Des algorithmes génétiques pour générer des modèles diversifiés

Mémoire de fin d'étude

Master *Sciences et Technologies*,

Mention *Informatique*,

Parcours Architecture et InGénierie du Logiciel et du wEb

Auteur
Florian Galinier

Superviseurs
Clémentine Nebut
Éric Bourreau
Annie Chateau
Adel Ferdjoukh

Lieu de stage
LIRMM UM5506 - CNRS, Université de Montpellier

soutenu publiquement le 15 juin 2016

Résumé

La génération de modèles à partir de méta-modèles est une solution apportée par l'ingénierie des modèles pour tester les transformations de modèles ou pour la validation de méta-modèles. Il existe dans la littérature un certain nombre de techniques combinatoires pour l'instanciation de méta-modèle, mais peu se sont intéressées au problème de la diversité. L'utilisation des algorithmes génétiques est une des plus prometteuses en terme de diversité. Durant nos travaux, nous avons expérimentés deux approches, basées sur l'algorithme évolutionnaire NSGA-II, pour la génération de nouveaux modèles, plus diversifiés. Les résultats ainsi obtenus mettre en évidence une réelle augmentation de la distance entre modèles, permettant de fournir des bancs d'essais plus réalistes et plus complets que ceux initiaux.

Abstract

Generation of models that conform to a meta-model is one of the solutions from model driven engineering for models transformations testing or meta-models validation. Some combinatorial techniques have been explored in previous works, but few have focused on problem of diversity. Amongst previous works, genetic algorithms usage seems to be one of the most promising for diversity generation. In this work, we experiment two approach, based on the evolutionnary algorithm NSGA-II, to generate more diverse models. These approaches provide results with an increased distance, enabling creation of more realistic and complete benchmark.

Remerciements

Je tiens tout d'abord à remercier Clémentine Nebut, Éric Bourreau, Annie Chateau ainsi qu'Adel Ferdjoukh pour leur encadrement et leurs nombreux conseils, qui ont permis la réalisation de cette étude. Ce stage m'a ainsi permis de m'ouvrir encore un peu plus au monde de la recherche, ce qui m'a encore une fois conforté dans mon objectif professionnel.

Merci à Jimmy Lopez et Clément Simon, pour tous les échanges fructueux autour d'un café qui ont permis de d'éclaircir certains points. Je remercie aussi Jocelyn Thiebaut, notamment pour ses conseils et astuces pour les présentations LaTeX. Merci à eux également pour tous les débats, pas toujours scientifiques, qui ont émaillés le stage et l'étude bibliographique.

Merci également à Marjorie, pour ses nombreuses relectures et pour son soutien tout au long du stage.

Table des matières

Table des figures

1

Introduction

L'ingénierie dirigée par les modèles (aussi nommée *IDM* ou encore *MDE* pour *Model-Driven Engineering*) est une méthode de développement logiciel qui place les modèles au cœur même du cycle de vie d'une application, accompagnant non plus seulement la phase de conception mais également la phase d'implémentation, de maintenance, de validation, etc.

Les modèles sont des représentations – des modélisations – de problèmes, à but initialement documentatif, dont des exemples parmi les plus connus sont sans doute les modèles entités-relations ou encore les modèles objets.

L'ingénierie dirigée par les modèles propose des outils pour la systématisation et l'automatisation de l'utilisation des modèles. Les transformations de modèles sont ainsi largement utilisées aujourd'hui, que ce soit dans un contexte d'évolution et de maintenabilité, ou dans un contexte de développement plus classique, et permettent la réalisation de nombreuses étapes du cycle de vie d'un logiciel (génération de code source, rétro-ingénierie, etc.). Les transformations sont par exemple un moyen de passer d'un modèle d'un domaine donné (par exemple un diagramme de classe *UML*) à un autre modèle dans un autre domaine (modèle *Java* par exemple, avec une génération possible de code source). Celles-ci s'appuient notamment sur le concept de méta-modèle.

Un méta-modèle est lui-même un modèle, à un niveau d'abstraction supérieur, associant concepts et relations entre concepts. Ainsi, un méta-modèle permet de décrire un

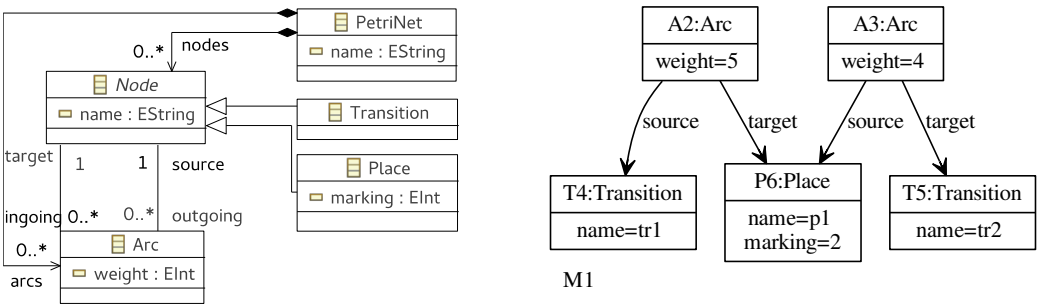


FIGURE 1.1 – Méta-modèle de réseau de Petri extrait de [1] et exemple de modèle instance de ce méta-modèle

domaine d'ingénierie particulier, et les modèles instances de ce méta-modèle permettent de spécifier un problème particulier de ce domaine (*e.g.* le méta-modèle *UML* permet de créer des modèles *UML* qui sont des représentations des problèmes de l'ingénierie logicielle). Ainsi, les méta-modèles sont une représentation de la syntaxe des modèles à la manière des grammaires des langages.

Dans l'exemple donné fig. 1.1, un méta-modèle de réseau de Petri est présenté ainsi qu'une instance de celui-ci. Cette dernière est représentée en syntaxe abstraite, i.e. comme les instances de son méta-modèle, et nous ne nous intéresserons qu'à celle-ci dans ce sujet. En effet, la représentation en syntaxe concrète, i.e. dans une syntaxe graphique définie pour un modèle donné (dont un exemple est donné fig. 2.6), ne nous intéresse pas dans ce cas, l'objectif étant l'instanciation de méta-modèle.

Les méta-modèles sont également des instances d'un niveau d'abstraction supérieur, les méta-méta-modèles (cf. fig. 1.2). Ces derniers sont auto-descriptifs, i.e. ce sont des instances d'eux-mêmes et permettent ainsi de décrire des langages de modélisation (*MOF - Meta-Object Facility* par exemple est le méta-méta-modèle utilisé pour l'ensemble des méta-modèles définis par l'*Object Management Group (OMG)*, dont UML).

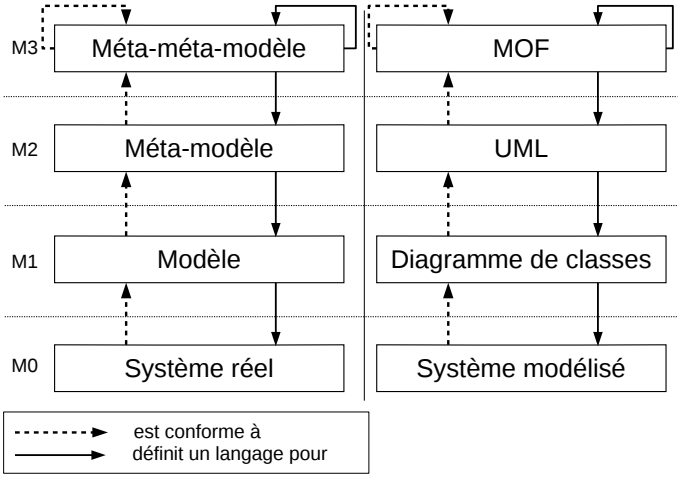


FIGURE 1.2 – Les quatre niveaux d'abstractions en modélisation : M0 : le système réel ; M1 : les modèles ; M2 : les méta-modèles ; M3 : les méta-méta-modèles.

L'instanciation automatique de modèles à partir d'un méta-modèle, aussi appelée génération de modèles, est une solution apportée à des problèmes actuels de l'IDM. Ainsi, dans le cas des transformations de modèles, la validité de celles-ci peut être établie par des tests. Cependant, les données manipulées étant des modèles, il est beaucoup plus complexe d'établir des stratégies de tests ([2]). L'instanciation manuelle de nombreux modèles nécessaires pour les tests est ainsi une tâche longue et fastidieuse. Une automatisation de ce processus permet l'accélération de l'écriture des tests et par conséquent une importante réduction de leurs coûts.

La génération de modèles instances d'un méta-modèle est en outre un moyen de tester la validité de ce méta-modèle ; il permet en effet de vérifier si un méta-modèle est instanciable et, par conséquent, valide. Un méta-modèle peut par exemple ne pas être instanciable s'il possède des contraintes *OCL* (*Object Constraint Language* ; langage

de contraintes pour UML défini par l'Object Management Group) contradictoires qui lui sont associées ou s'il est mal défini, par exemple au niveau des cardinalités des associations. Dans la fig. 1.3, le modèle n'est par exemple pas instanciable ; en effet, les relations entre A et B ainsi qu'entre B et C indiquent que chaque instance de A (resp. de B) devrait être en relation avec une unique autre instance de B (resp. de C), or la relation entre A et C impliquerait l'existence de deux instances de C en relation avec l'instance de A (une instance de A ne pouvant pas être reliée deux fois à la même instance de C). Il y a donc un paradoxe et ce modèle n'est par conséquent pas instanciable.

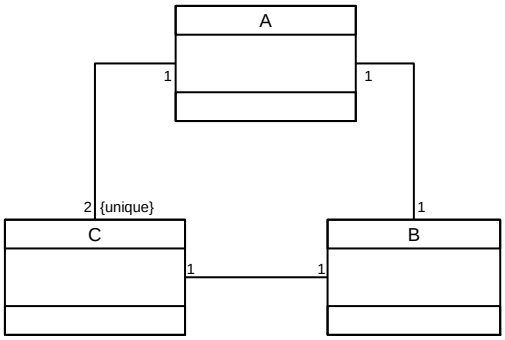


FIGURE 1.3 – Exemple de modèle non instanciable

La recherche de génération de modèles conformes (et son impossibilité) est une approche permettant la validation (ou l'invalidation) du meta-modèle. Quand il est possible de générer des modèles conformes, une propriété importante se situe dans la diversité de ces modèles générés. L'instanciation de modèles est en effet elle-même difficile, et on souhaiterait obtenir dans la plupart des cas des modèles les plus diversifiés possibles, afin d'obtenir la plus grande couverture du méta-modèle possible – un échantillon le plus représentatif et couvrant le plus grand nombre de cas différents possible. Cela implique un certain nombre de difficultés ; en effet, il faut pouvoir définir la diversité et, par conséquent, maintenir une certaine distance entre plusieurs modèles. Par ailleurs, les modèles ainsi générés doivent rester valides, notamment respecter les éventuelles contraintes *OCL* associées au modèle ; la détermination de cet ensemble de modèles valides est un problème combinatoire.

Des techniques combinatoires ont déjà été utilisées pour la génération de modèles. Nous présenterons quelques unes de ces approches. Les approches par *CSP* (*Constraint Satisfaction Problem* – [1, 3, 4]) ou par recuit simulé ([5]) ont par exemple été utilisées pour l'instanciation de méta-modèles. L'approche par algorithme génétique proposée dans [6] est une de celles qui semble produire des résultats très intéressants en terme de diversité, et le but de ce stage étant la production de modèles diversifiés nous présenterons ces algorithmes.

Le problème de la définition de la diversité dans le cadre des modèles sera également abordé, et nous étudierons les différentes mesures existantes dans le cadre des graphes (les modèles pouvant être représentés comme tels) ainsi que des mesures de distances plus classiques.

Le prochain chapitre sera par conséquent consacré à un état de l'art de ces trois axes. Le chapitre 3 présentera la représentation des modèles que nous avons utilisée ainsi que les différentes pistes explorées. Les résultats des différentes expérimentations ainsi que leurs discussions sont présentés chapitre 4. Enfin, nous conclurons ce rapport.

2

État de l'art

Notre travail s'articule autour de trois axes (cf. fig. 2.1). Nous avons exposé lors de l'étude bibliographique les différentes techniques combinatoires déjà existantes pour la génération de modèles, ainsi que les méta-heuristiques évolutionnaires et différentes mesures de distance possible pour les modèles. Nous rappelons dans ce chapitre les travaux utilisés lors du stage.

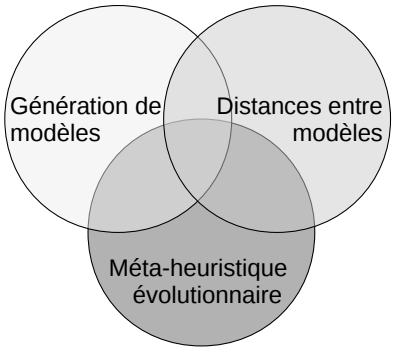


FIGURE 2.1 – Différentes problématiques du sujet.

2.1 Techniques combinatoires pour la génération de modèles

Plusieurs techniques combinatoires ont déjà été utilisées afin de générer des modèles. La génération de modèles à partir de CSP (programmation par contraintes) en est un exemple ([1, 3, 4]).

Les CSP sont définis par un triplet (X, D, C) où :

- $X = x_1, x_2, ..., x_n$ est l'ensemble des n variables du problème ;
- $D = d_1, d_2, ..., d_n$ l'ensemble des domaines tel que $\forall x_i \in X, d_i$ est le domaine de x_i ;
- C est l'ensemble des contraintes définies sur X.

Les solveurs CSP ont ainsi pour rôle d'associer pour chaque variable appartenant à X une valeur de leur domaine telle que les contraintes soient respectées. Un vecteur de valeurs $(v_1, ..., v_n)$ avec $\forall i, v_i \in d_i$ est ainsi un modèle particulier. Cette approche permet en outre la prise en compte des contraintes OCL de modèles en les incorporant dans les contraintes du CSP.

Dans [7], Sen et al. utilisent le langage de contraintes Alloy ([8]) pour la génération de modèles. Le méta-modèle est ainsi exprimé dans le langage qui permet de générer une formule logique en *FNC* (*Forme Normale Conjonctive*), et un solveur SAT permet d'obtenir des instances du méta-modèle d'entrée.

Cependant, peu de ces travaux ont abordé le problème de la diversité.

Dans [5], les auteurs proposent une approche de génération de modèles basée sur l'algorithme de recuit simulé (*SA - Simulated Annealing*). L'algorithme est utilisé afin de sélectionner un ensemble de modèles satisfaisant au mieux une fonction objectif (de nouveaux modèles sont générés aléatoirement – ici aussi le langage Alloy et un solveur SAT sont utilisés pour générer les nouvelles solutions – ou supprimés à chaque itération). Cette fonction objectif combine une mesure de dissimilarité du modèle avec les modèles existants ainsi que de couverture du méta-modèle.

Cependant, l'algorithme de recuit simulé ne travaillant que sur une unique fonction objectif, Cadavid et al. ont dû créer une fonction objectif qui est une somme des deux objectifs souhaités (couverture et dissimilarité) pondérés chacun par un poids. Ils obtiennent par conséquent des résultats biaisés par le choix de ces poids (ils obtiennent en effet de meilleurs résultats de couverture que de diversité, ayant donné un poids prépondérant à celle-ci).

Les algorithmes génétiques et plus particulièrement les *MOEA* (*Multi-Objective Evolutionary Algorithm*) semblent par conséquent plus adaptés dans les cas où l'on souhaite définir plusieurs fonctions objectif.

2.2 Algorithmes génétiques

Les algorithmes génétiques ([9, 10]) sont des méthodes méta-heuristiques inspirées du principe darwiniste de la théorie de l'évolution. Il existe un vocabulaire associé à ces algorithmes et nous donnons un certain nombre de définitions par la suite.

Définition 1 *Un **chromosome** (ou individu) est une représentation d'une solution possible de l'espace des solutions, constitué d'un ensemble de gènes (dans [9], Holland décrit un chromosome comme une chaîne de bits).*

Définition 2 *Un **gène** est un élément atomique permettant le codage de la représentation du chromosome (e.g. un bit dans une chaîne de bits).*

Définition 3 *La **population** est l'ensemble des solutions considérées à un moment donné.*

Les algorithmes génétiques ont pour objectif d'explorer une partie de l'ensemble des solutions possibles afin de déterminer une solution la plus adaptée (i.e. une solution obtenant le meilleur score à une fonction objectif). Pour ce faire, ils fonctionnent à partir d'une population initiale de *chromosomes*, eux-mêmes composés de *gènes*.

Définition 4 *Un **croisement** de chromosomes est une fonction qui permet d'obtenir de nouveaux chromosomes composés des gènes des chromosomes d'entrée.*

Définition 5 *Une **mutation** est une fonction qui permet d'obtenir un nouveau chromosome en modifiant un ou plusieurs gènes d'un chromosome déjà existant.*

Définition 6 *La **progéniture** est l'ensemble des nouveaux chromosomes obtenus par les processus de croisements et mutations.*

Afin d'explorer l'espace des solutions, des chromosomes sont sélectionnés dans la population existante et de nouveaux individus sont créés par des opérations de croisements et de mutations (cf. fig. 2.2), créant ainsi une nouvelle population (la progéniture). Les croisements permettent de conserver les gènes permettant d'obtenir de bons scores chez les parents, tandis que les mutations permettent d'explorer l'espace des solutions mais également d'éviter une convergence trop rapide en explorant des solutions plus « lointaines », et maintiennent ainsi une certaine diversité.

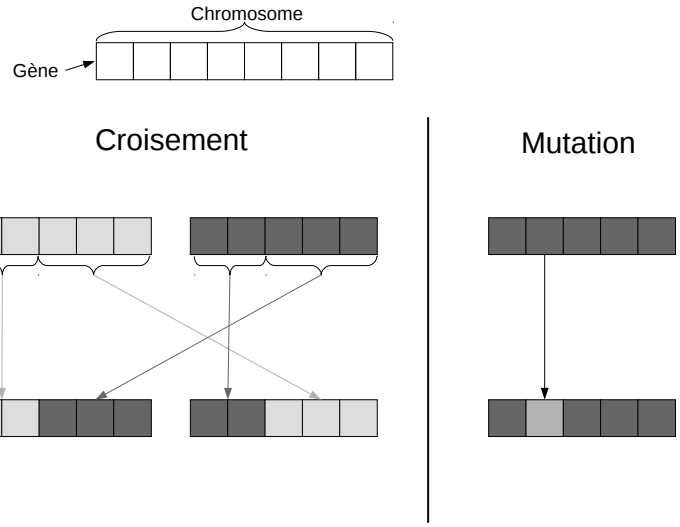


FIGURE 2.2 – Exemple de croisement et de mutation sur des chromosomes. À gauche, croisement en un point (translocation) de deux chromosomes ; à droite, mutation d'un chromosome.

