Exploration des types de solutions en conception architecturale à l'aide des algorithmes génétiques

Savoirs des Activités de Projet Instrumentées

Encadré par Anne TÜSCHER, Joaquim SILVESTRE, et François GUENA

Etienne LAGET – Etudiant ENSAPLV – Mémoire 2024

Résumé

À l'ère de l'intelligence artificielle, où les pratiques architecturales évoluent rapidement, cette recherche explore l'utilisation des algorithmes génétiques comme outil d'assistance aux architectes dans la phase initiale de conception. En se concentrant sur le processus de conception paramétrique, cette étude évalue les capacités des algorithmes génétiques mono-objectif (Galapagos) et multi-objectifs (Octopus) à enrichir l'exploration des types de solutions.

La méthodologie repose sur une approche expérimentale, utilisant un modèle paramétrique simplifié d'un parc pour enfants afin de tester les performances de ces algorithmes. L'étude examine leur capacité à générer et évaluer des solutions variées en fonction d'objectifs et de contraintes prédéfinis. Les résultats montrent que, bien que les algorithmes mono-objectif optimisent efficacement un paramètre unique, les algorithmes multi-objectifs offrent un éventail plus large de solutions, favorisant l'exploration des types de solutions.

Cette recherche démontre que les algorithmes multi-objectifs sont plus adaptés que les algorithmes mono-objectif à l'exploration des types de solutions en architecture, rendant le processus de conception plus riche et stimulant.

Mots-clés

Processus de conception, Outils paramétriques, Paramètres, objectifs de performance, Phase de conception, Expériences, Plug In Grasshopper, Solveur Algorithmique, algorithmes évolutionnaire, algorithmes génétiques, Mono-objectif, Multi-objectif.

Abstract

In the era of artificial intelligence, where architectural practices are rapidly evolving, this research explores the use of genetic algorithms as a tool to assist architects in the early stages of design. Focusing on the parametric design process, this study evaluates the capabilities of mono-objective (Galapagos) and multi-objective (Octopus) genetic algorithms to enhance the exploration of solution types.

The methodology is based on an experimental approach, using a simplified parametric model of a children's park to test the performance of these algorithms. The study examines their ability to generate and evaluate diverse solutions based on predefined objectives and constraints. Results show that while mono-objective algorithms efficiently optimize a single parameter, multi-objective algorithms provide a broader range of solutions, fostering the exploration of solution types.

This research demonstrates that multi-objective algorithms are better suited than mono-objective algorithms for exploring solution types in architecture, making the design process richer and more stimulating.

Keywords

Design process, Parametric tools, Parameters, Performance objectives, Design phase, Experiments, Grasshopper plugin, Algorithmic solver, Evolutionary algorithms, Genetic algorithms, Single-objective, Multi-objective.

Remerciements

Je tiens à remercier, tout d'abord, Mamoun LAHLOU, qui a réalisé la même année son mémoire sur l'architecture paramétrique de façades moucharabiées, avec qui j'ai pu échanger sur les sujets complexes liés aux logiciels d'architecture paramétrique.

Je souhaite également exprimer ma gratitude à toutes les personnes qui ont pris le temps de relire ce mémoire et de me faire des critiques constructives. J'ai tenu compte de chacune de leurs remarques pour améliorer ce document. Je pense notamment à Olivia HACQUET, mais aussi à Émilie LANDAIS et bien d'autres, dont le soutien et les conseils m'ont été précieux.

Enfin, je tiens à remercier mes professeurs, sans qui rien de tout cela n'aurait été possible. Je pense bien sûr à Anne TÜSCHER, Joaquim SILVESTRE, et François GUENA. Ils ont su m'accompagner dès le début sur ce sujet assez complexe et me permettre de développer ce qui m'intéresse depuis le départ, à savoir l'aide à la conception en architecture par les algorithmes.

Table des matières

I. Introduction

Contexte

À l'ère de l'intelligence artificielle, le rôle de l'architecte est en pleine évolution. Les outils numériques occupent une place de plus en plus importante dans la réalisation d'un projet d'architecture. Aujourd'hui, ils sont principalement utilisés comme moyen de représentation du projet, mais d'autres formes de conception sont envisageables. L'une d'entre elles s'appelle la conception paramétrique.

La conception paramétrique est une méthode de conception assistée par ordinateur où les dimensions et les relations géométriques d'un objet sont définies par des paramètres variables. Ces paramètres peuvent être ajustés pour modifier la conception de manière flexible et dynamique, sans nécessiter de redessiner manuellement chaque élément.

L'émergence de la conception paramétrique est étroitement liée au développement de logiciels comme Rhinoceros et Revit. Rhinoceros, est un outil de modélisation 3D qui offre une grande flexibilité dans la création de formes complexes. Il est compatible avec le plugin Grasshopper qui permet de développer des algorithmes visuels pour la conception paramétrique. Grâce à lui, les architectes peuvent définir des paramètres et des relations géométriques qui adaptent automatiquement le modèle en fonction des changements apportés aux paramètres d'entrée. D'après une étude de secteur réalisée en 2022 « La profession d'architecte en Europe », la majorité des praticiens de la conception paramétrique travaillent sur le logiciel Grasshopper.

C'est dans ce cadre que des solveurs algorithmiques[1] ont été développés. Parmi eux, il y a les algorithmes génétiques qui permettent d'explorer et optimiser les solutions conceptuelles à partir de la paramétrisation du modèle. Ces derniers sont divisés en deux familles, les algorithmes génétiques mono-objectif qui ne prennent en compte qu'un seul paramètre objectif et les algorithmes génétiques multi-objectifs qui en considèrent plusieurs.

[1] Un solveur algorithmique est un programme informatique conçu pour résoudre automatiquement des problèmes complexes en utilisant des algorithmes, notamment dans les domaines de l'optimisation et de la modélisation mathématique.

Problématique

Ce mémoire s'intéresse à la phase de conception du projet d'architecture, où le travail de l'architecte progresse par divers essais, que nous appellerons « essai-erreur » par corrélation avec le vocabulaire paramétrique. L'architecte, en quête du design idéal, explore un maximum de possibilités conceptuelles (géométries, formes) pour choisir celle qui correspond le mieux aux critères d'évaluation qu'il s'est fixés. Habituellement, ce processus est effectué manuellement, ce qui peut prendre beaucoup de temps. Cela oblige l'architecte à limiter ses recherches et à se contenter de la solution la plus intéressante trouvée, sans savoir s'il existe une meilleure alternative. Si l'architecte pouvait commencer ce travail avec un éventail de types de solutions à sa disposition, cela pourrait stimuler sa créativité et accélérer le processus de conception.

Comment utiliser les algorithmes génétiques dans le processus de conception architecturale pour explorer différentes options de solutions ? Et parmi eux, quelle approche est la plus efficace : les algorithmes à objectif unique ou les algorithmes multi-objectifs ?

Objectif

L'objectif est d'utiliser les algorithmes génétiques de Grasshopper pour proposer un grand nombre de types de solutions au début de la phase de conception du projet, enrichissant et facilitant ainsi le processus de décision de l'architecte.

Hypothèse

L'hypothèse de ce mémoire est que les algorithmes génétiques multi-objectifs sont plus adaptés pour explorer les différentes options de solutions dans le processus de conception que les algorithmes mono-objectifs. C'est à travers l'étude d'expériences que le mémoire apportera des éléments de réponses afin de vérifier la véracité de ce postulat.

Manque de recherche

L'architecte Hoda Esmaeilian, qui travaille au département d'architecture de l'université NEU situé à Chypre, a publié un article qui recense et classifie tous les articles d'architecture entre 2014 et 2020 s'intéressant aux méthodes d'exploration architecturale s'appuyant sur les algorithmes évolutionnaires[1]. Elle y relève un manque de recherche sur l'usage des algorithmes multi-objectifs au service de l'exploration des solutions conceptuelles.

Voici un extrait de l'article :

« *La plupart des études dans cette catégorie manquent d'une vision holistique des divers problèmes de conception impliqués, examinant principalement l'algorithme évolutionnaire dans des processus d'optimisation basés sur la performance avec peu de variables. Il existe une divergence de points de vue concernant l'efficacité des algorithmes génétiques (GA) dans la résolution des problèmes de conception multi-objectifs.* »

Le manque de recherches sur l'utilisation des algorithmes multi-objectifs pour l'exploration des solutions conceptuelles est l'une des raisons qui m'ont amené à travailler sur ce sujet.

[1] Les algorithmes évolutionnaires, qui englobent notamment les algorithmes génétiques, sont des techniques d'optimisation basées sur les principes de l'évolution naturelle. Ces concepts seront expliqués plus loin.

II. Etat de l’art

Depuis plusieurs années, les architectes s'intéressent au design computationnel[1]. Notamment, Kostas Terzidis, architecte et professeur d’architecture à l’université d’Harvard, explore dans son travail les méthodes algorithmiques, les types de solutions conceptuelles par le principe de permutation. Dans son ouvrage, « Permutation Design », il explique comment l’architecte peut se servir de la puissance de calcul des ordinateurs à travers des algorithmes simples pour explorer le champ des possibilités de composition spatiale en plan.

[1] Le *design computationnel en architecture*, apparu dans les années 1960, utilise des outils informatiques pour générer et optimiser des solutions de conception, permettant aux architectes d'explorer rapidement des options en réponse à des contraintes complexes.

A. Introduction à la conception paramétrique

Il est important de définir le terme paramétrique dans le concept de « conception paramétrique » pour comprendre la différence entre les logiciels de 3D comme Revit et les logiciels de conception paramétrique comme Grasshopper sur l'interface Rhinoceros. Aujourd'hui, quand on veut parler de conception paramétrique, la plupart des gens utilisent le terme d'architecture paramétrique, qui en réalité n'a pas de sens. Le résultat architectural n'est pas paramétrique mais c'est sa conception qui est réalisée au travers d'une approche paramétrique. Quand on parle de conception paramétrique, on fait référence à la programmation visuelle, comme le propose le plug-in Grasshopper. Les logiciels comme Revit et Archicad utilisent une approche différente dans la construction du modèle. Cette approche est basée sur la paramétrisation de familles d'éléments que l'on assemble ensuite dans le modèle 3D. L'ensemble du modèle ne possède pas cette capacité de transformation qu'offre le modèle paramétrique. Il existe sur Revit un plug In nommé Dynamo qui permet de faire de la conception, d'une façon similaire à celle de Grasshopper sur Rhinoceros.

Cette méthode de paramétrisation permet une grande flexibilité du modèle 3D en offrant la possibilité de faire varier chacun des paramètres à tout instant. Elle permet aussi de tirer parti de la puissance de calcul de l'ordinateur pour alimenter cette recherche, notamment par la simulation et l'optimisation algorithmique, qui fera plus particulièrement l'objet de ce mémoire.

Mario Carpo, théoricien de l'architecture, s'intéressant aux nouveaux outils numériques et à leur impact sur l'architecture a écrit dans son livre, « The Alphabet and the Algorithm » :

« L'un des principaux avantages des modèles paramétriques est qu'ils peuvent être facilement transformés, produisant ainsi des variations de différentes configurations avec les mêmes éléments géométriques. »

Philippe Marin, architecte français, explique le processus de conception paramétrique, dans sa thèse sur l'exploration des mécanismes évolutionnaires appliqués à la conception architecturale:

« Le processus paramétrique s'intéresse à la définition d'un ensemble de paramètres qui influencent la forme. La forme finale n'est pas au centre de la recherche, elle est induite. La modification de la valeur des paramètres engendre non pas un objet, mais un ensemble de variations. Le processus n'est pas simplement fondé sur des valeurs métriques, mais plutôt sur

l'ensemble des relations entre les objets qui composent la forme. Une modification d'un élément entraîne une transformation du système dans son intégralité. »

Les logiciels d'algorithmes visuels 3D

Aujourd'hui, les deux méthodes de conception paramétrique les plus utilisées sont le plug-in Grasshopper, disponible sur le logiciel 3D Rhinoceros, et le plug-in Dynamo, disponible sur le logiciel Revit. Ces deux outils fonctionnent de manière similaire. Dans ce mémoire, nous allons nous concentrer sur Grasshopper, qui est le plus répandu et qui offre un large choix de plug-ins additionnels, comme Octopus, un solveur algorithmique génétique multi-objectifs qui constituera en partie l'objet de ce mémoire. Pour ceux qui ne connaissent pas cet outil, je vais ensuite expliquer son fonctionnement ainsi que celui de ses algorithmes.

La conception paramétrique s'appuie sur un environnement de programmation visuelle qui diffère dans la forme à la programmation scriptée traditionnelle. Elle utilise des fonctions représentées par des nœuds : chaque nœud reçoit des données en entrée et produit une sortie. Ces nœuds, semblables à des blocs sont connectés entre eux par des fils. Cet ensemble de nœuds interconnectés s'appelle un modèle. Cet environnement est connecté à une interface 3D où les éléments géométriques sont dessinés. Le modèle 3D est l'image du modèle paramétrique.

Les composants initiaux, points, lignes ou surfaces, constituent par leurs assemblages et leurs mises en relation les hypothèses du projet. La modification de l'un des paramètres entraîne la modification du système dans son ensemble.

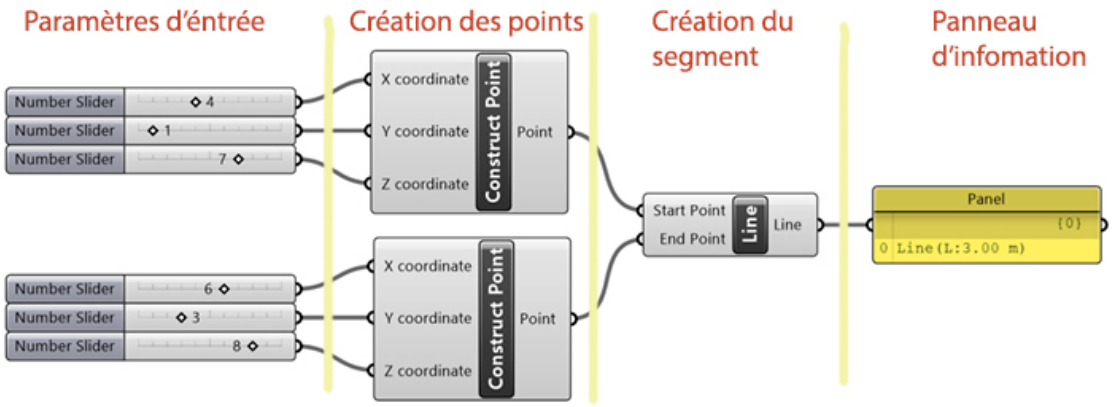

Figure 1 : Exemple de programmation d'une ligne sur Grasshopper

Dans l'exemple de la figure 1, on peut voir comment réaliser un segment dans Grasshopper. Pour créer un segment dans l'espace, on a besoin de connaître seulement les coordonnées spatiales des deux extrémités. Pour cela, on crée deux points à l'aide des blocs « Construct Point », que l'on positionne dans l'espace grâce aux coordonnées (X, Y, Z) qui proviennent des « sliders » (curseurs définissant les valeurs des paramètres d'entrée) et que l'on peut faire varier à tout moment. Ensuite, on connecte les deux points au bloc « Line », qui permet d'obtenir le segment.

De cette manière on créer petit à petit le modèle 3D. L'avantage est qu'il est très facile de modifier des paramètres qui régissent le modèle à tout instant. Le modèle n'est pas juste la succession de trait ou de mur mais bien une structure interconnectée, flexible et homogène.

Outils à la simulation : Les algorithmes génétiques

En conception paramétrique, les modèles sont créés sous forme de scripts visuels, ce qui offre un environnement propice aux développements de logiciels complémentaires (plug-ins) qui peuvent facilement venir se greffer. En effet, la conception paramétrique permet de créer un modèle basé sur des règles mathématiques, ce qui en fait une structure organisée et cohérente. Grâce à ce modèle programmatique, les éléments suivent des relations définies qui favorise l'utilisation de plug-in. En effet les relations mathématiques sous-jacentes au modèle garantissent que chaque modification apportée dans une partie du script se répercute de manière logique et automatisée sur l'ensemble du modèle.

De cette manière, on ne se contente plus simplement de dessiner le modèle, on utilise le logiciel pour concevoir. En effet, pour aller plus loin dans la conception, il est possible d'utiliser des outils algorithmiques qui manipulent les paramètres du modèle, afin d'optimiser un ou plusieurs critères définis par l'utilisateur dans le cadre de la simulation. Il existe un grand nombre d'outils disponibles avec différentes approches techniques. Nous allons nous intéresser en particulier à une méthode très utilisée : les algorithmes génétiques.

Pour mieux comprendre l'imbrication de tous les outils dont je viens de parler, voici ci-dessous, sur la figure 2, un schéma qui résume le cheminement jusqu'à l'utilisation d'algorithmes génétiques.

