Des algorithmes génétiques pour générer des modèles diversifiés

Étude bibliographique

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

21 mai 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. Dans cette étude bibliographique, nous présentons un état de l'art de ces algorithmes. La notion de diversité nécessitant le calcul de distances, nous présentons également un ensemble de mesures de distances entre modèles. Nous finissons en présentant les problématiques inhérentes à ce sujet, ainsi que les pistes qui seront explorées durant le stage pour tenter de les résoudre.

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 litterature review, we present a state of the art of these algorithms. Since diversity's notion requires distances computation, we present a set of distance measurements between models. We finish by presenting problems inherent to this topic, as well as research strategies that we will explore during the internship to try to solve them.

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

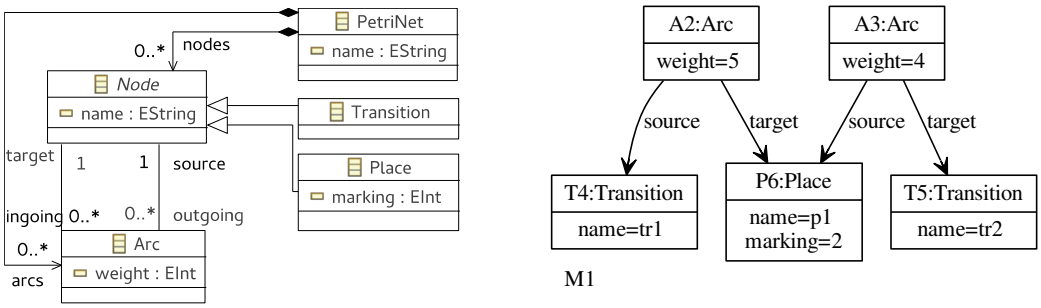


FIGURE 1.1 : Méta-modèle de réseau de Petri extrait de [1] et exemple de modèle instance de ce méta-modèle

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. 2.8), 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).

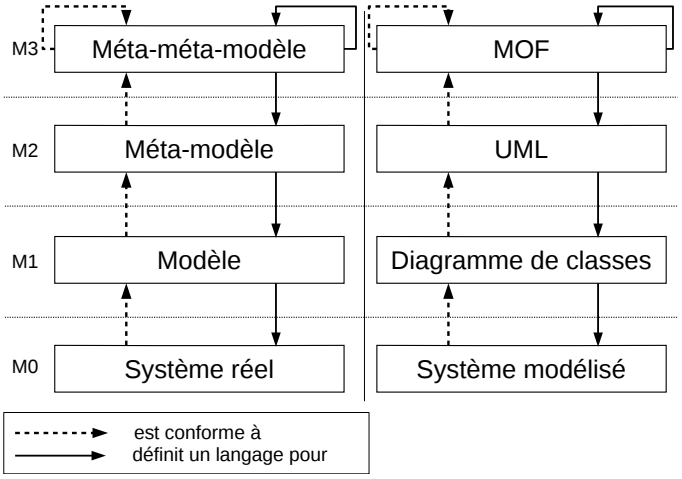


FIGURE 1.2 : Les quatre niveaux d'abstractions en modélisation : M0 : le système réel ; M1 : les modèles ; M2 : les méta-modèles ; M3 : les méta-méta-modèles.

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 ([5]). 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.

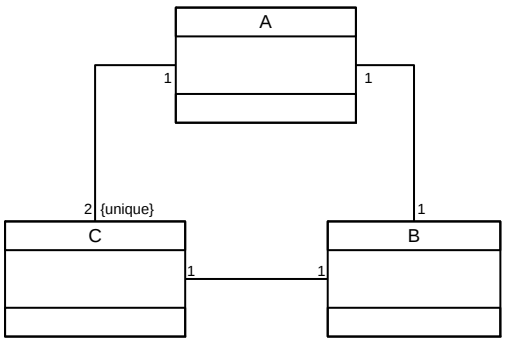


FIGURE 1.3 : Exemple de modèle non instanciable

Un des problèmes de la génération des modèles est un problème de diversité. 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 différent 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.

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, 6, 7]) ou par recuit simulé ([2]) ont par exemple été utilisées pour l'instanciation de méta-modèles. L'approche par algorithme génétique proposée dans [8] 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 les différentes problématiques liées au sujet ainsi que les pistes de recherche envisagées dans le cadre de la génération de modèles par algorithmes génétiques. Enfin, nous conclurons ce rapport.

2

État de l'art

Dans cette section, les différentes problématiques du sujet seront exposées. En effet, notre étude s'articule autour de trois axes (cf. fig. 2.1). Les techniques combinatoires déjà existantes pour la génération de modèles seront exposées, puis les problématiques de diversité et de distance entre modèles seront abordées.

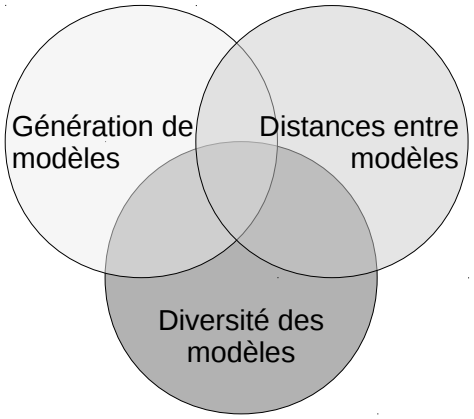


FIGURE 2.1 : Différentes problématiques du sujet

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, 6, 7]). Les CSP sont définis comme un ensemble de variables, un ensemble des domaines associés à ces variables ainsi qu'un ensemble de contraintes. Cette technique consiste à exprimer un méta-modèle comme un problème de satisfaction de contraintes et à utiliser un solveur pour produire des instances valides de ce méta-modèle. Cette

approche permet en outre la prise en compte des contraintes OCL de modèles en les incorporant dans les contraintes du CSP.