Définition 7 *Une **génération** n est l'ensemble des chromosomes obtenus lors de la* $n^{ième}$ *itération.*

Définition 8 *L'**élitisme** consiste en la conservation des meilleurs individus de la génération précédente (afin d'éviter de perdre les chromosomes obtenant les meilleurs scores).*

Les individus les plus forts de la progéniture (i.e. ceux possédant le meilleur score à la fonction objectif) sont ensuite sélectionnés pour former la nouvelle génération (les algorithmes *élitistes* conservent également les individus les plus forts de la génération

précédente). Plusieurs générations sont ainsi créées, jusqu'à obtention d'une population finale (cf. algorithme 1). Cette population finale est obtenue quand un nombre d'itérations maximum est atteint, quand un délai est écoulé ou encore lorsqu'un critère de qualité est atteint. Celle-ci sera alors composée d'individus ayant de meilleurs scores à la fonction objectif que les individus initiaux.

Algorithme 1 : Principe général de fonctionnement des algorithmes génétiques

```
P ← population initiale;
tant que les conditions de terminaisons ne sont pas atteintes faire
    O ← sélection des individus pour les croisements;
    O ← croisements et mutations de O;
    évaluer O;
    P ← sélection des meilleurs individus de O;
fin
```

Dans de nombreux cas réels cependant, ce sont généralement plusieurs objectifs que l'on cherche à atteindre et non uniquement un seul. Le problème du voyageur-acheteur est par exemple un problème multi-objectif proche du problème du voyageur de commerce. Le voyageur doit visiter différentes villes pour acquérir un ensemble d'objets – chaque ville ne proposant ni les mêmes objets, ni les mêmes prix – tout en minimisant à la fois la distance parcourue et les dépenses. Les algorithmes multi-objectifs (possédant plusieurs fonctions objectif) sont une réponse apportée à ces problèmes ; ils tentent d'approcher une solution qui obtient un score qui est un compromis entre les différents objectifs fixés. Pour cela, la sélection des individus est basée sur le principe de dominance et non plus sur les meilleurs scores.

Définition 9 *Un individu x_1* ***domine*** *un individu x_2 si x_1 obtient des résultats au moins aussi bons que x_2 pour toutes les fonctions objectifs et obtient un meilleur score dans au moins une fonction objectif (cf. eq. 2.1 – où F est l'ensemble des fonctions objectifs et dans le cas où on cherche à minimiser le score de la fonction objectif).*

$$\begin{aligned} &\forall f_i \in F, f_i(x_1) \leq f_i(x_2) \wedge \\ &\exists f_i \in F, f_i(x_1) < f_i(x_2) \end{aligned} \tag{2.1}$$

Définition 10 *La* ***sélection par tournoi binaire*** *est un mécanisme de sélection (notamment utilisé pour créer des couples pour les croisements). Deux individus sont sélectionnés au hasard dans la population ; leurs scores sont comparés, et le meilleur d'entre eux est sélectionné. Ce processus est ensuite répété afin d'obtenir autant de chromosomes que nécessaire.*

2.2.1 Approche évolutionnaire choisie

Un certain nombre d'algorithmes multi-objectifs ont été étudiés dans l'étude bibliographique de ce stage. Nous avons fait le choix d'utiliser l'algorithme multi-objectif *NSGA-II* (choix discuté section 3.3.1). Nous donnons ici pour rappel son étude.

NSGA-II et NSGA-III

L'algorithme *NSGA-II*[1] (*nondominated sorting genetic algorithm II* – proposé par Deb et al. dans [11]) est un algorithme génétique multi-objectif originellement conçu afin de déterminer un optimum de Pareto, i.e. un état dans lequel la population ne peut plus être améliorée sans entrainer une dégradation d'une partie des individus, répondant à de multiples objectifs. Cet algorithme est un successeur plus performant de *NSGA* (l'algorithme *NSGA* possédant une complexité $O(MN^3)$ contre $O(MN^2)$ pour *NSGA-II* – avec M le nombre de fonctions objectifs et N la taille de la population), et utilise une nouvelle façon de préserver la diversité.

L'algorithme *NSGA-II* s'articule autour d'une mécanique assez classique d'algorithme génétique ; c'est un algorithme élitiste qui effectue une sélection par tournoi afin de choisir les individus à croiser et muter pour générer les nouvelles populations.

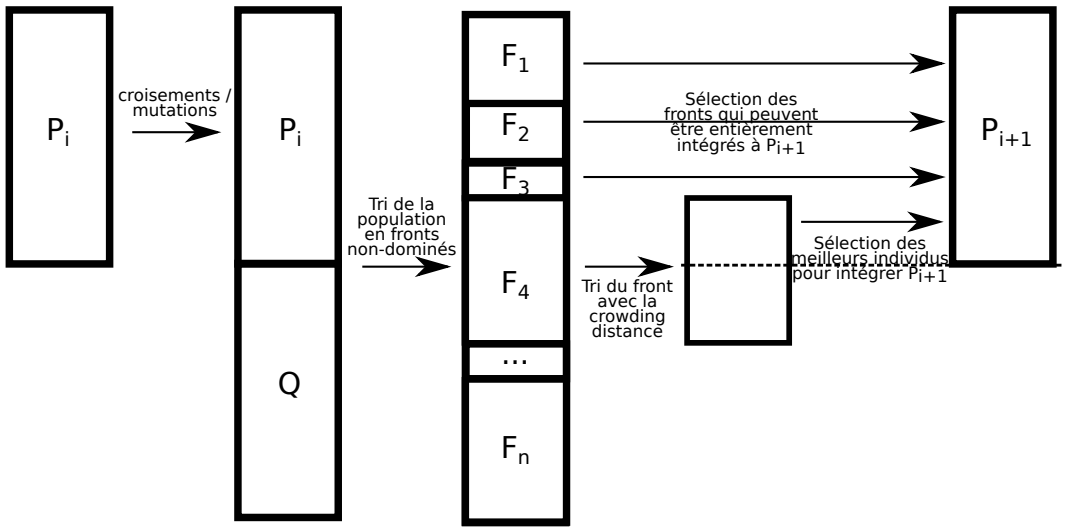


FIGURE 2.3 – Procédure de NSGA-II.

La principale nouveauté de cet algorithme est l'intégration d'un processus de conservation des distances : la *crowding distance* (cf. fig. 2.3). Lorsque la progéniture a été créée, les deux populations (parents et progénitures) sont regroupées et ordonnées selon leurs scores en différents fronts (les individus non dominés sont dans le front 1, ceux dominés uniquement par ceux du front 1 sont dans le front 2, etc.).

Chaque front est ainsi sélectionné (par ordre croissant) jusqu'à ce qu'un front ne puisse pas intégrer entièrement la population ($|F_i| > |P| - \sum_{j=0}^{i-1} |F_j|$ – avec F_i le front considéré et P la population). Ce front est alors trié selon la *crowding distance* et seuls les meilleurs individus en fonction de ce score sont sélectionnés.

Définition 11 *La **crowding distance** a pour objectif de préserver la diversité dans la population. Pour cela, elle va affecter à chaque individu un rang en fonction de la proximité de son score par rapport aux individus ayant les scores les plus proches ; plus*

[1] Une implémentation de cet algorithme en `C` est disponible sur le site du *Kanpur Genetic Algorithms Laboratory* – `http://www.iitk.ac.in/kangal/codes.shtml`

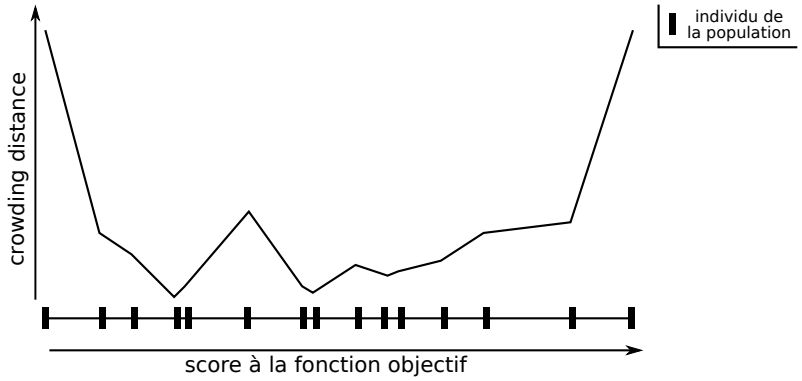


FIGURE 2.4 – Fonctionnement de la crowding distance : un score de distance de foule est attribué aux individus en fonction de leurs scores à la fonction objectif.

un individu est éloigné des autres, meilleur est son classement. Cette distance (cf. fig. 2.4) est calculée en fonction des scores des individus pour les fonctions objectifs. Pour chaque fonction objectif, les chromosomes sont triés et leurs scores de distances sont calculés à partir de la distance euclidienne qui les séparent de leurs voisins les plus proches. Les extremums sont tous conservés, et la distance finale d'un individu est la somme de ses distances pour chaque fonction objectif.

Il existe des versions plus récentes de cet algorithme (*NSGA-III* [12] et *U-NSGA-III* [13]) qui sont des optimisations pour les cas où le nombre de fonctions objectifs est situé entre 3 et 15 (elles possèdent une complexité $O(N^2 log^{M-2} N)$).

L'utilisation d'algorithmes évolutionnaires, quels qu'ils soient, implique la définition d'une (ou plusieurs) fonction(s) objectif. Dans notre cas d'étude, la diversité est l'objectif à atteindre ; nous avons ainsi choisi pour cela d'utiliser des fonctions objectif de distances afin de maximiser cette diversité.

2.3 Mesure de distance

La notion de diversité entre modèles est un problème riche. Elle est en effet étroitement liée à l'idée de distance, elle-même en rapport avec la notion de similarité/dissimilarité. Un certain nombre de travaux proposent de calculer la diversité entre modèles. Dans [14], Falleri et al. utilisent l'algorithme de *Similarity Flooding* ([15]) afin de mettre en relations des concepts proches dans deux méta-modèles pour la génération de transformations de modèles. Dans [16], Dolques et al. se servent quant à eux des égalités ainsi que de la présence de sous-chaîne entre les noms des différents concepts dans les modèles pour créer des paires qui seront utilisées dans un processus de génération de transformations de modèles par l'exemple ([17]). Cependant, ces différents travaux s'appuient sur la sémantique des noms des différents concepts des modèles, ce qui fonctionne dans le cas de modèles réels, mais qui ne peut s'appliquer dans notre cas. En effet, les modèles étant des modèles générés, la sémantique des noms n'a ici aucune valeur, et il convient alors de s'intéresser uniquement à la structure des modèles afin de déterminer la diversité. Nous

avons par conséquent choisi d'étudier dans cette section différentes mesures de distances entre graphes.

2.3.1 Distances entre modèles

Graph Edit Distance

L'une des distances les plus utilisées pour mesurer les dissimilarités entre graphes est celle proposée par Sanfeliu et Fu ([18]), où les auteurs adaptent la distance de Levenshtein ([19]) pour les graphes (*GED – Graph Edit Distance*). La distance d'édition entre un graphe G_1 et G_2 est définie comme le nombre minimum d'éditions, i.e. d'insertions, de suppressions et de substitutions de nœuds ou d'arêtes, nécessaires pour passer du graphe G_1 à un graphe isomorphe de G_2. Le calcul de cette distance est par conséquent un problème complexe. En effet, le problème d'isomorphisme de graphes est un problème dont on ne connaît pas d'algorithme de complexité polynomiale, et bien qu'il existe un certain nombre d'algorithmes différents pour le calcul de cette distance ([20]), la meilleure complexité de calcul connue pour un algorithme est exponentielle.

Il existe cependant un certain nombre d'heuristiques permettant d'approcher cette valeur. Dans [21], Zeng et al. donnent des algorithmes de complexité polynomiale afin de donner une borne inférieure et supérieure pour la distance d'édition de graphe. Ils obtiennent ainsi une approximation de la distance d'édition de graphes. Plus récemment, Fischer et al. ont également proposés dans [22] une approximation de cette distance en complexité quadratique.

Distance de centralité

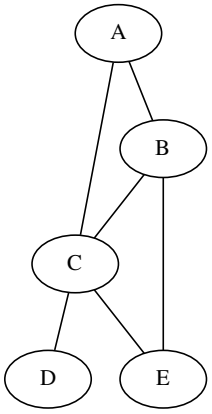
Dans [23], Roy et al. proposent une mesure de distance basée sur la centralité des nœuds d'un graphe ([24]). La centralité d'un nœud est une fonction qui, pour un nœud d'un graphe donné, associe une valeur réelle. Ainsi, pour un graphe $G = (V, E)$, la centralité est une fonction définie sur les sommets :

$$\begin{aligned} C\colon V &\longrightarrow \mathbb{R}^+ \\ v &\longmapsto C(G, v) \end{aligned} \tag{2.2}$$

Il existe ainsi plusieurs fonctions de centralité. Les principales mesures de centralité rappelées par Roy et al. sont :

- la centralité de degré, qui est égale au degré d'un sommet (on parle de centralités de degré entrant et de degré sortant pour le cas des graphes orientés) ;
- la centralité de proximité, définie par $CC(G, v) = \sum_{w \in V \setminus v} 2^{-d(v,w)}$, avec $d(v, w)$ la distance par le plus court chemin entre les sommets v et w ;
- la centralité d'intermédiarité, qui, pour un sommet v, est le nombre de plus court chemins passant par v divisé par le nombre total de plus courts chemins ($BC(G, v) = \sum_{x,w \in V} \frac{\sigma_v(x,w)}{\sigma(x,w)}$ avec $\sigma_v(x, w)$ le nombre de plus courts chemins entre x et w passant par v et $\sigma(x, w)$ le nombre de plus courts chemins entre x et w).

Un exemple de graphe et des différentes centralités associées à ses sommets est donné en fig. 2.5.



Sommet	Degré	Proximité	Intermédiarité
A	2	1.5	0
B	3	1.75	0.5
C	4	2.0	3.5
D	1	1.25	0
E	2	1.5	0

FIGURE 2.5 – Exemple de graphe et de centralités de ses sommets.

La mesure proposée par Roy et al. est définie comme la différence de centralité entre les nœuds de deux graphes (cf. équation 2.3).

$$d_C(G_1, G_2) = \sum_{v \in V} |C(G_1, v) - C(G_2, v)| \tag{2.3}$$

Ainsi, on obtient une information sur les différences structurelles d'un graphe (dans le cas de la distance de centralité de degré par exemple, cette distance nous donne une information structurelle sur les différences de degrés entre les graphes).

Le PageRank de Google [25] est une version modifiée de la centralité de degré qui donne plus d'importance à un arc rentrant s'il provient lui-même d'un nœud possédant une centralité élevée. Une version possible de la distance de centralité serait ainsi d'utiliser le PageRank comme mesure de centralité, l'édition de deux arcs n'ayant ainsi potentiellement plus la même influence ; en effet, dans le cadre de la centralité de degré, la suppression ou l'ajout d'un arc n'a un impact que de 1 sur la distance entre graphes, tandis que cette même modification dans le cadre du PageRank a un impact qui dépend des sommets de l'arc.

Cette distance possède cependant elle aussi un certain nombre de problèmes ; en effet, la distance de centralité est basée sur la différence de centralité pour un même sommet entre deux graphes. Il convient ainsi d'associer un sommet d'un graphe G_1 à un autre sommet d'un graphe G_2, ce qui nous ramène au problème de l'isomorphisme des graphes. On peut pour cela se ramener à une représentation vectorisée des graphes (par exemple, vecteur des centralités des nœuds triés par ordre décroissant).

2.3.2 Distances entre vecteurs

Les distances de Hamming ([26]) ou de Levenshtein ([19]) peuvent ainsi être appliquées pour deux modèles et donner des indications sur les différences entre les modèles. Des mesures plus récentes utilisées dans le domaine de l'exploration de texte peuvent également être appliquées. La distance cosinus ([27]) est un indicateur de la dissimilarité entre deux textes donnés. Habituellement, cette mesure est appliquée à deux vecteurs dont les éléments sont le nombre d'occurrences de chaque mot pour les deux textes donnés et est définie comme :

$$D_C(\overrightarrow{V_1}, \overrightarrow{V_2}) = 1 - S_C(\overrightarrow{V_1}, \overrightarrow{V_2}) \tag{2.4}$$

avec $S_C(\overrightarrow{V_1}, \overrightarrow{V_2})$ la similarité cosinus des vecteurs $\overrightarrow{V_1}$ et $\overrightarrow{V_2}$ définie comme :

$$\begin{aligned} S_C(\overrightarrow{V_1}, \overrightarrow{V_2}) &= \frac{\overrightarrow{V_1} \cdot \overrightarrow{V_2}}{\|\overrightarrow{V_1}\|\|\overrightarrow{V_2}\|} \\ &= \frac{\sum_{i=1}^{n} V_1[i]V_2[i]}{\sqrt{\sum_{i=1}^{n} V_1[i]^2}\sqrt{\sum_{i=1}^{n} V_2[i]^2}} \end{aligned} \tag{2.5}$$

Cette mesure est par ailleurs normalisée (le cosinus étant défini comme $cos : \mathbb{R} \longrightarrow [-1; 1]$), ce qui rend son interprétation plus facile que pour les autres mesures, mais nécessite cependant des vecteurs de longueurs égales.

