Optimisation coûteuse[1] : où en sommes-nous et que faire maintenant ?

Denis Pallez
Université Côte d'Azur, CNRS, I3S

Introduction à l'optimisation

Optimiser un problème fait partie du domaine de la recherche opérationnelle[2] qui se définit comme la mise en œuvre de méthodes scientifiques permettant d'identifier les meilleures solutions x^* parmi un ensemble Ω de solutions possibles x, également dénommées *échantillons* dans la suite de cet article. Ω est appelé *espace de recherche*. Pour savoir si un échantillon est le meilleur, les méthodes se fondent sur une fonction f qui définit la qualité des décisions, dénommée fonction objectif. Identifier le meilleur échantillon consiste à optimiser (minimiser ou maximiser) cette fonction et se formalise à l'aide de l'expression (1) pour la minimisation.

$$\begin{aligned} x^* &= \arg\min_{x\in\Omega} f(x) = \{f_1(x), \ldots, f_n(x)\} \\ \text{avec} \quad &\begin{cases} g_i(x) \leq 0, & i = 0, \ldots, p \quad p \geq 0 \\ h_j(x) = 0, & j = 0, \ldots, q \quad q \geq 0 \end{cases} \end{aligned} \tag{1}$$

Dans l'espace de recherche, il est possible qu'un certain nombre d'échantillons ne soient pas « réalisables ». Cette faisabilité s'exprime sous forme de contraintes à l'aide d'égalités h_j ou d'inégalités g_i sur x pour inclure ou exclure une partie de Ω. Pour faciliter la résolution, les contraintes d'égalités sont

1. Cet article est une adaptation francaise de l'article *Expensive Evolutionary Computation: Where Are We, and What's Next?* https://doi.org/10.1007/978-3-032-23607-4_18.
2. Société française de recherche opérationnelle et d'aide à la décision (ROADEF), https://roadef.org.

DOI:10.48556/SIF.1024.27.31

souvent transformées en contraintes d'inégalités ($|h_j(x)| - \varepsilon \leq 0$ pour une valeur positive de ε « très » petite, par exemple $\varepsilon \approx 10^{-4}$).

La qualité d'un échantillon s'exprime souvent à l'aide d'une seule valeur réelle ($n = 1$) et dans ce cas, on parle d'optimisation mono-objectif ($f : \Omega \to \mathbb{R}$, ou $f : \Omega \to [0, 1]$ pour une version normalisée) mais il est possible de devoir gérer plusieurs critères ($n > 1$) et dans ce cas, on parle d'optimisation multiobjectif ($f : \Omega \to \mathbb{R}^n$) ou *many-objective* ($n > 3$). Par exemple, identifier le plus court chemin entre deux villes peut être le résultat de la minimisation simultanée du temps de déplacement et de la distance parcourue, auxquels il est possible d'ajouter un critère de coût du transport. Souvent, cette optimisation simultanée est réduite à une optimisation mono-objectif quand une préférence est connue sur les objectifs en transformant le problème en une combinaison linéaire sur les critères affectés de leur poids (scalarisation). Dans le cas contraire, quand les critères doivent être considérés indépendamment l'un de l'autre comme le temps et la distance, la notion d'optimalité est redéfinie grâce au concept d'efficacité où aucun objectif n'est plus important qu'un autre et est formalisé par la relation de dominance ($\prec$) introduite par Pareto [37]. Dans ce cas, plusieurs solutions optimales sont possibles, au moins une pour chaque objectif, correspondant aux meilleurs compromis possibles entre les différents critères. Indépendamment du nombre de critères, quand plusieurs solutions optimales existent ($|x^*| > 1$), on parle d'optimisation multimodale [40].

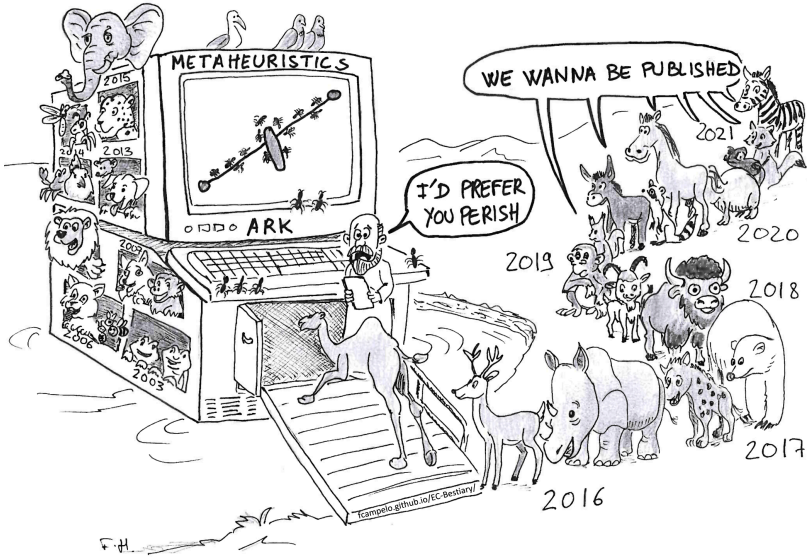
Un échantillon x est souvent caractérisé par plusieurs paramètres $x = (x_1, \ldots, x_m)$. Si tous les paramètres x_i sont réels ($\Omega \subseteq \mathbb{R}^m$) alors on parle d'optimisation continue, s'ils sont entiers ($\Omega \subseteq \mathbb{Z}^m$), on parle d'optimisation discrète. S'ils représentent un ensemble de possibilités sans relation entre elles comme des villes par exemple, on parle d'optimisation combinatoire, et enfin d'optimisation mixte s'il y a une disparité des domaines de définition des paramètres (i.e. $\Omega \subseteq \mathbb{Z}^d \times \mathbb{R}^c$ avec $c, d > 0$).

Évolution artificielle

De nombreuses méthodes ont vu le jour pour résoudre les différents problèmes d'optimisation précédents et sont souvent classées en deux grandes catégories : les méthodes déterministes (ou exactes) et les méthodes non-déterministes (ou stochastiques). Les techniques déterministes (programmation linéaire et non-linéaire, programmation dynamique…) fournissent toujours la même solution même si elles sont exécutées à plusieurs reprises et assurent souvent l'optimalité globale de la solution. Toutefois, elles nécessitent souvent des informations sur la dérivée de la fonction objectif ou des contraintes, ainsi qu'un point initial approprié de Ω ; ce qui les rend inapplicables quand, par exemple, la fonction f n'est pas dérivable, pas

continue ou quand elle est fondée sur des simulations numériques (optimisation boîte noire). À l'inverse, les méthodes stochastiques, basées sur des heuristiques, n'utilisent pas d'information sur la dérivée mais échantillonnent itérativement la fonction objectif pour converger vers la solution optimale ou une solution relativement proche (sous-optimale) sans pour autant être capables d'assurer l'optimalité de la solution.

Lorsque l'heuristique utilisée n'est pas spécifique à un problème en particulier, on parle de métaheuristique. Ces méthodes ont la capacité de s'échapper d'optima locaux mais fournissent souvent des décisions différentes d'une exécution à l'autre. Elles sont connues pour être capables de traiter des problèmes NP-difficiles, ce qui implique que l'on ne connaît pas d'algorithme de décision fonctionnant en temps polynomial pour traiter ces problèmes. De nombreuses métaheuristiques ont été proposées. À partir de techniques classiques comme les algorithmes génétiques (*Genetic Algorithm* – GA) s'inspirant de la théorie de l'évolution de Darwin ou des colonies de fourmis (*Ant Colony Optimization* – ACO) reproduisant artificiellement le comportement des insectes, en passant par exemple par des techniques combinant méthode exacte et recherche globale (*memetic algorithm* [12]), ces vingts dernières années ont été marquées par une explosion de métaheuristiques basées sur des métaphores d'un processus naturel (comportement des poissons, abeilles, oiseaux…) voire surnaturel (algorithme des zombies ou de réincarnation) [8] à tel point que la communauté s'est insurgée contre cette pratique [7], [49] qui n'était qu'une conséquence malsaine du *publish or perish*.



