Recherche coopérative d'optimum global

Damien Vergnet, Elsy Kaddoum, Nicolas Verstaevel, Frédéric Amblard

Recherche coopérative d'optimum global

D. Vergnet[1], E. Kaddoum[1], N. Verstaevel[1], F. Amblard[1]

[1] IRIT, Université de Toulouse, CNRS, Toulouse INP, UT3, **UT1**, **UT2**, Toulouse, France

damien.vergnet@irit.fr

Résumé

Dans cet article nous proposons une nouvelle métaheuristique basée sur la coopération pour chercher l'optimum global de fonctions à optimiser. Elle repose sur deux processus de recherche locale et semi-locale coopérative. Ses performances sont comparées à quatre autres métaheuristiques sur des problèmes d'optimisation mono-objectif sans contraintes. Les résultats montrent que l'approche proposée est capable de trouver le minimum global des fonctions testées plus rapidement que les méthodes comparées, tout en nécessitant un nombre d'itérations et d'appels à la fonction objectif plus faibles.

Mots-clés

coopération locale, décision collective, optimisation par métaheuristique, recherche locale

Abstract

This paper proposes a new cooperation-based metaheuristic for searching global optima of optimization functions. It relies on a local search process coupled with a cooperative semi-local search process. Its performances are compared against four other metaheuristics on unconstrained mono-objective optimization problems. Results show that the proposed metaheuristic is able to find the global minimum of the tested functions faster than the compared methods while reducing the number of iterations and the number of calls of the objective function.

Keywords

local cooperation, collective decision, metaheuristic optimization, local search

1 Introduction

La simulation est un outil précieux pour la compréhension du comportement et des limites de systèmes. Plusieurs études ont pour objectif de reconstruire des systèmes virtuels, appelés jumeaux numériques, pour simuler et vérifier le comportement de systèmes spécifiques [**?**]. De tels systèmes peuvent être mis en œuvre dans le domaine de la mobilité ou les études de catastrophes naturelles dans le but de reproduire des conditions de simulation spécifiques et comprendre les raisons de tels phénomènes [4]. La conception d'un jumeau numérique qui reproduirait le comportement exact d'un système réel n'est pas une tâche facile [**?**]. Étant donné que les système réels sont généralement des systèmes complexes qui présentent des interdépendances non-linéaires entre leurs paramètres, il est difficile de trouver la meilleure fonction à utiliser dans le modèle et d'adapter en temps réel ses paramètres pour garder le comportement de la simulation le plus proche possible de celui du système réel. De nombreuses études ont formalisé le problème de la calibration en un problème d'optimisation dans lequel les paramètres de la fonction modélisée sont ajustés par l'optimisation d'une fonction objectif : les paramètres de simulation deviennent des variables de décision et les sorties pertinentes du modèle sont intégrés dans des fonctions objectifs [2, 8]. Tout ceci implique la nécessité d'avoir un système d'optimisation capable de s'adapter rapidement aux changements qui pourraient survenir dans le système réel.

Plusieurs méthodes d'optimisation existent et pourraient être utilisées pour résoudre ce problème mais elles présentent d'important inconvénients tels qu'une tendance à converger vers des optimums locaux ou un manque de rapidité [6, 11, 14].

Dans ce papier, nous proposons une nouvelle métaheuristique d'optimisation locale appelée **CoBOpti**, pour ***Co****operation-****B****ased* ***Opti****mization* (Optimisation basée sur la coopération). Elle est basée sur une hypothèse de continuité locale de la dynamique de la fonction objectif, c'est-à-dire que la valeur de la fonction objectif ne varie pas significativement lorsque la valeur des variables de décision varie peu. Par rapport aux méthodes standards de l'état de l'art, CoBOpti atteint les solutions optimales en un nombre réduit d'itérations et d'évaluations de la fonction objectif.

Les principales contributions sont les suivantes :

— Nous introduisont une **nouvelle métaheuristique d'optimisation locale basée sur une hypothèse de continuité et de coopération**. Cette hypothèse permet de modéliser le problème de la recherche d'optimum global comme un **problème de coopération** où un point détermine le prochain point à explorer en exploitant l'information accumulée dans son voisinage.

— Nous menons des expériences visant à comparer notre approche sur des problèmes d'optimisation mono-objectif sans contraintes avec une seule variable de décision dans le but de démontrer que **l'approche proposée permet d'atteindre un optimum global tout**

en minimisant le nombre d'évaluations de la fonction objectif.

Ce papier est organisé de la manière suivante : la section 2 discute des limites de certaines métaheuristiques. La section 3 présente notre approche et comment elle répond à ces limites. Dans la section 4, nous introduisons les résultats de nos expériences qui sont ensuite discutées dans la section 5, avant de conclure avec ses limites actuelles et quelques suggestions de recherches futures.

