Système intelligents avancés

2 Décembre, 2018

Promotion Neumann (5th year)

Algorithmes génétiques avec le framework jMetal

Vincent CANDAPPANE

Prof. Wahabou Abdou[1]
ESIREM University
Software & Knowledge Engineering School,
9 Avenue Alain Savary,
21000 Dijon, France

Table des matières

Table des figures

Introduction

Les ingénieurs se heurtent quotidiennement à des problèmes technologiques de complexité grandissante, qui surgissent dans des secteurs très divers. Le problème à résoudre peut fréquemment être exprimé sous la forme générale d'un problème d'optimisation, dans lequel on définit une fonction objectif, que l'on cherche à minimiser (ou maximiser) par rapport à tous les paramètres concernés. Les algorithmes évolutionnistes[5] et plus précisément les algorithmes génétiques ont ce but d'obtenir une solution approchée à un problème d'optimisation, lorsqu'il n'existe pas de méthode exacte pour le résoudre en un temps raisonnable et en utilisant la notion de sélection naturelle.

Dans ces TPs, il sera présenté, les trois approches de résolution des différents problèmes d'optimisation. La première approche est une résolution mono-objective basée sur l'agrégation d'un seul objectif à savoir la minimisation de fonctions. La seconde est une approche scalaire(méthode d'aggrégation, méthode $\in$-contrainte), il s'agit de la résolution de problèmes multi-objective basée sur l'agrégation de deux fonctions objectifs. Enfin, la troisième méthode est un algorithme génétique multi-objectif de type NSGA-II qui a pour but de trouver l'ensemble des meilleurs compromis possible entre les deux objectifs du modèle.

Dans le cadre de ce TP, nous ne nous intéresserons qu'aux algorithme génétiques mono-objectifs et multi-objectifs via le framework jMetal[4].

1. TP1 : Découverte du framework jMetal

1.1 Introduction

jMetal est un framework implémenté en Java pour la résolution de problèmes d'optimisation. Il propose plusieurs métaheuristiques et est extensible. Nous nous intéresserons dans ce TP à un algorithme génétique mono-objectif. Lorsqu'un seul objectif (critère) est donné, le problème d'optimisation est mono-objectif. Le but étant de minimiser les fonctions de Schaffer et de Schwefel séquenciellement.

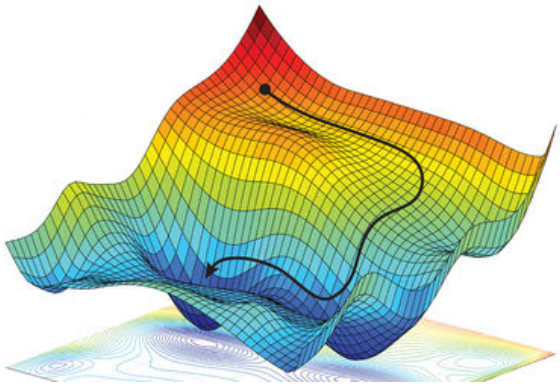

FIGURE 1.1: Cheminement du programme de Schaffer à minimiser

1.2 Schaffer

Le programme qui optimise la fonction de Schaffer est défini par la fonction suivante :

$$\sum_{i=1}^{N-1} (x_i^2 + x_{i+1}^2)^{0.25} \cdot [\sin^2(50 \cdot (x_i^2 + x_{i+1}^2)^{0.10}) + 1.0], \text{ avec } x_i \in [-100; 100]$$

Notre but est de minimiser cette fonction mathématique. Grâce à la classe *GA_main.java* (JMetal.metaheuristics.singleObjectif.geneticAlgorithm), il a été possible d'instancier un nouveau problème(formule de Schaffer) qui a été au préalable défini par nos soins de la manière suivante :

— Type de la solution : "Real",
— Nombre de variables au nombre de 10,
— Intervalle de la variable x_i dans [-100 ;100]

L'écriture de la fonction objectif[7] de Schaffer se développe dans la méthode evaluate(). En exécutant le tout, nous obtenons la solution égale à '0.0' avec 300 000 évaluations(maxEvaluations) et avec une taille de population(populationSize) de 100.

1.3 Schwefel

La fonction de Schwefel définie comme-ci est a optimiser :

$$f(s) = 418.9828872724339 \cdot N - \sum_{i=1}^{N} \cdot (x_i \sin(\sqrt{|x_i|})), \text{ avec } x_i \in [-500; 500]$$

Cette exercice utilise le même squelette et même base que l'exercice précédent. Il a fallu modifier l'intervalle de la variable x_i compris dans $[-500; 500]$ ainsi que l'écriture de la fonction adéquate de Schwefel à minimiser dans la méthode evaluate(). En exécutant le tout, nous obtenons la solution égale à '118.4383346144391' avec 2500000 évaluations(maxEvaluations) et avec une taille de population(populationSize) de 100.

