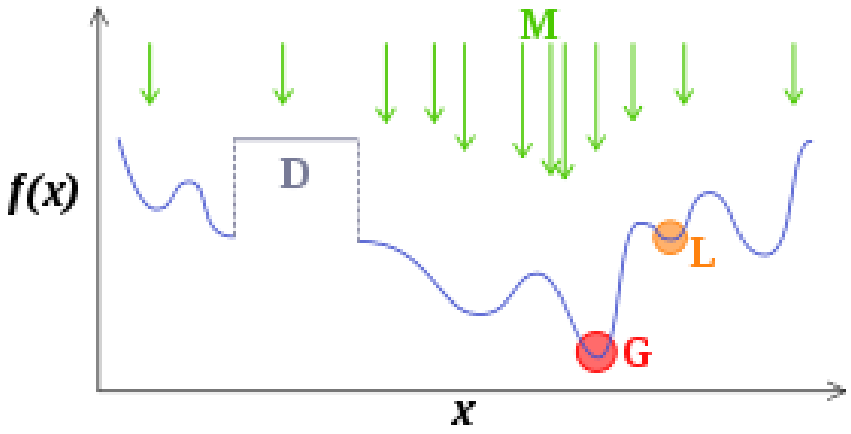


Rapport du Projet Master 1 MIAGE

Sujet : Analyse de l'algorithme Backtracking Search Optimization

Réalisé par :

Mohamed El Rahali

Mohamed Amine Fathalah

Youness Kerkar

Encadré par :

M. Lhassan Idoumghar

Table des matières

Abstraction

L'optimisation est un élément essentiel pour l'objectif de la recherche dans les domaines des mathématiques appliquées et en informatique. Les algorithmes d'optimisation visent principalement à obtenir l'optimum global pour les problèmes d'optimisation. Il y a beaucoup de différents types de problèmes d'optimisation dans le monde réel. Lorsqu'un problème d'optimisation a un gradient d'informations simples et explicites ou nécessite des budgets moins d'évaluations de la fonction, il permit la mise en œuvre de techniques d'optimisation classiques tels que la programmation mathématique souvent pourrait atteindre des résultats efficaces. Cependant, beaucoup d'ingénierie du monde réel peut avoir des problèmes d'optimisation non linéaire, complexe, qui les rendent difficiles à traiter à l'aide des techniques d'optimisation classiques. L'émergence d'algorithmes métaheuristique a surmonter les déficiences de techniques d'optimisation classiques dans une certaine mesure, car ils n'ont pas besoin de l'information et de gradient ont la capacité de s'échapper de optimums locaux. Algorithmes métaheuristique sont principalement inspirés de divers phénomènes naturels et/ou biologiques du comportement social. Parmi ces algorithmes métaheuristique, algorithmes intelligence en essaim et algorithmes évolutionnaires sont peut-être les plus intéressants. Les algorithmes de l'intelligence collective généralement simuler le comportement de l'intelligence des nuées de créatures, telles que l'optimisation de l'essaim de particules (PSO), ant colony optimization (ACO), cuckoo recherchez (CS), et la colonie d'abeilles artificielles (ABC). Ces types d'algorithmes sont généralement développés par des inspirations d'une série de comportements complexes dans les processus de coopération mutuelle avec des essaims et l'auto-organisation, dans laquelle la "coopération" est leur concept de base. Les algorithmes évolutionnaires (EAS), sont inspirés par le mécanisme de l'évolution de la nature, dans laquelle "évolution" est l'idée clé. Exemples d'EAs : l'algorithme génétique (GA), l'évolution différentielle (DE), covariance matrix évolution strategy (adaptation CMAES), et l'algorithme d'optimisation de recherche en arrière (BSO).

BSO est un processus itératif de l'évaluation environnementale fondée sur la population, qui a d'abord proposé par Civicioglu en 2013. BSO a trois opérateurs génétiques de base : sélection, mutation, et croisé. La principale différence entre la BSO et autres algorithmes similaires est que la BSO possède une mémoire pour stocker une population d'une génération précédente, choisis au hasard, qui est utilisée pour générer la matrice d'orientation de recherche pour l'itération suivante. En outre, la BSO a une structure simple, ce qui la rend efficace, rapide, et capable de résoudre les problèmes de transport multimodal. BSO n'a qu'un seul paramètre de contrôle appelé le taux de mélange (MIX-RATE), ce qui réduit considérablement la sensibilité de la valeurs initiales pour les paramètres de l'algorithme. En raison de ces caractéristiques, en moins de 5 ans, la BSO a été employé avec succès pour résoudre divers problèmes d'optimisation de l'ingénierie, tels que les systèmes d'alimentation, moteur à induction, d'antennes, le traitement des images numériques, les réseaux de neurones artificiels, et de l'énergie et de la gestion de l'environnement.

Cependant, la BSO a une faible capacité d'exploitation locale et sa vitesse de convergence est relativement lente. Ainsi, de nombreuses études ont tenté d'améliorer la performance de l'Alliance et de quelques modifications de la BSO ont été proposées pour combler les lacunes. Du point de vue de l'objet modifié, les modifications de BSO peut être divisé en quatre catégories :

- Modification de la population initiale.

- Modifications des opérateurs de reproduction, y compris la mutation et les opérateurs de croisement.

- Modification de la sélection des opérateurs, y compris la stratégie d'exploitation local.

- Modifications du facteur de contrôle et le paramètre.

La recherche sur le contrôle de paramètres de l'EAs est un des domaines les plus prometteurs de la recherche dans le calcul de l'évolution ; même une petite modification de paramètres dans un algorithme peut faire une différence considérable.Cela pourrait donner aux BSO une puissante capacité mondiale d'exploration à la première itération ; cependant, elle affecte également la capacité de l'exploitation ultérieure de la BSO.

Introduction

Optimisation des paramètres de SVM a toujours été une tâche complexe pour les chercheurs. Au cours des dernières années, de nombreux algorithmes ont été employés pour cette tâche, comme l'essai et erreur procédures, l'algorithme grille, La méthode de validation croisée, la méthode d'estimation de l'erreur de généralisation, la méthode de descente de gradient, et ainsi de suite.

Malheureusement, ces méthodes contiennent encore des inconvénients qui entravent l'efficacité de SVM. Par exemple, la méthode de la grille nécessite des calculs complexes et prend beaucoup de temps alors que la méthode de validation croisée nécessite également des calculs longs et compliqués. Les algorithmes heuristiques, tels que l'algorithme génétique (GA), l'essaim de particules PSO (optimisation), Et l'optimisation par colonie de fourmis (CAL) ont également été utilisées pour optimiser les paramètres de SVM. Cependant, PSO est facile à piéger dans les domaines d'optimisation locale alors que l'AG a un coût de calcul coûteux. Civicioglu, récemment mis au point la mandature Pinar Recherche Algorithme d'optimisation (BSO), qui est un algorithme évolutionnaire (EA) pour résoudre des problèmes d'optimisation. La méthode BSO pourrait résoudre les problèmes d'optimisation numérique d'une valeur pour un court laps de temps et le résultat était meilleur que d'autres EAs. Contrairement à d'autres méthodes dans le groupe EA, BSO n'a qu'un seul paramètre de contrôle dans l'algorithme. Ceci rend la méthode beaucoup plus simple à utiliser.