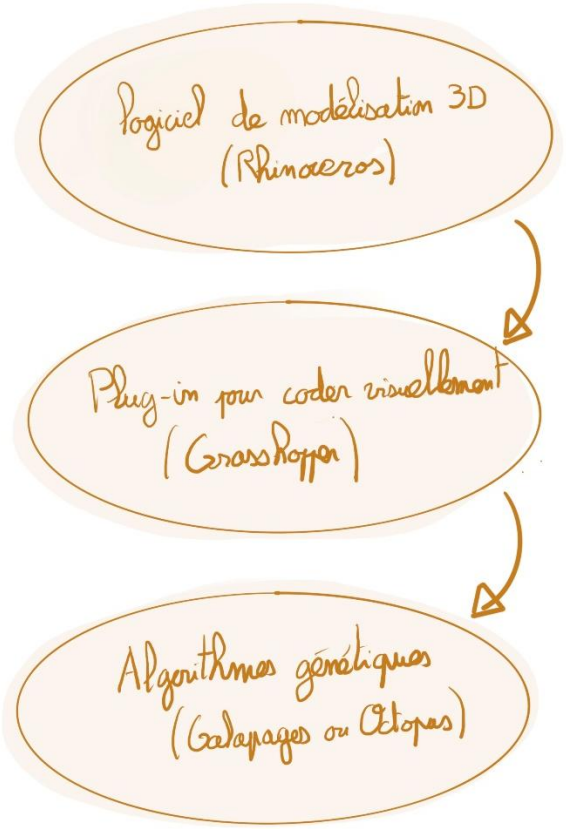

Figure 2 : Schéma d'imbrication des différents outils

Les algorithmes génétiques aident les architectes à explorer et améliorer différentes options de conception. Leur fonctionnement imite le processus de la sélection naturelle : ils génèrent de nombreuses versions d'un design et évaluent leur performance selon des critères définis et conservent les meilleurs résultats. Cela permet d'explorer rapidement un large éventail de possibilités et d'identifier des solutions innovantes et optimisées que les méthodes traditionnelles pourraient manquer.

Cette approche, inspirée des principes de la sélection naturelle et de la génétique est particulièrement efficace pour résoudre des problèmes complexes de conception où plusieurs objectifs doivent être prise en compte. Aujourd'hui, en architecture, ces algorithmes sont principalement utilisés pour optimiser des critères tels que l'efficacité énergétique, l'ensoleillement et la performance structurelle mais on peut imaginer d'autres usages comme on le verra dans la suite de ce mémoire.

Les deux solveurs algorithmiques les plus connus et les plus utilisés en conception paramétrique sont Galapagos et Octopus. Les solveurs algorithmiques sont des applications qui utilisent des algorithmes pour fonctionner ; ils constituent l'interface entre le logiciel et l'algorithme. Pour simplifier, j'utiliserai de la même manière dans le reste de ce mémoire les termes solveur algorithmique et algorithme génétique. Galapagos, est un solveur d'optimisation mono-objectif. Octopus, est un solveur d'optimisation multi-objectifs qui permet de considérer plusieurs objectifs. Galapagos est intégré à Grasshopper tandis que Octopus est un plug-in à télécharger puis à installer sur Grasshopper. Ce sont ces deux outils algorithmiques qui seront utilisés dans les expériences.

B. La conception algorithmique

La théorie du design

John Maeda, auteur américain, connu pour son travail à l'intersection de l'art, du design et de la technologie a dit lors d'une interview :

« Le design est une solution à un problème ; l'art est une question à un problème. »

Le processus de conception des humains semble être un enchaînement de phases où l'esprit humain reçoit, analyse et évalue les informations pour générer des réponses. Ce cheminement, illustré ci-dessous, sur la figure 3, par l'image d'un cerveau qui traite les données, montre comment les informations perçues sont transformées en solutions par notre cerveau.

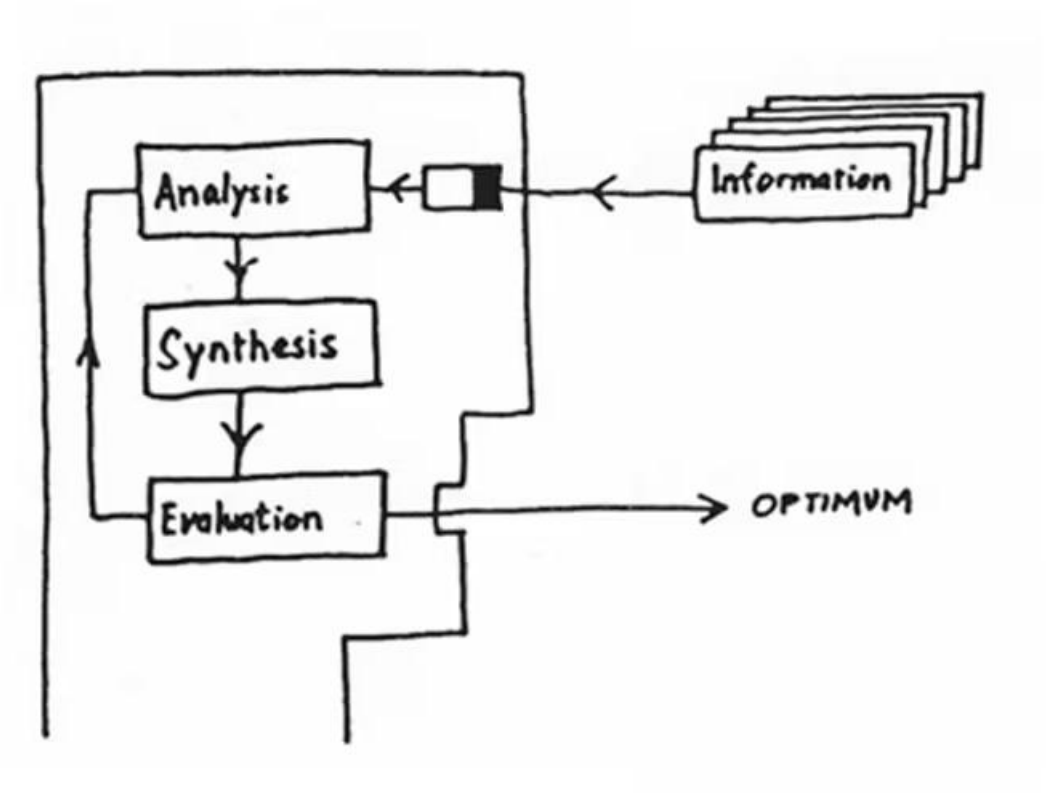

Figure 3 : Illustration du processus de conception, source : InfAR (Bauhaus-Universität)

En architecture ou en design, la distinction entre fonction et forme est cruciale dans le processus de conception : la fonction définit ce que l'objet doit accomplir et les qualités qu'il doit posséder, tandis que la forme exprime sa géométrie et sa matérialité. Ce processus implique de générer une forme à partir de la fonction définie et d'évaluer cette forme pour vérifier si elle répond aux critères établis. Cependant, ce parcours n'est pas linéaire ; il repose sur un système itératif d'essais et d'erreurs. Les multiples fonctions à satisfaire sont souvent ambiguës et sujettes à des changements, tandis que certains critères sont difficiles à quantifier et peuvent même être contradictoires.

Durant cette phase de recherche, l'éventail des possibilités formelles est très large, il est presque impossible de trouver la solution qui puisse satisfaire pleinement la fonction du premier coup.

Pour illustrer ce problème, Reinhard König et Sven Schneider, deux professeurs à l'université Bauhaus de Weimar en Allemagne, ont imaginé une expérience pour démontrer l'immensité de l'espace des possibilités qu'offre l'exploration des solutions. Ils ont utilisé une photo aérienne d'une ville composée de 6 400 pixels, chaque pixel pouvant prendre 256 x 256 x 256, soit 16 777 216 couleurs possibles. L'objectif était de retrouver l'image de la ville en modifiant aléatoirement les pixels. Cette expérience est illustrée sur la figure 4.

Figure 4 : Illustration de l'expérience de Reinhard König et Sven Schneider

Il existe ainsi $16\,777\,216^{6400}$ images différentes réalisables en modifiant les couleurs de chaque pixel. Autrement dit, il est pratiquement impossible de retrouver l'image d'origine de cette manière, car le nombre de possibilités est bien trop grand. Cette expérience montre que, pour certains processus de conception, il est nécessaire d'adopter une forme de raisonnement afin de réduire l'ensemble des possibilités.

L’ensemble des possibilités, appelé "set of all possibilities", est immense. Il englobe à la fois l’ensemble des solutions intéressantes, désigné par "performance space" et l’ensemble des possibilités que le designer va étudier, désigné par "design space". Ces deux sous-ensembles partagent l’ensemble des solutions, nommé "solution space". Pour passer de l’ensemble des possibilités au "design space", les concepteurs utilisent la méthode de l'élimination. Cela consiste à réduire drastiquement le champ des possibilités en formulant des hypothèses de travail. Ce travail initial, qui consiste à établir le problème conceptuel, est crucial. La figure 5, ci-dessous permet de comprendre les relations entre ces espaces.

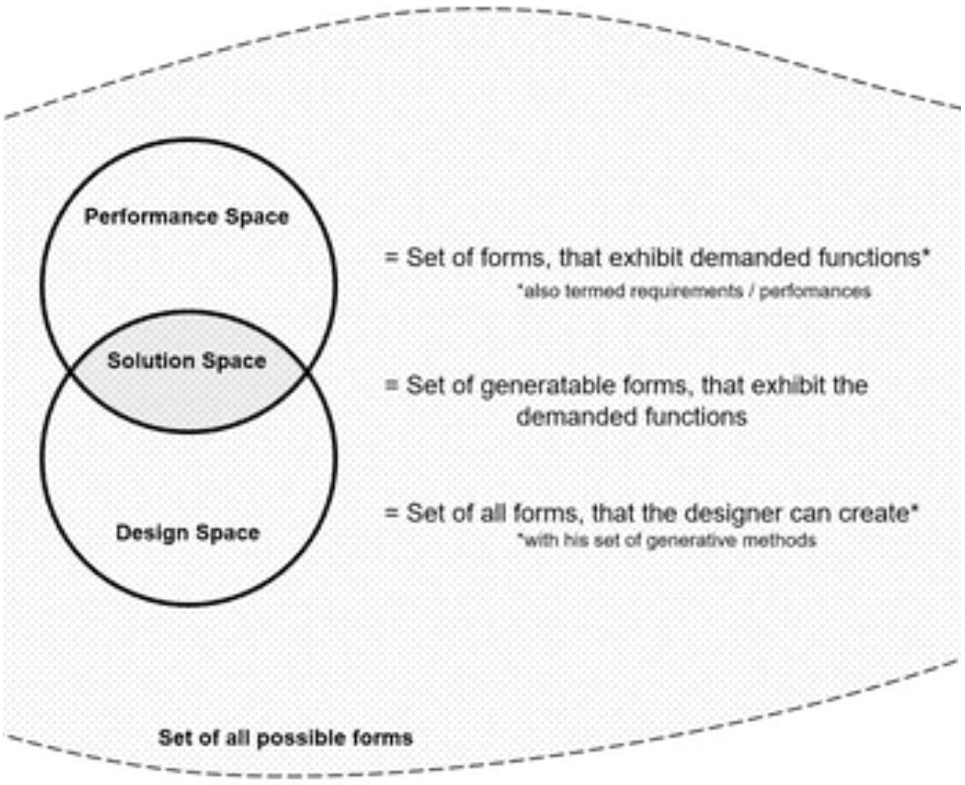

Figure 5 : Illustration des espaces de solutions, source InfAR (Bauhaus-Universität)

Automatique design process

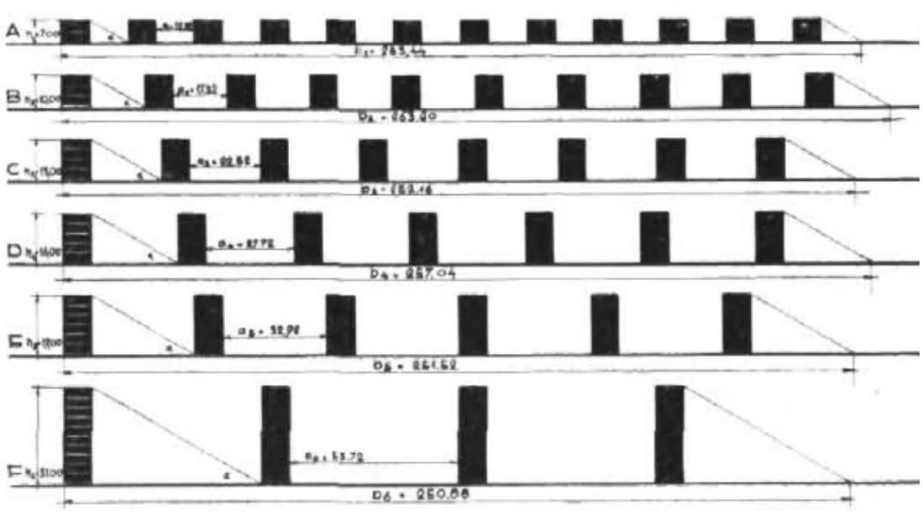

Figure 6 : « Distance between buildings based on lighting » de Gropius en 1931

En 1931, Walter Gropius propose une théorie sur la relation entre la distance qui sépare les bâtiments et leur hauteur pour un ensoleillement optimal, qui est illustrée sur la figure 6.

Cette théorie utilise une règle mathématique simple qui lie la hauteur d'un bâtiment à la distance avec son voisin par un coefficient. Cette relation géométrique pourrait facilement être traduite à un algorithme pour trouver les solutions.

Cet exemple peut nous amener à penser qu'une partie du processus de conception des architectes pourrait se traduire mathématiquement par l'utilisation d'une structure conditionnelle if/else (si/sinon), en fonction du respect des critères d'évaluation, dans ce cas, la distance entre les bâtiments.

En pratique, cela est plus complexe, car l'ensemble du processus conceptuel n'est pas explicitement interprétable sous la forme de critères d'évaluation et donc de structures conditionnelles if/else. De plus, pour représenter l'ensemble de ce processus, il faudrait mettre à la suite un très grand nombre de blocs if/else au sein de l'algorithme. Il semble donc très difficile de résoudre un problème conceptuel de cette manière.

Pour améliorer la performance du modèle, on peut utiliser une approche un peu différente qui consiste à renvoyer l'élément évalué au bloc précédent tant qu'il n'est pas satisfaisant. On passe alors d'une méthode itérative à une méthode récursive qui s'apparente à une boucle. Dans ce cas, un bloc est dit génératif car il est capable de créer des solutions en fonction des réponses du modèle d'évaluation et un bloc évalue les solutions jusqu'à validation des critères d'arrêts.

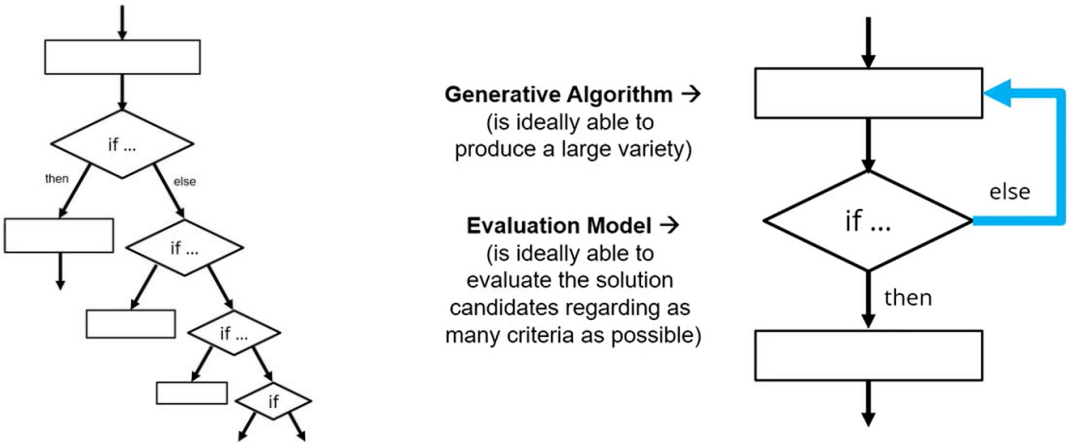

Figure 7 : Source InfAR (Bauhaus-Universität)

Dans la figure 7, le "design space" correspond aux modèles générés par le "generative algorithm" et le "performance space" correspond à tous les modèles acceptés par le "evaluation model". Pour créer le "generative algorithm" il existe plusieurs stratégies.

Parmi elles, la stratégie la plus connue et plus largement utilisée est celle des algorithmes génétiques (AG), comme Galapagos et Octopus, qui permettent d'optimiser des solutions en explorant un large éventail de possibilités, c'est celle-là qui fera l'objet de cette recherche. Il en existe d'autres moins utilisées comme les réseaux de neurones artificiels (ANN), utilisés pour prédire et modéliser des performances complexes ou encore les algorithmes inspirés des colonies de fourmis (ACO), qui offrent des approches spécifiques pour l'optimisation énergétique et les circulations dans les bâtiments et bien d'autres.

C. Les algorithmes génétiques

Le processus génétique

L'évolution génétique est un processus d'optimisation inspiré par la théorie de l'évolution naturelle de Charles Darwin. De nombreux chercheurs se sont intéressés à appliquer ce processus à des logiciels pour recréer la théorie de l'évolution dans un univers numérique, comme l'a montré Karl Sims en 1994 avec son travail "Evolving Creatures".

Le principe est que les meilleurs éléments d'une population permettent de générer la prochaine génération. Une première population d'individus est créée de manière aléatoire pour former la population parentale de la première génération. Ensuite, la qualité de chaque individu est évaluée face aux critères objectifs. Si l'une des solutions satisfait un critère d'arrêt, l'algorithme se termine. Sinon, les principes évolutifs de sélection, reproduction et mutation sont appliqués.

Voici les différentes phases du fonctionnement d'un algorithme génétique :

Initialisation : Une population initiale de solutions est générée aléatoirement.