1.4 Conclusion

Grâce à ce premier TP, il a été possible d'explorer l'espace de recherche de différent problème mono-objectifs et donc dans le but de trouver une solution la plus optimale possible. Nous avions remarqué durant les manipulations que :

— (Trop) Intensifier les recherches autour des bonnes solutions pourrait conduire à une perte de diversité et à une convergence prématurée.

— (Trop) Augmenter la diversité pourrait empêcher la convergence de l'algorithme.

Il est donc important de trouver un équilibre entre diversité et convergence.

2. TP2 - Optimisation mutli-objectifs

2.1 Introduction

Un problème d'optimisation multi-objectif non-évolutionnaire est un problème qui possède plusieurs fonctions objectif qui sont à minimiser ou à maximiser et un certain nombre de contraintes à satisfaire. Nous proposons dans cette section de survoler rapidement quelques méthodes classiques de l'optimisation multi-objectif non-évolutionnaire (méthode d'aggrégation, méthode $\in$-contrainte). À noter que toute ces méthodes reposent sur la transformation du problème initial en un problème d'optimisation mono-objectif.

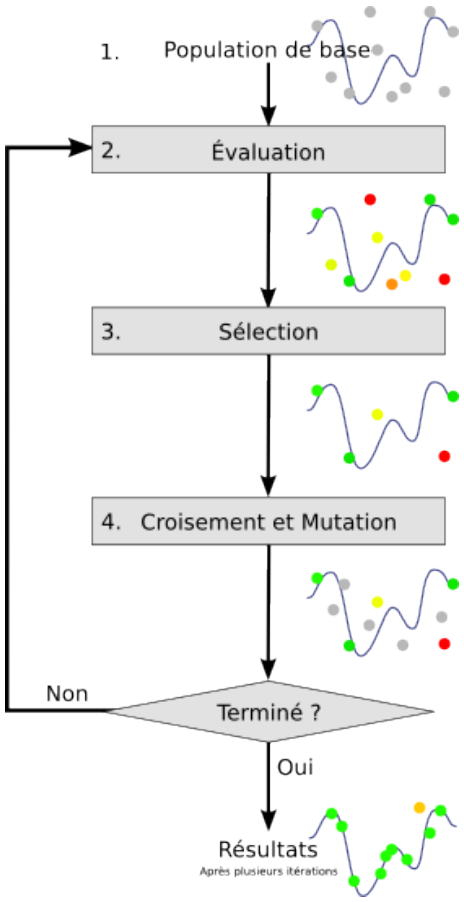

FIGURE 2.1: Fonctionnement général d'un algorithme génétique

2.2 Approche scalaires

Le but étant ici de minimiser les fonctions scalaires f_1 et f_2 permettant une optimisation multi-objectifs en utilisant une approche scalaire.

2.2.1 Méthode d'agrégation

La méthode d'agrégation populaire consiste à ramener un problème multi-objectif à un problème d'optimisation d'une combinaison linéaire des objectifs initiaux. Cette combinaison est caractérisée souvent par une pondération qui traduit l'importance relative des objectifs. Elles introduisent de nouveaux paramètres : des poids ou des contraintes sur des objectifs.

$$f_1 = x_1 + \frac{2}{|J_1|} \cdot \sum_{j \in J1} (x_j - x_1^{0.5(1.0+\frac{3(j-2)}{n-2})})^2$$

$$f_2 = 1 - x_1 + \frac{2}{|J_2|} \cdot \sum_{j \in J2} (x_j - x_1^{0.5(1.0+\frac{3(j-2)}{n-2})})^2$$

L'espace de recherche étant : $[0,1]^n$, avec $N = n = 10$

Satisfaisant la contrainte C :

$$f_1 + f_2 - \left|sin[N\pi(f_1 - f_2 + 1)]\right| - 1 \geq 0$$

où : $J_1 = \{j | j$ est pair, et $2 \leq j \leq n\}$ et $J_2 = \{j | j$ est impair, et $2 \leq j \leq n\}$

Afin de codifier tous cela, l'on a créé comme dans les TPs précedents , une nouvelle classe dans le package des problèmes : $tp3\_multi\_objectif\_optimisation.java$. Les cardinalités de J_1 et J_2 ont été défini de la manière suivante :

- $J_1 = \{2, 4, 6, 8, 10\}$, (tableaux de nombres pairs) avec pour cardinalité $|J_1| = 5$
- $J_2 = \{3, 5, 7, 9\}$, (tableaux de nombres impairs), avec pour cardinalité $|J_2| = 4$

Donc, l'on calcule les fonctions f_1 et f_2, puis on en fait une somme tout en les multipliant chacune par leur coefficient respectif (α, β). Nous prendront soin ici de prendre en considération, la contrainte définit ci-dessus : si la contrainte n'est pas satisfaite, l'on affecte tout simplement à f une très grande valeur sinon elle prends la valeur calculée.

