Exemple de logiciel d'OEP sans paramètres de réglage

Tribes, ou la coopération de tribus

Maurice Clerc
France Télécom R&D, Annecy
maurice.clerc@writeme.com

Le tribalisme : c'était un choix. Je commençais à penser que c'était le bon. En tout cas le seul défendable.

Francis Valéry
La cinquième tribu

Vers un programme ultime

Au commencement était l'unique. Tel pourrait être le début d'un processus d'Optimisation par Essaim Particulaire (OEP)[1] totalement autonome, dans la mesure où il doit être à même de trouver une solution en n'ayant, au début, qu'une seule particule, à charge pour lui d'en ajouter ou supprimer à bon escient. Jusqu'ici, même pour les versions d'OEP adaptatives, vous devez non seulement décrire le problème à résoudre, mais également donner des indications sur la *manière* de le faire, avec des instructions du type « Commence avec vingt particules », « Utilise un voisinage circulaire de taille trois » ou « Pondère la vitesse par un coefficient décroissant au cours du temps selon la loi suivante … ».

En OEP classique (OEPc), la description du problème consiste à délimiter l'espace de recherche (dans les cas simples en précisant pour chaque dimension l'intervalle des valeurs admissibles), à indiquer comment évaluer en chaque point de cet espace la fonction à minimiser et, enfin, à préciser quelle est l'erreur maximale que vous admettez. Egalement, mais à titre de précaution, il est prudent de poser un garde-fou, soit un nombre maximum d'évaluations, soit un temps de calcul maximum. Ceci reste évidemment nécessaire : le programme ne devinera ni votre problème, ni votre exigence de précision ! Mais cela devrait aussi être suffisant. Autrement dit, la méthode doit incorporer des règles définissant comment, à chaque instant, la structure de l'essaim doit être modifiée et aussi comment une particule donnée doit se comporter, le tout en fonction des informations progressivement recueillies durant le processus lui-même.

Naturellement, ces règles sont encore des manières détournées de donner des instructions de fonctionnement au programme. La différence essentielle tient au fait que vous pouvez les ignorer complètement si elles sont suffisamment robustes et générales pour satisfaire tous ses besoins pratiques. Il serait facile de coder « en dur » une règle comme « Utilise toujours vingt particules », mais l'expérience montre qu'avec certains types de problèmes, les résultats sont extrêmement mauvais, même s'ils sont très bons avec d'autres : une telle règle n'est pas robuste. Or, justement, ce que nous voulons, et nous nous plaçons ici plutôt du point de vue d'un ingénieur, ce sont des résultats peut-être pas toujours excellents, mais, au moins, jamais désastreux, d'autant qu'une stratégie d'arrêt/reprise judicieuse peut en améliorer de toutes façons la qualité. Ce que l'on gagne en simplicité d'utilisation devrait logiquement être parfois perdu en efficacité. Il est en effet assez normal, pour un problème donné, qu'un programme devant trouver tout seul ses propres paramètres, et ce au cours d'une seule exécution, ait parfois des résultats inférieurs à un autre dont les paramètres ont été longuement peaufinés à l'aide de nombreux tests. D'un autre côté, si l'on veut faire des comparaisons honnêtes, par exemple en nombre d'évaluations de la fonction à minimiser avant de trouver une solution, il faudrait justement inclure ces tests eux-mêmes.

[1] Les principes de bases de l'OEP sont supposés connus, même si quelques rappels sont faits dans ce texte.

La meilleure méthode pour prouver qu'un tel programme est possible est d'en présenter un. Nous allons donc décrire le programme Tribes et montrer qu'il répond plutôt bien à notre définition d'une boite noire facilement utilisable et aux performances satisfaisantes, même si, bien sûr, des améliorations sont possibles, en particulier concernant les problèmes à granularité non nulle (typiquement avec certaines variables entières). La description faite ci-dessous comporte des stratégies d'adaptation structurelles, régulant les modifications de la taille de l'essaim et des relations d'informations entre les particules, et des stratégies de déplacement, indiquant comment une particule donnée doit changer de position.

Nous supposerons d'abord que l'espace de recherche est muni d'une distance. Cette hypothèse permet de définir une stratégie de déplacement efficace, fondée sur des distributions de probabilité hypersphériques. Nous verrons ensuite comment, tout en conservant les stratégies d'adaptations, il est possible de définir d'autres stratégies de déplacement, utilisant par exemple des distributions de probabilités gaussiennes unidimensionnelles, pour traiter des problèmes plus généraux, en particulier partiellement combinatoires. Par « combinatoires », nous entendons ici combinatoires *simples*. En effet, les autres, du type « voyageur de commerce », peuvent également être traités efficacement par l'OEP, mais uniquement grâce à des stratégies hybrides, utilisant conjointement des algorithmes de recherche locale spécifiques.

Description de Tribes

Les tribus

Un *informateur* d'une particule A est une particule B dont la meilleure position atteinte jusqu'ici peut être « lue » par A. Notons que cette définition implique que A est un informateur pour elle-même. Un informateur est dit *externe* s'il n'appartient pas à la tribu de la particule concernée. Si chaque particule de l'essaim est vue comme le sommet d'un graphe, on peut représenter le lien d'information par un arc de B vers A. L'arc inverse, de A vers B, peut exister, et existe effectivement dans la plupart des versions d'OEP, y compris celle-ci, mais ce n'est pas une obligation. Par ailleurs, sauf topologie particulière, toutes les particules ne pointent pas vers A. On peut ainsi définir des sous-ensembles (des cliques symétriques au sens de la théorie des graphes) tels que, dans chacun d'eux, toute particule pointe (informe) toutes les autres. Nous les appellerons ici *tribus*, la métaphore étant celle de groupes d'individus de taille variable évoluant dans un environnement inconnu, à la recherche d'un « bon » emplacement. Cette structuration induira pour ainsi dire mécaniquement un processus analogue au nichage dans les algorithmes génétiques et dans le même but : explorer simultanément plusieurs régions prometteuses, généralement des minimums locaux.

Les relations tribales

Même si chaque tribu parvient à trouver un minimum local, il est nécessaire qu'elles s'informent entre elles pour décider collectivement lequel est le minimum global. Par conséquent le réseau d'information entre tribus doit être connexe. En pratique, cela signifie que de toute particule A vers toute particule B il existe un chemin d'information, du type « A informe A1 qui informe A2 … qui informe B ». Résumons la structure globale : dans chaque tribu, un réseau dense et, entre tribus, un réseau assurant simplement la connexité. Nous sommes typiquement dans un graphe de relations de type « petit monde », structure connue pour être un compromis efficace en matière de diffusion et d'exploitation de l'information. Seulement cette structure doit être générée et modifiée automatiquement, par le biais des créations, évolutions et suppressions de tribus.

Qualité d'une particule

Comme en OEP classique, chaque particule a une position courante et une « meilleure performance », qui est mémorisée. C'est donc d'abord à ce niveau de détail que l'on peut dire s'il y a progrès ou non. Une particule sera dite *bonne* si elle vient d'améliorer sa meilleure performance, *neutre* sinon. Notons bien que cette définition est qualitative, on ne mesure pas l'amélioration, on se contente d'examiner si elle est strictement positive (amélioration réelle) ou bien nulle (pas d'amélioration). Par définition, la meilleure performance d'une particule ne peut pas se détériorer, c'est pourquoi on ne définit pas dans l'absolu de « mauvaise » particule. Par contre, au sein d'une tribu, on peut déterminer celle qui a la moins bonne performance. Elle sera appelée la *pire* (relativement à sa tribu). De même, on peut déterminer la *meilleure* particule.