BSA, un nouvel algorithme inspiré de la nature proposée par Civicoglu, est efficace, rapide et capable de résoudre les problèmes d'optimisation numériques différentes avec une structure simple. Il a été prouvé que la BSA peut résoudre les problèmes de référence avec plus de succès que d'autres algorithmes en comparant par exemple PSO, CMAES, ABC et JDE. BSO divise le processus d'évolution en deux phases. Dans la première phase, l'algorithme proposé utilise la mutation et les opérateurs de croisement utilisés dans la norme BSA pour profiter de l'information obtenue à partir de la population précédente. Dans la deuxième phase, la mutation et les opérateurs de croisement employées dans l'évolution du différentiel standard algorithme est utilisé pour accélérer la convergence de l'algorithme et le guide pour trouver la solution optimale. En outre,

l'éleveur algorithme génétique mutation est utilisés pour améliorer la diversité de la population avec une faible probabilité dans la phase ultérieure.

Le reste de ce rapport est organisé comme suit :

La section 1 décrit la formulation de l'algorithme d'optimization BSO

La section 2 décrit la modélisation de l'algorithme et ses différentes fonctions.

La section 3 présente les caracteristiques des fonctions pour le benchmarking

La section 4 présente une comparison de BSA avec un algorithme de meme categorie.

La section 5 présente les resutats statistiques de notre algorithme, en finissant par la conclusion.

I – Backtracking Search Optimization

BSO est un algorithme de recherche adaptative qui utilise trois opérateurs génétiques de base y compris la sélection, mutation, croisés et de générer des procès de particuliers. Le principe de BSO, qui se compose de six étapes, les étapes sont présenté dans le diagramme de la Fig. 1.

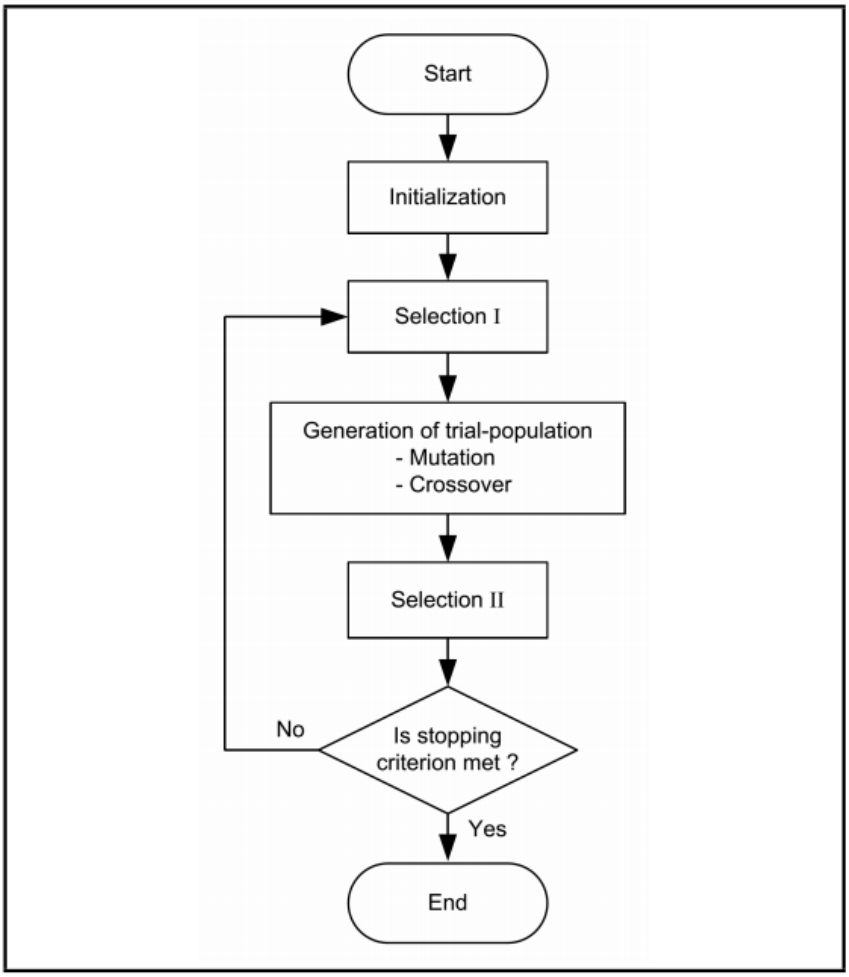


Figure 1 : l'organigramme de l'algorithme de Backtracking Search optimization

Input: ObjFun, N, D, max-iteration, mix-rate, $\text{low}_{1:D}$, $\text{up}_{1:D}$
Output: globalminimum, globalminimizer
//rand ~ $U(0,1)$, randn ~ $N(0,1)$, w = randint(·), randint ~ $U(\cdot)$ $w \in \{1,2,\ldots,\cdot\}$
(1) function BSAISA(ObjFun, D, N, max-iteration, low, up) // **Initialization**
(2) globalminimum = inf
(3) **for** i **from** 1 **to** N **do**
(4) **for** j **from** 1 **to** D **do**
(5) $P_{i,j} = \text{rand} \cdot (\text{up}_j - \text{low}_j) + \text{low}_j$ // Initialization of population P
(6) $\text{oldP}_{i,j} = \text{rand} \cdot (\text{up}_j - \text{low}_j) + \text{low}_j$ // Initialization of oldP
(7) **end**
*(8) $\text{fitness}P_i = \text{ObjFun}(P_i)$ $\text{fitnessoldP}_i = \text{ObjFun}(\text{oldP}_i)$ // Initial-fitness values of P and oldP
(9) **end**
(10) **for** *iteration* **from** 1 **to** *max-iteration* **do** // **Selection-I**
(11) **if** $(a < b \mid a, b \sim U(0,1))$ **then** $\text{oldP} \coloneqq P$ **end**
(12) $\text{oldP} \coloneqq \text{permuting}(\text{oldP})$
(13) **Generation of Trail-Population**
•(14) $\Delta I_i = \text{abs}(\text{ObjFun}(P_i) - \text{ObjFun}(\text{oldP}_i))$ // Modified F based on SA
•(15) $G = \text{iteration}$ // Modified F based on SA
•(16) $F_i \sim N\left(\exp\left(-\frac{1/\Delta I_i}{1/G}\right), 1\right)$ // Modified F based on SA
(17) $M = P + F \cdot (\text{oldP} - P)$ // **Mutation**
(18) $\text{map}_{1:N,1:D} = 1$ // Initial-map is an N-by-D matrix of ones
(19) **if** $(c < d \mid c, d \sim U(0,1))$ **then** // **Crossover**
(20) **for** i **from** 1 **to** N **do**
(21) $\text{map}_{i,u(1:\lceil \text{mix-rate-rand-}D \rceil)} = 0 \mid u = \text{permuting}(1, 2, 3, \ldots, D)$
(22) **end**
(23) **else**
(24) **for** i **from** 1 **to** N **do**, $\text{map}_{i,\text{randi}(D)} = 0$, **end**
(25) **end**
(26) $T \coloneqq M$ //Generation of Trial Population, T
(27) **for** i **from** 1 **to** N **do**
(28) **for** j **from** 1 **to** D **do**
(29) **if** $\text{map}_{i,j} = 1$ **then** $T_{i,j} \coloneqq P_{i,j}$
(30) **end**
(31) **end**
(32) **for** i **from** 1 **to** N **do**
(33) **for** j **from** 1 **to** D **do**
(34) **if** $(T_{i,j} < \text{low}_{i,j})$ **or** $(T_{i,j} > \text{up}_{i,j})$ **then**
(35) $T_{i,j} = \text{rand} \cdot (\text{up}_j - \text{low}_j) + \text{low}_j$
(36) **end**
(37) **end**
(38) **end**
(39) **end**
*(40) fitness T_i = ObjFun(T_i) // **Seletion-II**
(41) **for** i **from** 1 **to** N **do**
*(42) **if** fitness T_i < fitness P_i **then**
(43) fitness P_i = fitness T_i
(44) $P_i = T_i$
(45) **end**
(46) **end**
*(47) fitness P_{best} = min(fitness P) | best $\in \{1, 2, 3, \ldots, N\}$
*(48) **if** fitness P_{best} < globalminimum **then** // Export globalminimum and globalminimizer
(49) globalminimum $\coloneqq$ fitness P_{best}
(50) globalminimizer $\coloneqq P_{\text{best}}$
(51) **end**
(52) **end**