2 État de l'art

[3] définit les problèmes d'optimisation comme la recherche d'un vecteur $\bar{x}_n^* = (x_1^*, ..., x_n^*)$ qui optimise une fonction objectif

$$\bar{f}_k(\bar{x}_n) = (o_1(\bar{x}_n), ..., o_k(\bar{x}_n)) \quad (1)$$

où $\bar{x}_n = (x_1, ..., x_n)$ est un vecteur de n variables de décision.

De nombreuses méthodes de résolution de problèmes d'optimisation existent, chacune émettant des hypothèses sur la nature du problème. Ces méthodes sont généralement classées par catégorie, nous nous intéressons à celle appelée métaheuristique. [6] définit les métaheuristiques comme des méthodes qui présentent deux niveaux de recherche, local et de plus haut niveau, et qui sont capables de sortir d'optimums locaux. Cette définition inclut entre autres les méthodes qui mettent en œuvre le concept de voisinage. Le voisinage d'une solution s est l'ensemble des solutions atteignables depuis s.

Les métaheuristiques sont intéressantes pour résoudre des problèmes d'optimisation car elles sont conçues pour explorer efficacement les espaces de recherche complexes [6]. Sörensen *et al.* [12] ajoutent que la grande majorité des problèmes d'optimisation réels sont plus facilement résolus par des métaheuristiques, d'où notre focalisation sur ces méthodes dans ce papier.

Les métaheuristiques reposent sur deux notions importantes : l'**intensification** et la **diversification**. L'intensification est un processus qui focalise la recherche sur les régions de l'espace de recherche qui semblent prometteuses, c'est-à-dire celles dans le voisinage de la meilleure solution actuelle. La diversification est un processus dont l'objectif est l'exploration de régions encore inconnues de l'espace de recherche dans l'espoir de trouver de meilleurs solutions. Ces notions reposent généralement sur la mémorisation des solutions visitées [5].

Il existe de nombreuses métaheuristiques, chacune avec leurs propres hypothèses. Étant donné que notre approche est destinée à être mise en œuvre dans la calibration en temps réel, elle doit s'appuyer sur des algorithmes rapides et capables de gérer l'ensemble des solutions visitées. Les méthodes présentées ci-dessous sont donc celles reposant sur un processus de **recherche locale** et/ou basées sur la **notion de population**.

Les algorithmes de **recherche locale** explorent l'espace de recherche en visitant le voisinage immédiat de la solution courante s et en sélectionnant la solution voisine qui a une valeur objectif plus petite que s. Afin de sortir des optimums locaux, ils présentent une phase de *hill-climbing* qui autorise une dégradation temporaire de la fonction objectif. Ces méthodes incluent le recuit simulé (RS), le *Generalized Simulated Annealing* (GSA), la recherche locale itérée (*Iterated Local Search*), la recherche locale guidée (*Guided Local Search*), etc. [6]. Le principale avantage de ces méthodes est leur rapidité. Elles présentent cependant une limite importante : elles ont tendance à se coincer dans des optimums locaux [11]. Certaines métaheuristiques locales, telles que la recherche tabou, utilisent une mémoire des solutions visitées pour contourner cette limite [6].

Une autre catégorie de métaheuristiques regroupe les **algorithmes à base de population**. Ces méthodes s'appuient sur un ensemble de solutions, appelé population. L'espace de recherche est exploré en évaluant chaque solution et en les modifiant avec une ensemble de règles simples. Il existe deux sous-groupes : les méthodes évolutionnaires et les méthodes s'inspirant de la nature [6].

Les **algorithmes évolutionnaires** (AE) sont des méthodes itératives basées sur la notion de *fitness*. La *fitness* d'une solution représente sa qualité d'après la fonction objectif. Au cours de chaque itération, appelée un *génération*, la fitness de chaque solution est évaluée. Les solutions possédant une fitness suffisament élevée sont conservées pour la génération suivante, toutes les autres sont mises de côté. De nouvelles solutions sont générées par croisement et mutation stochastique des meilleurs solutions issues de la phase de sélection. Cette catégorie inclut des méthodes telles que les algorithmes génétiques, l'évolution différentielle (ED) et la programmation génétique [6, 9]. Contrairement aux méthodes de recherche locale, les AE sont moins susceptibles de se coincer dans des optimums locaux grâce à des tailles de population importantes. Cet avantage induit néanmoins un autre inconvénient : plus la taille de la population augmente, plus l'algorithme requière une puissance de calcul élevée et donc des temps de résolution plus longs.

Il existe d'autres méthodes basées sur des populations qui diffèrent des AE. Elles s'inspirent de systèmes biologiques complexes tels que les nuées d'oiseaux ou les colonies de fourmis. Elles présentent le même avantage que les AE, à savoir une plus faible tendance à rester coincé dans des optimums locaux que les méthodes locales, mais souffrent aussi de temps de calcul plus longs et couteux [6]. D'autres méthodes, telles que PSO (*Particle Swarm Optimization*), présentent cependant une pauvre distribution de l'information au sein de la population de solutions, ce qui implique une tendance à converger trop rapidement vers des optimums locaux [14].