De plus, par rapport à l'OEP classique, on augmente légèrement la mémoire de la particule, de façon à ce qu'elle se souvienne de ses deux dernières variations de performance, ébauchant ainsi un historique de ses déplacements. A partir de là, on peut définir un troisième statut : une particule sera dite *excellente* si ces deux variations sont des améliorations. Ceci nous sera utile pour choisir la stratégie de déplacement adaptée.

Qualité d'une tribu

Cependant, ce qui nous intéresse ici est la performance globale d'une tribu. Nous allons donc définir deux statuts, *bonne* et *mauvaise*, et postuler une règle floue très simple : « Plus le nombre de bonnes particules de la tribu est grand, plus elle est elle-même bonne et inversement ».

En pratique, le statut d'une tribu est évalué de la manière suivante. On considère sa taille *T* (son nombre de particules) et son nombre de bonnes particules *B*, au plus égal à *T*. On génère aléatoirement, selon une distribution uniforme, un nombre *p* entre 0 et *T*. Si *B* est inférieur ou égal à *p*, alors la tribu est dite mauvaise, sinon elle est dite bonne.

A ces statuts seront associés des règles d'évolution, tendant à favoriser la création de nouvelles tribus et, partant, l'exploration de l'espace de recherche.

Evolution des tribus

Suppression d'une particule

Le but est de trouver l'optimum, si possible à moindre frais, c'est-à-dire en effectuant le moins possible d'évaluations de la fonction. Par conséquent, dès que l'occasion se présente de supprimer à peu près sans risque une particule, il faut la saisir. Notons qu'il vaut mieux conserver à tort une particule (au pire l'on augmentera un peu le nombre d'évaluations strictement nécessaires) qu'en éliminer une à tort (avec le risque de rater complètement la solution). C'est pourquoi seule une bonne tribu pourra éventuellement éliminer une de ses particules et uniquement la pire d'entre elles. Dans le cas d'une tribu monoparticule, l'élimination ne se fera que si un des informateurs a une meilleure performance. En effet, on veut être sûr de conserver au moins une information de meilleure qualité que celle que l'on va éliminer.

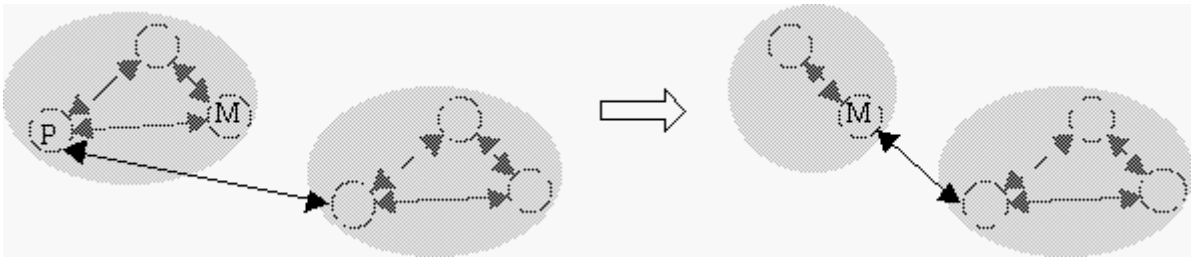

Figure 1. Suppression d'une particule d'une tribu pluriparticule. La particule P est la pire de sa tribu et la tribu a été déclarée « bonne ». Dans ce cas P est supprimée et la redistribution de ses liens externes (ici un seul lien symétrique) se fait sur M, la meilleure particule de la tribu

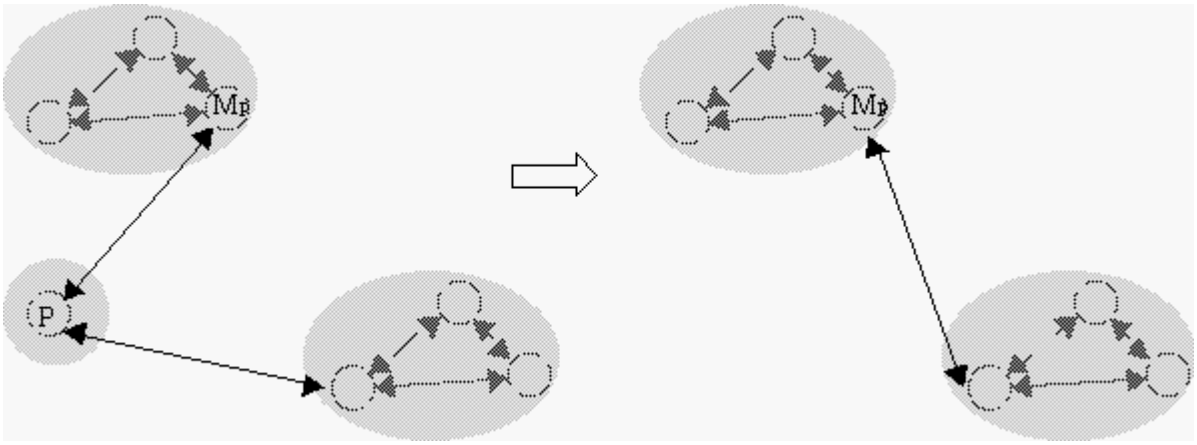

Figure 2. Suppression d'une tribu monoparticule. La tribu a été déclarée « bonne » et donc la particule unique P, qui est nécessairement la pire de la tribu, même si elle est en même temps « bonne », devrait être supprimée. Mais elle ne le sera effectivement que si son meilleur informateur externe M_p est meilleur qu'elle. L'hypothèse est en effet que l'information portée par P est alors moins précieuse que celle portée par M_p

Par ailleurs, l'élimination d'une particule implique une redistribution de ses liens d'information, montant et descendant. Dans le cas général, ce report a lieu sur la meilleure particule de la tribu (cf. Figure 1). Dans le cas d'une tribu monoparticule, puisque la suppression de la particule entraine celle de la tribu tout entière, ils sont reportés sur le meilleur informateur externe de la particule à supprimer (cf. Figure 2).

Génération d'une particule

Inversement, une mauvaise tribu est manifestement en manque d'information. Elle va donc générer une nouvelle particule, tout en gardant le contact avec elle. La génération se fait au plus simple, complètement aléatoirement selon une distribution uniforme dans l'espace de recherche. Plus précisément, toutes les mauvaises tribus vont générer ensemble chacune une particule et ces nouvelles particules vont former une nouvelle tribu. Le terme « garder le contact » signifie ici l'établissement d'un lien symétrique entre la particule générée et la tribu génératrice, représentée ici encore, par exemple, par son meilleur élément.

Fréquence des adaptations

Notons qu'il n'est pas nécessaire, ni souhaitable, de réaliser ces adaptations structurelles à chaque itération, car il faut laisser le temps à l'information de se propager entre les particules. Là encore plusieurs règles plausibles sont possibles. En théorie, après chaque adaptation, on devrait calculer le diamètre du graphe des relations. Pour cela, il faudrait considérer tous les couples de particules appartenant chacune à deux tribus différentes et trouver le plus court chemin d'information les reliant, en termes de nombre d'arcs. Le plus long des ces plus courts chemins nous donnerait une estimation du nombre d'itérations nécessaires pour être sûr qu'une information possédée par une particule puisse être transmise, plus ou moins directement et plus ou moins déformée, à toutes les autres. Néanmoins, ce calcul est un peu long et l'on pourra se contenter d'utiliser le nombre total de liens d'information. Si, après une adaptation structurelle, ce nombre est L, alors la prochaine aura lieu dans $L/2$ itérations.