Dans [9], Sen et al. utilisent le langage de contraintes Alloy ([10]) 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 [2], 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. Cette dernière est calculée à partir du modèle instance du méta-modèle et de l'ensemble des fragments du méta-modèle ; ces fragments sont définis par Cadavid et al. dans [2] tels que :

Définition 1 *Un fragment de modèle (MF) d'un méta-modèle MM est un tuple de :*

- *Une classe du méta-modèle MM.*
- *Une propriété ou une relation de la classe pour laquelle il est spécifié un intervalle de valeurs ou de multiplicité.*
- *L'intervalle de valeurs pour la propriété.*

Un fragment de modèle est couvert par un modèle si ce modèle contient un objet avec la propriété ayant une valeur dans l'intervalle du fragment de modèle.

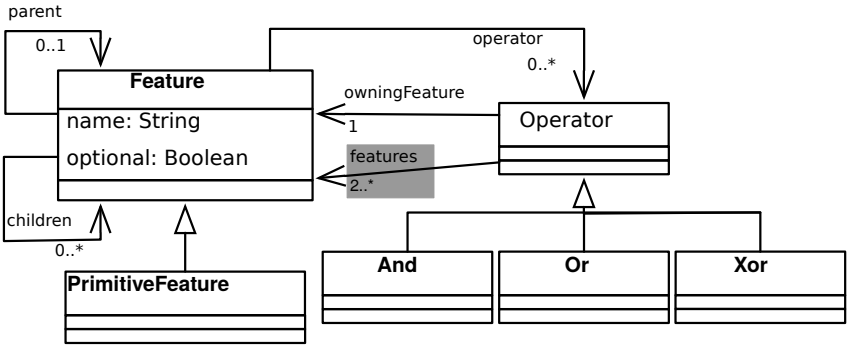
Dans la figure 2.2, le fragment surligné est couvert par tous les modèles ayant une instance de *Operator* en relation par la relation $features$ avec au moins 2 instances de $Feature$; c'est ainsi la relation surlignée sur le modèle qui est couverte par ce fragment (ces modèles ne couvriront par conséquent pas le fragment 13). Les auteurs définissent alors la couverture d'un modèle (cf. eq. 2.1) comme étant le nombre de fragments couverts par le modèle ($CMF(sm)$) divisé par le nombre total de fragments du méta-modèle (MFP).

$$coverage(sm) = \frac{|CMF(sm)|}{|MFP|} \tag{2.1}$$

La dissimilarité est également calculée à partir de ces fragments. En effet, Cadavid et al. la définissent à partir du nombre de fragments couverts en excès (plus d'une fois) ainsi que du nombre total de fragments couverts (cf. eq. 2.2 – avec EC le nombre de couverture excessive de fragments et $MFRT$ un paramètre de tolérance).

$$dissimilarity(sm) = max\left(0, 1 - \frac{EC}{MFRT \times |CMF(sm)|}\right) \tag{2.2}$$

Cette mesure peut cependant donner des résultats identiques pour des modèles différents dans le cas où ces modèles couvrent un nombre identique de fragments différents




```
1: mf(Feature,name,StringRange
   (regexp: ))
2: mf(Feature,name,StringRange
   (regexp: +))
3: mf(Feature,optional,BooleanRange
   (boolValue: true))
4: mf(Feature,optional,BooleanRange
   (boolValue: false))
5: mf(Feature,operator,IntRange
   (lower: 0, upper: 0))
6: mf(Feature,operator,IntRange
   (lower: 1, upper: 1))
7: mf(Feature,operator,IntRange
   (lower: 2, upper: MAX_INT))
```

```
8: mf(Feature,children,IntRange
   (lower: 0, upper: 0))
9: mf(Feature,children,IntRange
   (lower: 1, upper: 1))
10: mf(Feature,children,IntRange
   (lower: 2, upper: MAX_INT))
11: mf(Feature,parent,IntRange
   (lower: 0, upper: 0))
12: mf(Feature,parent,IntRange
   (lower: 1, upper: 1))
13: mf(Operator,owningFeature,IntRange
   (lower: 1, upper: 1))
14: mf(Operator,features,IntRange
   (lower: 2, upper: MAX_INT))
```

FIGURE 2.2 : Exemple de méta-modèle et de fragments associés utilisé par Cadavid et al. [2].

(CMF identiques) et avec des excès égaux (EC identiques). En outre, 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 ([11, 12]) sont des méthodes heuristiques basées sur le 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 2 *Un **chromosome** (ou individu) est une représentation d'une solution possible de l'espace des solutions, constitué d'un ensemble de gènes (dans [11], Holland*

décrit un chromosome comme une chaîne de bits).

Définition 3 *Un **gène** est un élément atomique permettant le codage de la représentation du chromosome (e.g. un bit dans une chaîne de bits).*

Définition 4 *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 la solution la plus adaptée (i.e. la 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 5 *Un **croisement** de chromosomes est une fonction qui permet d'obtenir de nouveaux chromosomes composés des gènes des chromosomes d'entrée.*

Définition 6 *Une **mutation** est une fonction qui permet d'obtenir un nouveau chromosome en modifiant un ou plusieurs gènes d'un chromosome déjà existant.*

Définition 7 *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. 2.3), 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.

