Recherche arborescente Monte Carlo et évolution artificielle pour l'identification de paramètres dynamiques dans les réseaux de régulation génétiques hybrides

Romain Michelucci

Recherche arborescente Monte Carlo et évolution artificielle pour l'identification de paramètres dynamiques dans les réseaux de régulation génétiques hybrides

Romain MICHELUCCI

Laboratoire d'Informatique, Signaux et Systèmes de Sophia-Antipolis (i3S)
UMR 7271 - Université Côte d'Azur - CNRS

Présentée en vue de l'obtention du grade de docteur en Informatique **d'**Université Côte d'Azur

Dirigée par : Jean-Paul Comet
Co-encadrée par : Denis Pallez
Soutenue le : lundi 16 décembre 2024

Devant le jury, composé de :
Tristan Cazenave, PR, Univ. Paris-Dauphine
Jean-Paul Comet, PR, Univ. Côte d'Azur
Olivier Goudet, MCF, Univ. d'Angers
Nicolas Jouandeau, PR, Univ. Paris 8
Pierrick Legrand, PR, Univ. de Bordeaux
Évelyne Lutton, DR, Univ. Paris-Saclay
Denis Pallez, MCF, Univ. Côte d'Azur

i3S sophia antipolis

Recherche arborescente Monte Carlo et évolution artificielle pour l'identification de paramètres dynamiques dans les réseaux de régulation génétiques hybrides

Jury :

Président du jury

Pierrick Legrand	Professeur	Université de Bordeaux

Rapporteurs

Évelyne Lutton	Directrice de Recherche	Université Paris-Saclay
Nicolas Jouandeau	Professeur	Université Paris 8

Examinateurs

Tristan Cazenave	Professeur	Université Paris-Dauphine
Olivier Goudet	Maître de Conférences	Université d'Angers

Directeur de thèse

Jean-Paul Comet	Professeur	Université Côte d'Azur

Co-encadrant de thèse

Denis Pallez	Maître de Conférences	Université Côte d'Azur

Recherche arborescente Monte Carlo et évolution artificielle pour l'identification de paramètres dynamiques dans les réseaux de régulation génétiques hybrides

La modélisation des systèmes biologiques a pour but d'aider à la compréhension des systèmes biologiques via une représentation de la dynamique de ces systèmes qui peut, elle-même, être étudiée de manière informatique. Le goulot d'étranglement de la démarche de modélisation est la recherche de paramétrisations de modèles qui sont en accord avec les connaissances biologiques disponibles. Ici, nous adressons ce problème dans le cadre de la modélisation hybride des réseaux de régulation génétiques.
Dans cette thèse, nous commençons par formuler le problème en un problème d'optimisation afin d'exhiber une paramétrisation adéquate à l'aide de l'évolution artificielle. Dans un second temps, nous le modélisons comme un problème de décision séquentiel, représenté par un processus de décision markovien avec des états et des actions continus. Pour résoudre ce dernier, nous employons des algorithmes de recherche arborescente Monte Carlo (MCTS) et proposons des améliorations.
Dans la modélisation de systèmes biologiques complexes, il est souvent insuffisant de se limiter à une seule paramétrisation, car l'évolution des connaissances biologiques peut rapidement rendre le modèle obsolète. Une approche exploratoire de modélisation exige donc de produire un ensemble de solutions diverses permettant au modélisateur d'appréhender la diversité des comportements potentiels issus des données biologiques disponibles. Cependant, les heuristiques stochastiques ne garantissent pas l'obtention de solutions distinctes lors d'exécutions multiples du même algorithme. C'est pourquoi nous avons mis au point des mécanismes spécifiques visant à extraire, en une seule exécution, un échantillon de solutions différentes. Pour cela, nous avons d'abord exploré l'optimisation multimodale, notamment à l'aide d'algorithmes génétiques cellulaires, puis nous avons appliqué des stratégies de planification diversifiée avec MCTS, en proposant de nouvelles heuristiques de recherche.

Mots clés : recherche arborescente Monte Carlo, évolution artificielle, apprentissage par renforcement, optimisation stochastique, heuristiques, réseaux de régulation génétiques

Monte Carlo tree search and artificial evolution for identifying dynamic parameters in hybrid gene regulatory networks

The aim of biological systems modelling is to help us understand biological systems through a representation of the dynamics of these systems, which can, in turn, be studied from a computer science point of view. The bottleneck in the modelling approach is the search for model parametrisations that are consistent with the available biological knowledge. Here, we address this problem in the context of hybrid modelling of gene regulatory networks.
In this thesis, we first formulate the problem as an optimisation problem in order to exhibit an adequate parametrisation using artificial evolution. In a second step, we model it as a sequential decision problem, represented by a Markov decision process with continuous states and actions. To solve the latter, we employ Monte Carlo Tree Search (MCTS) algorithms and propose improvements.
When modelling complex biological systems, it is often insufficient to limit oneself to a single parametrisation, as the evolution of biological knowledge can quickly render the model obsolete. An exploratory modelling approach therefore requires producing a set of diverse solutions, allowing the modeller to grasp the diversity of potential behaviours derived from the available biological data. However, stochastic heuristics do not guarantee that distinct solutions will be obtained from multiple runs of the same algorithm. We have therefore developed specific mechanisms to extract a sample of different solutions in a single run. To do this, we first explored multimodal optimisation, in particular using cellular genetic algorithms, and then applied diverse planning strategies with MCTS, proposing new search heuristics.

Keywords: Monte Carlo tree search, artificial evolution, reinforcement learning, stochastic optimization, heuristics, gene regulatory networks

Remerciements

Bien qu'il s'agisse d'une tautologie, il est tout de même important de souligner que cette thèse est le fruit d'un travail collectif. Je tiens donc à exprimer ma gratitude envers toutes les personnes qui m'ont soutenu tout au long de cette aventure académique, tant sur le plan professionnel que personnel.

Tout d'abord, je tiens à remercier les membres du jury, Tristan Cazenave, Olivier Goudet et Pierrick Legrand, ainsi que les rapporteurs de ce manuscrit, Nicolas Jouandeau et Evelyne Lutton, pour l'intérêt qu'ils ont porté à ces travaux et pour leur bienveillance.

Je remercie ensuite chaleureusement mon directeur de thèse Jean-Paul Comet et mon co-encadrant Denis Pallez pour leur expertise, leurs innombrables relectures et leurs conseils. J'ai apprécié la liberté et l'autonomie qu'ils m'ont laissé acquérir au cours de ces années de collaboration. Je tiens aussi à saluer et souligner la contribution de Tristan Cazenave dans cette collaboration.

Sans l'aide des différents financeurs, ce travail n'aurait pu être mené à son terme. Je remercie donc le laboratoire i3S pour (i) m'avoir octroyé un complément de thèse d'une durée de deux mois afin que je puisse terminer le manuscrit et préparer ma soutenance sereinement et (ii) la mise à disposition des ressources pour le calcul scientifique. À ce titre, j'ai pu utiliser le serveur de calcul `Calcul` et je remercie Florent Jaillet pour son assistance technique.

Ce travail a aussi bénéficié d'une aide du gouvernement français, gérée par l'Agence Nationale de la Recherche au titre du Plan d'investissement France 2030, dans le cadre de l'Initiative d'Excellence d'Université Côte d'Azur portant la référence ANR-15-IDEX-01, à travers l'accès aux moyens informatiques du Centre de calculs d'Université Côte d'Azur (infrastructure OPAL). Je remercie Maeva Antoine pour son aide technique.

Je suis reconnaissant envers Pierrick Legrand et Madalena Chaves pour leur implication dans le comité de suivi individuel.

Je souhaite aussi remercier particulièrement Hélène Collavizza, Gilles Bernot, Jean-Paul Comet et Nadia Abchiche-Mimouni de m'avoir fait confiance pour les enseignements. Ce mentorat a été important pour développer mes compétences d'enseignement.

Je remercie également chacun des membres permanents de l'i3S avec qui j'ai pu échanger et en particulier ceux que j'ai eus en tant qu'enseignants : Cinzia Di Giusto, Marie Pelleau, Bruno Martin, Sandrine Julia, Jean-Charles Régin, Elisabetta De Maria et Enrico Formenti, merci d'avoir contribué à l'avancement de mes études de manière pédagogique.

J'ai aussi eu la chance de faire de très belles rencontres durant ces trois années et je tiens à saluer l'importance de ces personnes dans cette aventure. En commençant par la génération des « anciens », je remercie Amélie, les deux Laetitia et Sara d'avoir veillé à ce que mon intégration se fasse dans les meilleures conditions possibles. Merci à (François, Loïc), cette amitié immuable que vous partagez est belle à voir. Merci à Marie, le rayon de soleil de l'i3S, qui a toujours eu le mot pour faire sourire (et bisous à Marius). Merci à Julie, à nous les fonds de l'adistic ! Et un grand merci à Oussama pour ces moments partagés au sein et à l'extérieur du laboratoire.
Dans la génération des « tout-nouveaux-docteurs », je tiens à remercier Thomas et Florian, avec qui j'ai apprécié discuter, et Aymeric pour m'avoir fait, notamment, découvrir les échecs alternatifs.
Pour finir, je remercie la génération des « bientôt-docteurs » : Baptiste, Guillaume, Margaux, Thomas, Valentin et Wessim pour leur énergie débordante. Je souhaite adresser un remerciement tout particulier à Juliette (et sa positivité légendaire) et à Steve : j'ai apprécié chacune de nos conversations.
Plus généralement, je remercie chaleureusement chacune des personnes du laboratoire avec qui j'ai eu des discussions amicales et des rires francs, que ce soit pendant la pause déjeuner ou les pauses cafés.

Je ne saurais trop remercier Magali Richir pour son efficacité, sa sympathie et sa bonne humeur en toute circonstance. Merci pour son aide administrative et les nombreux fous rires. Un grand merci aussi à Pierre Alfarroba pour sa bonhomie et son énergie débordante.

D'un point de vue plus personnel, je tiens à exprimer ma profonde gratitude à mes amis et à ma famille. Merci à ces derniers de m'avoir offert le luxe de faire de longues études. Je souhaite remercier tout particulièrement mon père et ma grand-mère d'avoir toujours été à l'écoute pendant les nombreux moments de doute.

Finalement, je voudrais exprimer ma sincère reconnaissance à Nina. C'est sans aucun doute grâce à ses conseils, ses encouragements, son soutien et sa présence au quotidien que cette thèse a débuté, s'est déroulée et s'est achevée de la plus belle des manières.

Table des matières

Introduction **11**

I Modélisation de RRGHs 17

1 Modélisation de réseaux de régulation génétiques **19**
1.1 Modélisation . . . 20
1.1.1 Caractéristiques générales . . . 20
1.1.2 Cadres de modélisation . . . 21
1.2 Cadre utilisé . . . 23
1.2.1 Du formalisme booléen à celui de René Thomas . . . 23
1.2.2 Cadre de René Thomas hybride . . . 27
1.3 Instances : réseaux de régulation génétiques hybrides étudiés . . . 35
1.3.1 Cycle négatif (2G) . . . 35
1.3.2 Cycle circadien (3G) . . . 37
1.3.3 Cycle cellulaire (5G) . . . 38
1.4 Présentation du simulateur . . . 40
1.5 Synthèse . . . 41

2 Connaissance biologique basée sur la logique de Hoare **43**
2.1 Vérification de programme . . . 44
2.2 Logique de Hoare génétiquement modifiée pour le cadre de modélisation hybride . . . 45
2.2.1 Langage de propriétés . . . 46
2.2.2 Langage de chemins . . . 48
2.2.3 Triplet de Hoare hybride . . . 50
2.3 Connaissance biologique utilisée . . . 51
2.3.1 Cycle négatif (2G) . . . 51
2.3.2 Cycle circadien (3G) . . . 52
2.3.3 Cycle cellulaire (5G) . . . 53
2.4 Première approche par résolution de contraintes . . . 54
2.4.1 Extraction de contraintes avec la plus faible précondition hybride . . . 54
2.4.2 Problème de satisfaction de contraintes . . . 54
2.4.3 Cas d'étude du solveur continu AbSolute. . . . 56
2.5 Synthèse . . . 57

II Recherche d'une paramétrisation 59

3 Problème d'optimisation en boîte noire avec EA 61
3.1 Reformulation du CSP en un problème d'optimisation numérique 62
3.1.1 Introduction à l'optimisation numérique 62
3.1.2 Formulation du problème d'optimisation 65
3.1.3 Choix de la technique d'optimisation 74
3.2 État de l'art 75
3.2.1 Analogie entre la théorie de l'évolution et le domaine de l'évolution artificielle 76
3.2.2 Structure générale 77
3.2.3 Métaheuristiques classiques 78
3.3 Résolution du problème d'optimisation 84
3.3.1 Prérequis des expérimentations 84
3.3.2 Étude expérimentale 86
3.4 Passage à l'échelle 94
3.4.1 État de l'art de la coévolution coopérative 95
3.4.2 Contribution 100
3.4.3 Étude expérimentale 101
3.5 Synthèse 106

4 Problème de décision séquentiel avec MCTS 107
4.1 D'un problème d'optimisation à un problème de décision séquentiel 108
4.1.1 Introduction aux problèmes de décision séquentiels 108
4.1.2 Formulation du PDS 114
4.1.3 Fonctions valeur et équations de Bellman 117
4.2 État de l'art recherche arborescente Monte Carlo 118
4.2.1 Recherche basée sur les simulations 118
4.2.2 Recherche arborescente Monte Carlo : cadre combinatoire 120
4.2.3 Améliorations 125
4.2.4 Recherche arborescente Monte Carlo : cadre continu 129
4.3 Contributions 132
4.3.1 GRAVE dans le domaine continu (cGRAVE) 133
4.3.2 Stratégie de décomposition des actions 134
4.3.3 Politique sélective basée sur les contraintes 135
4.4 Étude expérimentale 136
4.4.1 Plan d'expérimentation 136
4.4.2 Module CSP pour les paramètres dynamiques 137
4.4.3 Évaluation d'une simulation et construction de l'arbre de recherche. 138
4.4.4 Analyse des résultats. 138
4.4.5 Analyse statistique 140
4.4.6 Visualisation des résultats 141
4.5 Synthèse 142

III Recherche d'un ensemble de paramétrisations diverses 143

5 Optimisation multimodale 145
5.1 État de l'art . . . 146
5.1.1 Multimodalité . . . 146
5.1.2 Techniques employées . . . 147
5.1.3 Mesures de performance . . . 150
5.2 Investigation de deux stratégies de *niching* . . . 152
5.2.1 RS-CMSA-ESII . . . 153
5.2.2 Algorithmes cellulaires génétiques . . . 155
5.3 Étude expérimentale . . . 158
5.3.1 Expérimentations . . . 159
5.3.2 Résultats . . . 159
5.3.3 Analyse statistique . . . 162
5.3.4 Visualisation . . . 163
5.3.5 Conclusion de l'étude expérimentale . . . 164
5.4 Originalité du problème multimodal . . . 165
5.4.1 Une hypothèse . . . 165
5.4.2 Une proposition de solution . . . 167
5.4.3 Des exemples de fonctions de benchmarks . . . 168
5.5 Synthèse . . . 169

6 Planification diversifiée 171
6.1 État de l'art . . . 172
6.1.1 Planification top-k, top-qualité ou diversifiée . . . 172
6.1.2 Planification diversifiée avec MCTS . . . 174
6.2 Contributions . . . 178
6.2.1 Stratégies d'inhibition . . . 178
6.2.2 Diversité comme objectif dans MCTS multi-objectif . . . 179
6.3 Validation expérimentale . . . 181
6.3.1 Cadre expérimental . . . 181
6.3.2 Analyse des résultats . . . 182
6.3.3 Visualisation . . . 184
6.4 Synthèse . . . 186

Conclusion 187

Introduction

La majorité des fonctions de notre corps humain sont modulées par les rythmes biologiques. Les réponses de notre organisme à une perturbation environnementale sont par exemple liées à l'horloge circadienne (Zhang et al. 2014). Cette dernière est notamment impliquée dans la régulation de notre métabolisme, de notre physiologie, de notre comportement, de nos habitudes de sommeil, ainsi que de notre production d'hormones (Kaur et Bala 2013). En fonction du moment de la journée, chacune des cellules de notre organisme adapte son activité notamment à ces cycles endogènes, impactant ainsi notre température corporelle ou encore notre niveau d'éveil. La chronothérapie, branche de la chronobiologie, cherche à identifier le moment optimal pour administrer un traitement ou un médicament en fonction de ces rythmes biologiques. L'objectif est de maximiser les bénéfices thérapeutiques et de diminuer les effets secondaires. Cette stratégie se développe dans de nombreux domaines pour traiter des maladies diverses. Une de ses applications majeures est la cancérologie pour l'optimisation des chimiothérapies dont on sait qu'elles s'attaquent aux cellules cancéreuses, mais aussi aux cellules saines (Lévi 2006). Le but est alors de trouver la bonne fenêtre temporelle pendant laquelle le traitement élimine plus efficacement les cellules cancéreuses tout en minimisant les effets secondaires, c'est-à-dire en préservant les cellules saines. On peut aussi citer l'utilité de la chronothérapie dans plusieurs maladies comme l'asthme (Pincus, Beam et Martin 1995), l'épilepsie (D. A. Stanley, Talathi et Carney 2014), et les maladies cardiovasculaires (Smolensky 1996) avec des applications spécifiques au traitement de l'hypertension (Latha et al. 2010). L'effet de ces thérapies médicamenteuses en fonction de l'horaire de la prise varie d'un patient à l'autre et dépend des spécificités génétiques et physiologiques.

Dans le développement de ces thérapies, les formes de recherche *in vivo*, comme l'expérimentation animale et les essais cliniques, font office de référence. Toutefois, les méthodes *in silico*, désignant une forme de recherche effectuée au moyen de calculs informatiques, comme la simulation de données synthétiques, permettent d'apporter des résultats complémentaires. Elles ne cherchent pas à les remplacer, mais bien à venir compléter les informations obtenues lors des expériences humides en laboratoire. Quelques-uns des exemples reposent sur l'utilisation de méthodes d'apprentissage automatique. Ces dernières ont par exemple permis d'identifier un nouveau gène, composant fonctionnel de l'horloge circadienne (Anafi et al. 2014), ou encore de développer une méthode de détection pour comprendre comment l'expression des gènes de l'horloge circadienne est affectée par di-

verses perturbations (ici, les cycles lumière-obscurité et sommeil-éveil) (Hughey 2017). L'étude de la dynamique des systèmes et des phénomènes biologiques n'est pas en reste. La modélisation de ces systèmes complexes vise à donner un cadre synthétique et prédictif à des données ou des expériences, comme présenté dans (Leloup et Goldbeter 2008) et permet, entre autres, d'identifier de nouveaux mécanismes moléculaires entre les cycles circadiens et cellulaires (Gotoh et al. 2016). Les travaux de (Cornillon 2017) présentent les systèmes biologiques comme un ensemble d'entités biologiques qui interagissent pour assurer une fonction biologique. Ces interactions mènent à l'émergence de dynamiques. Plus un système est composé d'un grand nombre d'entités et d'interactions hétérogènes, plus ces dynamiques sont complexes à extraire et à étudier. La modélisation intervient alors comme un facilitateur permettant, d'une part, une meilleure compréhension du système et de ses dynamiques, et d'autre part, une meilleure prédiction de comportements inconnus.

Dans cette thèse, nous nous penchons spécifiquement sur l'étude d'une sous-classe de systèmes biologiques : les réseaux de régulation génétiques (RRGs). Ces réseaux décrivent des mécanismes biologiques entre différentes entités comme des protéines, des gènes, des influences environnementales, *etc.* Les entités d'un RRG interagissent entre elles à travers des interactions d'activation ou d'inhibition. Ces interactions peuvent représenter soit des réactions de synthèse, soit des réactions de dégradation des entités cibles. L'étude de la dynamique de l'évolution de ces réactions nous intéresse particulièrement.

Du point de vue de l'informatique, un tel réseau est représenté par un graphe d'interaction composé de nœuds représentant une abstraction d'une ou plusieurs entités biologiques qui agissent ensemble, que nous nommons variables, et d'arcs correspondant aux régulations entre ces variables. La modélisation de systèmes biologiques sous la forme de graphe d'interaction permet d'agréger certaines données expérimentales, mais ne permet pas de retranscrire leur évolution, leur dynamique au cours du temps.

C'est dans ce contexte que la modélisation joue un rôle. Le modélisateur choisit de travailler avec un cadre de modélisation qui a pour rôle de représenter la dynamique des variables du graphe d'interaction. Le choix d'un cadre de modélisation adéquat relève de la disponibilité et de la qualité des informations biologiques ainsi que du niveau d'abstraction attendu. Plusieurs cadres de modélisation existent. Un cadre de modélisation stochastique, par exemple, cherche à intégrer la variabilité des résultats des expériences biologiques dans la modélisation du système afin d'obtenir un modèle le plus fidèle possible de la réalité. À l'inverse, un cadre discret, générant des dynamiques représentées sous forme de transitions entre des états du système, est préféré si le modélisateur s'intéresse aux caractéristiques fondamentales du réseau comme, par exemple, l'adaptabilité du réseau à des changements environnementaux qui impactent l'expression de certaines variables.

Quel que soit le cadre choisi, un modèle du réseau de régulation génétique rend compte d'une dynamique gouvernée par des paramètres dynamiques contrôlant l'évolution des variables. L'identification de ces paramètres est une problématique générale en modélisation. Être capable d'identifier les paramètres dynamiques d'un réseau, c'est être capable de trouver des valeurs de paramètres qui mènent à l'obtention d'un modèle paramétré. Il faut s'assurer de la cohérence de ce modèle et des connaissances biologiques à disposition. On dira que le modèle est valide s'il est cohérent et qu'il reflète *a minima* les observations expérimentales. Les paramètres dynamiques qui mènent à un modèle valide génèrent chacun une dynamique qui permet de représenter un comportement du système biologique

étudié.

Ce travail s'inscrit dans la poursuite des recherches menées dans la thèse de (Jonathan Behaegel 2018). Dans ces travaux, le cadre de modélisation utilisé pour représenter un réseau de régulation génétique est un cadre hybride (RRGH) qui est une extension du cadre de René Thomas (René Thomas et d'Ari 1990) intégrant une évolution temporelle continue entre les différents événements discrets. La dynamique qui résulte d'une identification de ces paramètres dynamiques continus est une trajectoire linéaire par morceau. Pour valider ou réfuter un modèle, il est nécessaire de se fier à des connaissances biologiques. L'approche envisagée ne fait pas directement usage des données brutes obtenues à la suite d'expériences humides : elles sont agrégées et formalisées sous la forme d'une séquence de triplets. Ce choix provient de (J. Behaegel, J.-P. Comet et Folschette 2017) qui emploie des méthodes formelles pour identifier les paramètres dynamiques d'un modèle hybride. La logique de Hoare (Hoare 1969) est utilisée pour formaliser la représentation des connaissances biologiques d'un RRG. Elle a été détournée de son rôle usuel, la correction de programmes informatiques, dans le but de contraindre les paramètres dynamiques et d'automatiser leur identification. Un problème de satisfaction de contraintes (CSP pour *Constraint Satisfaction Problem*) a donc été formulé dans (J. Behaegel, J.-P. Comet et M. Pelleau 2018). La recherche de solutions de ce CSP a mené à l'utilisation de solveurs qui réduisent la taille de l'espace de recherche pour extraire un ensemble de solutions. Cependant, à mesure que le nombre de variables et d'interactions dans le RRG augmente, le nombre de paramètres dynamiques et le nombre de contraintes augmentent. Lorsque les graphes d'interaction comportent plus de trois nœuds, il n'a pas été possible d'intégrer les solveurs dans un processus d'automatisation complet.

Au regard de cette limitation, l'objectif de cette thèse est de développer de nouvelles méthodes d'échantillonnage de l'espace des solutions du problème d'identification des paramètres dynamiques des RRGHs, dont la connaissance biologique est formalisée sous la forme d'un triplet de Hoare. Nous illustrons nos expérimentations sur trois instances : le cycle négatif à deux variables, le cycle circadien à trois variables (Jonathan Behaegel et al. 2016) et le cycle cellulaire à cinq variables (Jonathan Behaegel 2018).

Des techniques heuristiques, d'intelligence artificielle, sont utilisées pour surmonter ce problème. Nous le considérons, dans une première approche, comme un problème d'optimisation numérique en boîte noire (Audet et Hare 2017). Nous introduisons une fonction objectif qui est une mesure de similarité entre l'exécution du modèle paramétré par un simulateur et des connaissances biologiques. Des métaheuristiques bio-inspirées du domaine de l'évolution artificielle (EA) (Eiben et J. E. Smith 2015b) sont considérées pour résoudre ce problème. Une alternative à cette approche consiste à considérer le problème comme un problème de décision séquentiel (Russell et Norvig 2016) où les paramètres dynamiques sont choisis l'un après l'autre. Dans ce contexte, nous l'abordons comme un processus de décision markovien dans lequel un agent doit agir dans un environnement (ici, le simulateur) afin de maximiser une récompense cumulative. Cette notion de récompense cumulative est déduite de la fonction objectif introduite dans l'approche d'optimisation. Nous faisons usage de la recherche arborescente Monte Carlo (MCTS) (Browne et al. 2012), comme technique d'apprentissage par renforcement (Sutton et Barto 2018), pour résoudre ce problème et proposons des contributions heuristiques.

Le modélisateur ne peut pas se contenter d'une seule paramétrisation du modèle, car les avancées des connaissances biologiques peuvent rapidement rendre ce modèle obsolète.

En effet, de nouvelles connaissances peuvent révéler des dynamiques ou des interactions jusqu'alors inconnues. Par conséquent, une approche exploratoire de la modélisation est essentielle. Cela implique de générer un éventail de solutions diverses, ce qui permet au modélisateur d'explorer différents scénarios et d'analyser la diversité des comportements potentiels qui peuvent émerger, en se référant toujours aux données biologiques disponibles.

Dans le domaine de l'optimisation, la recherche d'un ensemble de solutions au cours d'une unique exécution d'un algorithme est appelée optimisation multimodale (Preuss, M. Epitropakis et al. 2021). Il n'est pas possible de garantir qu'en relançant plusieurs fois une même exécution (avec une initialisation différente), un algorithme fournisse une solution différente. Il est donc nécessaire de développer des mécanismes de recherche alternatifs. Nous investiguons certaines métaheuristiques impliquant différents mécanismes de préservation de diversité dans le but d'identifier un ensemble de paramétrisations, chacune menant à un modèle valide. Dans le domaine de l'apprentissage par renforcement, et plus particulièrement avec MCTS, la question de la recherche d'un ensemble de solutions au cours du processus d'apprentissage a été récemment investiguée (Benke et al. 2023). Nous proposons des contributions heuristiques pour orienter la recherche vers des solutions prometteuses et diversifiées au cours d'une unique exécution.

Ce manuscrit a pour but de présenter tous les concepts et notions essentiels mis en place pour résoudre le problème d'identification d'une, puis de plusieurs, paramétrisations qui gouvernent la dynamique d'un réseau de régulation génétique hybride. Il est organisé en trois parties composées de deux chapitres chacune.

La première partie vise à présenter les travaux antérieurs dont les résultats ont abouti à la proposition de cette thèse. Elle se termine d'ailleurs par l'énoncé détaillé de la problématique.