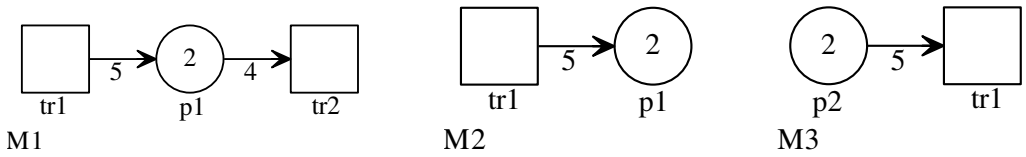
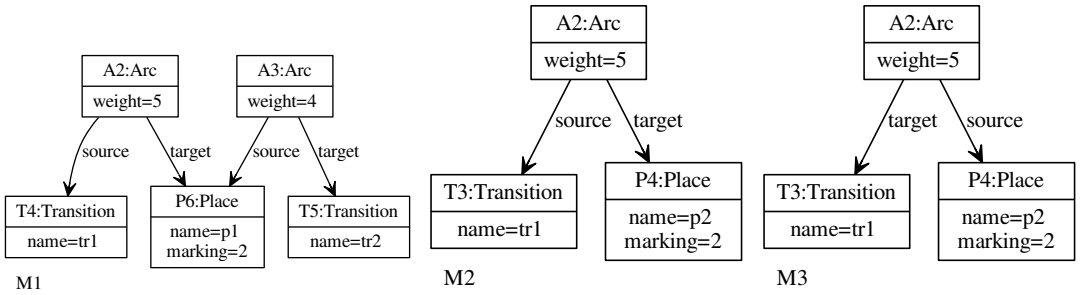


FIGURE 2.6 – Réseaux de Petri associés aux modèles de la fig. 2.7

Ces différentes distances peuvent cependant être complémentaires ; des distances entre des réseaux de Petri (dont le méta-modèle est donné fig. 1.1 et dont les représentations concrètes sont données fig. 2.6) sont données dans la fig. 2.7. On peut notamment observer que les modèles $M1$ et $M2$ sont relativement proches du point de vue de la centralité ; cependant, selon cette même distance et la distance de Hamming, les modèles $M2$ et $M3$ sont identiques. Or, la distance d'édition de graphe indique des différences structurelles, ce qui est effectivement le cas, le modèle $M2$ représentant un arc depuis une transition vers une place, tandis que le modèle $M3$ représente un arc depuis une place vers une transition. Cette dernière distance reste par conséquent intéressante bien que les algorithmes de calcul soient exponentiels.



Modèles considérés	**Distance d'édition de graphe**	**Distance de Hamming**	**Distance euclidienne des centralités**[1]
M1,M2	3	3	0.17
M1,M3	3	3	0.45
M2,M3	2	0	0

[1] Norme des vecteurs de centralités $d(A, B) = \sqrt{\sum_{i=0}^{|A|}(A_i - B_i)^2}$

FIGURE 2.7 – Mesures des différentes distances entre les modèles M1, M2 et M3.

3

Représentation des modèles

La représentation des modèles comme un chromosome composé de gènes est complexe, d'autant plus que l'on souhaite pouvoir effectuer sur eux des opérations de mutations et des croisements. Nous avons basé notre approche sur les travaux à base de CSP proposés par Ferdjoukh et al. dans [1].

3.1 Chromosome

Nous avons pris pour base un ensemble de modèles représentés sous la forme d'un vecteur de valeurs proposés par les approches par CSP (cf. section 2.1), celle-ci satisfaisant la représentation classique des chromosomes comme étant un vecteur de gènes (nos gènes étant ici des entiers – cf. fig. 3.1). En outre, le choix de cette représentation nous permet de vérifier la validité des nouveaux modèles générés facilement ; il nous suffit en effet de passer à un solveur CSP le vecteur ainsi que le réseau pour vérifier que le vecteur est bien une solution possible – et par conséquent, valide.

Nous avons également fait le choix de travailler sur des modèles ayant un nombre d'instances égal. En effet, plusieurs critères nous ont poussés vers cette décision :

- la distance cosinus est définie sur des vecteurs de taille égale : le calcul sur des vecteurs de tailles différentes aurait nécessité une transformation des vecteurs qui n'est pas aisée, chaque variable devant être associée à une variable équivalente pour obtenir un résultat cohérent ;

- les croisements dans les cas de vecteurs de longueurs variables ne pouvaient pas être effectué n'importe où : en effet, comme représenté fig. 3.1, des groupes de variables sont utilisés pour représenter des objets du modèle, les croisements n'auraient alors pas pu se faire n'importe où. L'utilisation de vecteurs de tailles égales permet de s'affranchir de cette contrainte.

Dans [28], Ferdjoukh et al. incorpore de la génération probabiliste afin de créer des modèles plus diversifiés. Nous avons fait le choix de nous baser sur une population de départ générée à l'aide de leur outil (nommé Grimm), composée de 100 modèles valides.

Nous avons choisi de considérer un chromosome en suivant deux approches :

- un individu est un ensemble de modèles (de taille m) ;
- un individu est un unique modèle.

Ces deux approches présentent chacune un certain nombre de problématiques.

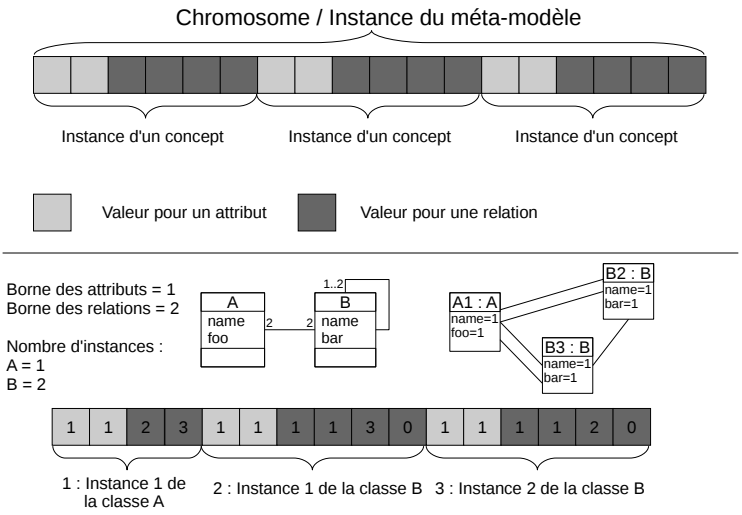


FIGURE 3.1 – Exemple de représentation d'une instance de méta-modèle comme vecteur.

En effet, si la première approche permet d'utiliser une fonction objectif s'appliquant sur un individu, de façon similaire à un algorithme évolutionnaire classique, la difficulté réside cependant dans ce calcul pour les cas où l'individu est composé de trois modèles ou plus. Par ailleurs, le nombre d'individus pour former une population devant être suffisant pour permettre des croisements, l'augmentation du nombre de modèles par individu nécessite la multiplication du nombre de modèles, augmentant le nombre requis d'individus en entrée.

La seconde approche quant à elle peut sembler plus intuitive dans sa représentation mais possède des difficultés sur le calcul de la fonction objectif. En effet, la vision de celle-ci est désormais au niveau de la population, et non plus d'un individu, et il convient alors de définir comment choisir les meilleurs individus. Les MA|PM (cf. étude bibliographique) ne peuvent pas être appliqué ici, ceux-ci travaillant certes d'un point de vue de la population, mais également d'un point de vue d'une métrique interne. Il faudrait alors déterminer un objectif à améliorer sur chaque modèle autre que la distance qui est multi-modèles.

3.2 Croisements, mutations, validité

À chaque génération, une sélection par tournoi binaire est effectuée afin de générer, à partir d'une population initiale de taille N, N nouveaux individus. Ces nouveaux individus sont obtenus grâce à un croisement en un point. Les opérations de mutations sont ensuite appliquées sur les individus (avec un pourcentage de chance déterminé).

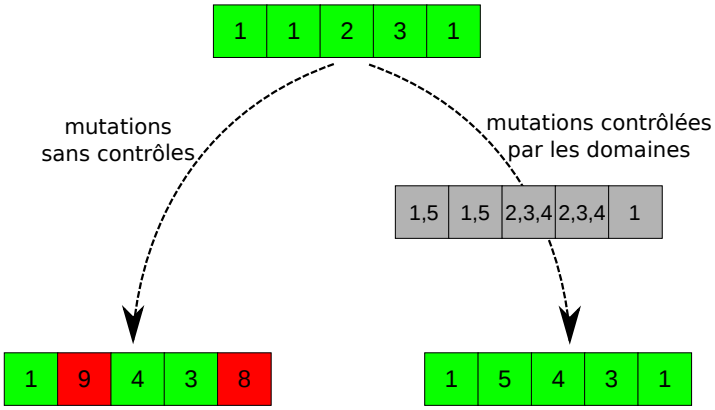


FIGURE 3.2 – Utilisation des domaines pour contrôler les mutations afin de réduire le nombre d'individus invalides.

Les premières expérimentations ont rapidement laissé apparaître que le nombre d'individus invalides générés avec les mutations était bien trop important, allongeant considérablement les temps d'exécution (si un individu est invalide, un nouvel individu doit être généré pour le remplacer afin de compléter la population). Nous avons fait le choix de stocker dans les chromosomes un certain nombre de données supplémentaires, comme les domaines des variables, afin d'effectuer des mutations plus contrôlées (cf. fig. 3.2).

Un modèle obtenu avec ces mutations contrôlées peut toutefois ne pas être valide, si le nouveau vecteur ne respecte pas les contraintes. Nous avons par conséquent choisi de conserver les contraintes du réseau CSP afin de pouvoir faire appel au solveur dans l'objectif de s'assurer que l'on possède toujours des modèles qui sont des instances valides du méta-modèle et qui respectent les contraintes OCL associées.

3.3 Sélection

3.3.1 Cas $m > 1$

Dans le cas $m = 2$, le calcul de la fonction objectif est trivial. En effet, celle-ci est égale, pour chaque individu, à la distance entre les deux modèles composants l'individu. Cependant, dans les cas où les individus sont composés de strictement plus de 2 modèles, on obtient pour chaque individu une matrice symétrique de distance, définie telle que :

$$\forall i \in pop, D_i = \begin{pmatrix} 0 & d_{1,2} & \cdots & d_{1,n} \\ d_{1,2} & 0 & \cdots & d_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ d_{1,n} & d_{2,n} & \cdots & 0 \end{pmatrix} \tag{3.1}$$

avec $d_{j,k}$ la distance entre le $j^{ième}$ et le $k^{ième}$ modèle de l'individu i.

La difficulté est alors d'effectuer une comparaison entre deux matrices. Nous avons choisi d'établir plusieurs stratégies de fonction objectif :

(1) une fonction objectif basée sur la valeur la plus *basse* des distances de la matrice ;

(2) une fonction objectif basée sur la valeur *moyenne* des distances de la matrice ;

(3) une fonction objectif qui est la *somme* des approches (1) et (2) ;

(4) deux fonctions objectifs : la (1) et la (2) ;

(5) considérer chacune des distances de la matrice comme une fonction objectif différente à améliorer.

Si les trois premières approches s'inscrivent naturellement dans le cadre d'un algorithme évolutionnaire, les approches (4) et (5) nécessitent cependant une approche multi-objective. Nous avons choisi d'utiliser l'algorithme *NSGA-II* ; celui-ci est en effet multi-objectifs, et est par ailleurs largement utilisé. Il existe par conséquent une large documentation accessible ainsi que de multiples implémentations.

Les résultats de ces différentes approches sont discutés section 4.1.

3.3.2 Cas $m = 1$

L'approche évolutionnaire classique fonctionne en calculant, à l'aide d'une fonction objectif, un score pour chaque individu. Les mesures de distance nécessitent cependant de travailler avec deux modèles ; par conséquent, dans le cas où chaque individu est composé d'un unique modèle, la métrique ne peut alors plus être interne mais nécessite d'être traitée au niveau de la population.

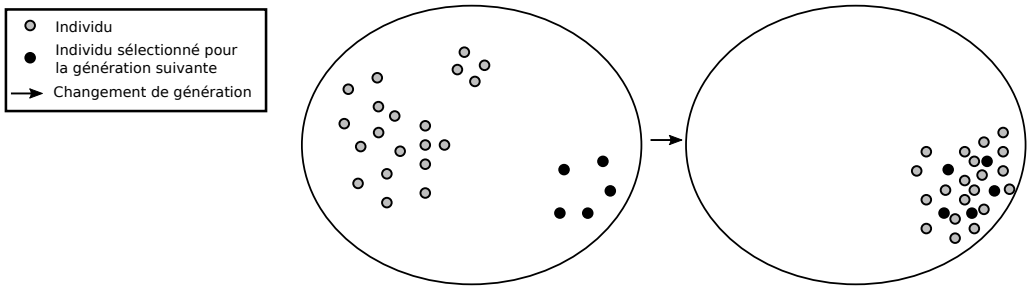


FIGURE 3.3 – Problématique de la sélection des meilleurs individus : la sélection des individus les plus éloignés ne permet pas d'augmenter la diversité.

Nous avons ainsi dû nous éloigner de la méta-heuristique classique, en ne traitant non plus un objectif interne mais un objectif au niveau de la population. Cet aspect gestion de la population est présent dans les MA|PM ; il y est cependant combiné à un objectif interne. Nous nous proposons une approche traitant uniquement cet aspect multi-individu.

Cette approche introduit cependant un nouveau problème : la détermination des meilleurs individus à conserver pour la génération suivante. Ceux-ci devront ainsi être les plus diversifiés entre eux-mêmes, car les résultats à l'itération suivante seront sinon moins bons. On peut par exemple observer fig. 3.3 qu'en ne sélectionnant que les individus étant en moyenne les plus éloignés (les points noirs sur la figure), il ne restera lors de la génération suivante que ceux-ci ainsi que de nouveaux individus générés à partir d'eux, relativement proche, réduisant par conséquent la diversité.

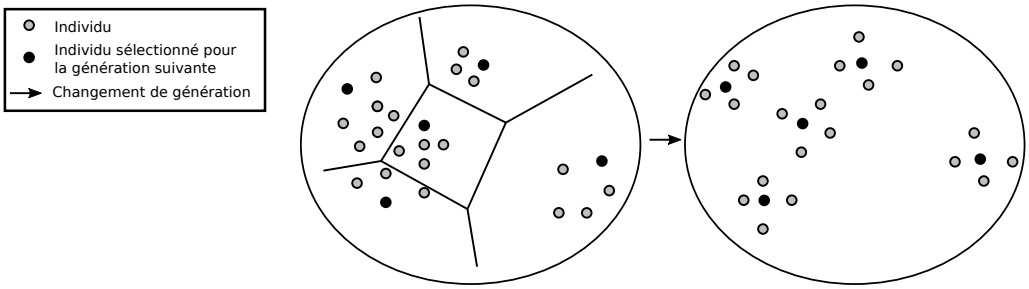


FIGURE 3.4 – Exemple de l'approche par partitionnement pour la sélection d'individus diversifiés.

L'idée retenue pour éviter cette régression de la diversité est d'utiliser une technique de partitionnement pour ne sélectionner que des individus étant distants les uns par rapport aux autres. En effet, une technique de partitionnement pourrait ainsi permettre d'éviter de choisir deux individus proches l'un de l'autre. Un partitionnement s'appuyant sur les diagrammes de Voronoï [29] est ainsi possible ; il consiste à regrouper les points en région de Voronoï, *i.e.* des régions composées d'un centre et de tous les points plus proches de ce centre que de n'importe quel autre centre. On voit par exemple fig. 3.4 que les individus sélectionnés grâce à cette technique sont beaucoup plus distants lors de la génération suivante que lors de la sélection simple des individus possédant la plus grande distance moyenne.

Si cette approche devait permettre d'éviter de sélectionner des individus membres d'une même partition et par conséquent, de garder une certaine diversité, elle a dans les faits entraîné une certaine stagnation au niveau des scores.

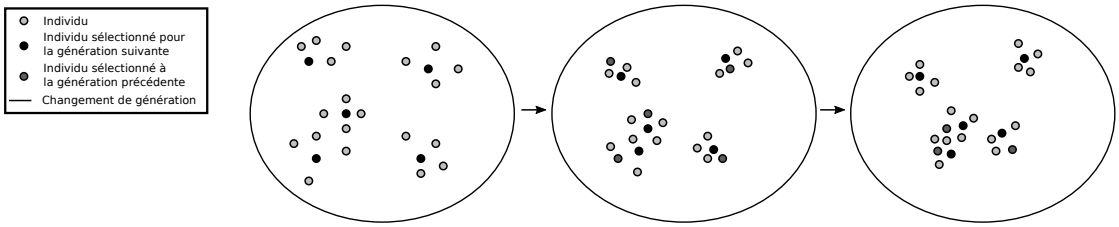


FIGURE 3.5 – Évolution possible de la population avec l'algorithme des k-médoïdes (les individus en noir sont ceux sélectionnés à chaque génération).

En effet, lors de la génération de nouveaux individus (par croisements et mutations), ceux-ci peuvent être moins distants des autres individus. On peut observer fig. 3.5 qu'une sélection d'un individu dans chaque partition peut entraîner non pas un éloignement des individus mais plutôt un regroupement. Dans le cas d'une évolution par k-médoïdes (l'individu sélectionné étant le médoïde), on peut ainsi avoir une évolution qui n'apporte pas d'amélioration notable de la diversité des individus.

Définition 12 *Dans une partition, le* ***médoïde*** *est l'élément d'un ensemble le plus central (de façon similaire à la médiane dans le cas d'une représentation en une dimension).*

L'approche que nous avons proposée pour surmonter ce problème est une modification de l'algorithme des k-médoïdes proposé par Park et Jun dans [30]. Celui-ci permet de se rapprocher itérativement du médoïde en sélectionnant à chaque itération le point de la partition dont la somme des distances aux autres éléments de la partition est minimale. Nous avons adapté cet algorithme pour sélectionner non pas l'élément le plus central mais l'élément le plus éloigné du reste de la population. On peut observer fig. 3.6 l'évolution que nous espérions ainsi obtenir avec cette approche.

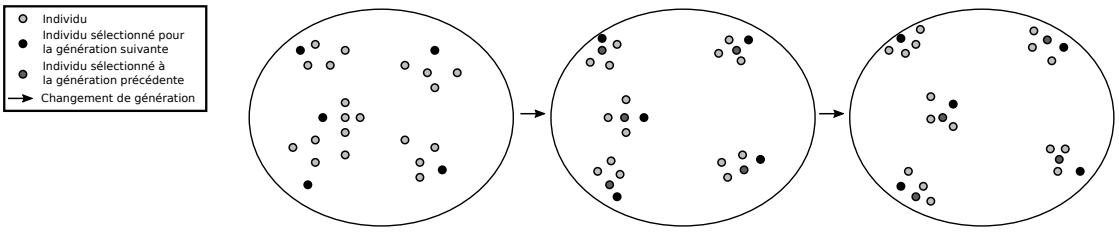


FIGURE 3.6 – Évolution théorique de la population avec l'algorithme des k-médoïdes modifié (les individus en noir sont ceux sélectionnés à chaque génération).