La figure 1 représente le principe général de ces métaheuristiques faisant évoluer artificiellement une population d'échantillons de l'espace de recherche Ω (évolution artificielle) en combinant ces échantillons pour en produire de nouveaux dans l'espoir d'aboutir à une solution optimale ou sous-optimale dans un temps raisonnable fixé *a priori*. Même si ces techniques sont employées pour résoudre des problèmes NP-difficiles, [57] considère qu'elles font face à des problèmes de plus en plus difficiles que les auteurs définissent comme les défis des 5M (multidimension, multichangement, multioptimum, multicontrainte et multicoût) que l'on retrouve dans de nombreux contextes industriels ou réels.

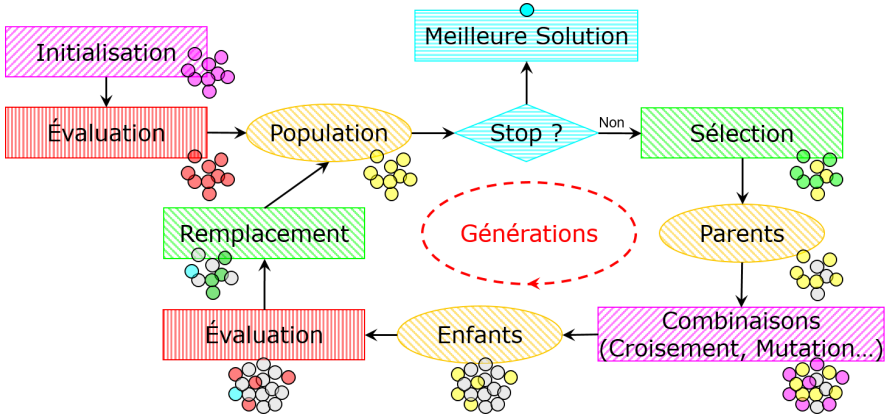


Fig. 1. Schéma général de l'évolution artificielle inspiré de la librairie logicielle *Evolving Object*.

Ces défis peuvent être caractérisés par de nombreux paramètres ($m \approx 1\,000$ à $10\,000$ par exemple), par une fonction objectif dynamique qui peut changer pendant le processus d'optimisation, par une fonction multimodale ayant de nombreux optima ($|x^*| \approx 8000$, *many-modal* [30]), par un problème défini par beaucoup de contraintes ($p + q$ relativement grand dans l'expression (1)) ou encore par une fonction dite coûteuse, que nous caractérisons dans la section suivante. Plusieurs conférences internationales sont dédiées à ces techniques comme *GECCO*[3], *CEC*[4], *PPSN*[5], *Evo**[6] ou encore *EA*[7].

3. https://dl.acm.org/conference/gecco.
4. https://ieeexplore.ieee.org/xpl/conhome/1000284/all-proceedings.
5. https://ppsn2026.disi.unitn.it.
6. https://link.springer.com/conference/evoapplications.
7. https://link.springer.com/conference/ae.

Optimisation coûteuse

La frontière entre une optimisation coûteuse et non coûteuse n'est pas clairement établie et la définir correctement n'est pas chose aisée. De prime abord, ce qui caractérise le mieux une optimisation coûteuse est le nombre très limité de fois où la fonction objectif peut être utilisée. Il est difficile de donner un chiffre précis car cela dépend étroitement du problème à résoudre et des ressources disponibles. À notre connaissance, [24] est le premier à avoir formalisé un tel coût :

$$\text{Coût} = \frac{\mathcal{O}(\text{Algorithme}) \times \mathcal{O}(\text{Évaluation})}{|\text{Processeurs}|} \tag{2}$$

où $\mathcal{O}(\textit{Évaluation})$ représente le coût moyen en temps de chaque évaluation de la fonction objectif, $\mathcal{O}(\textit{Algorithme})$ le coût moyen en temps de la méthode d'optimisation qui, elle aussi, peut être coûteuse, si, par exemple, elle est basée sur la construction de modèles d'approximation (*Surrogate-assisted evolutionary algorithm*). Il est naturel d'envisager le calcul intensif pour réduire le coût temporel en augmentant le nombre de processeurs ; toutefois, cela peut ne pas suffire pour faire en sorte que le problème ne soit plus coûteux. Et même si cette augmentation peut réduire le coût temporel (seul coût majoritairement considéré jusqu'à récemment), cela aura pour effet d'augmenter de manière non négligeable le coût financier ou énergétique. Alors que nous entrons dans une ère de sobriété, cela n'est pas souhaitable et devrait également être pris en considération.

L'optimisation coûteuse peut être induite par une fonction objectif basée sur un grand jeu de données (*data-driven optimization* [19]), ou par le fait que l'espace de recherche est bien trop vaste et qu'identifier un optimum prend du temps (*large-scale optimization* [32]). Certains considèrent également que l'optimisation multicritère (*multi-objective* ou *many-objective*) peut également être coûteuse car elle peut nécessiter beaucoup d'évaluations pour bien caractériser l'ensemble des compromis optimaux entre les différents critères (front de Pareto) [44]. C'est peut-être parce que ces types de problèmes (multiobjectif, à grandes dimensions, dynamiques, basés sur un grand volume de données…) sont assez bien formalisés et qu'ils peuvent être à l'origine d'une optimisation coûteuse que cette dernière n'est pas identifiée comme un axe de recherche significatif dans la communauté des métaheuristiques. En témoigne l'état de l'art de [9] et sa cartographie des différents domaines et sous-domaines de l'évolution artificielle où l'optimisation coûteuse est malheureusement absente.

Pourtant, grâce à Scopus[8], nous avons calculé le pourcentage d'articles de recherche dont le résumé, le titre ou les mots clés utilisent les termes d'évolution

8. `https://www.scopus.com`, données récoltées en décembre 2023.

artificielle coûteuse entre 2000 et 2023 (*expensive* + *evolutionary* + *optimization*) parmi l'ensemble des articles du domaine (*evolutionary* + *optimization*). On peut constater que le pourcentage ne cesse d'augmenter (figure 2), ce qui montre que l'optimisation coûteuse est bien une thématique émergente qui prend de l'ampleur.

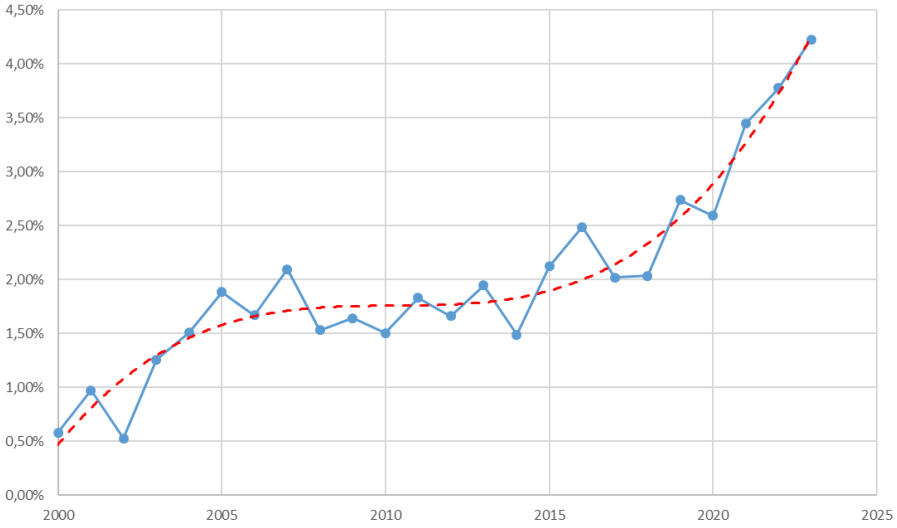


Fig. 2. Évolution du pourcentage d'articles (ligne brisée) traitant d'évolution artificielle coûteuse *Expensive Evolutionary Optimization*) et courbe de tendance (courbe en pointillés) sur les 25 dernières années.

