Optimisation du cyclage d'un ouvrage en béton armé via un algorithme génétique

Arthur Calvi[1], Xavier Jourdain[1], Farid Benboudjema[1]

[1] Université Paris-Saclay, ENS Paris-Saclay, CNRS
LMT - Laboratoire de Mécanique et Technologie
4 avenue des sciences 91190 Gif-sur-Yvette
contact : xavier.jourdain@ens-paris-saclay.fr

Résumé : Les données de la maquette numérique associée à un projet réalisé par une approche BIM vont permettre d'utiliser des algorithmes pour effectuer des tâches d'optimisation qui aujourd'hui encore sont majoritairement réalisées par des ingénieurs méthodes. À titre d'exemple, cet article présente une application de recherche de solution optimale de cyclage, *id est* déterminer l'ordre de construction des planchers et des murs d'un bâtiment dont la structure est en béton armé. Ce travail présente une recherche d'optimum par algorithme génétique en le comparant à un simple tirage aléatoire. Afin de déterminer la solution optimale, ce travail propose des critères afin d'objectiver le choix effectué. Parmi eux, la régularité du temps consacré à la réalisation des murs et des planchers pour chacune des phases, la viabilité des solutions suivant si la dalle peut être construite en fonction de l'avancement des murs mais aussi l'évaluation de la distance parcourue par les ouvriers pour chaque mur à réaliser pour chacune des phases, distance obtenue par un algorithme de type *fast marching*.

Mots-clés : Ingénierie des méthodes, cyclage murs et dalles, algorithme génétique, méthode fast marching

Abstract : The data from the digital model associated with a project carried out using a BIM approach will allow algorithms to be used to perform optimization tasks which still today are mainly carried out by method engineers. As an example, this paper presents an application for finding an optimal construction scheduling, in other words to determine for a reinforced concrete building the walls and floors order of construction. This work presents a search for an optimum using a genetic algorithm and comparing it to other approaches. In order to determine the optimal solution, the authors propose criteria to objectify the choice made. Among them, the regularity of the time spent to make walls and floors for each of the phases, the viability of the solutions depending on whether the slab can be built according to the progress of the walls but also the evaluation of the distance travelled by the workers for each wall to be made for each phase, distance obtained by a fast marching algorithm.

Key-words: Methods engineering, construction scheduling, genetic algorithm, fast marching method.

1 Introduction

La quatrième dimension est désormais entrée dans les concepts de base d'un projet réalisé par une approche BIM (Murat Tanyer A. et al. 2005, Tulke 2007). Des travaux ont déjà montré la possibilité de déterminer la séquence de construction d'un pont découpée en grandes tâches (Cengiz Toklu Y. 2011) et de manière plus poussée sur des bâtiments en structure métallique via un algorithme génétique (Faghihi et al. 2014). Ce dernier article se base sur la vérification de l'équilibre statique phase par phase pour valider les solutions mais malgré la qualité de l'article, un ingénieur expérimenté ne pourra que constater le manque de réalisme de certaines solutions proposées avec des poteaux construits sur plusieurs étages sans que les poutres supportant les planchers ne soient réalisées. Des solutions de cet algorithme sont donc viables par rapport au critère choisi de cet article (stabilité statique vérifiée à chaque phase) mais ne seraient pas appliquées sur un chantier de par l'utilisation « inutile » de matériels de type échafaudages ou nacelles élévatrices qui devraient être mobilisés pour rendre ces solutions réalisables en sécurité.

Il nous semble que peu d'articles présentent ce type d'approche d'optimisation appliquée à des structures en béton armé. La subtilité des positions des reprises de bétonnage pour le découpage des murs mais aussi l'impact de l'ordre de la construction des dalles en fonction des murs réalisés ou non sont en effet des sources d'aléas supplémentaires que les structures métalliques n'ont pas. En effet, pour ces dernières, tant que les longueurs des poutrelles et des poteaux à mettre en place restent inférieures à la longueur du gabarit routier il n'est pas nécessaire d'ajouter des assemblages intermédiaires et les assemblages sont uniquement présents aux jonctions poteaux/poutres. Le choix de l'ordre de réalisation des murs et des dalles dans un bâtiment en béton armé est pourtant une étape clé pour assurer la rentabilité d'un chantier. Ce « cyclage » est effectué par un ingénieur et permet ensuite d'optimiser l'utilisation du matériel soit à la main soit via des logiciels de type methoCAD[1]. Ce dernier, bien que très performant dans l'optimisation de l'utilisation du matériel et largement utilisé dans le monde ne propose pas de réalisation automatique de la séquence de construction. Si le cyclage est encore réalisé sans l'aide d'algorithmes c'est notamment par la difficulté de récupérer toutes les données qui vont faire que la réalisation d'un mur sera plus ou moins rapide : taux de ferraillage, présence et nombre de mannequins (réservations pour les fenêtres et les portes), ouvertures permettant de faire circuler les ouvriers, etc.

L'arrivée du BIM, cette méthode collaborative de travail autour d'une maquette numérique, peut bouleverser cette approche grâce aux nombreuses données présentes dans la maquette numérique qui sert de support à l'étude d'un bâtiment.

2 Démarche d'optimisation du cyclage à partir d'une maquette numérique

Le concept présenté dans ce travail est résumé sur la Figure 1. Le travail principal et l'intérêt de cet article réside dans la tâche "2" consistant à trouver une solution optimum de cyclage à partir de temps unitaires fournis par l'utilisateur et d'une structure lue sur un fichier IFC (tâche "1") puis d'écrire cette solution sous forme de phases dans le fichier originel (tâche "3").

Dans ce travail, les auteurs ont opté pour une approche neutre vis-à-vis des éditeurs de logiciels et plutôt que de développer un plug-in intégré dans un logiciel de modélisation comme Revit ou Tekla par exemple, le travail est externalisé via un script python qui travaille sur un

[1] https://www.methocad.com/

fichier au format IFC (BuildingSMART 2013), le format libre d'échange pour les maquettes numériques.

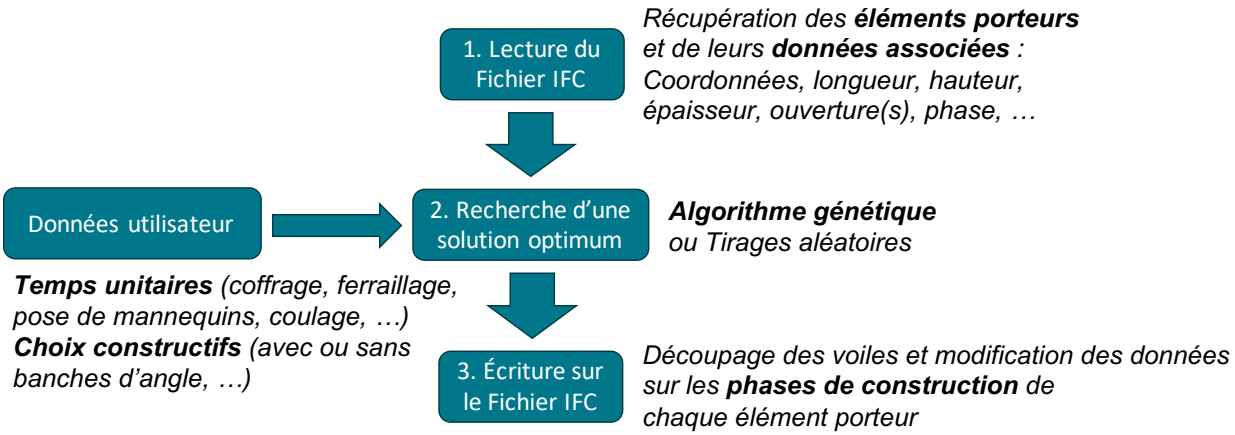


Figure 1 - Concept du travail présenté dans cet article

Les trois phases du script sont décrites ci-dessous.