Figure 2 : Pseudo Code de L'algorithme de BSO.

II – Modélisation de l'algorithme

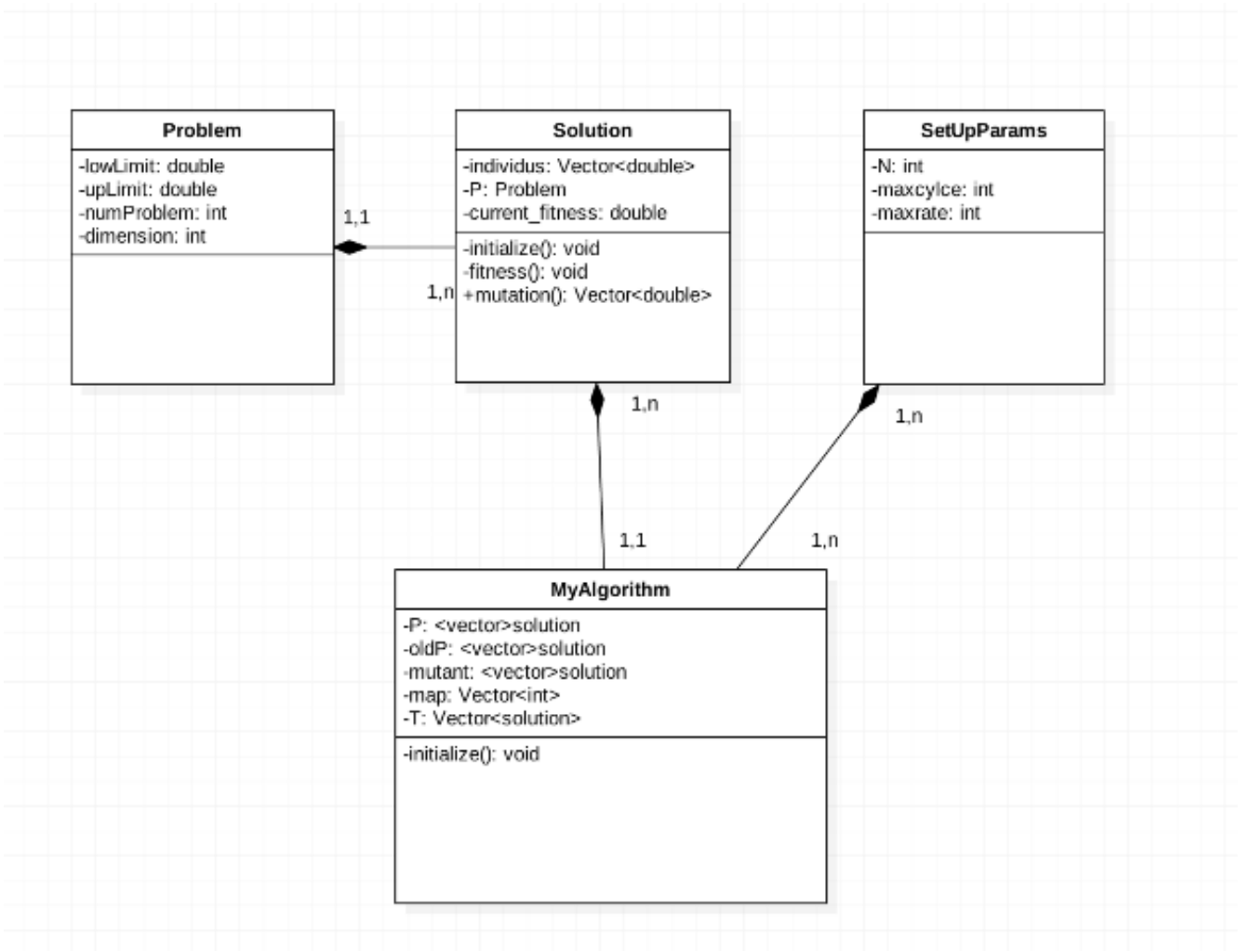


Figure 3 : Diagramme de classe correspondant à l'algorithme BSO

Public Member Functions

	SetUpParams (int Maxcycle, Double Mixrate, int Dim, int Popsize)
int	**getMaxcycle** ()
void	**setMaxcycle** (int Maxcycle)
Double	**getMixrate** ()
void	**setMixrate** (Double Mixrate)
int	**getDim** ()
void	**setDim** (int Dim)
int	**getPopsize** ()
void	**setPopsize** (int Popsize)

Figure 4 : Les fonctions de la Class SetUpParams

Public Member Functions

	Solution (Double FitnessBest, **Problem** Pbm)
Double	**getFitnessBest** ()
void	**setFitnessBest** (Double FitnessBest)
Problem	**getPbm** ()
void	**setPbm** (**Problem** Pbm)
Double	**Random** (Double Low, Double Up)

Figure 5 : Les fonctions de la Class Solution

Public Member Functions

	Problem (int NumFunc)
Double	**getLow** ()
void	**setLow** (Double Low)
Double	**getUp** ()
void	**setUp** (Double Up)
int	**getNumFunc** ()
void	**setNumFunc** (int NumFunc)

Figure 6 : Les fonctions de la Class Problem

Public Member Functions

	Algorithm (ArrayList< ArrayList< Double >> Population, ArrayList< ArrayList< Double >> OldPopulation, **Solution** sol, **SetUpParams** setup, double GlobalMin, ArrayList< Double > FitnessP)
	Algorithm (ArrayList< ArrayList< Double >> Mutant, ArrayList< ArrayList< Double >> T, ArrayList< ArrayList< Integer >> map, ArrayList< Integer > IndiceOldP)
	Algorithm (ArrayList< ArrayList< Double >> Globalminimizer, ArrayList< Double > fitnessT, int Randmut)
void	**Initialize** ()
Double	**Min** (ArrayList< Double > l)
Double	**Fitness** (int NumObjFun, ArrayList< Double > l)
void	**Mutation** (**Algorithm** a)
void	**CrossOver** (**Algorithm** a)
void	**Permutting** (**Algorithm** a)
void	**Boundary_Control** (**Algorithm** a)
void	**Selection_I** (**Algorithm** y)
void	**Selection_II** (**Algorithm** a, **Algorithm** b)