Certains auteurs tentent même d'identifier les caractéristiques des problèmes d'optimisation réels *via* une enquête [6]. Sur 45 réponses, 42 % des problèmes mettent de plus d'une minute à plusieurs jours pour évaluer les fonctions objectif et sur les 34 problèmes possédant des contraintes, 30 % mettent un temps similaire pour évaluer le respect des contraintes ; ce qui n'est pas négligeable. Dit autrement, sur 45 problèmes, 27 % d'entre eux ne peuvent recourir qu'entre 10^2 et 10^3 fois à la fonction objectif ; ce qui caractérise assez bien un problème coûteux. Même si le nombre de réponses de l'enquête est assez limité et n'est pas suffisamment représentatif de tous les problèmes d'optimisation auxquels font face les ingénieurs ou chercheurs, il apparait clairement que les fonctions de test (*benchmarks*) construites par la communauté scientifique pour comparer les algorithmes présentent des lacunes. Dans [5], on propose une bibliothèque publique contenant quatre problèmes d'optimisation coûteux provenant de différentes applications

réelles afin de pouvoir comparer les algorithmes dédiés à la résolution de ce type de problème.

En considérant la formule (2) et comme le soulignent [24], [57], il existe trois grandes pistes pour améliorer la résolution d'un problème coûteux :

– minimiser le coût de la fonction objectif ;
– minimiser le coût de l'algorithme d'optimisation et faire en sorte qu'il converge plus rapidement que d'habitude ;
– augmenter les ressources affectées aux différents calculs en augmentant la part de calculs parallèles et distribués compte tenu du fait que les métaheuristiques à base de population sont naturellement parallélisables.

Les sections suivantes présentent le point de vue de l'auteur sur chacune des catégories d'améliorations ci-dessus, en ajoutant quelques travaux et observations plus récents par rapport aux publications existantes sur le sujet. Le discours se poursuit ensuite par une conclusion et une section plus prospective qui présente plusieurs pistes de recherche.

Minimiser le coût de la fonction objectif

Les auteurs de [24] proposent une taxonomie contenant trois stratégies principales pour réduire le coût de l'évaluation des fonctions objectif et de respect des contraintes : la *simplification* du problème à optimiser, la *substitution multifidélité* et l'*approximation* des fonctions. La hiérarchie des stratégies abordées est illustrée dans la figure 3.

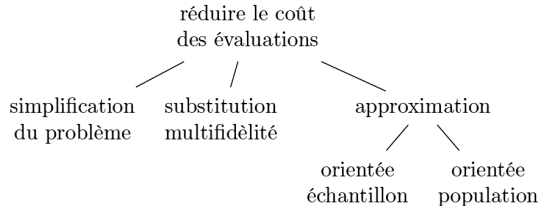


Fig. 3. Hiérarchie des stratégies pour réduire le coût d'évaluation des fonctions objectif et de la violation des contraintes.

L'objectif de la simplification du problème est de transformer un problème coûteux en une optimisation numérique du problème, par exemple en la basant sur de la simulation numérique, allégeant ainsi la charge de calcul du problème réel initial. Un exemple concret de ce type de stratégie est l'optimisation de structures aérodynamiques, pour laquelle les essais en soufflerie sont remplacées par des simulations numériques de mécanique des fluides en 2D ou 3D beaucoup moins coûteuses à réaliser [18]. Le principal inconvénient

de cette stratégie est qu'elle dépend de la nature du problème. Il n'est pas toujours possible de simplifier le problème initial afin d'en créer un moins coûteux. De plus, les auteurs omettent de mentionner la perte potentielle de précision qui peut survenir au cours du processus de simplification.

En revanche, la stratégie dite de *substitution multifidélité* prend en compte cette précision. Elle repose sur l'hypothèse forte que les évaluations sont basées sur des paramètres qui peuvent réduire le coût de l'évaluation au détriment de la précision du résultat. Le principe consiste à réaliser des simulations moins coûteuses ayant un niveau de précision assez faible afin d'explorer plus rapidement une zone intéressante de l'espace de recherche. Dans cette zone, des simulations plus précises peuvent être réalisées mais avec un coût d'évaluation plus important. Les évaluations coûteuses d'origine sont alors remplacées par un modèle multifidélité qui trouve un compromis entre des évaluations de différents niveaux de fidélité, par exemple avec une fidélité croissante au fil du temps. Des recherches récentes dans ce sous-domaine ont montré une préférence pour la combinaison de plusieurs modèles multifidélité [22]. Cependant, cette stratégie n'est pas applicable à tous les problèmes coûteux, même si elle est très prometteuse.

À l'inverse, la stratégie d'*approximation* s'applique à la fois aux évaluations coûteuses des objectifs et aux évaluations du respect des contraintes. Cette approche consiste à créer des modèles d'approximation qui tirent les leçons des évaluations coûteuses précédentes afin de prédire l'évaluation de nouveaux échantillons à moindre coût. Elle est largement adoptée dans la littérature [16] car elle peut être appliquée à de nombreux domaines sans hypothèse forte. Cependant, une question clé consiste à déterminer quels échantillons sont évalués à l'aide d'une évaluation coûteuse et lesquels sont approximés à l'aide d'un modèle d'approximation.

C'était également le cas pour son ancêtre, l'*héritage des évaluations* [48], qui déduit la valeur de la fonction objectif d'un échantillon à partir des valeurs de ses parents. [17] explique qu'il existe deux méthodes pour combiner les évaluations coûteuses (appelées *contrôlées* dans ce contexte) et les évaluations approximatives. La première consiste à diviser la population en deux parties, une petite partie étant contrôlée et l'autre approximée (contrôle de l'évolution basé sur les *échantillons*). L'autre méthode consiste à contrôler ou à approximer l'ensemble de la population (contrôle basé sur la *population*). Dans les deux cas, il est nécessaire de déterminer le pourcentage d'individus contrôlés ou le nombre de générations approximées entre deux évaluations contrôlées. À notre connaissance, aucune valeur particulière ne se démarque au sein de la communauté, sauf pour les problèmes simples ou les fonctions de référence. Comme pour l'approche multifidélité, la combinaison de plusieurs modèles de substitution a donné de meilleurs résultats que l'utilisation d'un seul modèle. Récemment, les modèles d'approximation ont également été

adaptés à l'optimisation multiobjectif afin de prédire la dominance de Pareto des échantillons [29].

De nombreux travaux ont proposé de combiner des modèles multifidélité et d'approximation. Cependant, ils ont été critiqués pour avoir basé leurs comparaisons avec d'autres uniquement sur la qualité des échantillons obtenus ou le nombre d'évaluations coûteuses effectuées. Ils négligent de vérifier si le coût temporel des modèles d'approximation est inférieur à celui du modèle coûteux initial. Le plus souvent, le goulot d'étranglement est déplacé des évaluations coûteuses vers l'algorithme d'apprentissage lui-même. De plus, les techniques omettent également de rapporter cette information. Cela est dû à l'hypothèse sous-jacente selon laquelle les évaluations coûteuses des objectifs ou des violations de contraintes constituent le goulot d'étranglement. Et ignorer complètement le temps de calcul des modèles d'approximation peut conduire au développement de techniques trop lentes pour être utilisées dans la pratique. Pour combler ce manque, [5] propose une bibliothèque contenant quatre problèmes d'optimisation coûteux issus de différentes applications réelles afin de comparer les nombreuses stratégies basées sur des modèles d'approximation au sein de la communauté. Étant donné que la complexité de ces modèles augmente à chaque nouvelle évaluation de fonction, [54] conseille d'indiquer quand il est préférable d'utiliser une technique basée sur l'approximation plutôt que la fonction objectif coûteuse initiale.