3.3.3 Distances

Dans le cas des expérimentations, nous nous sommes penchés sur deux mesures de distances :

- la distance cosinus, calculée directement entre les chromosomes ;
- une distance de Hamming, calculée sur des vecteurs représentatifs des modèles obtenus.

Pour la deuxième mesure, les modèles sont générés et sont représentés par des vecteurs ayant pour valeur les références et les attributs (de façon assez similaire au vecteur du CSP). Nous avons ainsi pu comparer les résultats des deux mesures.

La distance d'édition de graphe a également été testée, mais le temps de calcul exponentiel ne nous a permis de l'utiliser dans le cas d'un algorithme évolutionnaire. Une approche heuristique ainsi que l'implémentation d'autres mesures comme la distance de centralité sont également des mesures qu'il serait intéressant d'ajouter dans des travaux futurs.

4

Expérimentations

Dans cette partie, nous analyserons les différents résultats obtenus lors des phases d'expérimentations et les mettront en confrontation. Les résultats présentés ici sont basés sur des modèles Java instanciés à partir d'un méta-modèle Java (cf. fig. A.1) à l'aide de l'outil *Grimm*. Tous les résultats donnés dans ce chapitre se conforment à la légende présentée fig. 4.1.

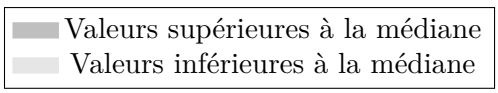


FIGURE 4.1 – Légende utilisée dans les courbes présentées chapitre 4.

4.1 Approche objectif intra-individu

L'objectif de cette approche était la génération d'un individu composé des modèles les plus diversifiés possibles. Cette approche est semblable à une approche évolutionnaire classique (fonction objectif sur un individu).

4.1.1 Cas $m = 2$ – Expérience témoin

Pour cette expérimentation, nous avons choisi de traiter trois cas pour voir comment évolue la population :

- le cas où nous ferions uniquement des croisements entre les modèles (croisements **inter-modèles**) : il s'agit dans ce cas là de trouver le meilleur appariement possible ;
- le cas où nous introduisons les **mutations** ;
- ainsi que le cas avec les **mutations** et les croisements au sein des modèles (croisements **intra-modèles**).

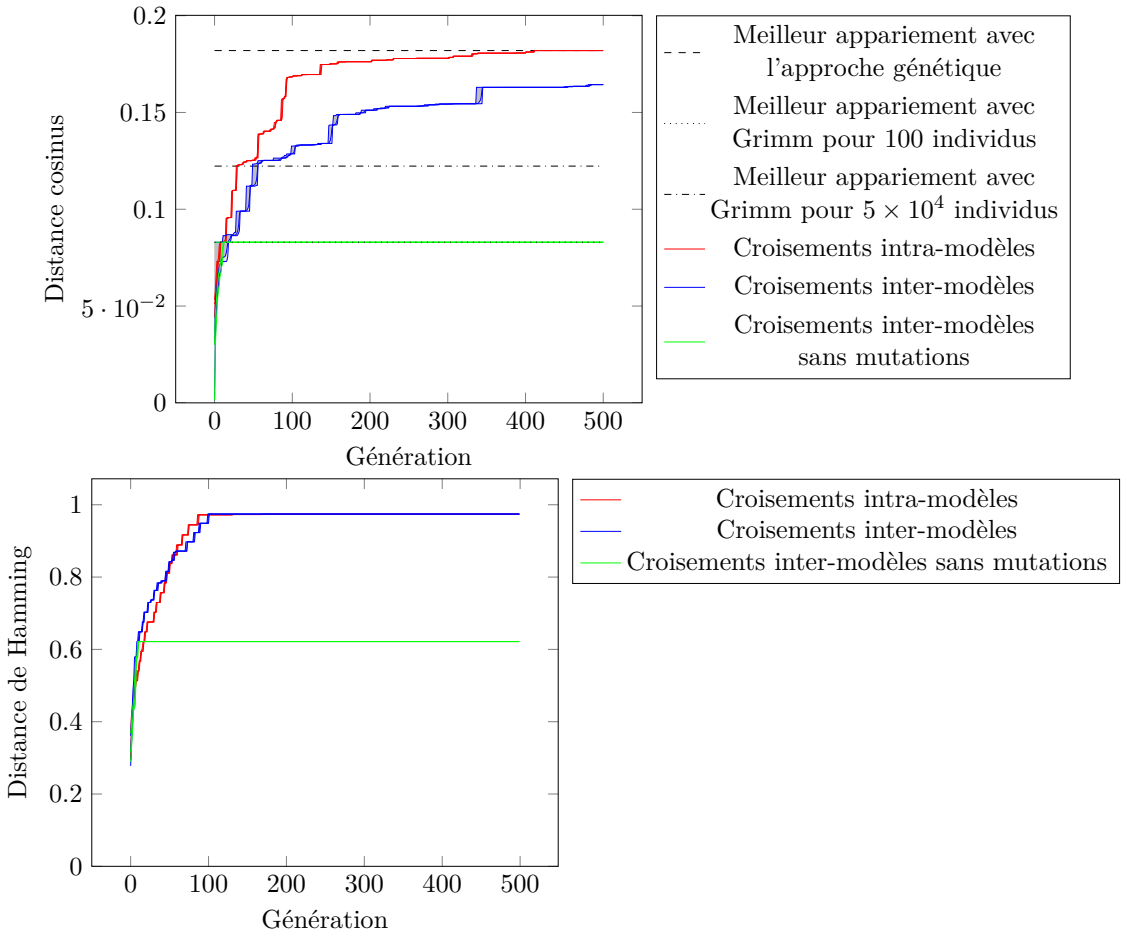


FIGURE 4.2 – Résultats des expérimentations sur une population de 50 individus avec $m = 2$

Dans la figure 4.2 sont données les courbes de répartitions des scores en fonction des générations. Les deux graphiques différents indiquent les évolutions suivant les deux objectifs différents ; sur chacun d'entre eux, les résultats de trois approches différentes sont données : un approche avec uniquement des croisements entre modèles, une approche avec mutations et croisements entre modèles et la dernière courbe indique les résultats obtenus grâce aux croisements au sein des modèles ainsi que des mutations. Les lignes pointillées indiquent les meilleurs scores pour la distance cosinus obtenus avec un algorithme polynomial de brute force testant toutes les combinaisons possibles. La distance de Hamming a été normalisée afin de faciliter l'interprétation.

Nous pouvons par conséquent observer l'apport de l'approche évolutionnaire. En effet, la première approche (celle sans mutations et avec des croisements entre modèles) permet d'obtenir rapidement (une vingtaine de générations) les scores que nous pourrions obtenir avec une approche algorithmique plus classique. Cependant, bien que l'on obtienne des scores meilleurs que ceux obtenus par l'approche par CSP (on peut en effet observer que le score minimum à la dernière génération est supérieur au meilleur score

de la première génération), l'approche méta-heuristique permet d'améliorer encore ces scores.

On observe ainsi que l'introduction de mutations permet d'éviter une convergence trop rapide et, dans le cas de la distance de Hamming, on observe une hausse du score de près de 57%. Les croisements intra-modèles permettent quant à eux d'améliorer encore un peu plus vite ce score, en sélectionnant les meilleures parties de chaque individu.

Les scores pour la distance cosinus reste quant à eux assez bas. On observe cependant sur la fig. 4.2 que notre approche permet de s'approcher nettement du meilleur score disponible à la dernière génération, indiqué par la ligne en tirets, obtenu avec l'algorithme de brute force.

Par ailleurs, nous avons utilisé cet algorithme sur 50000 modèles générés avec Grimm (en effet, avec notre approche nous générons 100 modèles par génération, sur 500 générations, soit un total de 50000 modèles). Le score ainsi obtenu est meilleur que celui obtenu avec ce même outil pour 100 modèles, mais il est encore inférieur à ceux que nous obtenons. La loi normale de distribution (cf. fig. A.4) permet de déduire une probabilité de $9 \times 10^{-13}‰$ chances d'obtenir le score que nous obtenons avec notre approche avec l'approche initiale.

On peut en déduire que l'approche méta-heuristique est plus intéressante que cet algorithme seul. En effet, le meilleur score initial possible est atteint avec une approche sans mutations ni croisements intra-modèles, mais est dépassé grâce à l'algorithme évolutionnaire.

Mutations

Lors des premières phases d'expérimentations, il est vite apparu que les mutations seraient responsables d'une augmentation du temps de calcul. En effet, l'appel au solveur externe afin de vérifier la validité des modèles mutés devant être répété pour chaque mutation, et chaque mutation rejetée entraînant la création d'un nouvel individu, il a semblé évident que ce taux de mutation devait rester assez faible. Par ailleurs, ce taux de mutation devait également rester relativement bas pour permettre d'explorer des nouvelles pistes sans altérer totalement la population existante.

La solution retenue est finalement celle présentée fig. 4.3. En effet, avec 0.5‰ chances de mutations par gènes, on obtient un pourcentage de la population mutée qui oscille entre 2% et 8% (environ 5% de la population mutée valide). Les tests précédents (1% – cf. fig. A.2 – et 1‰ – cf. fig. A.3) ont laissés apparaître une trop grande part de rejets, ce qui entrainait une trop grande perte de temps. Par ailleurs, ces taux de mutations faisaient muter respectivement environ 20% et 60% de la population, ce qui entraîne une trop grosse altération de la population.

Temps CPU

Comme dit précédement, l'appel à un solveur externe est un facteur de consommation de temps CPU. En effet, on peut observer fig. 4.4a que plus de 99% du temps est passé en appel externe. Par ailleurs, l'outil utilisé étant une application *Java*, les lancements de la machine virtuelle Java (*JVM*) sont responsables d'une grande partie de cette consommation (plus de 55%).

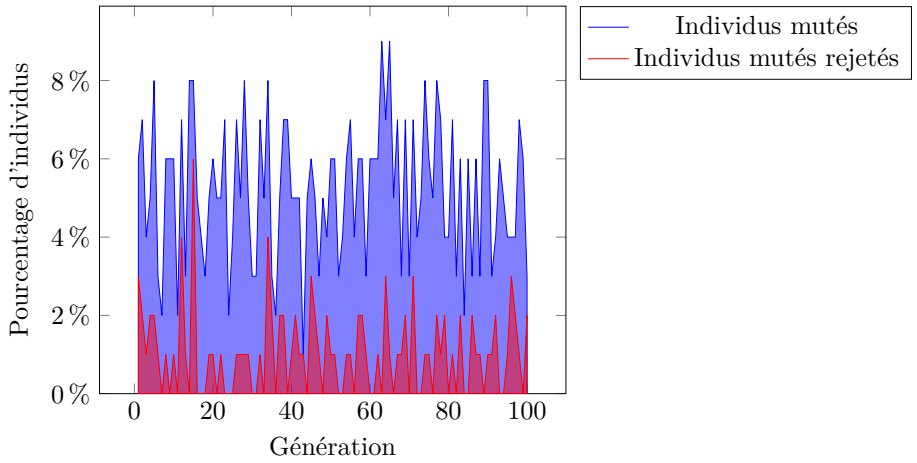
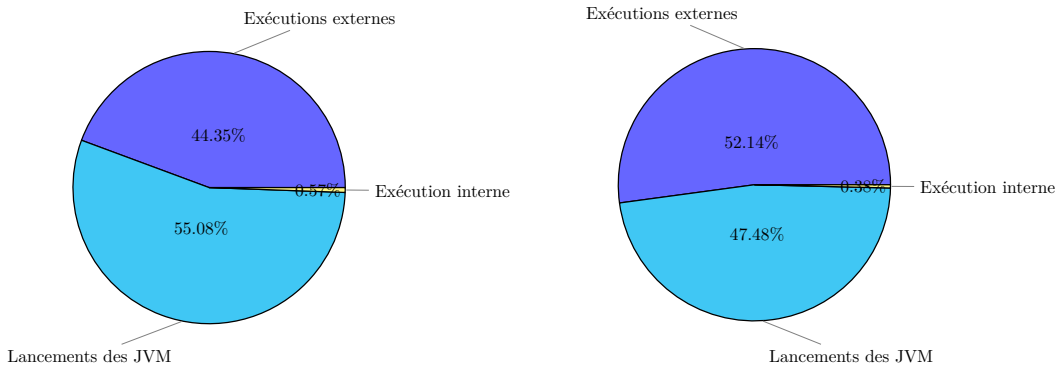


FIGURE 4.3 – Évolutions des mutations au cours des générations (0.5‰ chances de mutations).



(a) Distance cosinus (b) Distance de Hamming

FIGURE 4.4 – Répartition du temps moyen par génération (avec mutations et croisements intra-modèles).

De plus, le calcul de la distance de Hamming nécessite aussi le passage par des outils externes. En effet, le calcul sur la structure du graphe nécessite la génération des modèles à partir du vecteur de variables, et nous faisons également un appel à un outil externe pour le calcul de cette distance. On peut voir fig. 4.4b que ces appels supplémentaires ont un impact sur le temps de calcul pour une génération. La différence est encore plus grande sur les temps moyens d'exécution observés table 4.1. En effet, on observe une augmentation sur les temps moyens pour une génération de plus de 143%, ce qui s'explique par le plus grand nombre d'appels externes effectués (cf. table 4.2).

Ces appels externes sont les principaux impacts en terme de temps. Notre approche pourrait sans doute être grandement améliorée par l'implémentation de fonctions internes en remplacement de ces appels externes.

	Temps moyen par génération (ms)	**Temps d'appels moyen (ms)**	**Nombre d'appels moyens par génération**
Distance cosinus	63270.24	560.59	112.63
Distance de Hamming	90943.35	460.40	262.48

TABLE 4.1 – Temps d'exécution moyens (pour 100 modèles, 2 modèles par individu, 50 individus générés à chaque génération).

	Appels au solveur	**Appels à l'outil de génération**	**Calcul de distance (Hamming)**
Distance cosinus	112.63	0	0
Distance de Hamming	112.48	100	50[a]

[a]Seul le score des nouveaux individus est calculé, les scores étant conservés dans le but de limiter les appels externes.

TABLE 4.2 – Nombre moyen par génération d'appels aux outils externes (pour 100 modèles, 2 modèles par individu, 50 individus générés à chaque génération).

4.1.2 Cas $m = 3$ et $m = 5$

Cas $m = 3$

On observe sur la fig. 4.5 les résultats des différentes approches d'objectifs. Ceux-ci nous permettent de déduire que les approches multi-objectives (l'agrégat (3), le bi-objectifs (4) et l'approche définissant chaque distance comme objectifs (5)) produisent de meilleurs résultats que les approches mono-objective ((1) et (2)). L'approche bi-objective (4) est par ailleurs celle fournissant les meilleurs scores ; nous n'afficherons que celle-ci sur les résultats des expérimentations suivantes pour ne pas surcharger les figures[1].

On peut par ailleurs remarquer que la convergence survient bien moins vite sur une population composée d'individus possédants trois modèles que dans le cas des chromosomes composés de deux individus (cf. fig. 4.2). Les résultats obtenus sont cependant à terme assez similaires, le score maximum atteint à la dernière génération étant semblable.

Cas $m = 5$

Les résultats présentés fig. 4.6 permettent quant à eux de confirmer les résultats précédents. On observe en effet que l'on obtient toujours à terme des scores moyens équivalents aux scores obtenus avec des nombres de modèles par chromosome inférieurs, mais le nombre de générations nécessaire pour parvenir à une convergence est plus important que dans les cas précédents.

Discussions

On observe aussi bien dans le cas $m = 3$ que $m = 5$ que les distances des générations finales sont toujours meilleures que les distances générées par l'approche par CSP utilisée pour la population initiale. On observe cependant que l'augmentation du nombre de

[1]Les résultats obtenus dans les autres expérimentations pour les autres objectifs donnent un comportement similaire à ceux de $m = 3$ sur plusieurs exécutions (cf. annexe A.4.3).

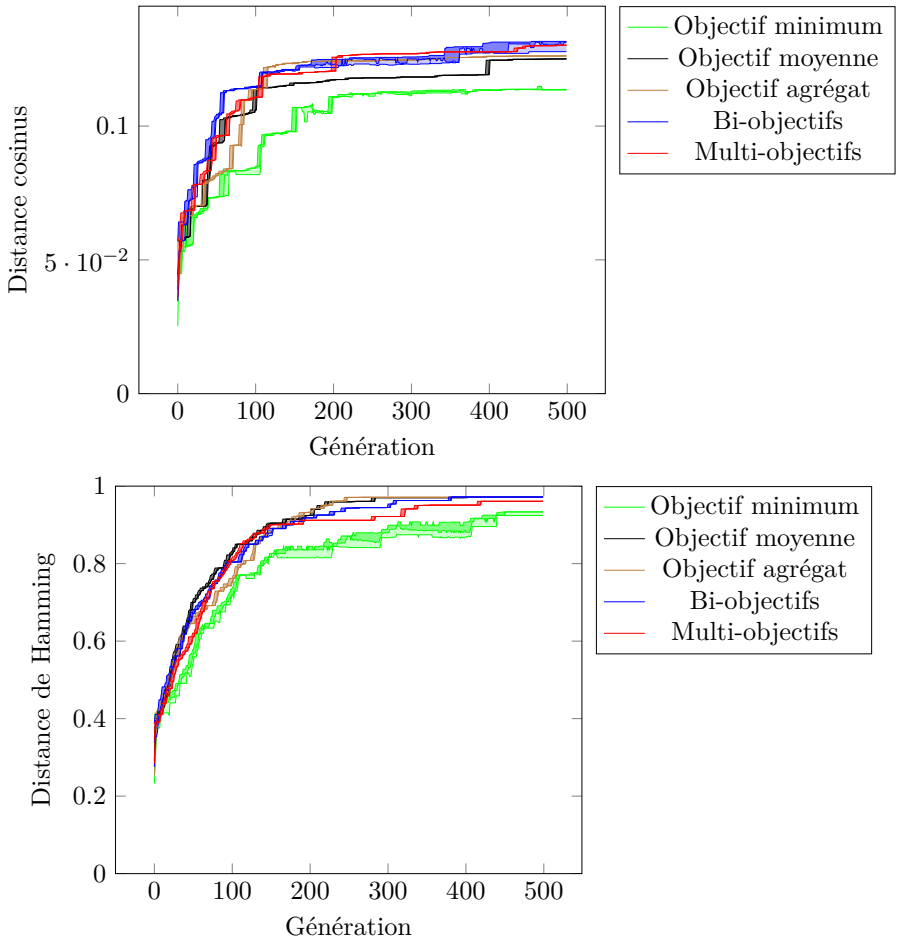


FIGURE 4.5 – Résultats des expérimentations sur une population de 33 individus avec $m = 3$.

modèles par chromosomes entraînent également un allongement du temps nécessaire pour obtenir une convergence (dans la plupart des cas, nous avons dû interrompre les calculs avant convergence dû à la durée excessive du temps de calcul).