Dans notre méthode, nous proposons de combiner la vitesse de la recherche locale et la distribution de l'information des méthodes à base de population. Afin d'atteindre ce but, nous empruntons les notions de voisinage et de raisonnement collectif de ces méthodes. En se basant sur l'hypothèse que la **dynamique de la fonction objectif ne varie pas significativement entre deux points très proches**, nous proposons un système qui **recherche un optimum global en s'appuyant sur le raisonnement collectif des**

solutions déjà visitées.
Des métaheuristiques de recherche locale et basées sur des populations ont été présentées ainsi que leurs limites dans le contexte de la calibration en temps réel. La prochaine section décrit notre méthode, CoBOpti, que nous évaluons dans la section 4.

3 CoBOpti : optimisation par coopération

Dans cette section nous introduisons CoBOpti, une métaheuristique d'optimisation basée sur la coopération. La méthode que nous proposons combine la vitesse de la recherche locale et la distribution de l'information des algorithmes à base de population.
La section 3.1 décrit le principe général de l'approche en présentant un aperçu des différentes phases de recherche. La section 3.2 détaille le processus de recherche locale. La section 3.3 détaille le processus de recherche semi-locale et comment il permet de sortir des optimums locaux. Enfin, la section 3.4 décrit comment les points coopèrent pour résoudre quelques situations spécifiques.

3.1 Principe général

L'objectif de CoBOpti est d'explorer itérativement la surface d'une fonction objectif dans le but de trouver un optimum global. Au cours de chaque itération, le système doit déterminer le prochain point à explorer. Un **point** p_i est défini par une paire $p_i = (x_i, o_i)$ où x_i est la valeur de la variable de décision et o_i la valeur de la fonction objectif pour x_i. La succession des points visités lors d'une recherche locale est appelée une **chaine**. L'algorithme est constitué de quatre phases (figure 1).
L'algorithme combine deux étapes de recherche différentes : une **recherche locale** (**phases 1, 2 et 3**) dont l'objectif est de découvrir un minimum local, et une **recherche semi-locale** (**phase 4**) qui exploite l'ensemble des minimums locaux déjà découverts pour chercher un minimum global.
Le but de la **recherche locale** (**phase 1**) est de trouver un minimum local. Chaque itération t débute avec une chaine contenant des points déjà visités $p(t), p(t-1)$, etc. Parmi tous les points de la chaine, le système choisit deux points pour déterminer dans quelle direction il doit explorer (**phases 2 et 3**). Ce processus continue jusqu'à ce qu'un minimum local soit trouvé, c'est-à-dire jusqu'à ce que la distance selon l'axe x entre les deux points avec la valeur objectif la plus basse de la chaine courante soit inférieure à ε_{dist}.
L'objectif de la **recherche semi-locale** est d'explorer la fonction en direction d'un minimum global. Ce processus doit décider quel point $p(t+1)$ il faut explorer en se basant sur les minimums locaux déjà visités (**phase 4**). Une fois que le processus a décidé quel point explorer en suivant, une nouvelle chaine est créée et la recherche locale reprend à partir de ce nouveau point.
La recherche s'arrête lorsque qu'un minimum local a une valeur objectif inférieure à un seuil prédéfini ε_{obj}.

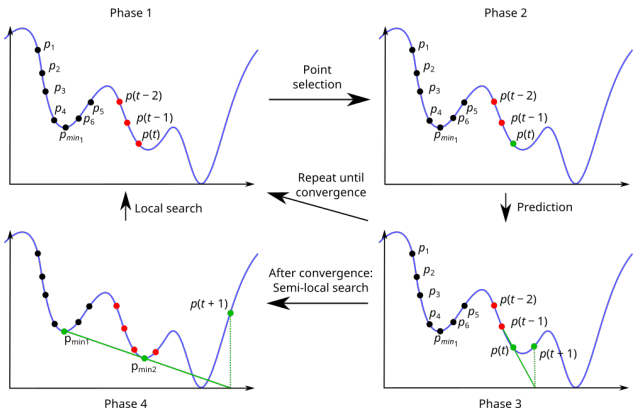

FIGURE 1 – Les phases de recherche de CoBOpti : sélection d'un point, recherche locale, recherche semi-locale

La notion de chaines est importante car elle permet l'isolation de groupes de points (points noirs et rouges dans la figure 1). Il n'est en effet pas désirable que des points distants interagissent lors du processus de recherche locale à cause de potentielles erreurs importantes entre la valeur réelle de la fonction et son estimation. L'utilisation de chaines implique que des points distants ne peuvent pas être utilisés pour calculer les approximations linéaires lors de la recherche locale (cf. section 3.2) et réduit donc le risque d'erreurs. Plusieurs chaines sont créées tout au long du processus d'optimisation.
Les sections suivantes détaillent la manière dont les points sont sélectionnés et comment $p(t+1)$ est calculé. La section 3.2 décrit comment le processus de recherche sélectionne les points successifs pour atteindre un minimum local. La section 3.3 décrit comment le système sort de minimums locaux et chercher un optimum global. Enfin, la section 3.4 décrit comment les points coopèrent pour résoudre quelques situations difficiles.