Accélérer la convergence

Le concept d'amélioration de la convergence d'un algorithme n'est pas nouveau et n'est pas spécifique à l'optimisation coûteuse. Cependant, cet aspect nécessite beaucoup plus d'attention dans ce cas. Comme indiqué dans [38], les stratégies d'amélioration peuvent être mises en œuvre à n'importe quelle étape de l'évolution artificielle et sont résumées dans la figure 4.

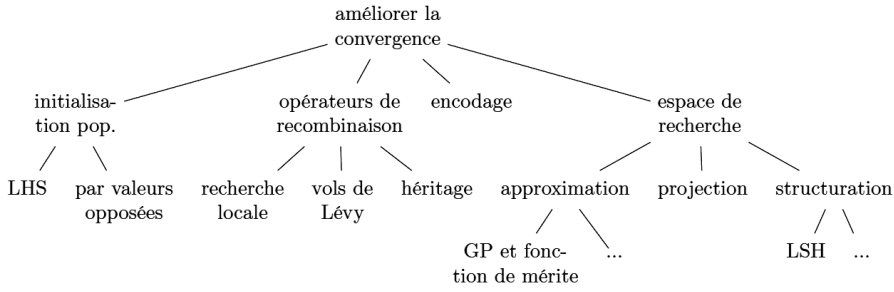


Fig. 4. Hiérarchie des stratégies pour améliorer la convergence.

Tout d'abord, la manière dont les individus sont codés (*encodage*) peut influencer la convergence, mais cela dépend entièrement du problème. À l'inverse, la construction de la *population initiale*, qui est indépendante du problème, a également été étudiée par le passé [21], [52]. Des techniques telles que l'échantillonnage par hypercube latin (*Latin hypercube sampling* – LHS), l'*initialisation par valeurs opposées* [31] et bien d'autres encore peuvent contribuer à garantir un certain degré de diversité lors de l'initialisation de la population, et ainsi favoriser une convergence plus rapide vers un optima. Des opérateurs de variation tels que la mutation et le croisement sont également utilisés pour améliorer la convergence, comme le conseil [57] en utilisant la recherche locale comme opérateur de mutation. Cependant, elle peut nécessiter de nombreuses évaluations, en particulier lorsque le voisinage de l'échantillon est très large. Cela peut poser problème dans un contexte d'optimisation coûteuse. Par conséquent, il est essentiel d'équilibrer correctement les évaluations classiques et celles dédiées à la recherche locale. Au lieu d'évaluer tous les voisins d'un échantillon et de garder le meilleur, il serait plus avantageux d'utiliser les vols de Lévy [47]. Il s'agit d'un mécanisme de marche aléatoire caractérisé par sa capacité à effectuer de grands sauts à des emplacements locaux avec une probabilité élevée. Cela permettrait d'obtenir des voisins plus diversifiées tout en réduisant le nombre d'évaluations.

La convergence de l'opérateur de croisement a également été étudiée par le passé, notamment grâce à l'héritage des évaluations (*fitness inheritance*) [42]. Cette technique est clairement identifiée comme une stratégie permettant de réduire le coût [24], [57]. La stratégie d'approximation discutée dans la section précédente diffère de l'héritage des évaluations car elle n'est liée à aucun opérateur génétique. Alors que l'approximation ne nécessite que le génotype d'un seul échantillon et le modèle appris pour prédire l'évaluation de l'objectif ou du respect des contraintes, la même prédiction dans le cas de l'héritage nécessite les génotypes et les évaluations de tous les parents utilisés par l'opérateur. Comme précédemment, les vols de Lévy ont également été utilisés dans les opérateurs de croisement mais aussi pour initialiser la population [23].

Comme mentionné dans [38], les opérateurs génétiques n'utilisent pas directement les informations sur l'espace de recherche. Pour remédier à cela, les auteurs ont proposé trois stratégies visant à accélérer la convergence. La première consiste à *approximer*, une fois encore, l'espace de recherche en une forme plus simple afin d'identifier plus facilement les optima dans le nouvel espace. Cette approximation a été réalisé grâce aux processus gaussiens (GP, *Gaussian process* [4]). L'avantage d'utiliser les GP réside dans leur capacité à prédire la valeur de la fonction objectif $\hat{y}$ d'un échantillon x' tout en disposant d'un intervalle de confiance $\hat{s}(x')$ correspondant à cette prédiction (figure 5).

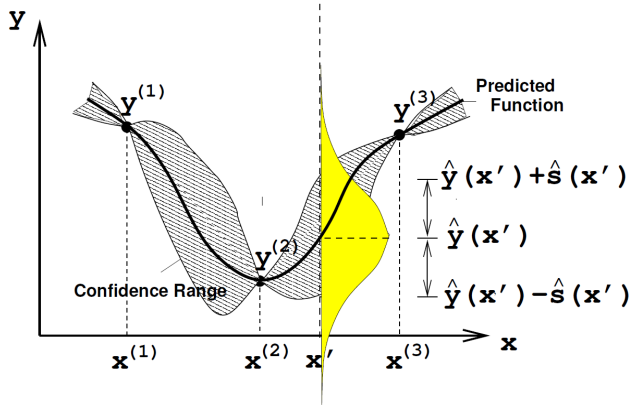


Fig. 5. Approximation basée sur les processus gaussiens [13].

Cependant, plutôt que d'utiliser simplement la valeur $\hat{y}(x')$ prédite par le PG comme évaluation approximative comme expliqué dans la section précédente, [10] suggère d'utiliser la limite inférieure de confiance d'une prédiction à l'aide d'une fonction de *mérite* $y_m(x) = \hat{y}(x) - \omega\hat{s}(x)$, où $\omega = 2$ pour une minimisation. L'idée est de donner du poids à la valeur minimale de l'intervalle de confiance $\hat{s}(x')$ de la prédiction $\hat{y}(x')$ d'un échantillon susceptible d'être un meilleur optimum que celui identifié jusqu'à présent $y^{(2)}$. Cela consiste à orienter la recherche vers des régions prometteuses mais moins explorées de l'espace de recherche. L'article [13] détaille ce principe et montre que cette fonction de mérite est un bon critère pour l'évolution artificielle utilisant des modèles d'approximation. [28], [36] appliquent par exemple la fonction de mérite au PG.

La seconde stratégie, telle que suggérée par [38], consiste à *projeter* l'espace de recherche initial vers un espace de dimension inférieure ou supérieure afin de faciliter la recherche d'optimum globaux. Au lieu de faire évoluer des solutions potentielles au problème à optimiser, [41] propose de faire évoluer des régions de solutions représentées par des hyperrectangles, chacun associé à un niveau de confiance dans l'intervalle $[0, 1]$ pour contenir l'optima global. Le défi de cette approche réside dans la définition d'opérateurs de variation appropriés. De plus, le génotype basé sur les régions induit un niveau supplémentaire de complexité qui peut ne pas être compensé par un gain de performance, un point qui n'est pas abordé par les auteurs. Une approche similaire utilisant des cellules de hachage (Locality sensitive hashing – LSH) [1] a été introduite par [39], [58] afin de réduire considérablement le temps nécessaire au calcul des distances entre les échantillons dans les problèmes d'optimisation multimodaux. L'idée est d'utiliser une famille de fonctions de hachage

pour structurer l'espace de recherche en cellules appelées *buckets* pouvant contenir plusieurs échantillons. Ces fonctions doivent être sélectionnées de manière à ce que les échantillons proches les uns des autres dans l'espace de recherche d'origine aient une probabilité forte d'être dans la même *bucket*. Cette approche est particulièrement utile lorsque l'espace de recherche est défini en extension, comme c'est souvent le cas dans l'optimisation basée sur des données. [20] a travaillé sur la définition d'opérateurs de variation dans de tels espaces de recherche. Le principal inconvénient de cette approche est l'identification des fonctions qui dépendent du problème. Ces fonctions pourraient toutefois être optimisées a priori avant le début du processus d'optimisation. Un autre avantage de cette technique est la possibilité de choisir la taille du nouvel espace de recherche car il s'agit d'un hyperparamètre de LSH. De plus, cette technique peut être utilisée pour approximer les individus dans la même cellule que ceux déjà évalués. C'est le principe utilisé par [53], mais uniquement pour une approximation locale de l'espace de recherche à l'aide d'un maillage de Voronoï au lieu de LSH. Les auteurs parviennent à réduire le coût d'évaluation à seulement cinq fois la taille du problème, mais uniquement sur des fonctions de tests et dans des espaces de dimension assez faible (entre 5 et 30). D'autres études ont également proposé de structurer l'espace de manière hiérarchique [55], [56].