Définition 8 *Une **génération** n est l'ensemble des chromosomes obtenus lors de* $\mathrm{n}^{ième}$ *itération.*

Définition 9 *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

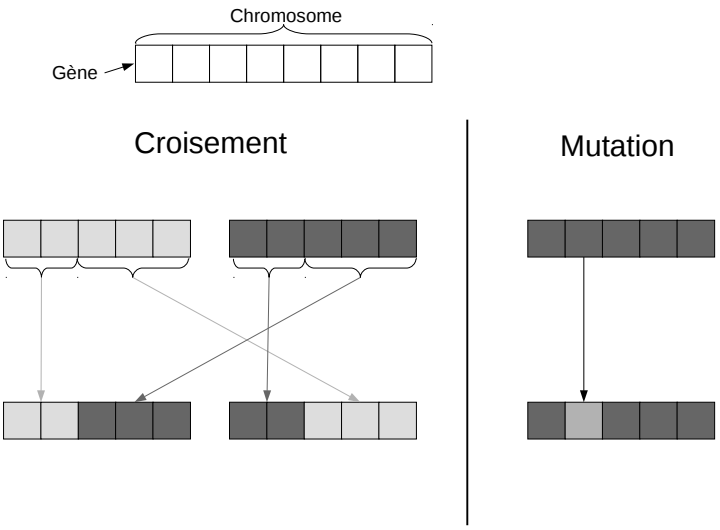


FIGURE 2.3 : Exemple de croisement et de mutation sur des chromosomes. À gauche, croisement en un point (translocation) de deux chromosomes ; à droite, mutation d'un chromosome.

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 10 *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. 2.3 – où F est l'ensemble des fonctions objectifs et dans le cas où on cherche à minimiser le score de la fonction objectif).

$$\begin{aligned} &\forall f_i \in F, f_i(x_1) \leq f_i(x_2) \wedge \\ &\exists f_i \in F, f_i(x_1) < f_i(x_2) \end{aligned} \tag{2.3}$$

Définition 11 *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 Utilisation dans l'ingénierie des modèles

Des algorithmes génétiques ont déjà été utilisés dans le cadre de l'IDM. En effet, dans [13], Bouktif et al. utilisent les algorithmes génétiques dans une optique de prédiction de la qualité d'un logiciel orienté-objet. Dans [14], les auteurs souhaitent apparier des éléments d'un modèle d'entrée d'une transformation aux éléments générés. Ils ont choisi pour cela d'utiliser l'algorithme génétique *NSGA-II* ([3]), en fragmentant leur modèle d'entrée et leur modèle de sortie. Ce même algorithme est également utilisé par Ouni et al. ([15]) pour la réingénierie, recherchant les opérations de refactorisation ayant les meilleurs résultats pour leurs trois objectifs.

Des travaux sur la génération de modèles grâce à des algorithmes génétiques existent également. Ainsi, dans [8], les auteurs proposent une approche de la génération de modèles de tests qui est basée sur un algorithme génétique. En effet, ils génèrent, à partir de fragments de modèles déjà existants, de nouveaux modèles qui sont des combinaisons de fragments. La décision des stratégies à appliquer lors des différents choix à faire influe fortement sur la diversité des modèles de sortie. Par exemple, lors de la complétion d'un modèle en cours de création, le choix de créer un nouvel objet à partir des fragments ou de réutiliser un objet existant va avoir une influence sur la forme générale du modèle généré.

Cette approche a cependant des limites. En effet, dans le cas ici présenté, les modèles d'entrées sont des modèles fournis par l'utilisateur et non pas générés automatiquement. En outre, les contraintes OCL ne sont ici pas vérifiées, ainsi on n'obtient pas que des modèles valides.

2.2.2 Quelques exemples d'algorithmes génétiques

Nous avons choisi d'étudier les algorithmes multi-objectifs existants et avons effectué une sélection d'algorithmes utilisés comme références dans la littérature lors des expérimentations, ainsi que d'un algorithme plus récent. Un récapitulatif des différents algorithmes est donné dans la table 2.1.

NSGA-II et NSGA-III

L'algorithme *NSGA-II*[1] (*nondominated sorting genetic algorithm II* – proposé par Deb et al. dans [3]) 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.

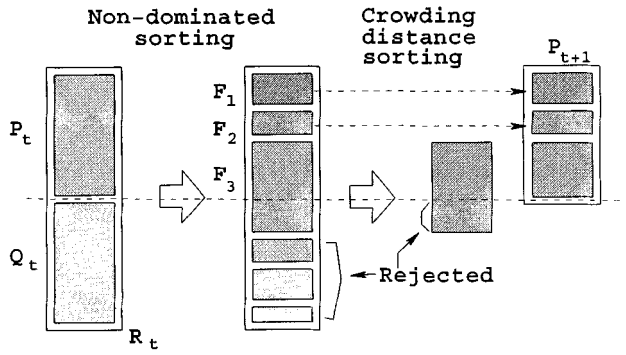


FIGURE 2.4 : Procédure de NSGA-II extraite de [3]

La principale nouveauté de cet algorithme est l'intégration d'un processus de conservation des distances : la *crowding distance* (cf. fig. 2.4). 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.

Cette distance 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* [16] et *U-NSGA-III* [17]) qui sont des optimisations pour les cas où le nombre de fonctions objectifs est situé entre 3 et 15 (elles possèdent une complexité $O(N^2 log^{M-2} N)$).

[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`

IBEA

Zitzler et Künzli proposent dans [18] l'algorithme *IBEA*[2] (*Indicator-Based Selection in Multiobjective Search*) qui possède une complexité similaire à *NSGA-II*. Celui-ci procède non pas par sélection des meilleurs individus, mais par élimination des moins bons. En effet, à chaque génération, un score objectif (basé sur un indicateur de qualité) est associé à chaque chromosome. Tant que la taille de la population excède la taille maximum, l'individu ayant le plus faible score est éliminé, et le score est recalculé. La sélection des individus pour la reproduction s'effectue également par tournoi binaire, avec croisements et mutations. Cet algorithme ne propose aucun mécanisme de préservation de la diversité supplémentaire, l'indicateur pouvant être un indicateur de distance par exemple.