3.2 Recherche locale

L'objectif de la recherche locale est de suivre la courbe de la fonction objectif pour trouver un minimum local. Durant chaque itération t, le prochain point $p(t+1)$ à explorer est déterminé en calculant une approximation linéaire de la fonction objectif à partir de deux points de la chaine courante.
Étant donné que ces deux points sont proches relativement au domaine de définition de la variable de décision, nous considérons que les fonctions linéaires sont une approximation acceptable de la fonction objectif dans ce contexte.
Le premier point sélectionné est celui qui possède la plus faible valeur objectif au temps t, noté p_{min}. Le second point sélectionné est un des voisins de p_{min}. Deux points p_1 et p_2 d'une chaine sont considérés **voisins** s'ils sont à côté l'un de l'autre, c'est-à-dire s'il n'existe pas de point p_3 entre eux selon l'axe des x. Un point peut avoir un maximum de deux voisins. Par exemple, sur la figure 1, les points p_1 et p_2 sont voisins alors que p_2 et p_4 ne le sont pas.
Puisque p_{min} est le point avec la valeur objectif la plus

basse, il possède à tout moment un ou deux voisins.
Les phases 2 et 3 de la figure 1 illustrent la première situation où $p(t) = p_{min}$ (point vert) a un seul voisin $p(t-1)$. La composante x du prochain point $p(t+1)$ est calculbe par approximation linéaire de la fonction objectif entre $p_{min} = (x_{min}, o_{min})$ et son seul voisin $p(t-1) = p_n = (x_n, o_n)$:

$$x(t+1) = x_n + \frac{-o_n(x_{min} - x_n)}{o_{min} - o_n} \quad (2)$$

Cette équation calcule la composante x du point qui aurait une valeur objectif nulle d'après l'approximation linéaire de la fonction objectif.
Afin de s'assurer que l'hypothèse initiale sur la dynamique de la fonction reste vraie, le prochain point ne doit pas être distant de plus de k_{dist} fois la distance entre p_{min} et p_n. Si c'est le cas, $x(t+1)$ est fixé à $x_{min} + k_{dist}(x_{min} - x_n)$. Dans nos expérimentations, $k_{dist} = 5$ a été utilisée.
Dans la seconde situation, p_{min} a deux voisins p_l et p_h, tout deux ayant une valeur objectif plus élevée que p_{min}. Ceci implique qu'un minimum local se situe quelque part entre p_l et p_h. $x(t+1)$ est donc déterminé par :

$$x(t+1) = \frac{x_{min} + x_n}{2} \quad (3)$$

où x_n est la composante x de p_l ou p_h alternativement. La figure 1 montre un exemple de cette situation (points noirs). Le point p_6 a été calculé en utilisant cette équation avec p_4 en tant que p_{min} et p_5 en tant qu'un de ses voisins.
Nous tenons à faire remarquer que la fonction objectif n'a pas besoins d'être réévaluée à l'emplacement du voisin sélectionné puisque sa valeur est supposée ne pas avoir changé depuis sa première évaluation.
Ce processus est répété jusqu'à ce qu'un minimum local soit trouvé. Le point p_{min} est considéré comme étant un minimum local lorsque sa distance à un de ses voisins est inférieure à ε_{dist}.

3.3 Recherche semi-locale

L'objectif de la recherche semi-locale est de trouver un minimum global en parcourant la surface définie par l'ensemble des minimums locaux déjà découverts. On considère que cette surface donne la tendance générale de la fonction objectif et peut donc être approximée par des fonctions linéaires sans engendrer trop d'erreurs.
Afin de calculer la composante x du prochain point $p(t+1)$ à partir d'approximations linéaires de la fonction, deux points doivent être sélectionnés : le dernier minimum local $p_{min1} = (x_{min1}, o_{min1})$ trouvé par le processus de recherche locale et un de ses voisins. Les **voisins** d'un minimum local sont les autres minimums locaux adjacents. À l'instar des points décrits dans la section 3.2, les minimums locaux possèdent un maximum de deux voisins.
La table 1 décrit quel voisin est sélectionné en fonction de la situation explorée, où $p_l = (x_l, o_l)$ (resp. $p_h = (x_h, o_h)$) est le voisin de p_{min1} avec une composante x plus basse (resp. plus élevée).
Dans les situations 1, 2, 3 et 4, le prochain point $x(t+1)$ est calculé à partir de l'équation 2 en remplaçant p_{min} par