La troisième stratégie, telle que décrite dans [38], était plus théorique et prospective. L'idée était de conserver à la fois l'espace de recherche initial et l'espace de recherche projeté, en appliquant l'évolution à un espace de recherche afin d'obtenir des informations pouvant être utilisées pour rechercher dans l'autre espace via un mapping et alterner entre les deux. Pour nous, cela ressemble à un transfert de connaissances d'un problème d'optimisation à un autre, un concept plus récemment connu sous le nom d'*apprentissage par transfert*, que nous aborderons dans la section suivante.

Augmenter les ressources

La dernière catégorie permettant de réduire le coût d'un problème d'optimisation est l'augmentation des ressources disponibles. Il en existe de deux types : les nœuds de calculs et les problèmes d'optimisation. S'il est logique d'envisager le premier type, réduire le coût de l'optimisation d'un problème en ajoutant au moins un autre problème à cette optimisation est une stratégie beaucoup moins évidente.

Les métaheuristiques à base de population sont intrinsèquement parallèles car elles manipulent un ensemble d'échantillons représentés par un ensemble de paramètres. De nombreux travaux ont déjà étudié et proposé plusieurs stratégies de distribution pour déployer ces échantillons et leurs paramètres sur des nœuds de calcul. [14] a proposé une hiérarchie d'algorithmes d'évolution

artificielle distribués (illustrée dans la figure 6 contenant deux méthodes principales pour répartir une population d'échantillons sur plusieurs nœuds de calcul. Soit un échantillon ou un groupe d'échantillons est affecté à un nœud (distribution orientée *population*), soit les échantillons sont divisés en plusieurs groupes de sous-échantillons en fonction de leurs paramètres ($x_1, \ldots, x_m$) pour former plusieurs sous-populations ($x^1 = (x_1, \ldots, x_k bnh)$, $x^2 = (x_{k+1}, \ldots, x_l)$, $x^3 = \ldots$), chacune étant affectée à un nœud de calcul (distribution orientée *dimension*).

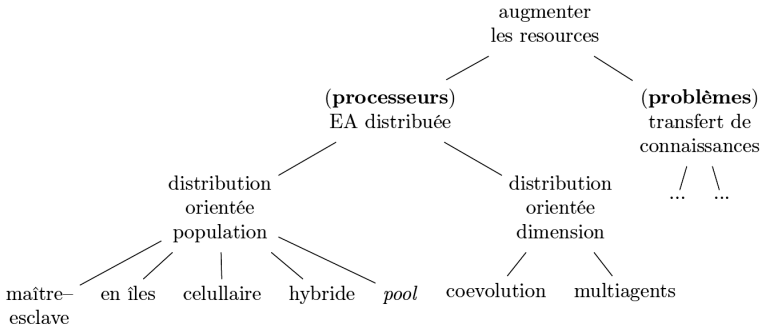


Fig. 6. Modification de la hiérarchie proposée par [14] en ajoutant d'autres problèmes à optimiser.

Il existe un certain nombre de stratégies disponibles pour chaque méthode de distribution, notamment la distribution *maître-esclave*, en *îles*, *cellulaire*, la *coévolution*, etc. Une présentation détaillée de chaque stratégie dépasse le cadre du présent article mais elle est particulièrement bien détaillée dans [14], et ces stratégies sont comparées entre elles sur plusieurs critères. Il est important de noter que réduire le temps d'exécution en augmentant le nombre de nœuds peut entraîner d'autres coûts. Des études ont montré que la stratégie maître-esclave peut être très efficace, à condition que le temps d'évaluation d'un échantillon soit nettement supérieur au temps de transmission des informations entre les nœuds. La raison en est le coût relativement élevé de la communication : le maître communique en effet en permanence avec les esclaves, leur envoyant les échantillons à évaluer et recevant en retour les valeurs de leurs fonctions objectif. De plus, le nombre d'esclaves ne peut pas être augmenté de manière excessive, car cela risquerait de saturer le maître. De plus, la plupart des stratégies maître-esclave sont synchrones, ce qui signifie que le maître attend que tous les esclaves aient terminé leurs évaluations respectives. Dans les implémentations où le temps d'évaluation varie considérablement d'un échantillon à

l'autre (par exemple dans la stratégie multifidélité), un temps d'inactivité important peut être introduit, réduisant ainsi l'avantage de cette distribution. Dans ce cas, il est nécessaire de transformer la technique d'évolution artificielle en sa variante asynchrone [46]. Le coût de la communication pourrait également être un facteur important dans la stratégie de distribution cellulaire, en particulier lorsque la taille de la population et le nombre de voisins associés à chaque échantillon sont élevés. Il en va de même pour la *coévolution* qui nécessite de nombreuses collaborations entre sous-populations (impliquant l'échange de nombreuses informations) pour l'évaluation.

Les auteurs de [24] notent que la plupart des recherches récentes utilisent le calcul haute performance pour résoudre des problèmes concrets mais qu'il n'existe que peu d'études théoriques, probablement parce que les fonctions de référence disponibles n'incluent pas la notion d'optimisation coûteuse. Cependant, même si les ordinateurs sont de plus en plus puissants, le simple fait d'ajouter plus de processeurs au processus d'optimisation ne suffit pas à résoudre tous les problèmes coûteux auxquels nous sommes confrontés. Premièrement, cela peut être dû au fait que le facteur de gain n'est pas suffisant. Deuxièmement, le fait d'avoir plus de processeurs entraîne d'autres coûts, tels que le coût de la communication entre les processus ou le temps d'inactivité. Cela peut affecter l'efficacité de la stratégie.

Comme indiqué précédemment, il semble logique d'augmenter le nombre de processeurs afin de réduire le temps d'exécution. Cependant, il est plus difficile de concevoir la réduction du coût en optimisant au moins un autre problème afin de l'exploiter. On parle alors d'optimisation *multitâche*, qui pourrait correspondre à un sixième M dans le défi 5M de [57]. Dans ce contexte, une stratégie potentielle pourrait consister à optimiser un problème moins coûteux en même temps que l'optimisation d'un problème plus coûteux, ou à optimiser deux problèmes coûteux différents. [15] sont les premiers à proposer l'optimisation simultanée de plusieurs problèmes avec la même population d'échantillons. Les encodages des individus pour chaque tâche sont regroupés en un encodage unifié pour toutes les tâches. La taille de cet encodage correspond à la plus grande taille des encodages de toutes les tâches. Par conséquent, certains paramètres d'un échantillon unifié peuvent ne pas être utilisés pour une ou plusieurs tâches. Les échantillons unifiés sont ensuite divisés en groupes au sein d'une même population selon le même principe que la coévolution à population unique proposée par [43], de sorte que chaque groupe se concentre sur une tâche spécifique. Sur cette base, le transfert de connaissances entre les groupes est réalisé à l'aide d'opérateurs génétiques spécifiques. L'optimisation multitâche s'est avérée plus efficace que l'optimisation monotâche en termes de vitesse de convergence sur une série de problèmes pratiques (problème du voyageur de commerce, problème de planification d'atelier, problème de routage de véhicules à capacité limitée,

