Cahier des Charges

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

Algorithme Genetique

5 mars 2018

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 un. 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.

1 Fondement du projet

1.1 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.

1.2 Personnes et organismes impliqués dans les enjeux du projet

Le projet a pour client principal madame Leila Kloul.

1.3 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.

2 Contraintes sur le projet

2.1 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.

2.2 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.
- La **reproduction** correspond à la sélection de solutions de l'ensemble de base afin de les utiliser pour former de nouvelles solutions potentielles.
- La **Mutation** correspond à une modification ponctuelle d'une solution.
- Le **crossover** est le croisement de solutions utilisé pour en créer de nouvelles.

2.3 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.

3 Exigences fonctionnelles

3.1 Portée du produit

3.2 Exigences fonctionnelles et exigences sur les données

4 Exigences non fonctionnelles

4.1 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.

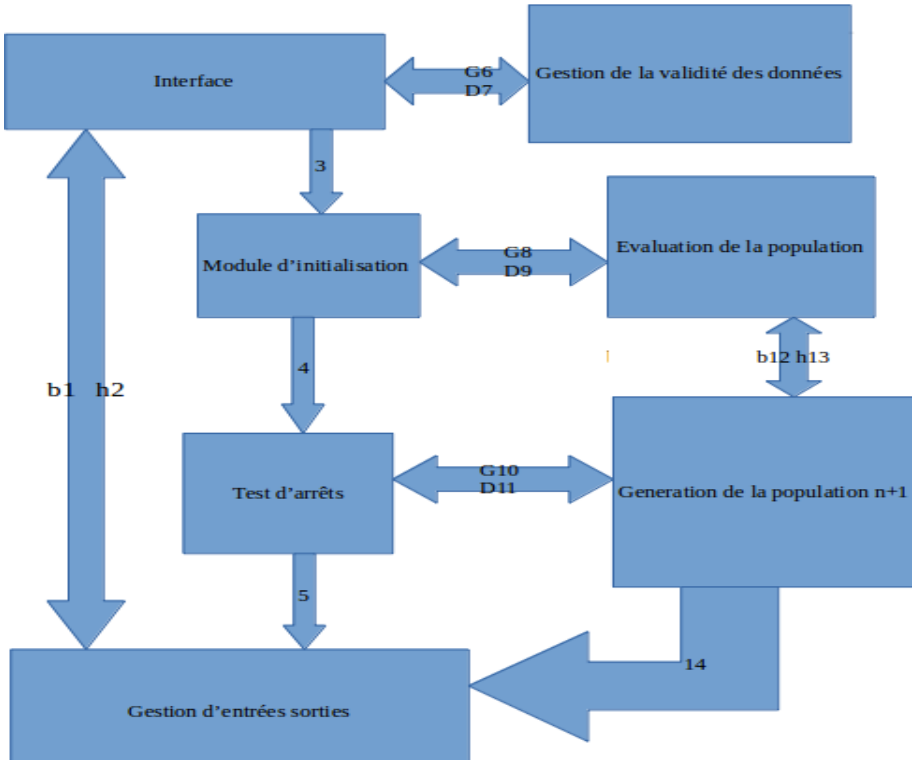
4.2 Facilité d'utilisation et facteurs humains

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.

4.3 Maintenance, support, portabilité, installation du produit

L'installation du logiciel devra être simple.
L'ajout ou la modification de module ne devra pas necessiter un de gros changement.
Le produit ne sera dans un premier temps developper que pour linux.

5 Organigramme



1. Fichier	8. Population
2. Fichier	9. Population + score
3. Données initiales	10. Génération de base 1 (pop size, t mutation, t cross over)
4. Génération 0 et Données initiales	11. Nouvelle génération après regénération (génération n+1)
5. Génération n + nom fichier	12. Population
6. Données initiales	13. Population + score
7. Données initiales	14. Génération n+1 + nom fichier

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
— Option de sauvegarde/chargement

Gestion de la validité des données :
— Test des valeurs reçues

Module d'initialisation :
— Transformation de valeurs en paramètres
— Creation de la population initiale

Evaluation de la population :
— Evaluation de chaques individus et attribution d'un score
Algorithme Evaluation :

Tests d'arrets :
— Test de convergence de la fonction fitness
— Test du nombre de générations
— Test de demande d'arrêt utilisateur

Génération de la nouvelle population :
— Selection par roulette
—
Algorithme Roulette :

Algorithme Crossover :

Algorithme Mutation :

Gestion d'entrées sorties :
— Description du problème
— Ecriture du paramètres
— Ecriture de la solution
— Calcul des statistiques de l'execution
— Ecriture des résultats dans un fichier

6 Autres aspects du projet

6.1 Estimation des coûts du projet

Module	Nombre de lignes	complexité	Affectation
Interface			
Gestion de la validité des données			
Evaluation de la population			
Tests d'arrets			
Generation de la nouvelle population			
Gestion d'entrées sortie			
Coût Total	**XXX**		

6.2 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.

6.3 Choix du langage

Conclusion

Bibliographie