Dans le **chapitre 1**, premier chapitre de la première partie, nous commençons par introduire la modélisation de réseaux de régulation génétiques. Dans un premier temps, nous présentons les caractéristiques générales qui composent la modélisation d'un système biologique ainsi que les différentes catégories de cadres de modélisation existants. Ensuite, nous détaillons le cadre choisi : le cadre hybride de René Thomas. Ce dernier repose sur l'approche discrète de René Thomas, dans laquelle les niveaux de concentration des variables du graphe d'interaction sont découpés et séparés par des seuils donnant lieu à des états discrets. Cette approche hybride ajoute à l'évolution entre états discrets (dynamique discrète) une évolution continue à l'intérieur des états au cours du temps (dynamique continue). En se reposant sur cette présentation de la dynamique du modèle hybride, nous détaillons les trois réseaux étudiés.

Le premier objectif du **chapitre 2** est de présenter les connaissances biologiques dont nous disposons pour résoudre le problème d'identification. Les observations expérimentales sont agrégées sous la forme d'une succession d'événements temporels et formalisées comme un triplet de Hoare. Pour atteindre cet objectif, nous proposons un bref rappel de la logique de Hoare sur les programmes impératifs et son adaptation pour les modèles discrets. Le second objectif est de détailler les limites de l'approche par résolution de contraintes dans le but de justifier l'utilisation de nouvelles méthodes pour solutionner notre problème d'identification.

La seconde partie cherche à introduire deux nouvelles formulations du problème. La première le considère comme un problème d'optimisation numérique en boîte noire et la

seconde comme un problème de décision séquentiel. Leur résolution respective est aussi présentée dans cette partie.

Le **chapitre 3**, premier chapitre de la deuxième partie, débute sur une brève introduction de l'optimisation numérique pour mener à la formulation du problème d'optimisation. Nous présentons par la suite les métaheuristiques bio-inspirées comme méthode d'optimisation en nous focalisant sur certaines d'entre elles. Une étude expérimentale est ensuite menée sur les trois réseaux étudiés précédemment introduits et une discussion sur les résultats obtenus en découle. Pour finir, nous envisageons une extension de notre approche pour gérer le problème du passage à l'échelle et permettre ainsi d'identifier les paramétrisations de réseaux avec un plus grand nombre de variables du graphe d'interaction. Pour cela, nous utilisons la technique classique de « diviser pour mieux régner » en utilisant plusieurs métaheuristiques simultanément qui doivent collaborer pour optimiser le problème (co-évolution). Nous proposons différentes stratégies de décomposition et les mettons en application à l'aide de méthodes faisant évoluer une ou plusieurs populations d'individus. Nous concluons ce chapitre en discutant des limites de cette approche.

Le **chapitre 4**, quant à lui, a pour objet la formulation du problème de décision séquentiel sur lequel nous envisageons la recherche arborescente Monte Carlo comme méthode d'apprentissage par renforcement. Nous commençons par introduire les notions relatives au processus de décision markovien, puis nous formulons, sur ce modèle, le problème de décision séquentiel à résoudre. Nous détaillons ensuite les différents variants de MCTS, d'abord dans un cadre combinatoire puis dans un cadre continu, où l'espace des états et celui des actions sont tous deux continus. Nous proposons trois heuristiques pour améliorer un de ces variants et réalisons une étude expérimentale pour démontrer l'intérêt de ces heuristiques sur notre problème.

La troisième et dernière partie a pour objectif l'identification de plusieurs paramétrisations qui mènent chacune à l'extraction d'un modèle valide au cours d'une unique exécution.

Le **chapitre 5**, premier chapitre de la troisième partie, emploie des techniques d'optimisation multimodale. Au moyen d'une étude expérimentale, nous cherchons à trouver la méthode et ses hyperparamètres les plus adéquats pour la recherche de solutions diverses au problème étudié. Nous concluons le chapitre sur la mise en évidence de l'originalité de notre problème et plus particulièrement d'une particularité de la fonction objectif par rapport aux fonctions de référence étudiées dans la communauté (X. Li, Andries Engelbrecht et Michael G Epitropakis 2013). Nous formulons une hypothèse, proposons une piste de recherche à envisager pour une nouvelle méthode de résolution et présentons des exemples de fonctions de référence pour l'évaluer.

Le **chapitre 6** s'intéresse à la recherche de plusieurs solutions optimales à l'aide de MCTS. Nous présentons les récentes avancées dans ce domaine et proposons des contributions heuristiques pour générer un ensemble de solutions diversifiées lors de la construction de l'arbre de recherche de MCTS. Nous évaluons ces méthodes à travers le même cadre expérimental que précédemment.

Pour clore ce manuscrit, nous commençons par synthétiser les contributions. Puis, nous présentons différentes pistes de recherche. D'abord, nous proposons d'étendre certains travaux entrepris lors de la thèse. Puis, nous suggérons d'ouvrir la discussion sur des axes de recherche qui s'abstraient du cadre applicatif de la thèse.

Première partie

Modélisation de réseaux de régulation génétiques hybrides

1

Modélisation de réseaux de régulation génétiques

> Les sciences construisent des modèles abstraits des phénomènes, représentation de la réalité.
>
> Extrait de (Granger et Atlan 1991)

En biologie, les bases de données sont communément utilisées pour l'étude des systèmes biologiques, car elles centralisent les informations collectées grâce à des expériences menées en laboratoire et à la littérature publiée. Il existe aujourd'hui une grande variété de bases de données hétérogènes portant sur des informations biologiques (Rigden et Fernández 2017). Ce volume et cette diversité proviennent à la fois de la variété des types de données, mais aussi de la pluralité des objectifs de conception. La question fondamentale est ainsi de savoir comment intégrer ces données biologiques afin de les rendre accessibles et exploitables plus facilement. La modélisation intervient alors comme une technique visant à expliciter et à représenter formellement les entités concernées et leurs relations, à partir de données biologiques (Morgat et Rechenmann 2002). En effet, elle permet d'acquérir une meilleure compréhension des systèmes biologiques et de donner des représentations à des niveaux de description plus ou moins détaillés. Dans ce manuscrit, nous nous intéressons à la modélisation d'une classe de systèmes biologiques : les réseaux de régulation génétiques (RRGs).

Ce chapitre débute par l'introduction des concepts clés de la modélisation de systèmes biologiques (section 1.1). Nous commençons par présenter les caractéristiques générales d'un RRG et une vue partielle des différents niveaux d'abstraction existants permettant de modéliser ce type de systèmes. Au regard de cette présentation, nous motivons et décrivons le cadre choisi pour modéliser les RRGs étudiés (section 1.2). Grâce à cette description, nous présentons ensuite les réseaux (section 1.3) et le simulateur (section 1.4), tous deux utilisés pour mener nos études expérimentales numériques. Finalement, ce chapitre se clôt sur une esquisse de la problématique de la thèse (section 1.5).

1.1 Modélisation

La **modélisation** sert à la représentation d'un phénomène biologique qui permet de valider ou réfuter des hypothèses biologiques. Les éléments fondamentaux de la modélisation sont introduits dans cette section.

1.1.1 Caractéristiques générales

Un réseau de régulation génétique est communément représenté de manière statique par un **graphe d'interaction** (appelé aussi graphe d'influence). Un graphe d'interaction est, *a minima*, un graphe orienté $G = (V, E)$ avec V un ensemble de nœuds et E un ensemble d'arcs. Un nœud représente une **variable du graphe d'interaction** qui abstrait le regroupement d'un gène, de son ARN, des protéines traduites, *etc.* La notion de variable est préférée à celle de gène pour deux raisons. Premièrement, un nœud d'un RRG peut tout aussi bien représenter un gène qu'une condition expérimentale (comme la présence de lumière). De plus, les formalismes de modélisation peuvent être appliqués dans un cadre plus général que la seule modélisation des réseaux de régulation génétiques, comme par exemple dans le métabolisme énergétique (Khoodeeram, Bernot et Trosset 2017) (on parle dans ce cas de réseaux de régulation biologiques). Un arc, éventuellement étiqueté, représente la **régulation** d'un nœud cible par une source. Le signe + fait référence à une activation, tandis que le signe − représente une inhibition. La figure 1.1 illustre un graphe d'interaction d'un RRG composé de deux variables : v_1 et v_2. Dans cet exemple, v_1 active v_2 ($v_1 \xrightarrow{+} v_2$) et v_2 inhibe v_1 ($v_2 \xrightarrow{-} v_1$) en retour. Il s'agit d'une boucle rétroactive négative.

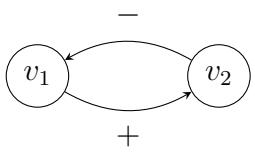


Figure 1.1 – Exemple de graphe d'interaction d'un RRG à deux variables (v_1 et v_2) dans lequel v_1 active v_2 ($v_1 \xrightarrow{+} v_2$) et v_2 inhibe v_1 ($v_2 \xrightarrow{-} v_1$) en retour.

Un graphe d'interaction est une représentation statique, car il ne décrit aucune évolution de la concentration des produits des gènes au cours du temps. Il y a plusieurs manières de représenter ou de modéliser la dynamique d'un tel système, et c'est le **cadre de modélisation** choisi qui va permettre de dériver d'un graphe d'interaction une ou éventuellement plusieurs dynamiques. Il existe une multiplicité de cadres de modélisation, chacun reflétant un niveau d'abstraction plus ou moins élevé et se basant sur des connaissances différentes du réseau étudié. Il est parfois possible de fournir une représentation très précise de l'évolution des concentrations de chacun des produits des gènes à l'aide d'équations différentielles ordinaires (*ordinary differential equations*, ODEs), comme il est parfois désirable de se concentrer sur les caractéristiques fondamentales de la structure du réseau en discrétisant la dynamique (Jamshidi 2013). Le choix d'un cadre de modélisation dépend donc fortement (i) de la connaissance (disponibilité et qualité des

données) sur la dynamique du RRG étudié et (ii) des objectifs qui dirigent la conception du modèle.

Quel que soit le cadre de modélisation choisi, la dynamique du réseau est contrôlée par des **paramètres dynamiques**. Une **paramétrisation** est une valeur donnée à chacun des paramètres dynamiques. La recherche d'une paramétrisation est un jalon dans la conception d'un **modèle valide**, c'est-à-dire d'un modèle qui décrit « correctement » le comportement du système biologique étudié, *i.e.*, en adéquation avec les informations biologiques à disposition.

Le problème d'identification des paramètres dynamiques d'un réseau de régulation génétique ne doit pas être confondu avec un autre type de problème bien connu sous le nom d'« inférence de réseau de régulation génétique » qui consiste à reconstruire le graphe d'interaction à partir de données mesurées par séquençage (Huynh-Thu et Sanguinetti 2019), procédé qui s'apparente à de la rétro-ingénierie.

Nous présentons maintenant, de manière non exhaustive et succincte, différentes catégories de cadres de modélisation en nous inspirant du classement réalisé par (Jonathan Behaegel 2018).

1.1.2 Cadres de modélisation

Tout d'abord, les cadres de modélisation différentiels cherchent à décrire précisément l'évolution des concentrations de chaque variable au cours du temps à l'aide d'ODEs. L'utilisation d'équations différentielles génère de fait un cadre continu, dans lequel le temps et les concentrations des variables sont des valeurs positives finies réelles qui évoluent de manière continue. Les régulations entre ces variables peuvent être décrites par des fonctions qui expriment la dépendance de la variation de la concentration d'une variable par rapport à la concentration d'une ou plusieurs autres variables. Ces fonctions représentent généralement la synthèse ou la dégradation d'une variable. Pour ce faire, la fonction de Hill ou sigmoïde (figure 1.2a) est généralement utilisée : elle augmente de manière monotone entre 0 et 1 et son point d'inflexion (point dont la dérivée seconde est nulle) est atteint à une valeur seuil notée θ. La fonction rampe (figure 1.2b), qui augmente de manière monotone de 0 à 1 dans un intervalle de seuil $[\theta_1, \theta_2]$ et prend la valeur 0 avant θ_1 et 1 après θ_2, a été introduite dans le modèle *piecewise multi-affine* (PMA) pour approximer

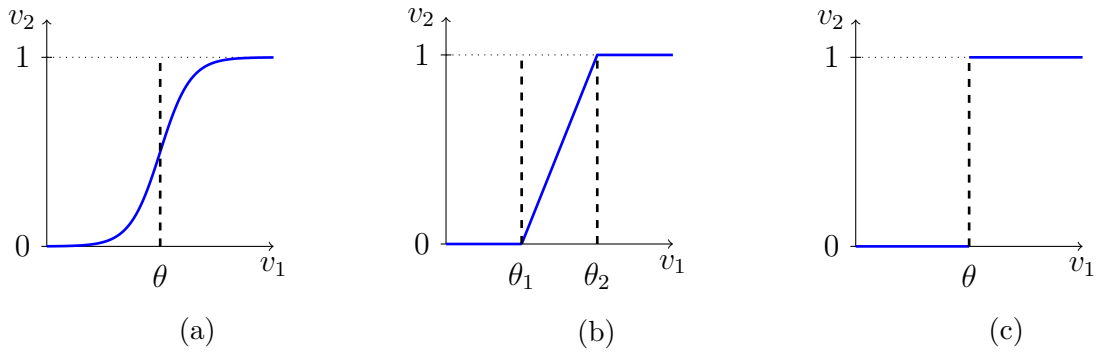


Figure 1.2 – Les fonctions de Hill (a), rampe (b) et de Heaviside (c) peuvent être utilisées pour approximer l'effet d'une variable (ici, v_1) sur le taux de synthèse d'une autre variable (ici, v_2) et donnent lieu à des modèles différents (ODE, PMA et PADE, respectivement).

la fonction de Hill et ainsi simplifier la résolution des systèmes d'équations différentielles. Dans tous les cas, les solutions d'un système d'équations différentielles sont des fonctions continues et déterministes. Ce cadre de modélisation offre donc une représentation quantitative de l'évolution de chaque variable qui doit être comparée à des données biologiques exploitables issues d'expériences.

Cependant, les informations sont souvent lacunaires et trop bruitées, ce qui justifie le développement des cadres de modélisation discrets (appelés aussi qualitatifs). Ils fournissent une représentation très abstraite, mais toujours réaliste, permettant de mettre en évidence les caractéristiques systémiques fondamentales. On peut penser au formalisme booléen (S. Kauffman 1969), dans lequel les niveaux de concentration des variables peuvent prendre deux valeurs, représentant leur présence ou leur absence dans les différents états du système ; ou aux réseaux de Petri (Peterson 1977) qui, quant à eux, ne prennent pas en compte le temps passé dans chacun des états discrets multivalués du RRG.

Finalement, les cadres de modélisation hybrides, que nous allons considérer, combinent les différents niveaux d'abstraction en proposant un compromis entre les méthodes de modélisation discrètes et continues. Par exemple, l'approche PADE (Glass et S. A. Kauffman 1972) (*Piecewise Affine Differential Equations*) approxime la fonction de Hill avec une fonction de Heaviside (figure 1.2c), qui prend la valeur 0 avant la valeur seuil θ et 1 après. Cette approximation est cohérente, car la fonction d'Heaviside est considérée comme une fonction de Hill avec une pente infinie. Il existe de nombreux autres cadres hybrides : certains, comme PADE, cherchent à approximer le cadre différentiel (Coutinho et al. 2006 ; Davidich et Bornholdt 2008) et d'autres visent à introduire des dynamiques continues au cadre discret (Ahmad et al. 2007 ; Siebert et Bockmayr 2006).

Dans ce travail, nous utilisons un **cadre de modélisation hybride**. Ce choix a été motivé par plusieurs facteurs clés. Du point de vue de la modélisation, les cadres différentiels décrivent de manière précise l'évolution des produits des gènes, mais au prix (i) d'une quantité importante de connaissance biologique nécessaire et (ii) d'un processus d'identification des paramètres dynamiques qui est loin d'être trivial, car il demande de trouver un nombre de valeurs réelles encore plus important que dans les cadres discrets. Les travaux de (Gérard et Goldbeter 2012) témoignent de ces désavantages sur l'exemple du couplage des cycles circadiens et cellulaires. De plus, le raffinement d'un modèle existant, avec l'ajout d'une ou plusieurs variables dans le RRG, peut rendre obsolète la précédente identification. Les cadres discrets, quant à eux, présentent l'avantage de réduire l'ensemble des valeurs possibles des paramètres par un ensemble fini de valeurs. Cependant, ces cadres ne prennent pas en compte le temps qui sépare les événements biologiques et ne tiennent compte que de l'ordre. Or, ces informations sont disponibles et précieuses, et elles permettent d'ajouter des contraintes sur les paramètres dynamiques et donc de les identifier. D'un point de vue applicatif, il est crucial de connaître l'évolution temporelle des concentrations des produits des gènes pour, par exemple, optimiser les traitements médicaux, comme c'est le cas avec la chronothérapie. C'est pour ces raisons qu'un cadre hybride a été choisi pour représenter la dynamique des RRGs étudiés. Ce cadre préserve la discrétisation des états et des événements biologiques et permet de définir une évolution temporelle continue entre ces événements. La section suivante vise à introduire le cadre de modélisation hybride choisi.

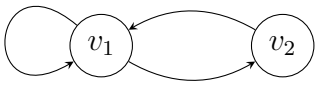
1.2 Cadre utilisé

Cette présentation est réalisée par incréments. Elle débute par la brique de base qu'est le cadre booléen (section 1.2.1), passe en revue les différentes évolutions et aboutit au cadre hybride (section 1.2.2).

1.2.1 Du formalisme booléen à celui de René Thomas

Initialement, (S. Kauffman 1969) introduit la dynamique d'un RRG booléen. Chaque sommet du graphe d'interaction est interprété comme une variable booléenne. Une variable peut donc être dans l'un des deux états suivants : activé (1) ou désactivé (0). L'état booléen du réseau est ainsi décrit par un vecteur booléen à chaque instant discret $T = 1, 2, 3, ...$ Par exemple, l'état $(0, 0, 1)$ à un instant T pour un RRG de trois variables (v_1, v_2, v_3) représente l'état du réseau où v_1 et v_2 sont inactifs et v_3 est actif.

L'évolution de l'état de chaque variable est décrite par une fonction booléenne, appelée aussi fonction de mise à jour, qui dépend de la valeur des **prédécesseurs** [1] de la variable dans le graphe d'interaction. Par exemple, dans le RRG booléen à deux variables (v_1 et v_2) en figure 1.3a, l'ensemble des prédécesseurs de v_1 est $\{v_1, v_2\}$ et v_2 a comme unique prédécesseur v_1. La fonction booléenne d'une variable définit le nouvel état d'une variable v à l'instant $T + 1$ ($f_{T+1}(v)$) à partir de l'instant T ($f_T(v)$). La mise à jour du réseau est synchrone, c'est-à-dire que tous les états sont mis à jour en appliquant de manière simultanée les fonctions booléennes de chaque variable. Avec les fonctions de mises à jour de la figure 1.3b, si l'état du système à un instant T est $(v_1, v_2) = (0, 1)$, alors $f_{T+1}(v_1) = \neg f_T(v_1) \vee f_T(v_2) = \neg 0 \vee 1 = 1$ et $f_{T+1}(v_2) = f_T(v_1) = 0$. À l'instant $T + 1$, l'état du système est $(v_1, v_2) = (1, 0)$.



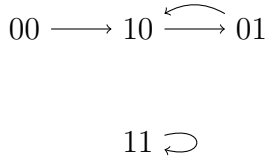
(a)

$$f_{T+1}(v_1) = \neg f_T(v_1) \vee f_T(v_2)$$
$$f_{T+1}(v_2) = f_T(v_1)$$

(b)

T		$T+1$	
v_1	v_2	v_1	v_2
0	0	1	0
0	1	1	0
1	0	0	1
1	1	1	1

(c)



(d)

Figure 1.3 – (a) : Exemple d'un RRG booléen de deux variables ($V = \{v_1, v_2\}$) avec son graphe d'interaction. (b) : Les fonctions booléennes de mise à jour de chaque variable. (c) : Chaque état possible du système et leurs états successeurs par mise à jour synchrone des fonctions booléennes. (d) : Le graphe d'états résultant.

1. Dans un graphe orienté G, si un arc joint deux sommets A et B ($A \to B$), alors on dit que A le prédécesseur de B.

En appliquant ces fonctions booléennes, une table de mise à jour peut être extraite. Elle définit le nouvel état du système à l'instant $T+1$ à partir de l'instant T. L'ensemble des transitions entre états booléens est répertorié dans la figure 1.3c. On y retrouve bien, dans la deuxième ligne, la transition $(0,1) \to (1,0)$ présentée en exemple précédemment. L'ensemble des états possibles définit l'univers du modèle et la dynamique est représentée par un **graphe d'états**. Le graphe d'états est une représentation de la dynamique du système, dans laquelle les trajectoires sont représentées par les chemins et les points fixes ou les boucles par des composantes fortement connexes terminales. Le graphe d'états associé à la figure 1.3a est représenté en figure 1.3d.

Afin de généraliser l'approche de la modélisation booléenne, (René Thomas 1973 ; René Thomas et d'Ari 1990 ; R. Thomas et Kaufman 2001) suggèrent de l'étendre aux variables discrètes multivaluées, ce qui permet de prendre en compte les niveaux d'activité des variables du réseau. Cette méthode consiste à abstraire les concentrations (quantitatives) des variables par des niveaux discrets (qualitatifs). Le principe est de découper l'espace des concentrations du formalisme des équations différentielles par morceaux et de numéroter les intervalles par ordre croissant à partir de 0. Plus précisément, cette abstraction repose sur la relation suivante : pour chaque régulation v_i vers v_j, la concentration de v_j dépend de la concentration de v_i lorsque la régulation a lieu. Cette dépendance est représentée, comme on l'a vu précédemment, par une fonction de Hill. L'ensemble des concentrations de v_i est alors découpé en deux intervalles de concentrations : le seuil qui sépare ces deux intervalles est le point d'inflexion θ. Ce cas de figure est illustré dans la figure 1.4b : étant donné le graphe d'interaction du RRG discret à deux variables (v_1 et v_2) de la figure 1.4, v_2 présente deux niveaux qualitatifs (0 inactif et 1 actif) séparés par le seuil θ. Dans ce formalisme, notons que les régulations sont de la forme $v_i \xrightarrow{+n} v_j$ (resp. $v_i \xrightarrow{-n} v_j$) pour une activation (resp. inhibition). Cela signifie : v_i active (resp. inhibe) v_j si son niveau discret est supérieur ou égal à n.

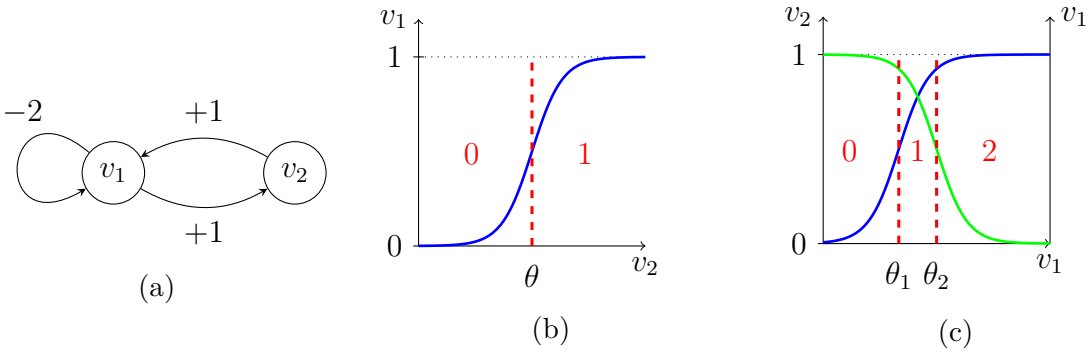


Figure 1.4 – Discrétisation des concentrations en niveaux d'expression. (a) : Le graphe d'interaction d'un RRG discret à deux variables v_1 et v_2. (b) : v_2 présente un seuil θ et deux niveaux qualitatifs (0 inactif et 1 actif). (c) v_1 présente deux seuils θ_1 et θ_2 et trois niveaux : v_1 est inactif (niveau 0), v_1 agit sur v_2 uniquement (niveau 1) ou il agit sur v_2 et lui-même (niveau 2).

Lorsque v_i régule plusieurs cibles, le seuil de chacune des différentes fonctions de Hill est placé différemment, ce qui découpe l'ensemble des concentrations en plus de deux intervalles. Dans la figure 1.4c, v_1 présente trois niveaux : v_1 est inactif (niveau 0), v_1 agit sur v_2 uniquement (niveau 1) ou il agit sur v_2 et lui-même (niveau 2). Ces trois niveaux

sont découpés par les deux seuils θ_1 et θ_2.

De ce fait, le codage de l'intervalle par un entier indique le nombre de cibles sur lesquelles la variable agit. Chaque intervalle de concentration est considéré comme étant homogène (l'activité des produits des gènes est stable), on parle alors de **niveau d'expression** d'une variable $v \in V$, noté η_v et son niveau maximal est noté b_v. Ainsi, $\forall v \in V, \eta_v \in [\![0, b_v]\!]$ où $[\![a, b]\!]$ désigne un intervalle de nombres entiers entre a et b. Le temps reste aussi discrétisé. L'**état discret** du réseau à un instant T est décrit par le vecteur des niveaux d'expression de chacune des variables du réseau. Par exemple, l'état discret $\eta = (\eta_{v_1}, \eta_{v_2}, \eta_{v_2}) = (2, 0, 1)$ à un instant T pour un RRG de trois variables (v_1, v_2, v_3) représente l'état du réseau où v_1 a un niveau d'expression de 2, v_2 est inactif et v_3 a un niveau d'expression de 1.

De manière analogue au formalisme booléen de Kauffman, l'ensemble des états discrets possibles, noté $\mathbb{S}$, définit l'univers du modèle et la dynamique est représentée par un graphe d'états. La figure 1.5a présente un graphe d'interaction à deux variables (v_1, v_2). Chaque variable possède deux niveaux d'expression ($b_{v_1} = b_{v_2} = 1$). Ainsi, il existe quatre états discrets ($\mathbb{S} = \{(0,0), (0,1), (1,0), (1,1)\}$). Chaque état discret est représenté par un carré unité à bord gris dans le graphe d'états présenté en figure 1.5b [2]. Les flèches noires montrent les transitions d'un état discret vers un autre ($\{(0,0) \to (0,1), (0,1) \to (1,0), (1,0) \to (1,1), (1,1) \to (0,0)\}$) modélisant ainsi la dynamique discrète qui forme un cycle.