etc.) et son efficacité a été analysée théoriquement par [26]. L'article [59] fournit un aperçu de l'état de l'art dans ce domaine de recherche récent. Certaines études, telles que [11], utilisent la transformation des espaces de recherche de tâches pour les unifier et ainsi pouvoir localiser les optima des tâches les moins coûteuses et transférer ces informations vers les tâches les plus coûteuses afin d'améliorer la convergence de l'algorithme. D'autres, comme dans [27], utilisent plusieurs modèles d'apprentissage (*ensemble learning*), mais une seule tâche pour approximer l'ensemble de l'espace de recherche combiné à une autre tâche pour approximer un espace de recherche plus local. Chaque solution optimale pour chaque tâche est évaluée à l'aide d'une évaluation coûteuse, puis utilisée pour mettre à jour les deux modèles d'approximation. Cette technique permet d'améliorer la convergence de l'algorithme. Comme mentionné dans la section précédente, le coût de la construction de modèles d'approximation visant à réduire le coût global de l'optimisation n'est toujours pas suffisamment pris en compte, comme le souligne [54], qui ne considère que le temps d'optimisation. Il serait utile de proposer un point précis à partir duquel le modèle de distribution devient une option viable, qu'il soit basé sur le nombre de processeurs ou sur des sous-populations.

Conclusion & Perspectives

Dans cet article d'opinion, nous caractérisons tout d'abord ce qu'est l'optimisation coûteuse à l'aide de la formule donnée dans [24] et tentons de fournir une définition qui n'est pas si naturelle. Nous montrons ensuite que la proportion d'articles scientifiques consacrés à l'optimisation coûteuse a considérablement augmenté au sein de la communauté. Nous résumons et commentons ensuite les différentes stratégies proposées dans la littérature, en les regroupant en trois catégories selon la formule susmentionnée : réduire le coût d'évaluation des objectifs et des contraintes, accélérer la convergence de l'algorithme d'optimisation et augmenter les ressources disponibles. Cependant, nous montrons au cours de cet article que la communauté se concentre principalement sur la réduction du temps d'exécution, alors que les stratégies proposées entraînent souvent d'autres types de coûts qui ne sont pas pris en compte lors de la comparaison des stratégies. Sur la base de ces considérations, nous présentons dans les paragraphes suivants plusieurs axes de recherche qui, selon nous, devraient contribuer à améliorer l'optimisation coûteuse, résumés dans la figure 7.

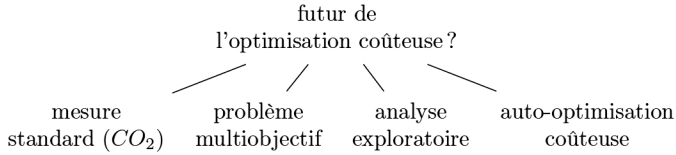


Fig. 7. Que pourrait être le futur de l'optimisation coûteuse ?

Une mesure standard pour le coût.

Le premier axe de recherche que nous suggérons consiste à se concentrer sur la caractérisation ou la normalisation des différents coûts induits par les stratégies utilisées dans l'optimisation coûteuse. Nous pensons qu'il est important de prendre en compte ces coûts dans la formule (2). Par exemple, [53] compare les stratégies de résolution d'un problème coûteux en termes de nombre d'évaluations en fonction de la dimension. Les auteurs proposent une stratégie qui permet de résoudre des fonctions de test en utilisant un nombre d'évaluations égal à seulement cinq fois la dimension du problème, alors que les meilleures stratégies précédentes en nécessitaient onze fois plus. Une fois encore, seul le temps de calcul est pris en compte. Tout comme il est possible de mesurer la qualité d'un problème d'optimisation multiobjectif à l'aide de l'hypervolume, ou la qualité d'un problème d'optimisation multimodale à l'aide de la proportion d'optima identifiés, il serait intéressant de proposer une ou plusieurs mesures de qualité pour un problème d'évolution artificielle combinant la qualité des résultats et les différents types de coûts impliqués (temps, communications, temps d'inactivité, etc.). Cela faciliterait la comparaison des stratégies existantes. À cet égard, nous pensons que les travaux menés par [35] en 2011 allaient dans la bonne direction. Ils ont proposé une technique d'optimisation biobjectif visant à réduire à la fois le temps de planification et la consommation d'énergie. Le principe consistait à ajuster dynamiquement la tension d'alimentation des processeurs au détriment de la fréquence d'horloge. Malheureusement, même aujourd'hui, peu d'études prennent en compte cet aspect énergétique lorsqu'elles comparent différentes techniques. Une façon de remédier à cela consiste à utiliser la quantité d'émissions de CO_2 comme indicateur, comme le propose [45].

Transformer le coûteux en multiobjectif ?

Comme indiqué précédemment, la résolution d'un problème d'optimisation par adaptation de différentes stratégies peut entraîner des coûts supplémentaires. Par conséquent, afin de minimiser les coûts, il serait avantageux de réduire le temps d'exécution, le nombre de communications, la consommation d'énergie et le temps d'apprentissage ou l'utilisation des données des modèles

d'approximation, tout en maximisant le nombre de cœurs de calcul et de populations utilisés, et éventuellement d'autres critères. Cependant, il est difficile de classer ces quantités car il n'existe pas d'ordre naturel entre elles tant qu'un système de mesure unifié n'a pas été établi pour les problèmes d'optimisation. Suivant l'approche décrite dans [35], cela implique de traiter un problème d'optimisation comme un problème multicritère, où les critères sont a priori antagonistes, ce qui donne lieu à un problème multiobjectif. Selon cette idée, un problème de minimisation dans lequel f est l'objectif coûteux serait converti en un problème multiobjectif auquel des objectifs supplémentaires seraient ajoutés, en fonction des stratégies employées pour réduire le coût de f. Cela pourrait être formalisé comme suit :

$$\arg\min_{x} \left\{ f(x), CO_2(f(x)), \frac{1}{\text{processeurs}}, \frac{1}{\text{populations}}, \dots \right\} \tag{3}$$

Dans [3], les auteurs proposent l'algorithme SMS-EMOA, qui est dédié à la résolution de problèmes multiobjectif en intégrant la maximisation de l'hypervolume dans l'opérateur de sélection. Pourrions-nous construire un nouvel algorithme dédié à l'optimisation coûteuse basé sur le principe proposé par ces auteurs, en combinant l'objectif initial coûteux avec les différents coûts impliqués dans l'optimisation du problème et les stratégies déployées ? Alternativement, pourrions-nous simplement utiliser cet algorithme en caractérisant tous les coûts impliqués dans les stratégies adoptées et en calculant l'hypervolume de ces coûts ?

Analyse de l'espace de recherche.

Une deuxième piste de recherche que nous jugeons prometteuse consisterait à développer une expertise sur les stratégies de réduction des coûts afin de choisir la stratégie ou la combinaison de stratégies la mieux adaptée au problème à résoudre. En d'autres termes, l'algorithme décrit dans le paragraphe précédent peut-il adapter ses stratégies en fonction des types de coûts mesurés afin d'améliorer ses performances ? Pour ce faire, dans le cas de problèmes classiques, [34] propose de sélectionner la technique d'optimisation la mieux adaptée à la résolution d'un problème donné, en extrayant des propriétés spécifiques à faible coût et en identifiant l'algorithme le plus efficace pour cette combinaison de propriétés. Ce processus est connu sous le nom d'*analyse exploratoire* de l'espace de recherche. Les propriétés identifiées sont assez générales et ne tiennent pas compte de la notion de coût. Ce n'est que dans [50] que cette notion est incluse dans les directives de choix mais uniquement pour savoir si le problème est coûteux ou pas, et en proposant uniquement la stratégie d'approximation, la plus répandue. [33] améliore l'approche exploratoire en extrayant automatiquement du problème des propriétés de bas niveau, et en les reliant aux propriétés de plus haut niveau

discutées précédemment. Le principal inconvénient est que le calcul de ces propriétés nécessite l'évaluation d'un grand nombre d'échantillons, ce qui est coûteux dans notre cas. De plus, [51] montre que certaines de ces propriétés sont trop coûteuses même pour guider la résolution de problèmes de grande taille. Pour ce type d'optimisation, une petite proportion du budget est réservée à la séparation des paramètres en différents groupes, afin que le reste du budget soit ensuite alloué à l'optimisation. Est ce que ce mécanisme peut aussi être adopté dans le cas coûteux ? Tenter de répondre à ces questions nous semble être un domaine de recherche prometteur qui n'a pas encore été exploré dans le contexte de l'optimisation coûteuse.

