

Cahier des Charges

Edouard Fouassier - Maxime Gonthier - Benjamin Guillot
Laureline Martin - Rémi Navarro - Lydia Rodriguez de la Nava

Algorithme Genetique

11 mars 2018

Table des matières

1	Introduction	1
2	Fondement du projet	1
2.1	But du projet	1
2.2	Personnes et organismes impliqués dans les enjeux du projet	1
2.3	Utilisateurs du produit	1
3	Contraintes sur le projet	1
3.1	Contraintes non négociables	1
3.2	Glossaire et conventions de dénomination	2
3.3	Faits et hypothèses utiles	2
4	Exigences fonctionnelles	2
4.1	Portée du produit	2
4.2	Exigences fonctionnelles et exigences sur les données	2
5	Exigences non fonctionnelles	3
5.1	Ergonomie et convivialité du produit	3
5.2	Facilité d'utilisation et facteurs humains	3
5.3	Maintenance, support, portabilité, installation du produit	3
6	Organigramme	3
7	Autres aspects du projet	4
7.1	Estimation des coûts du projet	4
7.2	Manuel utilisateur et formations	5
8	Conclusion	5
9	Bibliographie	6
10	Annexe	7

1 Introduction

Le nom algorithme génétique vient des ressemblances avec le monde du vivant, il y a en effet des mutations, des générations et de la sélection naturelles pour n'en citer que quelques uns. Cet algorithme a été étudié par John Holland de 1960 à 1975 dans son ouvrage *Adaptation in Natural and Artificial System* inspiré entre autres par la “learning machine” de Turing.

L'algorithme génétique s'applique à une grande variété de domaines. Tout d'abord, il peut s'appliquer en génétique pour étudier l'évolution des gènes d'une espèce donnée sur plusieurs générations. On peut aussi appliquer l'algorithme dans le cas d'apprentissage comme par exemple apprendre à un robot à se déplacer en fonction des obstacles. Un troisième domaine d'application possible est l'optimisation. On peut citer dans cette catégorie l'optimisation de portefeuille d'action en fonction de leurs risques ou bien l'optimisation de la ventilation dans le cas de feu dans des espaces confinées ou bien simplement l'optimisation d'une fonction.

L'algorithme génétique est particulièrement utile lorsque l'utilisateur étudie une population et qu'il recherche une solution approchée parmi ces valeurs. Les algorithmes génétiques se démarquent des autres car ils sont très facilement adaptables à différents problèmes. On utilise des règles de transition probabiliste, ce qui est pertinent pour des problèmes où les résultats sont des valeurs approchées.

2 Fondement du projet

2.1 But du projet

L'objectif du projet est de réaliser un programme utilisant un algorithme génétique permettant de délivrer une solution ou un ensemble de solutions optimale(s) d'un problème donné à l'utilisateur en fichier de sortie.

2.2 Personnes et organismes impliqués dans les enjeux du projet

Le projet a pour client principal madame Leila Kloul.

2.3 Utilisateurs du produit

Le produit se veut se veut employable par n'importe qui cherchant une réponse adaptée à un problème d'optimisation lié à une fonction.

3 Contraintes sur le projet

3.1 Contraintes non négociables

Le logiciel doit permettre à l'utilisateur d'entrer les données de son problème dans une interface textuelle ou en sélectionnant un fichier texte contenant déjà celles-ci. De plus, l'algorithme génétique doit être générique afin de fournir une utilisation indépendante du problème à traiter. Enfin, l'utilisateur doit pouvoir obtenir en sortie un fichier montrant l'évolution de l'algorithme durant l'opération. Il doit pouvoir choisir entre les formats suivants : Xfig et/ou LaTeX et/ou PostScript.

3.2 Glossaire et conventions de dénomination

Nous utiliserons dans la suite du cahier des charges les termes suivants pour parler de l'algorithme génétique :