2.1 Lecture du fichier IFC

Cette première phase consiste à récupérer les éléments porteurs d'une maquette numérique au fichier IFC pour chaque étage avec leurs données associées :

- identifiant de l'élément porteur ;
- coordonnées de l'élément porteur ;
- données géométriques : longueur, hauteur et épaisseur pour les murs ; longueur, largeur, épaisseur pour les dalles ;
- ouvertures dans les éléments porteurs (portes, fenêtres, trémies, etc.) ;
- phase de construction.

2.2 Recherche d'une solution de cyclage optimum

Les données d'entrées pour cette optimisation sont les suivantes :

- données extraites du fichier IFC ;
- temps unitaires pour le coffrage, le ferraillage, la pose des réservations (mannequins) pour les futures portes et fenêtres, le coulage des éléments structurels mais aussi leur décoffrage ;
- choix constructifs (utilisation ou non de banches d'angles, longueur maximum à couler en une seule fois, etc.).

Dans le travail proposé, des résultats obtenus par deux approches sont présentées :

- des tirages aléatoires ;
- un algorithme génétique.

2.3 Écriture du fichier IFC

Cette dernière étape consiste à créer le nombre de phases obtenues dans le cyclage optimum. Le script effectue ensuite un découpage des murs et des dalles suivant le phasage retenu puis modifie la phase associée à chaque élément porteur.

3 Description de l'algorithme génétique utilisé

L'objectif est d'obtenir une solution optimale à l'aide d'un algorithme génétique. À titre de comparaison, les temps de calcul seront comparés à un simple tirage aléatoire.

3.1 Présentation succincte des algorithmes génétiques

Depuis une trentaine d'années de nombreux problèmes d'optimisation autrefois traités à la main ont pu être résolus grâce à l'arrivée des algorithmes génétiques. Ces méthodes qui s'inspirent de la théorie de l'évolution de Darwin, tirent profit de la puissance de calcul des machines actuelles pour explorer l'espace de solutions et concurrencer dans une certaine mesure l'intelligence humaine (Riccardo P. et al. 2008a).

Pour les personnes non initiées au principe d'algorithme génétique, voici une courte introduction au domaine. L'algorithme génétique est une méthode de résolution automatique d'un problème qui n'est basée sur aucune loi physique. Nous allons représenter notre problème avec une certaine représentation (ex : une liste de trois nombres entiers). On appelle individu une possibilité de la représentation (ex : [1,4,2] ou [4,5,7]). Puis nous allons créer une population initiale, composée d'individus qui ont été initialisés aléatoirement. Pour différencier les bons individus des moins bons nous allons créer un critère capable d'évaluer objectivement un individu (ex : écart-type de la liste) et choisir un objectif (ex : obtenir le plus petit écart-type possible). Ainsi, nous sommes maintenant capables de juger les individus. Nous allons créer une nouvelle génération d'individus en **sélectionnant** les meilleurs individus de la génération précédente au sens de notre critère puis appliquer des **croisements** des meilleurs individus (ex : [1,1,2] croisé avec [2,2,1] peut par exemple donner [1,1,1] ou [2,2,2]) et en effectuant des **mutations** (ex : [1,1,2] devient [1,1,3]). Il s'agit en réalité ici d'imiter le processus de sélection naturelle qui a lieu chez les êtres vivants : les individus les plus adaptés à leur environnement sont conservés (sélection) puis se reproduisent (croisement et mutation). En répétant plusieurs fois ces opérations, la population évolue au fur et à mesure des générations vers des individus avec de meilleurs scores (Riccardo P. et al. 2008b).

3.2 Pourquoi choisir un algorithme génétique ?

Les algorithmes génétiques sont performants par rapport à d'autres méthodes d'optimisation lorsqu'il est impossible mathématiquement de fournir une solution analytique, lorsque la taille et la forme de la solution sont des inconnues ou lorsqu'il existe de nombreux simulateurs capables d'évaluer la solution mais pas de bons optimisateurs (Riccardo P. 2008c). Le cyclage de la réalisation de murs et de planchers présente tous ces aspects. Premièrement, même si une modélisation mathématique pour un cas de chantier serait éventuellement possible, la difficulté réside dans le fait de trouver une façon d'automatiser la modélisation pour n'importe quel chantier à partir des plans. Les chantiers sont tellement différents les uns des autres que ce serait une mission bien trop complexe, de plus le risque de ne pas être adaptable à n'importe quel chantier est trop élevé. Secondement, le point le plus important est que ni la taille ni la forme

de la solution ne sont connues lorsque nous attaquons le problème. En effet, le nombre de reprises de bétonnage pour chaque mur, le nombre de réservations pour des ouvertures (portes ou fenêtres) par phase de construction, le nombre de banches et leurs longueurs sont des exemples d'inconnues qu'il faut trouver pour optimiser notre solution selon différents critères. Il est donc impossible de prévoir à l'avance la forme de la solution ou alors on risquerait de se priver de certaines en apportant un biais *a priori*. Enfin, l'expérience acquise par les différents bureaux d'études permettrait de définir des critères de sélection robustes et viables.

L'algorithme génétique apporte une souplesse d'utilisation et de modélisation. Nous n'avons pas à nous soucier de la forme ni de la taille de la solution, ce sont les critères mis en place qui vont faire converger la solution vers une certaine forme. L'algorithme génétique nous permet de manipuler des structures de solution complexe. Dans notre cas nous avons opté pour une modélisation en programmation objet : dans notre code il existe des classes Mur, Dalle, Étage qui permettent de fournir les informations pour noter la solution (ex : quels murs sont construits sur cette dalle ? Quelle est la longueur de mur coulée pour cette phase ? etc.). Ainsi, les données IFC sont utilisées pour fournir les informations nécessaires à la représentation du problème.

3.3 Représentation utilisée

La représentation utilisée est une liste contenant des nombres entiers faisant référence à la phase à laquelle l'objet à la i^e position de la liste est coulée. La liste solution n'est donc rien sans sa liste support qui contient les objets. En revanche toutes les opérations durant l'algorithme génétique sont faites sur la liste solution, la liste support sert surtout pour la notation de la solution. Voici un exemple pour mieux comprendre.

Nous utilisons comme support à ce travail un bâtiment d'habitation "académique" répétitif dont un étage courant est présenté sur la Figure 2. Cette représentation en deux dimensions de l'étage considéré est obtenue par le script après lecture du fichier IFC et un découpage des murs du fichier IFC suivant une longueur maximale choisie par l'utilisateur.