Figure 7 : Les fonctions de la Class MyAlgorithm

2 – 1 Définition des paramètres

Public Attributes

ArrayList< ArrayList< Double > >	**Population**
ArrayList< ArrayList< Double > >	**OldPopulation**
ArrayList< ArrayList< Double > >	**Mutant**
ArrayList< ArrayList< Double > >	**T**
ArrayList< ArrayList< Integer > >	**map**
ArrayList< ArrayList< Double > >	**Globalminimizer**
ArrayList< Double >	**fitnessT**
ArrayList< Double >	**FitnessP**
ArrayList< Integer >	**IndiceOldP**
Solution	**sol**
SetUpParams	**setup**
int	**Randmut**
double	**GlobalMin**

Figure 8 : Les Attributs à initialiser dans la classe MyAlgorithm

Figure 9 : Illustration des principaux aspects de l'algorithme BSA

2 – 2 Initialisation

BSA génère population P initiale et l'Ancient Population initiale OldP à l'aide :

$$P_{i,j} \sim U\left(\text{low}_j, \text{up}_j\right),$$

$$\text{oldP}_{i,j} \sim U\left(\text{low}_j, \text{up}_j\right),$$

Où P(i,j) et OldP(i,j) sont deux matrices de D Dimension et N individu respectivement, Low(j) et Up(j) signifier la limite inférieure et la limite supérieure de chaque valeur stocker dans la matrice.

2 – 3 Sélection I

Sélection I processus de BSO est le début de chaque itération. Il vise à sélectionner une nouvelle OldP pour le calcul de la direction de recherche en fonction de la population P et de la population historique OldP . La nouvelle OldP est de nouveau sélectionnée par le "si-alors" règle :

$$\text{if } a < b \text{ then oldP} := P \mid a, b \sim U(0,1),$$

où := est l'opération de mise à jour ; a et b représenter des nombres aléatoires entre 0 et 1. L'opération de mise à jour permet de donner a BSO une mémoire. Après OldP et de nouveau sélectionnée, dans l'ordre des individus en est permutée aléatoirement.

2 – 4 Mutation

L'opérateur de mutation est utilisé pour produire la forme initiale de la population d'essai M avec :

$$M = P + F * (OldP – P)$$

où F est le facteur de contrôle de l'amplitude de l'opérateur de mutation utilisée pour contrôler l'amplitude de la direction de recherche. La valeur F = 3*RANDN , où RANDN ~ (0, 1) , et N est une distribution normale standard.

2 – 5 Crossover

Dans ce processus, la BSO génère la forme finale de l'épreuve de la population T . Equiprobable BSA utilise deux stratégies de croisement pour manipuler les éléments choisis des individus à chaque itération. Les deux stratégies de générer différentes matrices entières binaire (map) de taille N*D pour sélectionner les éléments d'individus qui seront manipulée .

Cette stratégie utilise le paramètre Mix-Rate pour contrôler le nombre d'éléments d'individus qui seront manipulés par l'utilisation de [Mix-Rate * RANDN * D], où Mix-Rate = 1. La stratégie II manipule le seul élément d'individus choisis au hasard par l'utilisation de RANDI , où RANDI ~ U(0,D).

Les deux stratégies sont employées pour manipuler les éléments d'individus à travers le "si-alors" règle : si map(n,m) = 1 , où n Appartient [1,2,3,……,N] , et m appartient [1,2,3,……,M] alors, T sera mise à jour avec T(n,m) = P(n,m).

À la fin de processus de raccordement, si certaines individuelles en T ont dépassé les limites de l'espace permis recherche, ils auront besoin d'être régénéré par a de la fonction Initialisation.

2 – 6 Sélection II

Dans le processus de sélection II de BSA, les valeurs de remise en forme (Fitness) P(i) et T(i) sont comparés, utilisé pour mettre à jour la population P basée sur une sélection gourmande. Si T(i) a une meilleure valeur Fitness de P(i) ,Donc il est mis à jour. La population P est mise à jour par l'utilisation de :

$$P_i = \begin{cases} T_i, & \text{if fitness}\,(T_i) < \text{fitness}\,(P_i)\,, \\ P_i, & \text{else.} \end{cases}$$

si le meilleur individu de P(Pbest) a une meilleure valeur Fitness que le minimum global obtenue, le global minimizer sera mis à jour pour être Pbest , et la valeur minimale sera mis à jour à la valeur de Fitness Pbest.

III – Caractéristiques des fonctions pour le Benchmarking

Function	Function expression	Dim	Search space
Sphere $f1$	$f(x) = \sum_{i=1}^{n} x_i^2$	30	$[-100, 100]$
Sumsquare $f2$	$f(x) = \sum_{i=1}^{n} i x_i^2$	60	$[-10, 10]$
Sumpower $f3$	$f(x) = \sum_{i=1}^{n} \lvert x_i \rvert^{i+1}$	60	$[-1, 1]$
Schwefel2.22 $f4$	$f(x) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	100	$[-10, 10]$
Exponential $f5$	$f(x) = \exp\left(0.5 * \sum_{i=1}^{n} x_i\right) - 1$	60	$[-1.28, 1.28]$
Alpine $f6$	$f(x) = \sum_{i=1}^{n} \lvert x_i \sin(x_i) + 0.1 x_i \rvert$	60	$[-10, 10]$
Rastrigin $f7$	$f(x) = \sum_{i=1}^{n} (x_i^2 - 10\cos 2\pi x_i) + 10$	100	$[-100, 100]$
Griewank $f8$	$f(x) = \frac{1}{4000} \sum_{i=1}^{n} (x_i)^2 - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{x_i}}\right) + 1$	60	$[-600, 600]$
Rosenbrock $f9$	$f(x) = \sum_{i=1}^{n-1} \left[100 \left(x_i^2 - x_{i+1}\right)^2 + \left(x_i - 1\right)^2\right]$	30	$[-30, 30]$
Ackley $f10$	$f(x) = -20 \exp\left(-0.2 \sqrt{\frac{1}{n} \sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n} \sum_{i=1}^{n} \cos(2\pi x_i)\right) + 20 + e$	60	$[-32, 32]$
R-H-Ellipsoid $f11$	$f(x) = \sum_{i=1}^{n} \sum_{j=1}^{i} x_j^2$	80	$[-100, 100]$
Schwefel1.2 $f12$	$f(x) = \sum_{i=1}^{n} \left(\sum_{j=1}^{i} x_j\right)^2$	10	$[-30, 30]$
Elliptic $f13$	$f(x) = \sum_{i=1}^{n} \left(10^6\right)^{(i-1)/(n-1)} x_i^2$	30	$[-100, 100]$
NCRastrigin $f14$	$f(x) = \sum_{i=1}^{n} \left(y_i^2 - 10\cos 2\pi y_i\right) + 10$ $y_i = \begin{cases} x_i, & \lvert x_i \rvert < \frac{1}{2}, \\ \frac{\text{round}(2x_i)}{2}, & \lvert x_i \rvert \geq \frac{1}{2} \end{cases}$	50	$[-5.12, 5.12]$
T-H camel $f15$	$f(x) = 2x_1^2 - 1.05x_1^4 + \frac{x_1^6}{6} + x_1 x_2 + x_2^2$	2	$[-5, 5]$
Schaffer $f16$	$f(x) = \left(\sum_{i=1}^{n} x_i^2\right)^{0.25} \left(\sin^2\left(50 \left(\sum_{i=1}^{n} x_i^2\right)^{0.1}\right) + 1\right)$	10	$[-100, 100]$
Powell $f17$	$f(x) = \sum_{i=1}^{n/4} \left[\left(x_{4i-3} + 10x_{4i-2}\right)^2 + 5\left(x_{4i-1} - x_{4i}\right)^2 + \left(x_{4i-2} - 2x_{4i-1}\right)^4 + 10\left(x_{4i-3} + x_{4i}\right)^4\right]$	70	$[5, -4]$
Weierstrass $f18$	$f(x) = \sum_{i=1}^{n} \left(\sum_{k=0}^{k\max} \left[a^k \cos\left(2\pi b^k \left(x_i + 0.5\right)\right)\right]\right) - n \sum_{k=0}^{k\max} \left[a^k \cos\left(2\pi b^k 0.5\right)\right]$ $a = 0.5\ b = 3\ k\max = 20$	60	$[-0.5, 0.5]$