Évaluation : Chaque solution est évaluée en fonction des critères.

Sélection : Les solutions les plus adaptées sont sélectionnées pour se reproduire.

Croisement : Les solutions sélectionnées échangent une partie de leur ADN (paramètres d'entrés) pour créer de nouvelles solutions (enfants).

Mutation : Des modifications aléatoires sont apportées aux nouvelles solutions pour maintenir la diversité génétique.

Nouvelle Génération : Les nouvelles solutions remplacent les anciennes et le processus se répète jusqu'à ce qu'un critère d'arrêt soit atteint ou que l'utilisateur décide d'arrêter la simulation (comme un nombre maximal de générations ou une convergence vers une solution optimale).

Les algorithmes génétiques font partie de la famille des algorithmes évolutionnaires, qui appliquent le principe de la loi de l'évolution, avec la particularité d'utiliser un système génétique dans le processus. Les algorithmes génétiques sont utilisés dans de nombreux domaines, notamment en architecture. Ils sont très polyvalents, simples à mettre en place et très efficaces.

Leur utilisation dans le domaine de l'optimisation est ancienne, par exemple, John Holland[1], en 1975, dans son ouvrage « *Adaptation in Natural and Artificial Systems* » décrit les bases théoriques et les premières applications des algorithmes génétiques.

Les méthodes analytiques classiques ne sont pas très adaptées aux problèmes non linéaires, discontinus, voire chaotiques, que l'on peut rencontrer en architecture. Les méthodes évolutionnaires sont beaucoup plus performantes dans ce contexte.

La créativité algorithmique

L'utilisation de ces algorithmes sur des problèmes complexes (non linéaire), comme c'est le cas en architecture, soulève la question de la créativité algorithmique et de son importance. Le résultat obtenu par l'algorithme peut être considéré comme créatif ou innovant, Il est donc important de définir et de développer le concept de « créativité algorithmique ».

Parmi les chercheurs qui ont travaillé sur ce sujet il y a notamment, Philippe Marin, dans son article sur Exploration des mécanismes évolutionnaires appliqués à la conception architecturale qui distingue trois formes (niveaux) de créativité : d'une part, l'algorithme et l'ordinateur qui peuvent avoir une capacité créative, d'autre part la solution produite qui peut être considérée comme créative et enfin le dispositif qui peut assister l'activité créative du concepteur.

D'un autre côté, Gero J. S., architecte et chercheur spécialiste sur les outils numériques en architecture, considère que l'ordinateur conçoit de manière créative, s'il est capable de faire évoluer à la fois la solution dans l'univers des possibles et l'espace des solutions lui-même, c'est-à-dire l'espace de recherche. Pour lui, si la solution générée présente un caractère de nouveauté ou d'innovation alors le produit du processus peut être qualifié de créatif.

Beaucoup de chercheurs ont réalisé des travaux sur le sujet, mais ce n'est pas l'objectif de mon mémoire, donc je ne vais pas le développer davantage. Néanmoins, j'aimerais donner mon opinion : il me semble que l'algorithme ne peut fournir des résultats intéressants que si le concepteur l'a mis dans les bonnes conditions. Autrement dit, le rôle du concepteur dans la mise en place des conditions de l'algorithme compte au moins tout autant que le travail de l'algorithme sur la notion de « créativité » des solutions.

[1] John Holland, chercheur américain, spécialisé dans les algorithmes génétiques.

Les algorithmes génétiques en architecture

John Frazer (1995) a été parmi les premiers chercheurs à utiliser les méthodes évolutives dans le design, notamment en architecture et en conception structurelle et à étudier l'aspect génératif des algorithmes évolutifs.

Il existe deux familles d'algorithmes génétiques, les mono-objectif et les multi-objectifs. Les algorithmes mono-objectifs cherchent à optimiser un critère d'évaluation, ce qui peut limiter la diversité des solutions générées mais offrir des résultats performants en optimisation avec un seul critère. En revanche, les algorithmes multi-objectifs, comme Octopus, permettent d'évaluer simultanément plusieurs critères, offrant ainsi une exploration plus riche et une capacité accrue à générer des solutions variées, ce qui les rend particulièrement pertinents pour la conception architecturale.

Nous allons étudier les deux algorithmes génétiques les plus connues et utilisés dans leur famille respective, Galapagos et Octopus.

Mono-objectif

L'un des algorithmes génétiques les plus utilisés en architecture paramétrique est **Galapagos**. Il est intégré à Grasshopper sur le logiciel Rhinoceros. C'est un algorithme très simple d'utilisation qui permet d'optimiser des solutions en fonction d'un seul objectif, en ajustant les paramètres d'entrée d'un modèle pour atteindre une solution optimale.

Dans le cadre de la prise en main de cet outil pour réaliser ce mémoire, j'ai configuré un modèle paramétrique permettant de tester son efficacité à la résolution d'un problème conceptuel n'ayant qu'une seule solution non évidente.

L'objectif de cet exercice est de trouver le plus grand rectangle dans une surface plane quelconque. J'ai donc transposé ce problème sur la toile de Grasshopper en réalisant un modèle paramétrique et en configurant Galapagos pour maximiser la surface du rectangle. Il est important de préciser que la qualité de la réponse algorithmique dépend de la bonne configuration du modèle paramétrique et qu'il existe plusieurs façons de réaliser le modèle. Si l'algorithme est rapide pour tester chaque génération du modèle dans son calcul et qu'il semble converger vers un groupe solution, alors la configuration est satisfaisante.

Sur l'illustration ci-dessous, figure 8, on peut voir à droite, l'interface de Galapagos et à gauche la meilleure solution proposée par l'algorithme.

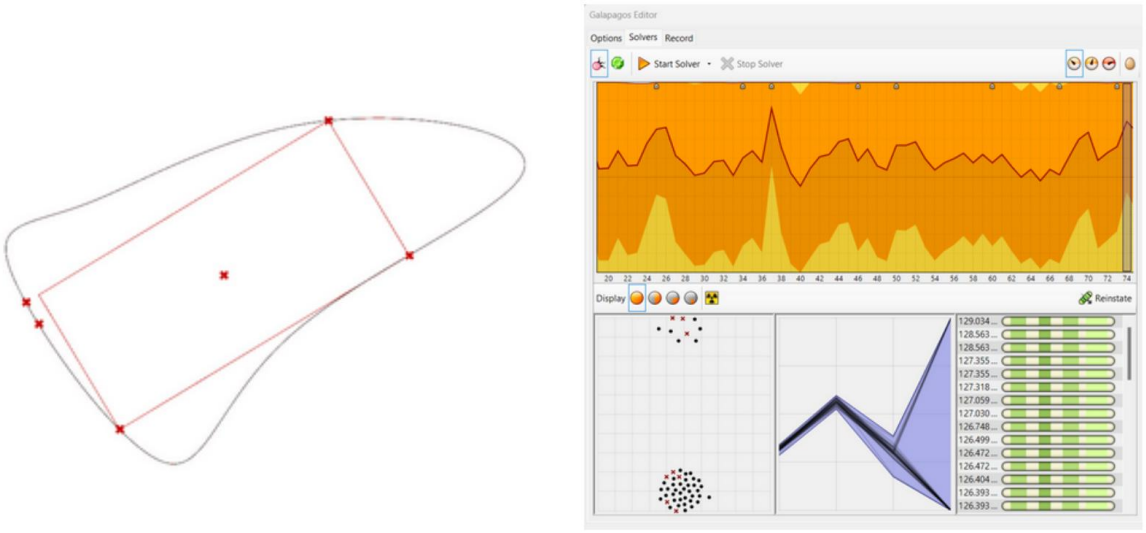

Figure 8 : L'interface Galapagos avec sa solution

Le résultat semble très satisfaisant, car l'algorithme génétique Galapagos converge bien vers la solution idéale. De plus, il l'a fait très rapidement, en seulement quelques secondes, car c'est pour ce type de problème qu'il a été conçu. En effet, les algorithmes génétiques mono-objectif sont idéaux pour converger vers une solution unique.

Multi-objectifs

D'un autre coté il y a la famille des algorithmes génétiques multi-objectifs, comme **Octopus**, qui permettent de considérer plusieurs objectifs simultanément. Ils sont particulièrement efficaces pour explorer les types de solutions, car ils permettent de naviguer dans un espace de solution qui trie chacune des solutions selon leur qualité grâce aux critères d'évaluation.

Il est plus intéressant que les critères soient contradictoires car cela permet d'obtenir un ensemble de solutions pertinentes qui correspond aux meilleurs compromis entre les différents objectifs. Cet ensemble de solutions optimales est appelé solutions de Pareto.

Ces conflits, entre critères contradictoires, permettent à Octopus de diversifier les solutions, ainsi le concepteur parcourt un éventail de propositions équilibrées, chacune répondant différemment aux objectifs fixés. Plus les critères sont contradictoires, plus l'algorithme explore

des solutions variées, ce qui permet à l'architecte de naviguer entre ces choix alternatifs. Cela stimule la créativité de l'architecte et peut l'amener à rencontrer des solutions inattendues.

Une des expériences que j'ai réalisé dans la suite de ce mémoire, est illustrée en-dessous. Pour expliquer brièvement, car une partie de mon mémoire est dédiée à cette expérience, l'objectif est d'explorer les types de solutions possibles d'un parc carré et composé de trois jeux circulaires.

Dans cet exemple, figure 9, on peut voir le diagramme d'affichage des solutions présent sur l'interface d'Octopus. Dans cet exemple, les solutions (carrés rouges), forme le front de Pareto, ce qui signifie que chacune des solutions affichées correspondent aux meilleurs compromis entre les deux objectifs présents sur l'axe horizontal et vertical.

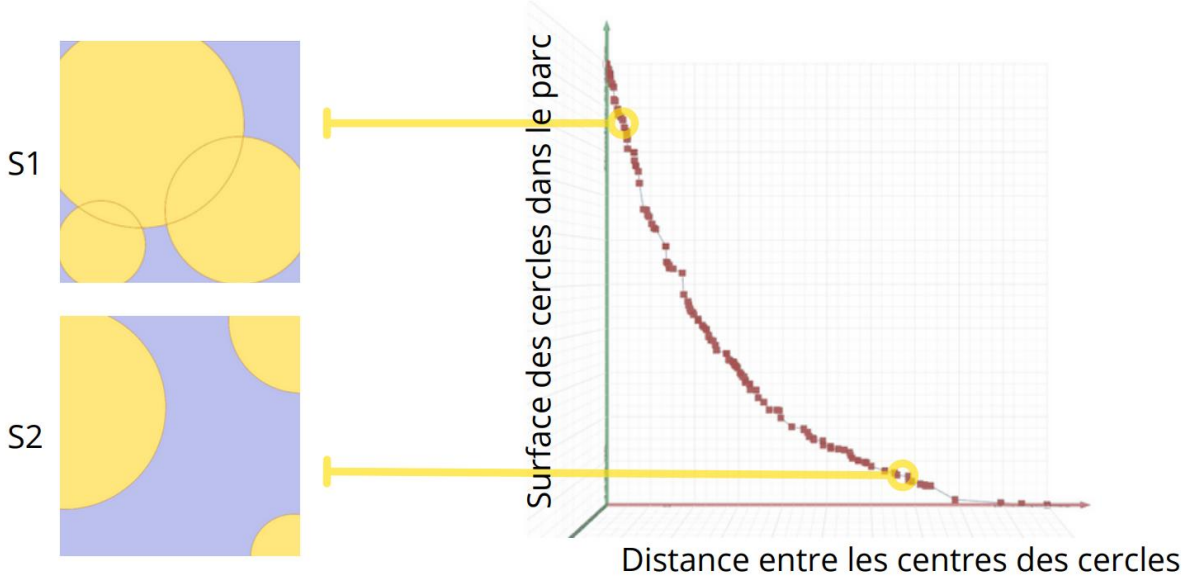

Figure 9 : Graphique d'Octopus avec un zoom sur deux solutions

Nous verrons après, dans la partie expérience le contexte dans lequel s'inscrit cet exemple. Pour l'instant, ce que l'on peut remarquer c'est l'étendue des solutions que l'on obtient à partir de deux objectifs contradictoires que sont : un, contenir les jeux à l'intérieur du parc et deux, avoir le plus de distance entre les jeux. Les algorithmes génétiques multi-objectifs semblent donc particulièrement intéressants pour explorer le champ des solutions conceptuelles.

Exploration et exploitation

On distingue deux types approches dans l'usage des algorithmes génétiques en architecture, l'exploitation et l'exploration.

L'exploitation se focalise sur un seul type de solution et tente de la rendre la meilleur possible. Il s'agit alors d'une recherche locale poussée, c'est-à-dire un zoom sur une partie du paysage des solutions. Dans le but de trouver la meilleure solution dans cette zone correspondant à un optimum local[1] du paysage de solution.

D'un autre côté, l'exploration est une approche qui a pour objectif de parcourir un maximum de types de solutions, permettant de proposer un éventail de solutions possibles pour le projet. Cette exploration permet à l'architecte de nourrir sa créativité, car il peut rapidement parcourir un ensemble de types de solutions et choisir celle qu'il juge intéressante.

Il semble que la majorité des recherches actuelles se concentrent sur un seul de ces deux aspects, l'exploitation. Pourtant les algorithmes génétiques permettent aussi l'exploration créative. Sans chercher à atteindre une solution idéale, mais plutôt à offrir aux concepteurs une meilleure compréhension de l'espace des solutions disponibles comme le développe Stouffs en 2015 dans son article « Types of Parametric Modelling ».

L'optimisation présente des limites lorsqu'il s'agit de problèmes conceptuels impliquant des aspects qualitatifs et subjectifs. En effet il semble compliqué pour l'algorithme d'avoir des critères d'évaluation sur des notions subjectives et/ou non quantifiables. C'est pourquoi l'exploration est intéressante car elle permet aux concepteurs de naviguer parmi les solutions proposées par l'algorithme et de choisir celle qui lui convient.

Cette phase d'exploration peut être représentée par un paysage de solution en plusieurs dimensions. Généralement, ce paysage est à deux ou trois dimensions pour être visualisé facilement. À chaque problème conceptuel correspond une carte de solutions différentes à appréhender.

[1] L'optimum local, dans un paysage de solutions, est une solution meilleure que ses voisines immédiates, mais qui peut être inférieure à l'optimum global.

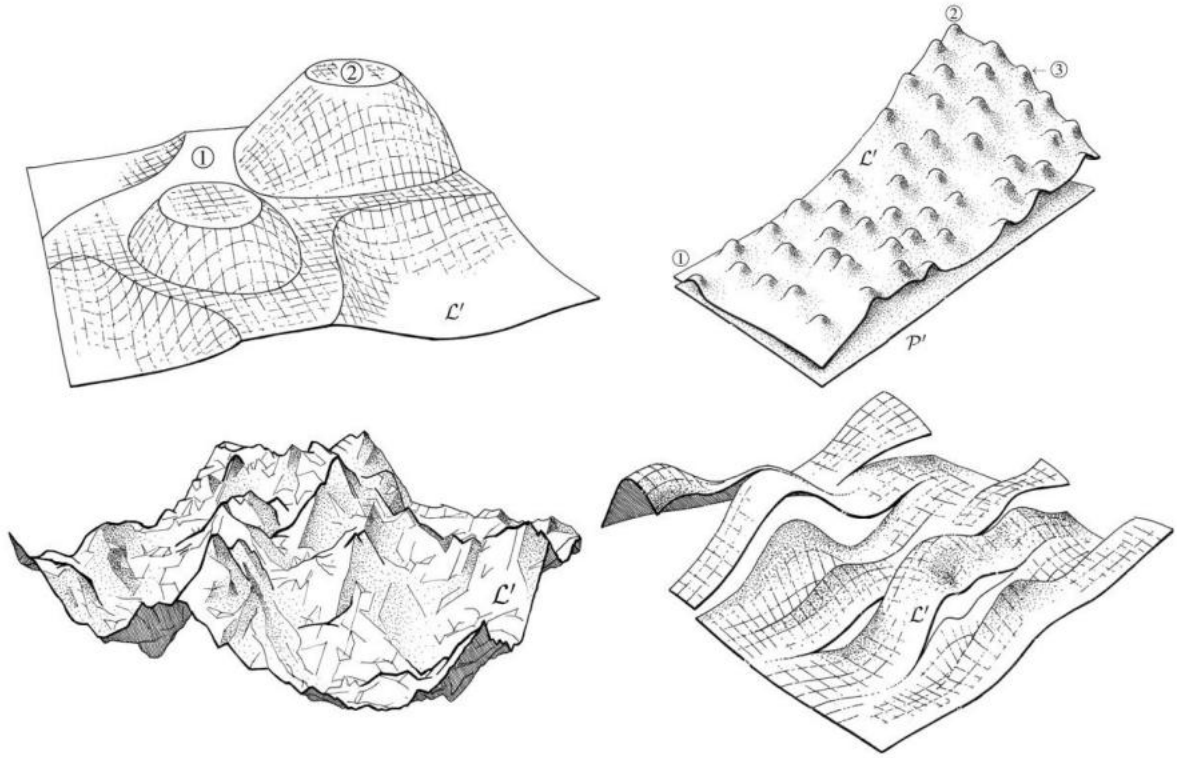

Figure 10 : Exemples de paysages de solutions décrits par D.Rutten en 2014

Sur la figure 10, on peut voir quatre types de paysages de solutions a 3 dimensions décrits par D.Rutten en 2014 qui est un développeur de logiciels et un influenceur majeur dans le domaine de l'architecture computationnelle, notamment grâce à son travail sur Grasshopper.