	Situation	Voisin sélectionné
1	1 voisin p_n	p_n
2	2 voisins, $o_l < o_{min1} < o_h$	p_l
3	2 voisins, $o_l > o_{min1} > o_h$	p_h
4	2 voisins, $o_l < o_{min1} > o_h$	p_l si $o_l < o_h$, sinon p_h
5	2 voisins, $o_l > o_{min1} < o_h$	p_l si $o_l < o_h$, sinon p_h

TABLE 1 – Voisin de p_{min1} sélectionné en fonction de la situation

p_{min1} et $p(t-1)$ par le voisin sélectionné. La phase 4 de la figure 1 illustre ce processus pour la situation 1. Dans ce diagramme, il y a deux minimums locaux connus, p_{min1} et p_{min2}, ce dernier étant celui découvert en dernier. Le prochain point $p(t+1)$ est estimé par approximation linéaire de la fonction entre ces deux minimums locaux. À l'instar de la recherche locale, $p(t+1)$ ne doit pas être à une distance supérieure à $k_{dist}|x_{min1} - x_n|$. Si tel est le cas, la même opération que celle décrite en section 3.2 est appliquée pour ramener le point sous cette distance.
Dans la situation 5, puisque p_l et p_h ont tous deux une valeur objectif supérieure à celle de p_{min1}, un optimum global ce trouve probablement entre p_l et p_h. L'équation 3 est utilisée pour déterminer le prochain point.
Une fois que $x(t+1)$ a été calculé, le processus de recherche locale reprend à partir de ce point avec une nouvelle chaine.

3.4 Mécanismes de coopération

Les sections 3.2 et 3.3 ont décrit le comportement nominal de CoBOpti. Au cours des processus de recherche locale et semi-locale, le système peut rencontrer un certains nombre de situations particulières. Cette section présente des règles de coopération pour les détecter et les résoudres.
Cas 1. Durant la recherche locale, lorsqu'une chaine est créée, soit parce qu'il s'agit de la première itération ou parce que la recherche semi-locale vient d'en créer une nouvelle, elle ne contient qu'un seul point. Ce point n'a donc aucun voisin à sa disposition pour calculer le prochain point. $x(t+1)$ est alors choisi aléatoirement parmi $\{x_{min} - \delta, x_{min} + \delta\}$ où $\delta = \frac{1}{k_{prop}}|x_{low} - x_{high}|$ et x_{low} (resp. x_{high}) est la borne inférieure (resp. supérieure) du domaine de définition de x. Dans nos expérimentations, $k_{prop} = 100$ a été choisi.
Cas 2. Durant la recherche semi-locale, une situation similaire peut survenir, dans laquelle il n'existe qu'un seul minimum local connu. Dans ce cas-ci, puisqu'aucune approximation linéaire ne peut être calculée, un processus de **hill-climbing** est enclanché pour sortir du minimum local calculé. Ce processus repose sur les deux points de la dernière chaine qui possèdent la valeur x la plus élevée et la plus faible. Ces points sont appelés les extremums. L'objectif est donc de remonter les pentes de la fonction autour du minimum local pour trouver une autre pente de direction opposée (figure 2).
La recherche se focalise sur la pente où se situe l'extremum avec la plus basse valeur objectif. Le prochain point est cal-

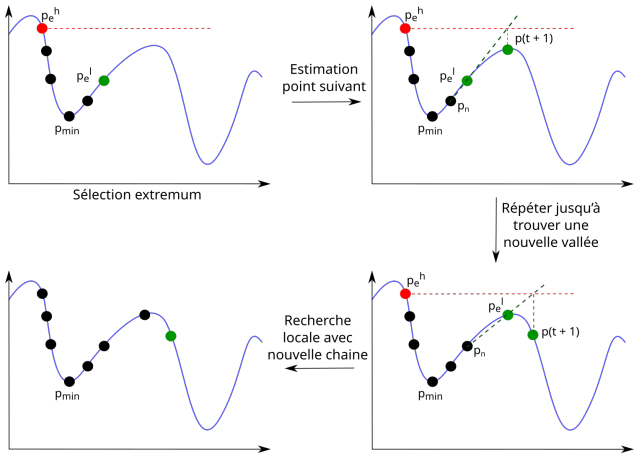

FIGURE 2 – Processus de hill-climbing

culé à partir de l'équation 4 où $p_e^l = (x_e^l, o_e^l)$ est l'extremum avec la plus basse valeur objectif et $p_e^h = (x_e^h, o_e^h)$ est l'autre extremum. $p_n = (x_n, o_n)$ est l'unique voisin de p_e^l. Cette équation calcule la composante x du prochain point qui aurait une valeur objectif égale à celle de l'autre extremum d'après l'approximation linéaire de la fonction entre p_e^l et son voisin p_n.

$$x(t+1) = x_e^l + \frac{(o_e^h - o_e^l)(x_n - x_e^l)}{o_n - o_e^l} \tag{4}$$

À l'itération suivante, si la valeur objectif réelle est supérieur à o_e^h, la recherche change de côté ; si ce n'est pas le cas, elle continue sur le même côté. Ce processus est répété jusqu'à ce que la valeur objectif réelle soit inférieure à o_e^l. Le processus de recherche locale reprend alors avec une nouvelle chaine.