(μ +λ)-PAES

Dans [19], Knowles et Corne proposent l'algorithme $(\mu + \lambda) - PAES$. L'initialisation s'effectue par la génération de μ solutions qui sont ajoutées à une archive. Une de ces solutions est ensuite sélectionnée par tournoi binaire, et est mutée en λ nouveaux chromosomes. Si les individus ne sont dominés par aucune des solutions déjà dans l'archive, ils sont ajoutés à celle-ci. Basés sur leur score et sur leur distance par rapport aux autres solutions, μ chromosomes sont sélectionnés comme solutions courantes jusqu'à ce que les critères de terminaisons soient atteints. Si sa variante *(1+1)-PAES* offre une meilleure complexité que les algorithmes évoqués précédement ($O(AN)$ avec N le nombre d'itérations et A la taille maximale de l'archive), cet algorithme ne possède cependant aucun mécanisme de croisement. En effet, *(1+1)-PAES* sélectionne pour chaque génération un unique candidat à la mutation et ne crée qu'un unique nouveau chromosome.

SPEA2

L'algorithme *SPEA2*[3] présenté dans [20] est une amélioration de l'algorithme *SPEA* (non élitiste). Si cet algorithme est en moyenne moins rapide que *NSGA-II*, il permet cependant en moyenne de générer des résultats qui possèdent une diversité plus élevée. Comme *IBEA*, l'algorithme *SPEA2* fonctionne avec une archive. À chaque itération, les individus non dominés de la population parente et ceux de la progéniture sont sélectionnés dans l'archive. S'il y a trop peu de chromosomes dans l'archive, alors les meilleurs individus dominés des deux populations sont utilisés pour compléter l'archive. Si le nombre de chromosomes est au contraire trop élevé, un processus de sélection environnemental est effectué, et les individus possédant les plus faibles distances aux autres chromosomes sont retirés lors de la troncature de l'archive.

[2]Une implémentation de cet algorithme en `C` est disponible sur le site de `http://www.tik.ethz.ch/~sop/pisa/`

[3]Une implémentation de cet algorithme en `C` est disponible sur le site de `http://www.tik.ethz.ch/~sop/pisa/`

ETEA

Plus récemment, l'algorithme *ETEA*[4] ([21]) a été proposé. Il possède un schéma similaire à *SPEA2* et, bien qu'étant de complexité plus élevée que les autres algorithmes récents cités jusqu'à présent ($O(N^3)$), l'utilisation de l'*ETCD* (*Euclidean minimum spanning Tree Crowding Distance*) lors de la troncature de trop grosses archives lui permet d'obtenir de très bons résultats dans la diversité des chromosomes conservés, ce qui en fait un bon candidat potentiel pour notre étude.

Algorithme	**Complexité**[a]	**Technique de diversité**	**Croisement**	**Mutation**	**Implémentation**[b]
NSGA	$O(MN^3)$	Technique de partage	✓	✓	✓
NSGA-II	$O(MN^2)$	Distance de foule	✓	✓	✓
NSGA-III	$O(N^2 log^{M-2} N)$	Ensemble de directions de référence	✓	✓	
IBEA	$O(MN^2)$	Sélection environnementale	✓	✓	✓
$(1+1)$-PAES	$O(AN)$	Algorithme de grille adaptative		✓	
SPEA2	$O(N^2 logN))$	Troncature d'archive	✓	✓	✓
ETEA	$O(N^3)$	Arbre euclidien couvrant de poids minimal au sens de la distance de foule	✓	✓	✓

[a] N est le nombre de générations, M est le nombre de fonctions objectifs, A est la taille de l'archive
[b] Au moment de la rédaction de ce rapport, le code de $(\mu + \lambda)$-PAES sur le site de Joshua Knowles (`http://www.cs.bham.ac.uk/~jdk/multi/`) n'était plus disponible.

TABLE 2.1: Différents algorithmes étudiés et leurs caractéristiques

2.2.3 Algorithmes mémétiques

Les algorithmes mémétiques ([22]) sont des algorithmes évolutionnaires hybrides. En effet, une recherche locale de solutions voisines (comme la recherche tabou ([23]) par exemple qui consiste en une exploration des solutions voisines de la solution actuelle tout en excluant les solutions déjà visitées) est effectuée sur les nouveaux individus dans le but de produire une plus grande diversité de solutions. Dans cette optique, ces algorithmes évolutionnistes semblent par conséquent être adaptés à notre problème.

Dans [4], Sörensen et Sevaux proposent un algorithme mémétique avec gestion de population (*MA/PM – Memetic Algorithms with Population Management*). Cet algorithme possède un fonctionnement similaire aux algorithmes génétiques. Cependant, la distance des nouveaux individus est considérée avant ajout de ces individus. Elle est calculée par rapport à la population actuelle et non pas par rapport aux scores des fonctions objectifs comme c'est le cas dans les algorithmes génétiques classiques. Cela permet de maintenir une certaine diversité, en imposant par exemple un seuil de distance Δ minimum. Ainsi, les individus dont la distance est inférieure à ce seuil sont mutés jusqu'à obtention

[4] Une implémentation de cet algorithme en `C` est disponible sur le site de *Miqing Li* – `http://www.brunel.ac.uk/~cspgmml1/Publication.html`

d'un chromosome suffisamment éloigné. Dans leurs travaux, les auteurs décrivent quatre stratégies possibles pour ce seuil (cf. fig. 2.5) :

(a) un Δ fixe ;

(b) un Δ décroissant (exploration des éléments de plus en plus proches) ;

(c) un Δ de départ faible évoluant jusqu'à un Δ fixe (exploration des éléments de plus en plus éloignés) ;

(d) un Δ de départ élevé et de plus en plus faible jusqu'à ce que les nouvelles solutions n'améliorent plus la fonction objectif ; le seuil est alors à nouveau élevé afin d'explorer plus de solutions.