Dans ces représentations, l'axe vertical correspond à l'objectif et les deux axes horizontaux sont des paramètres d'entrée. Le paysage de solutions peut être d'une grande complexité, il important de bien délimiter la zone d'exploration pour ne pas se perdre dans l'ensemble des possibilités.

Connaitre au mieux le problème architectural pour cibler les zones à fort potentiel de solutions intéressantes semble être la clé pour tirer profit au maximum des algorithmes génétiques. Nous ne pouvons pas avoir la puissance de calcul d'un ordinateur et il est donc difficile pour nous d'explorer autant de possibilité que lui mais nous pouvons imaginer que nous avons une meilleure intuition pour savoir où chercher à l'inverse de l'ordinateur. Donc une approche hybride semble parfaite pour rencontrer la meilleure solution.

Soddu, un architecte et chercheur italien reconnu pour ses travaux sur l'architecture générative et le design basé sur des processus algorithmiques a publié en 1998 des recherches sur le nouveau rôle des designers. Pour lui, avec l'avènement des technologies numériques dans le

champ de la conception, le rôle du designer s'est transformé : *« Du concepteur créateur d'une œuvre, d'une solution unique, on assiste aujourd'hui à l'émergence d'un méta designer créateur d'un ensemble élargi de solutions répondant aux contraintes du problème. Le concepteur ne travaille plus à l'élaboration d'un objet exclusif, mais plutôt à la conception d'une famille de formes, dont la solution retenue représentera un état significatif au sein de cet ensemble de potentialités. »*

Comme l'écrit Pierre Levy (Lévy 1992), le concepteur ne dessine plus un objet mais un système d'objets possibles, une machine à explorer les virtualités. Il parle alors d'une forme de « lâcher prise », car le concepteur doit accepter qu'une part des décisions soit prises par l'outil. L'émergence de nouveautés ou de solutions surprenantes est le résultat du processus. Le concepteur établit les conditions de génération des solutions et ne cherche plus la concrétisation d'une solution exclusive, il effectue des choix à partir de l'ensemble des possibles qui lui est offert.

III. Méthodologie

A. Approche expérimentale

Introdution

Dès le départ, je savais que ce travail m'amènerait à utiliser des logiciels paramétriques pour appuyer mes arguments et justifier mes hypothèses. Pour cela, je voulais mettre en place une expérience permettant d'analyser le processus d'exploration que proposent les algorithmes génétiques. Au travers de cette expérience, j'ai manipulé les deux types d'algorithmes génétiques que je souhaitais étudier : les mono-objectif et les multi-objectifs, représentés par Galapagos et Octopus.

Je rappelle que l'hypothèse principale de ce mémoire est que les algorithmes génétiques multi-objectifs sont plus adaptés pour explorer les différentes options de solution dans le processus de conception que les algorithmes mono-objectifs. J'ai donc réalisé une expérience permettant d'apporter des éléments de comparaison entre ces deux familles d'algorithmes.

Cette phase d'expérimentation qui s'inscrit dans la carde de mon travail sur le parc pour enfant a constitué pour moi un apprentissage continu, me permettant de découvrir ces outils et d'apprendre à les manipuler et ainsi de développer une compréhension avancée du mode de fonctionnement de ces algorithmes.

Naissance de l’idée

Mon point de départ pour cette expérience a été le thème du parc pour enfants. Cette idée m'est venue en raison de la vue depuis la fenêtre de ma chambre, qui, à l'époque où j'ai commencé mes recherches, donnait sur El Parque de las Heras à Buenos Aires, en Argentine. Situé dans le quartier de Belgrano, ce parc, très fréquenté, présente une forme carrée et est agrémenté de jeux pour enfants aux formes circulaires, comme on peut le vois sur la photo, figure 11.

Figure 11: Parque de las Heras, Buenos Aires, Argentina

L’objectif de ce travail est d’assister l’architecte dans la conception d’un parc de jeux pour enfants en lui proposant différents types de solutions, autrement dit, différentes configurations du parc avec ces jeux parmi lesquels il peut choisir.

La phase d’apprentissage

Pour l’expérience, le choix de la forme du parc, du nombre de jeux et des objectifs ont été définis à la suite d’une phase de recherche sur le logiciel, que je qualifierai de phase laboratoire, qui m’a permis de choir au mieux les paramètres de l’expérience.

Cette recherche a pour but de mettre en lumière les qualités d’exploration des algorithmes génétiques. J'ai choisi de réaliser une expérience simple afin de pouvoir analyser facilement les résultats obtenus. Pour représenter le parc, j'ai opté pour un mode de représentation presque

abstrait où le parc et les jeux sont symbolisés par des formes géométriques élémentaires (carrés, cercles, etc.).

Cette exploration des différentes configurations du parc pour enfants inclut l'évaluation des interactions entre les éléments du design. En manipulant des paramètres tels que la disposition des jeux et la relation entre les jeux, j’ai pu observer comment ces facteurs influencent les algorithmes génétiques. Cette approche m'a permis d'appréhender les implications de chaque décision dans la définition des paramètres. En fin de compte, cette phase d'expérimentation vise à enrichir la compréhension des processus de conception assistée par les algorithmes génétiques et à démontrer leur efficacité.

B. Le choix des paramètres

Bien construire le modèle paramétrique est essentiel pour tirer le meilleur parti d'un algorithme génétique. Pour traduire ce processus d'exploration en algorithmes visuels sur Grasshopper, il est important de bien définir les composants et leur connexion dans le modèle paramétrique pour mettre l'algorithme dans les bonnes conditions.

Cela se fait en orientant les recherches vers une zone de l'espace des possibilités qui semble propice à la rencontre de la solution idéale. En effet, l'algorithme manipule les paramètres d'entrée que l'on a définis et qui contiennent des caractéristiques. Ces caractéristiques peuvent prendre la forme, par exemple, d'un intervalle de valeurs possibles pour chaque paramètre ou un pas correspondant à la plus petite variation possible de la variable. Les interactions que l'algorithme a avec ces valeurs dépendent complètement de la configuration des paramètres établie par l'utilisateur.

À travers trois éléments — la manière de traduire le sujet architectural, la configuration des paramètres d'entrée et la mise en place des objectifs avec leur jeu de coefficients — l'architecte délimite un espace de recherche dans lequel l'algorithme pourra naviguer. Comme mentionné précédemment, la liberté que l'on donne à l'algorithme traduit les caractéristiques pour lesquelles le concepteur n'a pas pris de décision, mais également ce qu'il souhaite explorer.

Pour commencer il est préférable d'identifier les objectifs du projet puis de définir les contraintes (les limitations physiques, matérielles, réglementaires, etc.) et de choisir judicieusement les variables, autrement dit, les paramètres d'entrée qui vont être modifiées pour satisfaire les objectifs.

Les objectifs

Généralités sur les objectifs

Il existe deux types d'objectifs :

Les objectifs quantifiables : dimensions physiques, coûts, durée des travaux, efficacité énergétique, quantités de matériaux, propriétés structurelles, performance acoustique, performance lumineuse, flux, impact environnemental.

Un objectif quantifiable est facilement interprétable par l'ordinateur et donc il peut facilement l'évaluer au des critères définis par l'architecte et de continuer sa recherche.

Les objectifs non quantifiables : esthétique, symbolique, flexibilité d'usage, relation avec le contexte.

Les objectifs non quantifiables nécessitent une analyse humaine et ne se prêtent pas aujourd'hui à la modélisation paramétrique, l'algorithme ne peut pas évaluer ce type de critère. C'est pourquoi il semble nécessaire que le concepteur intervienne dans le processus pour apporter son jugement, et donc une approche hybride, intégrant l'optimisation algorithmique avec une évaluation qualitative humaine, semble être la meilleure méthode.

Les objectifs dans le cadre de mon expérience

Dans mon expérience du parc pour enfants, j'ai configuré des objectifs dans le but de trouver un assemblage intéressant des jeux dans le parc. Le problème est que la pertinence d'une solution par rapport à une autre ne semble pas être objective, c'est donc un objectif qui n'est pas quantifiable. J'ai donc choisi de donner comme mission à l'algorithme des objectifs qui soient quantifiables, comme maximiser la surface de jeu dans le parc sans qu'elle se chevauche, ou encore contrôler l'espacement entre les jeux. De cette manière, je peux quantifier ces objectifs et utiliser les algorithmes pour explorer les solutions.

L'objectif en conception paramétrique, correspond à la mesure d'une grandeur physique ou à la valeur permettant d'évaluer l'algorithme. Cela permet de se rendre compte des conséquences de la modification des paramètres d'entrée.

Pour rentrer un peu dans les détails, les objectifs dans les algorithmes génétiques sont conçus pour maximiser ou minimiser une valeur. C'est pour cela que le concepteur joue un rôle important dans ce processus, car c'est lui qui doit préciser la valeur qui lui semble intéressant de maximiser ou de minimiser. Mais il existe des astuces pour demander à l'algorithme de

s’approcher d’une valeur précise, par exemple une surface exacte en mètres carrés. Bien sûr, l’algorithme cherchera toujours à minimiser ou à maximiser une valeur, mais par un jeu de soustractions ou d’additions en amont, il est possible de faire converger l’algorithme vers le nombre choisi, comme illustré ci-dessous sur la figure 12 :

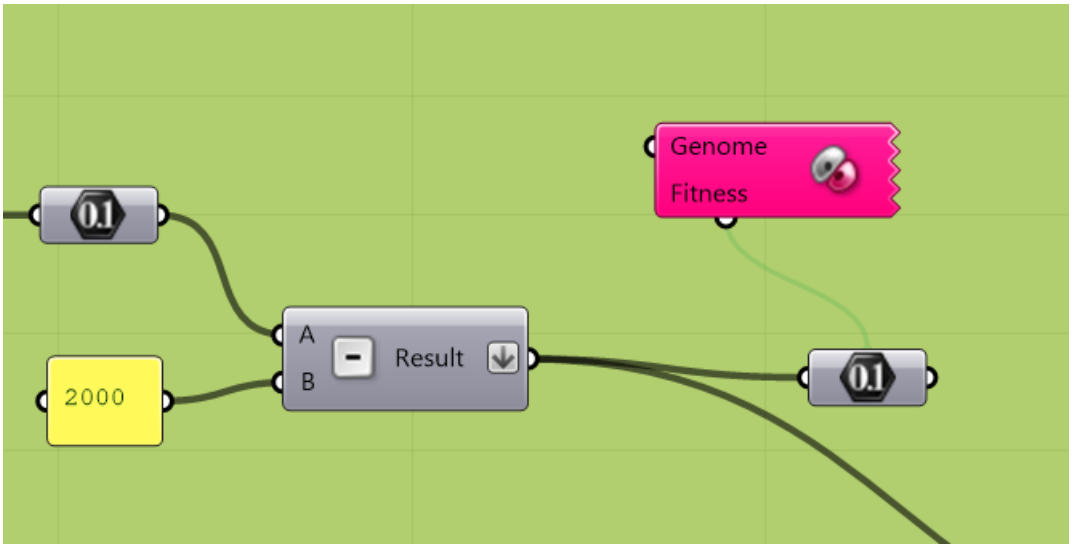

Figure 12 : Extrait du modèle paramétrique du parc pour enfant

Dans cette illustration, j’ai défini mon paramètre objectif pour avoir 2000 m^2 de surface de jeu dans le parc : à gauche j’ai la surface en m^2 des jeux dans le parc et pour « tromper » l’algorithme, j’utilise un bloc de soustractions qui vient soustraire en fonction de la valeur souhaitée le paramètre objectif. Ainsi, l’algorithme Galapagos en cherchant à minimiser le paramètre objectif c’est-à-dire à faire tendre vers 0 ce dernier, va en réalité me permettre d’obtenir une surface totale des jeux dans le parc de 2000 m^2.

Cette démonstration montre que, malgré les règles simples de fonctionnement des algorithmes, une multitude de possibilités d’élaboration du modèle s’offrent à nous en tant que concepteurs.

De la même manière, il est possible de pondérer les valeurs de certains critères pour rendre le paramètre objectif plus intéressant du point de vue de la réponse algorithmique. Par exemple, si l’on veut absolument éviter que les jeux dans le parc se chevauchent, on peut pondérer les surfaces en commun des jeux du parc par un coefficient important, sans affecter de coefficient aux valeurs des surfaces perdues en dehors du parc. Ainsi, on peut regrouper sous un même objectif de "surface perdue" les surfaces partagées par les jeux et les surfaces qui sortent du parc, avec des coefficients différents. Si cela n’est pas clair, je reviendrai plus précisément dessus dans la partie expérience.

En conclusion le rôle du concepteur est de choisir, au travers d'intentions et aussi d'une forme d'intuition, les coefficients qui vont venir pondérer les objectifs, car ils jouent un rôle très important dans le processus algorithmique. Quand je parle d'intention, je fais référence à l'intention architecturale, comme par exemple, dans le cas du parc pour enfants, décider si les jeux peuvent sortir du parc ou dans quelle mesure ils peuvent se chevaucher.

Les contraintes

Dans la conception architecturale, les contraintes jouent un rôle crucial en guidant et en limitant les choix de conception. Ces contraintes peuvent provenir de diverses sources, telles que les normes réglementaires, les exigences des clients, les caractéristiques du site, etc. Elles peuvent aussi provenir d'hypothèses de travail choisies par l'architecte dans la conception du projet. Ce dernier point est très important car, pour définir un modèle paramétrique performant dans la recherche des types de solutions, il faut restreindre judicieusement le nombre de possibilités de solutions que l'on demande aux algorithmes de traiter. Le nombre de possibilités doit être suffisamment grand pour que l'exploration soit pertinente mais pas trop grand pour éviter de se noyer dans l'immensité du champ des possibilités.

Dans le cas de mon expérience du parc pour enfants les contraintes se traduisent par la configuration des paramètres et de la manière dont j'ai créé le modèle du parc. Par exemple, la taille des jeux si elle est fixe ou encore la forme du parc. Dans l'expérience du parc les contraintes sur les paramètres d'entrée ont été expliqué dans la partie expérience (Partie IV p. 37).

Les paramètres d’entrée

Les paramètres d'entrée jouent un rôle majeur dans la programmation d'un modèle destiné à un travail algorithmique. L'ensemble des valeurs que ces paramètres peuvent prendre définit l'ensemble des possibilités du design, également appelé « design space ». Les algorithmes ne peuvent pas traiter des processus d'exploration offrant un trop grand nombre de possibilités. Ainsi, le premier travail consiste à déterminer quels sont les paramètres réellement pertinents pour l’exploration. Voici quelques exemples de paramètres :

- **Dimensions et proportions** : Largeur, hauteur, longueur, rapport d'aspect, etc.
- **Formes géométriques** : Types de formes de base à utiliser (carré, cercle, polygone, etc.).
- **Relations spatiales** : Distances entre les éléments, angles, alignements.
- **Matériaux et textures** : Type de matériaux, propriétés physiques.
- **Fonctions et usages** : Usages des espaces, circulation, accès.
- **Conditions environnementales** : Orientation du soleil, Force du vent

La modélisation d’un paramètre sur Grasshopper est une étape importante dans la définition des paramètres. Ces derniers peuvent prendre plusieurs formes : sliders, booléens, graph mappers, etc. Dans le cas le plus courant du slider, il est important de bien définir la "range" (intervalle d'évolution du paramètre) avec un minimum et un maximum, ainsi que la précision du paramètre pour déterminer le pas de sa variation (par exemple, 10, 1, 0.1, etc.). Ces choix nécessitent une anticipation, voir une intuition de l’architecte par rapport au comportement algorithmique, qui doit définir intelligemment l’ensemble des possibilités, afin d'obtenir une réponse algorithmique aussi pertinente que possible. Cette nouvelle forme de conception, dans laquelle l'architecte apprend à concevoir un modèle paramétrique qui sera traité par l’algorithme, est appelée par Neri Oxman la « conception hybride »[1].

[1] Neri Oxman, qui a travaillé sur la conception computationnelle et les techniques de fabrication avancée, évoque l’idée d’une "conception hybride" où l’homme et l'algorithme interagissent pour générer des solutions architecturales innovantes.

IV. Les expériences

A. Ecriture du modèle du parc pour enfants

Dans cette première expérience nous allons essayer d'utiliser les algorithmes génétiques Galapagos et Octopus pour explorer l'espace des solutions dans la conception d'un parc pour enfants. L'objectif est d'évaluer la performance de ces deux algorithmes l'un mono-objectif et l'autre multi-objectifs dans l'aide à la rencontre de type de solutions à un stade très précoce du processus conceptuel. L'analyse et la comparaison des résultats de ces deux algorithmes permettront d'alimenter les connaissances sur l'intérêt de ces outils en conception architecturale.

Pour trouver le modèle du parc pour enfants le plus intéressant dans l'analyse des résultats, j'ai réalisé plusieurs modèles du parc en expérimentant différentes configurations de ce dernier. Cela m'a permis d'apprendre à manipuler ces outils et à comprendre le fonctionnement des algorithmes génétiques pour éviter de commettre des erreurs dans l'analyse des résultats.

La phase de recherche et d'apprentissage préalable et nécessaire pour l'écriture finale du modèle

Le premier travail consiste à comprendre comment traduire sur l'interface Grasshopper le problème conceptuel du parc. Pour cela j'ai réalisé plusieurs essais de modèle du parc pour enfants.