Durant la phase de hill-climbing, la distance $|x(t+1) - x_e^l|$ ne doit pas être inférieure à un seuil δ_{min} afin d'éviter que le processus ne ralentisse trop.

Cas 3. Il est possible que le processus de recherche locale trouve un minimum local qui a déjà été découvert au cours des itérations précédentes. Afin d'éviter que le système ne boucle indéfiniment, deux décisions peuvent être prises. Si un processus de hill-climbing a déjà été initié précédemment à cet endroit, le prochain point $x(t+1)$ est calculé par approximation linéaire avec un facteur 2 afin d'explorer une nouvelle région un peu plus éloignée et d'éviter de revenir sur ce même minimum local. Si aucune phase de hill-climbing n'a déjà été entreprise, elle est démarrée pour explorer une nouvelle pente.

Deux minimums locaux sont considérés identiques si leur distance selon l'axe x est inférieure à un seuil ε_{same}.

Cette section a présenté notre approche. Elle s'appuie sur la notion de chaines de points. Nous avons d'abord présenté le processus de recherche locale sur une chaine qui permet de trouver des optimums locaux. Lorsqu'un optimum local est trouvé, un processus de recherche semi-locale permet de trouver de nouvelles régions à explorer dans l'espace de recherche. Des mécanismes de coopération ont été introduits pour répondre à quelques situations particulières, diversifier les solutions et créer de nouvelles chaines.

Dans la section suivante, nous évaluons les performances de notre méthode. Nous la comparons à quatre autres métaheuristiques sur des problèmes d'optimisation mono-objectifs sans contraintes.

4 Expérimentations et résultats

Cette section compare les performances de CoBOpti avec quatre autres méthodes citées dans la section 2 : le recuit simulé (RS), le *Generalized Simulated Annealing* (GSA), l'évolution différentielle (ED) et la *Particle Swarm Optimization* (PSO).

La section 4.1 présente les différentes fonctions utilisées pour tester les performances. La section 4.2 décrit le protocole pour comparer les performances de CoBOpti avec les autres méthodes. La section 4.3 présente les résultats des expériences. Enfin, les résultats sont discutés dans la section 5.

4.1 Fonctions de test

Quatres fonctions ont été sélectionnées pour les expériences de comparaison des performances :

1. Gramacy et Lee (domaine : $[0.5, 2.5]$) ;
2. Ackley (paramètres : $d = 1$, $a = 20$, $b = 0.2$, $c = 2\pi$; domaine : $[-32, 32]$) ;
3. Rastrigin (paramètres : $d = 1$; domaine : $[-5.12, 5.12]$) ;
4. Levy (paramètre : $d = 1$; domaine : $[-10, 10]$).

Ces fonctions ont été choisies car elles présentent de nombreux minimums locaux, un unique minimum global et un seul paramètre [1, 7, 10, 13].

4.2 Comparaison des méthodes

Les performances de chaque approche (RS, GSA, ED et PSO) sont comparées avec celle de CoBOpti. Chaque méthode a été implémentée en Python 3.8. GSA et ED ont été implémentées en utilisant `scipy.optimize`, PSO a été implémentée avec le package `pyswarm.pso` et RS a été implémenté par nous-même. Les paramètres optionnels de GSA, ED et PSO sauf ceux liés aux bornes, à l'état initial et au nombre maximal d'itérations ont été laissés à leur valeur par défaut.

Les variables de controle de CoBOpti sont définies comme suit : $\varepsilon_{dist} = 10^{-4}$ (seuil de détection d'un minimum local), $\varepsilon_{same} = 0.01$ (distance minimale entre deux minimums locaux), $\delta_{min} = 10^{-4}$ (le pas minimum lors de la phase de hill climbing) et $\varepsilon_{obj} = 5 \cdot 10^{-3}$ (seuil de détection du minimum global).

Pour chaque méthode, sauf PSO, la valeur initiale v_{init} de la variable de décision pour chaque exécution est sélectionnée par une séquence de Sobol. Sachant que les valeurs générées par cette séquence sont dans l'intervalle $[0, 1]$, elles sont ajustées au domaine de la variable de décision par la formule $v_{init} = s \cdot (d_{max} - d_{min}) + d_{min}$ où s est la valeur générée par la séquence. Aucune valeur initiale n'a pu

être spécifiée pour PSO car l'implémentation utilisée ne le permettait pas.

