Un design évolutif pour des solutions révolutionnaires

saidboudjelda

2024-11-16

Les algorithmes exacts (déterministes) jouent un rôle fondamental dans la résolution de nombreux problèmes dans divers domaines, qu’il s’agisse de tri de données, de recherche de chemins optimaux, ou encore de résolution d’équations complexes. Cependant, face à des problèmes dits ``NP-difficiles'' \footnote{En informatique théorique, le terme "NP-difficiles" (ou NP-hard en anglais) désigne une classe de problèmes qui sont au moins aussi difficiles à résoudre que les problèmes de la classe NP (Non-deterministic Polynomial time); Example : Le célèbre problème du voyageur de commerce (TSP, Travelling Salesman Problem) en version d’optimisation qui consiste à trouver le chemin optimal parmi plusieurs villes est un défi immense quand le nombre de villes augmente} ou à de vastes espaces de conception, ils révèlent rapidement leurs limites. Ces algorithmes, souvent déterministes, sont conçus pour parcourir de manière exhaustive toutes les solutions possibles pour garantir de trouver l’optimum, ce qui rend leur utilisation peu pratique, voire impossible, pour des problèmes de grande dimension ou en constante évolution.

Les algorithmes approximatifs ou méta-heuristiques, quant à eux, apportent une approche différente pour obtenir des solutions proches de l’optimum dites quasi-optimales dans des délais raisonnables, ce qui est souvent suffisant pour les applications pratiques.

Une des classes des méta-heuristiques est celle des algorithmes évolutionnaires, souvent assimilés aux ``algorithmes génétiques'' dont l’approche est inspirée des mécanismes de l’évolution naturelle. En simulant des processus tels que la sélection, le croisement et la mutation, les algorithmes évolutionnaires génèrent progressivement des solutions optimales ou quasi-optimales contrairement aux algorithmes exactes qui peuvent être bloqués par des solutions locales ou des configurations complexes.

Au-delà de la résolution de problèmes spécifiques, les algorithmes évolutionnaires se distinguent par leur efficacité dans l’exploration d’espaces de recherche vastes et complexes, surtout lorsque les dimensions du problème augmentent et entraînent une prolifération de configurations possibles. Ces algorithmes apportent une dynamique adaptative et flexible, élargissant considérablement le champ de recherche en pénétrant des zones inexplorées et souvent inaccessibles aux méthodes classiques ou à l’intuition humaine. Cette capacité d’exploration, amplifiée par la composante aléatoire, ouvre la voie à la découverte de solutions innovantes, inédites et potentiellement optimisées, qui auraient autrement échappé à toute détection.

Par conséquent, nous utilisons les algorithmes évolutionnaires pour concevoir de nouveaux produits ou systèmes de manière similaire à la méthodes MVP (Minimum Viable Product).

Imaginez les algorithmes évolutionnaires comme un processus de développement en plusieurs générations : au lieu de créer un produit final parfait dès le début, ils explorent diverses versions ``prototypes'' (solutions) à travers des itérations rapides. Chaque version est testée, puis les meilleures configurations sont sélectionnées, ajustées et combinées pour former une nouvelle génération améliorée. De la même façon que le MVP évolue par étapes en fonction du retour des utilisateurs, les algorithmes évolutionnaires évaluent, adaptent et optimisent chaque itération pour s’approcher de la solution optimale.

Évidemment, au contraire du MVP, les algorithmes évolutionnaires ne sont pas tenus de produire une solution immédiatement ``viable'' ou utilisable à chaque itération. Ils évoluent de manière itérative afin d’explorer l’espace de recherche pour converger progressivement vers des solutions optimales. Dans ce contexte, on utilise un critère de fitness pour évaluer et comparer les solutions, permettant de sélectionner et d’améliorer les meilleures configurations à chaque génération, même si elles ne sont pas directement applicables dans l’immédiat.