Evolution de l'essaim

Quel est le genre de fonctionnement induit par ces règles ? Au départ, donc, il n'y a qu'une seule particule, représentant une unique tribu. Après la première itération, si sa situation ne s'améliore pas, ce qui est fort probable (et même certain avec les stratégies de déplacement examinées ci-dessous, car elle ne bouge pas du tout au premier pas de temps) une autre particule va être générée, formant une seconde tribu. A l'itération suivante, si aucune des deux particules n'améliore sa situation, les deux tribus vont générer simultanément chacune une particule : une nouvelle tribu de deux particules sera créée, et ainsi de suite, en notant que plus le nombre de liens augmente, plus le nombre d'itérations entre deux adaptations est important. Ainsi, tant que les choses vont mal, des tribus de plus en plus grandes sont générées, augmentant le pouvoir exploratoire de l'essaim, mais de plus en plus rarement. Entre deux adaptations, l'essaim a donc de plus en plus de chances de trouver une solution.

Mais, inversement, dès qu'une ébauche de solution est trouvée, chaque tribu va progressivement éliminer sa pire particule, éventuellement jusqu'à disparaitre complètement. En situation idéale, quand la convergence se confirme, toutes les tribus, sauf éventuellement la dernière créée, se réduisent à une ou deux particules. Globalement, l'essaim a tendance à croître, de plus en plus lentement, de façon asymptotique (cf. exemples ci-dessous). Il n'est réellement amené à décroitre, au moins temporairement, que dans certains cas simples, quand les tribus sont majoritairement bonnes.

Stratégies de déplacement

Intuitivement, il semble judicieux qu'une particule adopte une stratégie de déplacement fonction de son passé récent. Par exemple, si elle vient d'améliorer deux fois de suite sa position, il n'est sans doute pas nécessaire de la faire trop s'éloigner de la région où elle se trouve. A priori, puisque son historique comprend deux variations de performances, on pourrait distinguer quatre cas. Cependant, nous nous contenterons ici de deux, pour illustrer le principe : la particule est excellente (deux améliorations successives) ou ne l'est pas (les trois autres possibilités).

Quand la particule est excellente, nous utiliserons une stratégie certes partiellement aléatoire, mais pas trop, la méthode des pivots simple. Sinon, nous ajouterons encore plus d'aléatoire, avec la méthode des pivots bruitée. Notons, par ailleurs, que le confinement de la particule dans l'espace de recherche est réalisé de la même manière qu'en OEP classique, si ce n'est qu'il n'y a pas de vitesse à modifier. Si une composante de la position tend à sortir des valeurs admissibles, on la ramène donc simplement à celle qui est la plus proche.

Méthode des pivots simple

La méthode des pivots originelle [SER 97], retranscrite dans le vocabulaire de l'OEP, consisterait à n'avoir à chaque itération qu'un nombre pair de particules, à les apparier et, dans chaque couple, à désigner comme pivot la meilleure des deux. Le pivot ne bouge pas et la nouvelle position de l'autre particule est choisie au hasard selon une distribution symétrique centrée sur le pivot, par exemple une gaussienne.

Ici, on procède un peu différemment. Pour chaque particule on considère encore deux points de l'espace de recherche, mais ce sont sa meilleure performance p et la meilleure performance de ses informateurs, g. On définit ensuite deux hypersphères, H_p et H_g, centrées sur ces points et de même rayon égal à leur distance. Ensuite, dans chaque hypersphère, on choisit un point au hasard selon une distribution uniforme. Un petit programme permettant de le faire est donné en annexe. C'est d'ailleurs un excellent exercice de statistique ! Ensuite, on lui affecte un poids fonction décroissante de la valeur de la fonction à minimiser évaluée au centre de l'hypersphère. Enfin la nouvelle position est calculée comme barycentre de ces deux derniers points.

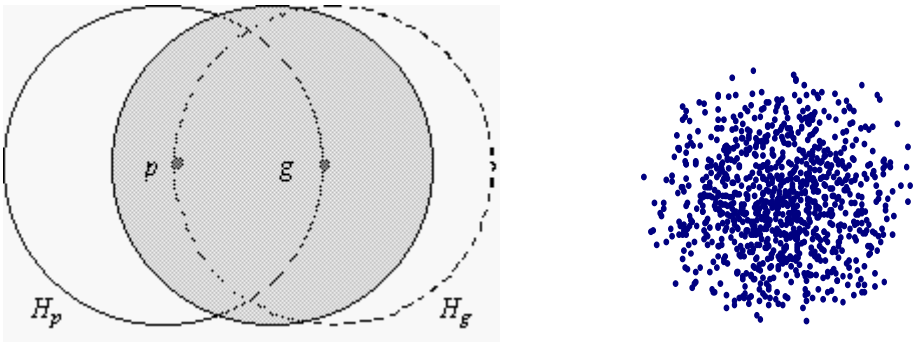

Figure 3. Choix d'une nouvelle position. Le point p est la meilleure performance de la particule, le point g la meilleure de celles de ses informateurs Le rayon commun aux hypersphères H_p et H_g est la distance entre ces deux positions. Un point est choisi au hasard uniformément dans H_p, un autre dans H_g et les deux combinés linéairement. Le résultat sera d'autant plus proche de H_q que g est meilleure que p. La distribution des nouvelles positions possibles est encore dans une hypersphère, mais n'est plus uniforme, comme on peut le voir sur l'échantillonnage de 1000 positions dans la partie droite de la figure

Première remarque, l'affectation d'un poids a un aspect arbitraire a priori assez gênant. Mais, en réalité, la méthode est extrêmement robuste vis-à-vis du choix de la fonction de poids, pourvu que celle-ci respecte quelques conditions très générales : décroissante stricte et valeur finie pour une valeur nulle de la fonction à

minimiser (supposée positive, cas auquel on peut, en pratique, toujours se ramener). Par exemple, on peut prendre comme coefficients de pondération $\frac{f(p)}{f(p)+f(g)}$ et $\frac{f(g)}{f(p)+f(g)}$.

Deuxième remarque, il semble que la position actuelle de la particule ne soit pas prise en compte pour calculer la position future, contrairement à la formule de l'OEP classique. Ceci peut sembler curieux, mais, justement, ce n'est qu'une impression. D'une part, si cette position est mauvaise, il n'y a en effet aucun intérêt à l'utiliser, au contraire. D'une certaine manière, on peut donc dire qu'elle est prise en compte précisément en l'ignorant ! D'autre part, si elle est bonne, alors en réalité elle coïncide avec sa meilleure performance *p* et, de ce fait, elle intervient bien dans le calcul.

Méthode des pivots bruitée

Cette seconde stratégie commence exactement comme la précédente. Simplement, une fois déterminée la nouvelle position, on la modifie encore selon un bruitage aléatoire gaussien, de moyenne zéro et d'écart-type d'autant plus faible que la meilleure performance de la particule est proche de celle de ses informateurs. Par exemple, si *f* est la fonction à minimiser, on pourra prendre comme écart-type $\frac{f(p)-f(g)}{f(p)+f(g)}$.

Ici encore, la formule exacte n'est pas très importante. Il suffit que le résultat soit au plus égal à 1 et strictement décroissant avec la différence des performances $f(p)-f(g)$.

En pratique, pour chaque composante de la position qui vient d'être calculée, on tire au hasard une valeur *b* selon la distribution de bruitage et l'on multiplie la composante par (1+*b*).