Trois métriques sont définies : le **taux de succès** (proportion d'exécutions qui ont abouti au minimum global), le **nombre d'itérations** nécessaire pour atteindre le minimum global et le **nombre d'évaluations de la fonction objectif**. Si une exécution ne parvient pas à atteindre le minimum global, le nombre d'itérations effectuées est fixé à la valeur maximale autorisée, 1000 dans les résultats suivants.

4.3 Résultats

La table 2 montre le taux de succès, le nombre moyen d'itérations et d'évaluations de la fonction objectif sur 200 exécutions pour chaque méthode et fonction, avec un maximum de 1000 itérations.

CoBOpti s'est montré capable de trouver le minimum global pour chacune des quatre fonctions. Le nombre moyen d'itérations se situe entre 35 et 100 ; le nombre d'évaluations de la fonction objectif est similaire.

Les 1000 itérations constantes pour RS et GSA s'expliquent par le critère d'arrêt de ces méthodes. Elles s'appuient sur le nombre d'itérations écoulé pour calculer des distributions de probabilités : plus le nombre d'itérations écoulé est grand, moins il y a de chance que l'algorithme sélectionne des actions qui n'améliorent pas la solution courante. Une fois que le nombre maximal d'itérations est atteint, aucune action qui dégraderait la solution ne peut être sélectionnée et l'algorithme s'arrête. La solution avec la valeur objectif la plus basse est ensuite retournée.

RS n'a pas donné de bons résultats, sauf dans le cas de la fonction de Gramacy et Lee avec un taux de succès de près de 100 %. Il a donné de très mauvais résultats pour la fonction d'Ackley avec seulement 2 % de réussite. Ces résultats sont cohérents avec ce qui a été décrit dans l'état de l'art (section 2).

GSA a donné de très bons résultats avec 100 % de réussite pour toutes les fonctions. Le nombre d'évaluations est cependant deux fois supérieur à RS, autour de 2000.

Le taux de succès de l'ED est un peu plus bas que les autres méthodes, sauf RS. Cependant, le nombre moyen d'itération est plutôt bas, entre 8 et 50 itérations sont nécessaires pour trouver le minimum global.

PSO a été en mesure de trouver le minimum global pour les quatre fonctions avec un faible nombre d'itérations, entre 20 et 50. Par contre, le nombre d'évaluations de la fonction est plus élevé que les autres méthodes, de 2000 à plus de 4500.

Méthode	Fonction	Succès	Nb. itér.	Nb. éval.
CoBOpti	G. & L.	100 %	49.31	50.31
	Ackley	100 %	95.94	96.94
	Rastrigin	100 %	80.69	81.69
	Levy	100 %	35.3	36.3
SA	G. & L.	99.5 %	1000	1000
	Ackley	2 %	1000	1000
	Rastrigin	10.5 %	1000	1000
	Levy	30 %	1000	1000
GSA	G. & L.	100 %	1000	2035.58
	Ackley	100 %	1000	2124.43
	Rastrigin	100 %	1000	2039.97
	Levy	100 %	1000	2019.60
DE	G. & L.	97.5 %	8.71	154.54
	Ackley	100 %	49.62	801.63
	Rastrigin	94 %	30.91	481.06
	Levy	100 %	50.45	773.75
PSO	G. & L.	100 %	20.61	2008.70
	Ackley	100 %	46.92	4638.51
	Rastrigin	100 %	25.57	2505.57
	Levy	100 %	20.02	1951.32

TABLE 2 – Taux de succès, nombre moyen d'itérations et d'évaluations de la fonction pour les méthodes testées

5 Analyse et discussion

L'hypothèse initiale de continuité de la dynamique de la fonction objectif a été validée par les expérimentations sur plusieurs fonctions standards. CoBOpti a montré des taux de succès plus élevés que RS et ED, et presque aussi bons que GSA et PSO. Même si le nombre d'itérations nécessaires à CoBOpti est comparable à celui de ED ou PSO, le nombre d'évaluations de la fonction est bien plus faible pour CoBOpti.

Ce faible nombre d'évaluations peut être attribué au fait que la fonction objectif n'est évaluée qu'une seule fois par point visité. Ce comportement découle de l'hypothèse initiale qui stipule que la dynamique de la fonction objectif ne change pas significativement entre deux points proches.

Les temps d'exécutions n'ont pas été montrés car les différences n'étaient pas significatives. Ceci est très certainement du à la faible complexité des fonctions sélectionnées.

Un analyse de sensibilité devrait être menée pour tester l'influence de k_{dist} et k_{prop} sur les performances de CoBOpti.

Notre méthode a été testée uniquement sur des problèmes d'optimisation mono-objectif sans contrainte avec une seule variable de décision. De plus amples recherches sont nécessaires pour généraliser cette approche aux problèmes multi-objectifs avec plusieurs variables de décision. Le principe central de la méthode que nous explorons reste très similaire à ce qui a été présenté ici. De nouveaux mécanismes de coopération seront ajoutés pour la sélection de l'objectif à minimiser et des variables de décision à ajuster à chaque itération. Quelques expériences ont été menées et semblent indiquer que l'augmentation du nombre de fonctions objectif (optimisation de problèmes multi-objectifs) n'impacte que peu les performances de CoBOpti.