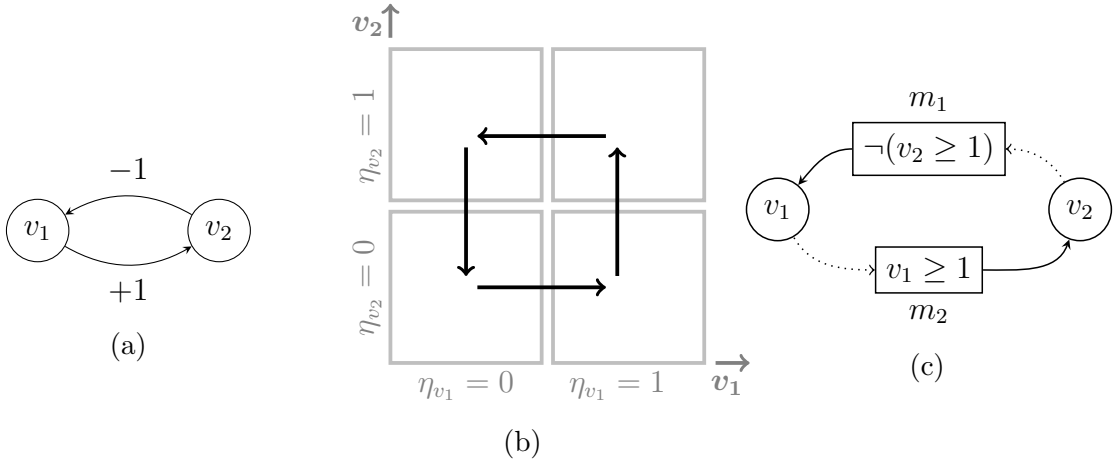


FIGURE 1.5 – (a) : Illustration d'un graphe d'interaction d'un RGG discret avec deux variables v_1 et v_2 et (b) : son graphe d'états. Le graphe d'interaction du RGG avec multiplexes (c) est analogue à celui présenté en (a).

À partir du formalisme discret de René Thomas, (KHALIS et al. 2009) proposent une nouvelle définition du graphe d'interaction. Cette dernière introduit des **multiplexes** sous forme de formules logiques. Un multiplexe permet de spécifier les conditions de régulation entre une ou plusieurs variables sources sur une même variable cible. La figure 1.5c présente deux multiplexes m_1 et m_2. m_1 spécifie que si le niveau discret de v_2 est supérieur ou égal

2. Les états discrets, représentés par des carrés gris, sont espacés. Ces espaces entre les états sont purement illustratifs, on aurait pu représenter les états collés. La transition entre deux états discrets adjacents indique qu'un seuil a été atteint, elle est ainsi représentée par une flèche qui sort d'un état et rentre dans l'état adjacent.

à 1, alors v_2 inhibe v_1. m_2 spécifie que si le niveau discret de v_1 est supérieur ou égal à 1, alors v_1 active v_2.

Le graphe d'interaction devient dès lors un graphe biparti composé de deux types de nœuds : les variables et les multiplexes, et de deux types d'arcs : les arcs pointillés et les arcs pleins. Un arc pointillé part d'une variable et pointe sur un multiplexe, indiquant que la variable participe à la régulation représentée par le multiplexe. Il s'agit donc d'une information redondante qui peut être supprimée sans perte d'information : ces arcs peuvent être reconstruits trivialement en identifiant les variables de la formule du multiplexe. Ils restent néanmoins une aide visuelle à la compréhension et sont donc gardés sur les figures, mais ne sont pas présents dans la définition des graphes d'interaction. Un arc plein part d'un multiplexe et arrive sur une variable cible qu'il régule [3]. Ainsi, les représentations des figures 1.5a et 1.5c sont des représentations équivalentes du même RRG. Pour s'en convaincre, il suffit de reconnaître que la transformation d'une régulation de la forme $v_i \xrightarrow{+n} v_j$ suit les étapes : (i) ajout d'un multiplexe m avec la formule logique $v_i \geq n$, (ii) un arc solide de m vers v_j et, de manière optionnelle, (iii) un arc pointillé de v_i vers m. La transformation d'une régulation $v_i \xrightarrow{-n} v_j$ suit aussi les mêmes étapes, à la différence que le multiplexe m contient la formule logique $\neg(v_i \geq n)$.

La définition 1 décrit un RRG discret avec multiplexes, noté $R = (V, M, E, K)$. V constitue l'ensemble des variables, M l'ensemble des multiplexes, E l'ensemble des arcs et K l'ensemble des paramètres dynamiques. Dans cette définition, l'ensemble des prédécesseurs d'une variable, noté $R^-(v)$, est constitué des multiplexes sources. Notons que les ressources d'une variable dans un état donné η sont les prédécesseurs dont les formules sont satisfaites dans η.

Réseau de régulation génétique discret avec multiplexes

Définition 1. Un RRG discret avec multiplexes est un quadruplet $R = (V, M, E, K)$ dans lequel :

- V est un ensemble fini d'éléments et $\forall v \in V, \exists b_v \in \mathbb{N}^*$ appelée borne limite qui correspond au niveau d'expression maximal de v.
- M est un ensemble fini de multiplexes dans lequel $\forall m \in M$, m est muni d'une formule ϕ_m formée des atomes $(v \geq n)$ avec $v \in V$ et $n \in [\![1, b_v]\!]$ et des connecteurs logiques $\{\neg, \vee, \wedge, \Rightarrow\}$
- E est un ensemble d'arcs de la forme $(m \to v) \in M \times V$.
- $K = \{K_{v,\omega}\}$ est une famille d'entiers indexés par un couple (v, ω) où :
  1. $v \in V$
  2. $\omega \subset R^-(v)$ avec $R^-(v) = \{m \mid (m \to v) \in E\}$, autrement dit, ω est un ensemble de prédécesseurs de v dans R
  3. $K_{v,\omega} \in [\![0, b_v]\!]$

  $K_{v,\omega}$ est appelé le paramètre de v pour l'ensemble de ressources ω.

3. Il est à noter qu'un multiplexe agit toujours positivement sur une cible. Si sa formule est satisfaite, il a une action positive sur sa cible, sinon il n'en a pas. Lors du codage d'une inhibition, l'absence d'inhibition permet à la variable cible d'être active, d'où la présence d'une négation.

Ces définitions formelles sont nécessaires à la construction des paramètres dynamiques. En effet, un paramètre dynamique dans K dépend de et est indexé par une variable et l'ensemble des ressources de cette variable. Par exemple, $K_{v_1,\{\}}$ correspond au paramètre dynamique de la variable v_1 lorsqu'aucune des formules des multiplexes prédécesseurs n'est satisfaite, *i.e.*, v_1 ne possède aucune ressource. La valeur de chaque paramètre dynamique correspond au niveau d'expression vers lequel la variable est attirée (la valeur focale). Par exemple, si $K_{v_1,\{\}} = 0$, alors lorsque v_1 n'a pas de multiplexes dont les formules sont évaluées à vrai, son niveau d'expression se dirige vers 0.

Exemple 1. Le graphe d'interaction de la figure 1.5c définit le réseau de régulation $R = (V, M, E, K)$ où $V = \{v_1, v_2\}$, $M = \{m_1, m_2\}$, $E = \{m_1 \to v_1, m_2 \to v_2\}$ et $K = \{K_{v_1,\{\}}, K_{v_2,\{\}}, K_{v_1,\{m_1\}}, K_{v_2,\{m_2\}}\}$.

— Dans l'état $\eta = (\eta_{v_1}, \eta_{v_2}) = (0, 0)$, le multiplexe m_1 est satisfait et m_2 ne l'est pas. Dans cet état, les paramètres $K_{v_1,\{m_1\}}$ et $K_{v_2,\{\}}$ contrôlent la dynamique. Si l'on en croit la dynamique matérialisée par la flèche noire partant de $(0, 0)$ et arrivant dans $(1, 0)$ de la figure 1.5b, alors $K_{v_1,\{m_1\}} = 1$ et $K_{v_2,\{\}} = 0$.

— Dans l'état discret $(1, 0)$, les formules des deux multiplexes sont fausses ainsi les paramètres dynamiques en jeu sont $K_{v_1,\{\}}$ et $K_{v_2,\{\}}$. D'après la dynamique représentée, $K_{v_1,\{\}} = 1$ et $K_{v_2,\{\}} = 1$.

— Pour $\eta = (1, 1)$, les deux multiplexes ont des formules évaluées à vrai et les paramètres ont pour valeur focale $K_{v_1,\{m_1\}} = 0$ et $K_{v_2,\{m_2\}} = 1$.

— Finalement, pour $\eta = (0, 1)$, le même raisonnement mène à $K_{v_1,\{\}} = 0$ et à $K_{v_2,\{m_2\}} = 0$.

Les connaissances biologiques et les contraintes sur les paramètres dynamiques, comme celle de (Snoussi 1989) par exemple, permettent de déterminer les valeurs de ces paramètres afin d'extraire des modèles discrets valides en adéquation avec certaines connaissances sur les réseaux biologiques étudiés. C'est le cas avec le cycle circadien simplifié (Jean-Paul Comet et al. 2012) ou encore le métabolisme énergétique (Khoodeeram, Bernot et Trosset 2017). Ces travaux démontrent l'intérêt de l'approche discrète de Thomas qui abstrait les concentrations des variables pour permettre de raisonner et d'extraire des contraintes sur les paramètres et ainsi d'automatiser la recherche et l'extraction de modèles valides. Cependant, le temps est lui aussi discrétisé. Or, l'analyse **chronométrique**, c'est-à-dire une analyse relative à la mesure exacte du temps, est cruciale en chronobiologie. En effet, dans des systèmes biologiques tels que le cycle circadien ou le cycle cellulaire, connaître la durée du cycle est un critère primordial de validation d'un modèle. Par exemple, le cycle circadien doit avoir une période de 24 heures.

C'est pourquoi nous utilisons un cadre hybride basé sur le cadre discret qui permet d'ajouter des informations chronométriques, et par conséquent continues, à la dynamique discrète.

1.2.2 Cadre de René Thomas hybride

L'approche discrète de Thomas ne permet d'étudier que la succession des changements d'états discrets : il manque la notion de temps chronométrique (par opposition au temps discrétisé). Ce que propose le cadre de modélisation, défini par (Cornillon 2017) et

repris dans (Jonathan Behaegel 2018), est d'y intégrer une dynamique temporelle. De la même manière que dans le cadre discret avec multiplexes, le cadre hybride abstrait les concentrations des produits des gènes par des niveaux d'expression, mais il permet en plus de mesurer le temps passé à l'intérieur de chaque état discret. On parle de **cadre de René Thomas hybride**. Nous définissons les éléments fondamentaux de ce cadre.

1.2.2.1 Définitions préliminaires du cadre hybride

Les **réseaux de régulation génétiques hybrides** (RRGHs) sont définis sur la base des RRGs discrets avec multiplexes de la définition 1.

Réseau de régulation génétique hybride (RRGH)

Définition 2. Un RRG hybride (RRGH) est un RRG discret avec multiplexe $R = (V, M, E, C)$ dans lequel :

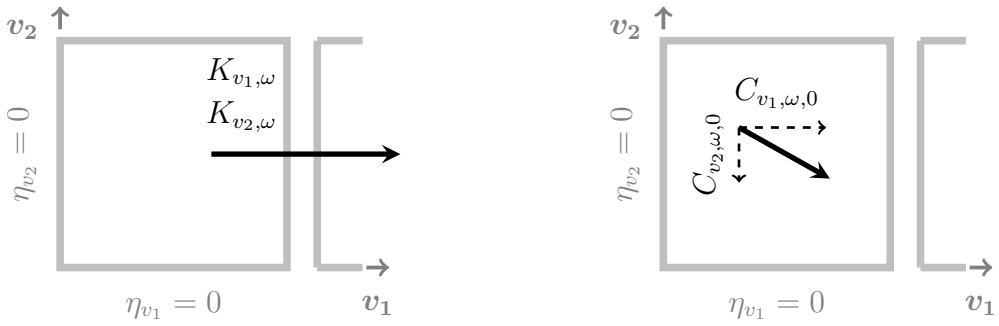
- V est un ensemble fini d'éléments et $\forall v \in V, \exists\ b_v \in \mathbb{N}^*$ appelée borne limite correspondant au niveau d'expression maximal de v.
- M est un ensemble fini de multiplexes dans lequel $\forall m \in M$, m est muni d'une formule ϕ_m formée des atomes $(v \geq n)$ avec $v \in V$ et $n \in [\![1, b_v]\!]$ et des connecteurs logiques $\{\neg, \vee, \wedge, \Rightarrow\}$
- E est un ensemble d'arcs de la forme $(m \to v) \in M \times V$.
- $C = \{C_{v,\omega,n}\}$ est une famille de nombres réels indexés par un triplet (v, ω, n) où :
  1. $v \in V$
  2. $\omega \subset R^-(v)$ avec $R^-(v) = \{m \mid (m \to v) \in E\}$
  3. $n \in [\![0, b_v]\!]$

  $C_{v,\omega,n}$ est appelée célérité de v au niveau d'expression n pour l'ensemble de ressources ω.

La définition des RRGHs diffère par la nature des paramètres dynamiques. La famille de paramètres discrets K est remplacée par une famille de paramètres réels, appelés **célérités** dont l'ensemble est noté C. Un paramètre dynamique dans C dépend de et est indexé par une variable, l'ensemble de ses ressources et son niveau d'expression [4]. Par exemple, $C_{v_1,\{\},0}$ correspond au paramètre dynamique de la variable v_1 lorsque son niveau d'expression est au niveau 0 ($\eta_{v_1} = 0$) et que v_1 ne possède aucune ressource. La valeur de chaque paramètre dynamique est une valeur réelle qui est similaire à la vitesse, *i.e.*, célérité, d'évolution de la variable en fonction du niveau d'expression indiqué et de l'ensemble de ses ressources. Notons que le signe de la célérité précise la direction, indiquant l'état discret vers lequel est attirée la variable. La figure 1.6 illustre la différence entre des paramètres discrets de K (figure 1.6a) et continus de C (figure 1.6b). Dans le

4. Dans le cadre discret, le niveau d'expression n'est pas mentionné, car étant donné un ensemble de ressources ω de la variable v, la valeur focale est constante quel que soit le niveau d'expression de v. Ce qui n'est pas le cas dans le cadre hybride. En effet, en ajoutant le temps chronométrique, la valeur focale reste la même, mais la célérité peut changer en fonction de l'état discret. Ainsi, il est important de distinguer *a priori* deux célérités ayant le même ensemble de ressources pour une même variable, mais à deux niveaux d'expression distincts.

premier cas, les paramètres discrets ($K_{v_1,\omega}$ et $K_{v_2,\omega}$) représentent la transition d'un état discret à un autre ($(0,0) \to (1,0)$), tandis que, dans le second cas, les paramètres continus ($C_{v_1,\omega,0}$ et $C_{v_2,\omega,0}$) représentent un angle et une direction à l'intérieur de l'état discret $(0,0)$ vers l'état discret $(1,0)$.



(a) (b)

FIGURE 1.6 – Illustration de la différence entre les paramètres dynamiques discrets K en (a) et continus C en (b).

Dans le cadre hybride, un état discret est représenté par un hypercube unité [5] $[0,1]^{|V|}$ où la dimension de l'hypercube correspond au nombre de variables dans le graphe d'interaction. Les bornes de l'intervalle fermé (représentées par les bords des hypercubes) représentent les seuils définissant les niveaux d'expression. La normalisation des valeurs de seuils (elles sont toutes des valeurs entières) a été choisie pour simplifier le problème de recherche des paramètres dynamiques, car la recherche des seuils est un problème de recherche complexe, récemment abordé dans (GRATALOUP et al. 2024). Il faudrait en fait représenter des hyperrectangles et placer les valeurs de seuils. Dans cette thèse, nous nous affranchissons de ce problème en considérant des hypercubes.

Un **état hybride** h est défini comme étant la donnée (i) d'un état discret η du réseau et (ii) d'une position exacte à l'intérieur de l'état discret, notée π et appelée **partie fractionnaire**, voir la définition 3.

État d'un RRG hybride

Définition 3. Soit $R = (V, M, E, C)$ un RRGH, un **état hybride** est un couple $h = (\eta, \pi)$ où :

- $\eta : V \to \mathbb{N}$ telle que $\forall v \in V, \eta(v) \in [\![0, b_v]\!]$; on appelle η l'état discret de h.
- $\pi : V \to [0,1]$; π est appelé partie fractionnaire de h.

Pour simplifier la notation, nous utiliserons η_v pour $\eta(v)$ et π_v pour $\pi(v)$. On note H l'ensemble des états hybrides de R.

La figure 1.7 présente deux exemples d'état hybride : un premier en dimension deux et un second en dimension trois.

Les ressources des célérités, se reposant sur la notion d'état hybride, sont introduites dans la définition 4.

5. Un hypercube unité est un cube dans un espace à plusieurs dimensions et de côté de longueur 1.

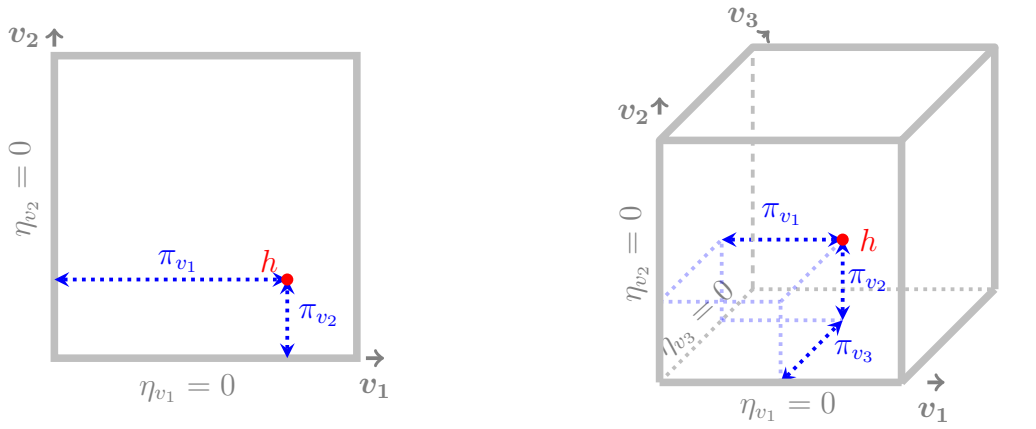


FIGURE 1.7 – Représentation graphique d'un état hybride h en deux dimensions (a) avec $h = (\eta, \pi) = \left(\begin{pmatrix}\eta_{v_1}\\ \eta_{v_2}\end{pmatrix}, \begin{pmatrix}\pi_{v_1}\\ \pi_{v_2}\end{pmatrix}\right)$ et en trois dimensions (b) avec $h = (\eta, \pi) = \left(\begin{pmatrix}\eta_{v_1}\\ \eta_{v_2}\\ \eta_{v_3}\end{pmatrix}, \begin{pmatrix}\pi_{v_1}\\ \pi_{v_2}\\ \pi_{v_3}\end{pmatrix}\right)$.

Ressources d'un RRGH

Définition 4. Soient $R = (V, M, E, C)$ un RRGH, ϕ une formule de multiplexe et $h = (\eta, \pi) \in H$ un état hybride de R. La relation $h \vDash \phi$ est définie inductivement par :

- Si ϕ est un atome de la forme $(v \geq n)$ avec $n \in [\![1, b_v]\!]$, alors $h \vDash \phi$ si et seulement si $\eta_v \geq n$,
- La sémantique usuelle des connecteurs logiques $\{\neg, \vee, \wedge, \Longrightarrow\}$ est utilisée.

L'ensemble des ressources de v pour un état hybride h est défini par : $\rho(h, v) = \{m \mid h \vDash \phi_m, m \in R^-(v)\}$. En d'autres termes, il s'agit de l'ensemble des multiplexes prédécesseurs de v dont la formule est satisfaite.

1.2.2.2 Dynamique d'un RRG hybride

Les définitions précédentes préfigurent la définition de la dynamique d'un RRGH. S'intéresser à cette dynamique, c'est se demander comment représenter l'évolution de la concentration des produits des gènes au cours du temps. Nous l'avons vu, la dynamique hybride se base sur la dynamique discrète et introduit le temps chronométrique entre les événements que sont les changements d'états discrets. Elle se décompose ainsi en une alternance de transitions discrètes (passage d'un état discret à un autre) et de transitions continues (comportement à l'intérieur des états discrets pour atteindre un seuil). On représente ainsi cette dynamique par une **trajectoire** linéaire par morceau : une trajectoire continue pour une transition continue et une trajectoire discrète pour une transition discrète. Une trajectoire commence à un état hybride initial (début de la transition continue)

et évolue en fonction des paramètres dynamiques de l'état discret de l'état hybride initial pour atteindre un seuil au bout d'un certain délai $\Delta t \in \mathbb{R}^*+$ (fin de la transition continue), la trajectoire franchit ensuite le seuil (transition discrète) et initie une nouvelle transition continue dans le nouvel état discret. Ce processus est itéré jusqu'à ce qu'un point fixe soit atteint.

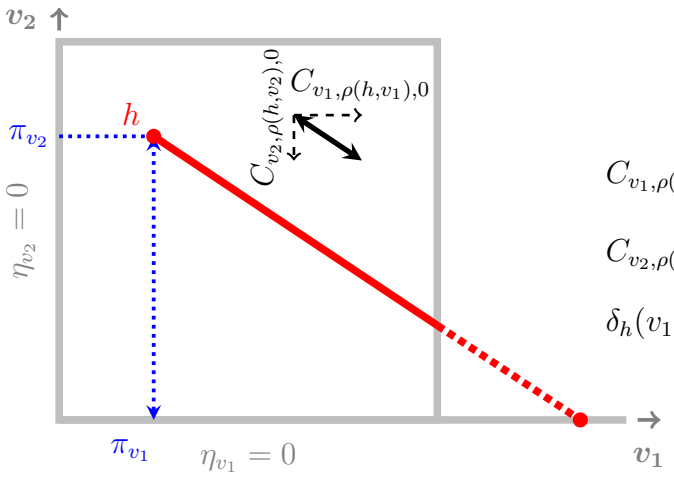
La notion de délai intervient dans la transition continue afin de connaître le temps nécessaire à chaque variable pour atteindre un seuil pour la variable considérée. Ce temps est appelé **délai de contact** de la variable. Il dépend de la distance à parcourir à partir de l'état hybride initial, voir la définition 5.

Délai de contact

Définition 5. Soient $R = (V, M, E, C)$ un RRG hybride, $v \in V$ et $h = (\eta, \pi) \in H$. On note $\delta_h(v)$ le délai de contact de v dans h, le temps nécessaire à v pour atteindre le bord de l'état discret η. $\delta_h : V \to \mathbb{R}^+$ est définie par :

- Si $C_{v,\rho(h,v),\eta_v} = 0$, alors $\delta_h(v) = +\infty$.
- Si $C_{v,\rho(h,v),\eta_v} > 0$ (resp. < 0), alors $\delta_h(v) = \frac{1-\pi_v}{C_{v,\rho(h,v),\eta_v}}$ (resp. $\frac{\pi_v}{C_{v,\rho(h,v),\eta_v}}$).

La figure 1.8 schématise un exemple de délai de contact des variables v_1 et v_2 à partir de l'état hybride h. La transition continue est représentée par une trajectoire continue (en rouge). On observe que le délai de contact de v_1 est inférieur à celui de v_2 permettant d'extraire la contrainte suivante : $\delta_h(v_1) < \delta_h(v_2)$.



(a)

$$C_{v_1,\rho(h,v_1),0} > 0 \implies \delta_h(v_1) = \frac{1-\pi_{v_1}}{C_{v_1,\rho(h,v_1),0}}$$

$$C_{v_2,\rho(h,v_2),0} < 0 \implies \delta_h(v_2) = -\frac{\pi_{v_2}}{C_{v_2,\rho(h,v_2),0}}$$

$$\delta_h(v_1) < \delta_h(v_2)$$

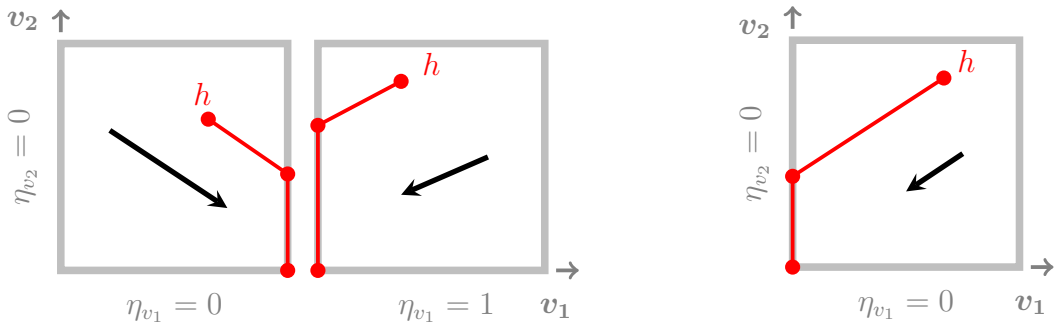
(b)

FIGURE 1.8 – (a) : Représentation graphique en deux dimensions d'un délai de contact dans $\eta = (0, 0)$ à partir de l'état hybride h en prenant en compte le vecteur célérité $\begin{pmatrix} C_{v_1,\rho(h,v_1),\eta_{v_1}} \\ C_{v_2,\rho(h,v_2),\eta_{v_2}} \end{pmatrix}$. (b) : Les équations correspondants à l'illustration (a).

Lorsque la trajectoire atteint un seuil (représenté par le bord d'un hypercube), il n'est pas sûr qu'elle puisse le franchir. En effet, il y a deux cas de figure dans lesquels il est impossible pour la trajectoire de franchir le bord auquel elle fait face :

1. si la trajectoire est attirée vers un état discret adjacent possédant un vecteur célérité contraire (de signe opposé) à celui de l'état discret actuel, alors la trajectoire fait face à un **mur interne**, voir la figure 1.9a,
2. s'il n'existe tout simplement pas d'état discret adjacent « de l'autre côté du bord », alors la trajectoire fait face à un **mur externe**, *cf.* figure 1.9b. [6]

Dans les deux cas illustrés en figure 1.9, v_1 fait face à un mur. On peut observer que la trajectoire glisse le long du mur en suivant la célérité de la variable v_2 : $s(h) = \{v_2\}$. Les variables qui font face à un mur sont dites glissantes et l'ensemble des variables glissantes est noté $s(h)$ (avec s pour *sliding*).



(a) (b)

FIGURE 1.9 – Illustration d'un mur interne (a) et d'un mur externe (b).

Dans la définition 6 présentant les deux types de murs, la fonction $sgn : \mathbb{R} \to \{-1, 0, 1\}$ est définie par :

$$sgn(x) = \begin{cases} -1 & \text{si } x < 0 \\ 0 & \text{si } x = 0 \\ 1 & \text{si } x > 0 \end{cases}$$

Murs internes et externes

Définition 6. Soient $R = (V, M, E, C)$ un RRG hybride, $v \in V$ et $h = (\eta, \pi) \in H$,

- v fait face à un mur externe en h si et seulement si :

$$(\delta_h(v) = 0) \wedge (((C_{v,\rho(h,v),\eta_v} < 0) \wedge (\eta_v = 0)) \vee ((C_{v,\rho(h,v),\eta_v} > 0) \wedge (\eta_v = b_v)))$$

- et soient $h' = (\eta', \pi') \in H$, un second état hybride défini par $\eta'_v = \eta_v + sgn(C_{v,\rho(h,v),\eta_v})$ et $\eta'_u = \eta_u, \forall u \in V,\ u \neq v$. v fait face à un mur interne en h si et seulement si v ne fait pas face à un mur externe et :

$$(\delta_h(v) = 0) \wedge (sgn(C_{v,\rho(h,v),\eta_v}) \times sgn(C_{v,\rho(h',v),\eta'_v}) = -1)$$

On note $slide(h)$, l'ensemble des variables qui sont soit sur un mur interne, soit sur un mur externe.