Sur la figure 13, on peut voir l'interface Grasshopper sur laquelle j'ai modéliser les différentes versions du modèle du parc, chaque arbre paramétrique correspond à un modèle du parc pour enfants. J'appelle arbre paramétrique un ensemble de bloc interconnecté par des fils, il y en a 15 sur la photo. Ils correspondent chacun à une configuration différente du modèle. Ils sont regroupés en quatre catégories, V1 : Modèles sur les surfaces, V2 : Modèles sur les volumes, V3 : Modèles avec variation du nombre de jeu, V4 : Modèles optimisés.

Figure 13 : Interface Grasshopper avec l'ensemble des versions du modèle du parc

Première version

Dans la première catégorie du modèle qui correspond à la première colonne à gauche, j'ai considéré que le parc était carré et que la taille des jeux ne variait pas. Aussi j'ai considéré que la valeur du paramètre objectif à évaluer par l'algorithme serait les surfaces des jeux du parc en comparaison avec la surface du parc. J'ai réalisé trois écritures différentes de cette version, avec soit des jeux carrés, soit des jeux circulaires, comme on peut le voir ci-dessous sur la figure 14.

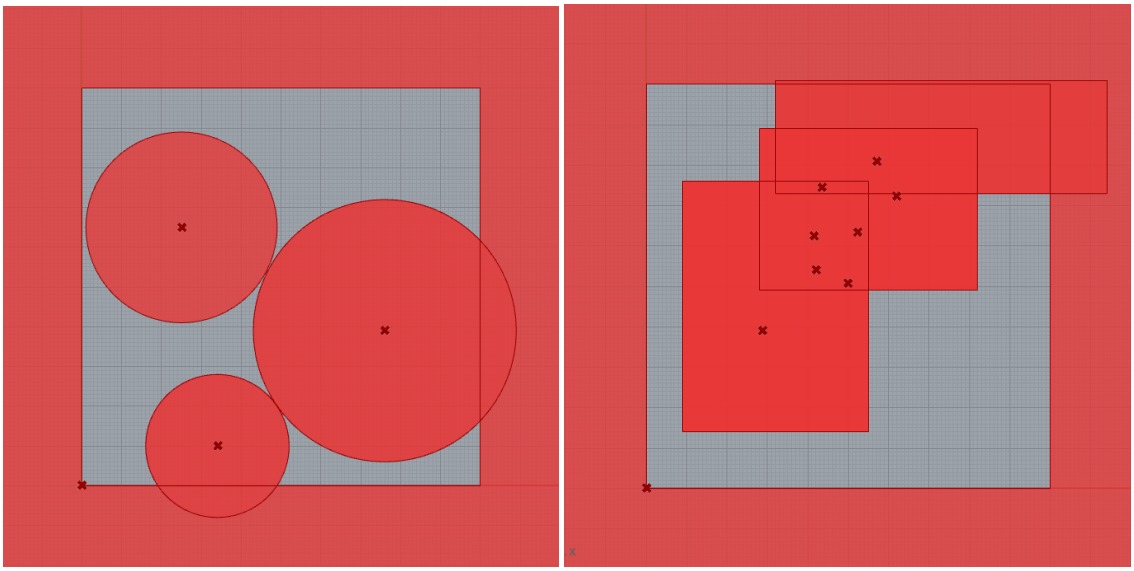

Figure 14 : Interface Rhinoceros des premières versions du modèle

Dans cette première version, il n'y a qu'un seul objectif, celui de minimiser les surfaces en commun entre les jeux avec les surfaces des jeux qui sont à l'extérieur du parc, c'est pourquoi j'ai utilisé Galapagos. Le but étant d'obtenir des types de solutions simples en plans d'assemblages des jeux dans le parc. Il était néanmoins difficile pour l'algorithme de comparer des éléments surfaciques, ce qui pouvait amener des erreurs dans les réponses algorithmiques. En effet, il semble que le traitement du problème par les surfaces ne soit pas adapté au logiciel Grasshopper. J'ai donc changé d'approche pour la deuxième version du modèle.

Deuxième version

Dans la deuxième version qui correspond à la colonne numéro deux sur la première illustration, j'ai utilisé des éléments volumiques pour représenter les éléments du parc et j'ai pu ainsi comparer des interactions de solides beaucoup plus facilement que les intersections de surfaces. En effet il semble que le logiciel Grasshopper soit plus adapté à la comparaison d'éléments volumiques. Comme mon expérience est un travail en plan, j'ai donné une épaisseur identique à tous les éléments pour le ramener à un travail à trois dimensions. Cela reste donc un travail en deux dimensions malgré tout, ce changement permettant seulement à l'algorithme de mieux comprendre. En effet cela a grandement amélioré la réponse algorithmique car il y a eu moins d'erreurs d'interprétation sur les interactions entre éléments.

J'ai donc pu développer plusieurs modèles du parc pour enfants comme on peut le voir ci-dessous, figure 15, en modifiant les caractéristiques du parc comme sa forme : carré, circulaire ou quelconque ; la forme des jeux : Carré ou circulaire ; La variation de la taille des jeux ; Lien entre les jeux ; Choix de la surface total des jeux.

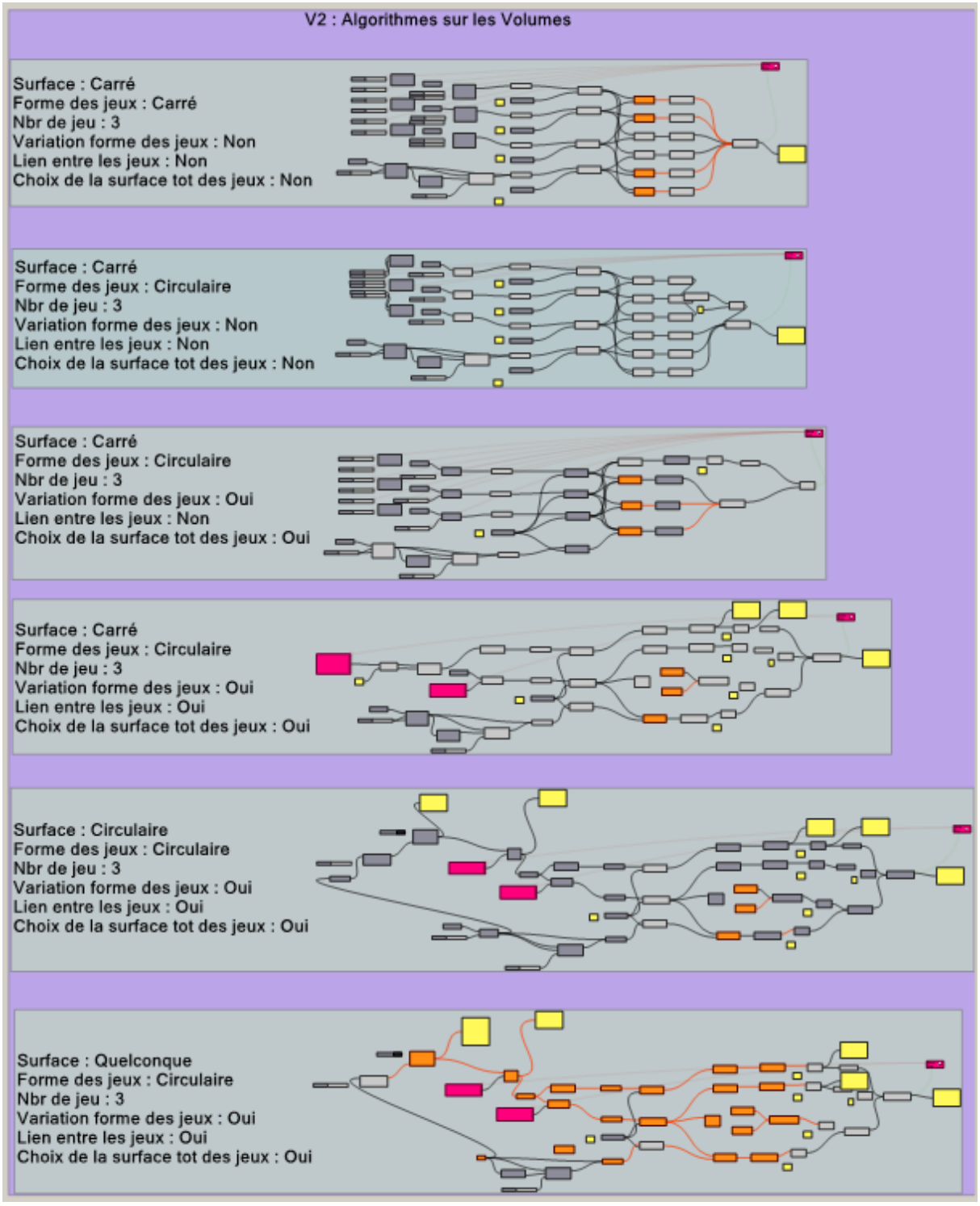

Figure 15 : Interface Grasshopper des modèles de la deuxième catégorie basé sur les volumes

Une fois que l'objectif de maximiser la surface d'occupation des jeux a été maitrisé, j'ai commencé à explorer l'approche multi-objectifs en ajoutant un autre objectif qui a pour but de contrôler la dispersion des jeux. Je reviendrai sur le fonctionnement précis de ces objectifs dans le détail de l'algorithme final. Cette phase d'exploration des modèles m'a permis de comprendre comment réaliser l'algorithme final dont on exploitera les résultats. Voici, figure 16, les représentations géométriques sur Rhinoceros des parcs pour enfants de la deuxième version des modèles :

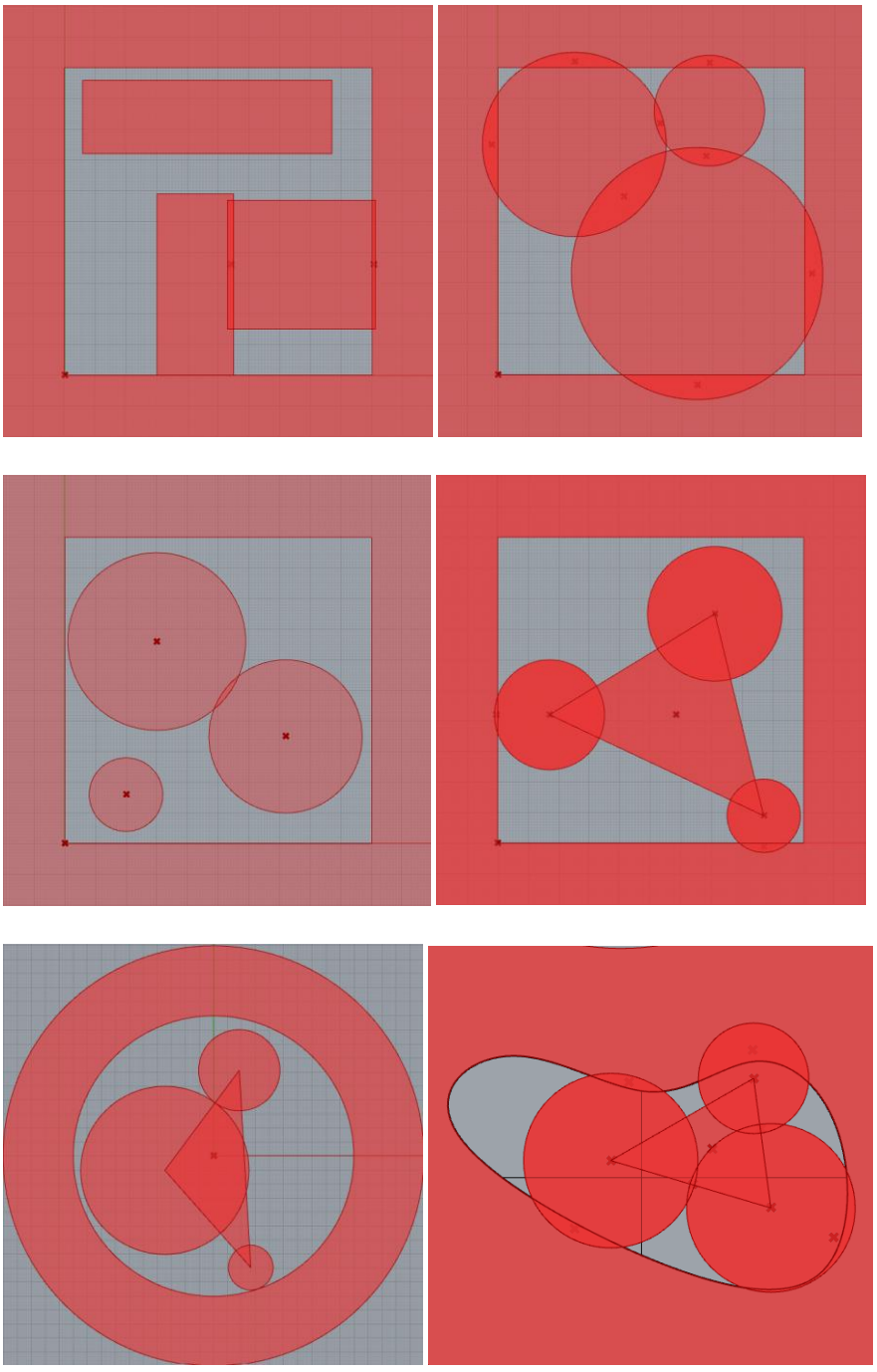

Figure 16 : géométries des modèles du parc pour enfant de la deuxième version

Troisième version

Afin de terminer ma phase d'exploration, j'ai essayé d'ajouter la possibilité de faire varier le nombre de jeux dans l'algorithme. Ce nouveau paramètre est difficile à mettre ne place. Voici ci-dessous, figure 17, une photo de l'algorithme et de sa géométrie sur Rhinoceros.

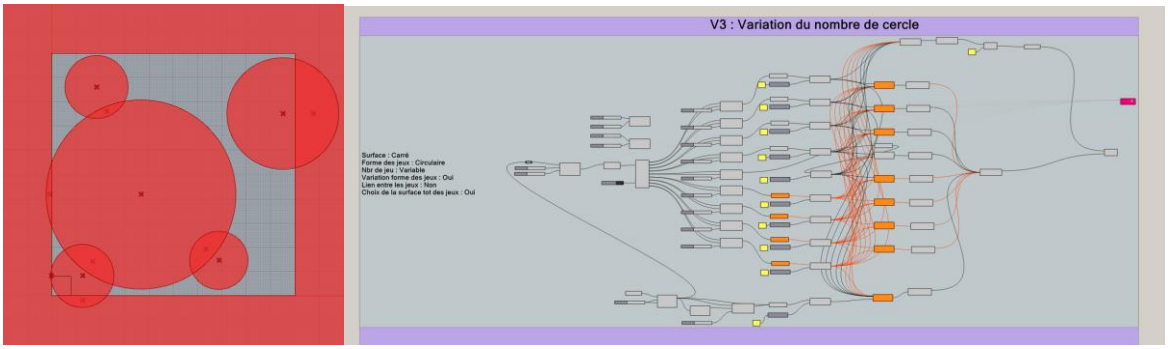

Figure 17 : Algorithme avec modification du nombre de jeu et sa géométrie

Cette version du modèle permet à l'algorithme de faire apparaitre jusqu'à 7 jeux. Mais dans l'écriture de mon modèle, le paramètre du nombre de jeux rendait la convergence des algorithmes génétiques impossible. Autrement dit, il n'y avait pas de correspondance entre le choix des paramètres et la qualité des solutions donc l'étape de convergence génétique ne pouvait pas fonctionner. De plus, la possibilité de modifier le nombre de jeux ne semble pas nécessaire dans le cadre de mon objectif. J'ai donc écarté cette hypothèse.

Quatrième version

Dans la quatrième version du modèle du parc, j'ai mis en place l'algorithme final qui servira à l'exploitation des résultats. Il correspond à la configuration du modèle qui m'a semblé la plus intéressante pour mettre en avant la capacité d'exploration des algorithmes génétiques et pouvoir les comparer. Nous allons commencer par voir le choix des paramètres de ce modèle.

Choix des paramètres du modèle

a) Choix de la forme du parc

J'ai choisi d'utiliser un parc carré, car cela permet de simplifier la continuité des positions que peuvent prendre les jeux dans le parc. Autrement dit, pour l'algorithme, faire varier légèrement la position (x, y) des jeux entraine un petit déplacement de ces derniers, tandis que si la forme est complexe, il faut discrétiser la surface et cela rend complexe la modification des positions des jeux et rend plus difficile pour les algorithmes génétiques de converger vers un type de solution. Il est sûrement possible de résoudre ce problème pour utiliser les algorithmes génétiques dans de meilleures conditions, mais ce n'est pas l'objet de mon mémoire.

b) Choix du nombre, de la forme et de la taille des jeux :

J'ai choisi de manipuler seulement 3 jeux car comme je l'ai dit précédemment, sinon cela complexifie dans la convergence de l'algorithme. Quant au choix de la forme et de la taille des jeux, j'ai remarqué en manipulant les modèles du parc que les expériences avec un parc de forme carré et des jeux circulaires offraient des solutions intéressantes d'emboitement. En effet, si on positionne 3 jeux circulaires dont les tailles ne permettent pas qu'ils rentrent entièrement dans la surface du parc, les solutions au problème peuvent se regrouper par type de solution et sont facilement exploitables pour l'analyse des résultats. C'est pourquoi, j'ai choisi d'utiliser 3 jeux circulaires, de trois tailles fixes différentes, un petit, un moyen, un grand.