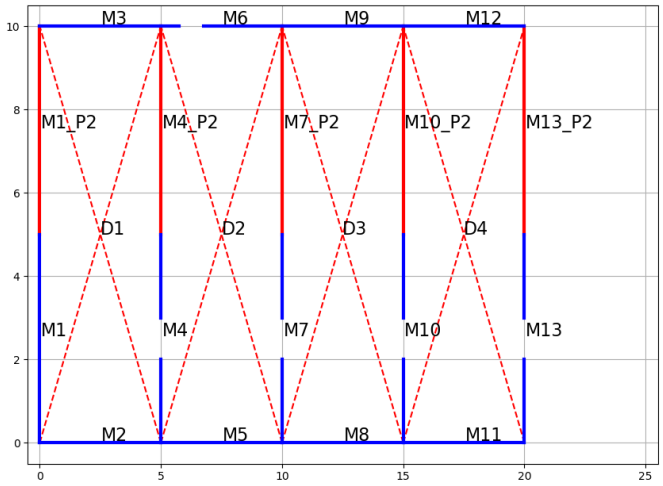


Figure 2 - Plan d'ensemble d'un étage représenté avec un interprétateur python - coordonnées en mètres

Ainsi la liste support contient 22 objets (4 dalles et 18 murs) :
`[Slab_(D1), …, Slab_(D4), Wall_(M1),…, Wall_(M13),Wall_(M1_P2),…,Wall_(M13_P2)]`

La liste solution est alors par exemple :
[2, 1, 4, 3, 3, 4, 3, 2, 1, 3, 3, 4, 4, 4, 2, 2, 3, 2, 2, 1, 1, 2]

Chaque nombre entier représente la phase où le i^e élément de la liste support est réalisé. La représentation graphique phase par phase de cet individu est illustrée sur la

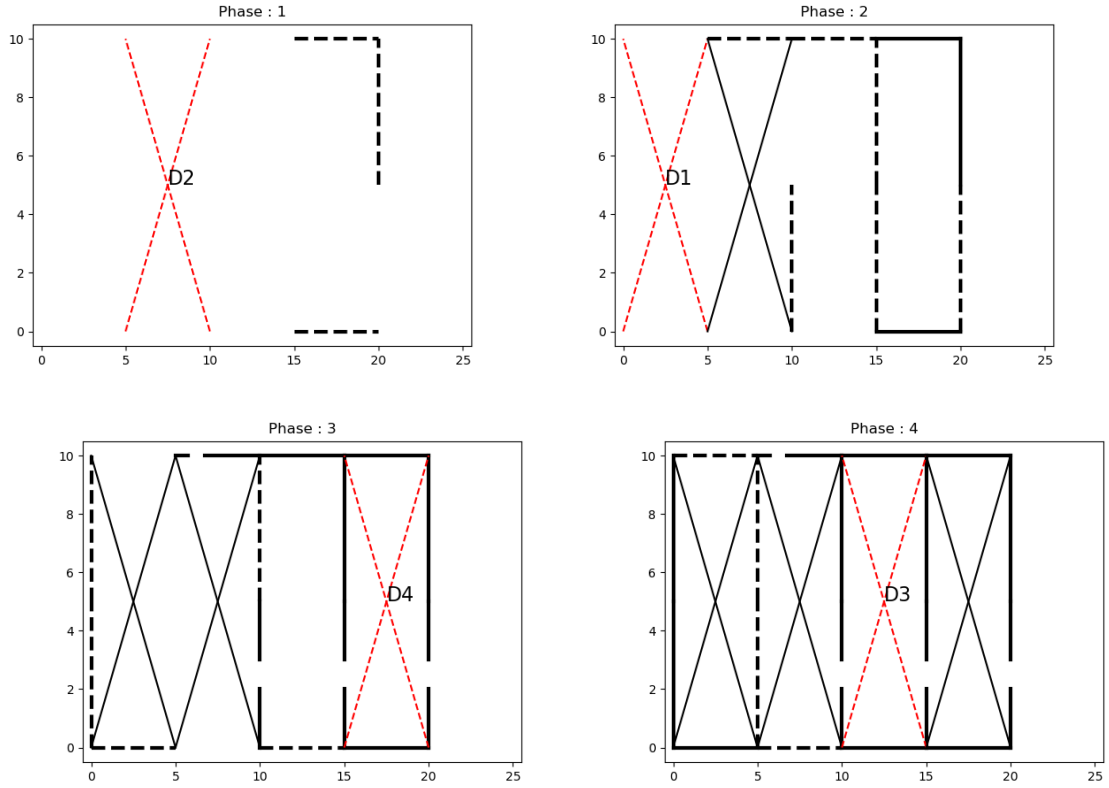
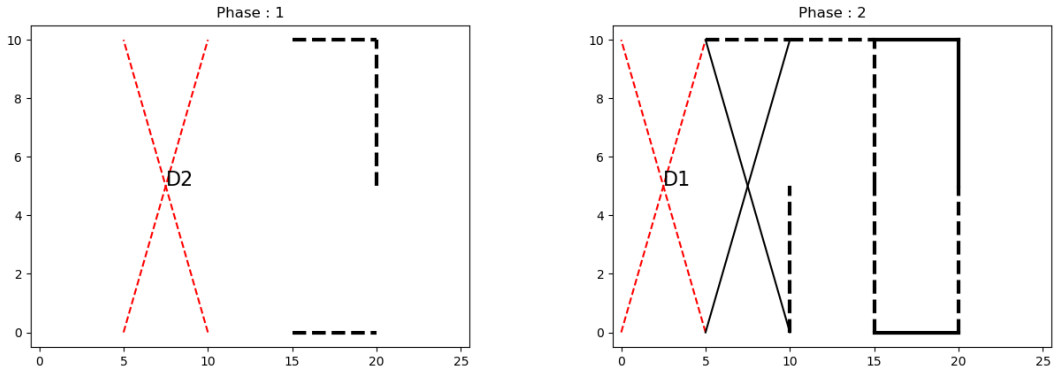


Figure 3.



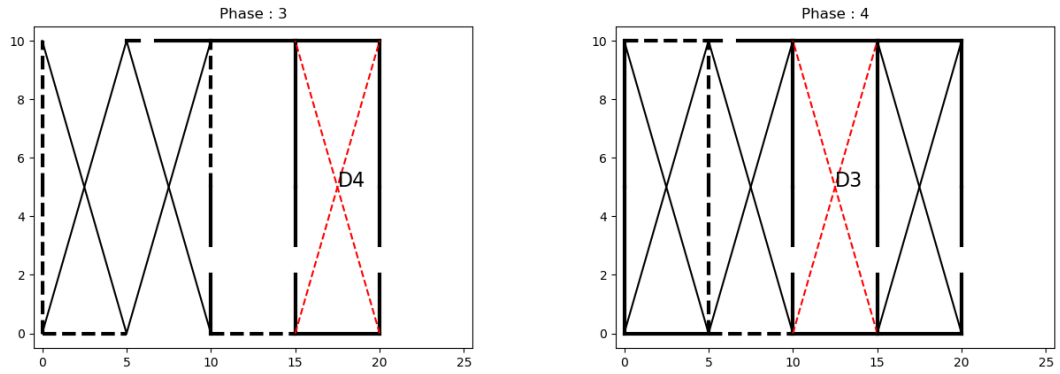


Figure 3 - Exemple de représentation d'une réalisation d'initialisation du vecteur cyclage obtenu par tirage aléatoire et application de la matrice de stabilité (cf Figure 5)

3.4 Algorithme génétique utilisé

L'algorithme génétique utilisé est décrit sur le schéma de la Figure 4. La structure est la plus simple possible : une population est initialisée, les meilleurs individus sont retenus. Avec ces meilleurs individus on réalise des croisements pour créer de nouveaux individus. Tous les individus de la nouvelle génération peuvent subir ensuite des mutations. On répète toutes ces étapes sur un nombre fini de générations – généralement entre 30 et 50 – pour trouver une solution pertinente.