Figure 10 : Les paramètres d'initialisation pour chaque fonction.

IV – Comparaison du BSA avec d'autres algorithme d'optimisation

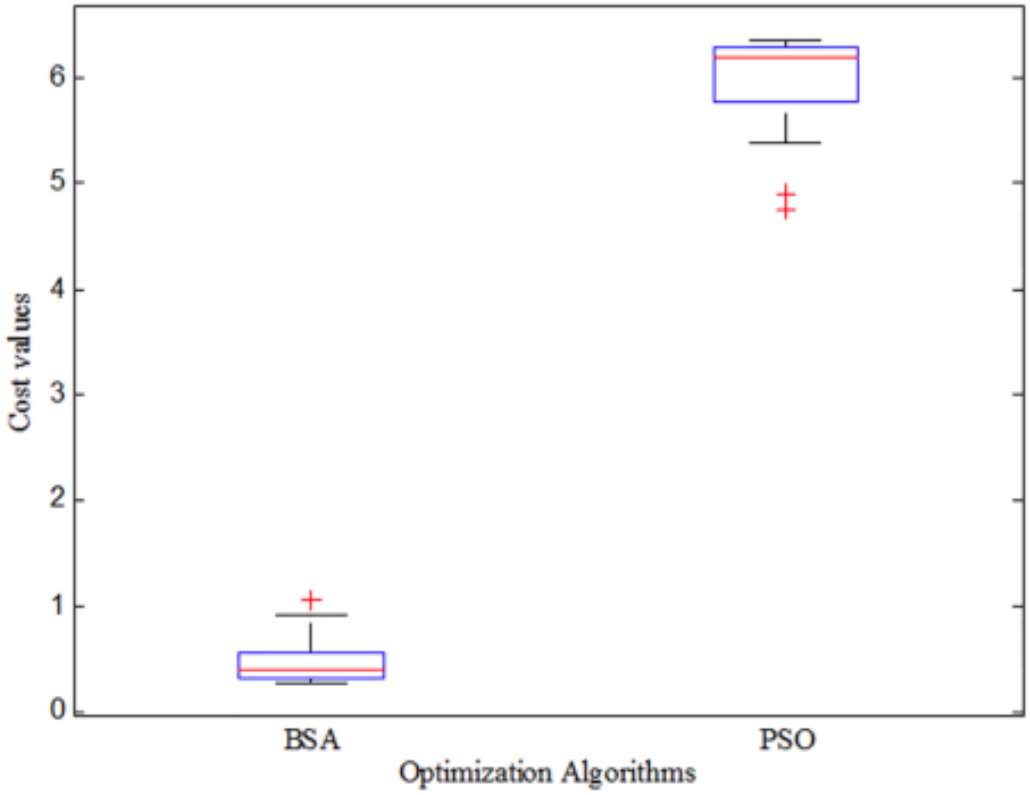


Figure 11 : La différence de couts entre BSA et PSO.

System	Objective function values	
	BSA	PSO
Best	0.2744	4.7645
Mean	0.4816	5.9176
Median	0.3946	6.1833
Worst	1.0642	6.3481
Standard deviation	0.2321	0.5166

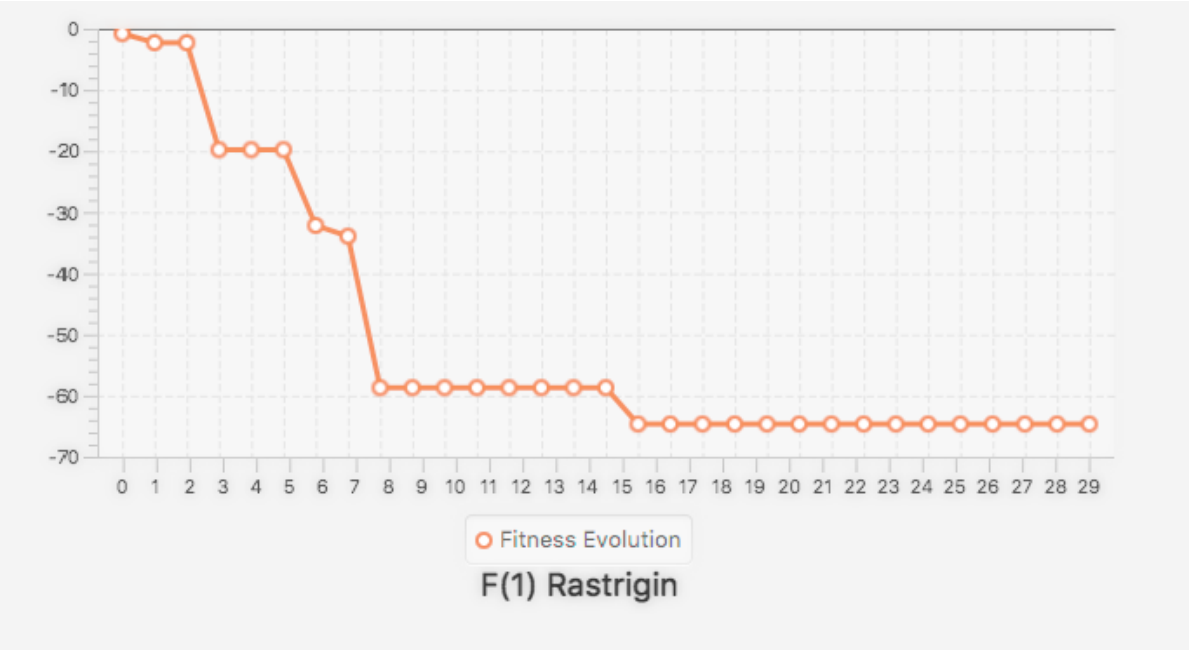
Figure 12 : La différence des valeurs obtenues pour les deux algorithmes BSA et PSO