L’importance de l’optimisation dans l’écriture

Grasshopper offre une large palette de fonctions ou blocs qui permettent de définir notre modèle paramétrique. Souvent, plusieurs chemins sont possibles, c’est-à-dire qu’il existe une autre manière d’écrire le modèle sur le logiciel. Il est donc important de bien connaitre le fonctionnement du logiciel pour réaliser le modèle le plus optimisé.

Dans le cadre de l’utilisation de solveur algorithmique, si le modèle n’est pas optimisé, il y aura plus de calcul à chaque itération de la simulation. En conséquence, cela pourrait soit allonger le temps de réponse algorithmique, soit entraîner l’échec du processus. Autrement dit, s’il existe un code du modèle moins coûteux en calcul pour l’ordinateur, il est préférable de l’utiliser.

C’est pourquoi dans l’écriture du modèle final, j’ai essayé d’utiliser le moins de blocs possible pour favoriser la qualité des solutions algorithmiques. Voici ci-dessous, figure 18, le modèle optimisé.

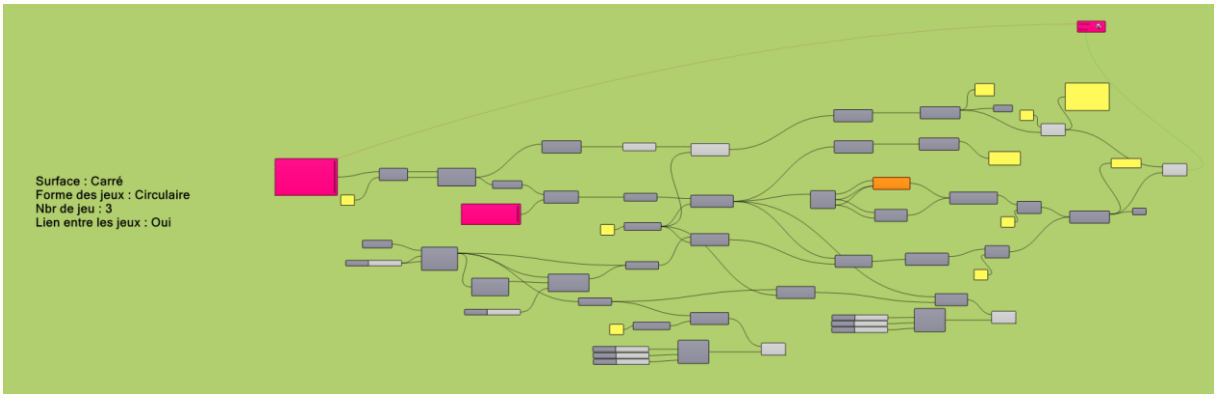

Figure 18 : Algorithme du modèle final

La lecture du modèle se fait de la gauche vers la droite, des paramètres vers les objectifs. Pour expliquer le fonctionnement de l’algorithme, je vais le découper en deux parties, la création des jeux avec leur paramètres et l’évaluation des objectifs.

Explication du modèle final

Première partie, mise en place des éléments :

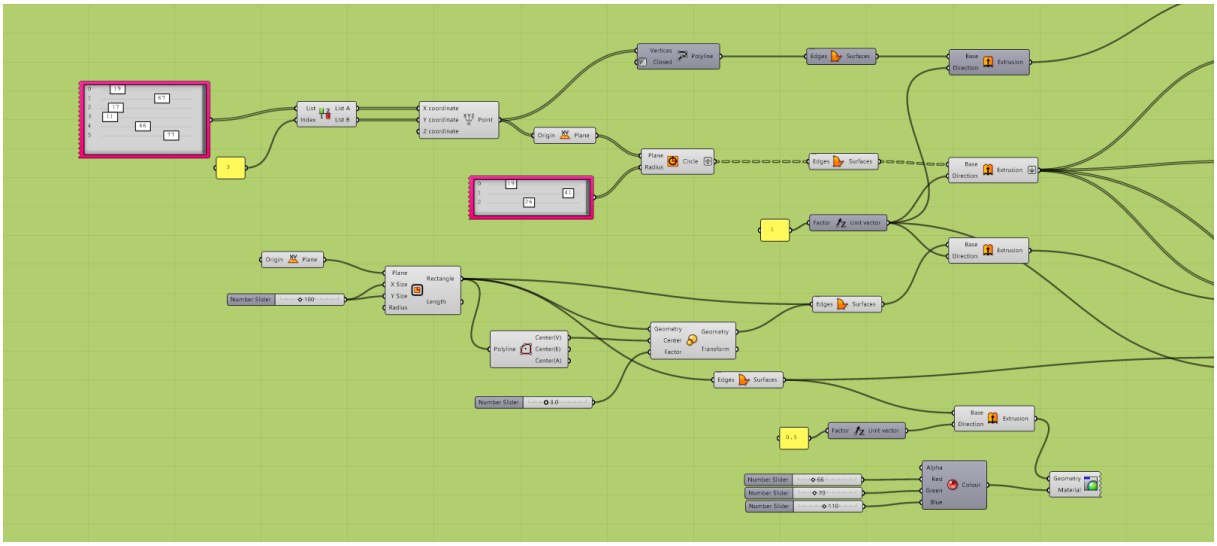

Figure 19 : Première partie de l'algorithme du modèle final

Comme on peut le voir dans la figure 19, j'ai commencé par créer le parc en utilisant le composant « rectangle » en définissant une taille de 100x100 à l'aide d'un composant « slider ». J'ai ensuite voulu créer la bordure du parc qui servira par la suite pour évaluer si les jeux sortent du parc. Pour cela j'ai connecté mon rectangle avec l'élément « scale » qui m'a permis d'avoir une projection plus grande de ce rectangle ; puis en connectant mon rectangle et sa projection au composant « boundary surfaces » j'ai pu récupérer sa bordure. J'ai ensuite extrudé la bordure d'un mètre en utilisant le composant « extrusion ».

D'un autre côté, j'ai créé les jeux en utilisant le composant « contruct point » qui permet de créer des points en fonction des coordonnées sur X et Y que je lui donne en entrée. Pour les coordonnées, j'ai utilisé le composant « gene pool » qui permet de configurer plusieurs sliders en même temps. C'est justement sur ces paramètres que l'algorithme va agir. Comme je souhaitais avoir trois jeux circulaires, j'ai construit trois points que j'ai utilisé pour être le centre des cercles des jeux. Pour ça, j'ai connecté ces points aux composants « circle » en indiquant avec un autre composant « gene pool » la taille des trois jeux que je souhaitais. Ensuite, j'ai extrudé les surfaces des jeux à l'aide du composant « extrusion » comme dans le cas de la bordure du parc.

Deuxième partie, l'évaluation des éléments :

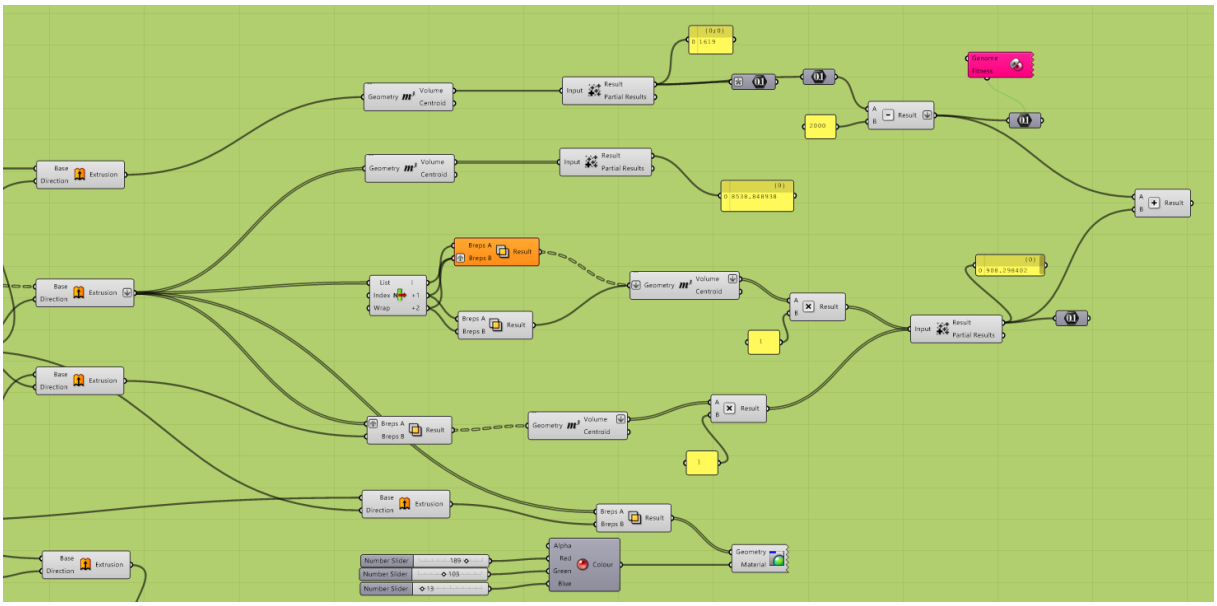

Figure 20 : Deuxième partie de l'algorithme du modèle final

Comme on peut le voir dans la figure 20, pour savoir si les jeux se chevauchent ou s'ils sortent du parc, j'ai utilisé le composant « solid intersection », qui me permet d'obtenir les volumes d'intersection entre les jeux. De la même manière, j'ai comparé les volumes des jeux avec la bordure pour déterminer s'ils sortaient. J'ai ainsi pu récupérer les valeurs, en mètres cubes, du volume de chevauchement des jeux entre eux et avec l'extérieur du parc. À ce moment-là, j'ai attribué des coefficients à ces valeurs pour affiner le paramètre objectif, qui, dans ce cas, est la somme des volumes que je viens de calculer.

Pour évaluer le deuxième objectif, j'ai créé un triangle grâce au composant « polyline » à partir du composant « construct point », qui m'a servi à créer les centres des trois jeux. J'ai ensuite converti cette polyline en surface à l'aide du composant « boundary surfaces », puis je l'ai extrudée d'un mètre. Enfin, j'ai mesuré la surface, en mètres cubes, du triangle extrudé à l'aide du composant « volume ». De cette manière, j'ai pu utiliser cette valeur en mètres cubes comme paramètre objectif. Plus cette valeur est importante, plus les jeux sont dispersés.

B. Analyse des résultats

Voici ci-dessous les configurations des algorithmes génétiques Galapagos et Octopus dont je me suis servi pour exploiter les résultats.

Ces paramètres ont été choisis pour favoriser l'exploration de l'espace des solutions. Je donne tous ces informations sur le réglage des algorithmes génétiques pour que toute l'expérience soit transparente et reproductible. Je ne développerai pas d'avantage sur le sujet de la configuration des algorithmes qui ne fais pas l'objet ici de mon mémoire.

Configuration Galapagos

L'étude a été menée sur 25 générations, avec une population initiale de 100 individus et une population par génération de 50 individus. Les paramètres spécifiques définissant l'évolution des générations étaient les suivants :

Maintien (5 %) : Ce paramètre assure que 5 % des individus les plus performants d'une génération sont conservés dans la suivante, ce qui permet de préserver les meilleures solutions identifiées tout en laissant de la place pour l'exploration de nouvelles solutions.

Taux de consanguinité (70 %) : Ce taux élevé favorise le croisement entre individus ayant des caractéristiques similaires, renforçant l'exploitation des meilleures solutions en affinant leurs variantes.

Population initiale (100 individus) : Une taille initiale modérée permet de commencer l'optimisation avec une diversité raisonnable, offrant un bon équilibre entre performance de calcul et diversité génétique.

Population par génération (50 individus) : La population à chaque génération.

Configuration Octopus

L'étude a été menée sur 25 générations, avec une population initiale de 250 individus. Voici les paramètres spécifiques définissant l'évolution des générations :

Taux d'élitisme (0,5) : Ce paramètre garantit que 50 % des individus les plus performants d'une génération sont automatiquement conservés pour la génération suivante, préservant ainsi les meilleures solutions et évitant leur perte au cours de l'évolution.

Probabilité de mutation (0,8) : Cette valeur indique qu'une mutation est appliquée à 80 % des individus, permettant d'introduire une diversité génétique importante et de réduire le risque de convergence prématurée vers une solution sous-optimale.

Taux de mutation (0,9) : Pour les individus sélectionnés pour la mutation, 90 % de leurs gènes sont modifiés, favorisant une exploration plus large de l'espace des solutions possibles.

Taux de croisement (0,6) : Ce paramètre contrôle le mélange des caractéristiques entre deux individus lors du croisement, en fixant à 60 % la proportion d'individus créés par recombinaison des gènes parentaux.

Ces réglages permettent de maintenir un équilibre entre l'exploration de nouvelles solutions et l'exploitation des meilleures solutions identifiées à chaque itération.

Le tableau des résultats

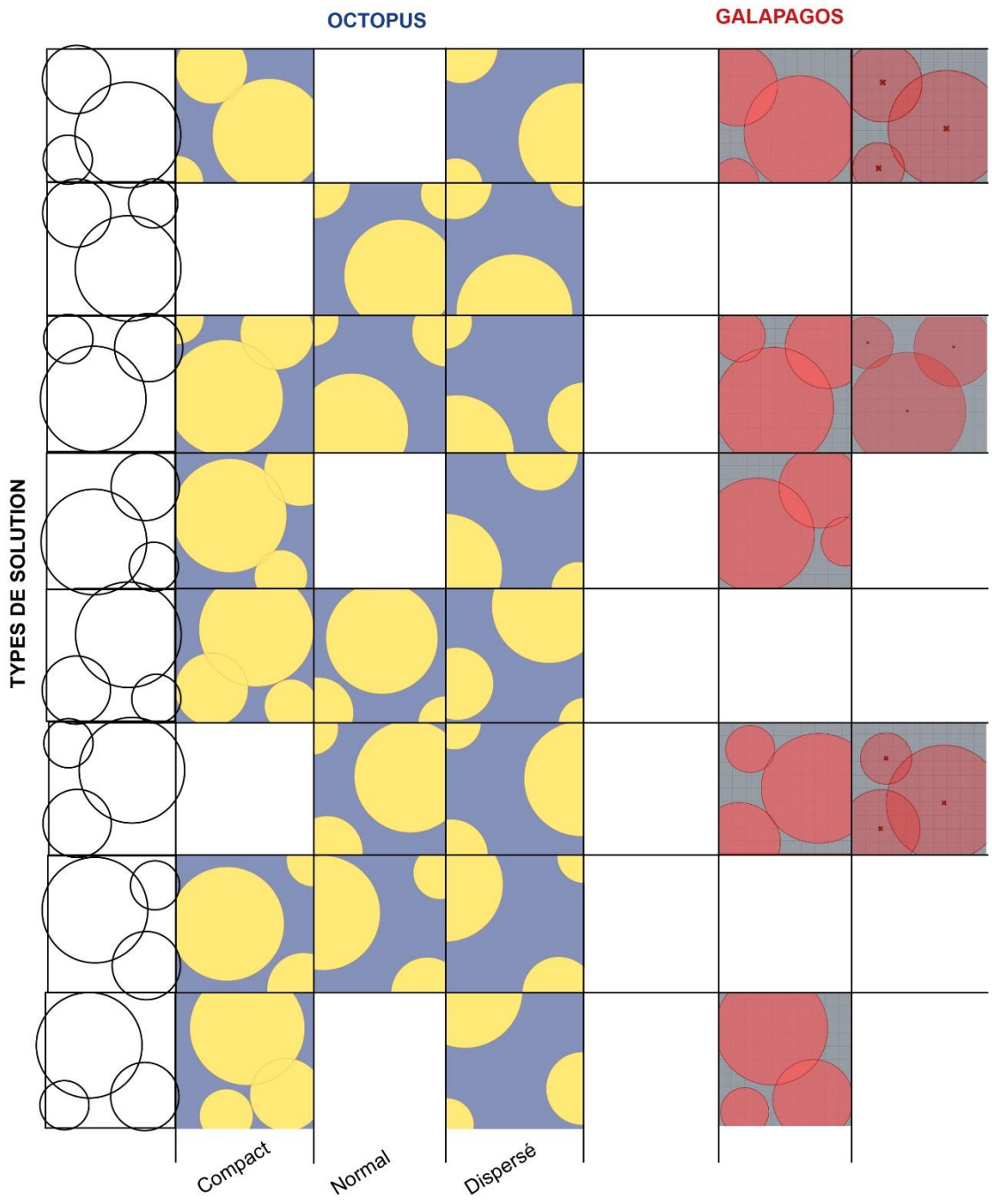

Figure 21 : Tableau des résultats

Classification des types de solutions

La configuration du modèle du parc pour enfants offre un nombre suffisamment petit de type de solutions pour qu'il soit possible de les trouver manuellement. En effet, pour satisfaire le critère de maximisation de surface de jeux dans le parc, il faut que les deux jeux de plus grande taille se mettent sur une diagonale et que le petit jeu occupe l'un des deux angles restants. Il existe donc 4 configurations possibles des jeux de grandes tailles sur les diagonales et pour chacune d'entre elles, il existe deux positions possibles de jeux de petite taille dans les angles restants. Il y a donc 8 solutions évidentes pour satisfaire au mieux l'objectif de maximisation de surface de jeux dans le parc.