Par ailleurs, on observe clairement une différence au niveau des mesures de distances choisies. L'approche par cosinus, effectuée directement sur le chromosome, produit ainsi peu d'augmentations, le chromosome possédant un certain nombre de gènes ne pouvant que peu varier dû aux contraintes, ce qui engendre une distance cosinus faible entre deux individus.

La distance de Hamming possède elle aussi un certain nombre d'inconvénients. En effet, la génération de modèles étant nécessaire pour la calculer, elle allonge grandement le temps d'exécution, chaque nouvel individu créé évalué entraînant la génération de modèles (sur une population de 20 individus avec $m = 5$, 20 nouveaux individus sont créés, engendrant la génération de 100 nouveaux modèles à chaque itération, soit 10000 générations sur 100 itérations).

Ces deux distances ne donnent pas les mêmes individus finaux. En effet, on peut

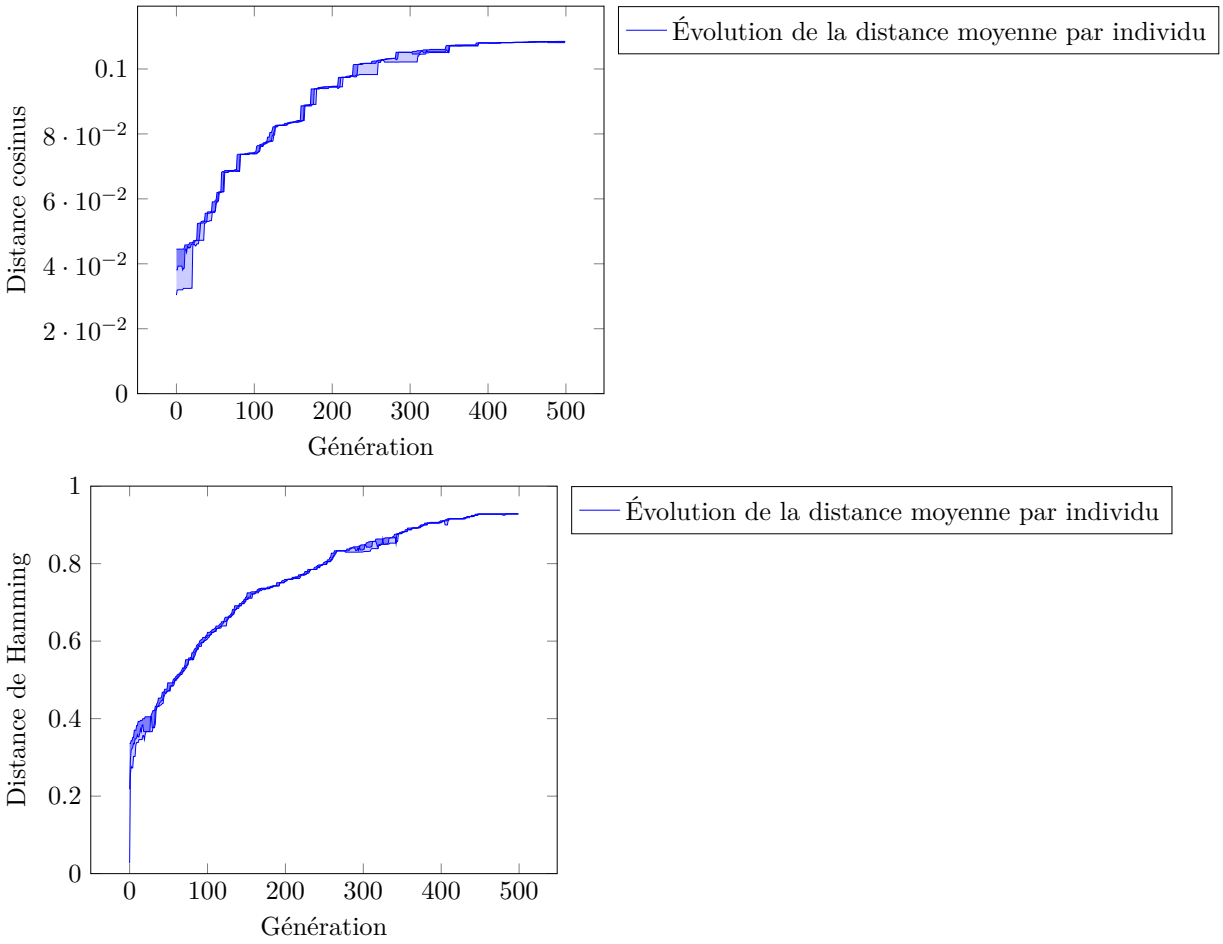


FIGURE 4.6 – Résultats des expérimentations sur une population de 20 individus avec $m = 5$.

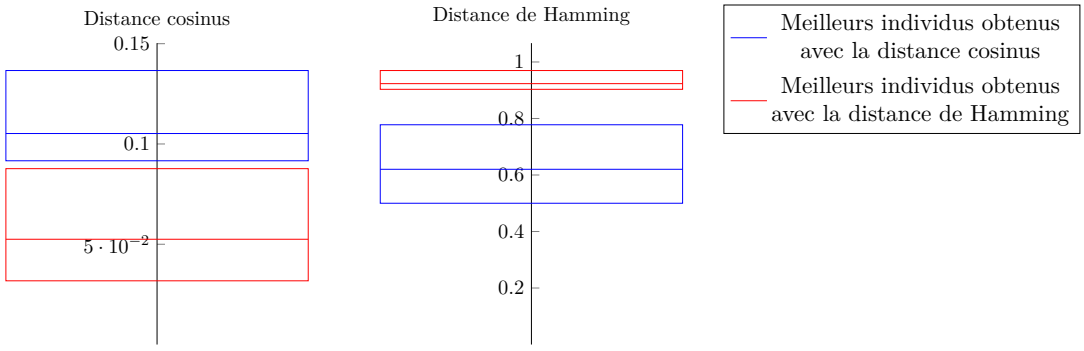


FIGURE 4.7 – Comparaison des scores des meilleurs individus.

observer fig. 4.7 que les meilleurs individus pour une distance obtiennent des scores inférieurs au maximum pour la seconde distance. Une amélioration possible serait ainsi de définir les deux distances comme des objectifs différents afin d'essayer d'augmenter

le score final.

4.2 Approche objectif inter-individu

Bien que les résultats obtenus avec la première approche permettent déjà une augmentation de la diversité, nous avons choisi d'expérimenter les cas où $m = 1$. En effet, cette approche devrait pouvoir nous permettre d'améliorer de façon plus efficace la diversité dans une population plus importante ; l'accès à plus d'individus permet ainsi de ne sélectionner que les plus diversifiés, tandis que l'approche précédente nous contraignait à choisir un groupe d'individus, y compris ses éléments les plus faibles.

4.2.1 Résultats

Les résultats obtenus grâce à la nouvelle approche permettent de mettre en évidence une amélioration des résultats par rapport à ceux de la première approche. On peut en effet observer fig. 4.8 que les résultats obtenus avec cette technique sont en moyenne meilleurs que ceux obtenus avec l'approche précédente (à titre de comparaison, les scores obtenus avec l'approche $m = 5$ sont également donnés sur la figure).

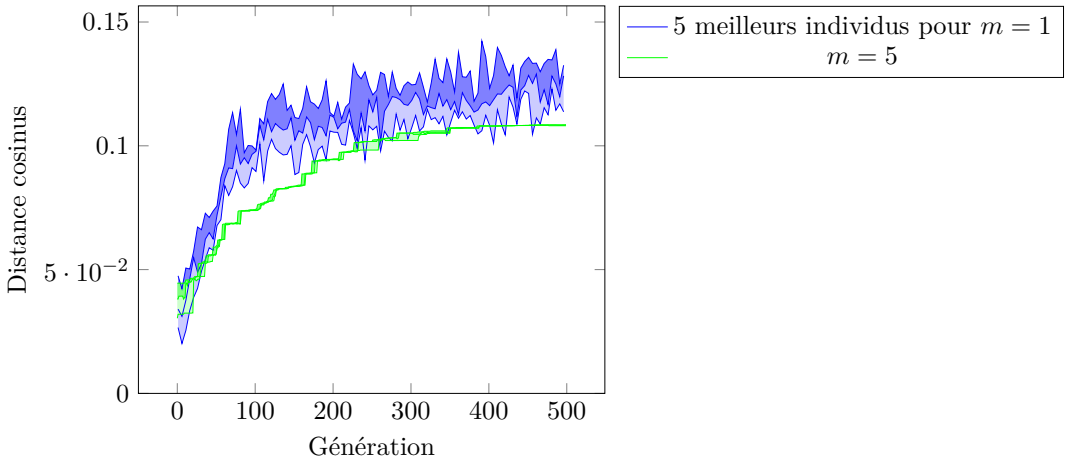


FIGURE 4.8 – Évolution des scores moyens de la population pour $m = 1$ et $m = 5$.

On observe cependant également que la distance moyenne entre les 5 meilleurs individus varie : en effet, si la distance moyenne des 100 individus augmentent en moyenne (cf. fig. A.16), le découpage en partition peut sélectionner des individus ayant des distances moyennes plus faibles. Cela n'impacte au final cependant pas les résultats finaux ; on observe en effet que les scores obtenus sont généralement supérieurs à ceux obtenus avec l'approche $m = 5$ après convergence (en raison du temps d'exécution excessif dû aux appels externes, l'expérimentation avec la distance de Hamming a été interrompue avant convergence).

Nous constatons également sur la fig. 4.9 que cette approche, en plus d'augmenter la distance moyenne des modèles, permet également d'uniformiser ces scores. En effet, lors des premières générations, on peut observer qu'il n'y a pas de répartition homogène des

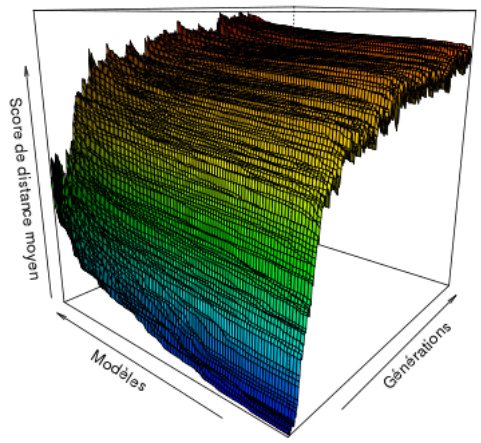


FIGURE 4.9 – Évolution des distances moyennes pour une population de taille 100 avec $m = 1$.

scores de distances. Cependant, la progression dans les générations permet d'obtenir des scores plus réguliers, avec non seulement de meilleures distances, mais également une meilleure répartition sur l'ensemble de la population. Ceci se traduit par une meilleure distribution dans l'espace.

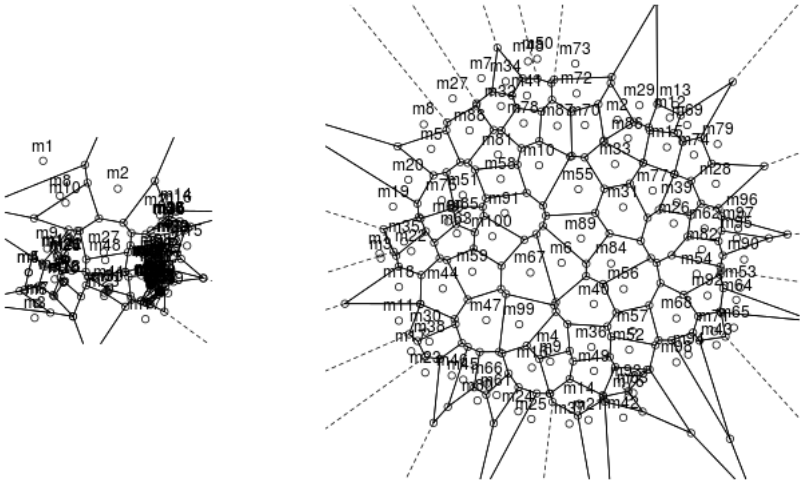


FIGURE 4.10 – Répartition des modèles avec l'approche $m = 1$ représentés à l'aide de diagrammes de Voronoï (génération 1 à gauche, génération 490 à droite).

À titre d'exemple, la fig. 4.10 permet d'observer cette nette amélioration de la distribution entre la population initiale et la population obtenue après convergence. En effet, la distance moyenne entre modèles est augmentée ; on peut également voir que la répartition des modèles est beaucoup plus uniforme. Par ailleurs, il est important de noter que les scores pour $m = 5$ sont les scores moyens de distances entre 5 modèles, tandis que cette approche permet d'obtenir des scores moyens plus élevés entre 100 modèles. Ces

moyennes sont en outre encore améliorées sur les générations suivantes (cf. fig. A.16). Il est donc aussi rapide de sélectionner les 5 meilleurs individus avec cette technique que les 20 meilleurs.

Cette approche permet par conséquent d'augmenter la distance entre un grand nombre d'individus simplement. Par rapport à l'approche $m = 5$, elle nécessite un temps de calcul plus élevé ; il faut cependant tenir compte de l'écart d'échelle avec les résultats précédent.

En effet, pour calculer les distances dans les cas avec $m > 1$, il faut effectuer $\binom{m}{2}$ calculs de distances par individu. On obtient alors :

$$nb_{calculs}^{m>1} = \binom{m}{2} * p \tag{4.1}$$

pour une population de taille p. Dans le cas où $m = 1$, les calculs s'effectuent sur l'ensemble de la population, et par conséquent, on a :

$$nb_{calculs}^{m=1} = \binom{p}{2} \tag{4.2}$$

par génération. Pour sélectionner un grand nombre de modèles, il faut cependant donc moins de calculs de distances avec cette seconde approche, les algorithmes génétiques nécessitant une valeur pour p suffisamment grande afin d'effectuer les opérations de croisements (pour un nombre de modèles total égal, la première approche nécessiterait ainsi plus de calculs).

On peut ainsi observer fig. 4.11 que dans les cas où l'on souhaite obtenir un grand nombre de modèles (plus de 10 modèles), il est plus intéressant d'utiliser une approche avec 1 modèle par chromosome. En effet, dans le cas où l'on souhaite obtenir 10 modèles diversifiés, avec la première approche ($m = 10$), pour $p = 50$, le nombre de calculs nécessaires est supérieur à la seconde approche pour une taille de population égale avec $m = 1$.

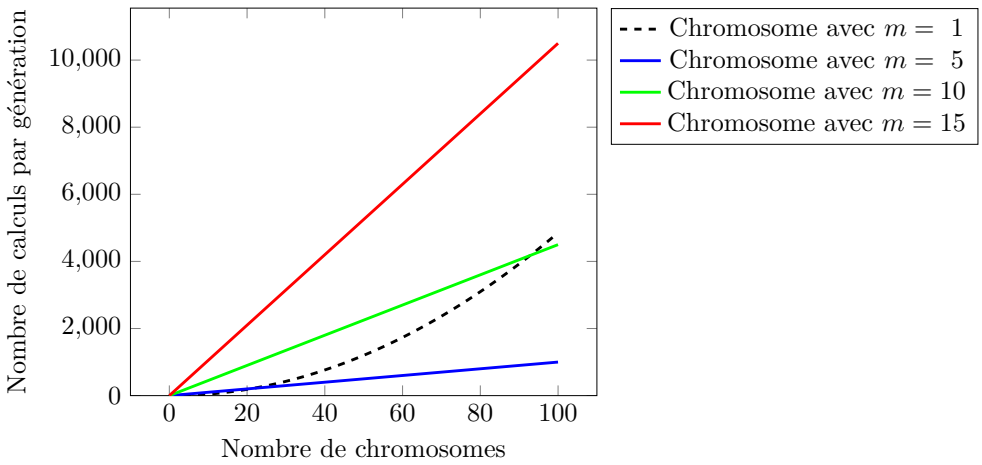


FIGURE 4.11 – Évolution du nombre de calculs nécessaires par génération en fonction de la taille de la population et du nombre de modèles par individu.

5

Conclusion et perspectives

La diversité des modèles est un problème majeur à résoudre afin d'obtenir une génération de modèles complète. En effet, la génération de modèles diversifiés est un moyen d'obtenir un banc d'essai ayant la plus grande couverture de cas possibles. Les données différentes obtenues rendent ainsi possible la validation la plus exhaustive possible.

L'aspect méta-modélisation de notre approche permet en outre de représenter des problèmes très différents, nous permettant de nous abstraire des cas concrets ; les expérimentations préliminaires sur un autre méta-modèle (réseaux de Petri) laissent apparaître des résultats similaires. L'approche générative peut ainsi être appliquée à d'autres domaines qu'au seul génie logiciel, en générant par exemple des bancs d'essai pour valider de nouvelles approches algorithmiques.

Nous pouvons prendre l'exemple des graphes de scaffold ; ces graphes sont utilisés en biologie pour modéliser des séquences d'ADN. L'obtention de graphes réels séquençant des brins réels est une opération longue et coûteuse, et la génération de graphes peut ainsi permettre de créer des données de tests pour les différents algorithmes qui permettent de produire un génome complet.

L'approche évolutionnaire proposée dans cette étude permet d'obtenir des résultats satisfaisant notre objectif initial de diversité. En effet, on observe que notre approche permet d'atteindre des scores de diversités non atteint avec la génération à l'aide de CSP initiale (même dans le cas d'une génération probabiliste). En effet, la méta-heuristique permet de favoriser les modèles les plus singuliers, augmentant au fur à mesure des générations la diversité globale de la population.

De nos deux approches, la seconde est celle qui permet le passage à l'échelle le plus aisé ; en effet, les sélections des 5 individus les plus diversifiés ou des 20 individus sur la population finale de 100 individus sont similaires en terme de temps (une simple approche par partitionnement permet de rapidement obtenir les individus les plus diversifiés sur cette population). En effet, la sélection de 20 individus avec une approche $m = 20$, sur une population de 20 individus par exemple, nécessiterait 7600 opérations de calcul, contre toujours 4950 calculs pour une approche $m = 1$.