6. En biologie, un mur externe correspond au phénomène de saturation ou de dégradation de la concentration des produits du gène.

À partir d'un état hybride h, les variables qui présentent le délai de contact *minimum* dans un état discret sont celles qui atteignent leur bord en premier. Cependant, ces variables peuvent faire face à un mur et ainsi ne pas franchir leur seuil. C'est pourquoi une distinction est réalisée pour les variables qui présentent le délai de contact *minimum*, entre celles qui peuvent franchir leur seuil et celles qui ne le peuvent pas.

Dans la figure 1.10, on remarque que la trajectoire partant de h touche le bord bas de l'état discret $(0, 0)$, signifiant que v_2 a un délai de contact *minimum*. Cependant v_2 fait face à un mur externe et glisse le long du mur ($s(h) = \{v_2\}$). Lorsque la trajectoire atteint le bord droit, la transition discrète est possible. L'ensemble **first**, introduit dans la définition 7, contient les variables qui atteignent leur bord en premier et qui peuvent le franchir, c'est-à-dire qu'elles ne font pas face à un mur. Ainsi, dans notre exemple, $first(h) = \{v_1\}$ et on note que $first(h) \cap s(h) = \emptyset$.

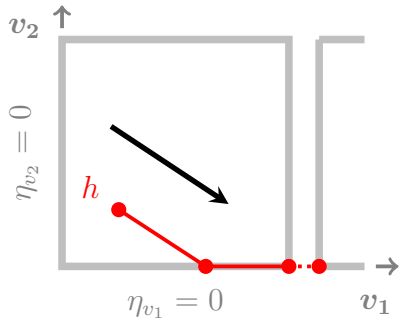


FIGURE 1.10 – La variable v_2 atteint son bord en premier. Cependant, v_2 fait face à un mur externe et ne peut pas franchir de seuil. Ainsi v_1 est la première variable qui peut franchir son seuil : $first(h) = \{v_1\}$.

Ensemble *first*

Définition 7. Soient $R = (V, M, E, C)$ un RRG hybride et $h = (\eta, \pi) \in H$. L'ensemble *first* est défini par :
$first(h) = \{v \in V \setminus slide(h) \mid \ (\delta_h(v) < +\infty) \wedge (\forall u \in V \setminus slide(h), \delta_h(u) \geq \delta_h(v))\}$

Les éléments appartenant à l'ensemble *first* sont ceux qui génèrent les transitions discrètes, car ce sont eux qui atteignent leur seuil en premier et qui peuvent les franchir. Ainsi, ils imposent le délai avant la prochaine transition discrète et les autres éléments évoluent en fonction de ce délai.

La définition 8 formalise ce qui a été mentionné précédemment, c'est-à-dire le graphe d'états hybride représentant la dynamique. Cette dynamique est constituée de deux types de transitions :

— la transition continue (cas 1 de la définition) partant d'un état hybride h et arrivant à un état hybride h' correspond à la traversée d'un état discret. Dans le cas où il existerait un élément dans l'ensemble $first(h)$ (1.a), alors il existe une durée de transition non nulle (1.a.i), l'état discret d'arrivée reste le même que celui de départ (1.a.ii), mais les parties fractionnaires sont mises à jour (1.a.iii et 1.a.iv). Dans le cas où il n'existerait pas d'élément dans $first(h)$ (1.b), les variables dans $slide(h)$ atteignent leur bord (1.b.i) et les autres restent sur place (1.b.ii).

— la transition discrète (cas 2 de la définition) n'est possible que si une variable est sur un des bords de l'état discret de départ (délai de contact nul, 2.b) et que cette variable peut franchir ce bord (2.a). Ainsi, l'état hybride de cette variable est mis à jour (2.c) et les autres restent identiques (2.d).

Graphe d'états hybride

Définition 8. Soit $R = (V, M, E, C)$ un RRG hybride, on note $\mathcal{R} = (H, T)$ le graphe des états hybrides de R avec H l'ensemble des états hybrides et T l'ensemble des transitions. Il existe 2 types de transitions $T = (cT, dT)$: les transitions continues (cT) et discrètes (dT) qui sont définies comme suit.

1. Il existe une unique transition continue dans cT à partir de l'état hybride de départ $h = (\eta, \pi)$ vers l'état hybride d'arrivée $h' = (\eta', \pi')$ donné par :
   (a) Soit $first(h) \neq \emptyset$ et $\exists v \in first(h)$ tel que :
      i. $\delta_h(v) \neq 0$
      ii. $\eta = \eta'$
      iii. $\pi'_u = \begin{cases} 0 & \text{si } C_{u,\rho(h,u),\eta_u} < 0 \\ 1 & \text{si } C_{u,\rho(h,u),\eta_u} > 0 \end{cases}, \forall u \in first(h)$
      iv. $\forall z \in V \setminus first(h), \pi'_z = \pi_z + \delta_h(v) \times C_{z,\rho(h,z),\eta_z}$
   (b) Soit $first(h) = \emptyset$ et l'une des deux conditions suivantes est satisfaite :
      i. $\forall u \in slide(h), \pi'_v = \begin{cases} 0 & \text{si } C_{v,\rho(h,v),\eta_v} < 0 \\ 1 & \text{si } C_{v,\rho(h,v),\eta_v} > 0 \end{cases}$
      ii. $\forall u \notin slide(h), \pi'_u = \pi_u$
2. Il existe une transition discrète dans dT de l'état hybride $h = (\eta, \pi)$ vers l'état $h' = (\eta', \pi')$ si et seulement si il existe une variable $v \in first(h)$ telle que :
   (a) $first(h) \neq \emptyset$
   (b) $\forall v \in first(h), \delta_h(v) = 0$
   (c) $\eta'_v = \eta_v + sgn(C_{v,\rho(h,v),\eta_v})$ et $\pi'_v = \begin{cases} 0 & \text{si } C_{v,\rho(h,v),\eta_v} > 0 \\ 1 & \text{si } C_{v,\rho(h,v),\eta_v} < 0 \end{cases}$
   (d) $\forall u \in V \setminus \{v\}, \eta'_u = \eta_u$ et $\pi'_u = \pi_u$

Une succession de transitions continues et discrètes forme une **trace biologique hybride**[7], que l'on peut définir biologiquement comme la succession des comportements observés à partir d'un état initial jusqu'à un état final du RRGH.

Dans l'exemple de la figure 1.11, selon les célérités choisies, il existe une transition continue de h_0 vers h'_0 dans l'état discret $(0, 0)$: la trajectoire glisse selon la célérité de v_1, car la variable v_2 atteint un mur externe sur son bord inférieur. Elle est suivie d'une transition discrète $h'_0 \to h_1$ permettant de franchir le seuil (il n'y a ni mur interne, ni mur externe) et d'atteindre l'état discret $(1, 0)$. Finalement, une dernière transition continue mène à l'état hybride h'_1.

7. Dans la suite du manuscrit, on utilisera plutôt le mot « trace » ou « trace hybride », par simplification.

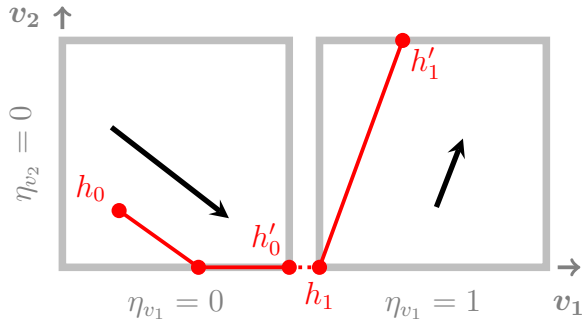


FIGURE 1.11 – Graphe d'états hybrides comprenant des transitions continues (traits rouges pleins) $cT = \{(h_0 \to h'_0), (h_1 \to h'_1)\}$ et discrètes (traits rouges pointillés) $dT = \{(h'_0 \to h_1)\}$.

C'est une trajectoire linéaire par morceau qui peut cependant mener à un comportement non déterministe. En effet, si les célérités génèrent une trajectoire qui atteint un coin d'un état discret, alors plusieurs transitions discrètes sont possibles, voir la figure 1.12. Dans ce cas, l'ensemble $first(h)$ peut contenir plusieurs variables ($|first(h)| > 1$). Bien que ce cas ait une probabilité nulle d'apparaître, s'il intervient, nous choisissons aléatoirement de manière uniforme une des variables présentes dans l'ensemble.

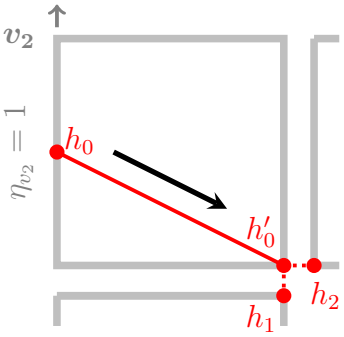


FIGURE 1.12 – Transition discrète non déterministe. Depuis h'_0 deux transitions discrètes sont possibles : $h'_0 \to h_1$ et $h'_0 \to h_2$.

1.3 Instances : réseaux de régulation génétiques hybrides étudiés

Dans cette section, nous présentons les différents RRGHs dont nous serons amenés à trouver une paramétrisation pour l'obtention d'un modèle valide biologiquement.

1.3.1 Cycle négatif (2G)

Nous considérons, dans un premier temps, le graphe d'interaction du réseau à deux variables (2G) de la figure 1.13a. Cet exemple modélise une boucle de rétroaction négative entre les variables v_1 et v_2. Chaque variable possède deux niveaux d'expression : 0 pour

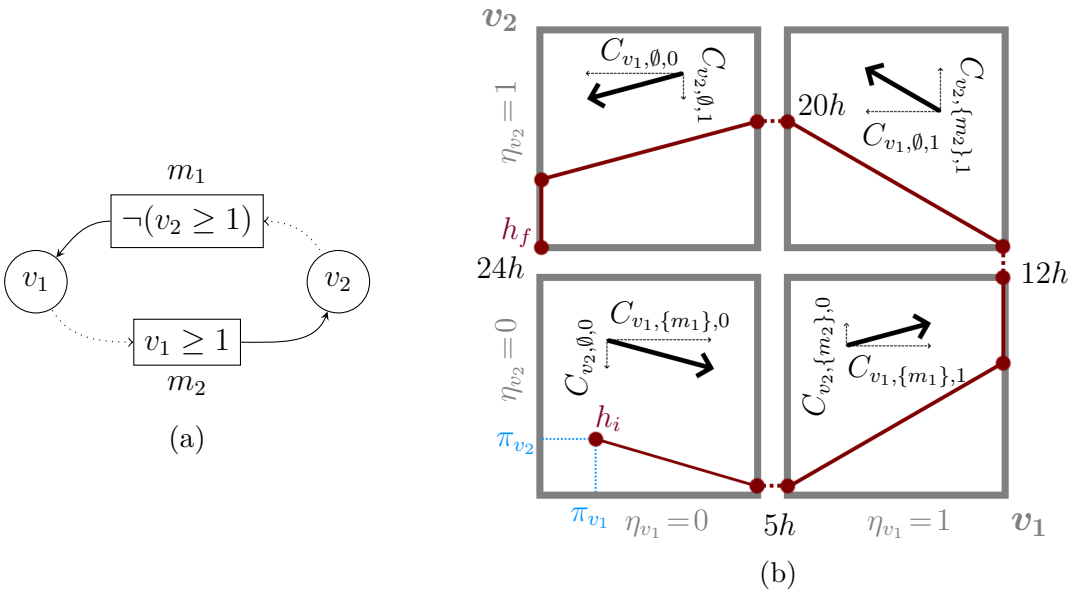


FIGURE 1.13 – Illustration représentant une trace biologique possible dans le graphe d'états hybride (b) construit à partir du graphe d'interaction en (a). Le temps auquel les transitions discrètes ont lieu y est inscrit ($\{5h, 12h, 20h, 24h\}$). Figure extraite de (MICHELUCCI, Jean-Paul COMET et PALLEZ 2022)

l'inactivité, 1 pour l'activité ($b_{v_1} = b_{v_2} = 1$). L'ensemble des prédécesseurs de v_1 est le singleton $\{m_1\}$ et celui de v_2 est $\{m_2\}$. L'ensemble des états discrets possibles est $\mathbb{S} = \{(0,0),(0,1),(1,0),(1,1)\}$. Ces états sont matérialisés dans le graphe d'états hybride illustré en figure 1.13b. Les célérités gouvernant la trace biologique sont :

$$C = \{C_{v_1,\emptyset,0}, C_{v_1,\emptyset,1}, C_{v_2,\emptyset,0}, C_{v_2,\emptyset,1}, C_{v_1,\{m_1\},0}, C_{v_1,\{m_1\},1}, C_{v_2,\{m_2\},0}, C_{v_2,\{m_2\},1}\}$$

Les valeurs des célérités ne sont pas trivialement identifiables. Néanmoins, en faisant des hypothèses sur les interactions du réseau représenté en figure 1.13a, il est possible d'instancier leur signe. Partons de h_i, l'état hybride initial que nous fixons ici arbitrairement. Lorsque v_1 et v_2 sont inactifs ($\eta_{v_1} = \eta_{v_2} = 0$), m_2 n'est pas ressource de v_2 puisque $\eta_{v_1} < 1$, ce qui impliquerait que la concentration de v_2 diminue : $C_{v_2,(0,0)} < 0$. De manière analogue, m_1 est ressource de v_1 signifiant que v_2 n'inhibe pas v_1 et donc, par conséquent, on aurait $C_{v_1,(0,0)} > 0$. Dans l'état suivant ($\eta = (0,1)$), v_1 active v_2, car le multiplexe m_2 est ressource de v_2. Ainsi, $C_{v_1,(0,0)} > 0$ et $C_{v_2,(0,0)} > 0$. En poursuivant ce raisonnement d'état discret en état discret, on obtiendrait bien la dynamique continue par morceaux observée en figure 1.13b.

Si l'on considère maintenant des valeurs de célérités qui respectent les signes déduits, on peut étudier la trace générée. À partir de h_i, la trajectoire continue, matérialisée par un trait rouge plein, mène au bord supérieur de v_1, le bord droit du carré représentant l'état discret $(0,0)$. La transition discrète (instantanée) qui en résulte permet d'atteindre l'état discret suivant (trait rouge pointillé). La transition continue dans ce nouvel état présente un glissement (v_1 atteint un mur externe). Puis, la transition discrète permet d'atteindre l'état discret suivant : $\eta = (1,1)$. L'alternance de transitions continues et discrètes se poursuit jusqu'à atteindre l'état hybride final h_f, arrêt final de la trace, choisi de manière *ad hoc*. Notons que si nous poursuivons cette trajectoire, elle atteindra probablement

un cycle limite, dans lequel la trajectoire repassera par les mêmes transitions continues et discrètes, ou un point fixe duquel elle ne pourra pas s'échapper. Nous verrons dans le chapitre suivant que les connaissances biologiques sur le fonctionnement du RRGH spécifient que la trace doit suivre un cycle périodique d'une durée de 24 heures.

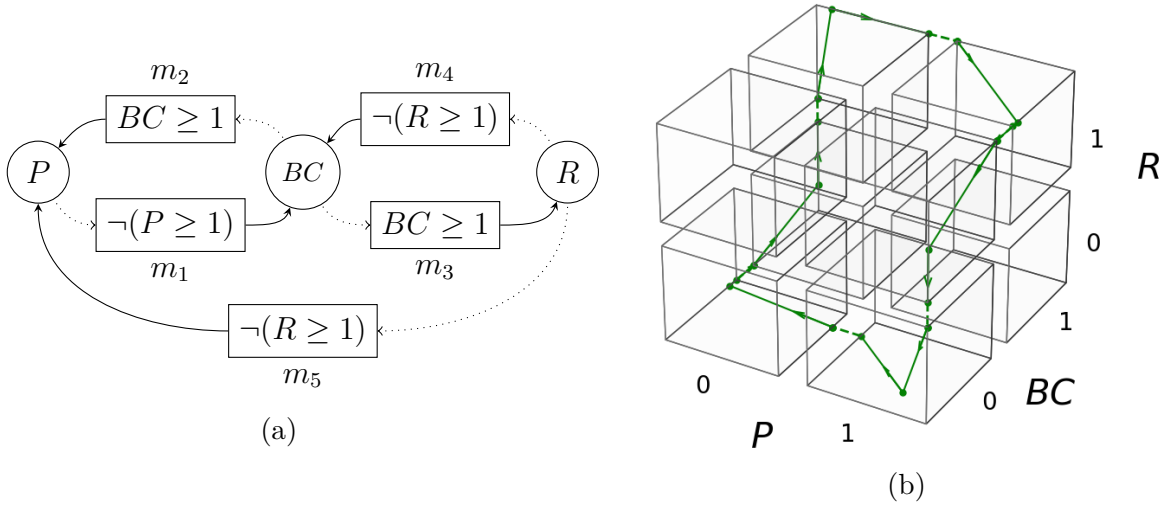
1.3.2 Cycle circadien (3G)

Le rythme circadien englobe tous les rythmes biologiques prenant la forme d'un cycle d'environ 24 heures. C'est en quelque sorte l'horloge interne du corps de nombreux mammifères, comme le corps humain. Il régit un certain nombre de processus physiologiques tels que le sommeil, la régulation de la température corporelle ou encore la sécrétion hormonale. Les rythmes circadiens permettent à l'organisme de maintenir son bon fonctionnement malgré les changements de l'environnement (Zhang et al. 2014). Le mécanisme sous-jacent fonctionne à l'échelle cellulaire et implique des gènes et des protéines horloges. Trois principaux mécanismes moléculaires assurent un comportement oscillatoire. Au cours de la journée, le complexe protéique BMAL1/CLOCK permet la synthèse des protéines du complexe PER/CRY et de la protéine REV-ERBα, qui, en retour, inhibent BMAL1/CLOCK durant la nuit.

Le graphe d'interaction du cycle circadien (3G) utilisé est présenté sur la figure 1.14a et provient de (Jonathan Behaegel 2018). Le réseau est composé de trois variables ($V = \{P, BC, R\}$) : la variable P correspond au complexe PER/CRY, BC au complexe BMAL1/CLOCK et R à la protéine REV-ERBα. Chacune des variables possède un niveau d'expression maximal de 1 ($\forall v \in V, b_v = 1$). L'univers du modèle est donc composé de huit états discrets :

$$\mathbb{S} = \{(0,0,0), (0,0,1), (0,1,0), (0,1,1), (1,0,0), (1,0,1), (1,1,0), (1,1,1)\}$$

Cinq multiplexes sont présents pour conditionner les régulations entre ces trois variables.



(a) (b)

Figure 1.14 – (a) : Graphe d'interaction de l'horloge circadienne à trois variables avec multiplexes, introduit dans (Jonathan Behaegel 2018). (b) : Graphe d'états hybride sur lequel le tracé vert représente une trace biologique possible.

Il y a un total de 20 célérités à identifier. Notons que certaines célérités sont impliquées dans plusieurs états discrets. Nous fournissons la liste des célérités à identifier, par état discret, dans le tableau 1.1.

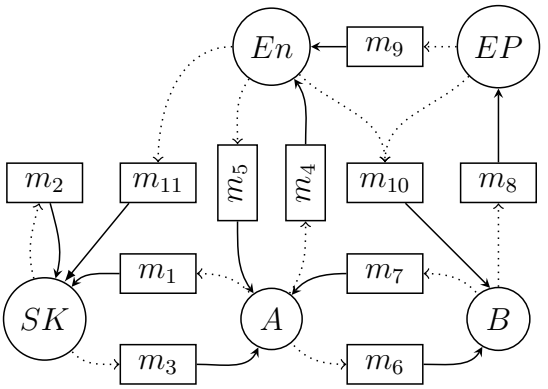
η			C		
P	BC	R	P	BC	R
0	0	0	$C_{P,\{m_5\},0}$	$C_{BC,\{m_1,m_4\},0}$	$C_{R,\{\},0}$
0	0	1	$C_{P,\{\},0}$	$C_{BC,\{m_1\},0}$	$C_{R,\{\},1}$
0	1	0	$C_{P,\{m_2,m_5\},0}$	$C_{BC,\{m_1,m_4\},1}$	$C_{R,\{m_3\},0}$
0	1	1	$C_{P,\{m_2\},0}$	$C_{BC,\{m_1\},1}$	$C_{R,\{m_3\},1}$
1	0	0	$C_{P,\{m_5\},1}$	$C_{BC,\{m_4\},0}$	$C_{R,\{\},0}$
1	0	1	$C_{P,\{\},1}$	$C_{BC,\{\},0}$	$C_{R,\{\},1}$
1	1	0	$C_{P,\{m_2,m_5\},1}$	$C_{BC,\{m_4\},1}$	$C_{R,\{m_3\},0}$
1	1	1	$C_{P,\{m_2\},1}$	$C_{BC,\{\},1}$	$C_{R,\{m_3\},1}$

Tableau 1.1 – La liste des célérités à identifier pour le cycle circadien à trois variables de la figure 1.14.

1.3.3 Cycle cellulaire (5G)

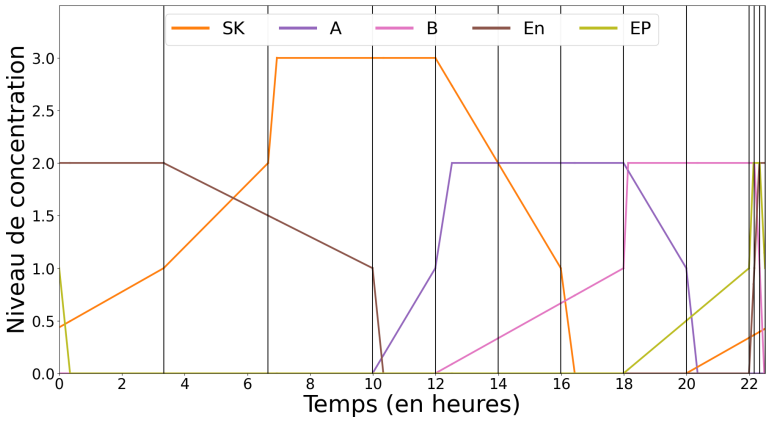
Le cycle cellulaire est le phénomène biologique permettant à une cellule de se reproduire. Ce cycle est composé d'un ensemble de quatre phases. Les trois premières phases sont observées lors de l'étape d'interphase. L'interphase débute lors de la phase *G1* (*Growth 1*) qui assure l'accroissement du volume cellulaire et la préparation à la réplication de l'ADN. Ensuite, lors de la phase *S* (*Synthesis*) a lieu la réplication de l'ADN. Finalement, la phase *G2* (*Growth 2*) continue, si cela est nécessaire, la réplication de l'ADN et maintient la croissance de la cellule pour sa future division. Cette division se produit lors de la dernière phase : la mitose *M*. Elle assure la division de la cellule en deux cellules identiques, préservant les caractéristiques de la cellule mère. Les processus biologiques des phases du cycle cellulaire sont contrôlés par trois familles de protéines : les Cdks (*Cyclin-Dependant Kinases*), les Cyclines et les CKIs (*Cyclin-dependant Kinases Inhibitors*).

Le dernier réseau que nous étudions est un modèle du cycle cellulaire extrait de (Jonathan Behaegel et al. 2016) qui se repose sur ces informations. Son graphe d'interaction est représenté en figure 1.15a dans lequel la variable SK (*Starter Kinase*) représente le complexe CycE/Cdk2, En (*Cdk Enemies*) représente les protéines APC, Wee1 et les CKIs p21 et p22, EP (*Exit Proteins*) représente les protéines APC et Phosphatases, A représente la protéine CycA/Cdk1 et, finalement, B représente la protéine CycB/Cdk1. Les choix de modélisation peuvent être retrouvés dans (Jonathan Behaegel et al. 2016) et (Jonathan Behaegel 2018). Chaque variable possède deux niveaux d'expression (0 et 1) sauf SK dont le niveau *maximum* d'expression est deux ($b_{SK} = 2$). L'univers du modèle est ainsi composé de 48 états discrets que nous ne listerons que partiellement : $\mathbb{S} = \{(0,0,0,0,0), (0,0,0,0,1), ..., (2,1,1,1,1)\}$. Un total de 11 multiplexes conditionnent les régulations de ces cinq variables. Il en résulte 56 célérités distinctes à identifier. Elles sont énumérées dans le tableau 1.2.



$\phi_{m_1} = \phi_{m_4} = \neg(A \geq 1)$
$\phi_{m_2} = SK \geq 1$
$\phi_{m_3} = SK \geq 2$
$\phi_{m_5} = \neg(En \geq 1)$
$\phi_{m_6} = A \geq 1$
$\phi_{m_7} = \neg(B \geq 1)$
$\phi_{m_8} = B \geq 1$
$\phi_{m_9} = EP \geq 1$
$\phi_{m_{10}} = \neg(SK \geq 1)$
$\phi_{m_{11}} = \neg(En \geq 1 \ \vee (EP \geq 1 \wedge En \geq 1))$

(a) (b)



(c)

FIGURE 1.15 – (a) : Graphe d'interaction du cycle cellulaire à cinq variables avec multiplexes, proposé dans (Jonathan BEHAEGEL et al. 2016). (b) : Les formules des multiplexes. (c) Une trace hybride possible représentée par les courbes d'évolution des concentrations des variables au cours du temps. Chaque courbe est associée à une variable et une couleur.

η					C				
SK	A	B	En	EP	SK	A	B	En	EP
0	0	0	1	0	$C_{SK,\{m_1\},0}$	$C_{A,\{m_7\},0}$	$C_{B,\{\},0}$	$C_{En,\{m_4,m_{11}\},1}$	$C_{EP,\{\},0}$
1	0	0	1	0	$C_{SK,\{m_1,m_2\},1}$	$C_{A,\{m_7\},0}$	$C_{B,\{\},0}$	$C_{En,\{m_4\},1}$	$C_{EP,\{\},0}$
2	0	0	1	0	$C_{SK,\{m_1,m_2\},2}$	$C_{A,\{m_3,m_7\},0}$	$C_{B,\{\},0}$	$C_{En,\{m_4\},1}$	$C_{EP,\{\},0}$
2	0	0	0	0	$C_{SK,\{m_1,m_2\},2}$	$C_{A,\{m_3,m_5,m_7\},0}$	$C_{B,\{m_{10}\},0}$	$C_{En,\{m_4\},0}$	$C_{EP,\{\},0}$
2	1	0	0	0	$C_{SK,\{m_2\},2}$	$C_{A,\{m_3,m_5,m_7\},1}$	$C_{B,\{m_6,m_{10}\},0}$	$C_{En,\{\},0}$	$C_{EP,\{\},0}$
1	1	0	0	0	$C_{SK,\{m_2\},1}$	$C_{A,\{m_5,m_7\},1}$	$C_{B,\{m_6,m_{10}\},0}$	$C_{En,\{\},0}$	$C_{EP,\{\},0}$
0	1	0	0	0	$C_{SK,\{\},0}$	$C_{A,\{m_5,m_7\},1}$	$C_{B,\{m_6,m_{10}\},0}$	$C_{En,\{m_{11}\},0}$	$C_{EP,\{\},0}$
0	1	1	0	0	$C_{SK,\{\},0}$	$C_{A,\{m_5\},1}$	$C_{B,\{m_6,m_{10}\},1}$	$C_{En,\{m_{11}\},0}$	$C_{EP,\{m_8\},0}$
0	0	1	0	0	$C_{SK,\{m_1\},0}$	$C_{A,\{m_5\},0}$	$C_{B,\{m_{10}\},1}$	$C_{En,\{m_4,m_{11}\},0}$	$C_{EP,\{m_8\},0}$
0	0	1	0	1	$C_{SK,\{m_1\},0}$	$C_{A,\{m_5\},0}$	$C_{B,\{m_{10}\},1}$	$C_{En,\{m_4,m_9,m_{11}\},0}$	$C_{EP,\{m_8\},1}$
0	0	1	1	1	$C_{SK,\{m_1\},0}$	$C_{A,\{\},0}$	$C_{B,\{\},1}$	$C_{En,\{m_4,m_9,m_{11}\},1}$	$C_{EP,\{m_8\},1}$
0	0	0	1	1	$C_{SK,\{m_1\},0}$	$C_{A,\{m_7\},0}$	$C_{B,\{\},0}$	$C_{En,\{m_4,m_9,m_{11}\},1}$	$C_{EP,\{\},1}$

Tableau 1.2 – La liste des célérités à identifier pour le cycle cellulaire de la figure 1.15.