Les auteurs ont notamment testé les première et troisième stratégies et obtenu de meilleurs résultats pour cette dernière. L'exploration des autres stratégies n'a pas été effectuée.

Cet algorithme permet par conséquent de travailler avec une population beaucoup plus réduite, la stratégie d'exploration de l'espace des solutions étant beaucoup plus agressive (i.e. la combinaison des croisements, mutations et recherche locale permettant l'exploration plus rapide de nouvelles solutions) qu'avec les algorithmes évolutionnaires classiques. Cependant, les mutations répétées des individus peut entraîner une importante perte d'informations, et dans notre cas, générer un nombre important de modèles invalides.

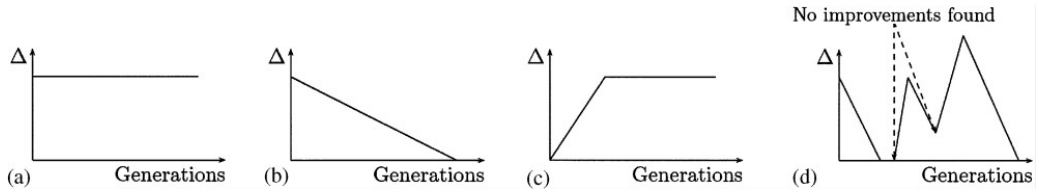


FIGURE 2.5 : Différentes stratégies proposées par Sörensen et Sevaux [4].

Dans [24], les auteurs proposent un algorithme sans mutation afin d'éviter cette dégradation. Ainsi, au lieu de muter à répétition un individu dont la distance est trop faible, les auteurs font le choix de simplement l'éliminer. Afin de réduire les pertes, Prins et al. proposent cependant de ne pas systématiser le processus de recherche locale et imposent ainsi un pourcentage de chance pour cette recherche.

Boudia et Prins proposent dans [25] une nouvelle version de cet algorithme sans mutation. Une population de taille N est générée aléatoirement à l'initialisation et est triée, le meilleur individu étant le premier. À chaque itération, un nouveau chromosome est obtenu par croisement (après une sélection par tournoi binaire des parents) et possède également une chance d'être remplacé par un voisin par recherche locale. Si le nouvel individu ainsi obtenu possède un meilleur score que le meilleur individu de la population (le premier) ou est à distance suffisante de la population, il est sélectionné pour remplacer un des individus possédant un des moins bons scores. La population est ensuite partiellement renouvelée.

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 [26], Falleri et al. utilisent l'algorithme de *Similarity Flooding* ([27]) afin de mettre en relations des concepts proches dans deux méta-modèles pour la génération de transformations de modèles. Dans [28], 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 ([29]). 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 ([30]), où les auteurs adaptent la distance de Levenshtein ([31]) 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 ([32]), 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 [33], 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 [34] une approximation de cette distance en complexité quadratique.

Distance de centralité

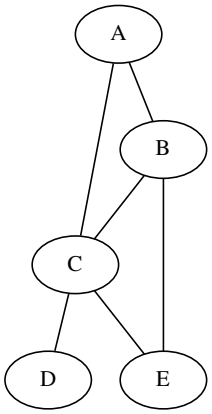
Dans [35], Roy et al. proposent une mesure de distance basée sur la centralité des nœuds d'un graphe ([36]). La centralité d'un nœud est une fonction qui, pour un nœud d'un graphe donné, associe une valeur réelle. Ainsi, pour un graphe $G = (V, E)$, la centralité est une fonction définie sur les sommets :

$$\begin{aligned} C\colon V &\longrightarrow \mathbb{R}^+ \\ v &\longmapsto C(G, v) \end{aligned} \tag{2.4}$$

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. 2.6.



Nœud	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 2.6 : 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 2.5).

$$d_C(G_1, G_2) = \sum_{v \in V} |C(G_1, v) - C(G_2, v)| \tag{2.5}$$

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 [37] 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

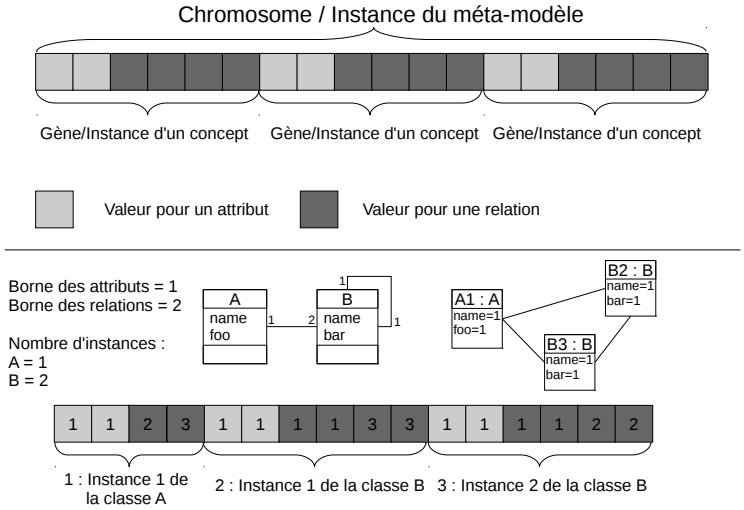


FIGURE 2.7 : Exemple de représentation d'une instance de méta-modèle comme vecteur.

Dans leurs travaux ([1]), Ferdjoukh et al. modélisent le méta-modèle comme un ensemble de variables, de contraintes et de domaines CSP. Ainsi, une instance de ce méta-modèle peut être représentée comme un vecteur qui pour chaque variable associe une valeur de son domaine (cf fig. 2.7). Une différence entre deux modèles peut alors être définie comme la différence entre les deux vecteurs représentatifs.