La résolution de ce problème se résume à la définition d'une fonction d'agrégation permettant le calcul de ($f = \alpha f_1 + \beta f_2$), et d'en faire varier les coefficient (α, β) afin de pouvoir visualiser les changement et résultats obtenus. Pour notre cas, les valeurs restent cohérentes lorsque les coefficients ont une valeur supérieure à 1.

2.2.2 Méthode $\in$-contrainte

En utilisant une méthode $\in$-contrainte, il va falloir optimiser f_1 avec f_2 sujette à la contrainte suivante : $f_2(x) \leq 1.2$ Afin d'utiliser la méthode $\in$-contrainte, il suffit de valider les conditions suivantes :

- Une seule fonction objectif est à optimiser (ici appellé f_c)
- Les autres fonctions objectifs sont sujettes à des contraintes.
- Minimiser $f_c(x)$ avec $x \in \omega$
- $f_i \leq \in_i$, 1,...,k avec $i \neq c$.

Dans cette méthode, une seule fonction objectif f_c est optimisée tandis que les autres fonctions objectifs sont sujettes à des contraintes.

2.3 Résultat

En utilisant la méthode d'agrégation , nous obtenons une solution égale à '1.0004378042795967' avec 2500080 évaluations et avec une taille de population de 100.

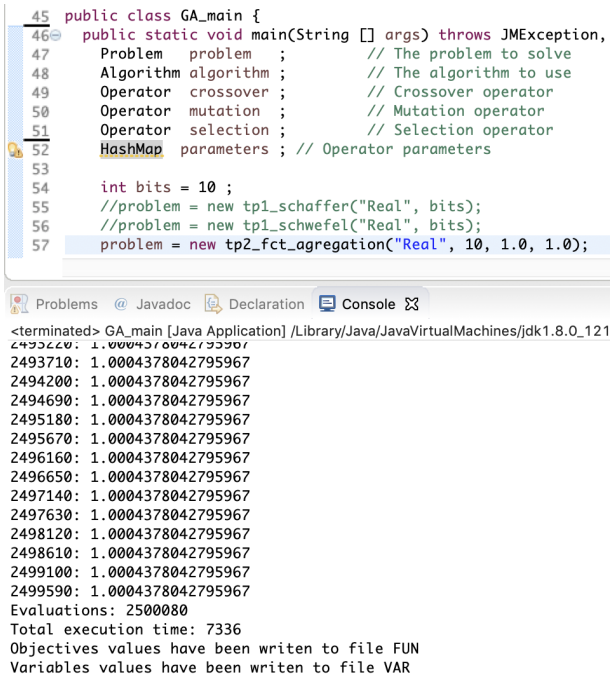

FIGURE 2.2: Simulation du problème multi-objectifs avec 2500080 évaluations et 100 populations

3. TP3 - Optimisation multi-objectifs

3.1 Introduction

NSGA-II est une version modifiée de l'algorithme NSGA[3]. C'est une approche rapide, élitiste et sans paramètres qui manipule une population de solutions et utilise un mécanisme explicite de préservation de la diversité. Le but étant de minimiser la fonction scalaire permettant une optimisation multi-objectifs en utilisant une approche Pareto.

3.2 Mécanismes de l'algorithme NSGA-II

Initialement, une population parent P_0 de N solutions (ou individus) est créée aléatoirement. Cette population est triée sur une base de non-dominance à l'aide d'un algorithme rapide. Ce tri associe un rang de dominance à chaque individu. Les individus non dominés ont un rang de 1 et constituent le front F1. Les autres fronts Fi sont ensuite définis récursivement en ignorant les solutions des fronts précédemment détectés. Ce tri est utilisé et illustré dans le cas d'un problème à deux objectifs (f1,f2) que nous avons étudié.

3.3 Approche Pareto

Les approches Pareto utilisent la notion de dominance pour comparer les solutions et leur affecter un score ou sélectionner des solutions. Elles sont devenues la principale approche employée pour résoudre les problèmes multi-objectif du fait de leur capacité à trouver un ensemble potentiellement efficace via la recherche menée sur une population de solutions.