Exemples de fonctionnement

Nous avons maintenant en main tous les éléments pour coder le programme et le faire fonctionner. Un source en langage C est d'ailleurs disponible sur Internet (cf. le *Particle Swarm Central* [PSC], section *Programs/Maths Stuff*). Les résultats présentés ici ont été obtenus avec la version 5.0. Traitons donc quelques cas d'écoles, juste pour illustrer les principaux aspects de son fonctionnement, surtout comparés à une version d'OEP plus classique. Celle utilisée ici comme référence est l'OEP avec constriction (OEPc) [CLE 02, KEN 01]. Rappelons très brièvement que, dans cette version, le déplacement d'une particule est défini pour chaque dimension *d* par

$$\begin{cases} v_d \leftarrow \chi\left(v_d(t) + alea(0,\varphi/2)(p_d - x_d) + alea(0,\varphi/2)(g_d - x_d)\right) \\ x_d \leftarrow x_x + v_d \end{cases}$$

avec comme coefficient de constriction $\chi = 2\Big/\left(\varphi - 2 + \sqrt{\varphi^2 - 4\varphi}\right)$. La fonction *alea*(*a*,*b*) délivre un nombre au hasard selon une distribution uniforme entre les réels *a* et *b*. Les paramètres utilisés ici sont classiques : un voisinage circulaire de taille trois et un coefficient de confiance φ égal à 4,1. Si les particules d'un essaim de taille *N* sont numérotées de 0 à *N-1*, le voisinage circulaire de taille trois de la particule *n* est le groupe d'informateurs de rangs {*n*-1 [modulo *N*], *n*, *n*+1 [modulo *N*]}. L'essaim est de taille constante, mais différentes tailles seront essayées. Pour Tribes, naturellement, tous ces paramètres n'ont pas lieu d'être définis.

En tant qu'utilisateur, vous pouvez souhaiter soit minimiser le temps calcul nécessaire pour trouver une solution avec une erreur inférieure à un seuil donné, soit trouver la meilleure solution possible en un temps donné. En pratique, nous remplacerons « temps calcul » par « nombre d'évaluations de la fonction ». Ces deux notions ne sont pas toujours équivalentes, mais c'est bien le cas ici, car les calculs intermédiaires ne contribuent que très faiblement au temps total. Ceci est vrai même pour Tribes, les calculs nécessaires aux adaptations étant simples et effectués relativement rarement.

Minimisation du nombre d'évaluations

Rosenbrock 2D

Considérons d'abord l'exemple très simple de la fonction Rosenbrock[2] dans le carré $[-10,10]^2$. Nous savons ici que son minimum global est la valeur zéro. Recherchons-le en indiquant au programme que nous souhaitons trouver une valeur au plus égale à 10^{-3}. Nous travaillons ici avec des algorithmes ayant un fonctionnement partiellement aléatoire. On peut donc essentiellement évaluer des valeurs moyennes et des probabilités de réussite, en exécutant plusieurs fois le programme (et sans réinitialisation du générateur de nombres aléatoires !).

Ainsi on peut estimer les distributions du nombre d'évaluations nécessaires pour descendre en dessous du seuil d'erreur, avec une probabilité de réussite supérieure à 99 %, et ceci pour Tribes d'une part et pour OEPc d'autre part, avec différentes tailles d'essaim. Déjà, l'on constate (cf. Figure 4), à l'examen des estimations calculées à partir de 500 exécutions, que ces distributions ne sont certainement pas gaussiennes. Elles ne sont même pas symétriques et correspondent plutôt à des lois Béta. Ce phénomène reste encore à interpréter, mais rappelle que les seules informations moyenne et écart-type peuvent être insuffisantes pour comparer deux algorithmes. Ce n'est pas le cas ici, les différences étant très nettes. Pour OEPc, la taille de 15 particules est optimale, donnant un nombre moyen d'évaluations de 2436, avec un écart-type de 1792. En deçà la probabilité de réussite est inférieure au seuil choisi, au-delà le nombre moyen d'évaluations est plus grand. En moyenne, Tribes est un peu moins bon, avec une moyenne de 2680 évaluations et un écart-type de 1454, mais n'oublions pas qu'il a fallu une vingtaine d'essais avec OEPc (toutes les tailles entre 10 et 30 particules) pour trouver la « bonne » taille d'essaim. Avec 30 particules, par exemple, on obtient un nombre moyen d'évaluations supérieur (3131, écart-type 1738).

Avec Tribes, ce problème ne se rencontre pas, puisque, de toutes façons, il démarre toujours avec une seule particule. La Figure 5 présente l'évolution de la taille de l'essaim dans un cas moyen. Nous avons vu que le nombre d'évaluations entre deux adaptations est fonction croissante de la taille de l'essaim. Il en résulte qu'un palier plus élevé que le précédent est aussi nécessairement plus long (et inversement). Par ailleurs, les règles pour les suppressions et les ajouts accordent un léger avantage à ces derniers. Finalement, en moyenne, l'essaim ne peut pas grandir très vite, ce qui limite le nombre d'évaluations, tout en devenant quand même de plus en plus efficace.

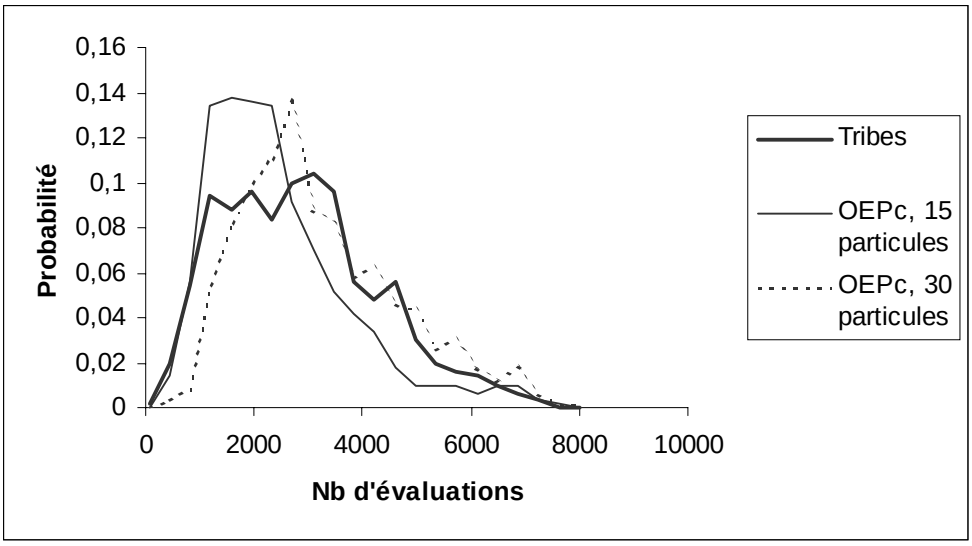

Figure 4. Rosenbrock 2D. Distributions de nombres d'évaluation. L'efficacité de l'algorithme, estimée en nombre d'évaluations de la fonction à minimiser pour trouver une valeur inférieure à 10^{-3} avec une probabilité supérieure à 99 %, varie selon la taille de l'essaim pour OEPc. La taille optimale est de 15, trouvée après de nombreux essais, ce qui donne une valeur moyenne de 2436 évaluations. Avec Tribes, on obtient un résultat global un peu moins bon (moyenne de 2680 évaluations), mais sensiblement meilleur, par exemple, qu'avec OEPc 30 particules. Les distributions ont été calculées à partir de 500 exécutions dans chaque cas

[2] Les fonctions de test très classiques (Rosenbrock, Sphère, Griewank, etc.) ne sont pas explicitées ici.