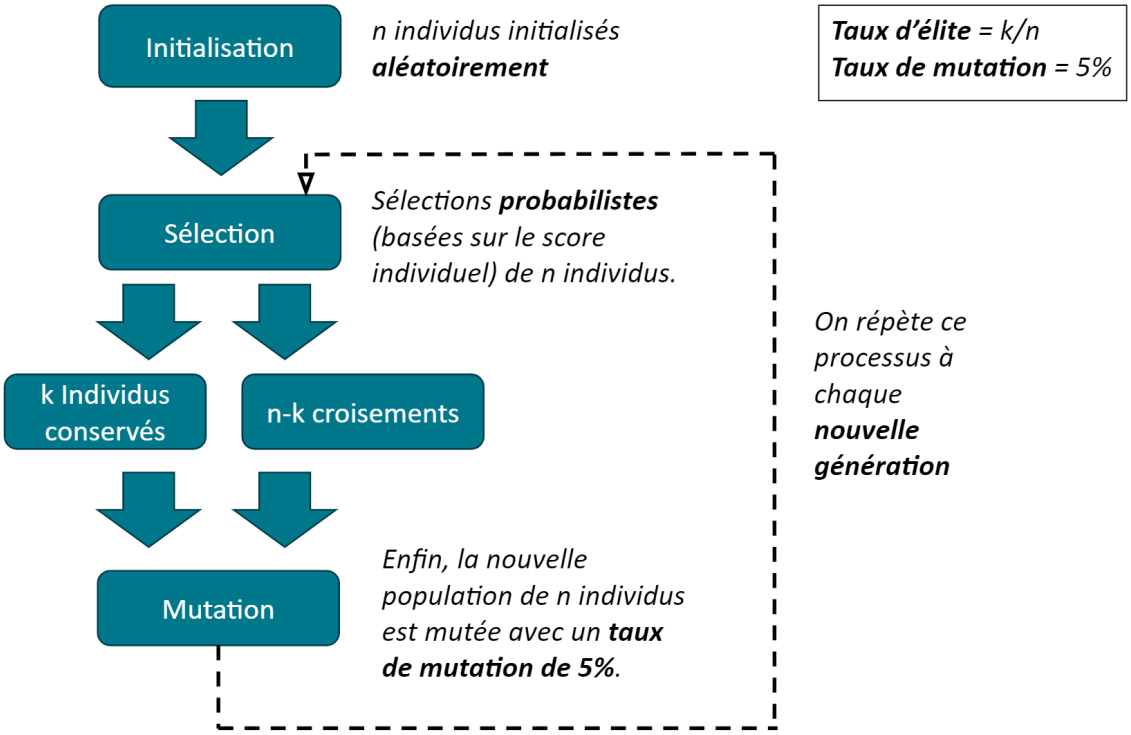


Figure 4 - Schéma de l'algorithme génétique utilisé

3.4.1 Initialisation

L'initialisation est particulièrement importante dans les algorithmes génétiques. En effet, l'espace de recherche dépend du lien étroit entre la population initiale et les opérateurs qui vont agir dessus. Trois types d'initialisations ont été envisagés pour cette étude.

1. Pour la plupart des algorithmes génétiques **la population est initialisée aléatoirement**. Cela permet de ne pas introduire de biais et théoriquement de ne se priver d'aucune possibilité. Dans notre cas, cela voudrait dire que l'on prend un objet (mur ou dalle) en lui attribuant un numéro de phase au hasard. En procédant ainsi, des solutions impossibles à réaliser physiquement seraient générées : par exemple une dalle pourrait être construite à une phase pour laquelle tous les murs qui soutiendront cette dalle ne seraient pas encore coulés. Connaissant la taille de l'espace des solutions à parcourir, il n'est pas intéressant de faire évoluer une population dont la majorité des individus n'est pas viable. Même si le critère mis en place note la constructibilité (faisabilité physique) de la solution, l'algorithme risque de ne pas converger et la solution se dégraderait au fur et à mesure des générations.
2. La deuxième consiste à initialiser en équilibrant les temps des différentes phases. Concrètement, des phases sont attribuées aléatoirement aux murs sauf si une phase a déjà trop de temps "utilisé" auquel cas une autre phase lui est attribuée. Là encore, trop d'individus non viables seraient générés.
3. La troisième approche est celle qui a été retenue dans ce travail. Elle consiste à construire à partir des données d'entrée de la maquette numérique IFC une matrice de stabilité (cf Figure 5) contenant les liens entre les différents objets de la structure. La « Matrix of Constructability Constraints (MoCC) » présentée dans les travaux de Faghihi sur des constructions métalliques (Faghihi et al. 2014) est composée de 0 et de 1. Pour le travail présenté dans cet article, chaque coefficient $a_{i,j}$ de la matrice de stabilité est un entier (0 ; 1 ; 3 ou -3). Le coefficient $a_{i,j}$ permet de savoir si les objets i et j sont côte à côte ou s'ils doivent être coulés dans des phases différentes. Le processus d'initialisation est le suivant : on attribue aléatoirement à chaque dalle une phase puis en se servant de la matrice de stabilité on attribue des phases aux murs en conséquence. Comme nous nous intéressons à un problème d'étage courant d'un bâtiment répétitif, la matrice permet de prendre en compte deux cas de construction de dalles : le cas où les murs sont construits puis la dalle est coulée (dit "cas dalle supérieure"), ou le cas où la dalle est construite puis les murs dessus (dit "cas dalle inférieure").

$$\begin{array}{c} \\ objet_1 \\ objet_2 \\ \vdots \\ objet_n \end{array} \begin{array}{c} \begin{array}{cccc} objet_1 & objet_2 & \cdots & objet_n \end{array} \\ \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n,1} & a_{n,2} & \cdots & a_{n,n} \end{pmatrix} \end{array}$$

$$a(i,j) = \begin{cases} \text{Phase(objet i) si } i = j \\ \text{1 si } objet_i \text{ et } objet_j \text{ sont côte à côte} \\ \text{3 si } objet_i \text{ est une dalle inférieure par rapport au mur } objet_j \\ \text{-3 si } objet_i \text{ est une dalle supérieure par rapport au mur } objet_j \end{cases}$$

Figure 5 – Matrice de stabilité utilisée dans ce travail inspirée de la « Matrix of Constructability Constraints (MoCC) » présentée dans (Faghihi et al. 2014)

Ce genre de méthode est couramment utilisée lorsque la solution d'un problème est contrainte de manière élevée (Riccardo P. 2008d). Bien sûr, nous avons introduit un biais et l'espace de recherche a été réduit mais les solutions non constructibles physiquement ne nous intéressent pas (Riccardo P. 2008e). Le seul bémol à cette approche étant que la présence de solutions "non constructibles" amène une certaine richesse qui peut être utile pour converger vers la meilleure solution. De telles solutions peuvent être introduites au fur et à mesure des générations grâce aux opérateurs "croisement" et "mutation" qui sont présentés dans la partie suivante.