Représentations graphiques d'une trace. La représentation graphique de la dynamique donnée en figure 1.15c est différente de celles proposées en figure 1.13b et en figure 1.14b. En effet, le graphe d'états hybride est plus difficilement représentable, car le nombre d'états discrets est trop important ($|\mathbb{S}| = 48$) et chaque état discret est un hypercube de dimension cinq. Comme il est complexe de visualiser un hypercube de dimension supérieure à trois, nous avons fait le choix de représenter l'évolution de la concentration de chacune des variables au cours du temps. Les passages de seuils entre deux états discrets sont représentés par les barres noires verticales. Ainsi, en un sens, cette représentation se focalise sur la trace hybride et, de fait, sur les états discrets qu'elle traverse. Les états non traversés par la trace ne sont pas visibles dans cette représentation, contrairement à la figure 1.14b où l'on peut observer que la trace ne passe jamais par l'état discret $(1, 1, 0)$. Cette nouvelle représentation n'illustre donc pas le graphe d'états hybride, mais se concentre sur la trace. Si on représentait la trace de la figure 1.14b sous forme de courbe, on obtiendrait la figure 1.16.

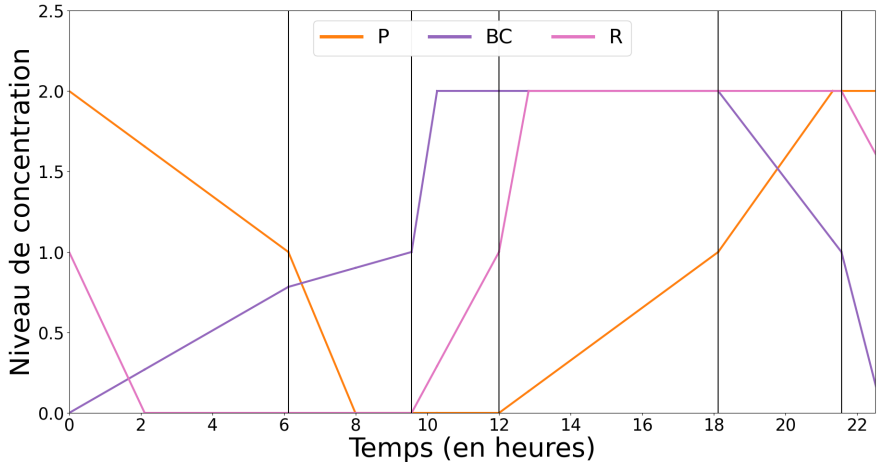


FIGURE 1.16 – Représentation analogue de la trace de la figure 1.14b.

Comme nous venons de le voir dans cette section, les figures 1.13b, 1.14b et 1.15c illustrent une dynamique possible du RRGH sous la forme d'une trace. Ces traces sont toutes obtenues à l'aide d'un simulateur que nous présentons.

1.4 Présentation du simulateur

Réaliser une simulation nécessite de connaître un état hybride initial, composé d'un état discret et des parties fractionnaires de chaque variable du graphe d'interaction, ainsi que de tous les paramètres dynamiques du RRGH. À partir de ces informations, le simulateur calcule l'évolution des concentrations des variables au cours du temps. Cette évolution correspond à la séquence des évènements rencontrés par la trajectoire : il s'agit des états hybrides représentés par les différents points le long des trajectoires de la figure 1.13b et 1.14b. Le simulateur s'arrête lorsque la durée de simulation de la trace (et non du simulateur) atteint le temps fixé par l'utilisateur. Lorsque le simulateur termine, un fichier CSV est écrit. Chaque ligne est composée d'un instant précis t et d'un état hybride. Le tableau 1.3 présente une partie du fichier obtenu à partir de la simulation de la trace de la figure 1.14b.

Temps t	Liste des évènements (h)					
t	η_P	η_{BC}	η_R	π_P	π_{BC}	π_R
0.0	1	0	0	1.0	0.0	1.0
2.109	1	0	0	0.655	0.269	0.0
6.12	1	0	0	0.0	0.783	0.0
6.12	0	0	0	1.0	0.783	0.0
7.991	0	0	0	0.0	0.901	0.0
9.559	0	0	0	0.0	1.0	0.0
...	...	...	...	...	...	...
21.56	1	0	1	1.0	1.0	1.0
22.713	1	0	1	1.0	0.0	0.527
24.0	1	0	1	1.0	0.0	1.0

Tableau 1.3 – Extrait du fichier CSV permettant de visualiser la trace de la figure 1.14b.

Au temps $t = 0.0$, l'état hybride initial h_i de la trace est localisé dans l'état discret $\eta = (\eta_P, \eta_{BC}, \eta_R) = (1, 0, 0)$ et à la position indiquée par les parties fractionnaires $(\pi_P, \pi_{BC}, \pi_R) = (1.0, 0.0, 1.0)$. La ligne suivante indique qu'au bout de $t = 2.109$ (environ 2 heures et 7 minutes), la variable R atteint son bord inférieur (sa partie fractionnaire π_R passe de 1.0 à 0.0) et glisse jusqu'à $t = 6.12$ (sa partie fractionnaire reste inchangée). À ce même moment, la variable P atteint son bord inférieur (la partie fractionnaire atteint $\pi_P = 0.0$). Ensuite, sans que le temps ne change, P franchit son seuil et voit son niveau d'expression diminuer (de 1 à 0) et sa partie fractionnaire augmenter (de 0.0 à 1.0). La trace atteint ainsi le nouvel état discret $\eta = (0, 0, 0)$. Jusqu'à $t = 7.991$, la variable R continue de glisser sur son bord inférieur ($\pi_R = 0.0$), puis la variable P atteint aussi son bord inférieur ($\pi_P = 0.0$) à $t = 7.991$. Au final, à $t = 9.559$, P et R continuent de glisser jusqu'à ce que BC atteigne son bord (la trace atteint un coin du cube, de l'état discret). La trace continue d'évoluer au cours du temps jusqu'à ce qu'en $t = 24$ heures elle atteigne l'état hybride final qui est identique à l'état hybride initial.

Les informations contenues dans ce fichier sont utilisées pour générer une représentation graphique de la dynamique du RRGH mais aussi, comme nous le verrons plus tard lors des expérimentations numériques, pour évaluer la « qualité » d'une paramétrisation donnée.

1.5 Synthèse

Le cadre applicatif de cette thèse concerne l'étude des systèmes biologiques, plus particulièrement les réseaux de régulation génétiques. Pour modéliser la dynamique d'un RRG, il est nécessaire de s'inscrire dans un cadre de modélisation, qui représente sa dynamique à un niveau d'abstraction donné. Nous utilisons un cadre de modélisation hybride provenant du formalisme discret de René Thomas avec multiplexes. Nous focalisons notre étude sur trois RRGHs : le cycle négatif, le cycle circadien et le cyle cellulaire.

Les paramètres qui régulent la dynamique du réseau doivent être identifiés à partir des informations biologiques disponibles afin de fournir un modèle hybride valide. La démarche consiste à trouver une paramétrisation, à obtenir la trace associée à l'aide d'un

simulateur et à la confronter avec les connaissances biologiques. Pour ce faire, il est nécessaire de se reposer sur des informations biologiques tirées d'expériences menées en laboratoire et de la littérature scientifique. Contrairement à de nombreux travaux (J. Sun, Garibaldi et Hodgman 2012 ; Buchet et al. 2021 ; H. Sun 2023), nous n'utilisons pas de données brutes, car elles sont trop souvent bruitées et lacunaires. En effet, si l'on considère le cas des données séquencées de l'ADN et de l'ARN, les technologies de séquençage utilisées possèdent de nombreux défauts, comme la génération d'erreurs de séquençage (Le et al. 2013), de biais liés à la longueur des transcriptions (Oshlack et Wakefield 2009) et de fragmentation (Tuerk, Wiktorin et Güler 2017). Bien qu'il existe de nombreuses avancées pour réduire ces problèmes, il reste néanmoins de nombreux défis (Han et al. 2015). Nous préférons laisser ce travail aux spécialistes d'analyse de données biologiques pour se focaliser plutôt sur la démarche de modélisation.

Nous avons fait le choix de travailler avec des informations interprétées, analysées et formalisées par des experts en biologie. Ces informations à notre disposition se présentent sous la forme d'une succession d'évènements qui spécifient par exemple qu'une variable change de niveau d'expression au bout d'un certain temps. Ces spécifications sont à rapprocher des traces de Hoare pour la vérification de programme informatique. Cette approche formelle repose bien évidemment sur des informations biologiques et sera l'objet du prochain chapitre.

2

Connaissance biologique basée sur la logique de Hoare

The most important property of a program is wether it accomplishes the intentions of its user.

Extrait de (Hoare 1969)

Le cadre de modélisation sur lequel repose notre recherche de paramètres dynamiques est issu de la formalisation discrète de René Thomas. Un des avantages de son utilisation est de pouvoir développer des méthodes formelles pour raisonner sur le comportement d'un modèle. La connaissance biologique utilisée pour valider le modèle est extraite de traces expérimentales observées en laboratoire, mais ces dernières ne sont pas exploitées telles quelles. En effet, elles sont analysées et formalisées par l'expertise biologique sous la forme d'une séquence d'informations. Elles présentent ainsi l'avantage d'être « plus matures » en ce sens qu'elles proviennent d'une méta-analyse des données et des connaissances expertes. Cependant, elles sont moins précises, car certaines informations sont relaxées en raison d'un manque de consensus, par exemple. À partir de cette extraction d'informations biologiques, la logique de Hoare (Hoare 1969), méthode formelle permettant de raisonner sur la correction des programmes informatiques, a été adaptée d'abord dans le cadre de modélisation discret (Bernot, Jean-Paul Comet et Roux 2015), puis dans le cadre hybride (J. Behaegel, J.-P. Comet et Folschette 2017) dans le but de contraindre les paramètres dynamiques.

L'enjeu de ce chapitre est d'exposer brièvement, dans un premier temps, comment la logique de Hoare, introduite en section 2.1, a été adaptée pour la vérification de modèles hybrides (section 2.2). Nous définissons ainsi le formalisme dans lequel est exprimé l'ensemble des informations relatives à la connaissance biologique d'un RRGH. À la suite de quoi, nous mettons en lumière la connaissance biologique dont nous disposons pour les trois RRGHs à l'étude (section 2.3). Dans la section 2.4, nous nous attardons sur la manière dont cette formalisation a mené, lors d'un travail antérieur à cette thèse, au développement d'une première approche visant à résoudre un problème de satisfaction de contraintes. Émergeant de cette précédente tentative, la problématique de cette thèse est finalement énoncée.

2.1 Vérification de programme

Charles Antony Richard Hoare définit en 1969 une méthode formelle permettant de raisonner sur la correction de programmes informatiques impératifs : la **logique de Hoare**. Cette méthode vise à démontrer par induction sur la structure du programme informatique sa correction. Démontrer la correction du programme consiste à prouver que le programme termine et qu'il retourne un résultat satisfaisant la spécification énoncée. La logique de Hoare possède des règles d'inférence pour les instructions syntaxiques comme l'assignation, la conséquence (si... alors... sinon) ou encore l'itération (tant que). À chaque instruction correspond une règle de la logique de Hoare qui permet de vérifier formellement un programme sans avoir à l'exécuter.

Le raisonnement repose sur la notion de **triplet de Hoare** noté $\{P\}\ Q\ \{R\}$ (*cf.* définition 9) qui peut être interprété comme suit : « Si l'assertion P est vraie avant d'initier un programme Q, alors l'assertion R sera vraie à la fin de l'exécution du programme. ».

Triplet de Hoare

Définition 9. Un triplet de Hoare est de la forme $\{P\}\ Q\ \{R\}$ où :

- P et R sont des prédicats décrivant respectivement les conditions des états initiaux et finaux du système (de la mémoire dans le cas des programmes impératifs).
- Q est un programme.

En règle générale, le prédicat P est appelé **précondition** et décrit l'ensemble des états admissibles en entrée du programme Q. Le prédicat R est appelé **postcondition** et décrit les propriétés du système lorsque l'exécution de Q est terminée.

Ainsi, un triplet de Hoare est dit correct lorsqu'en partant d'un état du système satisfaisant la précondition P, l'exécution du programme Q mène à un état du système satisfaisant la postcondition R. Et ce sont les règles d'inférence qui permettent de prouver la correction du programme.

Exemple 2. Prenons l'exemple de la règle de composition qui s'applique pour deux programmes Q_1 et Q_2 s'ils sont exécutés séquentiellement (ici, Q_1 est exécuté avant Q_2). La règle stipule que "Si $\{P\}\ Q_1\ \{R_1\}$ et $\{R_1\}\ Q_2\ \{R_2\}$, alors $\{P\}\ Q_1; Q_2\ \{R_2\}$. Si on considère maintenant les triplets suivants :

$$\{x := 10\}\ y := x + 31\ \ \{y := 41\}$$

$$\{y := 41\}\ z := y + 1\ \{z := 42\}$$

D'après, la règle de composition, on peut écrire :

$$\{\underbrace{x := 10}_{P}\}\ \underbrace{y := x + 31, z := y + 1}_{Q}\ \{\underbrace{z := 42}_{R}\}$$

Edsger Wybe Dijkstra définit en 1975 une sémantique des langages impératifs à l'aide d'un transformateur de prédicats appelé *weakest precondition* (Dijkstra 1975), ou « plus

faible précondition » en français : à chaque instruction élémentaire, est affectée une transformation de prédicats. L'idée de la plus faible précondition est de construire itérativement à partir de la spécification du résultat attendu, exprimée sous la forme de postcondition, la plus faible précondition. Cette plus faible précondition exprime les propriétés minimales sur l'état initial du système, *i.e.*, avant exécution du programme, pour que le programme mène au résultat attendu. Avec le calcul de la plus faible précondition, on s'assure que si le programme Q termine, alors il termine dans un état satisfaisant R.

La plus faible précondition, notée $wp(Q, R)$ est le résultat d'une fonction qui associe à chaque postcondition R la plus faible précondition sur l'état initial assurant que l'exécution de Q termine dans un état final satisfaisant R. Pour construire cette plus faible précondition, la **stratégie backward** est un processus itératif qui permet de déterminer la plus faible précondition d'un programme séquentiel formé d'une succession d'instructions élémentaires. L'idée est d'appliquer une transformation de prédicats sur la postcondition R pour déterminer la plus faible précondition avant la dernière instruction et d'itérer sur chacune des instructions élémentaires jusqu'à la première itération du programme Q.

Exemple 3. Prenons l'exemple du programme impératif qui consiste à permuter les valeurs de deux variables (x et y) en utilisant une autre variable temporaire (aux). Étant donnés le programme $Q = \left\{ \begin{array}{l} aux := x \\ x := y \\ y := aux \end{array} \right\}$ et la postcondition $R = \left\{ \begin{array}{l} x := 4 \\ y := 2 \end{array} \right\}$, on a :

$$\left\{ \begin{array}{l} x := 2 \\ y := 4 \end{array} \right\} aux := x \left\{ \begin{array}{l} aux := 2 \\ x := 4 \end{array} \right\} ; x := y \left\{ \begin{array}{l} aux := 2 \\ y := 4 \end{array} \right\} ; y := aux \left\{ \begin{array}{l} x := 4 \\ y := 2 \end{array} \right\}$$

Finalement, on obtient :

$$\underbrace{\left\{ \begin{array}{l} x := 2 \\ y := 4 \end{array} \right\}}_{wp(Q,R)} \underbrace{aux := x \,;\, x := y \,;\, y := aux}_{Q} \underbrace{\left\{ \begin{array}{l} x := 4 \\ y := 2 \end{array} \right\}}_{R}$$

Comme nous venons de le décrire, la logique de Hoare est initialement développée pour les programmes impératifs, mais elle a aussi été adaptée pour la modélisation des comportements biologiques. Dans ce contexte, le programme Q est remplacé par la trace biologique et les pré et postconditions sont les propriétés biologiques observées sur les états du système biologique au début et à la fin de l'expérience.

2.2 Logique de Hoare génétiquement modifiée pour le cadre de modélisation hybride

(Bernot, Jean-Paul Comet et Roux 2015) définit une version modifiée de la logique de Hoare, appelée « logique de Hoare génétiquement modifiée », adaptée au cadre de modélisation discret de Thomas. Les auteurs introduisent (i) un langage d'assertion

qui exprime les propriétés des états et (ii) un langage de spécification de trace qui exprime les propriétés sur les transitions des états. De manière intuitive, les passages des seuils des produits des gènes sont assimilés à des instructions. Ainsi, une trace discrète est interprétée comme une suite d'instructions, *i.e.*, un programme. Les prédicats (précondition et postcondition) sont les conditions sur les états initiaux et finaux de la trace observée.

Exemple 4. Reprenons l'exemple du cycle négatif à deux variables (se référer au cycle négatif présenté en section 1.3.1), dans lequel la trace discrète observée suit le cycle : $(0,0) \to (1,0) \to (1,1) \to (0,1) \to (0,0)$. La spécification de la connaissance biologique de ce cycle est la suivante :

$$\underbrace{\left\{\begin{array}{l} v_1 := 0 \\ v_2 := 0 \end{array}\right\}}_{P} \underbrace{v_1+;\ v_2+;\ v_1-;\ v_2-}_{Q} \underbrace{\left\{\begin{array}{l} v_1 := 0 \\ v_2 := 0 \end{array}\right\}}_{R}$$

La stratégie *backward* de Dijkstra est utilisée de la même manière et permet de contraindre les paramètres dynamiques K afin que le système puisse être compatible avec la spécification donnée (le triplet de Hoare). La logique de Hoare génétiquement modifiée a été prouvée :

— correcte. On s'assure que si la stratégie *backward* de Dijkstra prouve que le triplet est correct, alors en partant d'un état satisfaisant la précondition, la trace est possible et mène à un résultat satisfaisant la postcondition ;

— complète. Si un triplet de Hoare est correct, alors il existe une preuve de ce triplet avec les règles d'inférence introduites. Autrement dit, si un triplet de Hoare est correct, alors il existe un arbre de preuve formé d'arêtes représentant les règles d'inférence de la logique et de nœuds identifiant les déductions logiques. La racine de cet arbre est le triplet de Hoare.

De manière analogue, (J. Behaegel, J.-P. Comet et Folschette 2017) conçoit une nouvelle version de la logique de Hoare génétiquement modifiée pour le cadre hybride. Le triplet de Hoare est renommé $\{Pre\}\ Q\ \{Post\}$. Les pré et postconditions, représentant des conditions sur l'ensemble des états initiaux et finaux de la trace, sont définies par le **langage de propriétés** tandis que le chemin Q, modélisant la trace, est exprimé dans le **langage de chemins**.

2.2.1 Langage de propriétés

Le langage de propriétés définit les propriétés des états hybrides du système qui pourront être utilisées pour spécifier la précondition Pre et la postcondition $Post$ (définition 10). Chaque propriété est décrite par un couple (D, H) où D correspond à une condition discrète et H à une condition hybride. La condition discrète D définit les formules composées des termes discrets d'un état hybride $h = (\eta, \pi)$. Autrement dit, la condition D porte exclusivement sur les niveaux d'expression des variables de h. Quant à la condition hybride H, elle porte sur les célérités des variables C_{v,ω,η_v}, leurs parties fractionnaires π dans h et le délai de transition continue, noté Δt. Un état hybride h satisfait une propriété $\phi = (D, H)$ si et seulement si D et H sont vérifiées dans h, on le note $h \models (D, H)$.

Langage de propriétés $\mathcal{L}_P$

Définition 10. Les termes du langage de propriétés, noté $\mathcal{L}_p$, sont définis inductivement comme suit :

- Un terme discret est une variable η_v avec $v \in V$ ou une constante de $\mathbb{N}$;
- Un terme continu est une variable π_v avec $v \in V$ ou une célérité C_{v,ω,η_v} avec $v \in V, \omega \in R^-(v)$ et $\eta_v \in [\![1, n]\!]$ ou une constante de $\mathbb{R}$;
- les connecteurs arithmétiques $\{+, -, \times, /\}$ créent des nouveaux termes par composition. / est seulement défini sur les termes continus.

Les atomes sont de la forme $t \diamond t'$ avec t et t' des termes et $\diamond$ les comparateurs $\{<, >, \neq, =, \leq, \geq\}$. Lorsque les termes sont discrets (resp. continus), l'atome est discret (resp. continu).
Les conditions discrètes sont définies par $D := a_d | \neg D \mid D \wedge D \mid D \vee D$ avec a_d un atome discret. Les conditions hybrides sont définies par $H := a_d \mid a_c \mid \neg H \mid H \wedge H \mid H \vee H$ avec a_d un atome discret et a_c un atome continu.
Une propriété est un couple (D, H) formé par une condition discrète D et une condition hybride H. L'ensemble des propriétés forme le langage de propriétés $\mathcal{L}_P$.

La figure 2.1 illustre les termes discrets et continus dans l'état discret $(0, 0)$. Il y a deux types de termes : les termes discrets, en bleu, sont les niveaux d'expression des variables de l'état discret courant (η_{v_1} et η_{v_2}) et les termes continus, en vert, sont le délai de transition (Δt), les parties fractionnaires des états hybrides ($\{\pi^0_{v_1}, \pi^0_{v_2}, \pi^1_{v_1}, \pi^1_{v_2}\}$), et les célérités (représentées par le vecteur). La transition continue, de h_0 à h_1 est représentée en trait plein rouge et la transition discrète, de h_1 à h'_1, est représentée en trait pointillé rouge. La condition discrète D, qui spécifie l'état discret courant, est composée des atomes discrets de $\mathcal{L}_P$: $D \equiv (\eta_{v_1} = 0) \wedge (\eta_{v_2} = 0)$, tandis que la condition hybride H, qui spécifie la position précise de h_0 dans l'état discret ainsi que la transition continue de h_0 à h_1, est composée des atomes continus : $H \equiv (\pi^0_{v_1} = 1.0) \wedge (\pi^0_{v_2} \approx 0.25) \wedge (\pi^1_{v_1} = 0.0) \wedge (\pi^1_{v_2} = 0.5) \wedge (\Delta t = 5.0) \wedge (C_{v_1,\omega,0} > 0) \wedge (C_{v_2,\omega,0} < 0)$.

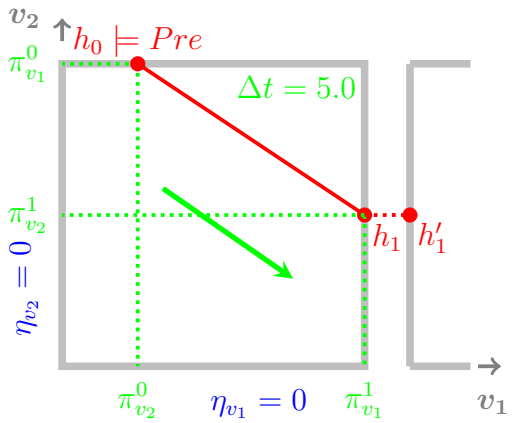


FIGURE 2.1 – Le langage de propriétés $\mathcal{L}_p$ permet de décrire la transition $(h_0 \to h_1)$ à l'intérieur d'un état discret (ici, $(0, 0)$).

2.2.2 Langage de chemins

Le langage de chemins $\mathcal{L}_c$ permet, quant à lui, de modéliser la trace hybride. Rappelons que cette trace joue le rôle du programme Q et décrit un comportement biologique comme une succession d'événements arrivant à des instants précis, que l'on nomme **chemin**. Le chemin vide, noté ϵ, dénote l'absence d'observation d'événement, contrairement au **chemin élémentaire**, noté $(\Delta t, a, dpa)$, qui représente l'observation d'un évènement isolé. Cet évènement isolé est constitué d'une transition continue et d'une transition discrète. Ce triplet de connaissances exprime les observations suivantes :

— la transition continue passe Δt temps dans l'état discret courant,

— des comportements de glissement de variables (la trajectoire glisse le long d'un mur lorsqu'elle fait face à un mur, se référer à la définition 6 dans la section 1.2.2.2) sont observés (a), et

— une transition discrète a lieu (*dpa* pour *discrete path atom*) à la fin de la transition continue.

Un chemin non élémentaire est composé d'une séquence de chemins élémentaires, décrivant un ensemble d'observations au cours du temps.

Le langage de chemin est formellement défini en définition 13. Nous présentons d'abord la transition discrète définie par un **atome de chemin discret** (définition 11). Puis, nous introduisons le comportement observé correspondant à la transition continue est décrit par le **langage d'assertion** $\mathcal{L}_A$ (définition 12).

Atome de chemin discret *dpa*

Définition 11. Un atome de chemin discret ou *dpa* est défini par :

$$dpa := v+ \mid v-$$

où $v \in V$ est le nom d'une variable du graphe d'interaction. Pour tout état $h = (\eta, \pi)$ et $h' = (\eta', \pi')$, la transition $h \xrightarrow{v+} h'$ (resp. $h \xrightarrow{v-} h'$) est satisfaite si et seulement si il existe une transition discrète de h vers h' tel que $\eta'_v = \eta_v + 1$ (resp. $\eta'_v = \eta_v - 1$).

La transition discrète est décrite par l'atome de chemin discret, noté *dpa*. Si le *dpa* spécifié est $v+$, alors la variable v est située sur son bord supérieur et le niveau d'expression de v peut être incrémenté. Au contraire, $v-$ désigne que v franchit son bord inférieur et que le niveau d'expression de v est décrémenté.

Langage d'assertion $\mathcal{L}_A$

Définition 12. Le langage d'assertion, noté $\mathcal{L}_A$, est défini par la grammaire suivante :

$$a :== \top \mid C_{v,\omega,\eta_v} \diamond const \mid slide^+(v) \mid slide^-(v)$$

$$\mid noslide(v) \mid noslide^+(v) \mid noslide^-(v) \mid \neg a \mid a \wedge a \mid a \vee a$$

où $v \in V$, $\omega \in R^-(v)$, $\eta_v \in [\![1, b_v]\!]$, $\diamond$ représente les comparateurs classiques et

const $\in \mathbb{R}$. Si l'assertion indique $a \equiv \top$, alors aucune information sur le comportement à l'intérieur de l'état discret courant n'est connue.
Un couple $(\Delta t, a) \in \mathbb{R}^+ \times \mathcal{L}_A$ est appelé couple d'assertion.