- Une **population d'individu** est un ensemble de solutions potentielles. C'est l'ensemble de données sur lequel l'application sera utilisée afin d'en tirer le meilleur résultat possible.
- Un **individu** ou un **chromosome** est une solution potentielle au problème donné.
- La **taille** d'un individu s'exprime en puissance de 2. C'est la taille de la solution potentielle au problème (elle peut s'exprimer sous la forme d'un tableau, d'une chaîne,...).
- Un **gène** est une partie de solution.
- L'**alphabet** est l'ensemble de caractères composant une solution (ex : 0,1 si on utilise des solutions codées en binaire)
- Une **génération** est une itération de l'algorithme génétique utilisé dans l'application. Elle correspond à une modification de l'ensemble de solution.
- La **fonction fitness** est la fonction que l'on cherche à optimiser dans l'application. Elle servira également à évaluer la qualité des solutions de l'ensemble.
- L'**algorithme génétique** est composé de ces trois étapes :
 - La **reproduction** correspond à la sélection de solutions de l'ensemble de base afin de les utiliser pour former de nouvelles solutions potentielles. Nous avons choisi une sélection par roulette. C'est à dire que les parents sont sélectionnés selon leurs scores. Plus leurs scores sont élevés, plus ils ont de chance d'être choisis. Il faut imaginer une roulette de casino où l'on lance une bille à chaque itération et on choisi l'individu sur lequel la bille est tombée. L'avantage de cette sélection est qu'elle est preservative, cela signifie que même les individus les moins performants conservent une chance d'être choisis. Cela empêche une disparition totale prématurée des individus aux scores les plus faibles.
 - La **mutation** correspond à une modification ponctuelle d'une solution.
 - Le **crossover** est le croisement de solutions utilisé pour en créer de nouvelles.

3.3 Faits et hypothèses utiles

Il est impossible de produire un programme totalement générique, c'est pourquoi nous limitons ici l'utilisation du logiciel aux problèmes d'optimisation et de prédiction. Pour l'utiliser dans des problèmes d'apprentissage, il faudrait le coupler avec un autre programme comme un système de classifieur, ou encore un réseau de neurone, ce qui n'est pas le cas dans ce projet.

4 Exigences fonctionnelles

4.1 Portée du produit

Il n'y a qu'un seul type d'utilisateur (le client). Le programme est générique, et donc applicable dans toutes sortes de situations.

4.2 Exigences fonctionnelles et exigences sur les données

On doit proposer une interface graphique permettant de rentrer les données du problème. De plus il faut trouver un ensemble de solution à ce problème et les afficher dans un fichier Latex avec une représentation de l'évolution des générations (grâce Xfig).

5 Exigences non fonctionnelles

5.1 Ergonomie et convivialité du produit

Le produit devra avoir une interface intuitive et lisible pour faciliter son utilisation et les fichiers de sortie devront être simple à comprendre.

5.2 Facilité d'utilisation et facteurs humains

L'utilisation du logiciel ne doit nécessiter aucun pré-requis à l'utilisateur autre que la connaissance du problème auquel il cherche une solution. Le logiciel doit ainsi procéder à une vérification de tous les champs remplis.

5.3 Maintenance, support, portabilité, installation du produit

Le logiciel se veut simple à installer sous Linux. Il sera aussi fait en sorte que l'on puisse ajouter de nouveaux module au programme simplement.

6 Organigramme

Voir Organigramme page annexe (Page 7)

Les données initiales sont : taux de mutation, identifiant plus tableau pour le test, fonction fitness, nombre de génération, score à atteindre sur la fonction, taille individu, critère d'évaluation, nombre de critères, alphabet, taille population, probabilité de crossover, nom de fichier.

Interface :

- Affichage des différents champs de saisie puis des résultats
- Réception des valeurs (Données initiales)
 - par fichier
 - par saisie des champs de l'interface
- Option d'arrêt pendant l'exécution de l'AG : on peut arrêter le programme à tout moment et récupérer les résultats jusqu'à l'arrêt

Gestion de la validité des données :

- Test des valeurs reçues
- Enregistrement des valeurs dans un fichier

Module d'initialisation :

- Transformation de valeurs en paramètres
- Creation de la population initiale
- Lecture du fichier des données

Evaluation de la population :

- Evaluation de chaque individus et attribution d'un score

Algorithme Evaluation : paramètre : une population

Si le nombre de critères == 1

Si l'option d'évaluation du critère == '>'

Tri de la population puis attribution du score en fonction du rang après le tri
Sinon Pour chaque individu de la population faire :
Utilisation de la fonction fitness, le score correspond au résultat renvoyé

Complexité dans le pire des cas : $O(n^2)$ (n : taille de la population)

Tests d'arrêts :

- Test de convergence de la fonction fitness
- Test du nombre de générations

Génération de la nouvelle population :

- Sélection par roulette
- Cross-over/mutation
- Création de la nouvelle population
- Génération de nombres aléatoires

Algorithme Roulette : paramètre : les individus

si le nombre d'individus est pair

limite = population/2

sinon

limite = population-1/2

scoremax = $\sum score$

Pour chaque individus on calcule sa valeur (en pourcentage) en fonction du scoremax

Tant qu'on a pas assez d'individu sélectionné

Si le pourcentage du premier individu est supérieur à une valeur aléatoire
il est sélectionné

Pour chaque individu sauf le premier faire :

si (le pourcentage de l'individu $>$ à une valeur aléatoire

ET que le pourcentage de celui qui le précède est inférieur à cette valeur)
cet individu est sélectionné.