3.4.2 Opérateurs

Pour modifier les individus à chaque génération, plusieurs opérateurs sont nécessaires. Les opérateurs les plus courants en algorithmie génétique sont : la **sé*lection***, opérateur choisissant des individus de la génération courante pour la nouvelle génération (il s'agit d'imiter la sélection naturelle de la théorie de l'évolution) ; le ***croisement,*** opérateur réalisant la combinaison du matériel génétique de deux individus pour former un ou plusieurs descendants (il s'agit d'imiter la reproduction d'individus) et enfin la ***mutation,*** opérateur pouvant changer le matériel génétique d'un individu. Toutes ces méthodes sont utiles pour créer de nouveaux individus variés au fur et à mesure des générations (Riccardo P. 2008f).

Sélection **:** l'opérateur utilisé attribue une probabilité de sélection à un individu en fonction de son score. Les meilleurs individus au sens de ce critère sont alors sélectionnés pour être les parents de la prochaine génération. Le nombre d'individus sélectionnés pour la nouvelle génération est un paramètre de l'algorithme génétique.

Croisement **:** l'opérateur utilisé combine le matériel de deux individus pour en créer au moins un nouveau (Koza et al. 1996). Concrètement, une partie de longueur aléatoire et de coordonnée aléatoire du matériel génétique du Parent 1 est remplacée par celle du Parent 2. La Figure 6 illustre ce concept dans le cas où un seul enfant est généré à partir de deux parents.

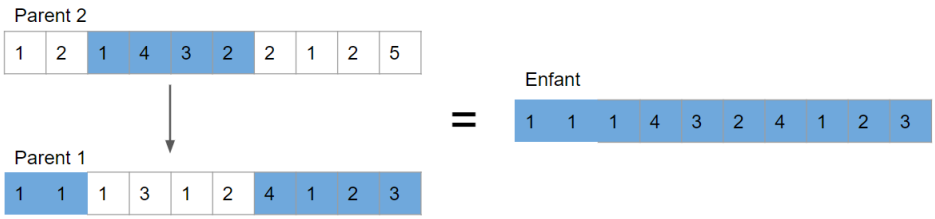


Figure 6 - Illustration de l'opérateur "croisement" dans un cas où un seul enfant est généré à partir de deux parents

Seulement si l'on procède ainsi, nous perdons facilement le critère de constructibilité. L'idée qui a été retenue pour pallier ce problème est dans un premier temps de chercher des individus qui pourraient se croiser : c'est la recherche du point de croisement (Riccardo P. 2008g). Dans cette recherche il s'agit de choisir un individu et de regarder si d'autres individus ont le même numéro de phase pour une ou plusieurs dalles. Une fois cette recherche faite, nous croisons les deux individus d'une certaine façon : nous regardons les murs construits sur la dalle qui partage le même numéro de phase pour les deux individus ; puis nous échangeons une partie (choisie aléatoirement) des numéros de phases de ces murs pour créer un nouvel individu. La Figure 7 illustre ce concept.

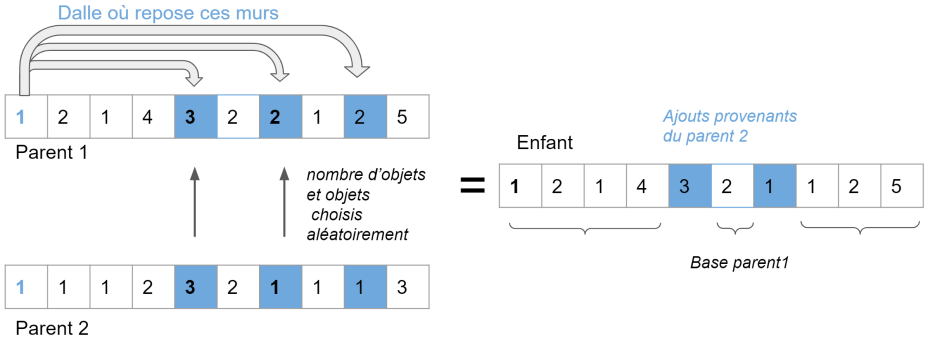


Figure 7 - Illustration du croisement avec recherche du point de croisement (toujours dans le cas où un seul enfant est généré à partir de deux parents)

Ce croisement a moins de chance de produire des individus non viables physiquement, mais introduit encore une fois un biais car l'opérateur croisement est contraint (Riccardo P. 2008d). Durant la construction de notre algorithme génétique nous avons pu constater que cet opérateur réduit les oscillations du score du meilleur individu durant les générations et stabilise l'écart type des scores des individus d'une génération autour d'une valeur de 0.2, le score étant une note sur 1.

Mutation (swap) : l'opérateur utilisé parcourt la liste de phases dans l'ordre et décide de changer la phase d'un objet avec une probabilité de 5%. Koza, scientifique de renommée en algorithmie génétique, suggère d'utiliser une probabilité de mutation faible (Riccardo P. 2008h, Koza et al. 1996). La Figure 8 illustre ce principe.

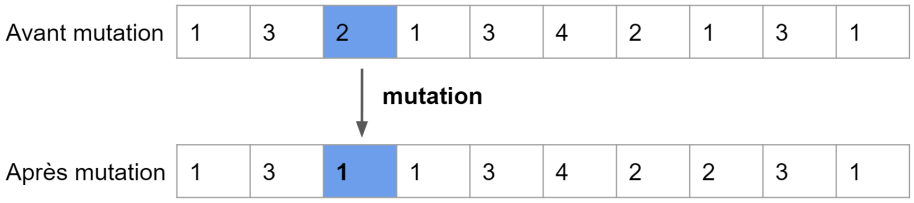


Figure 8 - illustration du principe de mutation

4 Définition de la métrique : critères pour l'évaluation des solutions

Dans les deux cas, chaque solution est évaluée en se basant sur une métrique **M** basée sur des critères objectifs.

4.1 Combinaison et description des critères

Les critères peuvent être combinés selon la formule (1). Dans ce cas, la métrique M_s est une moyenne pondérée variant entre 0 et 1. Plus M tend vers 1, plus la solution sera évaluée comme étant optimale :

$$M_s = \frac{\gamma_1 C_{liss\_mur} + \gamma_2 C_{liss\_dalle} + \gamma_3 C_{liss\_mannequin} + \gamma_4 C_{stab} + \gamma_5 C_{dist}}{\sum_{i=1}^{n} \gamma_i} \quad \dots (1)$$

Une autre possibilité est de définir une métrique M_p basée sur un produit :

$$M_p = {C_{liss\_mur}}^{1/\gamma_1} \times {C_{liss\_dalle}}^{1/\gamma_2} \times {C_{liss\_mannequin}}^{1/\gamma_3} \times {C_{stab}}^{1/\gamma_4} \times {C_{dist}}^{1/\gamma_5} \quad (2)$$

Les coefficients γ_i permettent de pondérer l'importance relative de chaque critère choisi. Cette métrique permet d'avoir une valeur de 0 dès qu'un des critères est nul. Néanmoins, cette métrique tend à donner des comportements plus chaotiques à l'évolution des individus générés par l'algorithme génétique présenté dans ce travail. Seuls les résultats obtenus avec la métrique M_s seront présentés.

Les critères des métriques décrites dans les formules (1) & (2) sont les suivants :

- C_{liss_mur} : le temps de réalisation des murs pour chaque phase doit être lissé.
  - Le coefficient de variation (rapport entre l'écart-type σ_{temps} et la moyenne μ_{temps}) des temps passés pour la réalisation des murs de chacune des phases devant être le plus faible possible, la formule utilisée est la suivante :
  - $C_{liss\_mur} = \left[1 - \frac{\sigma_{temps}}{\mu_{temps}}\right]_+$
- C_{liss_dalle} : le temps de réalisation des dalles pour chaque phase doit être lissé.
  - Le coefficient de variation (rapport entre l'écart-type σ_{temps} et la moyenne μ_{temps}) des temps passés pour la réalisation des dalles sur chacune des phases devant être le plus faible possible, la formule utilisée est la suivante :
  - $C_{liss\_dalle} = \left[1 - \frac{\sigma_{temps}}{\mu_{temps}}\right]_+$
- $C_{liss\_mannequin}$: le nombre de mannequins utilisé pour chaque phase doit être lissé.