Les travaux entrepris peuvent cependant encore être améliorés. Il subsiste en effet des pistes de recherche non exploitées.

Diversification des mesures de distance

Nous avons dans le cadre du stage traité deux mesures de distances ; un certain nombre d'autres mesures intéressantes, présentées dans l'étude bibliographique, pourrait également être traitée afin d'obtenir des données diversifiées d'un autre point de vue (la distance de centralité pourrait par exemple produire des données plus diversifiées d'un point de vue structurel).

Diversification des données d'entrée

Nous nous sommes concentré dans le cadre de cette étude sur les cas de modèles de tailles égales. Une généralisation à des modèles de taille variable, en définissant des points de coupes (afin de conserver une consistance sur les données) et en échelonnant les vecteurs afin que ceux-ci soient de tailles égales pour les mesures le requérant. Ceci permettrait ainsi d'obtenir des modèles divers dans leurs structures, mais également dans leurs tailles.

Améliorations techniques

Une des grosses contraintes de l'outil actuel est une contrainte technique. En effet, la dépendance à des outils externes auquel nous devons faire appel augmente grandement les temps d'exécution. L'incorporation des outils dans notre processus permettrait ainsi d'obtenir des résultats plus rapides.

Généralisation du processsus

Le processus de génération nécessite actuellement qu'on lui fournisse en entrée des modèles. Notre approche pourrait par conséquent être généralisée, en s'affranchissant de l'outil initial et en générant les premiers modèles directement depuis le méta-modèle. Un processus d'apprentissage pourrait également être introduit, afin qu'un expert du domaine puisse définir quels sont les individus jugés comme intéressants, dans l'objectif d'accentuer les recherches sur des individus diversifiés tout en possédant des caractéristiques désirés par l'utilisateur.

Bibliographie

[1] Adel Ferdjoukh, Anne-Elisabeth Baert, Éric Bourreau, Annie Chateau, Rémi Coletta, and Clémentine Nebut. Instantiation of Meta-models Constrained with OCL : a CSP Approach. In *International Conference on Model-Driven Engineering and Software Development (MODELSWARD)*, pages 213–222, 2015.

[2] Vincent Aranega, Jean-Marie Mottu, Anne Etien, Thomas Degueule, Benoit Baudry, and Jean-Luc Dekeyser. Towards an automation of the mutation analysis dedicated to model transformation. *Software Testing, Verification and Reliability*, 25(5-7) :653–683, 2015.

[3] Jordi Cabot, Robert Clarisó, and Daniel Riera. Verification of UML/OCL class diagrams using constraint programming. In *IEEE International Conference on Software Testing Verification and Validation Workshop (ICSTW)*, pages 73–80, 2008.

[4] Carlos Alberto González Pérez, Fabian Buettner, Robert Clarisó, and Jordi Cabot. EMFtoCSP : A tool for the lightweight verification of EMF models. In *Formal Methods in Software Engineering : Rigorous and Agile Approaches (FormSERA)*, 2012.

[5] Juan José Cadavid Gomez, Benoit Baudry, and Houari Sahraoui. Searching the Boundaries of a Modeling Space to Test Metamodels. In *IEEE International Conference on Software Testing, Verification and Validation (ICST)*, pages 131–140, April 2012.

[6] Erwan Brottier, Franck Fleurey, Jim Steel, Benoit Baudry, and Yves Le Traon. Metamodel-based test generation for model transformations : an algorithm and a tool. In *ISSRE, International Symposium on Software Reliability Engineering*, pages 85–94, 2006.

[7] Sagar Sen, Benoit Baudry, and Jean-Marie Mottu. Automatic model generation strategies for model transformation testing. In *Second International Conference, ICMT 2009, Zurich, Switzerland, June 29-30, 2009.*, pages 148–164, 2009.

[8] Daniel Jackson. Alloy : a lightweight object modelling notation. *ACM Transactions on Software Engineering and Methodology*, 11(2) :256–290, 2002.

[9] John H. Holland. *Adaptation in natural and artificial systems : an introductory analysis with applications to biology, control, and artificial intelligence.* The University of Michigan Press, 1975.

[10] David E. Goldberg and John H. Holland. Genetic algorithms and machine learning. *Machine learning*, 3(2) :95–99, 1988.

[11] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm : NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2) :182–197, Apr 2002.

[12] Kalyanmoy Deb and Himanshu Jain. An Evolutionary Many-Objective Optimization Algorithm Using Reference-Point-Based Nondominated Sorting Approach, Part I : Solving Problems With Box Constraints. *IEEE Transactions on Evolutionary Computation*, 18(4) :577–601, Aug 2014.

[13] Haitham Seada and Kalyanmoy Deb. U-NSGA-III : A unified evolutionary algorithm for single, multiple, and many-objective optimization. *COIN Report*, (2014022).

[14] Jean-Rémy Falleri, Marianne Huchard, Mathieu Lafourcade, and Clémentine Nebut. Metamodel matching for automatic model transformation generation. In *Model Driven Engineering Languages and Systems*, pages 326–340. Springer, 2008.

[15] Sergey Melnik, Hector Garcia-Molina, and Erhard Rahm. Similarity flooding : A versatile graph matching algorithm and its application to schema matching. In *Data Engineering, 2002. Proceedings. 18th International Conference on*, pages 117–128. IEEE, 2002.

[16] Xavier Dolques, Aymen Dogui, Jean-Rémy Falleri, Marianne Huchard, Clémentine Nebut, and François Pfister. Easing model transformation learning with automatically aligned examples. In *Modelling Foundations and Applications*, pages 189–204. Springer, 2011.

[17] Manuel Wimmer, Michael Strommer, Horst Kargl, and Gerhard Kramler. Towards model transformation generation by-example. In *System Sciences, 2007. HICSS 2007. 40th Annual Hawaii International Conference on*, pages 285b–285b. IEEE, 2007.

[18] Alberto Sanfeliu and King-Sun Fu. A distance measure between attributed relational graphs for pattern recognition. *IEEE Transactions on Systems, Man and Cybernetics*, (3) :353–362, 1983.

[19] Vladimir Iosifovich Levenshtein. Binary Codes Capable of Correcting Deletions, Insertions and Reversals. In *Soviet Physics Doklady*, volume 10, page 707, 1966.

[20] Xinbo Gao, Bing Xiao, Dacheng Tao, and Xuelong Li. A survey of graph edit distance. *Pattern Analysis and applications*, 13(1) :113–129, 2010.

[21] Zhiping Zeng, Anthony KH Tung, Jianyong Wang, Jianhua Feng, and Lizhu Zhou. Comparing stars : on approximating graph edit distance. *Proceedings of the VLDB Endowment*, 2(1) :25–36, 2009.

[22] Andreas Fischer, Ching Y Suen, Volkmar Frinken, Kaspar Riesen, and Horst Bunke. Approximation of graph edit distance based on Hausdorff matching. *Pattern Recognition*, 48(2) :331–343, 2015.

[23] Matthieu Roy, Stefan Schmid, and Gilles Tredan. Modeling and measuring graph similarity : The case for centrality distance. In *Proceedings of the ACM international workshop on Foundations of mobile computing*, pages 47–52, 2014.

[24] Linton C. Freeman. Centrality in social networks conceptual clarification. *Social networks*, 1(3) :215–239, 1978.

[25] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The PageRank citation ranking : bringing order to the web. *Technical Report 1999-66, Stanford InfoLab*, 1999.

[26] Richard W Hamming. Error detecting and error correcting codes. *Bell System technical journal*, 29(2) :147–160, 1950.

[27] Amit Singhal. Modern information retrieval : A brief overview. *IEEE Data Engineering Bulletin*, 24(4) :35–43, 2001.

[28] Adel Ferdjoukh, Éric Bourreau, Annie Chateau, and Clémentine Nebut. A Model-Driven Approach to Generate Relevant and Realistic Datasets. In *Software Engineering and Knowledge Engineering (SEKE), 2016 28th International Conference on*, Jul 2016.

[29] Franz Aurenhammer. Voronoi diagrams — a survey of a fundamental geometric data structure. *ACM Comput. Surv.*, 23(3) :345–405, September 1991.

[30] Hae-Sang Park and Chi-Hyuck Jun. A simple and fast algorithm for k-medoids clustering. *Expert Systems with Applications*, 36(2) :3336–3341, 2009.

A

Annexes

A.1 Méta-modèle Java

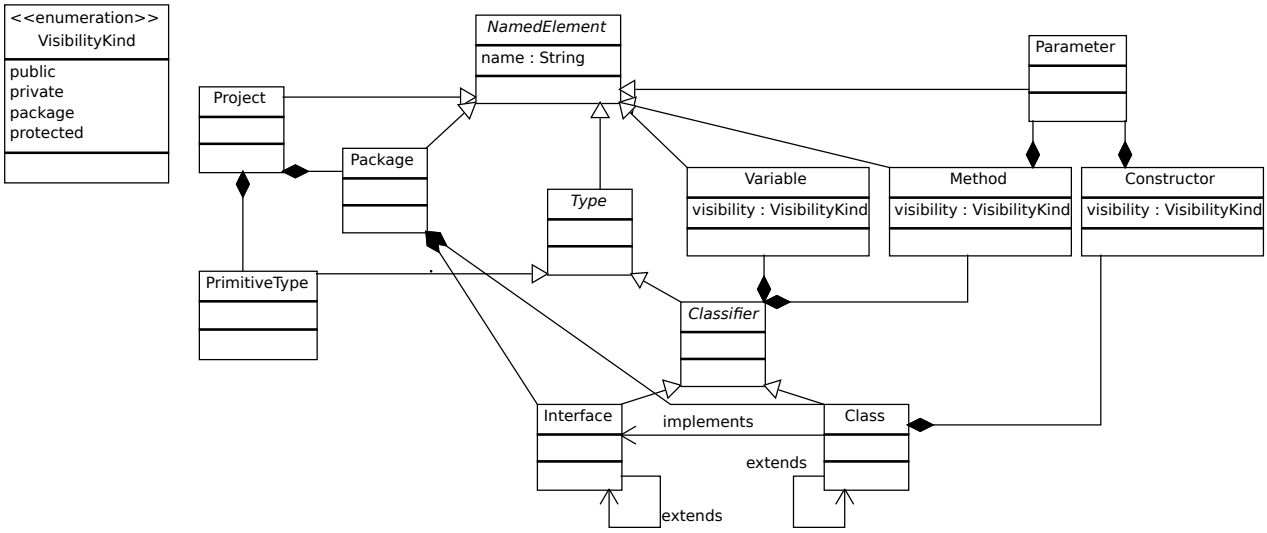


FIGURE A.1 – Méta-modèle Java utilisé pour la génération de modèles.

A.2 Mutations

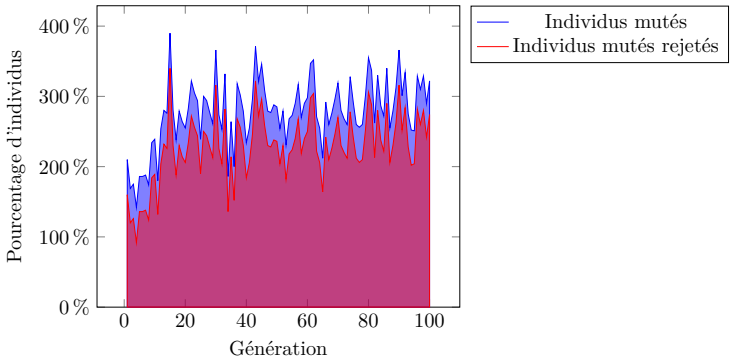


FIGURE A.2 – Évolution des mutations au cours des générations (1% chances de mutations).

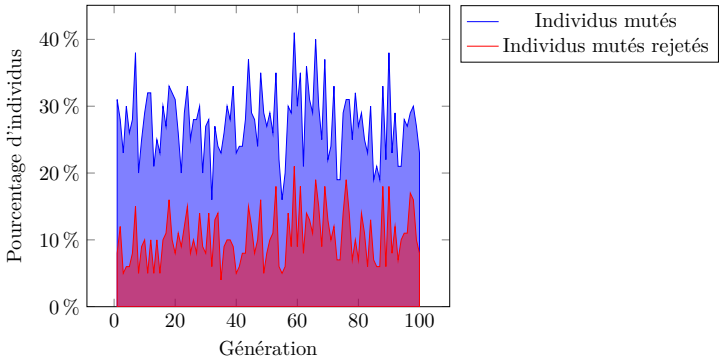


FIGURE A.3 – Évolution des mutations au cours des générations (1‰ chances de mutations).

A.3 Distribution des paires avec l'approche initiale

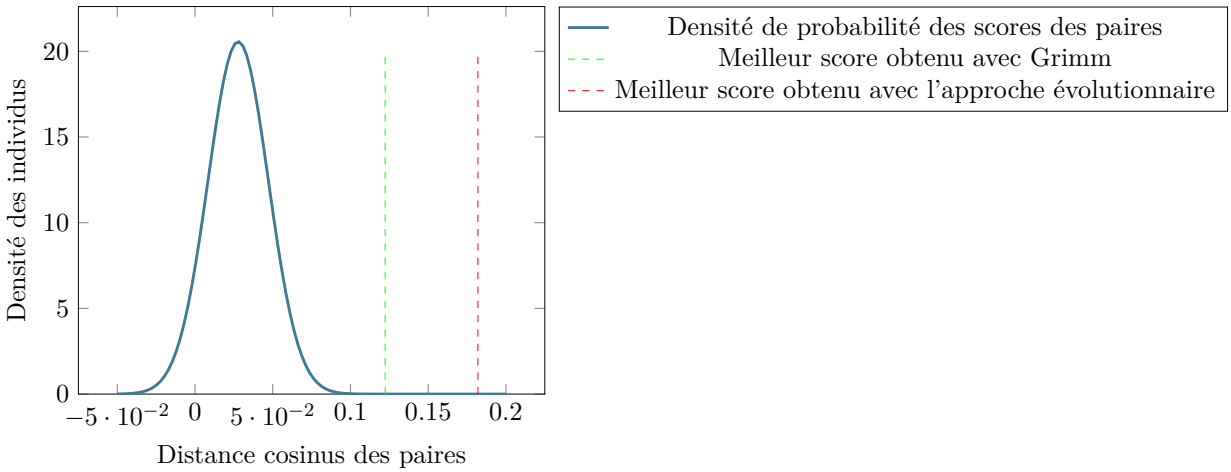


FIGURE A.4 – Distribution des scores pour chaque paire obtenue à partir de 10000 modèles générés avec Grimm.

A.4 Résultats des expérimentations

A.4.1 Cas $m = 2$

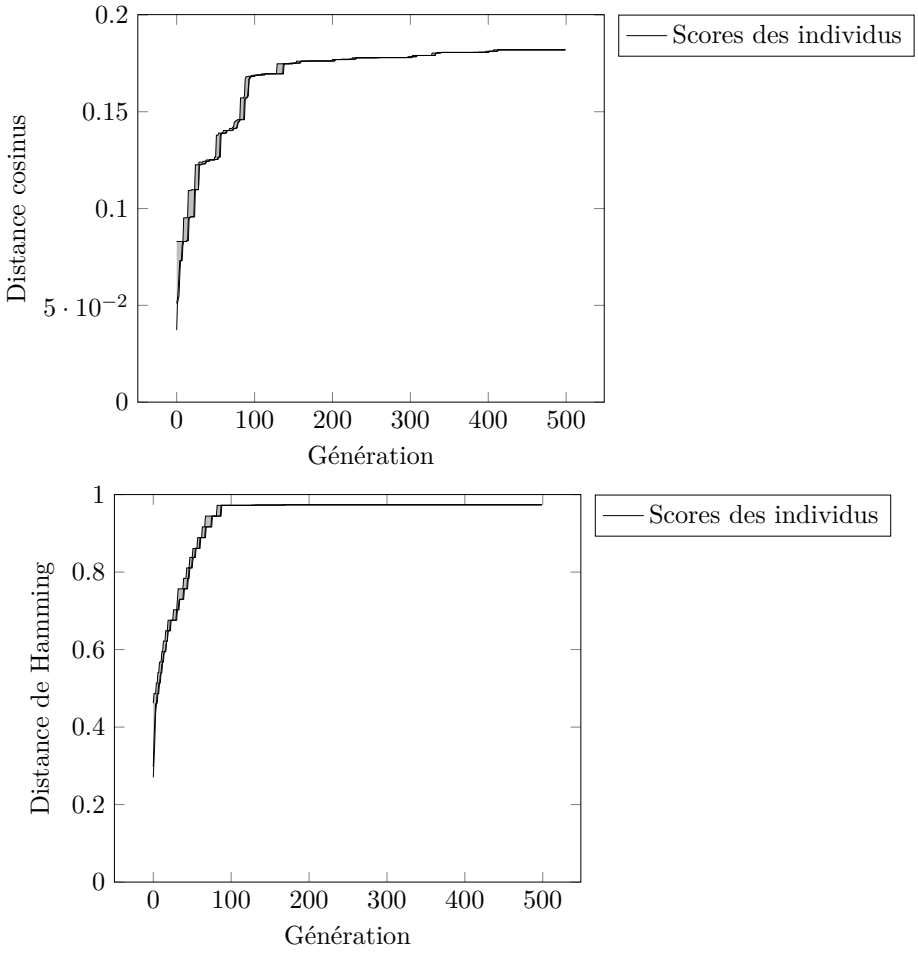


FIGURE A.5 – Évolution des distances des modèles par individu sur une population de 50 individus avec $m = 2$.

