.17e+01	1.68e+01	1.40
			HBA	6.31e-04	2.00e+01	1.78e+01
			MBA	3.21e-02	1.29	3.68e-01
			NABA	1.21e-01	1.25e+01	4.26
			BA-SAM	2.09e-02	6.44	1.54
			BATA	**7.46e-12**	**1.46e-08**	**4.96e-09**
		50	BA	7.05e+02	1.13e+03	9.97e+01
			HBA	4.83e-09	2.89e-03	1.66e-07
			MBA	1.85e+02	5.58e+02	1.21e+02
			NABA	2.29e+02	5.20e+02	8.47e+01
			BA-SAM	2.02e+02	5.49e+02	8.47e+01
			BATA	**0**	**8.49e-07**	**3.10e-07**
F3	Rastrigi	10	BA	8.59e+01	1.94e+02	2.53e+01
			HBA	5.12	2.38e+01	1.69e+01
			MBA	1.46e+01	3.48e+01	4.35
			NABA	1.34e+01	5.08e+01	9.33
			BA-SAM	6.73	3.83e+01	8.85
			BATA	**1.98e-11**	**3.03e-08**	**1.09e-08**
		50	BA	7.05e+02	1.13e+03	9.97e+01
			HBA	4.83e-09	2.89e-03	1.66e-07
			MBA	1.85e+02	5.58e+02	1.21e+02
			NABA	2.29e+02	5.20e+02	8.47e+01
			BA-SAM	2.02e+02	5.49e+02	8.47e+01
			BATA	**0**	**8.49e-07**	**3.10e-07**
F4	Rosenbrock	10	BA	1.50e+03	1.22e+05	3.66e+04
			HBA	6.34e-02	5.10e+02	7.73
			MBA	4.83e-09	2.89e-03	1.66e-07
			NABA	5.57e-01	4.19e+04	1.02e+04
			BA-SAM	2.72	4.70e+02	8.75e+01
			BATA	**2.63**	**8.03**	**1.86**
		50	BA	1.96e+05	2.06e+06	4.48e+05
			HBA	4.83e-09	2.89e-03	1.66e-07
			MBA	4.23e-09	2.85e-03	1.26e-07
			NABA	2.11e+02	4.76e+05	1.18e+05
			BA-SAM	3.14e+02	1.08e+06	1.96e+05
			BATA	**1.89e+00**	**4.39e+01**	**1.23e+01**

Le tableau II illustre l'application de notre algorithme BATA sur les quatre fonctions benchmark. Il montre qu'il réalise de meilleures résultats et ce, pour les différentes dimensions utilisées.

V. CONCLUSION ET PERSPECTIVES

Dans cet article, nous avons introduit une amélioration de l'algorithme de chauve-souris, modifiant certaines règles d'évolution et introduisant la règle génétique de croisement. Ces modifications ont induit moins de temps d'exécution et les caractéristiques d'exploration et de convergence de l'algorithme proposé améliorent la convergence de l'algorithme et la qualité des résultats.

Nous avons mis en œuvre, testé et évalué notre approche en l'appliquant sur un certain nombre de fonctions benchmark de référence, portant sur l'optimisation numérique. Les résultats obtenus (qualité finale de la solution et convergence) démontrent clairement que l'algorithme proposé est meilleure que non seulement l'algorithme de chauve-souris standard, mais aussi que d'autres algorithmes modifiant l'algorithme standard et présentés dans la littérature. Cette amélioration est démontrée à travers l'utilisation de fonctions benchmarks bien connues dans la littérature spécialisée. Comme perspectives de continuité de ce travail, nous comptons appliquer cet algorithme sur des problèmes de data mining et sur des applications temps réel où le respect des contraintes de temps est primordial.

REFERENCES

[1] S. S. Rao, *Engineering Optimization*, vol. 56, no. 772801201. 2009.

[2] X. Yang, *Firefly Algorithm*. 2010.

[3] A. L. Parrill, "Introduction to Evolutionary Algorithms," *Evolutionary Algorithms in Molecular Design*, vol. 8, pp. 1–13, 2008.

[4] X.-S. Yang, "A New Metaheuristic Bat-Inspired Algorithm," *Nat. Inspired Coop. Strateg. Optim. (NICSO 2010)*, pp. 65–74, 2010.

[5] I. Fister, D. Fister, and X. S. Yang, "A hybrid bat algorithm," *Elektroteh. Vestnik/Electrotechnical Rev.*, vol. 80, no. 1–2, pp. 1–7, 2013.

[6] S. Yilmaz and E. U. Kucuksille, "Improved Bat Algorithm (IBA) on Continuous Optimization Problems," *Lect. Notes Softw. Eng.*, vol. 1, no. 3, pp. 279–283, 2013.

[7] C. Zhen, Z. Yongquan, and L. U. Mindi, "A simplified Adaptive Bat Algorithm Based on Frequency," *J. Comput. Inf. Syst.*, vol. 9: 16 (201, pp. 6451–6458, 2013.

[8] A. H. Gandomi and X. S. Yang, "Chaotic bat algorithm," *J. Comput. Sci.*, vol. 5, no. 2, pp. 224–232, 2014. [9] S. Yilmaz, E. U. Kucuksille, and Y. Cengiz, "Modified bat algorithm," *Elektron. ir Elektrotechnika*, vol. 20, no. 2, pp. 71–*78, 2014.*

[10] M. Wasi, U. Kabir, and M. S. Alam, "Bat Algorithm with Self-adaptive Mutation: A Comparative Study on Numerical Optimization Problems," Int. J. Comput. Appl., vol. 100, no. 10, pp. 975–8887, 2014.

[11] A. Fouad Ali, "Accelerated Bat Algorithm for Solving Integer Programming Problems.," Egypt. Comput. Sci. J., vol. 39, no. 1, p. 25, 2015.

[12] X.-S. Yang and X. He, "Bat algorithm: literature review and applications," Int. J. Bio-Inspired Comput., vol. 5, no. 3, pp. 141–149, 2013.

[13] Hegazy Zaher et al., An Improved Approach for Bat Algorithm, International Journal of Advanced Research in Computer Science and Software Engineering 7(2), February - 2017, pp. 134-140

[14] S. Induja, V.P. Eswaramurthy, Bat Algorithm: An Overview and its Applications, International Journal of Advanced Research in Computer and Communication Engineering Vol. 5, Issue 1, January 2016

[15] Ramesh, B., Mohan, V. C. J., Reddy, V. C. V., (2013). Application of bat algorithm for combined economic load and emission dispatch, Int. J. of Electrical Engineering and Telecommunications, Vol. 2, No. 1, pp. 1–9.