On retourne les individus sélectionnés

Complexité dans le pire des cas : $O(n^2)$

Gestion d'entrées sorties :

- Calcul des statistiques de l'exécution : les données d'entrées et les résultats
- Ecriture des résultats dans latex
- Ecriture des résultats dans PostScript
- Ecriture des résultats dans Xfig
- Lecture d'un fichier d'entrée avec les données
- Ecriture des valeurs dans un fichier

Dans tous les modules il faut une fonctionnalité qui lit les valeurs nécessaires dans le fichier des données.

7 Autres aspects du projet

7.1 Estimation des coûts du projet

Module	Nombre de lignes	Temps	Affectation
Interface	250-300	5-6 heures	Rémi et Edouard
Gestion de la validité des données	50-60	1-2 heures	Maxime et Laureline
Initialisation du programme	transformation des valeurs en paramètres : 30 création de la population initiale : 40 lecture du fichier de données : 40	2-3 heures	Edouard et Rémi
Evaluation de la population	25-30	1-2 heures	Lydia et Ben
Tests d'arrets	par itération ou forcé : 40 par convergence : 70	2-3 heures	Laureline et Maxime
Generation de la nouvelle population	40-50	2-3 heures	Ben et Lydia
Gestion d'entrées sorties	200	4-5 heures	tout le monde
Coût Total	745-820	14-24 heures	

7.2 Manuel utilisateur et formations

Un manuel d'utilisation sera fourni pour conseiller à l'utilisateur les données les plus performantes pour avoir des solutions plus efficaces.

8 Conclusion

Pour résoudre les exigences du client, nous avons élaboré un cahier des charges qui décrit les différents dispositifs nécessaire pour conceptualiser un algorithme génétique générique. Dans ce cahier des charges, nous proposons une solution qui permettra au client de simuler des problèmes variés, tels que des problèmes génétiques, mathématiques, d'optimisation, etc., dont les solutions sont approchées et obtenir les résultats pour chaque générations. Ce sujet nous intéresse tout particulièrement. De plus, l'écriture d'un cahier des charges est un travail inédit pour nous, il nous a permis de travailler en groupe de six personne pour la première fois, cela nous a appris à développer une organisation particulière.

Le langage qui nous semble le plus adapté est le C++. Nous choisissons un langage hybride. D'une part, nous aurons besoin d'un langage objet pour implémenter les individus. En effet, un individu sera un objet caractérisé par plusieurs champs. Les individus formeront une population qui sera elle aussi un objet. D'autres part, nous aurons besoin d'un langage procédural afin d'effectuer des calculs sur les individus. Le C++ nous semble également avantageux dans l'utilisation d'un parser (utilisé avec la fonction fitness). Pour cela nous avons fait le choix de prendre un parser