Les comportements observés par une transition continue sont représentés par le langage d'assertion $\mathcal{L}_A$ et définis selon la définition 12. L'assertion a traite des informations sur la dynamique de la transition continue « à l'intérieur » de l'état discret courant de la trace hybride :

- le terme $C_{v,\omega,\eta_v} \diamond c$ contraint la célérité de v dans l'état discret courant à être comparée à la constante c selon $\diamond$;
- le terme $slide^+(v)$ (resp. $slide^-(v)$) exprime le glissement de la variable v sur le bord supérieur (resp. inférieur) de l'état discret courant : v fait face à un mur interne ou externe ;
- inversement, $noslide^+(v)$ (resp. $noslide^-(v)$) empêche v d'atteindre son bord supérieur (resp. inférieur) avant que la variable du *dpa* atteigne le sien : il n'y a pas de glissement observé lors de la transition continue ;
- finalement, $slide(v)$ (resp. $noslide(v)$) désigne la disjonction (resp. conjonction) de $slide^+(v)$ et $slide^-(v)$ (resp. $noslide^+(v)$ et $noslide^-(v)$), se référer aux équations 2.1 et 2.2.

$$slide(v) = slide^+(v) \vee slide^-(v) \tag{2.1}$$
$$noslide(v) = noslide^+(v) \wedge noslide^-(v) \tag{2.2}$$

La figure 2.1 traduit un comportement représenté par $noslide(v_2)$. En effet, v_2 ne glisse ni le long de son bord supérieur ni le long de son bord inférieur et v_1, la variable du *dpa*, est la première à atteindre son bord supérieur.

Langage de chemin $\mathcal{L}_C$

Définition 13. Le langage de chemin, noté $\mathcal{L}_C$, est défini par :

$$Q :== \epsilon \mid (\Delta t, a, v\pm) \mid Q\,;\,Q$$

où ϵ est le chemin vide, $(\Delta t, a)$ est un couple d'assertion et $v\pm$ est un atome de chemin discret. La sémantique d'un chemin Q est donnée par la relation $\xrightarrow{Q}$ entre deux états hybrides définis inductivement par :

- Si $Q = \epsilon$, alors $h_1 \xrightarrow{Q} h_2$ si et seulement si $h_1 = h_2$.
- Si $Q = (\Delta t, a, v\pm)$, alors $h_1 \xrightarrow{Q} h_2$ si et seulement s'il existe un état h_1' tel qu'il existe une transition continue $h_1 \xrightarrow{(\Delta t, a)} h_1'$ suivie d'une transition discrète $h_1' \xrightarrow{v\pm} h_2$.
- Si $Q \equiv Q_1; Q_2$ alors $h_1 \xrightarrow{Q} h_2$ si et seulement s'il existe un état hybride h_3 tel que $h_1 \xrightarrow{Q_1} h_3$ et $h_3 \xrightarrow{Q_2} h_2$.

Un chemin contenant uniquement $(\Delta t, a, v\pm)$ est appelé un chemin élémentaire.

En résumé, le chemin de la figure 2.1 est décrit par le chemin élémentaire suivant : $(5.0, noslide(v_2), v_1+)$. Le couple d'assertion $(\Delta t, a)$ indique bien que la transition continue met cinq heures pour aller de h_0 à h_1 et qu'aucun glissement de v_2 n'est observé ni sur le bord supérieur, ni sur le bord inférieur de l'état discret. L'atome de chemin discret valide que la transition discrète observée $(h_1 \rightarrow h_1')$ est faite sur le bord supérieur de v_1 (le niveau d'expression de v_1 augmente).

2.2.3 Triplet de Hoare hybride

À partir du langage de propriétés $\mathcal{L}_P$ et du langage de chemin $\mathcal{L}_C$, la définition des triplets de Hoare hybrides génétiquement modifiés est donnée en définition 14.

Triplet de Hoare hybride

Définition 14. Un triplet de Hoare hybride pour un RRG hybride est une expression de la forme $\{Pre\}\ Q\ \{Post\}$ où Pre et $Post$, appelés respectivement précondition et postcondition, sont des propriétés de $\mathcal{L}_P$, et Q est un chemin de $\mathcal{L}_C$. Un triplet de Hoare est satisfait si et seulement si, pour tout état $h_1 \models Pre$, il existe un autre état h_2 tel que $h_1 \xrightarrow{Q} h_2$ et $h_2 \models Post$.

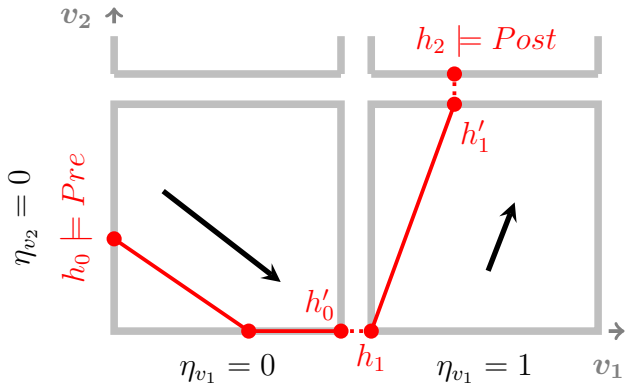


FIGURE 2.2 – Illustration d'une trace satisfaisant le triplet de Hoare hybride : $\{Pre\}\ Q\ \{Post\}$ avec $Pre = (D_0, H_0)$, $Post = (D_2, H_2)$ et $Q \equiv (\Delta t_1, slide^-(v_2), v_1+); (\Delta t_2, noslide(v_1), v_2+)$.

La figure 2.2 représente une trace suivant la connaissance biologique suivante :

$$\underbrace{\begin{Bmatrix} D_0 \\ H_0 \end{Bmatrix}}_{\{Pre\}} \underbrace{\begin{pmatrix} \Delta t_1 \\ slide^-(v_2) \\ v_1+ \end{pmatrix} ; \begin{pmatrix} \Delta t_2 \\ noslide(v1) \\ v_2+ \end{pmatrix}}_{Q} \underbrace{\begin{Bmatrix} D_2 \\ H_2 \end{Bmatrix}}_{\{Post\}}$$

où Pre est une propriété caractérisant l'état hybride h_0 et est exprimée selon le couple (D_0, H_0). $Post$ est une propriété caractérisant l'état hybride h_2 et est composée d'une condition discrète D_2 et d'une condition hybride H_2. À partir de l'état hybride $h_0 \models (D_0, H_0) \equiv Pre$, une transition continue $h_0 \rightarrow h_0'$ de durée Δt_1 glissant sur le bord inférieur de v_2 est suivie immédiatement par une transition discrète $(h_0' \rightarrow h_1)$. Une nouvelle

transition continue $h_1 \rightarrow h_1'$ de durée Δt_2 a ensuite lieu. Finalement, la transition discrète $(h_1' \rightarrow h_2)$ mène à l'état hybride final $h_2 \models (D_2, H_2) \equiv Post$. On observe que h_0 satisfait la précondition, les transitions continues respectent le couple d'assertion, tandis que les transitions discrètes mènent, dans les deux cas, à l'incrément du niveau d'expression de la variable concernée. L'état final h_2 satisfait la postcondition définie par la propriété (D_2, H_2).

2.3 Connaissance biologique utilisée

Pour chaque RRGH présenté dans la section 1.3, nous détaillons ici les connaissances biologiques utilisées, provenant d'expériences biologiques répétées et d'expertise, sous la forme de triplets de Hoare hybrides.

2.3.1 Cycle négatif (2G)

La formalisation des connaissances biologiques du RGGH présentée en section 1.3.1 s'exprime par le triplet de Hoare suivant :

$$\{Pre\} \begin{pmatrix} 5.0 \\ noslide\,(v_2) \\ v_1+ \end{pmatrix}; \begin{pmatrix} 7.0 \\ slide^+\,(v_1) \\ v_2+ \end{pmatrix}; \begin{pmatrix} 8.0 \\ noslide\,(v_2) \\ v_1- \end{pmatrix}; \begin{pmatrix} 4.0 \\ slide^-\,(v_1) \\ v_2- \end{pmatrix} \{Post\} \tag{2.3}$$

avec Pre correspondant à l'état hybride initial h_i et $Post$ correspondant à l'état hybride final h_f et $h_i = h_f = ((0,0)^t, (0.0, 1.0)^t)$.

Notons que pour le reste du manuscrit, nous préférons utiliser pour la notation de la précondition et de la postcondition directement les états hybrides initiaux et finaux : $\{h_i\}$ au lieu de $\{Pre\}$ et $\{h_f\}$ au lieu de $\{Post\}$, dans le but de faire directement référence aux états initiaux et finaux de la trace.

Un modèle est valide si le graphe d'états hybride possède une trace compatible avec le triplet de Hoare fourni en équation (2.3). Cette trace doit :

— débuter dans l'état hybride h_i : $\{h_i\}$,

— passer $\Delta t_1 = 5$ heures dans l'état discret de h_i, ne pas glisser sur un bord de v_2 et passer dans l'état discret $(1,0)$: $\begin{pmatrix} 5.0 \\ noslide\,(v_2) \\ v_1+ \end{pmatrix}$;

— passer $\Delta t_2 = 7$ heures dans cet état, glisser sur le bord supérieur de v_1 (il s'agit d'un mur externe, car $\eta_{v_1} = b_{v_1} = 1$) puis se diriger vers une transition discrète vers l'état discret $(1,1)$: $\begin{pmatrix} 7.0 \\ slide^+\,(v_1) \\ v_2+ \end{pmatrix}$;

— passer $\Delta t_3 = 8$ heures dans ce nouvel état discret, ne glisser sur aucun bord de v_2 et passer par une transition discrète à l'état $(0,1)$: $\begin{pmatrix} 8.0 \\ noslide\,(v_2) \\ v_1- \end{pmatrix}$;

— passer $\Delta t_4 = 4$ heures, glisser le long du bord inférieur de v_1 et aller vers l'état discret $(0,0)$: $\begin{pmatrix} 4.0 \\ slide^- (v_1) \\ v_2- \end{pmatrix}$;

— atteindre l'état hybride final h_f : $\{h_f\}$.

La durée totale de ce cycle est de 24 heures.

Un ensemble de traces valides respectant l'équation (2.3) est représenté sur la figure 2.3. Deux traces hybrides compatibles sont représentées, l'une en trait pointillé bleu et l'autre en trait plein rouge. Les zones colorées représentent l'ensemble des possibles. Cet ensemble des possibles dans chaque état discret dépend de l'état hybride de sortie de l'état précédent. Les zones violettes montrent l'intersection des zones bleu et rouge.

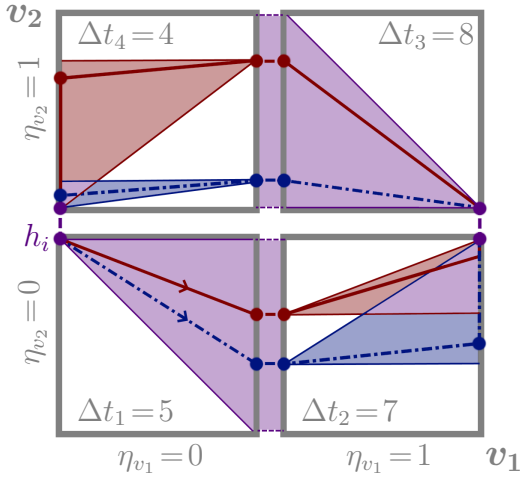


FIGURE 2.3 – Représentation visuelle de l'ensemble des traces hybrides possibles respectant les observations du triplet de Hoare.

2.3.2 Cycle circadien (3G)

La connaissance biologique pour le cycle circadien à trois variables est spécifiée par le triplet de Hoare suivant :

$$\{h_i\} \begin{pmatrix} 6.12 \\ slide^- (R) \\ P- \end{pmatrix}; \begin{pmatrix} 3.44 \\ \top \\ BC+ \end{pmatrix}; \begin{pmatrix} 2.44 \\ slide^+(BC) \wedge slide^-(P) \\ R+ \end{pmatrix}; \begin{pmatrix} 6.12 \\ slide^+ (R) \\ P+ \end{pmatrix}$$

$$\begin{pmatrix} 3.44 \\ \top \\ BC- \end{pmatrix}; \begin{pmatrix} 2.44 \\ slide^- (BC) \wedge slide^+(P) \\ R- \end{pmatrix} \{h_f\}$$

avec $h_i = h_f = \left((\eta_P, \eta_R, \eta_{BC})^t, (\pi_P, \pi_R, \pi_{BC})^t\right) = \left((1,0,0)^t, (1.0, 0.0, 1.0)^t\right)$

Ce triplet de Hoare, introduit dans (Jonathan BEHAEGEL 2018), présente six chemins élémentaires. Le premier indique que la variable P devient inactive ($P-$) au bout de 6 heures et 20 minutes (6.12) pendant lesquelles la variable R est totalement dégradée ($slide^-(R)$). Le second spécifie que la transition continue doit durer 3.44 heures et aboutir

à une transition discrète pendant laquelle le niveau de BC augmente ($BC+$). Il n'y a aucune information sur le comportement de la trace à l'intérieur de l'état discret courant.

En poursuivant les différentes transitions continues et discrètes, la trace revient à son état initial au bout de 24 heures.

2.3.3 Cycle cellulaire (5G)

La spécification de la connaissance biologique pour le cycle cellulaire à cinq variables est formalisée comme suit :

$$\Big\{h_i\Big\} \underbrace{\begin{pmatrix} 3.33 \\ slide^-(EP) \\ SK+ \end{pmatrix}; \begin{pmatrix} 3.33 \\ \top \\ SK+ \end{pmatrix}; \begin{pmatrix} 3.33 \\ slide^+(SK) \\ En- \end{pmatrix}}_{G1}$$

$$\underbrace{\begin{pmatrix} 2.0 \\ slide^-(En) \\ A+ \end{pmatrix}; \begin{pmatrix} 2.0 \\ \top \\ SK- \end{pmatrix}; \begin{pmatrix} 2.0 \\ slide^+(A) \\ SK- \end{pmatrix}; \begin{pmatrix} 2.0 \\ slide^-(SK) \\ B+ \end{pmatrix}}_{S}$$

$$\underbrace{\begin{pmatrix} 2.0 \\ \top \\ A- \end{pmatrix}; \begin{pmatrix} 2.0 \\ slide^+(B) \\ EP+ \end{pmatrix}}_{G2}$$

$$\underbrace{\begin{pmatrix} 0.17 \\ slide^+(EP) \\ En+ \end{pmatrix}; \begin{pmatrix} 0.17 \\ slide^-(A) \wedge slide^+(En) \\ B- \end{pmatrix}; \begin{pmatrix} 0.17 \\ slide^-(B) \\ EP- \end{pmatrix}}_{M}$$

$$\Big\{h_f\Big\}$$

où $h_f = \Big((\eta_{SK}, \eta_A, \eta_B, \eta_{En}, \eta_{EP},)^t, (\pi_{SK}, \pi_A, \pi_{En}, \pi_{EP})^t\Big)$
$= \Big((0,0,0,1,0)^t, (0.5, 0.0, 0.0, 1.0, 1.0)^t\Big)$ et h_f est égal à h_i.

Cette connaissance biologique provient de (Jonathan Behaegel et al. 2016). Elle présente un cycle de 22 heures et 30 minutes (22.5 heures), pendant lesquelles les différentes phases du cycle cellulaire sont respectées. Pour rappel, le cycle cellulaire est découpé en quatre phases. Trois phases sont observées au cours de l'interphase entraînant l'accroissement du volume cellulaire et la duplication des chronomoses (G1, S, G2) et une phase est nécessaire pour la mitose (M) qui assure la division cellulaire. $G1$ dure 10 heures, S dure 8 heures, $G2$ dure 4 heures et M dure 30 minutes. 12 évènements biologiques connus et observés sont représentés par les 12 chemins élémentaires du triplet : trois pour $G1$, quatre pour S, deux pour $G2$ et trois pour M.

On peut analyser par exemple le deuxième chemin élémentaire de la phase $G1$ pour lequel la transition continue passe 3 heures et 20 minutes et se dirige vers une activation de SK atteignant ainsi son niveau d'expression maximal : $b_{Sk} = 2$. Aucune information n'est observée pour les autres variables ; par conséquent, on note l'assertion « $\top$ ».

2.4 Première approche par résolution de contraintes

De précédents travaux ont mené au développement d'une méthode d'automatisation d'identification des paramètres dynamiques dans les RRGHs basée sur la connaissance biologique spécifiée dans la section précédente. De par ses limites, cette méthode permet de faire le lien avec la problématique de la thèse. Nous l'introduisons dans cette section.

2.4.1 Extraction de contraintes avec la plus faible précondition hybride

Le calcul de la plus faible précondition de Dijkstra a aussi été adapté à la transformation de prédicats dans le cadre du formalisme hybride. Il est introduit dans (Jonathan Behaegel 2018). Chaque chemin élémentaire de la connaissance biologique correspond à une transition continue d'une durée Δt ayant un comportement défini par le langage d'assertion $\mathcal{L}_A$ et est suivie d'une transition discrète instantanée représentée par l'atome de chemin discret de $\mathcal{L}_P$.

Dans le cadre hybride, la plus faible précondition est notée $wp(Q, Post) = (D, H)$ et correspond aux contraintes minimales que doivent satisfaire les paramètres dynamiques pour qu'il existe une trace hybride compatible avec les spécifications de Q. Et cette trace doit terminer dans un état hybride final satisfaisant la postcondition $Post$. Le calcul de la plus faible précondition permet donc d'extraire les contraintes que doit satisfaire une paramétrisation pour qu'elle soit valide, c'est-à-dire qu'elle corresponde à la connaissance biologique exprimée sous la forme d'un triplet de Hoare. Le calcul de la plus faible précondition dans le cadre hybride a été prouvé correct et complet inductivement. La preuve de correction montre que s'il existe une preuve que le triplet de Hoare hybride est correct à l'aide de la stratégie *backward*, alors il existe une trace qui mène à un état hybride final satisfaisant la postcondition. La complétude démontre que si un triplet de Hoare hybride est correct, alors il existe une preuve de ce triplet avec les règles d'inférence de la logique.

La construction des contraintes minimales obtenues avec le calcul de la plus faible précondition ne suffit pas à trouver les valeurs des paramètres. Il s'agit en effet d'affecter pour chaque paramètre une valeur qui satisfait à toutes ses contraintes : c'est un **problème de satisfaction de contraintes**, abrégé CSP pour *Constraint Satisfaction Problem*. Pour les RRGHs de petite taille, il est possible de résoudre les contraintes à la main. Cependant, lorsque le nombre de paramètres à identifier augmente, *i.e.*, la dimension de l'espace des paramètres augmente, il est nécessaire d'automatiser ce processus.

Nous n'entrons que sommairement dans les détails techniques mis en œuvre dans ces précédents travaux, car ils ne font pas l'objet du travail de cette thèse. Nous nous intéressons néanmoins à la formulation du CSP, primordiale pour comprendre les enjeux de cette approche par contraintes.

2.4.2 Problème de satisfaction de contraintes

Après un bref rappel de ce qu'est un CSP, nous présentons comment le problème d'identification des paramétrisations des RRGHs a été formulé dans (J. Behaegel, J.-P. Comet et M. Pelleau 2018).

2.4.2.1 Introduction aux CSPs

Problème de satisfaction de contraintes (CSP)

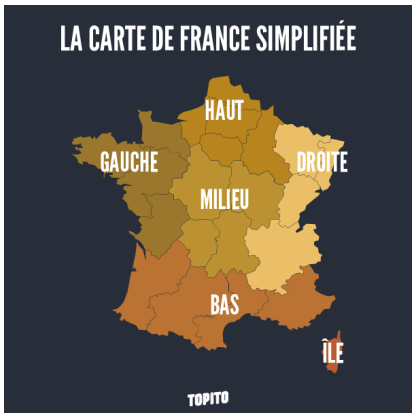
Définition 15. Un CSP (Brailsford, Potts et B. M. Smith 1999) est un problème modélisé par un triplet de trois composantes X, D et C :

- $X = \{X_1, ..., X_n\}$ est l'ensemble des variables du problème à identifier,
- $D = \{D_1, ..., D_n\}$ est l'ensemble des domaines [a] de ces variables ; un pour chaque variable $\forall i \in n, X_i \in D_i$, et
- C est un ensemble de contraintes que doivent satisfaire les variables pour que les valeurs choisies représentent une solution possible.

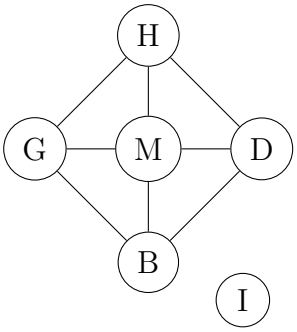
a. Un domaine de X_i est l'ensemble des valeurs que peut prendre X_i.

Un domaine D_i peut être discret ou continu en fonction du type de la variable X_i. Différentes variables peuvent avoir différents domaines de taille distincte. Une contrainte est une relation entre différentes variables, elle possède une arité en fonction du nombre de variables sur lesquelles elle porte (unaire pour 1, binaire pour 2,..., n-aire pour n) et il existe différents types de contraintes en fonction des domaines des variables : numériques, booléennes, etc. Les formules des contraintes sont munies des opérateurs d'égalité, de différence ou d'inégalité. Les contraintes peuvent être linéaires ou non linéaires lorsque les expressions arithmétiques contiennent des produits de variables ou des fonctions exponentielles, par exemple.

Exemple 5. Intéressons-nous à une carte administrative (très simplifiée [1]) de la France représentée en figure 2.4a. Nous souhaitons colorer chaque région en rouge, bleu ou vert, de manière à ce que deux régions voisines n'aient pas la même couleur (attention à ne pas se fier aux couleurs de la carte !).



(a)



(b)

Figure 2.4 – (a) : Coloration de carte de la France formulée comme un problème de satisfaction de contraintes. (b) : Graphe de contraintes représentant le problème de coloration.

1. https://www.topito.com/top-cartes-france-insolites-region-finito

On peut définir le CSP comme le triplet (X, D, C) suivant :

— $X = \{H, M, B, G, D, I\}$ avec H pour HAUT, M pour MILIEU, B pour BAS, G pour GAUCHE, D pour DROITE et I pour ÎLE ;

— chaque variable a pour domaine l'ensemble $D_i = \{rouge, vert, bleu\}$;

— les contraintes exigent des couleurs distinctes pour les régions voisines. Il y a huit contraintes : $C = \{G \neq H, G \neq M, G \neq B, H \neq M, H \neq D, M \neq D, M \neq B, D \neq B\}$.

La résolution d'un CSP consiste à affecter une valeur à chaque variable. Une affectation qui ne viole aucune contrainte est dite cohérente, c'est-à-dire que la contrainte est vérifiée pour les valeurs des variables de la contrainte. Une affectation est complète si chaque variable a une affectation. Une solution est une affectation cohérente et complète. Une solution du CSP, présentée en exemple, est $G = rouge$, $H = bleu$, $M = vert$, $B = bleu$, $D = rouge$ et $I = vert$.

2.4.2.2 Formulation du CSP pour les RRGHs

(J. Behaegel, J.-P. Comet et M. Pelleau 2018) formule le problème comme un problème de satisfaction de contraintes (CSP) de la forme (X, D, C) :

— les variables X sont les célérités, les parties fractionnaires des états hybrides d'entrée et de sortie de chacun des états discrets par lesquels passent la trace et le temps passé dans chacun des états ;

— concernant le domaine D des variables, les célérités sont définies sur $\mathbb{R}$, les parties fractionnaires sur $[0, 1]$ et le temps passé sur $\mathbb{R}^*+$;

— les contraintes C sont celles obtenues en appliquant la stratégie *backward* du calcul de la plus faible précondition.

Une solution est une affectation cohérente et complète dépendant de l'ensemble des contraintes extraites. L'application de la stratégie *backward* sur le cycle négatif est détaillée pas à pas en section 4.6 de (Jonathan Behaegel 2018).

2.4.3 Cas d'étude du solveur continu AbSolute.

Le solveur AbSolute (Marie Pelleau et al. 2013) a été utilisé pour résoudre le CSP introduit en section 2.4.2.2. Il s'agit d'un solveur de contraintes linéaires et non linéaires sur les réels qui s'appuie sur la notion de domaines abstraits (P. Cousot et R. Cousot 1992) pour résoudre les contraintes du CSP. Cette hybridation lui permet de gérer des contraintes avec des variables discrètes (problème combinatoire) ou continues (problème continu).

La particularité des solveurs de contraintes sur des problèmes combinatoires est de pouvoir garantir de ne trouver que des solutions (correction) et de n'en omettre aucune (complétude). Dans le cas des problèmes dont le domaine des variables du CSP est continu, il est impossible de maintenir ces deux propriétés pour des raisons numériques qui interviennent dans la résolution des contraintes, comme l'approximation des réels en flottants. Le solveur AbSolute surapproxime l'ensemble des solutions satisfaisant les contraintes des problèmes continus. Ainsi, la complétude (toutes les solutions sont trouvées) est respectée

au détriment de la correction (il y a surapproximation, donc il n'y a pas que des solutions identifiées).

Les solutions d'AbSolute, comme c'est généralement le cas des solveurs continus, sont des produits d'intervalles appelés **boîtes**. Celles ne contenant que des solutions du CSP sont appelées **boîtes sûres**, et celles contenant au moins une solution du CSP sont appelées **boîtes non sûres**. AbSolute propose différentes formes géométriques pour paver l'espace de recherche et la taille des boîtes dépend d'un paramètre de précision fourni au solveur.

La figure 2.5 illustre un CSP continu composé d'une unique contrainte $y \leq x^2$. L'objectif est de paver l'espace de recherche avec des boîtes (ici des rectangles) de manière à obtenir toutes les solutions possibles (complétude). Les boîtes sûres, ne contenant que des solutions du CSP, sont vertes, tandis que les boîtes non sûres, contenant au moins une solution, sont représentées en rose. Les boîtes non sures sont composées des solutions sur et sous la parabole, mais elles comprennent des points de l'espace de recherche qui ne sont pas des solutions : ceux au-dessus de la parabole.

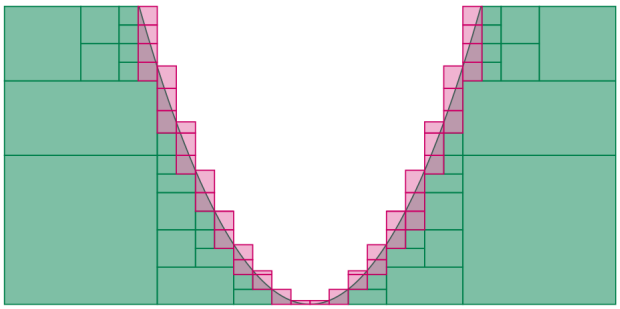


FIGURE 2.5 – Illustration des solutions obtenues par AbSolute sur le CSP continu composé d'une seule contrainte $y \leq x^2$.

L'utilisation du solveur pour notre problématique ne permet pas d'extraire des solutions dans tous les cas. En effet, pour le cycle cellulaire à cinq variables, AbSolute ne renvoie que des boîtes non sûres et une technique d'échantillonnage à la main a dû être développée pour extraire une solution. Cette technique est détaillée dans (Jonathan BEHAEGEL 2018).