A.4.2 Cas $m = 3$

Objectif minimum

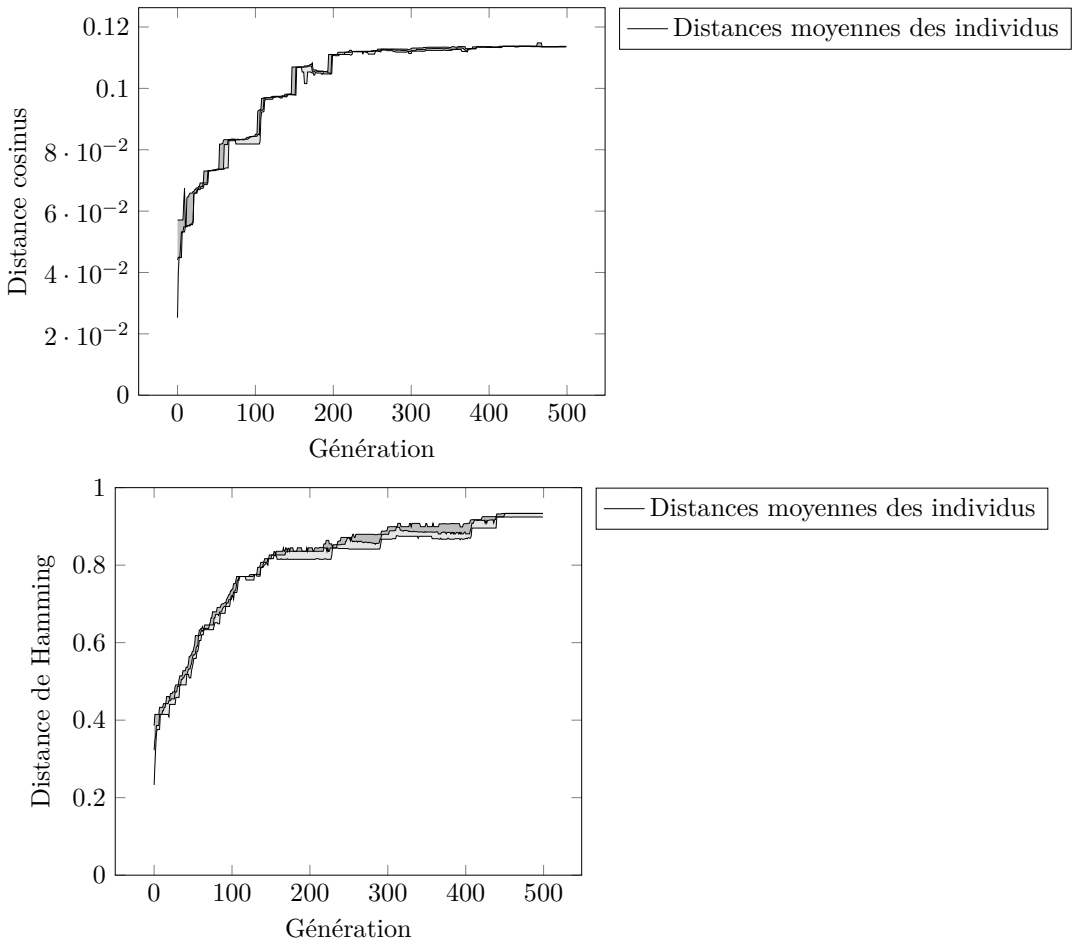


FIGURE A.6 – Évolution des distances moyennes des modèles par individu sur une population de 33 individus avec $m = 3$ avec comme objectif le minimum des distances.

Objectif moyenne

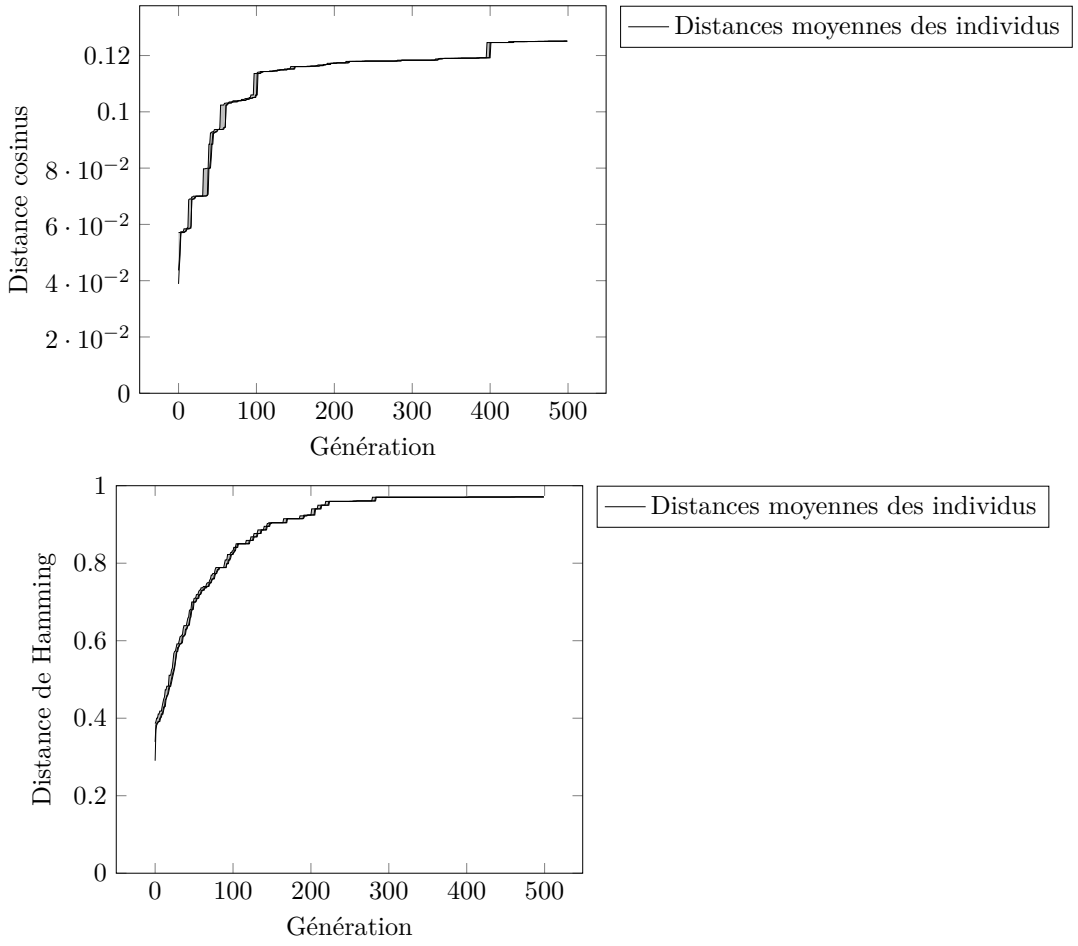


FIGURE A.7 – Évolution des distances moyennes des modèles par individu sur une population de 33 individus avec $m = 3$ avec comme objectif la moyenne des distances.

Objectif agrégat minimum et moyenne

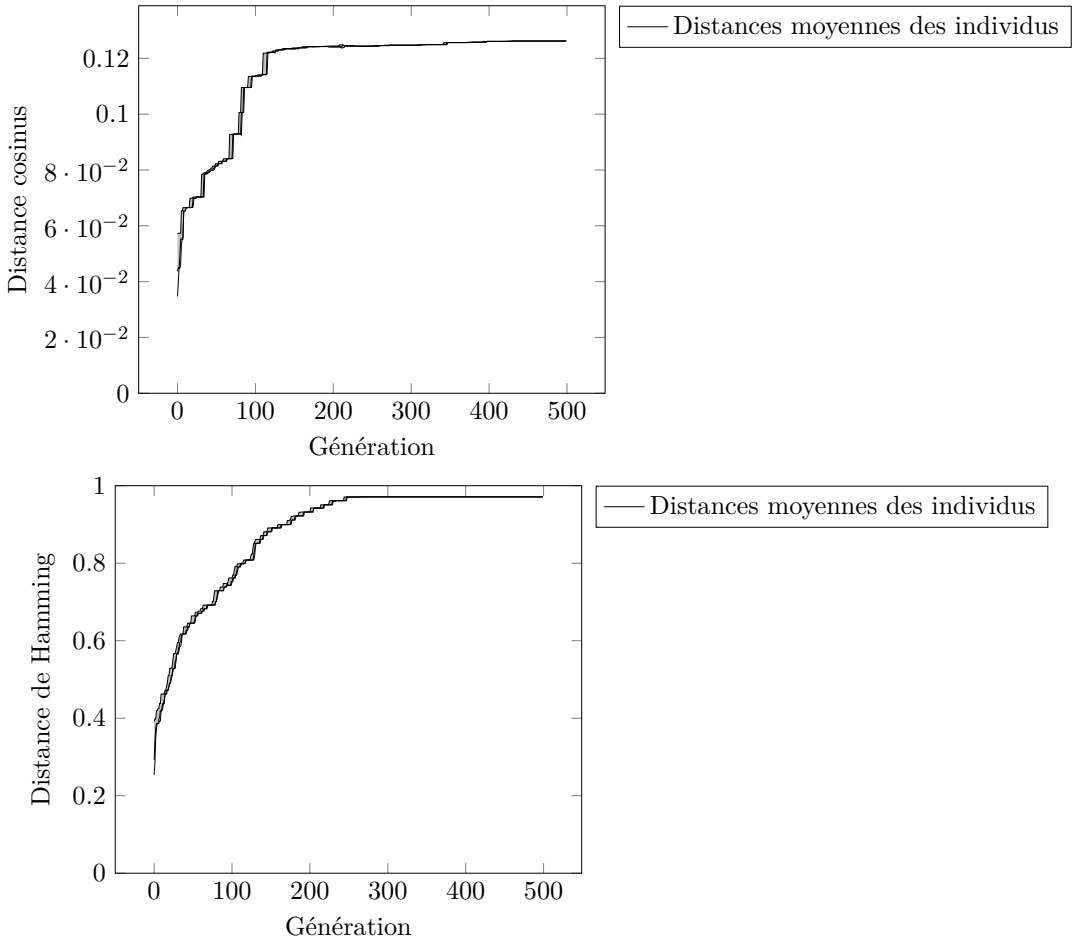


FIGURE A.8 – Évolution des distances moyennes des modèles par individu sur une population de 33 individus avec $m = 3$ avec comme objectif la somme du minimum et du la moyenne des distances.

Bi-objectifs minimum et moyenne

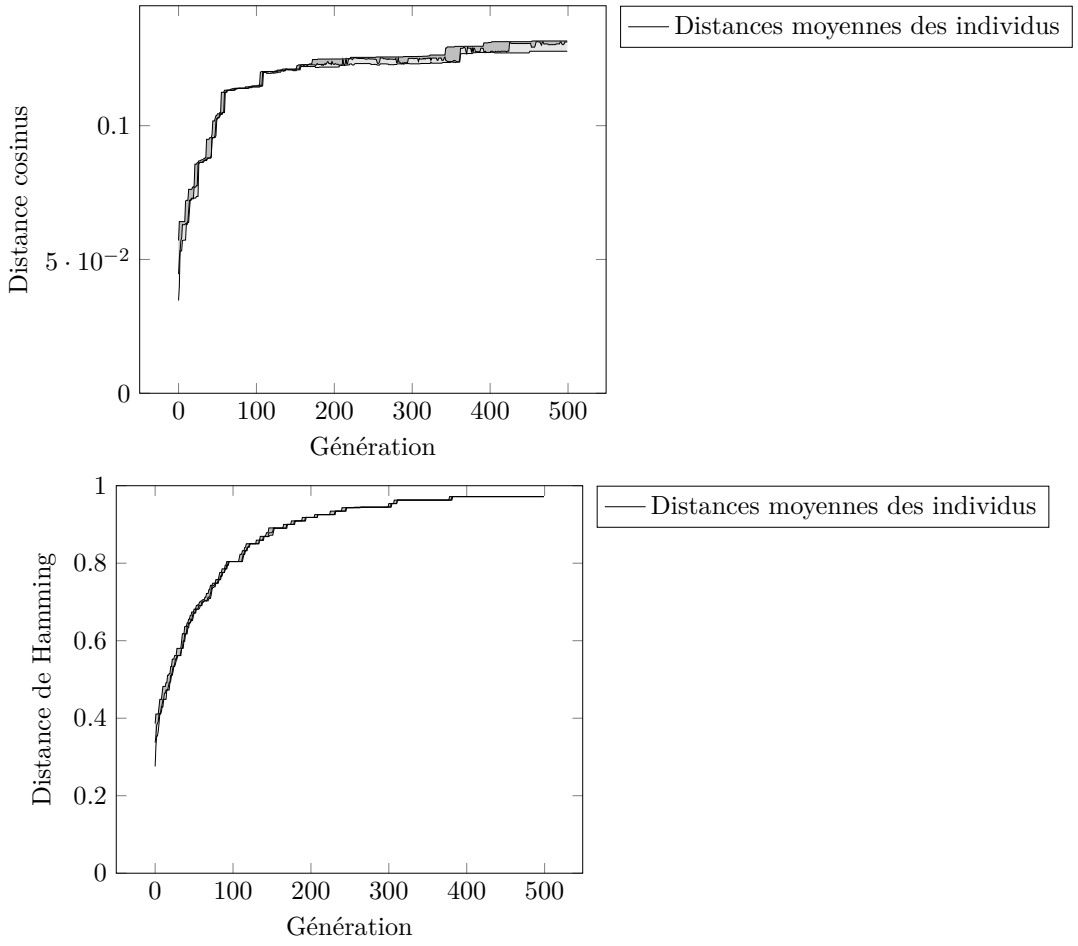


FIGURE A.9 – Évolution des distances moyennes des modèles par individu sur une population de 33 individus avec $m = 3$ avec comme objectifs le minimum et la moyenne des distances.

Multi-objectifs

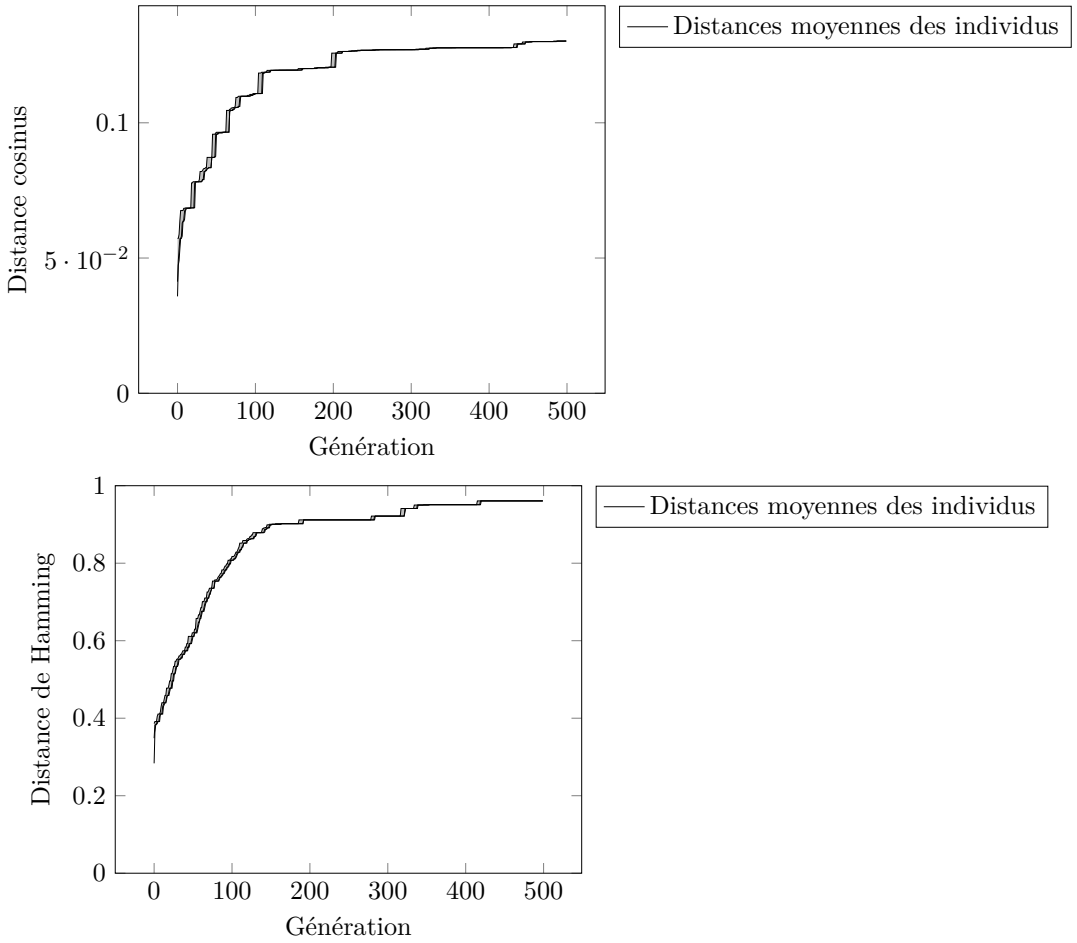


FIGURE A.10 – Évolution des distances moyennes des modèles par individu sur une population de 33 individus avec $m = 3$ avec chaque distance traitée comme un objectif.

A.4.3 Cas $m = 5$

Objectif minimum

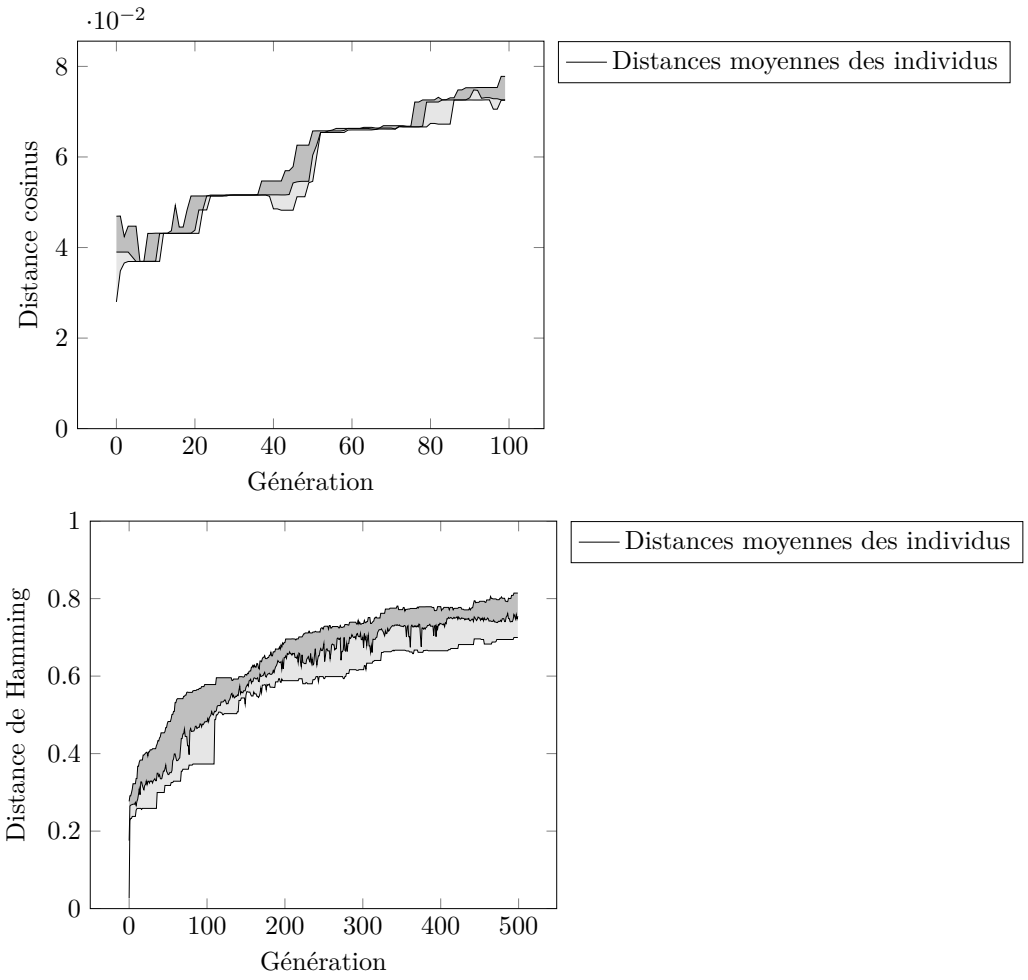


FIGURE A.11 – Évolution des distances moyens des modèles par individu sur une population de 20 individus avec $m = 5$ avec comme objectif le minimum des distances.