Les distances de Hamming ([38]) ou de Levenshtein ([31]) 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 ([39]) 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 :

$$D_C(\overrightarrow{V_1}, \overrightarrow{V_2}) = 1 - S_C(\overrightarrow{V_1}, \overrightarrow{V_2}) \tag{2.6}$$

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

$$\begin{aligned} S_C(\overrightarrow{V_1}, \overrightarrow{V_2}) &= \frac{\overrightarrow{V_1} \cdot \overrightarrow{V_2}}{\|\overrightarrow{V_1}\| \|\overrightarrow{V_2}\|} \\ &= \frac{\sum_{i=1}^{n} V_1[i] V_2[i]}{\sqrt{\sum_{i=1}^{n} V_1[i]^2} \sqrt{\sum_{i=1}^{n} V_2[i]^2}} \end{aligned} \tag{2.7}$$

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.

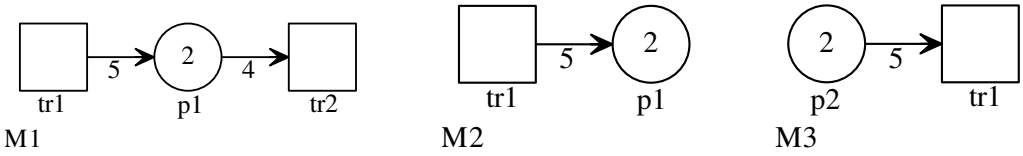
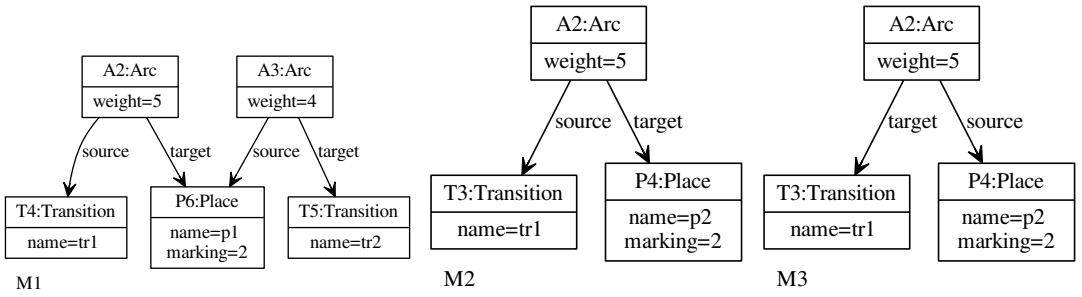


FIGURE 2.8 : Réseaux de Petri associés aux modèles de la fig. 2.9



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 2.9 : Mesures des différentes distances entre les modèles M1, M2 et M3

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. 2.8) sont données dans la fig. 2.9. 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.

3

Discussions

3.1 Problématiques

Dans [1], les auteurs définissent des critères pour la génération de modèle, dont :

- la scalabilité, le temps de calcul pour la génération de modèles de grandes tailles devant rester raisonable ;
- la validité, les modèles fournis devant être des instances du méta-modèle respectant les contraintes ;
- la diversité, les modèles générés devant être suffisamment distincts les uns des autres.

Les travaux à base de CSP ([1, 6, 7]) donnent des résultats qui sont conformes aux méta-modèles mais également qui respectent les contraintes OCL et par conséquent satisfont les deux premiers points. Le troisième point est plus difficile à mettre en œuvre dans le cadre de la programmation par contrainte ; en effet, il est possible d'introduire la notion de distance entre deux solutions pour tenter de les éloigner mais la génération unique de solutions diverses dans leur ensemble est complexe, tandis que l'approche d'exploration itérative des algorithmes génétiques permet quant à elle d'approcher cet objectif.

3.1.1 Fonction objectif inter-chromosomes ou intra-chromosome

Les *MOEA* (*Multi-Objective Evolutionary Algorithm* – dont font partie les différents algorithmes présentés en 2.2.2) fonctionnent à partir d'une population de chromosomes. Un chromosome est un des éléments dont on souhaite maximiser les fonctions objectif étudiées.

Cependant, notre problème étant un problème de diversité, la question qui se pose est alors un problème de définition de la diversité des modèles générés. Le choix de la fonction objectif est discuté en 3.1.3. Le problème posé par l'utilisation d'une fonction objectif de distance dans le cadre d'un algorithme génétique est que les fonctions objectif sont normalement calculées sur un chromosome (en intra-chromosome – cf. fig. 3.1). Or,

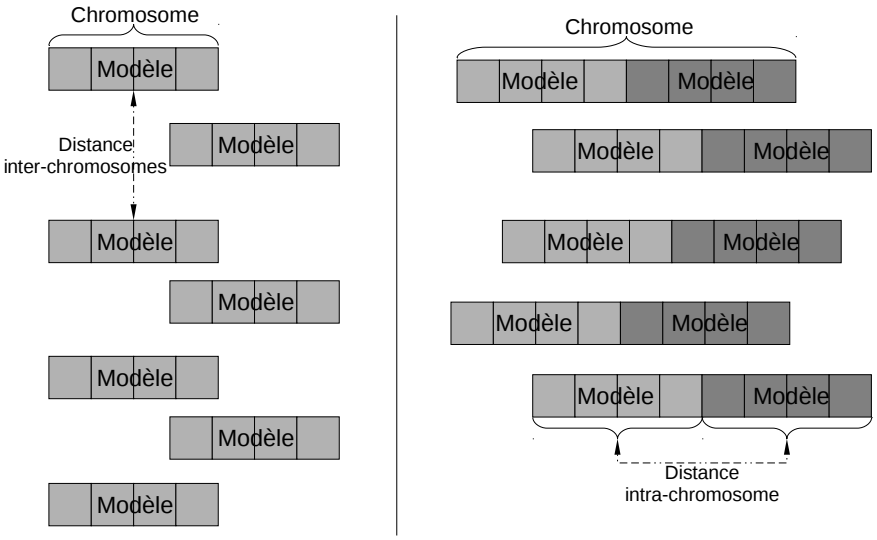


FIGURE 3.1 : Deux hypothèses possibles : à gauche, chromosome constitué d'un seul modèle et fonction inter-chromosomes ; à droite, chromosome constitué de plusieurs modèles et fonction intra-chromosome.

une fonction objectif de distance implique l'utilisation de l'ensemble de la population pour son calcul, ce qui implique une fonction objectif inter-chromosomes.