Cette limitation est une des raisons pour lesquelles cette thèse a été proposée : nous cherchons à automatiser le processus d'identification des paramètres dynamiques dans son intégralité. Nous énonçons dans la prochaine section la problématique de la thèse.

2.5 Synthèse

La première partie de ce manuscrit s'est attachée à exposer le problème d'identification de paramètres dynamiques dans les RRGHs. Dans le chapitre 1, nous avons décrit la dynamique du cadre de modélisation biologique utilisé, le cadre hybride de R. Thomas. Dans ce second chapitre, nous avons mis en avant le développement de méthodes formelles permettant de modéliser et d'analyser ces systèmes biologiques. Et, dans ce contexte, nous avons introduit la formalisation de la connaissance biologique à l'aide de la logique de Hoare qui, couplée au calcul de la plus faible précondition, a précédemment mené à

l'extraction de contraintes sur les paramètres dynamiques. Nous avons ensuite brièvement évoqué une méthode permettant l'automatisation de la résolution de ces contraintes. Cette recherche de solutions a été réalisée par un solveur de contraintes continu, AbSolute. Ce travail a montré les limites de son efficacité pour les réseaux de taille supérieure ou égale à cinq variables.

Au regard de cette limitation, l'objectif de cette thèse est de développer de nouvelles méthodes d'échantillonnage de l'espace des solutions du problème d'identification des paramètres dynamiques des RRGHs dont la connaissance biologique est formalisée sous la forme d'un triplet de Hoare hybride.

Ce problème est abordé à travers deux paradigmes différents : l'**optimisation** et la **théorie de la décision**. Le problème de contraintes a d'abord été reformulé comme un problème d'optimisation en boîte noire et des techniques d'évolution artificielle (EA) ont été mises en place pour identifier un modèle valide des réseaux étudiés. Le problème a par la suite été envisagé comme un problème de décisions séquentiel où le choix des paramètres dynamiques suit l'hypothèse de Markov. Dans ce contexte, les techniques de recherche arborescente Monte Carlo ont été étudiées.

Ainsi, dans la seconde partie du manuscrit, nous allons partir à la recherche d'une solution au problème qui vient d'être décrit, c'est-à-dire au problème d'identification d'un ensemble de paramètres dynamiques (une paramétrisation), qui permet au modèle d'exhiber des comportements compatibles avec les connaissances et expertises biologiques formalisées par un triplet de Hoare hybride. Pour ce faire, après avoir reformulé le problème dans chacun des paradigmes, nous menons des études expérimentales dans le but de montrer que la reformulation du problème et l'emploi de méthodes adéquates nous permettent bien d'identifier une solution globale, *i.e.*, un modèle valide, pour les RRGHs.

Deuxième partie

Recherche d'une paramétrisation

3

Problème d'optimisation en boîte noire avec l'évolution artificielle

Intelligence can be defined in terms of the capability of a system to adapt its behavior to meet its goals in a range of environments. [...] the process of evolution accounts for such behavior and provides the foundation for the design of artificially intelligent machines.

Extrait de (D. B. Fogel 1995)

L'optimisation, dans son sens commun, consiste à s'engager dans une action pour trouver la meilleure solution à un problème. En mathématiques, cette discipline peut être vue comme l'ensemble des méthodes et techniques analytiques ou numériques de recherche de solutions à des problèmes de minimisation ou de maximisation d'une fonction. Ces problèmes se posent naturellement dans presque tous les domaines de la recherche moderne. L'optimisation joue, entre autres, un rôle clé dans la recherche de solutions aux problèmes de la vie réelle allant de la gestion de projets jusqu'à l'économie en passant par la biologie (Floudas et Pardalos 2008). Pour résoudre le problème étudié dans cette thèse, nous considérons le scénario dit boîte noire (*blackbox*) dans lequel on ne dispose pas de la formulation analytique de la fonction, mais uniquement d'un oracle, un simulateur dans notre cas, permettant de l'évaluer pour guider l'optimisation.

L'objectif de ce chapitre est de reformuler le problème d'identification des paramètres dynamiques d'un RRGH comme un problème d'optimisation et de le résoudre à l'aide de techniques d'optimisation stochastiques, ici de l'évolution artificielle (EA) (section 3.1). Nous présentons ensuite un état de l'art des techniques d'EA (section 3.2), puis nous menons une étude expérimentale sur les trois instances de notre problème (section 3.3). Finalement, dans la section 3.4, nous abordons la question du passage à l'échelle pour identifier une paramétrisation dans des RRGHs de plus grande dimension que ceux étudiés.

3.1 Reformulation du CSP en un problème d'optimisation numérique

Nous commençons par introduire brièvement ce qu'est l'optimisation numérique avant de reformuler le problème d'identification des paramètres dynamiques. Ce dernier a été initialement traité comme un CSP, nous le formulons ici comme un problème d'optimisation (PO) numérique. Ensuite, nous justifions le choix des métaheuristiques d'évolution artificielle comme optimiseur stochastique pour résoudre ce problème.

3.1.1 Introduction à l'optimisation numérique

Dans cette brève introduction, nous présentons une définition du problème d'optimisation et du vocable associé, ainsi que des propriétés permettant de classifier un problème d'optimisation.

3.1.1.1 Définition d'un problème d'optimisation

L'optimisation est l'étude des problèmes de minimisation ou de maximisation d'une fonction f :

Problème d'optimisation

Définition 16. Étant donnée une fonction $f : \mathcal{S} \to \mathbb{R}$, un problème d'optimisation consiste à trouver le ou les éléments de :

— $\mathcal{D} = \underset{x \in \mathcal{S}}{\operatorname{argmin}}\ f(x) := \{x \in \mathcal{S} \mid \forall x' \in \mathcal{S}, f(x) \leq f(x')\}$, pour un problème de minimisation ;

— $\mathcal{D} = \underset{x \in \mathcal{S}}{\operatorname{argmax}}\ f(x) := \{x \in \mathcal{S} \mid \forall x' \in \mathcal{S}, f(x) \geq f(x')\}$, pour un problème de maximisation.

La fonction f, définie sur l'**espace de recherche** $\mathcal{S}$ à valeurs dans $\mathbb{R}$, est appelée **fonction objectif** (ou fonction coût) et associe une valeur à un élément x de l'espace de recherche. En d'autres termes, une fonction objectif est une expression mathématique définissant ce qu'on souhaite optimiser (par exemple, maximiser les bénéfices ou minimiser les coûts). Il est à noter que l'espace de recherche est, dans la plupart des cas, un sous-ensemble de l'espace euclidien $\mathbb{R}^n$ ($\mathcal{S} \subseteq \mathbb{R}^n$) mais il peut être contraint par des bornes (*box-constrained*). x est un élément ou point de l'espace de recherche $\mathcal{S}$ et il est composé de n **variables de décision** : $x = (x_1, ..., x_n) \in \mathcal{S} \subset \mathbb{R}^n$. On les appelle de cette manière car ce sont les composantes du problème sur lesquelles on peut agir. On appelle x le **vecteur de décision**.

Selon cette définition, notons que résoudre un problème de maximisation d'une fonction f est strictement équivalent à résoudre un problème de minimisation de l'opposé de f : $\forall x \in \mathcal{S}, \max_x f(x) = -\min_x(-f(x))$. Cette équivalence signifie que les valeurs optimales sont opposées et que les solutions du problème sont les mêmes. Ainsi, optimiser s'apparente systématiquement à minimiser (ou maximiser) sans perte de généralité. Nous nous focalisons sur les problèmes de minimisation pour le reste du manuscrit : lorsque

cela n'est pas mentionné explicitement, nous traitons le problème d'optimisation comme un problème de minimisation.

Un point x^* est **solution globale** (ou *minimum* global) du problème d'optimisation s'il appartient à $\mathcal{D}$. Plusieurs situations peuvent avoir lieu en fonction du nombre d'éléments dans l'ensemble de solutions globales formé par l'ensemble $\mathcal{D}$. Il peut être :

— un ensemble vide ($\mathcal{D} = \emptyset$) : il n'y a pas de solution globale ;

— un singleton ($|\mathcal{D}| = 1$) : il y a une unique solution globale ;

— un ensemble fini ($|\mathcal{D}| \in \mathbb{N}$) : il y a un nombre fini de solutions globales et il y en a plus qu'une ;

— un ensemble infini ($|\mathcal{D}| = \infty$) : il y a un nombre infini de solutions globales.

Par opposition à la notion de solution *globale*, on appelle **solution locale** (ou *minimum* local) du problème d'optimisation un point de l'espace de recherche dont la valeur de f est minimale dans son voisinage. Cela requiert que $\mathcal{S}$ soit un espace topologique sur lequel est défini un voisinage. Ainsi, x est une solution locale, si x est *minimum* dans son voisinage, *i.e.*, $\exists V(x) \subset \mathcal{S}$ voisinage de x, tel que $x \in \underset{x' \in V(x)}{\operatorname{argmin}} f(x')$.

On peut représenter ces différentes notions à l'aide d'un **paysage de l'espace de recherche**, voir l'illustration de la figure 3.1. Chaque point x possède une « altitude ». Cette dernière est définie par la valeur de la fonction objectif f. L'élévation traduisant la fonction objectif, le but est de trouver le ou les creux le(s) plus bas, le *minimum* global (ou les *minima* globaux). Un *minimum* local, quant à lui, sera aussi représenté comme un creux, mais il ne sera pas le plus bas.

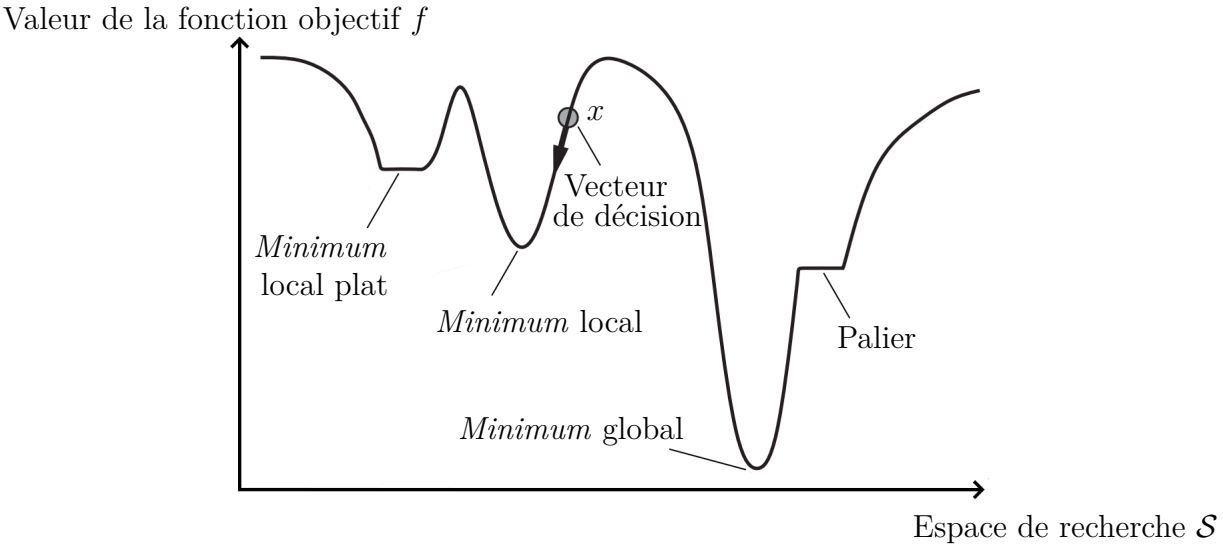


FIGURE 3.1 – Paysage d'un espace de recherche unidimensionnel (en abscisse) en fonction de la valeur de la fonction objectif (en ordonnée). Le but est de trouver un *minimum* global. Illustration inspirée de (RUSSELL et NORVIG 2016).

Le paysage de l'espace de recherche peut présenter des particularités. Un plateau, par exemple, est une région de l'espace de recherche dans laquelle tous les points ont la même valeur de fonction objectif ; la région est plate. On distingue deux cas : dans le premier, la région est constituée uniquement de solutions locales et il n'existe pas de

point dont l'élévation est plus basse : on parle de ***minimum* local plat** ; dans le second, il est possible de progresser : on parle alors de **palier**. Cette distinction est fine : sur un *minimum* local plat, tous les points sont des *minima* locaux, alors que sur un palier, tous les points sauf au moins un sont des *minima* locaux.

Ces analogies sont utiles pour appréhender les problèmes auxquels font face les algorithmes d'exploration dans le cadre de l'optimisation numérique. Cependant, ces analogies ne sont pas suffisantes et il est utile, voire nécessaire, de s'intéresser à la caractérisation du problème d'optimisation (et de sa fonction), c'est-à-dire des propriétés qui le définissent, afin d'envisager au mieux la technique d'exploration adéquate.

3.1.1.2 Quelques propriétés pour la classification des problèmes d'optimisation

Les algorithmes d'optimisation sont conçus pour résoudre un type de problème donné et peuvent ne pas être efficaces pour un type de problème différent. La classification des problèmes d'optimisation peut être très variable en fonction des auteurs, nous proposons ici une liste non exhaustive qui distingue les problèmes en fonction de différentes propriétés :

Types des variables de décision. On peut distinguer les problèmes d'optimisation **continus** et **discrets**. C'est le type des variables de décision qui détermine la classe du problème et non pas la propriété topologique de la fonction objectif (qui peut être continue ou non). S'il s'agit de variables sous la forme d'entiers ou binaires, le problème est discret (on parle aussi de problème combinatoire). Si les variables ne prennent que des valeurs réelles, le problème est continu. Il existe aussi des problèmes mêlant des variables continues et discrètes : on parle de problème **mixte**. Par exemple,

- $\forall x \in \mathcal{S} = [0,1] \cup [2,3], \text{argmin}_x f(x)$ est un problème d'optimisation dans lequel le domaine des variables de décision ne comprend que des valeurs réelles ;
- $\forall x \in \mathcal{S} = \{-1,0,1\}, \text{argmin}_x f(x)$ est un problème d'optimisation discret, car la variable de décision ne peut prendre qu'une des trois valeurs entières : -1, 0, ou 1 ;
- $\forall x = (x_1, x_2) \in \mathcal{S} = [0,1] \times \{0,1\}, \text{argmin}_x f(x)$ est un problème d'optimisation mixte, car x_1 a un domaine continu tandis que x_2 est une variable binaire.

Existence de contraintes. Les problèmes pour lesquels il existe des contraintes sur les variables de décisions ou les objectifs sont des problèmes contraints (*constrained optimisation problem*, COP). Pour définir un problème d'optimisation contraint, on ajoute à la définition 16, un système d'équations de type égalité ou inégalité. $f(x)$ est la fonction objectif à optimiser, $\forall i \in [\![1,n]\!], g_i(x) = c_i$ forment les n contraintes d'égalité à résoudre et $\forall j \in [\![1,m]\!], h_j \geq d_j$ sont les m contraintes d'inégalité à résoudre. Un exemple de problème d'optimisation contraint pourrait consister à maximiser la fonction $f(x,y) = x \times y$ avec $f : [1,20]^2 \to \mathbb{R}$ et la contrainte d'égalité $x + y = 10$.

Nombre de fonctions objectif. Il existe des problèmes définis par une unique fonction objectif et d'autres dont la solution est un compromis entre plusieurs objectifs contradictoires (l'amélioration d'un objectif provoque l'altération des autres) ou invariants (l'amé-

lioration d'un objectif n'améliore pas les autres). Les premiers sont des problèmes **mono-objectifs** et les autres **multi-objectifs**. Dans la communauté d'optimisation numérique, il est intéressant de noter que le terme « multi-objectif » comprend les problèmes ayant entre 2 et 4 objectifs alors qu'au-delà, il s'agit d'un problème « *many-objective* » (littéralement, nombreux objectifs) (Ishibuchi, Tsukamoto et Nojima 2008).

Nature de la fonction objectif et des contraintes. Il existe de nombreuses classes de problèmes dépendant des propriétés de la fonction objectif et des équations des contraintes : linéaire (de la forme $f(x) = \sum_{i=1}^{n} c_i x_i$), non-linéaire (équations polynomiales), quadratique, convexe, différentiable, continue, et bien d'autres. Identifier la nature et les propriétés de la fonction demande soit des connaissances *a priori*, soit une analyse à réaliser en amont de la résolution du problème.

Boîte noire. D'après (Audet et Hare 2017), la définition d'un problème d'optimisation boîte noire (BBO) est un problème dont la fonction objectif et/ou des contraintes sont données par des boîtes noires. Cela signifie que la fonction objectif est inconnue, inexploitable ou inexistante. La situation BBO la plus fréquente se présente lorsque l'évaluation de la fonction objectif implique l'exécution d'un code informatique ou d'une simulation (Alarie et al. 2021).

En 1995, (Wolpert et Macready 1997) (*No Free Lunch Theorems for Optimization*) montre qu'en moyenne, sur tous les problèmes d'optimisation, le comportement de n'importe quel algorithme est le même. Ce résultat théorique établit qu'il n'y a pas d'algorithme d'optimisation meilleur que les autres en général, mais plutôt qu'il existe un algorithme optimal par classe de problèmes. L'interprétation est que ce qu'un algorithme d'optimisation gagne sur un problème par rapport à tous les autres est perdu sur un autre problème. Ainsi, les algorithmes de recherche ne sont pas tous équivalents sur une classe de fonctions donnée. Ce résultat amène notamment à la conclusion suivante : la recherche en optimisation doit faire correspondre l'algorithme au problème.

Au regard de cette introduction, nous cherchons maintenant à formuler le problème d'optimisation et à identifier ses propriétés.

3.1.2 Formulation du problème d'optimisation

Pour formuler un problème d'optimisation, il est important, dans un premier temps, de définir l'ensemble des éléments constitutifs de la définition 16, à savoir : la fonction objectif, les variables de décision et leur domaine de définition. Ensuite, nous montrons qu'il y a équivalence entre la résolution du problème de satisfaction de contraintes, présenté en section 2.4 et le problème d'optimisation nouvellement formulé. Ce résultat est important pour contrôler que la formulation introduite permette bien de trouver une solution au problème d'identification des paramètres dynamiques des RRGHs. Finalement, nous cherchons à catégoriser le problème en déterminant ses propriétés et nous justifions les choix réalisés quant à la formulation du problème d'optimisation.

3.1.2.1 Composition d'un vecteur de décision

Le problème d'identification des paramètres dynamiques d'un RRGH, défini jusqu'alors comme un CSP (*cf.* section 2.4.2.2), est transformé en un PO. Dans cette formulation, un vecteur de décision est composé des variables de décisions suivantes :

— l'état initial de la trace biologique h_i,

— les célérités de C,

— et l'état hybride final de trace biologique h_f.

On notera que les états hybrides d'entrée et de sortie de chacun des états discrets n'apparaissent pas dans cette définition, alors qu'ils apparaissent dans la définition du CSP. Cela est dû au fait que ces derniers sont calculés au cours de l'étape simulation à partir des informations contenues dans le vecteur de décision, se référer à section 1.4.

On définit formellement un vecteur de décision x comme :

$$x = (h_i, \{C_{v,\omega,\eta} | v \in V, \eta \in \mathbb{S}\}, h_f) \tag{3.1}$$

Le domaine de définition d'un état hybride est le produit cartésien entre le domaine d'un état discret et celui d'une partie fractionnaire : $(\prod_{v\in V} [\![0, b_v]\!]) \times [0,1]^{|V|}$ avec V l'ensemble des variables du graphe d'interaction et b_v le niveau d'expression maximal de la variable $v \in V$. Une célérité étant définie sur $\mathbb{R}$, le domaine de x est $(\prod_{v\in V} [\![0, b_v]\!]) \times [0,1]^{|V|} \times \mathbb{R}^{|C|} \times (\prod_{v\in V} [\![0, b_v]\!]) \times [0,1]^{|V|}$.

Notre objectif est de trouver un vecteur de décision qui génère une trace hybride cohérente, satisfaisant les informations de la connaissance biologique présentée en section 2.3. Ainsi, nous cherchons à évaluer à quel point la trace hybride satisfait les connaissances à notre disposition (*cf.* section 2.3). Un vecteur de décision seul ne fournit que très peu d'information sur la dynamique du RRGH. Pour évaluer sa qualité ou plus précisément sa proximité avec les connaissances biologiques, nous utilisons le simulateur, introduit en section 1.4, qui permet d'obtenir la trace biologique simulée à partir du vecteur de décision. Ainsi, f est définie en fonction de la connaissance biologique CB formulée comme un triplet de Hoare (*cf.* définition 14), et de la trace biologique simulée à partir du vecteur de décision x, notée $tr(x)$.

Comme l'information associée à chaque étape de la trace de Hoare est composée de trois éléments : le délai de la transition continue (Δt), l'assertion ou comportement de glissement (a) et l'atome de chemin discret (dpa), nous décomposons f en trois critères.

3.1.2.2 Trois critères pour la fonction objectif

Trouver des valeurs de célérité qui satisfont aux contraintes de la connaissance biologique (CB) consiste à trouver une trace hybride simulée qui :

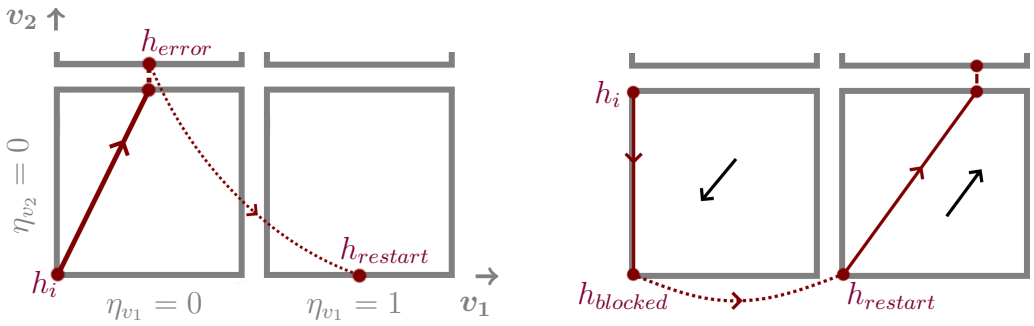
1. passe par la bonne séquence d'états discrets (dpa) ;
2. passe le bon temps requis dans chaque état rencontré (Δt) ; et
3. satisfait le bon comportement dans chaque état discret en glissant ou non (a).

Dans le cas d'une trace qui ne satisfait pas CB, nous mesurons à quel point elle ne respecte pas cette connaissance. En prenant l'exemple du cycle négatif à deux variables décrit en section 1.3.1 et à ses informations biologiques fournies en équation (2.3), lorsque CB spécifie qu'une trajectoire continue doit passer 5 heures dans l'état discret $(0,0)$, une

trajectoire passant 5 heures et 10 minutes est considérée comme étant meilleure qu'une trajectoire qui ne passe que 2 heures dans ce même état discret. Autrement dit, nous utilisons la notion de **distance** entre une trace et les propriétés attendues exprimées par CB : cette distance devient nulle lorsque toutes les propriétés de CB sont satisfaites. Puisque CB spécifie les propriétés d'une séquence d'états, nous pouvons décomposer cette distance en calculant comment une trace hybride simulée, $tr(x)$, à l'intérieur de chaque état η est éloignée des propriétés CB de l'état correspondant. Par conséquent, les trois critères sont calculés en fonction de l'information idéale, notée (t^*, a^*, dpa^*), et de l'information de la trace hybride générée à partir du vecteur de décision et du simulateur.

Pour illustrer chacun de ces critères, nous nous référons à l'exemple du cycle négatif à deux variables décrit en section 1.3.1 et à ses informations biologiques fournies en équation (2.3).

Critère de transition (d_{dpa}). Commençons par introduire le critère de transition de manière intuitive. Nous cherchons à comparer l'état discret d'arrivée, noté η^{+^*}, après la transition discrète mentionnée par CB, avec celui vers lequel transitionne effectivement la trace hybride, noté η^+. Cependant, il n'est pas toujours possible d'effectuer cette comparaison. L'hypothèse sous-jacente est qu'une trajectoire continue mène forcément à une trajectoire discrète. Or, ce n'est pas garanti pour une trace simulée. En effet, en fonction du choix du signe des célérités dans l'état discret courant (se référer à la définition 6 concernant la notion de murs), la trajectoire peut être **bloquée**. Prenons l'exemple illustré en figure 3.2b : dans cette situation, le vecteur célérité de l'état discret $(0, 0)$ pointe vers la direction du sud-ouest (la célérité de v_1 et de v_2 sont négatives). La trajectoire continue initiée à h_i est bloquée, car la concentration des produits des gènes diminue progressivement pour atteindre leur niveau de concentration minimal (il n'y a donc pas d'état discret voisin dans ces directions).



(a) (b)

FIGURE 3.2 – Illustration d'un blocage de la trace hybride. (a) : La transition discrète $(0, 0) \to (1, 0)$ est erronée. La transition attendue, spécifiée par v_1+, aurait du être $(0, 0) \to (0, 1)$. La trace est arrêtée à l'état hybride h_{error} et redémarrée à $h_{restart}$. (b) : Dans l'état $(0, 0)$ la trajectoire continue est bloquée au niveau de l'état hybride $h_{blocked}$ et redémarre à l'état hybride $h_{restart}$ puis transitionne vers l'état $(1, 1)$.

Ce problème apparaît à la lumière d'une contrainte intrinsèque à la formulation du problème d'optimisation : un vecteur de décision est évalué en bloc, c'est-à-dire comme la somme de ses parties où les parties sont les variables de décision. Un moyen de pénaliser ce

blocage serait de considérer que dès lors que la trajectoire est bloquée, alors elle est forcément non satisfaisante et qu'il faut la pénaliser totalement. Il est donc naturel d'imaginer une pénalisation qui prendrait en compte la somme des états restants à la trajectoire. Par exemple, si la trace est bloquée dans le deuxième état discret qu'elle rencontre sur un total de sept états discrets spécifiés par CB, alors la distance du critère *dpa* serait égale à $d_{dpa}(tr(x), CB) = 5/7$. Cependant, cette évaluation ne serait pas pertinente. En effet, il est important de ne pas ignorer le reste des vecteurs célérités. Si l'on considère le cas dégénéré dans lequel il existe un vecteur de décision qui génère une trace bloquée dès le premier état, mais qui ensuite possède les bonnes valeurs de célérité pour générer une trace hybride satisfaisante, alors dans ce cas spécifique, utiliser cette distance reviendrait à trop pénaliser la trace. Cela équivaudrait à mettre un poids plus important sur cette erreur, de par le fait qu'elle arrive « plus tôt ». Ce n'est pas ce que l'on cherche et cela aurait un impact négatif sur le processus d'optimisation.

Ainsi, nous avons développé un moyen de **redémarrer** la trace pour continuer son processus d'évaluation. L'exemple de la figure 3.2b montre que la trace tombe dans un point fixe en $h_{blocked}$ et est artificiellement redémarrée à l'état hybride $h_{restart}$. Cette étape est illustrée par la ligne pointillée rouge partant de $h_{blocked}$ et arrivant à $h_{restart}$. $h_{restart}$ est construit de manière automatique suivant cette règle : $h_{restart} = \left(\eta^{+^*}, \pi_{blocked}\right)$. Dans l'exemple, on a $h_{restart} = ((1,0)^t, (0.,0.))$. Cette règle est indépendante de la situation, ce qui présente l'avantage d'être instantanée et donc d'avoir un coût computationnel négligeable lors de la simulation. Néanmoins, le choix de la partie fractionnaire est discutable. Une piste d'amélioration pourrait être de modifier cette règle pour la faire dépendre des directions du vecteur célérité de l'état actuel η ainsi que celui de l'état suivant η^{+^*}.