En utilisant ces 8 types de solutions, cela permet de définir un cadre dans l'analyse des résultats et leur comparaison avec les solutions de l'algorithme mono-objectif Galapagos. Quant au second critère objectif, qui est de maximiser la distance entre les centres des jeux, il est représenté en trois catégories dans le tableau : compact, normal, dispersé. Ces catégories je les ai choisies pour permettre une représentation de ce critère objectif.

Galapagos, l'algorithme génétique mono-objectif

Pour l'utilisation de l'algorithme mono objectif Galapagos, j'ai réalisé huit fois l'expérience et j'ai obtenu à chaque fois un type de solution qui correspondait parfaitement aux attentes du critère de maximisation de surface de jeux dans le parc mais Galapagos ne m'offrait pas d'autre choix de type de solution. Après avoir effectué huit itérations de Galapagos, j'ai donc obtenu cinq types de solutions qui j'ai inscrit dans le tableau des résultats.

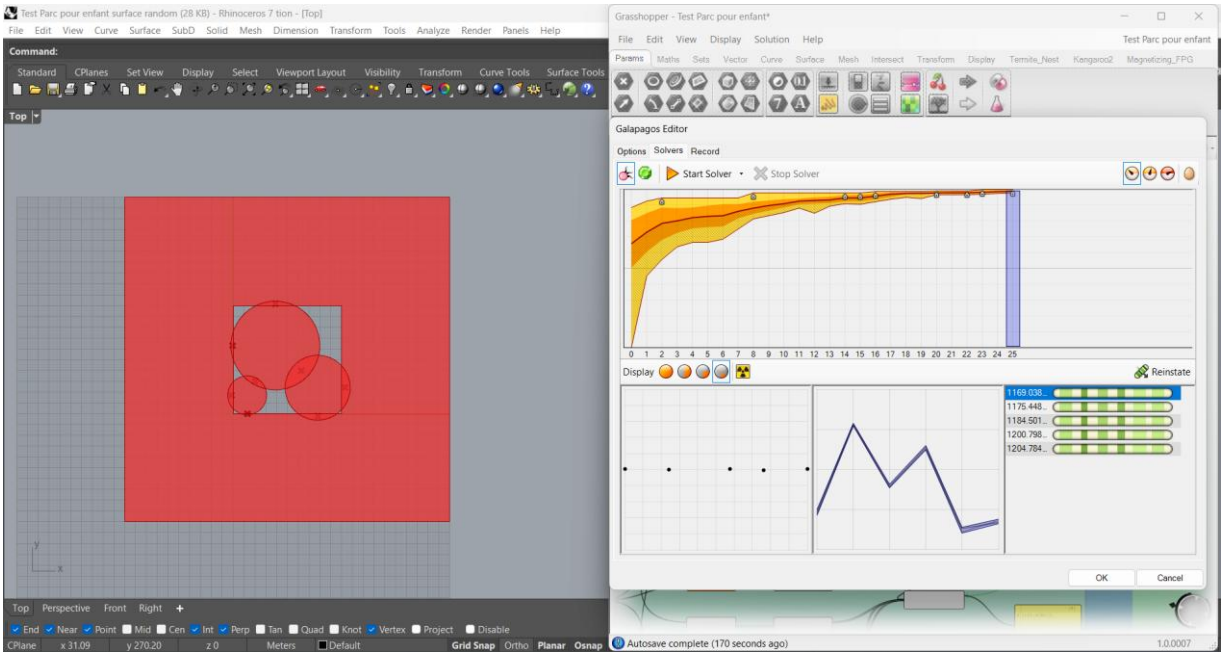

Figure 22 : Interface Galapagos après l'arrêt de l'algorithme

La figure 22 représente l'interface d'affichage des résultats de Galapagos. On peut voir la vitesse de convergence du modèle sur la partie de droite. Mais aussi que le graphique en dents de scie juste en-dessous qui indique la différence génétique entre les 5 meilleurs solutions ne montre quasiment aucune différence entre ces solutions. Autrement dit, les meilleures solutions conservées par l'algorithme au bout d'un certain moment sont presque identiques car elles correspondent à une valeur élevée du paramètre objectif.

Il est intéressant de noter que Galapagos ne cherche pas de juste milieu entre deux objectifs car il ne peut contrôler qu'une valeur qui correspond au paramètre objectif. Autrement dit, si l'on a dissimulé des objectifs en les pondérant derrière ce paramètre objectif, il ne saura pas les différencier. Cette notion est très importante car pour choisir quel objectif va prendre le dessus il suffit de l'indiquer dans la pondération des valeurs de deux objectifs. Dans cet exemple, l'objectif principal est de maximiser les surfaces de jeux dans le parc et le second est d'écarter le plus possible les centre des jeux entre eux. J'ai choisi de pondérer le critère d'écartement des

jeux de telle manière à ce que l'objectif des surfaces reste prioritaire. J'ai expliqué cette stratégie, Partie III, B), dans la partie sur la définition du paramètre objectif.

Cette notion permet de comprendre le principe fondamental qui sépare ces deux algorithmes. Galapagos se concentre sur une valeur à optimiser alors que Octopus peut différencier les objectifs et donc apporter une exploration spécifique à chaque objectif. Ce qui se traduit dans le graphique par un front de Pareto qui permet une exploration plus large de l'espace des solutions.

Pour trouver un type de solution Galapagos est très performant, cela permet à l'architecte d'avoir une piste pour commencer ces recherches mais cela ne permet pas d'avoir une vision d'ensemble sur l'éventail des solutions possibles.

Octopus, l'algorithme multi-objectifs

Pour regrouper les résultats dans le tableau par type de solution évidente, je me suis aidé de l'option Parameter Diversity qui permet de trouver plus simplement les types de solutions décrit précédemment. Cette option ne sert qu'à organiser différemment le nuage de solutions après le travail de l'algorithme en offrant une troisième dimension dans le graphique d'affichage des solutions. Cela permet de regrouper les solutions dont les valeurs des paramètres sont proches. Pour rappel, les deux autres axes dans le graphique représentent les objectifs de surface des jeux dans le parc et de dispersion de ces derniers. Après avoir parcouru les solutions du graphique, j'ai rangé les solutions trouvées dans les 8 types de solutions déterminées précédemment.

Voici ci-dessous le graphique des solutions obtenue avec l'utilisation d'Octopus :

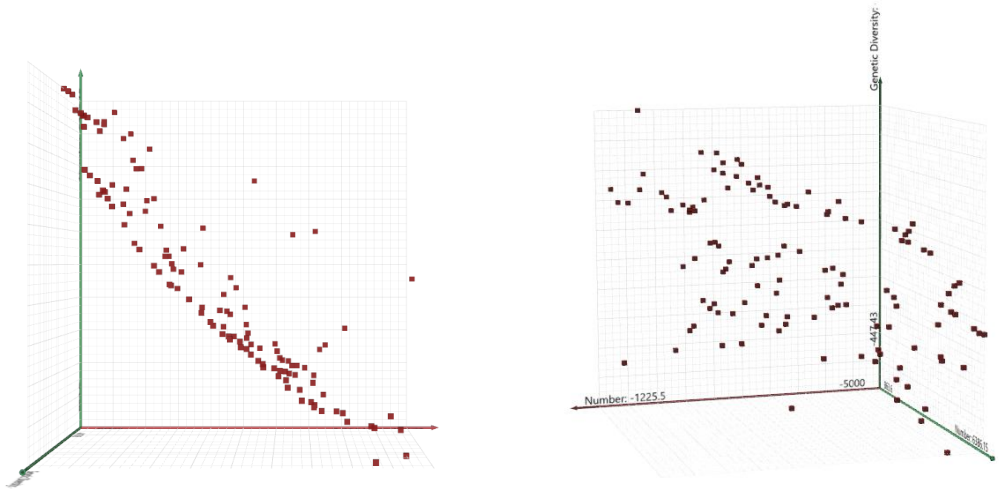

Figure 23 : Graphique de représentation des solutions

Les deux graphiques, figure 23, représentent le même nuage de solutions mais observé sous un angle différent. Dans le graphique de droite, les axes horizontaux correspondent aux paramètres objectifs et l'axe vertical à la diversité génétique. Dans le graphique de gauche en revanche, les deux paramètres objectifs sont sur le plan frontal et cela permet d'observer ce que l'on appelle un front de Pareto. Pour rappel, cela correspond aux solutions qui sont les meilleurs compromis entre les objectifs. Toutes les solutions ne sont pas affichées pour ne pas perdre en lisibilité dans l'utilisation du graphique, dans ce cas, ne sont affichées que les meilleures solutions, celles qui appartiennent au front de Pareto. Comme le graphique est en 3 dimensions avec le paramètre de Genetic Diversity, le front de Pareto est un peu particulier car il est n'est plus une simple courbe mais une nappe à trois dimensions qui considère aussi les diversités génétiques comme un

objectif. L'option de diversité génétique permet de simplifier la phase de recherche des types de solutions qui se fait manuellement. Pour visualiser une solution, il suffit de cliquer sur l'une d'entre elles dans le graphique et elles s'affichent sur l'interface 3D de Rhinocéros.

Conclusion intermédiaire

Octopus se distingue par sa capacité à explorer un large spectre de solutions grâce à la prise en compte de multiples objectifs. Cette exploration permet de générer un front de Pareto qui met en évidence les compromis possibles entre les différents critères. Il semble que son principe de fonctionnement soit favorable à l'exploration des types de solutions.

En revanche, Galapagos, bien qu'efficace pour optimiser un objectif unique, ne propose pas de variété significative dans ses solutions. Les cinq itérations effectuées ont produit des résultats similaires, se concentrant uniquement sur la meilleure valeur pour l'objectif défini. Cela limite la capacité à explorer différentes alternatives de conception. Il semble que son intérêt dans l'explorations des types de solutions soit moins pertinent que celui d'Octopus.

Critique de l'expérience du parc

Dans l'expérience du parc, les deux objectifs sont très liés (pour rappel : la surface des jeux dans le parc et l'écartement des jeux entre eux). Il serait intéressant de faire une expérience avec deux objectifs qui sont beaucoup moins en relation pour observer la réponse algorithmique, soit les compromis trouvés.

D'un autre côté, le nombre de types de solutions lié au critère de surface était suffisamment simple pour être identifiable manuellement donc il n'y pas eu de réel enjeu de recherche sur les solutions obtenues.

C'est pourquoi, pour finir cette partie expérience, je vais vous présenter une expérience que j'ai développé dans l'objectif d'aller plus loin dans la compréhension des algorithmes multi-objectif pour l'exploration des types de solution.

C. Pour aller plus loin en multicritères : La grille du jeu d'échec

Objectifs

L'objectif de cette expérience est d'utiliser l'algorithme génétique multi-objectifs Octopus dans le cadre d'une exploration en phase de conception architecturale dont on ne connait pas le nombre de type de solutions. Autrement dit, dans une expérience où l'on ne connait pas d'avance les types de solutions car il n'y a pas de solution évidente. Cela permettra de compléter mon analyse sur la capacité d'exploration des algorithmes multi-objectifs.

Pour cela j'ai choisi de configurer une expérience ou les deux critères objectifs sont contradictoires pour avoir un front de Pareto intéressant mais aussi qu'ils n'aient aucun rapport entre eux pour que les solutions ne soient pas évidentes.

Configuration du modèle

L'idée de cette expérience est de travailler sur l'implantation de bâtiments en plan masse sur une grille de huit par huit suivant des règles définis par l'architecte. La surface de bâti correspond à 7 unités qui sont représentées par des carrées blancs dans la grille.

Première objectif

Le premier objectif correspond à une règle de voisinage entre les las cases bâties blanches. Pour conserver une bonne illumination de la façade ainsi qu'une bonne aération, chaque case est pondérée en fonction de son nombre d'angles partagés avec les voisins. La règle est illustrée ci-dessous, figure 24.

En fonction du nombre d'angles que la case partage avec les autres cases voisines, un score est attribué. Moins elle possède d'angles en commun plus son score est élevé comme on peut le voir dans le tableau de la règle. J'ai créé l'algorithme de tel sorte qu'il soit impossible d'avoir deux cases blanches qui se superposent.

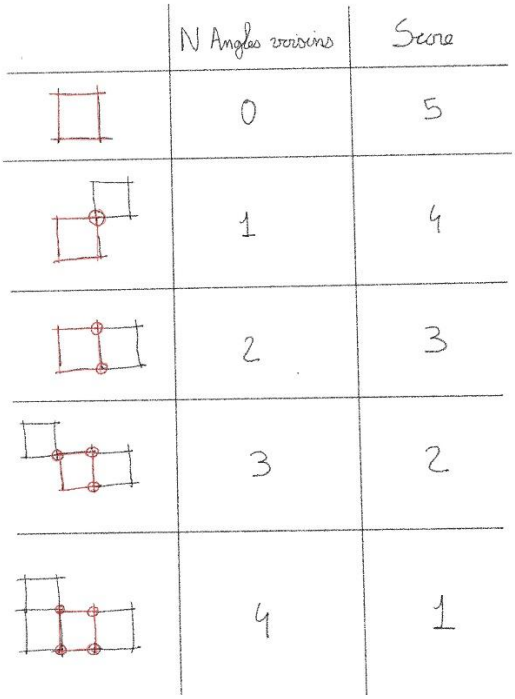

Figure 24 : Schéma explicatif de la règle de voisinage

Octopus vise à maximiser le score de chaque case en minimisant le nombre d'angles en commun. Je tiens à préciser qu'une multitude de règles de voisinage sont applicables et que j'ai choisi celle-ci de manière arbitraire pour tester la réponse algorithmique. On pourrait imaginer que l'architecte a des attentes plus précises sur les règles de voisinages qu'il souhaite mettre en place. Par exemple, en favorisant des systèmes d'assemblage entre voisins en leur donnant un meilleur score.

Pour aller plus loin dans cette expérience, on aurait pu imaginer autoriser les superpositions de cases blanches et considérer que cela correspond à un nombre d'étages. Cela aurait donné une proposition a trois dimensions qui peut être représentée en volume dans Rhinoceros. A partir de règles simples comme celle du voisinage que je viens de présenter, on peut obtenir des modèles complexes.

Il existe un jeu appelé **le Jeu de la Vie**, créé par John Conway, illustré ci-dessous, figure 25, dont le principe est de simuler la naissance et la mort des cellules sur une grille à deux dimensions. À chaque itération, les cellules peuvent naître ou disparaître en fonction de règles simples liées à leur environnement immédiat. Ce modèle m'a inspiré pour cette seconde expérience, car il illustre parfaitement comment des systèmes complexes peuvent émerger de conditions initiales élémentaires. En reprenant cette logique, mon expérience vise à démontrer comment des solutions variées et pertinentes peuvent naître à partir de règles et de contraintes bien définies, exprimant ainsi la richesse de l'exploration algorithmique de solutions en architecture.

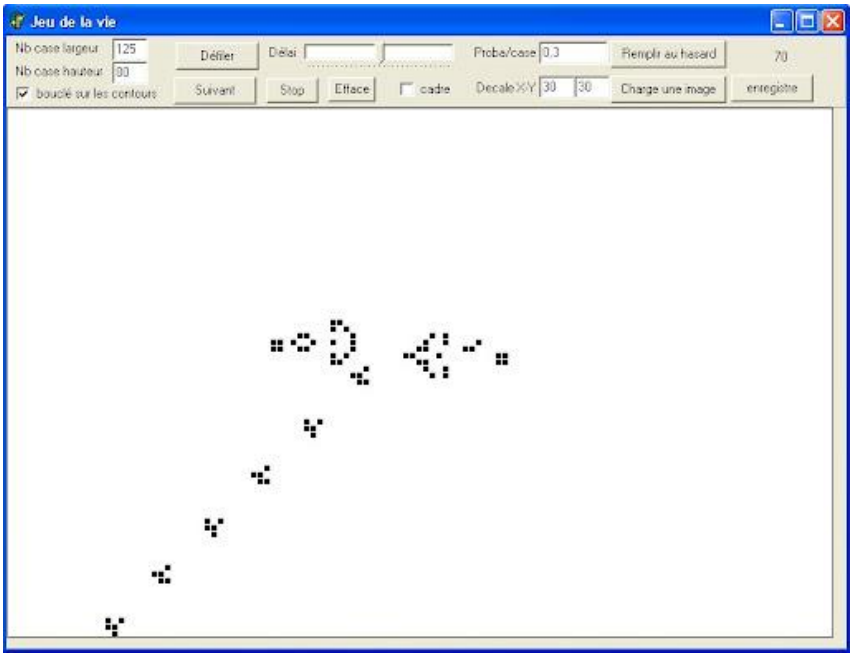

Figure 25 : L'interface du jeu de la vie

Second objectif

Le second objectif correspond à la dispersion des cases blanches. Autrement dit, si les cases sont contenues dans un cercle de petite taille, elles sont compactes en revanche, si elles sont contenues dans un cercle de grande taille, elles sont dispersées.

Le but de cet objectif est de rendre le système de cases blanches le plus compacte possible sachant que cet objectif va en opposition avec celui d'attribuer un score élevé aux blanches qui ont peu de voisins.

Pour réaliser cet objectif dans le modèle algorithmique j'ai réalisé un cercle qui contient tous les centres des cases blanches et qui a pour centre la position moyenne de toutes les cases blanches et pour rayon la plus grande distance entre le ce centre la et le centre de la case le plus éloigné. Octopus a pour objectif de minimiser la taille de ce cercle.