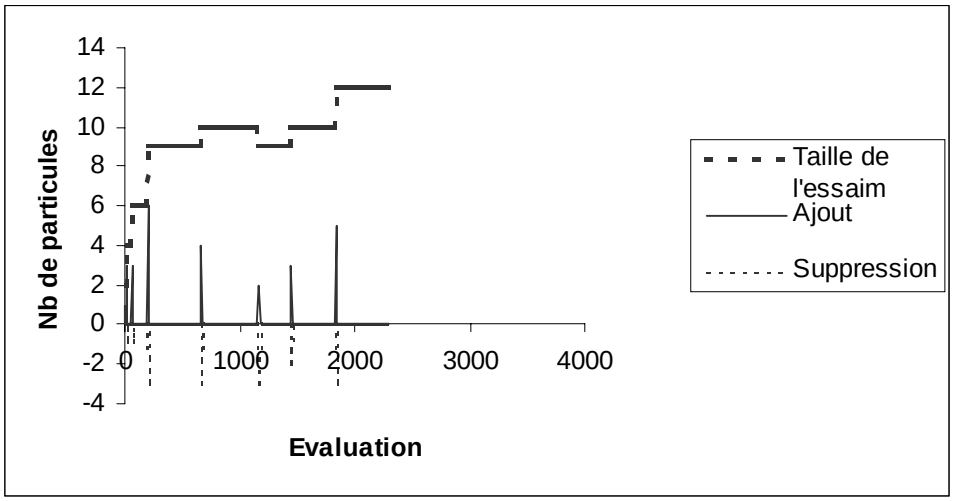

Figure 5. Rosenbrock 2D. Evolution de la taille de l'essaim avec Tribes. Pratiquement à chaque adaptation, les ajouts (générations de particules) l'emportent légèrement sur les suppressions, comme c'est généralement le cas grâce aux règles dissymétriques. Un palier représente l'intervalle entre deux adaptations. Un palier plus haut que le précédent est aussi plus long et inversement, ce qui ralentit en moyenne la croissance. Dans cette exécution, on obtient une erreur inférieure à 10^{-3} après 2291 évaluations

Les ajouts de particules à l'essaim augmentent son énergie cinétique et, partant, son pouvoir exploratoire, et ce de façon plus pertinente que les seules variations aléatoires apparaissant avec OEPc. La Figure 6 présente les variations de l'énergie cinétique dans les deux cas. On constate bien que les accroissements significatifs pour Tribes sont rares, mais ils sont réalisés à bon escient. En particulier, le dernier pic peu après la 1800ème évaluation, très important, a lieu précisément parce très peu de particules progressent encore et qu'il faut donc en générer de nombreuses nouvelles.

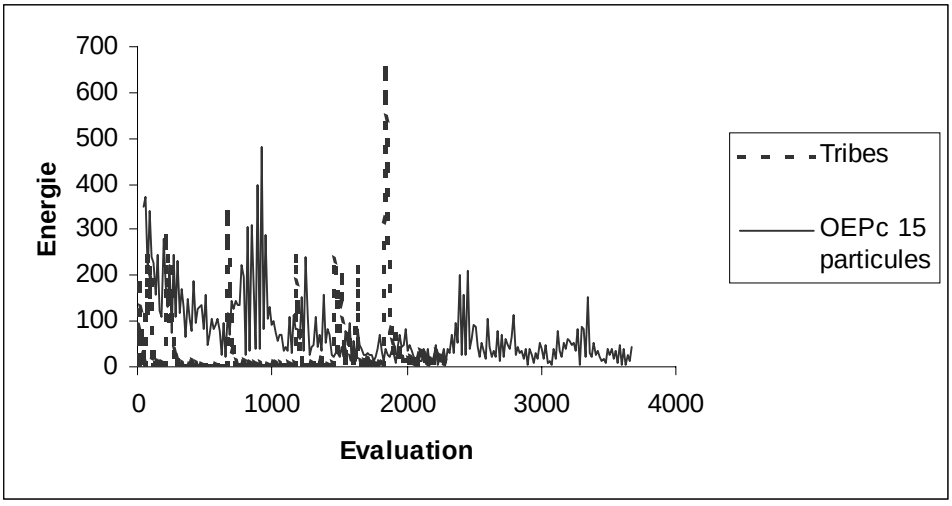

Figure 6. Rosenbrock 2D. Evolutions de l'énergie cinétique. Pour OEPc, les fluctuations aléatoires ne suffisent pas à l'empêcher de converger plutôt lentement. Il lui faut 3675 évaluations pour trouver une solution. Les pics d'énergie pour Tribes correspondent aux augmentations nettes de la taille de l'essaim visibles sur la Figure 5

Griewank 30D

Lorsque de nombreuses exécutions consécutives sont sans échec, il est facile de calculer différents éléments statistiques, comme vu ci-dessus. Echec est pris ici au sens « pas de solution en moins de *N* évaluations ». Evidemment, plus *N* est grand, plus le taux d'échecs diminue, mais, pour un algorithme donné, il peut tendre vers une limite non nulle. Nous nous intéressons ici à ce cas, avec l'exemple de la fonction Griewank sur

$[-300,300]^{30}$, pour laquelle toutes les versions d'OEP, Tribes inclus, sont souvent piégées dans un minimum local. Expérimentalement, on constate que si une valeur inférieure à 10^{-3} n'est pas trouvée en moins de 40000 évaluations, on peut tenir pour très probable qu'il n'est pas utile de continuer et qu'il vaut mieux relancer le programme avec une autre position initiale. Nous prendrons donc N=40000.

Pour OEPc, c'est la taille de 20 particules qui est optimale. On obtient un taux d'échec d'environ 55 %. Avec Tribes, le taux d'échec est pratiquement le même (légèrement inférieur, en fait). Cela signifie qu'il suffit, dans les deux cas, d'exécuter 8 fois le programme (sans réinitialiser le générateur de nombres aléatoires !) pour être sûr, à 99 %, d'obtenir au moins une exécution réussie, c'est-à-dire donnant une solution en moins de 40000 évaluations.

Puisque les taux d'échec sont équivalents, on peut comparer en ne considérant que les exécutions réussies. Pour OEPc 20 particules la moyenne des nombres d'exécutions est de l'ordre de 35900, alors que pour Tribes, on obtient une moyenne de 24704. Ainsi, même dans ce cas, Tribes se révèle significativement meilleur.

Recherche à nombre d'évaluations constant

Une autre manière d'aborder la question de l'efficacité est de se dire : j'ai droit à 40000 évaluations, quel est le meilleur résultat que je peux espérer obtenir ? Plus précisément, pour chaque niveau de précision, quelle est la probabilité de réussite ? Avec OEPc, pour « dégrossir » le problème, il faut déjà trouver la meilleure taille d'essaim, en exécutant le programme un nombre de fois suffisant pour avoir une bonne idée du meilleur résultat accessible, et ceci pour toute une gamme de tailles d'essaim.

Considérons les résultats obtenus pour la fonction Rosenbrock sur $[-10,10]^{30}$, représentés sur la Figure 7. Pour chaque taille, cent exécutions de 40000 évaluations ont été lancées, et la moyenne des cent résultats obtenus est indiquée en ordonnée. On constate que la taille d'essaim optimale est de 20, mais il n'y a pas du tout monotonie : les tailles de 15, 16 ou 23 sont presque aussi bonnes, mais pas les tailles intermédiaires.