Auto-hybridation des stratégies.

De nombreuses stratégies ont été proposées pour traiter l'optimisation coûteuse, depuis le processus d'initialisation jusqu'à la condition d'arrêt, même si cette dernière n'a pas été suffisamment étudiée selon [38]. Pour la dernière orientation de recherche, nous recommandons d'étudier la combinaison automatique de toutes les stratégies précédentes exactement comme l'automatisation de l'apprentissage automatique (AutoML) s'est développé dans le domaine de l'apprentissage machine [2], [25]. Pourquoi ne pas envisager un processus d'automatisation de l'optimisation coûteuse capable d'optimiser des problèmes coûteux en tenant compte de tous les coûts induits par cette optimisation et les stratégies adoptées, et cela sans assistance humaine ?

Références

[1] A. Andoni & P. Indyk. 2008. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. *Commun. ACM*. https://doi.org/10.1145/1327452.1327494.

[2] M. Baratchi, C. Wang, S. Limmer et al. 2024. Automated Machine Learning: Past, Present and Future. *Artificial Intelligence Review*. https://doi.org/10.1007/s10462-024-10726-1.

[3] N. Beume, B. Naujoks &t M. Emmerich. 2007. SMS-EMOA: Multiobjective selection based on dominated hypervolume. *European Journal of Operational Research*. https://doi.org/10.1016/j.ejor.2006.08.008.

[4] M. Binois & N. Wycoff. 2022. A Survey on High-dimensional Gaussian Process Modeling with Application to Bayesian Optimization. *ACM Trans. Evol. Learn. Optim.* https://doi.org/10.1145/3545611.

[5] L. Bliek, A. Guijt, R. Karlsson et al. 2023. Benchmarking surrogate-based optimisation algorithms on expensive black-box functions. *Applied Soft Computing*. https://doi.org/10.1016/j.asoc.2023.110744.

[6] K. van der Blom, T. M. Deist, V. Volz et al. 2021. Identifying Properties of Real-World Optimisation Problems through a Questionnaire. https://doi.org/10.48550/arXiv.2011.05547.

[7] F. Campelo & C. Aranha. 2023. Lessons from the Evolutionary Computation Bestiary. *Artificial Life*. https://doi.org/10.1162/artl_a_00402.

[8] F. Campelo & C. Aranha. 2023. EC Bestiary: A bestiary of evolutionary, swarm and other metaphor-based algorithms. https://doi.org/10.5281/zenodo.1293352.

[9] J. Del Ser, E. Osaba, D. Molina et al. 2019. Bio-inspired computation: Where we stand and what's next. *Swarm and Evolutionary Computation*. https://doi.org/10.1016/j.swevo.2019.04.008.

[10] J. E. Dennis & V. Torczon. 1995. Managing approximation models in optimization. *Multidisciplinary Design Optimization: State-of-the-Art*.

[11] J. Ding, C. Yang, Y. Jin & T. Chai. 2019. Generalized Multitasking for Evolutionary Optimization of Expensive Problems. *IEEE Transactions on Evolutionary Computation*. https://doi.org/10.1109/TEVC.2017.2785351.

[12] A. E. Eiben & J. E. Smith. 2015. Hybridisation with Other Techniques: Memetic Algorithms. In *Introduction to Evolutionary Computing*. https://doi.org/10.1007/978-3-662-44874-8_10.

[13] M. T. M. Emmerich, K. C. Giannakoglou & B. Naujoks. 2006. Single- and multiobjective evolutionary optimization assisted by Gaussian random field metamodels. *IEEE Transactions on Evolutionary Computation*. https://doi.org/10.1109/TEVC.2005.859463.

[14] Y.-J. Gong, W.-N. Chen, Z.-H. Zhan et al. 2015. Distributed evolutionary algorithms and their models: A survey of the state-of-the-art. *Applied Soft Computing*. https://doi.org/10.1016/j.asoc.2015.04.061.

[15] A. Gupta, Y.-S. Ong & L. Feng. 2016. Multifactorial Evolution: Toward Evolutionary Multitasking. *IEEE Transactions on Evolutionary Computation*. https://doi.org/10.1109/TEVC.2015.2458037.

[16] C. He, Y. Zhang, D. Gong & X. Ji. 2023. A review of surrogate-assisted evolutionary algorithms for expensive optimization problems. *Expert Systems with Applications*. https://doi.org/10.1016/j.eswa.2022.119495.

[17] Y. Jin, M. Olhofer & B. Sendhoff. 2001. Managing approximate models in evolutionary aerodynamic design optimization. In (CEC). https://doi.org/10.1109/CEC.2001.934445.

[18] Y. Jin & B. Sendhoff. 2009. A systems approach to evolutionary multiobjective structural optimization and beyond. *IEEE Computational Intelligence Magazine*. https://doi.org/10.1109/MCI.2009.933094.

[19] Y. Jin, H. Wang, T. Chugh et al. 2019. Data-Driven Evolutionary Optimization: An Overview and Case Studies. *IEEE Transactions on Evolutionary Computation*. https://doi.org/10.1109/TEVC.2018.2869001.

[20] C. G. Johnson. 2012. Search-based evolutionary operators for extensionally-defined search spaces: Applications to image search. In *IEEE Congress on Evolutionary Computation*. https://doi.org/10.1109/CEC.2012.6256551.

[21] B. Kazimipour, X. Li & A. K. Qin. 2014. A review of population initialization techniques for evolutionary algorithms. In (CEC). https://doi.org/10.1109/CEC.2014.6900618.

[22] G. Li, Q. Zhang, Q. Lin & W. Gao. 2022. A Three-Level Radial Basis Function Method for Expensive Optimization. *IEEE Transactions on Cybernetics*. https://doi.org/10.1109/TCYB.2021.3061420.

[23] J. Li, Q. An, H. Lei et al. 2022. Survey of Lévy Flight-Based Metaheuristics for Optimization. *Mathematics*. https://doi.org/10.3390/math10152785.

[24] J. Li, Z. Zhan & J. Zhang. 2022. Evolutionary Computation for Expensive Optimization: A Survey. *Machine Intelligence Research*. https://doi.org/10.1007/s11633-022-1317-4.

[25] N. Li, L. Ma, T. Xing et al. 2023. Automatic design of machine learning via evolutionary computation: A survey. *Applied Soft Computing*. https://doi.org/10.1016/j.asoc.2023.110412.

[26] Y. Lian, Z. Huang, Y. Zhou & Z. Chen. 2019. Improve Theoretical Upper Bound of Jumpk Function by Evolutionary Multitasking. In *Proceedings of the 2019 3rd High Performance Computing and Cluster Technologies Conference*. https://doi.org/10.1145/3341069.3342982.

[27] P. Liao, C. Sun, G. Zhang & Y. Jin. 2020. Multi-surrogate multi-tasking optimization of expensive problems. *Knowledge-Based Systems*. https://doi.org/10.1016/j.knosys.2020.106262.