- Le coefficient de variation (rapport entre l'écart-type σ_{temps} et la moyenne μ_{temps}) des temps passés sur chacune des phases devant être le plus faible possible, la formule utilisée est la suivante :
- $C_{liss\_mannequin} = \left[1 - \frac{\sigma_{nb\_mannequin}}{\mu_{nb\_mannequin}}\right]_+$

- C_{stab} : la constructibilité et donc la stabilité des éléments construits à chaque phase est essentielle. Afin d'avoir un critère évolutif entre 0 et 1 permettant d'améliorer la convergence de l'algorithme, la formule utilisée est le rapport entre le nombre d'objets physiquement réalisables et le nombre d'objets total.
- C_{dist} : la distance totale parcourue par les ouvriers pour atteindre les deux faces de chaque mur à couler pour chacune des phases doit être la plus faible possible.
  - Cette distance est évaluée par un algorithme fast marching (Sethian 1996) présentée dans le paragraphe 4.2.
  - Le coefficient de variation (rapport entre l'écart-type σ_{dist} et la moyenne μ_{dist}) des distances parcourues par les ouvriers pour la réalisation du (des) mur(s) sur chacune des phases devant être le plus faible possible, la formule utilisée est la suivante :
  - $C_{dist} = \left[1 - \frac{\sigma_{dist}}{\mu_{dist}}\right]_+$
  - En cas de zone inaccessible, C_{dist} est fixé à zéro.

4.2 Distances parcourues par les ouvriers par la méthode Fast Marching C_{dist}

Une des originalités de ce travail est de proposer une méthode efficace pour évaluer la distance à parcourir par les ouvriers pour chaque phase du travail.

Pour illustrer les résultats obtenus par cette méthode, prenons une salle de 4,0 m sur 3,5 m avec deux portes et un point d'accès situé en (X=-0,5 ;Y=0,0) (cf Figure 9 à gauche).

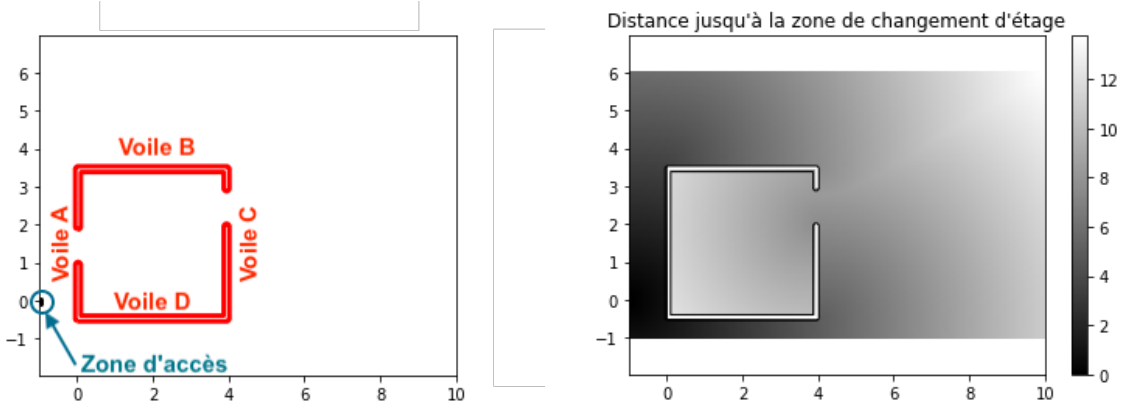


Figure 9 - Géométrie de la salle servant d'illustration à l'algorithme de Fast Marching (à gauche) et résultat de la méthode appliquée sur une salle ouverte avec la porte opposée à la zone d'accès (à droite) – distances en mètres

Selon les solutions obtenues, 3 propositions de phases parmi les 16 possibles $\left(\binom{4}{0} + \binom{4}{1} + \binom{4}{2} + \binom{4}{3} + \binom{4}{4}\right)$ peuvent présenter un intérêt particulier pour illustrer l'intérêt de calculer cette distance parcourue par les ouvriers pour fermer la banche du mur D :

- coulage des murs A et D (cf Figure 9 à droite) pendant la phase étudiée ;

- coulage uniquement du mur D (cf Figure 10 à gauche) pendant la phase étudiée ;
- construction des murs A, C et D (cf Figure 10 à droite) pendant la phase étudiée.

La Figure 9 (à droite) présente ainsi une solution où les murs A et D sont réalisés durant la phase considérée et les autres murs (B et C) ont été réalisés durant la ou les phases précédentes. Dans cet exemple, le point de coordonnées (2,0 ;0,0) à l'intérieur de cette salle contre le mur D est ainsi à une distance de marche d'environ 10,5 m du point d'accès à l'étage étudié de coordonnées (-0,5 ;0).

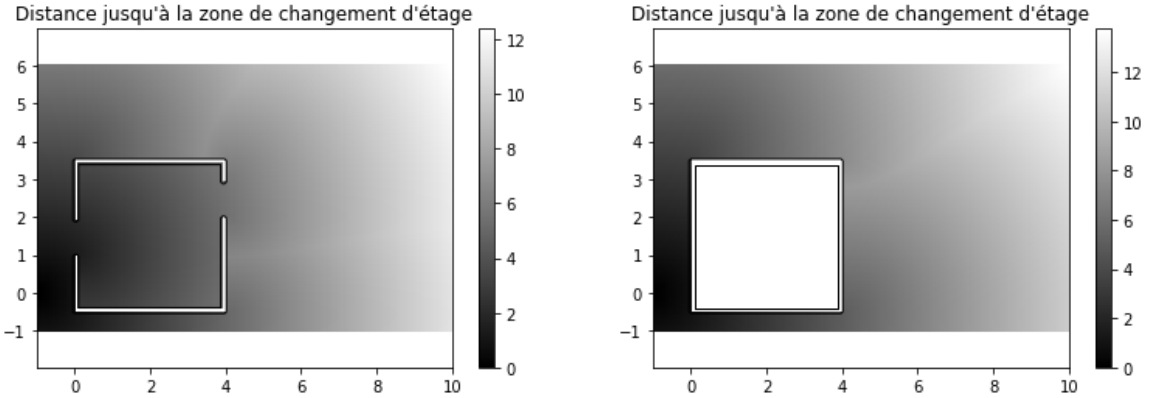


Figure 10 - Résultat de Fast Marching sur une salle ouverte avec une porte proche de la zone d'accès (à gauche) et sur une salle fermée (à droite) – distances en mètres

La Figure 10 (à gauche) présente une solution où le mur D est réalisé durant la phase considérée et les autres murs (A, B et C) ont été réalisés durant la ou les phases précédentes. Dans cet exemple, le point de coordonnées (2,0 ;0,0) à l'intérieur de cette salle contre le mur D est à une distance de marche d'environ 3,6 m du point d'accès à l'étage étudié.