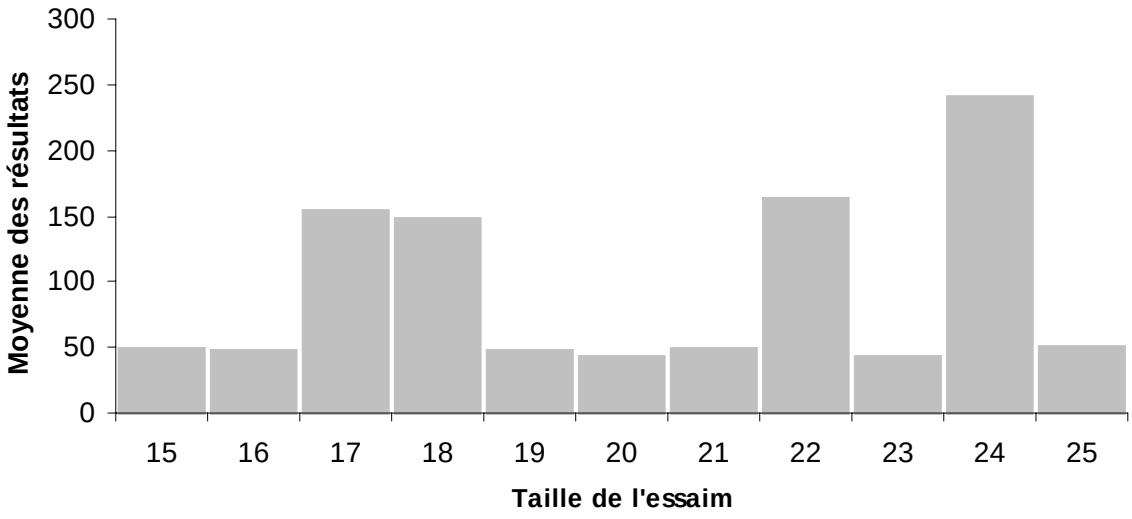

Figure 7. Rosenbrock 30D. Ici, on s'astreint à effectuer 40000 évaluations. Avec l'OEP classique, le résultat est éminemment variable selon la taille de l'essaim, sans que l'on puisse dégager de règle. Pour chaque taille on effectue cent exécutions. Le meilleur résultat (43,3) est obtenu pour une taille de 20 particules

Une fois ce résultat acquis, pour mieux estimer les probabilités, on peut lancer 500 exécutions, avec OEPc 20 particules d'une part, avec Tribes d'autre part. Dans le premier cas, on trouve une valeur finale moyenne de 43,5 (écart-type 49,6) et de 42,6 (écart-type 39,6) dans le second. La Figure 8 confirme que Tribes est en effet dans l'ensemble légèrement meilleur que OEPc, essentiellement grâce à la fréquence nettement plus grande avec laquelle il obtient des valeurs inférieures à 25 après les 40000 évaluations : dans 11,5 % des cas, contre 4 % pour OEPc.

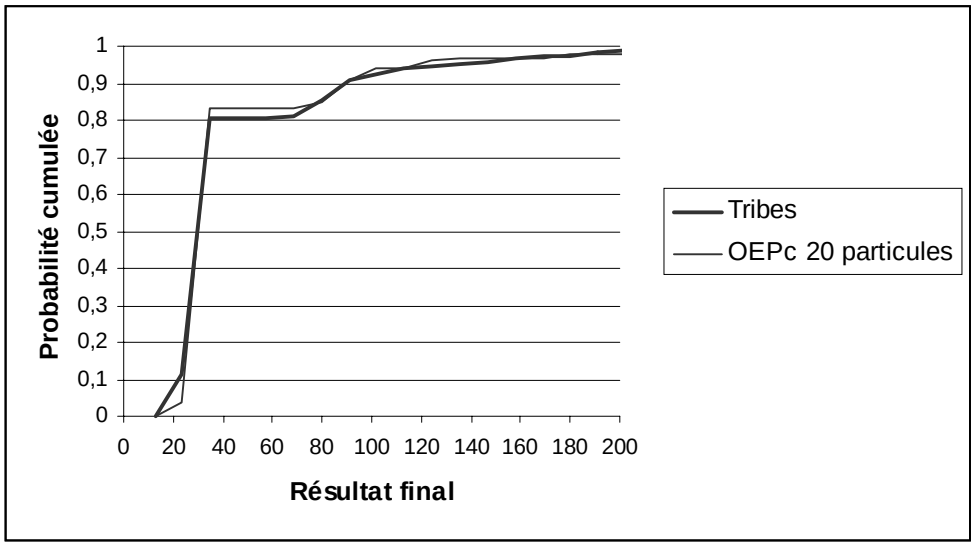

Figure 8. Rosenbrock 30D. Probabilités d'obtenir au plus une valeur donnée après 40000 évaluations. Par exemple, avec OEPc 20 particules, on a 83 % de chances d'obtenir une valeur inférieure à 40, contre 80 % pour Tribes. Pour les faibles valeurs, Tribes est nettement meilleur : 11,5 % de chance d'obtenir une valeur inférieure à 25, contre 4 % pour OEPc

Problèmes hétérogènes

En théorie, l'utilisation d'hypersphères limite l'application des stratégies de déplacements aux espaces de recherche munis d'une distance. Néanmoins, on peut être tenté d'appliquer telle quelle la méthode à d'autres espaces de recherche, en particulier partiellement combinatoires (codés, par exemple, avec des valeurs entières sur certaines dimensions), sans se soucier de justification mathématique. Le fait est que cela fonctionne, et même plutôt bien (sur les petits problèmes, comme déjà signalé), mais il est quand même possible d'améliorer encore les performances en définissant une stratégie de déplacement adaptée à ce cas.

Pour cela, il suffit de considérer chaque dimension indépendamment. Il serait tout à fait possible de reprendre exactement les stratégies « standard » vues plus haut, auquel cas, pour chaque dimension, les hypersphères se réduiraient à des intervalles. Cependant la distribution globale résultante se situerait alors dans un hyperparallélépipède (légèrement déformé pour la stratégie non bruitée), ce qui, nous l'avons vu, n'est pas très satisfaisant. Profitons donc de l'occasion pour montrer qu'il ne faut pas se polariser sur *une* méthode (les hypersphères) et que d'autres sont possibles. Définissons, par exemple, une stratégie inspirée de celle de l'OEPc, mais à partir de distributions gaussiennes indépendantes. Pour chaque dimension *d*, la nouvelle valeur de la composante de la position est calculée ainsi

$$\begin{cases} v_d = x(t)_d - x(t-1)_d \\ \delta_i = p_{i,d} - x(t)_d \\ \delta_g = p_{g,d} - x(t)_d \\ x(t+1)_d = x(t)_d + \chi\left(v_d + alea\_gauss\left(\delta_i, |\delta_i|/2\right) + alea\_gauss\left(\delta_g, |\delta_g|/2\right)\right) \end{cases}$$

avec

$$\begin{cases} \varphi = 2/0{,}97225 \\ \chi = 1\Big/\left(\varphi - 1 + \sqrt{\varphi^2 - 2\varphi}\right) \end{cases}$$

et la fonction *alea_gauss*(*m*,*s*) délivrant un nombre selon la distribution statistique normale de moyenne *m* et d'écart-type *s*. La Figure 9 donne une idée du résultat pour un espace à deux dimensions (réelles, pour faciliter la comparaison avec la méthode des hypersphères). La distribution est d'une part plus concentrée, ce qui est pénalisant pour l'exploration à moyenne distance, mais, d'autre part, la probabilité d'explorer à grande distance

est non nulle : n'importe quel point a une chance d'être examiné, ce qui est précieux pour les espaces finis, au moins selon certaines dimensions. Ainsi, si une des variables, comme dans un des exemples ci-dessous, est un entier entre 1 et 100, tout entier de cet intervalle a une probabilité non nulle d'être considéré.