D'autres expérimentations sont également à mener avec des fonctions plus complexes. Les cas d'applications réels étant sujets à des données bruitées, la résilience au bruit doit être testée.

6 Conclusion

Dans ce papier, CoBOpti, une nouvelle métaheuristique pour l'optimisation globale, a été présentée. Elle se fonde

sur une hypothèse de continuité locale de la dynamique de la fonction objectif. CoBOpti explore l'espace de recherche en s'appuyant sur la coopération des solutions visitées, elle-même basée sur l'hypothèse mentionnée plus tôt.
Ce papier se focalise sur des problèmes d'optimisation globale mono-objectifs non contraints avec une seule variable de décision. Les expérimentations montrent que CoBOpti nécessite moins d'évaluations de la fonction objectif par rapport à d'autres métaheuristiques communes tout en maintenant des taux de succès similaires voire meilleurs sur des fonctions unidimensionnelles.
CoBOpti est une proposition prometteuse pour la calibration en temps réel. En effet, son faible nombre d'évaluations de la fonction objectif pourrait être utile dans le context de la calibration en ligne de modèles de simulation complexes avec des fonctions objectif couteuses en temps de calcul. Cette propriété pourrait aider à réduire le temps nécessaire pour calibrer ce type de modèle.

Références

[1] Md. Alauddin. Mosquito flying optimization (MFO). In *2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT)*, pages 79–84, March 2016.

[2] Richard Arsenault, Annie Poulin, Pascal Côté, and François Brissette. Comparison of Stochastic Optimization Algorithms in Hydrological Model Calibration. *Journal of Hydrologic Engineering*, 19(7) :1374–1384, July 2014. Publisher : American Society of Civil Engineers.

[3] Jin-Hee Cho, Yating Wang, Ing-Ray Chen, Kevin S. Chan, and Ananthram Swami. A Survey on Modeling and Optimizing Multi-Objective Systems. *IEEE Communications Surveys Tutorials*, 19(3) :1867–1901, 2017. Conference Name : IEEE Communications Surveys Tutorials.

[4] Chao Fan, Cheng Zhang, Alex Yahja, and Ali Mostafavi. Disaster City Digital Twin : A vision for integrating artificial and human intelligence for disaster management. *International Journal of Information Management*, 56 :102049, February 2021.

[5] Michel Gendreau and Jean-Yves Potvin. Tabu Search. In Edmund K. Burke and Graham Kendall, editors, *Search Methodologies : Introductory Tutorials in Optimization and Decision Support Techniques*, pages 165–186. Springer US, Boston, MA, 2005.

[6] Michel Gendreau and Jean-Yves Potvin, editors. *Handbook of Metaheuristics*, volume 146 of *International Series in Operations Research & Management Science*. Springer US, Boston, MA, 2010.

[7] Robert B. Gramacy and Herbert K. H. Lee. Cases for the nugget in modeling computer experiments. *Statistics and Computing*, 22(3) :713–722, May 2012.

[8] Jingtao Ma, Hu Dong, and H. Michael Zhang. Calibration of Microsimulation with Heuristic Optimization Methods. *Transportation Research Record*, 1999(1) :208–217, January 2007. Publisher : SAGE Publications Inc.

[9] Karol R. Opara and Jarosław Arabas. Differential Evolution : A survey of theoretical analyses. *Swarm and Evolutionary Computation*, 44 :546–558, February 2019.

[10] Mitchell A. Potter and Kenneth A. De Jong. A cooperative coevolutionary approach to function optimization. In Yuval Davidor, Hans-Paul Schwefel, and Reinhard Männer, editors, *Parallel Problem Solving from Nature — PPSN III*, Lecture Notes in Computer Science, pages 249–257, Berlin, Heidelberg, 1994. Springer.

[11] Rainer Storn and Kenneth Price. Differential Evolution – A Simple and Efficient Heuristic for global Optimization over Continuous Spaces. *Journal of Global Optimization*, 11(4) :341–359, December 1997.

[12] Kenneth Sörensen, Marc Sevaux, and Fred Glover. A History of Metaheuristics. *Handbook of Heuristics*, to appear, January 2017.

[13] Fevrier Valdez and Patricia Melin. Parallel Evolutionary Computing using a cluster for Mathematical Function Optimization. In *NAFIPS 2007 - 2007 Annual Meeting of the North American Fuzzy Information Processing Society*, pages 598–603, June 2007.

[14] Yudong Zhang, Shuihua Wang, and Genlin Ji. A Comprehensive Survey on Particle Swarm Optimization Algorithm and Its Applications. *Mathematical Problems in Engineering*, 2015 :e931256, October 2015. Publisher : Hindawi.