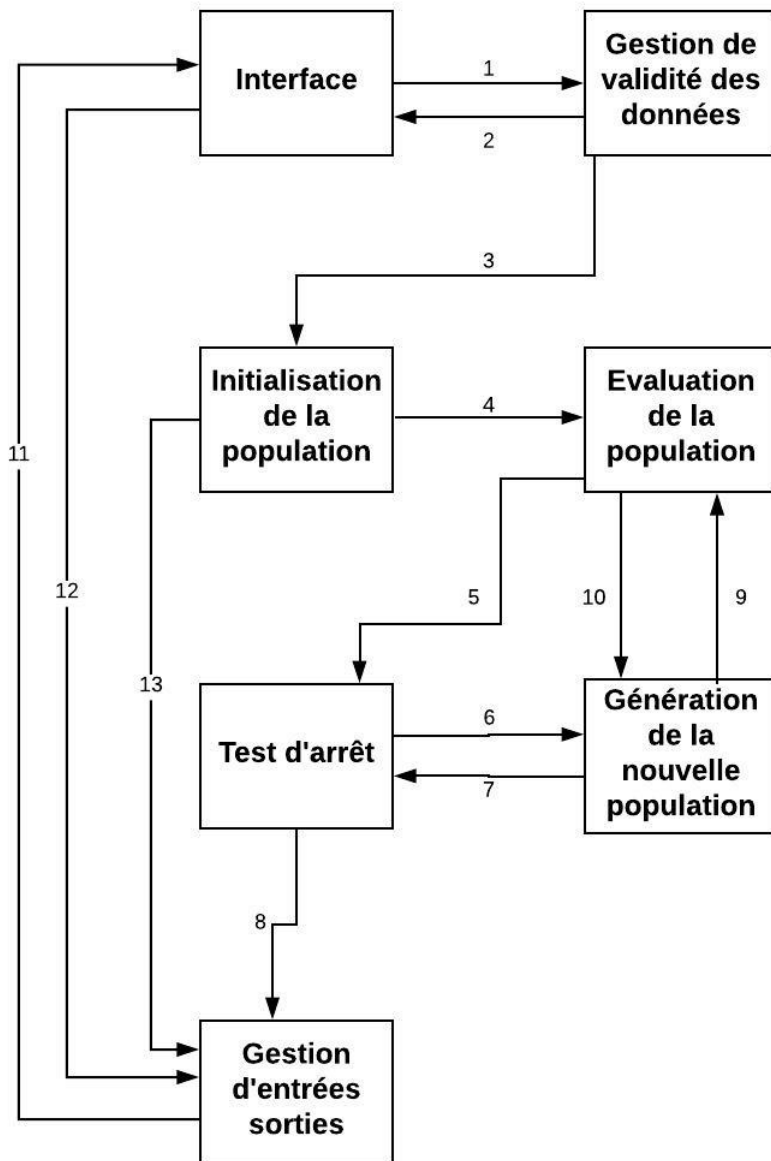
déjà existant. En effet le travail à fournir pour écrire notre propre parser nous aurait demandé un temps long et fastidieux ainsi qu'un grand nombre de lignes de codes pour une seule fonction. Nous avons estimé qu'en utilisant un parser existant nous pourrions ainsi mettre à profit le temps gagné pour travailler au mieux notre application.

9 Bibliographie

- Présentation des algorithmes génétiques et de leurs applications en économie :
http://deptinfo.unice.fr/twiki/pub/Minfo04/IaDecision0405/Prsentationdesalgorithmesgntiquesetdeleursapplicationsenconomie_P.pdf
- La page wikipedia sur l'algorithme génétique :
https://fr.wikipedia.org/wiki/Algorithme_g%C3%A9n%C3%A9tique
- Tutoriel Python sur les algorithmes génétiques :
<https://skyduino.wordpress.com/2015/07/16/tutorielpython-les-algorithmes-genetiques-garantis-sans-ogm/>
- Etude sur les algorithmes génétiques :
http://souqueta.free.fr/Project/files/TE__AG.pdf
- Exemple d'utilisation sur un jeu vidéo :
<https://www.youtube.com/watch?v=9tCySY6TLk8>
- Vidéos d'explications de l'algorithme génétique :
<https://www.youtube.com/watch?v=VOIPosVDkEE>
<https://www.youtube.com/watch?v=bLdrDnZfpPo>
<https://youtu.be/mAybMmf0veY>
- L'application des algorithmes génétique dans la science des données :
<https://www.analyticsvidhya.com/blog/2017/07/introduction-to-genetic-algorithm/>
- Explications de l'aglorithme générique :
<http://khayyam.developpez.com/articles/algo/genetic/>
<http://informatique.coursgratuits.net/methodes-numeriques/algorithmes-genetiques.php>
- Etude sur l'inertage de feu :
<http://www.ligeron.com/telecharger.php?id=11>
- Comparaison de l'algorithme génétique et du réseau de neurones :
<https://stackoverflow.com/questions/1402370/when-to-use-genetic-algorithms-vs-when-to-use-neural-networks>

10 Annexe

Organigramme :



Légende :

1. Nom fichier
2. Message d'erreur
3. Nom fichier
4. Population initiale
5. Population initiale
6. Population n- 1
7. Nouvelle population générée
8. Population
9. Population sans score
10. Population avec score
11. Nom du fichier
12. Nom du fichier
13. Nom du fichier