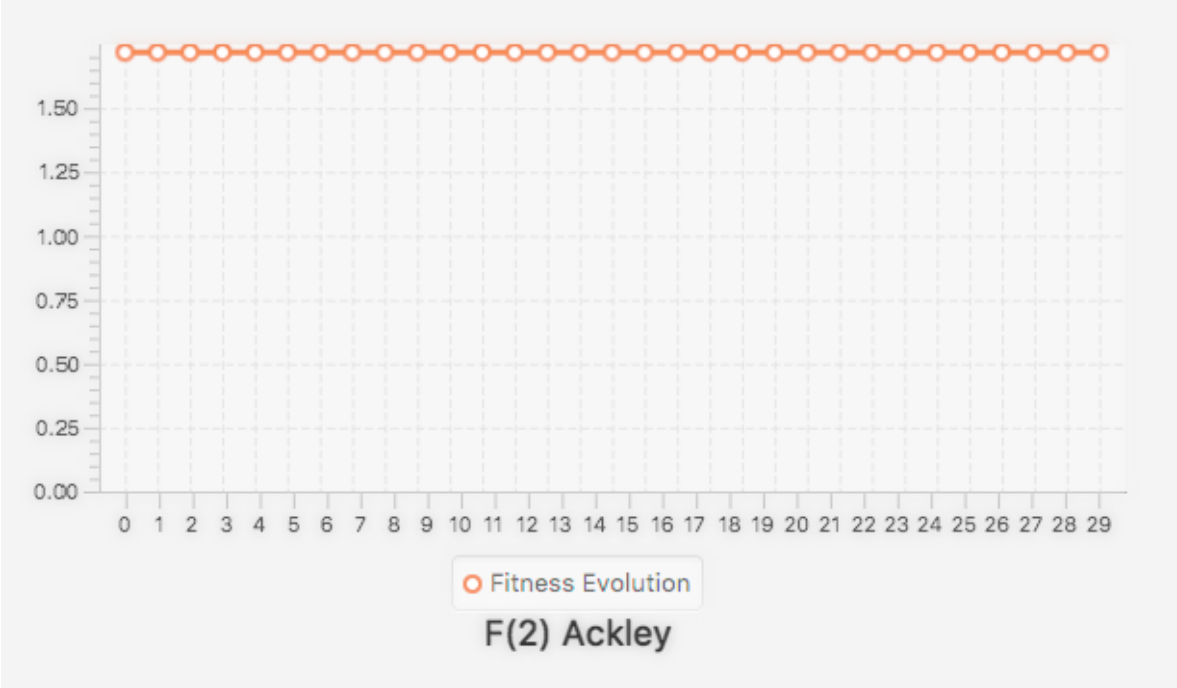
V – Résultats Statistiques

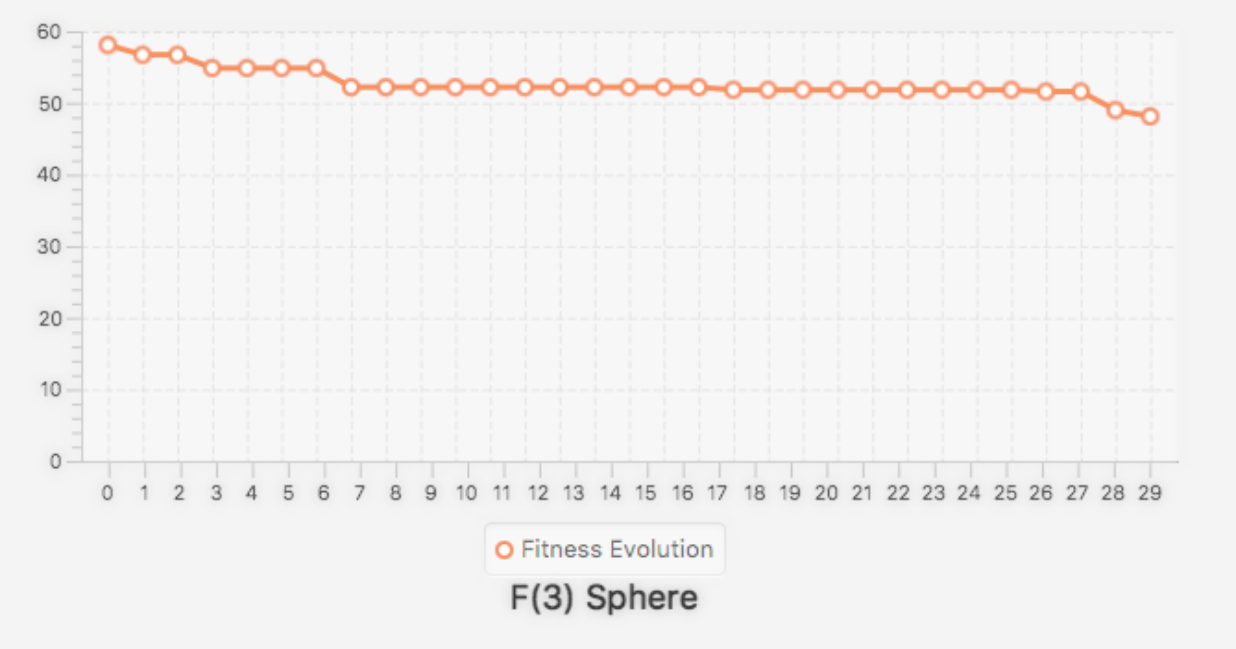
Benchmark test Function	Type	Dim	LB	UB	BSA Results	DE results	Diff. in BSA and DE results	Remarks
Ackley	MN	30	-32	32	Mean=0.2419432100000000000 Best=0.0015044900000000	Mean=5.147006037927210000 Best=3.906507044854710000	Diff. in Mean=-4.905063 Diff. in Best=-3.905003	Superior
Beale	MN	2	-4.5	4.5	Mean=0.0000000000000000000 Best=0.0000000000000000	Mean=0.0000000000000000000 Best=0.0000000000000000	Diff. in Mean=0.000000 Diff in Best=0.000000	Equivalent
Bohachecsky1	MS	2	-100	100	Mean=0.0000000000000000000 Best=0.0000000000000000	Mean=0.0000000000000000000 Best=0.0000000000000000	Diff. in Mean=0.000000 Diff. in Best=0.000000	Equivalent
Bohachecsky2	MN	2	-100	100	Mean=0.0000000000000000000 Best=0.0000000000000000	Mean=0.0000000000000000000 Best=0.0000000000000000	Diff. in Mean=0.000000 Diff. in Best=0.000000	Equivalent
Bohachecsky3	MN	2	-100	100	Mean=0.0000000000000000000 Best=0.0000000000000000	Mean=0.0000000000000000000 Best=0.0000000000000000	Diff. in Mean=0.000000 Diff. in Best=0.000000	Equivalent
Booth	MS	2	-10	10	Mean=0.0000000000000000000 Best=0.0000000000000000	Mean=0.0000000000000000000 Best=0.0000000000000000	Diff. in Mean=0.000000 Diff. in Best=0.000000	Equivalent
Branin	MS	2	-5	10	Mean=0.3978870000000000000 Best=0.397887000000000	Mean=0.397887357729738156 Best=0.397887357729738156	Diff. in Mean=-3.57E-07 Diff. in Best=-3.57E-07	Superior
Dixonprice	UN	30	-10	10	Mean=2.799248400000000000 Best=0.501522000000000	Mean=150.3156054817980000 Best=42.78478415186690000	Diff. in Mean=-147.5163 Diff. in Best=-42.283262	Superior
Easom	UN	2	-100	100	Mean=-1.0000000000000000000 Best=-1.0000000000000000	Mean=-1.0000000000000000000 Best=-1.0000000000000000	Diff. in Mean=0.000000 Diff. in Best=0.000000	Equivalent
Eggholder	MN	2	-512	512	Mean=-958.8674000000000000 Best=-959.641000000000000	Mean=-959.640662720850000 Best=-959.640662720850000	Diff. in Mean=0.773267 Diff. in Best=3.373E-04	Inferior
Goldstein Price	MN	2	-2	2	Mean=2.999999999999988764 Best=2.999999999999988764	Mean=2.999999999999880000 Best=2.999999999999880000	Diff. in Mean=7.64E-15 Diff. in Best=7.64E-15	Equivalent
Griewank	MN	30	-600	600	Mean=0.320525066972000000 Best=0.000008177860000000	Mean=4.395746676311910000 Best=2.099809713347540000	Diff. in Mean=-4.075222 Diff. in Best=-2.099802	Superior
Hartmann 3	MN	3	0	1	Mean=-3.865260000000000000 Best=-3.865260000000000	Mean=-3.865263785321100000 Best=-3.865263785321100000	Diff. in Mean=-3.78E-06 Diff. in Best=-3.78E-06	Superior
Hartmann 6	MN	6	0	1	Mean=-3.212560000000000000 Best=-3.212560000000000	Mean=-3.215227860086380000 Best=-3.212563960451090000	Diff. in Mean=-2.66E-03 Diff. in Best=-3.96E-06	Superior
Matyas	UN	2	-10	10	Mean=0.0000000000000000000 Best=0.0000000000000000	Mean=0.0000000000000000000 Best=0.0000000000000000	Diff. in Mean=0.000000 Diff. in Best=0.000000	Equivalent
Powell	UN	24	-4	5	Mean=0.128509234538000000 Best=0.000005399690000000	Mean=4.787954618004700000 Best=17.03519464816760000	Diff. in Mean=-4.659445 Diff. in Best=-17.035189	Superior
Schwefel	MS	10	-500	500	Mean=0.000154408000000000 Best=0.000127276000000000	Mean=0.006673260189018020 Best=0.000301591492643638	Diff. in Mean=-6.52E-03 Diff. in Best=-1.74E-04	Superior
Sixhump-Camelback	MN	2	-5	5	Mean=-1.031628453489877 64 Best=-1.031628453489877 64	Mean=-1.031628453489877 64 Best=-1.031628453489877 64	Diff. in Mean=0.000000 Diff. in Best=0.000000	Equivalent
Sumsquares	US	30	-10	10	Mean=0.025615835481704800 Best=0.000000000152524000	Mean=27.254133620712900000 Best=9.597323444431770000	Diff. in Mean=-27.22852 Diff. in Best=-9.597323	Superior
Zakharov	UN	10	-5	10	Mean=0.0000000000000000000 Best=0.0000000000000000	Mean=0.0000000000000000000 Best=0.0000000000000000	Diff. in Mean=0.000000 Diff. in Best=0.000000	Equivalent