La Figure 10 (à droite) présente ainsi une solution où les murs A, C et D sont réalisés durant la phase considérée et le mur B a été réalisé durant une phase précédente. Dans cet exemple, le point de coordonnées (2,0 ;0,0) à l'intérieur de cette salle contre le mur D n'est pas accessible. Concrètement, l'algorithme donne une valeur de 0,0. C'est un motif pertinent pour ne pas choisir une solution où le(s) dernier(s) mur(s) d'une salle à réaliser durant un phasage serai(en)t celui(ceux) contenant la(les) porte(s) puisque le coffrage de ce(s) mur(s) empêchera d'aller serrer les tiges de serrages.

Ces exemples simples illustrent l'aspect discriminant du calcul de la distance à parcourir par les ouvriers pour la réalisation de chaque mur d'une phase :

- soit un côté du mur à réaliser n'est pas accessible (sauf à utiliser des échelles pour franchir les murs déjà construits) et il faut tout simplement pénaliser fortement la note de cette solution ;
- soit une solution permet d'avoir l'ouverture de porte du côté de l'accès réalisé au plus tôt permettant aux ouvriers de circuler plus librement sur le chantier et cette solution est à privilégier.

Notons que bien que très efficace, le temps utilisé par cette méthode n'est pas négligeable puisqu'elle utilise environ $5{,}2.10^{-1}$ seconde par calcul de solution contre $7{,}6.10^{-4}$ seconde sans. Si ce temps de 0,5 seconde peut paraître faible, comme des milliers de tirages sont effectués, le rapport de temps entre un calcul avec et sans calcul de la distance étant d'environ 680, l'impact est donc rapidement conséquent. Ainsi, les distances par phase sont uniquement calculées

lorsque les autres critères donnent des résultats jugés suffisamment bons, à savoir, une note moyenne supérieure à 0,8.

4.3 Autres critères envisageables

De nombreux autres critères sont envisageables et pourraient être directement intégré dans l'initialisation d'une solution :

- vérifier si un joint de dilatation et présent, les murs de part et d'autre de ce joint devant être réalisés durant deux phases successives ;
- la possibilité de choisir de ne pas avoir de banches d'angle et donc traiter les angles avec des reprises de bétonnage ce qui engendre des conséquences sur le phasage ;
- la présence de cotes bloquées est une situation à éviter, si elle a lieu, il faudrait vérifier que les clés sont réutilisées régulièrement mais de préférence les éviter.

5 Présentation des résultats

Les résultats sont présentés sur une structure volontairement simple (cf Figure 2) afin de pouvoir bien identifier les avantages et les inconvénients des méthodes employées.

Cette structure possède 22 éléments, ce qui fait avec 4 phases possibles pour chacun de ces éléments, 4^{22} possibilités de solutions, soit environ $1{,}76.10^{13}$ solutions.

5.1 Recherche d'un optimum par tirages aléatoires successifs

Même en faisant 100 000 de tirages aléatoires successifs et en imaginant qu'il n'y ait que dix très bonnes solutions, ce calcul n'aura qu'une chance sur 17.10^6 de trouver une de ces dix très bonnes solutions : c'est à dire la même chance que de gagner au loto au rang 1 (une chance sur 19.10^6 environ).

Avec 10 000 tirages aléatoires, il est par exemple possible d'obtenir un meilleur score de 0,58 avec une métrique définie ainsi :

$$M_s = \frac{5\,C_{liss\_mur} + 5\,C_{liss\_dalle} + 1C_{liss\_mannequin} + 10\,C_{stab} + 3\,C_{dist}}{24}.$$

La Figure 11 présente ainsi l'histogramme des scores obtenus. Le tirage ayant obtenu un score de 0,58 ne se voit même pas et on voit que les solutions ont une moyenne centrée autour de 0,15. La structure en béton armé académique proposée dans ce travail étant très simple (cf Figure 2), il est évident qu'il serait très chanceux de trouver ne serait-ce qu'une solution viable sur une structure plus réaliste.

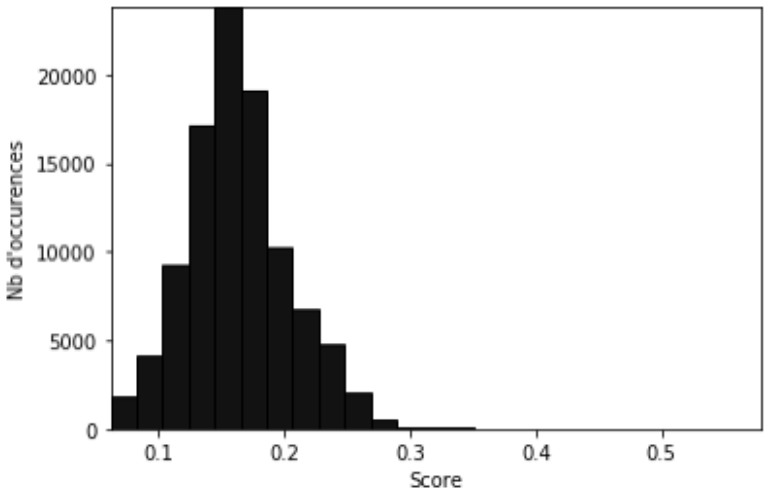


Figure 11 - Exemple d'histogramme de scores obtenus avec des tirages aléatoires

5.2 Recherche d'un optimum par un algorithme génétique avec une métrique basée sur une somme pondérée (formule 1)

Les algorithmes génétiques évoluent très rapidement en fonction des opérateurs, de la méthode d'initialisation, des paramètres de sélection des élites et de mutation mais aussi des critères d'évaluation des individus. Chaque changement peut avoir de grandes conséquences sur la convergence de l'algorithme. Les résultats présentés sont encourageant par rapport aux méthodes habituellement utilisées et offrent même de belles perspectives.

Avec l'algorithme présenté, les meilleurs résultats ont été obtenus avec une population de 250 individus, 50 générations, un taux de mutation de 5% et 5% des élites conservés. Les courbes présentées sur la Figure 12 montrent l'évolution du meilleur score au fil des générations ainsi que l'écart-type des scores par génération sur un exemple de calcul ayant pris 3567 secondes soit environ une heure (calculs réalisés sur un ordinateur avec un processeur i7 quatre cœurs cadencé à 2,2 GHz avec 16 Go de RAM).

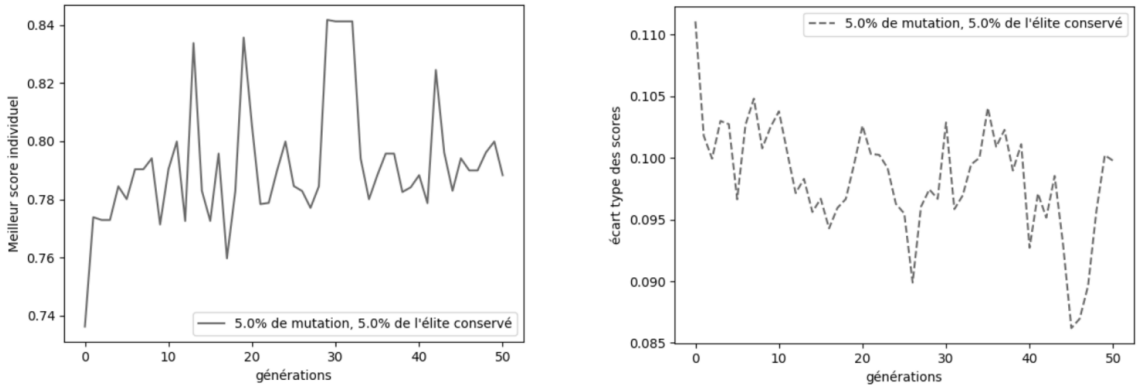


Figure 12 - Évolution des meilleurs scores et des écarts-types en fonction des générations

La convergence de l'algorithme n'est pas encore optimale : la tendance est à une augmentation du meilleur score individuel au fur à mesure des générations mais des fluctuations importantes persistent. De même l'écart-type des scores au fil des générations a tendance à diminuer et en même temps le score monte ce qui témoigne d'un resserrage des scores autour

d'une moyenne plus élevée. Il est important de noter que l'échelle des meilleurs scores varie entre 0,74 et 0,84 avec uniquement des solutions constructibles. Toutes les solutions obtenues sont donc pertinentes du point de vue du constructeur.