Présentation de l'algorithme

Voici ci-dessous, figure 26, l'algorithme sur l'interface Grasshopper de l'expérience de la grille.

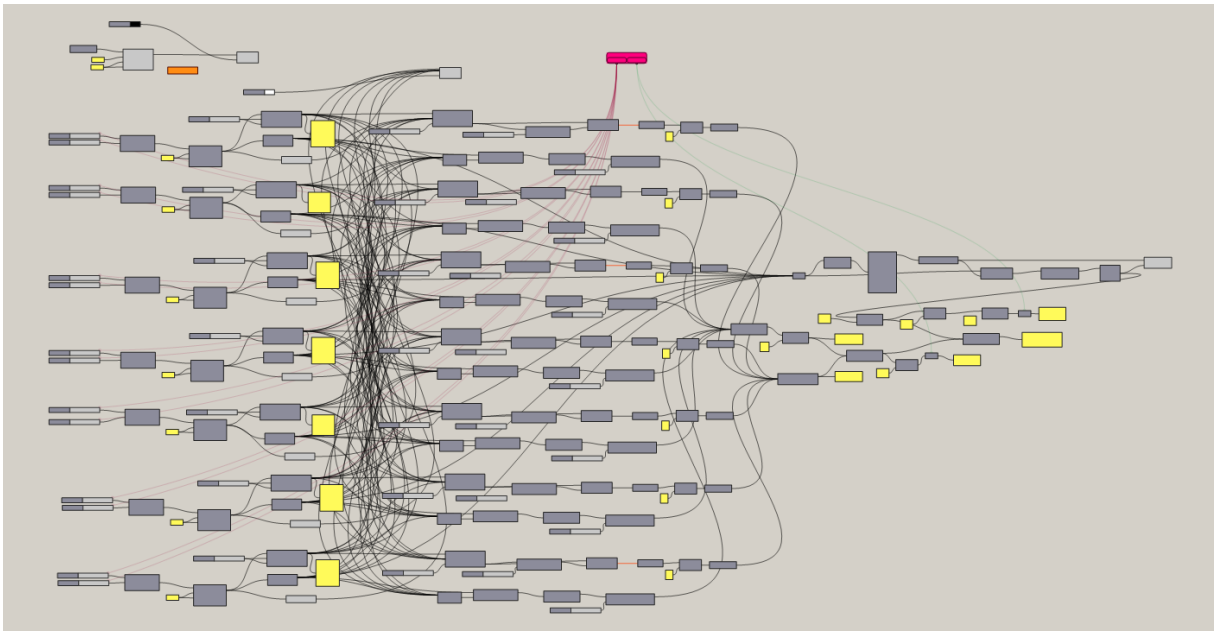

Figure 26 : Algorithme de l'expérience de la Grille

Je ne vais pas développer l'écriture de l'algorithme, mais j'aimerais montrer que l'on retrouve de la même manière que pour l'algorithme précédent une configuration en deux parties : la mise en place des éléments et l'évaluation des critères objectifs.

Analyse des résultats

Dans cette expérience Octopus a pour objectif de maximiser le score de la règle de voisinage et de contenir les cases blanches. Cela se traduit dans le graphique des résultats d'Octopus par en abscisse l'objectif lié à la règle de voisinage et en ordonnée l'objectif lié à la règle de dispersion comme présentée ci-dessous.

Les paramètres d'Octopus utilisés sont les mêmes que pour l'expérience précédente. Je rappelle qu'ils ont été choisis pour favoriser l'exploration des types de solutions.

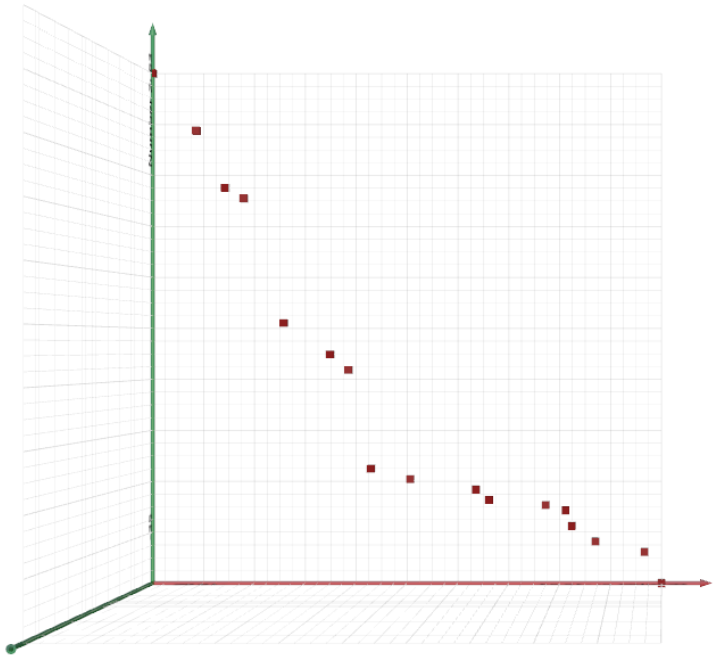

Figure 27 : Graphique d'Octopus de représentation des résultats

Dans ce graphique, nous pouvons observer le front de Pareto, qui correspond aux meilleurs compromis entre les deux objectifs. Je n'ai pas utilisé l'option Parameter Diversity dans cette expérience car l'objectif n'était pas de faire une analyse aussi poussée que dans l'expérience précédente. En effet pour gagner du temps dans l'analyse, je me suis concentré sur les résultats obtenus sur le front de Pareto en deux dimensions comme vous pouvez le voir dans le graphique. Pour aller plus loin, une analyse plus complète de l'ensemble des types de solutions serait intéressante avec cet outil.

A partir des solutions de ce graphique, j'ai récupéré 9 solutions parmi celles proposées qui me semble pertinentes pour montrer différents types de solutions obtenues par Octopus. Ces solutions sont rangées dans un ordre précis : de la plus compacte a la plus dispersée qui correspond à un parcours des solutions en partant d'en haut à gauche à en bas à droite de l'illustration du graphique, figure 28.

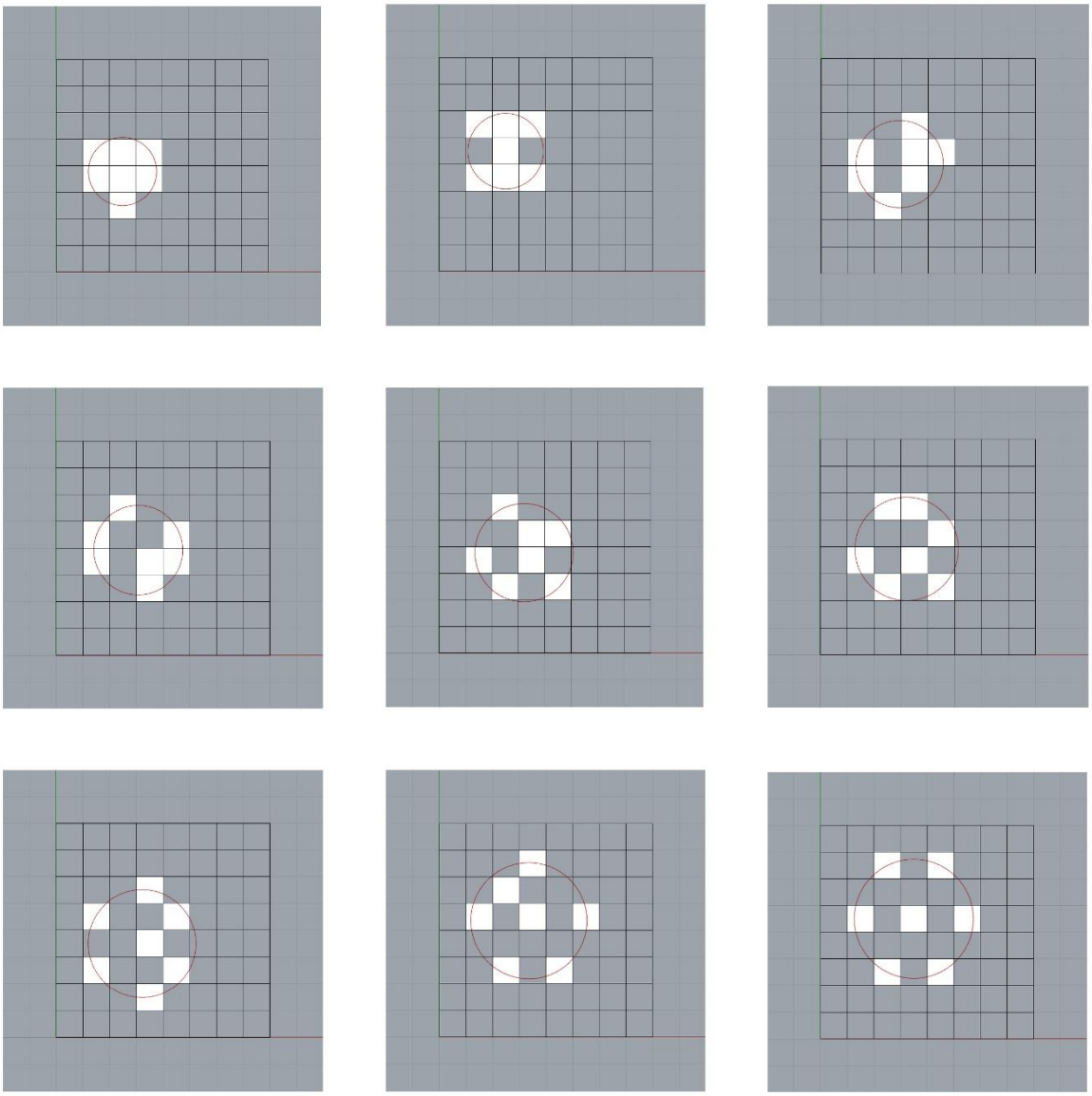

Figure 28 : Extrait des solutions issues du front de Pareto

J’ai choisi d’afficher ces solutions car elles sont pour moi représentatives de l’ensemble des solutions rencontrées. Cette matrice de solutions permet de se rendre compte de la diversité des solutions obtenues. De la première solution choisie, la plus compacte, à la dernière, la plus dispersée, une graduation de la dispersion des cases blanches s’effectue, avec le respect de la règle de voisinage qui vient offrir des assemblages intéressants. La richesse de ces assemblages intermédiaires représente tout l’intérêt de l’exploration des algorithmes génétiques multicritères. Ainsi la première solution et la dernière sont les plus évidentes à trouver alors que celles qui proviennent d’un compromis entre les deux critères ont des assemblages intéressants à analyser.

Dans cette expérience, il semble exister un très grand nombre de possibilités. En effet, le nombre de combinaisons possibles pour choisir 7 cases blanches parmi 64 cases possibles est 621 216 192. Ce nombre de possibilités d’assemblages montre bien l’incapacité pour l’architecte de parcourir toutes les solutions possibles manuellement. De la même manière, Il semble donc difficile de savoir si l’ensemble des types de solutions a été parcouru par l’algorithme. Malgré cela, les 17 solutions obtenues sur le front de Pareto dans le graphique précédent donnent une quantité raisonnable de solutions prometteuses à l’architecte. Même si l’algorithme ne parcourt pas entièrement l’ensemble des solutions, il peut analyser un grand nombre de solutions rapidement et les retranscrire simplement pour passer d’une quantité traitable seulement à l’ordinateur à une quantité traitable par un humain par le biais du front de Pareto.

En effet cet inventaire de solutions offre une source de possibilités à l’architecte dans la réflexion sur son assemblage d’espace bâti. Cela permet à l’architecte d’utiliser des types d’implantations qui favorisent la densité et l’éclairement. La variation progressive de l’assemblage des cases blanches dans l’illustration ci-dessus permet une variété de choix importante pour choisir exactement ce qui correspond au besoin du projet de l’architecte.

V. Conclusion

Conclusion sur cette méthode d'exploration

Pour réussir cette méthode d'exploration, il est essentiel de définir avec précision les paramètres, les contraintes et les objectifs qui structureront le modèle numérique. La qualité des réponses générées par l'algorithme dépend directement de ces choix. Il est également crucial de trouver un équilibre en offrant suffisamment de liberté à l'outil pour explorer des solutions intéressantes, tout en évitant de le submerger par un éventail excessif de possibilités.

Dans cette phase d'exploration, les algorithmes multi-objectifs se révèlent plus adaptés que les algorithmes mono-objectif. En effet, ils permettent de générer des compromis pertinents entre différents objectifs, enrichissant ainsi le processus de conception. De plus, leur représentation graphique des solutions, souvent sous forme de diagrammes, simplifie le travail de sélection pour l'architecte, rendant les résultats plus exploitables et visuellement clairs.

Pistes pour le futur

Une des pistes d'amélioration pour le futur est l'intégration de critères subjectifs, comme l'esthétique ou la symbolique, au sein du processus algorithmique. Ces aspects, difficilement quantifiables, pourraient être pris en compte grâce aux réseaux de neurones. Ces derniers, en étant entraînés sur des ensembles de données architecturales incluant des évaluations humaines, pourraient apprendre à reconnaître et à prioriser des qualités subjectives dans les solutions générées.

Cette intégration pourrait transformer la manière dont les algorithmes participent au processus de création en architecture. Par exemple, les réseaux de neurones pourraient identifier des tendances esthétiques en fonction de contextes culturels ou historiques spécifiques, ou encore proposer des solutions équilibrant harmonie visuelle et contraintes techniques. En interagissant de manière itérative avec l'architecte, ces outils offriraient non seulement des solutions optimisées mais aussi enrichies d'une dimension qualitative, laissant à l'architecte la possibilité d'affiner ou de réorienter les propositions selon ses intentions. Cela permettrait de dépasser les limites actuelles des algorithmes génétiques, souvent cantonnés à des critères purement quantitatifs, pour ouvrir la voie à une conception véritablement hybride.

Bibliographie

Esmaeilian Toussi, H. (2020). *L'application des approches évolutionnaires, génératives et hybrides dans l'optimisation du design en architecture*. *Journal de la Faculté d'Architecture*, *2*(2), 1-20. Récupéré du Département d'Architecture, Université NEU.

Marin, P. (2010). *Exploration des mécanismes évolutionnaires appliqués à la conception architecturale : mise en œuvre d'un algorithme génétique guidé par les qualités solaires passives de l'enveloppe*. Thèse de doctorat, Institut National Polytechnique de Lorraine, École Nationale Supérieure d'Architecture de Nancy.

Khabazi, Z. (2012). *Generative Algorithms : using Grasshopper*. Livre publié pour l'apprentissage des techniques de conception algorithmique à l'aide de Grasshopper, Morphogenesism.

Terzidis, K. (2006). *Algorithmic Architecture*. Livre, Architectural Press, Elsevier. Ce livre explore les aspects théoriques et philosophiques de l'architecture algorithmique, en discutant le rôle des algorithmes dans le processus de conception et la relation entre l'humain et l'ordinateur dans le domaine de l'architecture.

Terzidis, K. (2015). *Permutation Design : Buildings, Texts, and Contexts*. Livre, Routledge. Cet ouvrage explore les théories, techniques et exemples de la conception par permutation dans le design, en mettant l'accent sur l'utilisation de la puissance computationnelle pour analyser et optimiser les solutions architecturales.

Dissaux, T. (2017). *Optimisation en conception architecturale : Les alternatives aux algorithmes génétiques*. Travail de fin d'études, Université de Liège, Faculté d'Architecture, sous la direction de Sylvie Jancart.

Barreto, G. (2018). *Generative Design for Building Information Modeling*. Résumé étendu, Instituto Superior Técnico. Ce document explore l'application de la conception générative au BIM, en utilisant l'environnement Rosetta et ArchiCAD.

Rohrmann, J. (2019). *Design Optimization in Early Project Stages : A Generative Design Approach to Project Development*. Thèse de Master, iniversité Technique de Munich, Faculté de Génie Civil, Géosciences et Ingénierie Environnementale.

Couwenbergh, J.-P., & Gallas, M.-A. (2021). *Conception paramétrique avec Rhino et Grasshopper : Applications en architecture, ingénierie et design*. Livre, Éditions Eyrolles. Cet ouvrage présente les bases de la conception paramétrique, les outils Rhino/Grasshopper, et des études de cas pour illustrer les applications pratiques.

Stals, A., Elsen, C., & Jancart, S. (2016). *Ruptures et démesures de l'architecture non standard à l'ère du numérique : la paramétrisation comme outil de réconciliation*. Article scientifique, Université de Liège.

Ben Abdallah, Y. (2017). *Conception architecturale et modélisation paramétrique*. Mémoire, École Nationale Supérieure d'Architecture de Toulouse.

De Beusscher, G., & Rogeau, N. (2017). *Conception paramétrique de structures architecturales en bois drone-compatibles*. Mémoire, Université Catholique de Louvain, École Polytechnique de Louvain.

Hesselgren, L., Kilian, A., Malek, S., Olsson, K.-G., & Sorkine-Hornung, O. (2018). *Advances in Architectural Geometry*. Actes de conférence, Klein Publishing GmbH. Cet ouvrage rassemble les contributions sur les techniques computationnelles et géométriques dans l'architecture, présentées lors de la conférence AAG 2018.

Fabbri, R., Jiricna, E., Palazetti, C., & Wells, M. (2017). *The Miles Stair in Somerset House*. Actes du symposium international sur les BFUP, Montpellier, France. Cet article traite de la conception et de la réalisation de l'escalier hélicoïdal en BFUP au Somerset House.

Table des figures