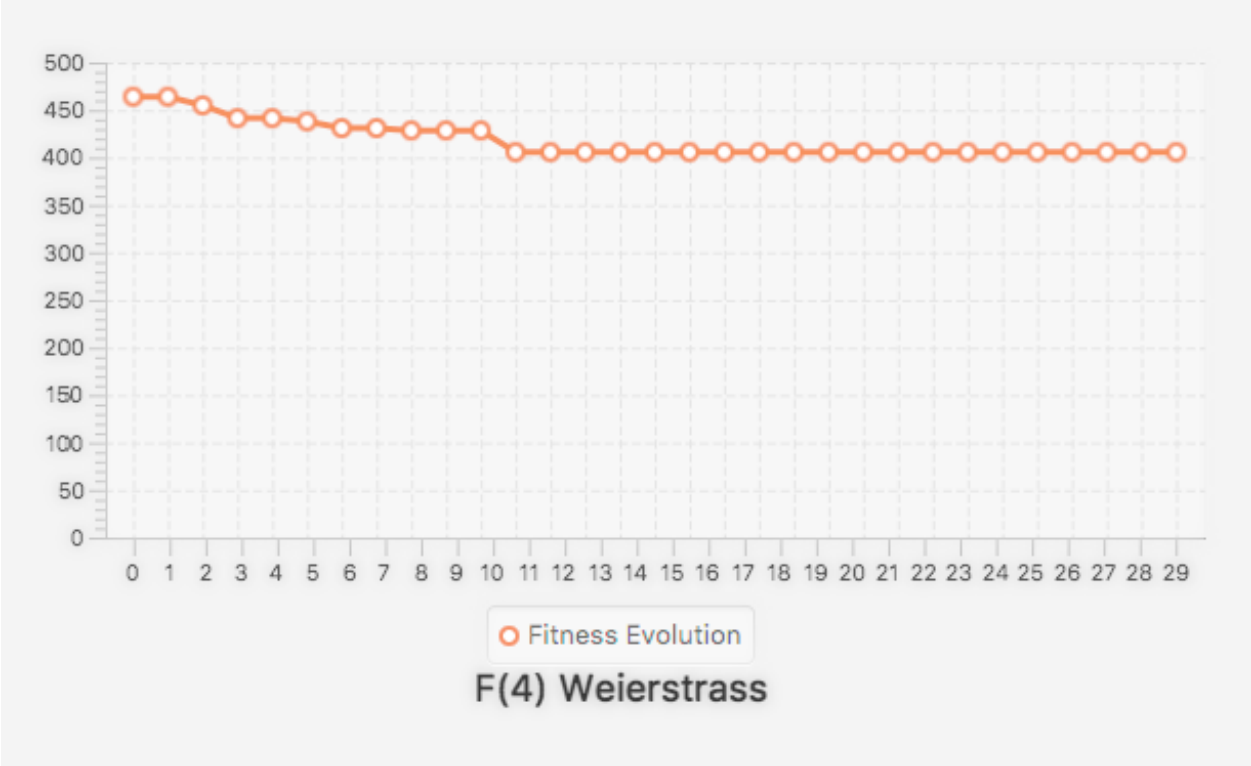
Figure 13 : Résultats statistiques de chaque fonction avec les paramètres d'initialisation.

VI – Visualisation des Résultats :






60
50
40
30
20
10
0
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29
Fitness Evolution
F(3) Sphere


500
450
400
350
300
250
200
150
100
50
0
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29
Fitness Evolution
F(4) Weierstrass