Les résultats sont donc encourageants puisque des scores supérieurs à 0,9 ont régulièrement été obtenus et que le programme est relativement rapide. Le temps de calcul pour cet exemple est d'environ une heure sans optimisation du temps de calcul par l'algorithme de Fast Marching qui est la phase la plus chronophage : le rapport entre le temps de calcul avec et sans Fast Marching est d'environ 500.

De futurs travaux ne manqueront pas d'améliorer la convergence et de réduire le temps de calcul de ce premier travail.

6 Conclusion

Cet article bien que basé sur un exemple de projet que l'on qualifiera d'académique (cf Figure 2) permet de montrer la faisabilité de réaliser un algorithme génétique basé sur des données associées à une maquette numérique permettant de générer des cyclages réalistes.

Les conclusions du travail présenté dans cet article sont les suivantes :

- le travail présenté a permis de montrer que l'application d'un algorithme de *Fast marching* (Sethian, 1996) permet de calculer phase par phase les distances minimales des trajets des ouvriers pour chacun des murs à réaliser ;
- le champ des solutions des cyclages est très grand puisque l'ordre de grandeur est de 10^{13} pour un bâtiment que l'on nommera "académique" avec seulement 4 salles ;
- parcourir cet espace par des tirages aléatoires pour trouver une séquence réaliste nécessiterait 8.10^{9} s soit 254 ans sur un ordinateur standard de 2020 ;
- un algorithme génétique simple permet d'avoir des solutions viables.

7 Perspectives

Quelques perspectives à ce premier travail :

- Deux pistes d'améliorations sont envisageables pour l'algorithme génétique :
    - La création d'opérateurs contraignants qui contrôleraient la forme de la solution le long des générations à l'aide de la matrice de stabilité. Cette approche a fait ses preuves dans plusieurs cas d'algorithmes (Riccardo et al. 2008d).
    - Une étude plus approfondie sur les paramètres de l'algorithme présenté (taux de mutation, nombre d'individus conservés) à l'aide de fonctions d'optimisation.
- Une fois un phasage entré dans une maquette IFC, il est possible de générer simplement et automatiquement des plans de circulation des ouvriers avec les trajets optimaux calculés grâce à l'algorithme de *Fast marching* et dans un futur qui pourrait être proche, les intégrer à des lunettes de réalité augmentée de leur casque de chantier.

- Grâce au suivi de chantier effectué sur la maquette numérique par l'entreprise de gros œuvre, il sera aisément faisable de lancer l'algorithme d'optimisation de cyclage sur les murs restants d'un étage donné dans le cas où une équipe aurait été plus rapide qu'une autre par exemple.

Après cette approche académique, une confrontation à des structures réellement construites dans une démarche de retour d'expériences serait intéressante. Deux axes de travail différents sont envisageables :

- Une confrontation de la métrique proposée dans ce travail sur des exemples réellement réalisés pourrait permettre d'améliorer cette dernière et notamment la pondération associée à chaque critère. Une étude paramétrique des différents critères intervenant dans la métrique proposée permettrait de vérifier que la solution optimale tend vers une solution ayant donné de bons résultats sur un chantier réel.
- Une approche complètement différente pourrait être adoptée en faisant une analyse de type machine learning sur des images de cyclages ayant été pré-sélectionnées par les entreprises comme ayant permis d'avoir des rendements journaliers élevés afin de pouvoir générer des formes de structures permettant une réalisation la plus efficiente possible.

Références

BuildingSMART. (2013). buildingSMART, International home of openBIM. Retrieved June 22, 2013, from http://www.buildingsmart-tech.org/

Faghihi V., Reinschmidt K. F., Kang J. H. (2015). Construction scheduling using Genetic Algorithm based on Building Information Model, Expert Systems with Applications, Volume 41, Issue 16, pp. 7565-7578, ISSN 0957-4174. https://doi.org/10.1016/j.eswa.2014.05.047.

Koza, J. R et al. (1996). Four problems for which a computer program evolved by genetic programming is competitive with human performance, V.5.5 Control Parameters, p.7 https://doi.org/ 10.1109/ICEC.1996.542327

Murat Tanyer A., Aouad G. (2005) Moving beyond the fourth dimension with an IFC-based single project database, Automation in Construction, Volume 14, Issue 1, pp. 15--32, ISSN 0926-5805, https://doi.org/10.1016/j.autcon.2004.06.002.

Riccardo P., William B. Langdon et al. (2008a). A Field Guide to Genetic Programming, 12.3 Human Competitive Results : The Humies, pp.117-121.

Riccardo P., William B. Langdon et al. (2008b). A Field Guide to Genetic Programming, 1. Introduction, pp.10-13.

Riccardo P., William B. Langdon et al. (2008c). A Field Guide to Genetic Programming, 12.1 Where GP has Done Well, pp.111-113.

Riccardo P., William B. Langdon et al. (2008d). A Field Guide to Genetic Programming, 6.2.1 Enforcing Particular Structures , p.52.

Riccardo P., William B. Langdon et al. (2008e). A Field Guide to Genetic Programming, 6.2.4 Constraints and Bias , pp.55-57.

Riccardo P., William B. Langdon et al. (2008f). A Field Guide to Genetic Programming, 2.4 Recombination and Mutation , pp.15-17.

Riccardo P., William B. Langdon et al. (2008g). A Field Guide to Genetic Programming, 5.3, GP Crossover , pp.44-46.

Riccardo P., William B. Langdon et al. (2008h). A Field Guide to Genetic Programming, 5.2.1 Is Mutation ecessary , p.42.

Riccardo P., William B. Langdon et al. (2008i). A Field Guide to Genetic Programming, 9.2 MO GP via Operator Bias , pp.81-82.

Riccardo P., William B. Langdon et al. (2008j). A Field Guide to Genetic Programming, 3.4 Step4: GP Parameters , pp.26-27.

Sethian J.A. (1996). A Fast Marching Level Set Method for Monotonically Advancing Fronts, Proc. Natl. Acad. Sci., 93, 4, pp.1591—1595. https://doi.org/10.1073/pnas.93.4.1591

Tulke J., et Jochen H. (2007). 4D construction sequence planning–new process and data model. Proceedings of CIB-W78 24th International Conference on Information Technology in Construction, Maribor, Slovenia. Pp 79—84. https://doi.org/10.1.1.116.4602

Cengiz Toklu Y. (2011). Application of genetic algorithms to construction scheduling with or without resource constraints - Canadian Journal of Civil Engineering 29(3):421-429. https://doi.org/10.1139/l02-034