[28] B. Liu, Q. Zhang & G. G. E. Gielen. 2014. A Gaussian Process Surrogate Model Assisted Evolutionary Algorithm for Medium Scale Expensive Optimization Problems. *IEEE Transactions on Evolutionary Computation*. https://doi.org/10.1109/TEVC.2013.2248012.

[29] Y. Lu, B. Li & A. Zhou. 2024. Are you concerned about limited function evaluations: data-augmented pareto set learning for expensive multi-objective optimization. In (AAAI'24/IAAI'24/EAAI'24). https://doi.org/10.1609/aaai.v38i13.29331.

[30] W. Luo, Y. Qiao, X. Lin et al. 2019. Many-Modal Optimization by Difficulty-Based Cooperative Co-evolution. In *IEEE Symposium Series on Computational Intelligence (SSCI)*. https://doi.org/10.1109/SSCI44817.2019.9003005.

[31] S. Mahdavi, S. Rahnamayan & K. Deb. 2018. Opposition based learning: A literature review. *Swarm and Evolutionary Computation*. https://doi.org/10.1016/j.swevo.2017.09.010.

[32] S. Mahdavi, M. Shiri & S. Rahnamayan. 2015. Metaheuristics in large-scale global continues optimization: A survey. *Information Sciences*. https://doi.org/10.1016/j.ins.2014.10.042.

[33] O. Mersmann, B. Bischl, H. Trautmann et al. 2011. Exploratory Landscape Analysis. In (GECCO). https://doi.org/10.1145/2001576.2001690.

[34] O. Mersmann, M. Preuss & H. Trautmann. 2010. Benchmarking Evolutionary Algorithms: Towards Exploratory Landscape Analysis. In (PPSN). https://doi.org/10.1007/978-3-642-15844-5_8.

[35] M. Mezmaz, N. Melab, Y. Kessaci et al. 2011. A parallel bi-objective hybrid metaheuristic for energy-aware scheduling for cloud computing systems. *Journal of Parallel and Distributed Computing*. https://doi.org/10.1016/j.jpdc.2011.04.007.

[36] D. Pallez, J. Gardès & C. Pasquier. 2017. Prediction of miRNA-disease Associations using an Evolutionary Tuned Latent Semantic Analysis. *Scientific Reports*. https://doi.org/10.1038/s41598-017-10065-y.

[37] V. Pareto. 1964. *Cours d'économie politique. Œuvres complètes publiées sous la direction de Giovanni Busino. Tomes 1 et 2 en un volume*. https://doi.org/10.3917/droz.paret.1964.01.

[38] Y. Pei & H. Takagi. 2011. A survey on accelerating evolutionary computation approaches. In *International Conference of Soft Computing and Pattern Recognition (SoCPaR)*. https://doi.org/10.1109/SoCPaR.2011.6089140.

[39] R. Pighetti, D. Pallez & F. Precioso. 2015. Improving SVM Training Sample Selection Using Multi-Objective Evolutionary Algorithm and LSH. In (SSCI). https://doi.org/10.1109/SSCI.2015.197.

[40] M. Preuss. 2015. *Multimodal Optimization by Means of Evolutionary Algorithms*. https://doi.org/10.1007/978-3-319-07407-8.

[41] S. Puechmorel & D. Delahaye. 2014. Order Statistics and Region-Based Evolutionary Computation. *Journal of Global Optimization*. https://doi.org/10.1007

/s10898-013-0079-5.

[42] M. Reyes-Sierra & C. A. C. Coello. 2005. A study of fitness inheritance and approximation techniques for multi-objective particle swarm optimization. In (CEC). https://doi.org/10.1109/CEC.2005.1554668.

[43] E. Sanchez, G. Squillero & A. Tonda. 2011. Group evolution: Emerging synergy through a coordinated effort. In (CEC). https://doi.org/10.1109/CEC.2011.5949951.

[44] L. V. Santana-Quintero, A. A. Montaño & C. A. C. Coello. 2010. A Review of Techniques for Handling Expensive Functions in Evolutionary Multi-Objective Optimization. In *Computational Intelligence in Expensive Optimization Problems*. https://doi.org/10.1007/978-3-642-10701-6_2.

[45] G. Saraiva, M. Vasconcelos, S. Mazzini Bruschi et al. 2025. Estimating CO_2 emissions of distributed applications and platforms with SimGrid/Batsim. https://doi.org/10.48550/arXiv.2508.13693.

[46] E. O. Scott & K. A. De Jong. 2015. Understanding Simple Asynchronous Evolutionary Algorithms. In *Proceedings of the 2015 ACM Conference on Foundations of Genetic Algorithms XIII* (FOGA '15). https://doi.org/10.1145/2725494.2725509.

[47] H. Sharma, S. S. Jadon, J. C. Bansal & K. V. Arya. 2013. Lèvy Flight Based Local Search in Differential Evolution. In *Swarm, Evolutionary, and Memetic Computing*. https://doi.org/10.1007/978-3-319-03753-0_23.

[48] R. E. Smith, B. A. Dike & S. A. Stegmann. 1995. Fitness inheritance in genetic algorithms. In *Symposium on Applied Computing*. https://doi.org/10.1145/315891.316014.

[49] K. Sörensen. 2015. Metaheuristics – the metaphor exposed. *International Transactions in Operational Research*. https://doi.org/https://doi.org/10.1111/itor.12001.

[50] J. Stork, A. E. Eiben & T. Bartz-Beielstein. 2022. A New Taxonomy of Global Optimization Algorithms. *Natural Computing: An International Journal*. https://doi.org/10.1007/s11047-020-09820-4.

[51] R. Tanabe. 2021. Towards Exploratory Landscape Analysis for Large-Scale Optimization: A Dimensionality Reduction Framework. In *Proceedings of the Genetic and Evolutionary Computation Conference*. https://doi.org/10.1145/3449639.3459300.

[52] A. Tharwat & W. Schenck. 2021. Population initialization techniques for evolutionary algorithms for single-objective constrained optimization problems: Deterministic vs. stochastic techniques. *Swarm and Evolutionary Computation*. https://doi.org/10.1016/j.swevo.2021.100952.

[53] H. Tong, C. Huang, J. Liu & X. Yao. 2019. Voronoi-based Efficient Surrogate-assisted Evolutionary Algorithm for Very Expensive Problems. In (CEC). https://doi.org/10.1109/CEC.2019.8789910.

[54] V. Volz & B. Naujoks. 2019. On Benchmarking Surrogate-Assisted Evolutionary Algorithms. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. https://doi.org/10.1145/3319619.3326866.

[55] L. Wang, R. Fonseca & Y. Tian. 2020. Learning Search Space Partition for Black-Box Optimization Using Monte Carlo Tree Search. In *Proceedings of the 34th International Conference on Neural Information Processing Systems* (NIPS'20). https://doi.org/10.5555/3495724.3497361.

[56] H. Xia, C. Li, S. Zeng et al. 2022. Learning to Search Promising Regions by a Monte-Carlo Tree Model. In (CEC). https://doi.org/10.1109/CEC55065.2022.9870281.

[57] Z. Zhan, L. Shi, K. Chen Tan & J. Zhang. 2022. A survey on evolutionary computation for complex continuous optimization. *Artificial Intelligence Review*. https://doi.org/10.1007/s10462-021-10042-y.

[58] Y.-H. Zhang, Y.-J. Gong, H.-X. Zhang et al. 2017. Toward Fast Niching Evolutionary Algorithms: A Locality Sensitive Hashing-Based Approach. *IEEE Transactions on Evolutionary Computation*. https://doi.org/10.1109/TEVC.2016.2604362.

[59] H. Zhao, X. Ning, X. Liu et al. 2023. What makes evolutionary multi-task optimization better: A comprehensive survey. *Applied Soft Computing*. https://doi.org/10.1016/j.asoc.2023.110545.