		Fitness Population(1)	Fitness Population(2)	Fitness Population(3)	Fitness Population(4)	Fitness Population(5)
2		-60.83873957	-16.79859033	-16.93052083	-45.22432289	-101.9661582
3		-47.46781775	-63.25731586	-60.208169	-16.54147788	-79.68752056
4		-15.88841395	-63.74046246	-57.22476332	-56.03716111	-9.986335083
5		-30.15820998	-50.47680485	-38.22389363	-52.19820057	-72.94873525
6		-58.20804341	-59.17554687	-19.30940035	-74.58028317	-76.65033666
7		-53.62708888	-20.73101248	-43.30579612	-53.75897692	-81.2347631
8		-50.97665893	-71.98063255	-25.59590223	-50.40140589	-92.73357718
9		-30.95605641	-37.91714714	-2.57437058	-17.13257073	-45.01355179
10		-70.35504642	-31.41754179	-37.48569476	-20.94383047	9.97147946
11		-22.48282505	-53.23540515	-48.65706895	-93.43877175	-58.68991925
12		-63.44442978	-84.17640446	-59.13734442	-44.36177022	-65.6788266
13		-69.93165898	-4.542246909	-43.64964264	-8.598371515	-17.23472464
14		-155.9069632	-14.58259484	-66.08733621	-52.48455126	-52.30610758
15		-50.07721836	-46.42151003	-62.03601955	-42.53634795	-44.48607769
16		-40.15902458	-53.49059327	-39.18105569	-11.80775188	-44.3593666
17		-53.360901	-66.00136272	-27.42314912	-4.866887345	-43.4134718
18		-24.74671153	-46.7066824	-10.96260381	-57.30960076	-77.29734527
19		-45.32551382	-66.6445022	2.347871069	-54.84078086	-21.02717746
20		-1.720097349	-61.41938316	-13.36532261	-50.09945939	-53.56914178
21		-44.5025826	-50.33366373	-17.15377211	12.01046391	-30.54678338
22		-16.42840614	-2.090572512	-73.40088774	-30.86968537	-29.73161659
23		-99.76799997	17.10625824	-28.26612272	-50.49038403	-64.52967344
24		-60.25672767	-26.5006654	-1.00246237	-28.01300895	22.57595948
25		-4.323803778	6.610380369	-45.22783941	-62.14291767	-53.13661308
26		-74.12686541	-18.31459561	-52.53385245	-19.235601	-99.35590166
27		-38.72038548	-50.85397883	-46.07025942	-61.5396089	-45.03506296
28		-83.39347783	-39.67583297	23.66737243	-83.83540616	-56.25400755
29		-45.2081926	-5.235284617	-53.63503742	-53.4958353	-60.67773364
30		6.632248006	13.65343443	-13.91968728	-60.19489041	-36.61770454
31		-62.92264067	-61.29765551	-20.41711318	-27.75582591	-39.85765076
32	**Average**	**-48.95500844**	**-37.65493052**	**-33.23232815**	**-42.42417408**	-50.71594817
33	**Deviation**	**31.73400033**	**26.99334113**	**22.99691676**	**23.76957122**	28.92747583

Figure 14 : Calcule de la moyenne et l'écart-type des fitness de population pour 5 Exécutions de l'algorithme BSO.

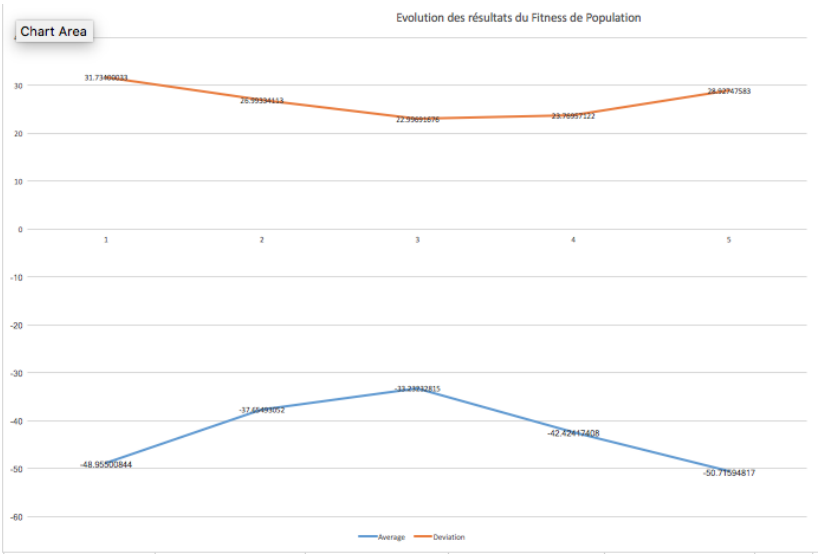


Figure 15 : L'évolution des résultats obtenues par l'exécution de l'algorithme BSO

Conclusion

Dans ce rapport, on a introduit un algorithme, appelé BSA, qui a été proposé pour résoudre les problèmes d'optimisation continue sans contrainte à l'échelle mondiale. La structure BSA algorithmique permet de bénéficier de la génération précédente de populations par l'utilisation de solutions qu'il a trouvé, dans le passé, pour un problème donné, il recherche les solutions avec de meilleures valeurs de remise en forme (Fitness). Les BSO bio inspiré, philosophie est analogue à la déclaration d'un groupe social d'êtres vivants à des intervalles aléatoires pour les zones de chasse qui ont été précédemment jugé fructueux pour l'obtention de nourriture. Les valeurs obtenues par l'algorithme de BSO sont comparés à d'autres algorithmes évolutionnaires pour prouver que la BSO est plus efficace en termes de temps d'utilisation du processeur (comme c'est plus rapide que la plupart des algorithmes de comparaison) ainsi que dans la détermination de la solution optimale. Il a été prouvé que, dans la plupart des cas présentés dans ce document fourni de BSO ou des solutions optimales, avec très peu d'effort de calcul. L'algorithme est recommandé pour les grands, des problèmes complexes avec une dimension supérieure à 30. Les tests détaillés discutés dans ce rapport démontre que la BSO est statistiquement réussi à résoudre des problèmes d'optimisation numérique. Les facteurs responsables de de succès de la BSO par rapport à d'autres algorithmes sont comme suit :

la BSO et la mutation des opérateurs de croisement produisent des populations d'essai très efficace à chaque génération. La BSO pour la stratégie de génération de paramètre F, qui contrôle l'amplitude de la direction de recherche, peut produire à la fois de grande amplitude numérique nécessaires à une recherche globale et la faible amplitude des valeurs nécessaires pour une recherche locale d'une façon très équilibrée et efficace. Cela améliore nettement la capacité du BSO à résoudre des problèmes.

Table des figures

Liste des acronymes

PSO : L'essaim de particules

ACO : Colony optimization

CS : Cuckoo recherchez

ABC : La colonie d'abeilles artificielles

EAS : Les algorithmes évolutionnaires

GA : L'algorithme génétique

DE : L'évolution différentielle

CMAES : Covariance matrix évolution strategy

BSO : L'algorithme d'optimisation de recherche en arrière

SVM : Support vector machine

Webographie

[1]http://journals.plos.org/plosone/article/file?id=10.1371/journal.pone.0146277&type=printable

[2]http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.735.2432&rep=rep1&type=pdf

[3] https://www.sciencedirect.com/science/article/pii/S1877050915020050

[4] https://ac.els-cdn.com/S1876610217354309/1-s2.0-S1876610217354309-main.pdf?_tid=ea3f2767-86a6-401d-85e3-de0a56fc1489&acdnat=1529462525_3a8375cb1a40d0c631f17f805a6249e0

[5] https://www.deepdyve.com/lp/hindawi-publishing-corporation/a-hybrid-backtracking-search-optimization-algorithm-with-differential-61EhZ50S6R?articleList=%2Fsearch%3Fquery%3Dbacktracking%2Bsearch%2Boptimization

[6]https://www.researchgate.net/publication/299488767_Backtracking_Search_Optimization_Algorithm_BSA?channel=doi&linkId=56fb8da808ae1b40b8050641&showFulltext=true

[7]https://pdfs.semanticscholar.org/d99e/01532f493f273642ff4a0626c0ef54b5e92b.pdf