En utilisant l'algorithme NSGA-II[6], il nous sera donc possible de résoudre ce problème en utilisant un calcul des fronts de Pareto. Dans notre cas, le Front de Pareto prend en compte simultanément de toutes les fonctions objectifs et introduit la notion de dominance entre f_1 et f_2. Le Front de Pareto permet ainsi de vérifier les deux facteurs suivant :

- Convergence
- Diversité

Dans ce TP, il suffisait d'ré-implémenter le code du TP2, avec quelques modifications majeures telles que le nombre de nombre d'objectifs ($numberOfObjectives$=2;) ainsi que les sorties adéquates à récupérer :

- solution.setObjective(0,f1) ;
- solution.setObjective(1,f2) ;

3.4 Résultat

Après traitement, nos 2 valeurs objectifs (f_1, f_2) seront par la suite stockées dans le fichier FUN. Ces données, nous servirons à tracer notre courbe de Front de Pareto via Excel. En réalisant cela, nous vérifions la cohérence et la similitude avec celui de l'énoncé. En effet, l'on recherche ici 10% des points sur le front de Pareto pour accepter que les résultats sont bonnes. Cela est bien le cas ici.

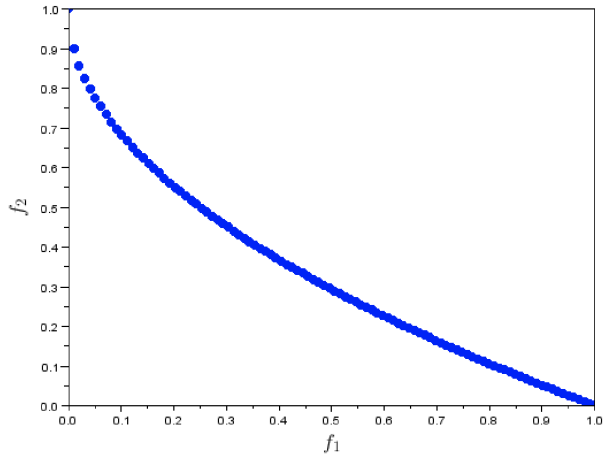

FIGURE 3.1: Front de Pareto pour $N = 10$

Conclusion

Grâce aux TP réalisés, il a été étudié les notions de l'optimation mono-objectif et multi-objectif(approches scalaires, approches Pareto).

Comme décrit ci-dessus, il existe deux types de méthodes d'optimisation, la première est l'optimisation mono-objectif, qui se base sur la minimisation (ou la maximisation) d'une seule fonction objective où le but est de trouver la meilleure solution appelée solution optimale qu'est facilement définie suivant une seule performance du problème étudié (temps de réponse, temps de monté, la robustesse, taux d'erreur,... etc). D'autre part, l'optimisation multi-objectif optimise simultanément plusieurs fonctions objectives qui sont souvent contradictoires, on cherche à trouver la meilleure solution suivant un ensemble de performance du problème (temps de réponse plus la robustesse, temps de réponse plus temps de monté plus la robustesse,... etc). Dans la même continuité d'études des optimisations possibles vu au cours de ces TPs, il serait intéressant d'envisager une suite d'étude sur les méthodes d'optimisation hybrides[2].

Le framework jMetal a permis l'exploration d'espace de recherche de différent problème dans le but de trouver une solution la plus optimale possible. Grâce aux simulations réalisées, nous savons maintenant qu'intensifier les recherches autour des bonnes solutions pourrait conduire à une perte de diversité et à une convergence prématurée. Il est donc important de trouver un juste équilibre entre diversité et convergence.

Bibliographie

[1] Wahabou Abdou. *Adaptive Multi-objective Genetic Algorithm Using Multi-Pareto-Ranking*. URL : http://w-abdou.fr/fr/publications.html.

[2] Université Biskra. *Etude sur les méthodes d'optimisation hybrides*. URL : http://thesis.univ-biskra.dz/2124/4/chapitre02.pdf.

[3] Clément Chatelain. *Optimisation multi-objectif pour la sélection de modèles SVM*. URL : https://hal.archives-ouvertes.fr/hal-00671129/document.

[4] Antonio J. Nebro. *jMetal Documentation*. URL : https://github.com/jMetal/jMetalDocumentation.

[5] Master UPMC. *Algorithmes évolutionnistes*. URL : http://www.isir.upmc.fr/files/2002ACTN353.pdf.

[6] Manolo Venturin. *Multi-objective optimization genetic algorithm*. URL : http://www.openeering.com/sites/default/files/Multiobjective_Optimization_NSGAII_0.pdf.

[7] Mouadh Yagoubi. *Optimisation évolutionnaire multi-objectif parallèle*. URL : https://tel.archives-ouvertes.fr/tel-00734108/document.