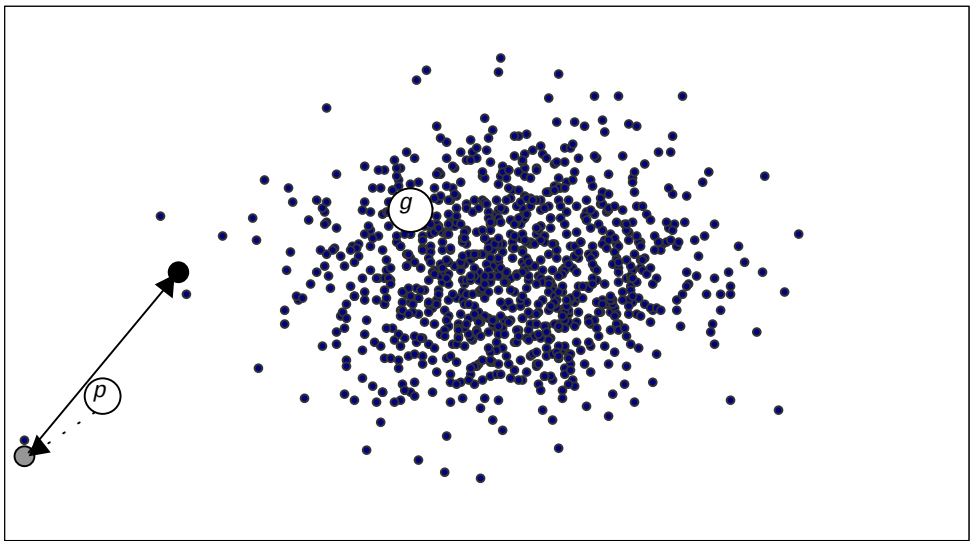

Figure 9. Distribution à deux gaussiennes indépendantes. La particule (disque noir) vient de se déplacer. La position précédente (disque gris à bord noir) définit l'origine de son vecteur vitesse. Elle a été atteinte, après une ou plusieurs itérations, à partir de p, meilleure performance réalisée. La meilleure performance des informateurs est g. Le dessin représente un échantillonnage de 1000 possibilités pour la prochaine position. On remarque sa plus forte densité plus ou moins vers g, mais aussi que quelques points, même s'ils sont rares, offrent des possibilités d'exploration plus lointaine

Nous allons illustrer cette variante de stratégies spécifiques sur deux petits exemples. Notons que cette distribution contient son propre bruit gaussien, c'est pourquoi nous n'utiliserons qu'une seule stratégie, quel que soit le statut de la particule à déplacer, sans nous préoccuper de son passé récent.

Sac à dos

L'énoncé est simple : trouver dix nombres entiers différents entre 1 et 100 dont la somme fasse 200. L'espace de recherche est donc de dimension dix et pour chaque dimension les valeurs admissibles sont tous les entiers entre 1 et 100 compris. La Figure 10 représente les résultats obtenus d'une part avec l'OEPc (une fois trouvée la meilleure taille d'essaim, qui est ici de 9 particules pour une probabilité de succès supérieure à 99,9%) et d'autre part avec Tribes, soit en utilisant le couple de stratégies «standard », soit uniquement la stratégie avec distributions gaussiennes indépendantes que nous venons de voir. Dans tous les cas, on applique une contrainte de confinement qui consiste, une position ayant été trouvée, à se déplacer vers la plus proche ayant ses composantes toutes différentes. Comme on peut le constater, avec les hypersphères et la double stratégie, Tribes est satisfaisant, mais sans plus, nécessitant un nombre moyen d'évaluations égal à 188 (écart-type 134) contre 92 (écart-type 41) pour OEPc.

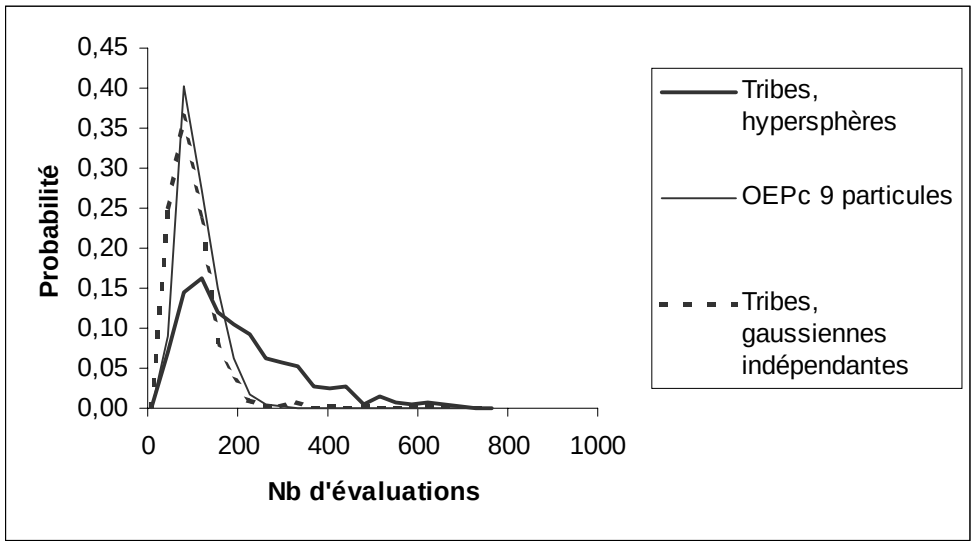

Figure 10. Sac à dos 10D. Distributions des résultats obtenus par différentes méthodes. Tribes « standard » est moins bon qu'OEPc (une fois que l'on a trouvé la taille d'essaim optimale de neuf particules), avec des nombres moyens d'évaluations respectivement de 188 et 92. L'utilisation de la stratégie de déplacement avec distributions gaussiennes indépendantes est par contre nettement plus efficace (nombre moyen d'évaluations égal à 86)

Les courbes reflètent les résultats de 500 exécutions pour chaque cas. Dans ces conditions, l'excellent résultat obtenu avec les distributions gaussiennes (moyenne 86, écart-type 88) a moins de 5 % de chances d'être simplement dû au hasard.

Hybride JM

Cet intéressant petit problème à trois dimensions a été proposé par B. Jeannet et F. Messine [JEA 03]. La fonction à minimiser est

$$f(x_1, x_{2,} x_3) = 20a_1(x_1)x_2^2 + 2a_2(x_1)x_2x_3$$

et l'espace de recherche est défini par

$$\begin{cases} x_1 \in \{1,2,3,4,5,6\} \\ x_2 \in [-15,25] \\ x_3 \in [3,10] \end{cases}$$

La variable x_1 n'est en fait qu'un indice utilisé pour renvoyer des valeurs à partir de deux listes a_1=(0,5 0,3 0,8 0,1 0,9 0,12) et a_2=(-0,5 0,6 0,1 1,5 -1, 0,8). Le minimum -112,5 est obtenu pour x_1=4, x_2=-7,5 et x_3=10.