Une des solutions envisagées est l'adaptation de l'algorithme afin d'intégrer l'usage d'une fonction objectif inter-chromosomes dans l'exécution de celui-ci. L'algorithme permettrait de générer une population de modèles diversifiés, les mutations et croisements entre éléments du modèles permettant de créer de nouveaux modèles.

L'autre solution envisagée est l'utilisation non pas d'un modèle pour un chromosome, mais d'un ensemble de modèles pour un chromosome. La fonction objectif redeviendrait ainsi une fonction objectif intra-chromosome, et l'algorithme génèrerait alors une population d'ensembles de modèles, chacun d'entre eux maximisant la fonction objectif de diversité. Bien qu'en apparence plus simple à mettre en place, cette solution possède cependant un certain nombre de points faibles ; en effet, la taille des chromosomes serait potentiellement très importante (un chromosome représentant plusieurs modèles), et la question des points de croisements serait alors potentiellement plus complexe (si l'on ne croise que les modèles et non leurs éléments, les résultats seraient-ils aussi satisfaisants ?).

Nous avons ainsi fait le choix d'analyser le code de *NSGA-II* et *ETEA* fournis afin d'examiner la difficulté d'intégrer une fonction objectif inter-chromosome et de tester les deux solutions dans le but de voir quels sont les meilleurs résultats.

Il existe également un certain nombre de bibliothèques d'algorithmes génétiques pouvant être utilisées. Ainsi, la bibliothèque Python *DEAP*[1] (*Distributed Evolutionary Algorithms in Python*) met à disposition un certain nombre d'outils pour la création d'al-

[1] `https://pypi.python.org/pypi/deap`

gorithme et la manipulation de chromosomes, tout comme la bibliothèque C *GAUL*[2] ou la plus récente API Java *Jenetics*[3]. La bibliothèque *GAlib*[4] est quant à elle disponible en C++ et fournit un certain nombre de fonctionnalités, notamment la possibilité d'utiliser n'importe quel type C++ comme chromosome (en utilisant le polymorphisme pour redéfinir les méthodes de croisements et mutations) ainsi que la possibilité d'utiliser une fonction objectif intra ou inter-chromosomes.

3.1.2 Croisements et mutations

Lors du stage, une population initiale déjà existante sera utilisée (générée à partir de l'outil *Grimm*[1]). La génération complète (sans utilisation d'une population pré-existante mais plutôt à partir de fragments par exemple comme proposé dans [8]) est une piste également possible, qui pourrait éventuellement être étudiée si nous avons le temps. L'outil *Grimm* crée des instances valides d'un méta-modèle grâce aux CSP. Il utilise pour cela un solveur CSP supportant le format XSCP.

L'utilisation de celui-ci nous permettrait de vérifier à chaque croisement et mutation que les nouveaux modèles sont des vecteurs valides. En outre, la possibilité de mettre en place un mécanisme de réparation lors des croisements non valides sera abordée.

Par ailleurs, il existe un certain nombre de croisements possibles, les implémentations fournies de *NSGA-II* et *ETEA* utilisant des croisements en 2 points pour les valeurs binaires et les croisements *SBX* (*Simulated Binary Crossover*) pour les valeurs réelles. Le choix de ceux-ci pourrait également avoir une influence sur les résultats générés.

En effet, nous ne pouvons pas utiliser la simulation de croisement binaire comme croisement, celle-ci effectuant des opérations modifiant la valeur des gènes dans le but de conserver une valeur moyenne constante sur le chromosome. Or, nous ne voulons pas que ce cas advienne, nos valeurs étant des valeurs appartenant à un domaine bien défini.

Une des autres difficultés sera de définir les points de coupe pour nos modèles afin que les croisements génèrent le moins possibles de modèles non conformes.

3.1.3 Choix de la fonction objectif

La notion de diversité entre modèles est un problème difficile. En effet, les modèles sont des données complexes et il est difficile de définir quelle distance sépare deux concepts.

Nous avons décidé d'explorer plusieurs pistes dans le cadre de la fonction objectif à utiliser :

- la distance d'édition de graphe ;
- la distance de centralité ;
- l'idée d'utiliser une version modifiée du PageRank pour donner un poids à certains éléments du graphe a également été abordée ;
- la distance cosinus.

[2] `http://gaul.sourceforge.net/`
[3] `http://jenetics.io/`
[4] `http://lancet.mit.edu/ga/`

Pour chaque version de l'algorithme choisi (inter-chromosomes et intra-chromosome), nous pourrons ainsi étudier les résultats fournis par les différentes fonctions objectif afin d'en déduire laquelle permet d'obtenir la plus grande diversité.

Par ailleurs, les algorithmes choisis étant des algorithmes multi-objectifs, nous avons également la possibilité d'utiliser plusieurs de ces mesures afin d'affiner nos résultats, en utilisant par exemple une mesure structurelle comme la centralité couplée avec une mesure plus comparative comme la distance cosinus.

La principale difficulté sera cependant le calcul de ces différentes distances lors des différentes itérations. En effet, certaines distances possédants une complexité élevée (cf. 2.3.1), l'exécution à chaque génération sera potentiellement coûteuse. Or, l'admission de nouveaux individus dans la population à chaque génération impliquera forcément d'effectuer ce calcul. L'utilisation d'heuristiques pour le calcul des distances sera ainsi à étudier, et les algorithmes qui proposent une utilisation réduite des fonctions objectifs (en ne les recalculant pas pour tous les individus à chaque génération par exemple) seront également potentiellement beaucoup plus intéressant.