Objectif moyenne

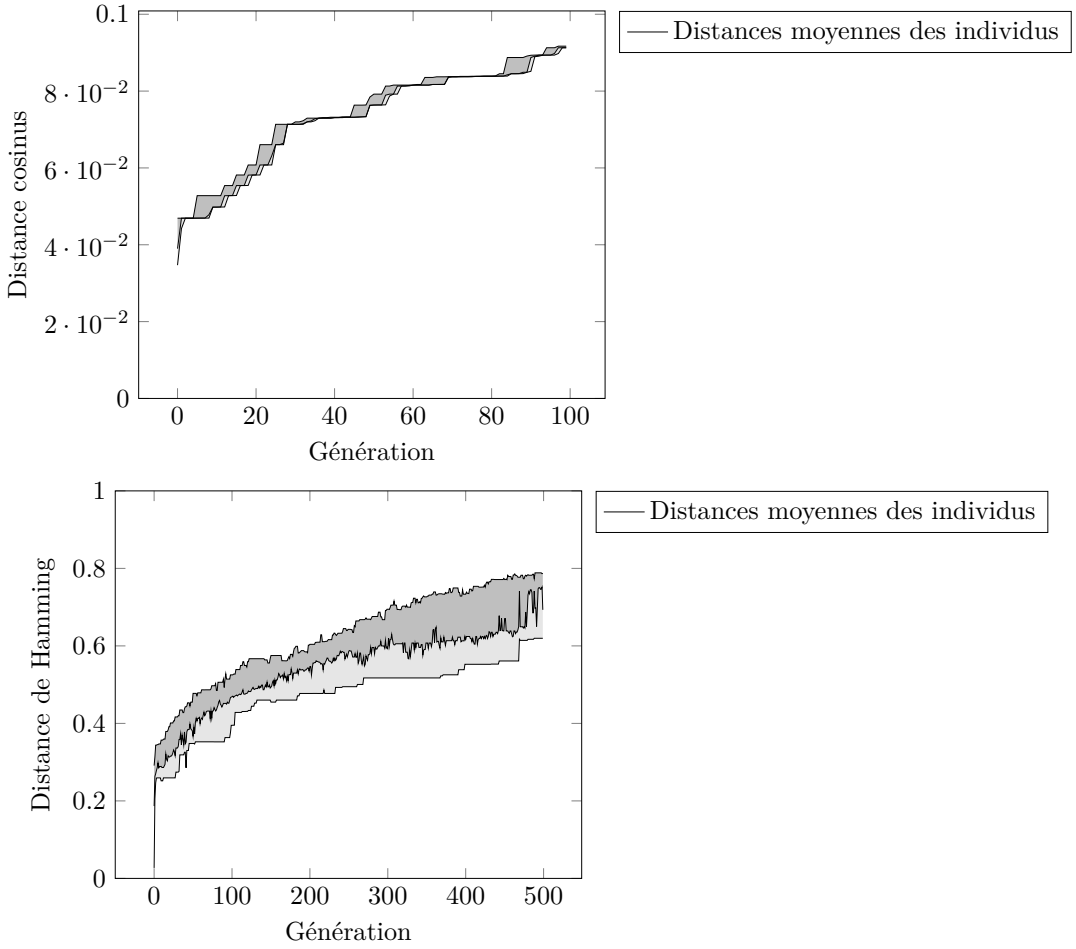


FIGURE A.12 – Évolution des distances moyens des modèles par individu sur une population de 20 individus avec $m = 5$ avec comme objectif la moyenne des distances.

Objectif agrégat minimum et moyenne

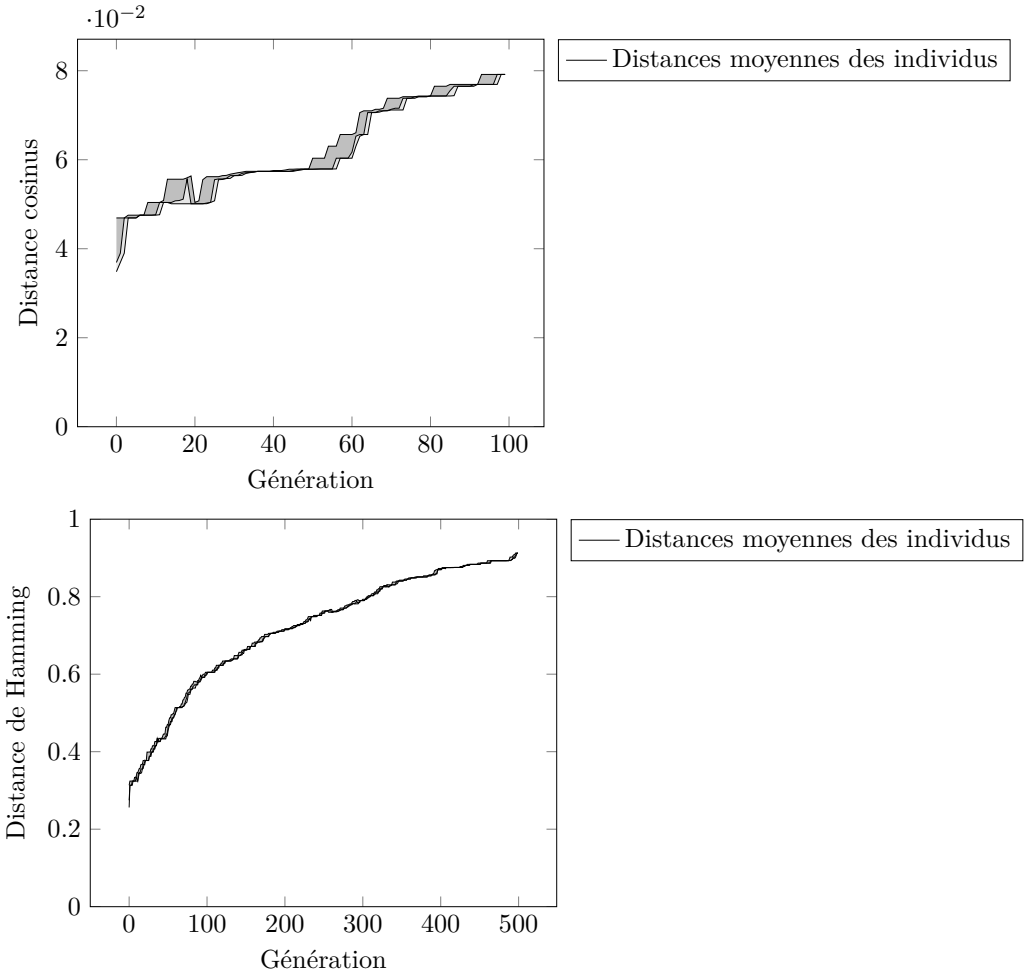


FIGURE A.13 – Évolution des distances moyens des modèles par individu sur une population de 20 individus avec $m = 5$ avec comme objectif la somme du minimum et du la moyenne des distances.

Bi-objectifs minimum et moyenne

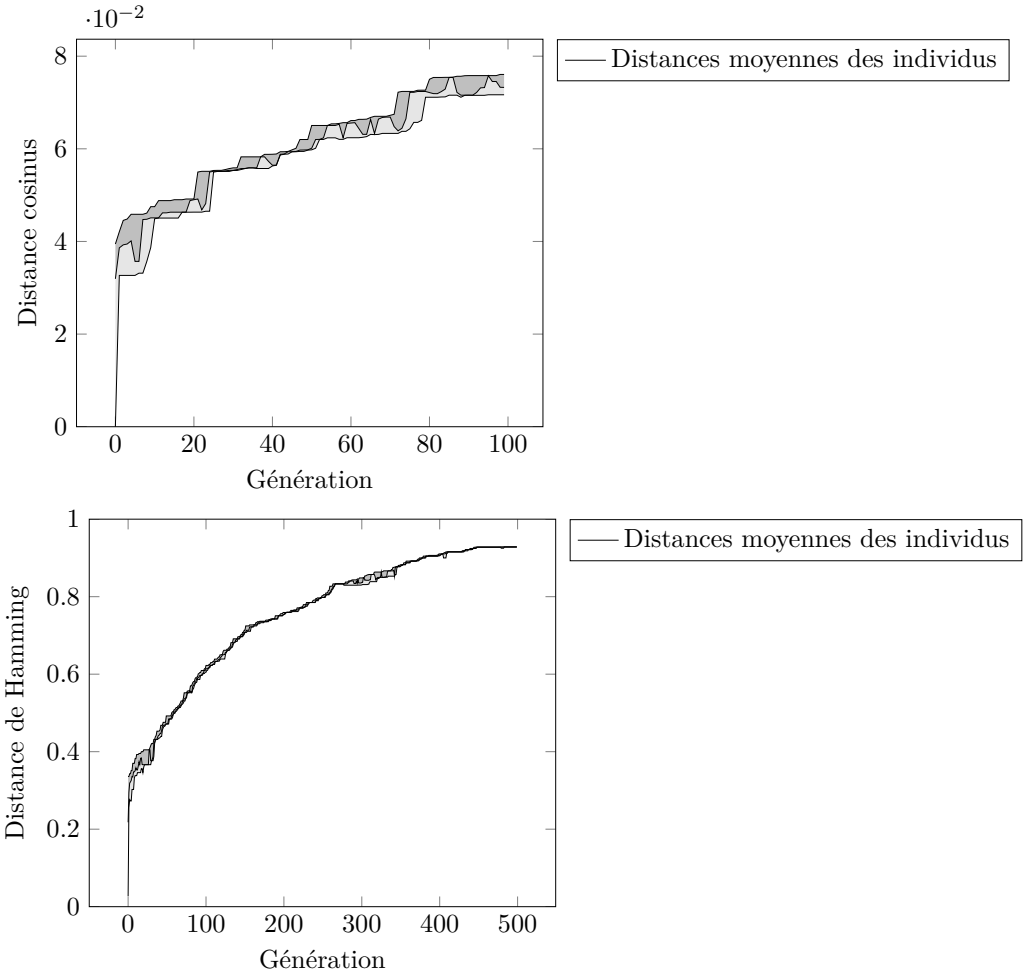


FIGURE A.14 – Évolution des distances moyens des modèles par individu sur une population de 20 individus avec $m = 5$ avec comme objectifs le minimum et la moyenne des distances.

Multi-objectifs

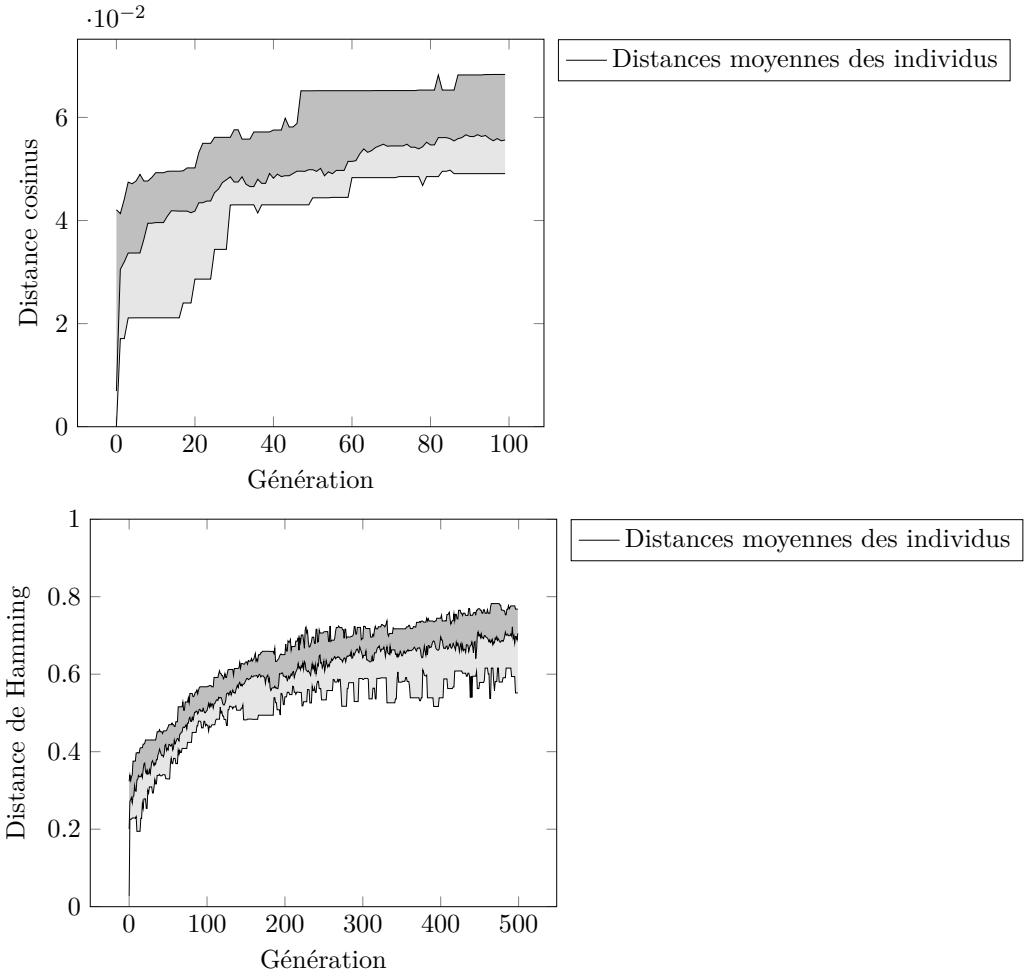


FIGURE A.15 – Évolution des distances moyens des modèles par individu sur une population de 20 individus avec $m = 5$ avec chaque distance traitée comme un objectif.

A.4.4 Cas $m = 1$

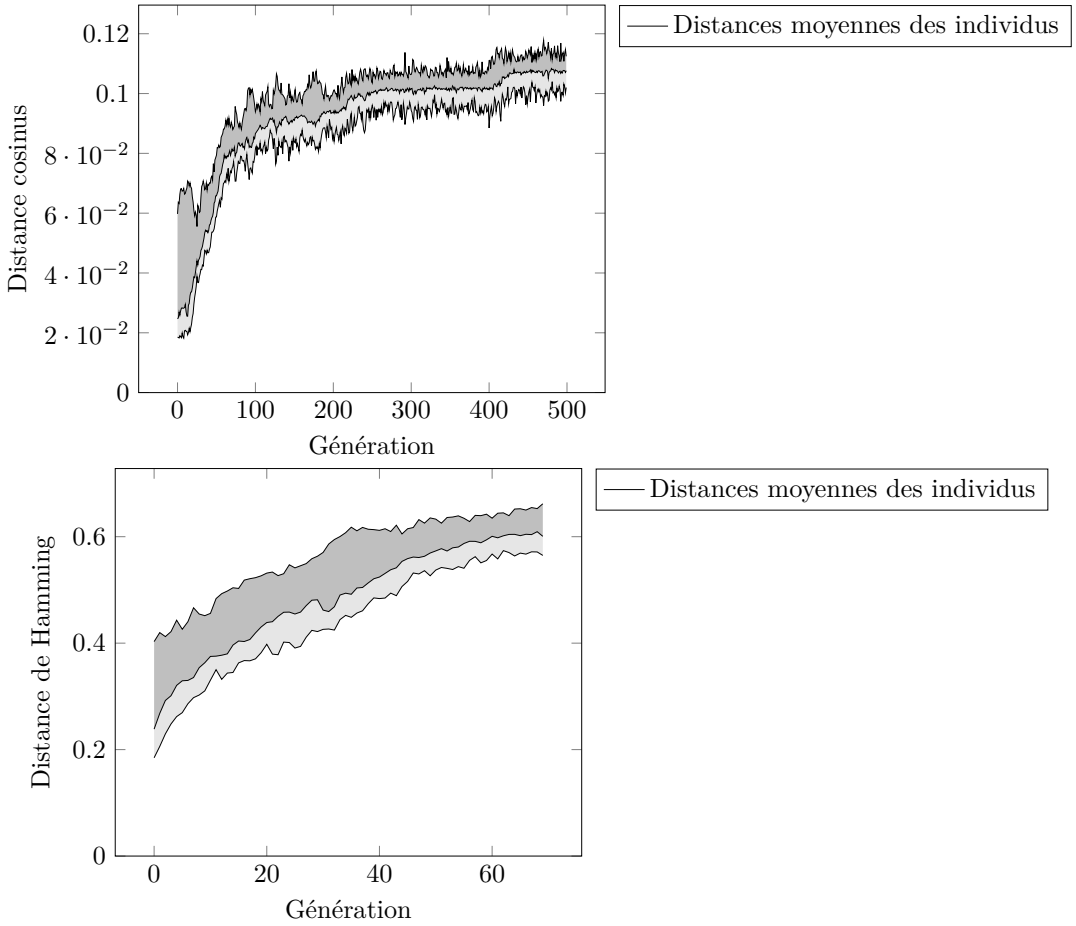


FIGURE A.16 – Évolution des distances moyens des modèles par individu sur une population de 100 individus avec $m = 1$