La méthode proposée par les auteurs (à partir d'une arithmétique d'intervalles et de fonctions d'inclusions) donne la solution après 3271 évaluations. Cette méthode étant déterministe, le résultat est absolument certain, ce qui n'est évidemment jamais le cas avec l'OEP. Pour une comparaison honnête, nous devons donc imposer une probabilité de succès très élevée, par exemple 99,99 %. Comme le montre clairement la Figure 11, on peut alors faire mieux avec l'OEPc, toujours sous la réserve classique qu'il faut trouver la bonne taille d'essaim (ici 20 particules). Le nombre d'évaluations moyen est en effet de 1007. Il est de 1040 pour Tribes avec distributions hypersphériques, mais descend à 600 avec les distributions gaussiennes indépendantes.

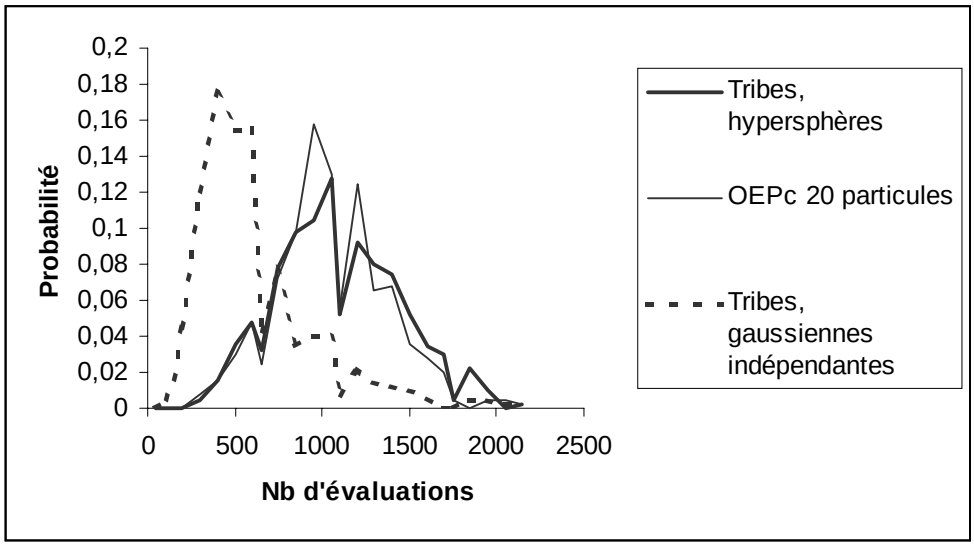

Figure 11. Problème hybride 3D. Une dimension est continue, les deux autres sont discrètes. On s'impose une probabilité de succès supérieure à 99,99 %. Dans ce cas, l'utilisation de stratégies de déplacement à l'aide de distributions gaussiennes indépendantes est nettement plus efficace. Le nombre moyen d'évaluations est en effet seulement de 600, contre 1040 pour les distributions hypersphériques et 1007 pour l'OEPc à 20 particules, taille optimale pour le taux de réussite souhaité

Différencier les problèmes partiellement combinatoires

Evidemment, une question se pose : pourquoi ne pas choisir la stratégie des gaussiennes indépendantes pour *tous* les types de problèmes, que les espaces de recherche soient partiellement combinatoires (hybrides) ou non ? En fait, après de nombreux essais, la réponse est mitigée, comme l'illustre le Tableau 1. Pour le premier problème (Rosenbrock à 2 dimensions), les distributions gaussiennes indépendantes sont nettement meilleures. Mais ce n'est plus du tout le cas pour les deux autres problèmes continus. Par contre, elles sont effectivement plus performantes pour les trois problèmes ayant un aspect combinatoire.

Problème	**Nature du problème**	**Tribes, hypersphères**	**Tribes, gaussiennes indépendantes**
Rosenbrock sur $[-10, 10]^2$, précision 0,001	continu	2680 (1454)	1768 (1125)
Sphère sur $[-20, 20]^{10}$, précision 0,001	continu	987 (224)	1665 (330)
Griewank sur $[-300, 300]^2$, précision 0,001	continu	24704 (23087) taux d'échec de 30 % pour un nombre maximum d'évaluations égal à 10000	taux d'échec supérieur à 80 % pour un nombre maximum d'évaluations égal à 100000
Sac à dos 10D	combinatoire	188 (134)	86 (88)
Hybride JM	hybride	1040 (356)	600 (548)
Moitié-moitié 20D (20 entiers différents entre 1 et 100 dont la somme des dix premiers est égale à la somme des dix derniers)	combinatoire	332 (293)	70 (48)

Tableau 1. Intérêt du changement de stratégie pour les problèmes hétérogènes. Chaque problème a donné lieu à 500 exécutions et le résultat indiqué est le nombre moyen d'évaluations (avec l'écart-type entre parenthèses). Sauf mention contraire, le taux d'échec est inférieur à 0,1 %

Ainsi, il vaut mieux choisir judicieusement telle ou telle stratégie selon la nature du problème. Malheureusement, c'est à vous de dire si le problème est, à votre avis, partiellement combinatoire ou non, le

simple fait que certaines dimensions soient discrètes ne suffisant pas à en décider automatiquement. En effet, si un problème combinatoire implique toujours un espace de recherche discret, la réciproque n'est pas vraie. Notons cependant que la méthode par défaut fonctionne dans tous les cas, même si, donc, parfois, elle a des performances inférieures à celles que l'on peut obtenir en affinant la description du problème.

Résumé

Il est possible de concevoir des algorithmes d'OEP sous forme de « boites noires », l'utilisateur n'ayant à définir que l'espace de recherche, la fonction à minimiser, la précision souhaitée et, par précaution, un nombre maximum d'évaluations.

Le programme Tribes, dont les sources sont disponibles sur l'Internet, est un exemple de réalisation d'un tel algorithme. Il fonctionne par coopération de tribus de particules. Dans chaque tribu, les liens d'information forment un graphe fortement connexe. Entre tribus, les liens sont plus lâches, mais le graphe total reste toujours connexe. Les stratégies structurelles mises en œuvre automatiquement concernent l'ajout ou la suppression de particules et de leurs liens d'information. Les stratégies de déplacement d'une particule sont fondées sur des distributions de probabilités hypersphériques, bruitées ou non, le choix tenant compte de l'historique à court terme de cette particule. Pour les problèmes partiellement combinatoires, il semble intéressant d'utiliser plutôt des distributions gaussiennes indépendantes pour chaque dimension.

Du simple fait qu'il permet d'éviter la fastidieuse recherche de « bons » paramètres, en particulier la taille de l'essaim, Tribes se révèle supérieur à l'OEP classique (avec constriction), d'autant que les résultats, évalués en termes de « nombre moyen d'évaluations nécessaires pour obtenir une solution » ou de « meilleure solution obtenue après un nombre donné d'évaluations », sont de meilleure qualité ou équivalents.

Références

[JEA 03] Jeannet B., Messine F., "Méthode déterministe d'optimisation globale pour les problèmes hybrides", *Cinquième congrès de la Société Française de Recherche Opérationnelle et d'Aide à la Décision (ROADEF 2003)*, Avignon, France, 2003.

[SER 97] Serra P., Stanton A. F., Kais S., "Pivot method for global optimization", *Physical Review*, vol. 55, 1997, p. 1162-1165.

[CLE 02] Clerc M., Kennedy J., "The Particle Swarm-Explosion, Stability, and Convergence in a Multidimensional Complex Space", *IEEE Transactions on Evolutionary Computation*, vol. 6, 2002, p. 58-73.

[KEN 01] Kennedy J., Eberhart R., Shi Y., *Swarm Intelligence*, Morgan Kaufmann Academic Press, 2001.

[PSC] Particle Swarm Central, http://www.particleswarm.net.