3.2 Pistes de recherches

La génération de modèles à l'aide d'algorithmes génétiques pose un certain nombre de problématiques. Afin de tenter de trouver des solutions à celles-ci, des pistes sont à explorer :

- dans le choix des fonctions objectifs ;
- dans le choix de l'application de ces fonctions ;
- dans le choix de l'application des croisements et mutations.

Par conséquent, l'objectif du stage sera de valider ou invalider expérimentalement les différents choix possibles. Pour cela, nous explorerons les combinaisons entre :

- fonction intra-chromosome ;
- fonction inter-chromosomes ;

et les différentes fonctions objectifs (et potentiellement des combinaisons de celles-ci) et observerons les modèles ainsi générés.

4

Conclusion

La diversité est un problème important dans la génération de modèles ; elle permet d'apporter des exemples fiables et couvrant le maximum de cas de figure. De nombreuses approches pour la génération de modèles ont été étudiées, notamment à partir de CSP, de l'algorithme de recuit simulé ou d'algorithmes génétiques. Si l'approche par CSP permet d'obtenir de très bons résultats en terme de validité des modèles par rapport au méta-modèle, la problèmatique de diversité semble cependant être un problème plus facilement résolu par les méta-heuristiques de recuit simulé et d'algorithmes génétiques. C'est cette dernière approche que nous avons décidé d'utiliser.

Un certain nombre de problématiques demeurent cependant, notamment sur le calcul de cette diversité. Le choix d'une ou plusieurs mesures, calculable de façon efficace (ces mesures devant être effectuées pour chaque nouvelle génération), et donnant des résultats fiables, est ainsi un des problèmes que nous devrons résoudre.

Un autre des problèmes sera également celui de la représentation des modèles en tant que chromosome. La représentation intuitive d'un chromosome comme étant un modèle apporte la problématique du calcul d'une fonction objectif inter-chromosomes ainsi que des croisements sur ces chromosomes (où couper et comment muter), tandis que la représentation d'un ensemble de modèles par chromosome apporte sans doute un problème de taille (et donc potentiellement de lourdeur de calcul), mais permettrait un calcul facile des fonctions objectifs et des croisements potentiellement plus aisé (définition d'un gène comme étant un modèle).

L'objectif durant le stage sera ainsi de trouver une implémentation de ces différentes combinaisons et de définir expérimentalement laquelle donne les meilleurs résultats. Les données d'entrées seront générées par une approche par CSP, et nous utiliserons cette même méthode pour vérifier à chaque itération la validité de nos nouveaux modèles. Les résultats seront ensuite comparés aux autres résultats déjà existants.

Bibliographie

[1] Adel Ferdjoukh, Anne-Elisabeth Baert, Eric 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] 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.

[3] 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.

[4] Kenneth Sörensen and Marc Sevaux. MA|PM : memetic algorithms with population management. *Computers & Operations Research*, 33(5) :1214–1225, 2006.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

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

[11] 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.

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

[13] Salah Bouktif, Danielle Azar, Doina Precup, Houari Sahraoui, and Balazs Kegl. Improving rule set based software quality prediction : A genetic algorithm-based approach. *Journal of Object Technology*, 3(4) :227–241, 2004.

[14] Hajer Saada, Marianne Huchard, Clementine Nebut, and Houari Sahraoui. Model matching for Model Transformation a meta-heuristic approach. In *Model-Driven Engineering and Software Development (MODELSWARD), 2014 2nd International Conference on*, pages 174–181, Jan 2014.

[15] Ali Ouni, Marouane Kessentini, Houari Sahraoui, and Mohamed Salah Hamdi. The use of development history in software refactoring using a multi-objective evolutionary algorithm. In *ACM Proceedings of the annual conference on Genetic and evolutionary computation*, pages 1461–1468, 2013.

[16] 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.

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

[18] Eckart Zitzler and Simon Künzli. Indicator-based selection in multiobjective search. In *Parallel Problem Solving from Nature*, pages 832–842. Springer, 2004.

[19] Joshua D. Knowles and David W. Corne. Approximating the nondominated front using the Pareto archived evolution strategy. *Evolutionary computation*, 8(2) :149–172, 2000.

[20] Eckart Zitzler, Marco Laumanns, and Lothar Thiele. SPEA2 : Improving the strength Pareto evolutionary algorithm. *TIK-Report*, (103), 2001.

[21] Miqing Li, Shengxiang Yang, Jinhua Zheng, and Xiaohui Liu. ETEA : A Euclidean Minimum Spanning Tree-Based Evolutionary Algorithm for Multiobjective Optimization. *Evolutionary computation*, 22(2) :189–230, 2014.

[22] Pablo Moscato et al. On evolution, search, optimization, genetic algorithms and martial arts : Towards memetic algorithms. *Caltech concurrent computation program, C3P Report*, 826 :1989, 1989.

[23] Fred Glover. Tabu Search—Part I. *ORSA Journal on Computing*, 1(3) :190–206, 1989.

[24] Christian Prins, Marc Sevaux, and Kenneth Sörensen. A genetic algorithm with population management (ga| pm) for the carp. In *5th Triennal Symposium on Transportation Analysis (Tristan V), Le Gosier, Guadeloupe*, 2004.

[25] Mourad Boudia and Christian Prins. A memetic algorithm with dynamic population management for an integrated production–distribution problem. *European Journal of Operational Research*, 195(3) :703–715, 2009.

[26] 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.

[27] 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.

[28] 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.

[29] 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.

[30] 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.

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

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

[33] 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.

[34] 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.

[35] 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.

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

[37] 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.

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

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