Dans le cas de figure où la trace « se trompe » de direction et transitionne vers le mauvais état discret (η^+ diffère de η^{+^*}), le coût de l'erreur est égal à la valeur de la distance de Manhattan entre η^+ et η^{+^*}. Dans l'exemple de la figure 3.2a, la distance entre $\eta^{+^*} = (1,0)$ et $\eta^+ = (0,1)$ vaut 2. La stratégie de redémarrage de la trace est aussi appliquée. Cette fois, la trace est redémarrée à partir de l'état hybride h_{error}. $h_{restart}$ est construit de manière automatique suivant la même règle : $h_{restart} = \left(\eta^{+^*}, \pi_{error}\right)$.

Un exemple plus complexe agrège, l'un après l'autre, les deux cas de figure décrits en figure 3.3 : le blocage de la trace et la mauvaise transition discrète. En effet, suite au blocage de la trace hybride (situation strictement identique à la figure 3.2b), cette dernière est redémarrée en $h_{restart,1}$. Or, le vecteur célérité pointe vers le nord-ouest (la célérité de v_1 est négative et celle de v_2 est positive). Comme le vecteur de célérité de l'état discret $(0,0)$ n'empêche pas la transition discrète (dans ce sens, le bord n'est pas un mur interne) et que $first(h_{restart,1}) = \{v_1\}$, la prochaine transition discrète observée est v_1- (vers $(0,0)$) alors qu'elle aurait dû être v_2+ (vers $(1,1)$), d'après la connaissance biologique. Au lieu d'effectuer la transition, la trace est pénalisée et redémarrée artificiellement dans le bon état $\eta^{+^*} = (1,1)$.

d_{dpa} prend ainsi en compte la pertinence de la transition discrète grâce à la distance de Manhattan et le blocage avec la fonction indicatrice :

$$d_{dpa}(tr(x), CB) = \sum_{\eta} \left(\underbrace{d_{Manhattan}\left(\eta^+, \eta^{+^*}\right)}_{\text{transition discrète}} + \overbrace{\mathbb{1}_{blocage(\eta)}}^{\text{blocage}} \right) \tag{3.2}$$

$\mathbb{1}_{blocage(\eta)}$ est la fonction indicatrice valant 1 si la trajectoire est bloquée dans l'état discret

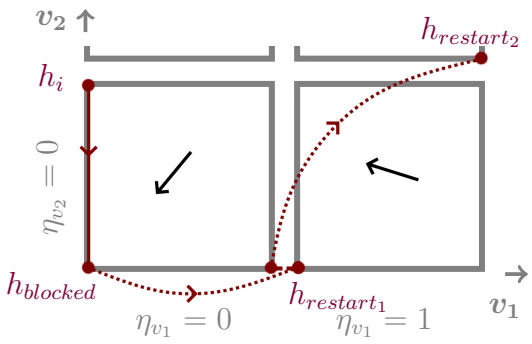


FIGURE 3.3 – Illustration d'un blocage suivi d'une mauvaise transition discrète. Le blocage est observé à $h_{blocked}$. La trace est redémarrée dans $h_{restart,1}$. Ensuite, la trajectoire transitionne vers $(0, 0)$, alors qu'elle devrait aller vers $(1, 1)$. La trace est redémarrée dans dans $h_{restart,2}$.

actuel, 0 sinon. Dans le cas où un blocage apparaîtrait, η^+ n'est pas défini, on choisit donc de donner la valeur 0 à $d_{Manhattan}\left(\eta^+, \eta^{+^*}\right)$. La pénalisation ne provient donc que de la fonction indicatrice pour cet état.

Si $\top$ est spécifié dans un triplet de CB, alors la distance de la trajectoire simulée à la connaissance biologique est nulle. Attention, cela ne signifie pas forcément que la trajectoire simulée soit cohérente biologiquement, mais plutôt que faute de connaissance, on ne peut rien en dire.

Critère temporel ($d_{\Delta t}$). Ce critère, abrégé d_{Δ_t}, vise à quantifier la pénalisation (ou le coût) du temps passé par chaque trajectoire continue de la trace simulée lorsqu'il est différent du temps attendu et spécifié par CB. Il s'agit ainsi de calculer la distance euclidienne entre le temps t_η simulé à partir du vecteur de décision pour un état discret rencontré par la trace η et le temps idéal noté t^*_η :

$$d_{\Delta t,\eta}(tr(x), CB) = d_{Euclidean}(t_\eta, t^*_\eta) \tag{3.3}$$

Un exemple est proposé en figure 3.4 dans lequel une trajectoire continue ne satisfait pas la contrainte temporelle. La trajectoire continue initiée à h_i et terminée à h_{exit} passe 2 heures et 30 minutes dans l'état discret $(0, 0)$. La trajectoire aurait dû passer 5 heures, elle est donc pénalisée en proportion : $d_{\Delta t}(tr(x), CB) = d_{Euclidean}(2.5, 5)$.

Cette distance est sommée pour chacun des états discrets par lesquels passe la trace : $d_{\Delta t}(tr(x), CB) = \sum_\eta d_{\Delta t,\eta}(tr(x), CB)$.

Critère de glissement (d_a). Le dernier critère, abrégé d_a, évalue la distance entre les comportements de la trajectoire continue à l'intérieur de chaque état discret rencontré et les propriétés de glissement de CB pour chacun de ces états. On détaille trois cas de figure (A, B et C), illustrés en figure 3.5. Dans chacun d'entre eux, la surface verte représente l'ensemble des trajectoires continues satisfaisantes, dont l'une d'elles est matérialisée (ligne verte pointillée) et la ligne continue rouge représente une trajectoire simulée non-satisfaisante. Chaque flèche noire à deux pointes, annotée d_a, correspond à la valeur de la distance correspondant à la pénalisation du vecteur de décision x.

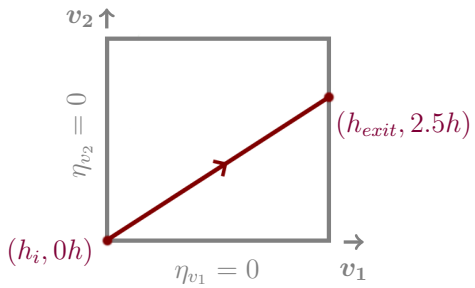
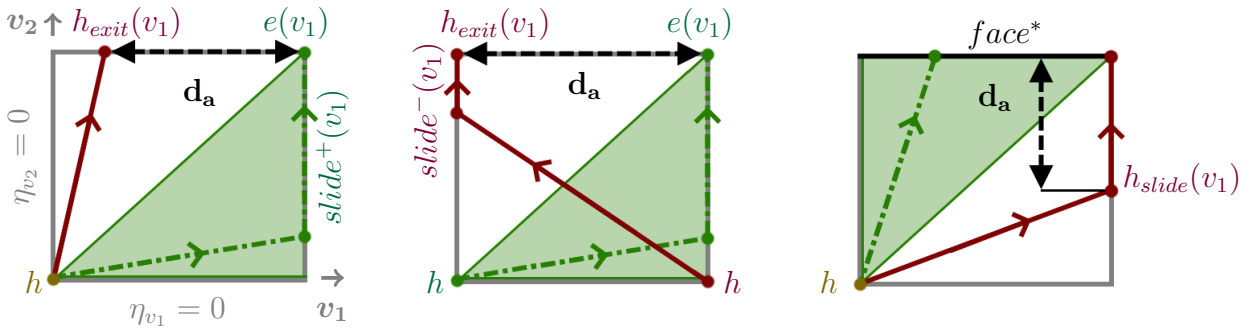


FIGURE 3.4 – Illustration d'une trajectoire continue dans le premier état discret dont la contrainte temporelle n'est pas satisfaite.



(a) Cas A : « v doit glisser, mais ne glisse pas »

(b) Cas B : « v glisse sur le mauvais bord »

(c) Cas C : « v ne doit pas glisser, mais glisse »

FIGURE 3.5 – Illustration des différents cas d'évaluation pour le critère de glissement. Les surfaces vertes représentent l'ensemble des trajectoires continues satisfaisantes dont l'une d'entre elles est matérialisée (ligne verte pointillée). Une trajectoire simulée $tr(x)$ non-satisfaisante (ligne continue rouge) est représentée. Chaque flèche noire matérialise la distance dont sera pénalisée le vecteur de décision x.

Cas A. Pour comprendre l'utilité de ce premier cas, on peut formuler la situation de la façon suivante : « la trajectoire continue doit glisser suivant la variable v selon CB, mais elle ne glisse pas ». On peut distinguer deux sous-cas. Dans le premier, on connaît le bord sur lequel doit glisser la variable : $slide^+(v)$ ou $slide^-(v)$. En suivant l'exemple en figure 3.5a, on décide de calculer la différence en valeur absolue entre la partie fractionnaire de l'état hybride de sortie de la trajectoire continue dans l'état actuel, notée $\pi_{exit}(v)$ (de $h_{exit}(v)$) et $e(v)$ qui correspond à la partie fractionnaire de l'état hybride de sortie si la trajectoire avait glissée selon CB. $e(v)$ est une fonction qui dépend de l'information de CB.

$$e(v) = \begin{cases} 0 & \text{si } slide^-(v) \\ 1 & \text{si } slide^+(v) \end{cases}$$

L'équation de pénalisation est formulée de la façon suivante et est matérialisée par la flèche pointillée noire sur la figure 3.5.

$$d_{a,\eta}(tr(x), CB) = |\pi_{exit}(v) - e(v)| \tag{3.4}$$

Dans le second sous-cas, on n'a pas l'information du signe ($slide(v)$), on ne sait pas sur quel bord doit glisser la variable et on sait qu'elle doit glisser, mais elle ne glisse pas.

Rappelons que $slide(v)$ est la disjonction de $slide^+(v)$ et $slide^-(v)$. On décide alors de pénaliser la distance entre la partie fractionnaire $\pi_{exit}(v)$ et le bord le plus proche tel que la trajectoire glisse selon v. En d'autres termes, il s'agit de calculer la distance minimale au respect de la connaissance biologique :

$$d_{a,\eta}(tr(x), CB) = min(|\pi_{exit}(v) - 1|, |\pi_{exit}(v) - 0|)$$

Cas B. Le cas B, illustré en figure 3.5b, considère l'éventualité suivante : « la trajectoire continue doit glisser suivant la variable v sur son seuil *maximum* (resp. *minimum*) d'après CB, mais la trajectoire continue glisse sur son seuil *minimum* (resp. *maximum*) ». En d'autres termes, un glissement est observé à partir de la trajectoire continue simulée, mais il est de signe opposé à celui attendu. Nous considérons cet évènement comme un cas spécial du cas A dans lequel le point hybride de sortie de la trajectoire de l'état actuel $\pi_{exit}(v)$ est soit égal à 0 si $slide^-(v)$, soit à 1 si $slide^+(v)$. Quoi qu'il arrive ici, on a $\pi_{exit}(v) = 1 - e(v)$.

On notera que si $slide(v)$ est spécifié, la distance est nulle, car n'ayant pas d'information sur le signe du glissement, la variable glisse et quel que soit le signe, c'est suffisant.

Cas C. Une troisième possibilité peut être observée : « la trajectoire continue ne devrait pas glisser selon v, mais la trajectoire simulée glisse ». Ce cas est illustré en figure 3.5c. Lorsque le signe est connu ($noslide^+(v)$ ou $noslide^-(v)$), le coût de cette erreur correspond à la distance entre la partie fractionnaire de l'état hybride $h_{slide}(v)$ qui marque le début du glissement de v et la face espérée de sortie, notée $face^*$. L'intuition derrière cette pénalisation est que l'on souhaiterait que $h_{slide}(v)$ devienne un point de l'hyperplan $face^*$: lorsque c'est le cas, la trajectoire continue satisfait la connaissance biologique.

$$d_{a,\eta}(tr(x), CB) = d_{Manhattan}\left(h_{slide}\left(v\right), face^*\right) \tag{3.5}$$

Si $noslide(v)$ est spécifié, alors peu importe la face de sortie tant que v ne glisse pas. Dans cette situation, la face espérée est celle dont la distance à l'état hybride marquant le début du glissement $h_{slide}(v)$ est minimale.

Que faire en cas de disjonction ($\vee$) ou conjonction ($\wedge$) d'assertions ? Bien q'il n'y a pas de conjonction ni de disjonction dans les assertions des connaissances biologiques des réseaux étudiés, le cas général requiert que la fonction objectif les prenne en compte. En effet, ci-dessus, nous ne mentionnons la pénalisation que d'une seule variable dans a. Prenons l'exemple d'un réseau de 4 variables $V = \{v_1, v_2, v_3, v_4\}$ dont la connaissance biologique spécifie, pour un η donné, l'assertion suivante : $slide^+(v_1) \wedge (slide(v_2) \vee slide^-(v_3))$.

Nous avons fait le choix de gérer les contraintes indirectement (voir la section 3.1.2.4). L'objectif est donc de guider au mieux le processus d'exploration lorsque la trace n'est pas valide. Dans le cas de la conjonction, choisir *min* ne semble pas être un choix judicieux, car on cherche à valider les deux assertions, or dès que l'une des deux est valide, la pénalisation est nulle. On préférera choisir l'opérateur *max* ou la somme. Tant que les deux assertions ne sont pas valides, il existe une pénalisation, et si les deux assertions sont respectées, alors leur distance est nulle.

La disjonction peut marquer le doute du modélisateur : ses connaissances ne permettent pas de conclure suivant quelle variable la trajectoire doit glisser. Le *max* ou la

somme imposent que les deux clauses soient respectées, ce qui n'est peut-être pas possible. Dans ce cas, cela poserait un problème, car une trace idéale aurait une valeur non nulle. Le *min* permet que l'une des deux clauses seulement soit satisfaite. On préférera le *min*, mais ce choix doit dépendre du réseau étudié et être fait en accord avec le modélisateur.

Comme pour les deux autres critères, la distance d_a globale de la trace est calculée comme la somme des distances obtenues dans les différents états discrets par lesquels passe la trace : $d_a(tr(x), CB) = \sum_\eta d_{a,\eta}(tr(x), CB)$.

3.1.2.3 Minimisation de la distance entre une trace simulée et la connaissance biologique

Nous proposons d'introduire la fonction f, à minimiser, comme une somme des trois critères : $d_{\Delta t}$ pour la distance temporelle, d_a pour la distance d'assertion et d_{dpa} pour la distance de transition discrète :

$$\begin{aligned} f(x) &= d(tr(x), CB) \\ &= d_{\Delta t}\left(tr(x), CB\right) + d_a\left(tr(x), CB\right) + d_{dpa}\left(tr(x), CB\right) \end{aligned} \tag{3.6}$$

Cette fonction est définie sur $(\prod_{v\in V}[\![0, b_v]\!]) \times [0,1]^{|V|} \times \mathbb{R}^{|C|} \times (\prod_{v\in V}[\![0, b_v]\!]) \times [0,1]^{|V|}$ et est à valeurs dans $\mathbb{R}^+$. En effet, un vecteur de décision générant une trace hybride correcte possède une valeur de fonction objectif nulle ($f(x) = 0$), au contraire, la valeur de f d'un vecteur de décision simulant une trace hybride incorrecte est non nulle et plus elle est éloignée de la connaissance biologique, plus sa valeur augmente. Le but est de trouver une solution globale x^* tel que $x^* \in \text{argmin}_{x\in\mathcal{S}} f(x)$.

À partir de cette définition, on cherche maintenant à caractériser le problème d'optimisation en fonction de ses propriétés.

3.1.2.4 Classification du problème d'optimisation

Certaines variables du vecteur de décision sont discrètes et d'autres continues. Il n'y a pas de contraintes dans ce problème d'optimisation. Il y a une seule fonction objectif composée de trois critères qui ne sont ni contradictoires, ni invariants : il existe des solutions qui optimisent simultanément chaque critère. Les informations et données du problème sont connues. Il s'agit donc d'un problème d'optimisation :

- mixte car x est composé de variables (i) continues : les parties fractionnaires des états hybrides initiaux et finaux et les célérités ; et (ii) discrètes : les états discrets des états hybrides initiaux et finaux ;
- contraint par les bornes uniquement, car chaque variable de décision a un domaine précis : l'ensemble des états discrets est fini, les parties fractionnaires des états hybrides sont bornées entre $[0, 1]$;
- mono-objectif car la fonction objectif n'associe à un vecteur de décision qu'une seule valeur numérique ;
- boîte noire car l'évaluation de la fonction objectif dépend exclusivement de l'exécution du simulateur.

La formulation du problème amène des choix de modélisation d'optimisation qui sont importants à spécifier ici. D'après (Eiben et J. E. Smith 2015a), il existe trois types de problèmes, résumés dans le tableau 3.1 :

— un problème non-contraint (*free optimisation problem*, FOP) est un problème dans lequel il y a une ou plusieurs fonction(s) objectif, mais où il n'y a pas de contraintes ;

— un CSP est un problème dans lequel un vecteur de décision doit respecter un système de contraintes, mais où il n'y a pas de fonction objectif explicite ;

— un COP est un problème dans lequel il y a une ou plusieurs fonction(s) objectif à optimiser et un système de contraintes à satisfaire.

	Fonction(s) objectif(s)	
Contraintes	Oui	Non
Oui	COP	CSP
Non	FOP	-

Tableau 3.1 – Types de problème en fonction de la présence ou l'absence de fonction objectif et de contraintes (Eiben et J. E. Smith 2015b). S'il n'y a pas de fonction objectif à optimiser ni de contraintes à résoudre il n'y a pas de problème d'optimisation (case annotée du symbole « - »).

En paraphrasant (Eiben et J. E. Smith 2015b), lors de l'étape de formulation d'un problème d'optimisation, déterminer sa nature est moins évident qu'il n'y paraît. Cela dépend, en fait, de la manière dont on choisit de le formuler. Ce choix est donc sujet à discussion. On peut affirmer que certaines formulations sont plus naturelles, ou mieux adaptées au problème, que d'autres, mais ce choix reste toujours subjectif.

Nous avons vu dans le chapitre précédent que l'approche CSP est limitée à mesure que le nombre de paramètres dynamiques à identifier augmente. Il est naturel de considérer que la reformulation du problème conduise à un COP. Or, nous avons fait le choix de nous émanciper des contraintes en développant une approche de **gestion des contraintes indirecte**. Pour les vecteurs de décision qui violent les contraintes du triplet de Hoare, la fonction d'évaluation peut être vue comme une fonction de pénalisation. La valeur de la fonction d'évaluation diminue à mesure que la distance entre le vecteur de décision et l'espace des solutions diminue. De ce fait, nous avons fait le choix de formuler le problème comme un FOP.

3.1.2.5 Preuve de l'équivalence entre CSP et PO

Afin de procéder au choix du type d'algorithmes d'optimisation et aux expérimentations numériques, il est d'abord nécessaire de montrer qu'il existe une équivalence entre la résolution du CSP et la résolution du PO. En effet, cette démonstration permet de se rassurer quant aux solutions obtenues. S'il est équivalent de résoudre le CSP et le PO, alors les solutions du problème d'optimisation obtenues pourront bien être considérées comme des solutions du problème d'identification des paramètres dynamiques d'un RRGH.

Theorème 1. x est solution du problème de satisfaction de contraintes si et seulement si x est solution du problème d'optimisation.

Démonstration. Raisonnons par équivalence. D'après la définition du CSP (introduit en section 2.4.2.2), x est solution du CSP équivaut à x satisfait l'ensemble des contraintes du CSP. D'après (J. Behaegel, J.-P. Comet et Folschette 2017), x satisfait toutes les contraintes du CSP est équivalent au fait que le triplet de Hoare, $CB \equiv \{Pre\}\ Q\ \{Post\}$, est satisfait. Et le triplet de Hoare est satisfait si et seulement si, pour tout état $h_i \models Pre$, il existe un autre état h_f tel que $h_i \xrightarrow{Q} h_f$ et $h_f \models Post$. En d'autres termes, le triplet de Hoare est satisfait si et seulement si la trace de x, notée $tr(x)$, satisfait les informations du triplet de Hoare, c'est-à-dire qu'elle part de l'état hybride initial h_i, suit un ensemble $T = \{cT, dT\}$ de transitions continues et discrètes devant satisfaire les informations du chemin Q (le temps Δt, le comportement a et l'atome discret dpa) et termine au point hybride final h_f. Cette équivalence est satisfaite, car la logique de Hoare est prouvée correcte et complète.

La trace $tr(x)$ satisfaisant à toutes ces contraintes équivaut au fait qu'elle débute à h_i, finit à h_f et que la distance entre $tr(x)$ et les informations du chemin est minimale. C'est-à-dire que pour tout chemin, les informations de la connaissance biologique sont en adéquation avec les informations obtenues par $tr(x)$, d'après la section 3.1.2.2 :

1. pour le critère temporel, le temps simulé t_η et le temps idéal t^*_η sont identiques pour chaque η : $\sum_\eta d_{Euclidean}(t_\eta, t^*_\eta) = 0$; par conséquent, on a : $\sum_\eta d_{\Delta t,\eta}(tr(x), CB) = d_{\Delta t}\left(tr(x), CB\right) = 0$;
2. pour le critère de glissement,
   (a) dans l'état η, la trajectoire continue doit glisser suivant v et elle glisse effectivement, alors $\pi_{exit}(v) = e(v)$, donc $d_{a,\eta}(tr(x), CB) = 0$;
   (b) dans l'état η, la trajectoire continue, ne doit pas glisser et elle ne glisse effectivement pas, alors $h_{slide}(v)$ est positionné sur l'hyperplan $face^*$: $d_{Manhattan}(h_{slide}(v), face^*) = 0$.

   Par conséquent, on a $\sum_\eta d_{a,\eta}\left(tr(x), CB\right) = d_a\left(tr(x), CB\right) = 0$;
3. pour le critère de transition, la trace suit la bonne séquence d'états, *i.e.* le nouvel état attendu η^{+^*} est identique à l'état discret simulé η^+ pour chaque η rencontré : $\sum_\eta d_{Manhattan}\left(\eta^+, \eta^{+^*}\right) = 0$ et la trace ne rencontre aucun blocage : $\sum_\eta \mathbb{1}_{blocage(\eta)} = 0$. Par conséquent, on a : $\sum_\eta d_{dpa,\eta}(tr(x), CB) = d_{dpa}\left(tr(x), CB\right) = 0$.

Finalement, on a $d_{\Delta t}\left(tr(x), CB\right) + d_a\left(tr(x), CB\right) + d_{dpa}\left(tr(x), CB\right) = d(tr(x), CB) = f(x) = 0$. On vient de montrer que $x \models C \Leftrightarrow f(x) = 0$. □

3.1.3 Choix de la technique d'optimisation

Il existe de nombreuses techniques d'optimisation numérique et nous cherchons ici à introduire les algorithmes stochastiques de l'évolution artificielle comme un choix raisonnable pour résoudre le problème d'optimisation nouvellement formulé.

Problème d'optimisation en boîte noire. La fonction d'évaluation dépend de la distance entre la trace générée à l'aide d'un simulateur externe et de la connaissance biologique. Nous choisissons de travailler sur un problème d'optimisation en boîte noire dans lequel cette fonction est considérée comme inconnue. Ce faisant, les méthodes reposant sur des hypothèses sur la nature de la fonction objectif comme la descente de gradient ou

la méthode de Newton-Raphson (dépendant respectivement du calcul de gradient et de la hessienne) ne sont pas considérées dans ce manuscrit.

Complexité du problème d'optimisation. Le nombre de célérités $|C|$ à identifier dépend du nombre de variables du graphe d'interaction ($|V|$) et du niveau d'expression maximum de chaque variable v, noté b_v. Pour un RRGH $R = (V, M, E, C)$, le nombre de paramètres dynamiques à identifier est, au maximum, de $|V| \times \prod_{v \in V}(b_v + 1)$. Nous avons vu dans le chapitre 2 qu'il n'est pas possible d'obtenir une énumération exhaustive des solutions du problème d'identification des paramètres dynamiques pour des instances de graphe d'interaction avec un nombre de variables supérieur à trois.

Du fait de ces deux arguments principaux, notre choix s'oriente vers les heuristiques. En algorithmique, une heuristique est une méthode numérique de résolution d'un problème d'optimisation qui ne garantit pas de trouver la solution optimale, mais qui permet la plupart du temps d'obtenir des solutions locales avec un budget raisonnable. On peut penser à l'algorithme d'exploration par escalade (Selman et Gomes 2006) qui consiste simplement à partir de la configuration initiale, à évaluer les configurations voisines, et à choisir la meilleure d'entre elles. L'opération est itérée jusqu'à arriver à une configuration meilleure que ses configurations voisines. Le principal inconvénient de ces approches, à savoir leur incapacité à poursuivre la recherche lorsqu'elles sont piégées dans un *optimum* local ou un plateau, nous conduit à envisager des heuristiques connues pour surmonter l'optimalité locale (Voss 2001). L'une d'entre elles sont les métaheuristiques, terme introduit par (Glover 1986). Une métaheuristique « *se réfère à une stratégie maîtresse qui guide et modifie d'autres heuristiques pour produire des solutions au-delà de celles qui sont normalement générées dans une recherche d'optimalité locale.* » (Glover et Laguna 1998). Les métaheuristiques utilisent un ensemble de vecteurs de décisions. Cette particularité leur confère la capacité de maintenir un ensemble diversifié de solutions potentielles permettant non seulement d'échapper aux *optima* locaux, mais aussi de faire face à des espaces de recherche vastes et discontinus. Comme les optimiseurs stochastiques, elles présentent l'avantage de ne pas faire d'hypothèses particulières sur les propriétés de la fonction objectif f (différentiabilité, continuité ou convexité) ou de l'espace de recherche $\mathcal{S}$. Les deux principales composantes de tout algorithme d'optimisation métaheuristique sont : l'intensification et la diversification, plus connues sous le nom d'exploitation et d'exploration. Sans forcément la garantir, une bonne combinaison de ces deux composantes majeures permet généralement d'atteindre l'optimalité globale (Blum et Roli 2003).

Dans nos travaux, nous envisageons les métaheuristiques du domaine de l'évolution artificielle (EA).

3.2 État de l'art

L'EA met en œuvre des techniques d'optimisation stochastique se basant sur le darwinisme. C'est pourquoi nous débutons cette section avec un préambule décrivant l'analogie entre théorie de l'évolution et EA. Par la suite, nous détaillons les principales métaheuristiques d'EA, ainsi que leur structure.

3.2.1 Analogie entre la théorie de l'évolution et le domaine de l'évolution artificielle

La théorie de l'évolution a été développée conjointement par Charles Darwin et Alfred Russel Wallace en 1858 (Darwin et Wallace 1858), travail précurseur de l'Origine des espèces (Darwin 1859). L'idée centrale est que des variations se produisent lors de la reproduction entre individus et sont préservées dans les générations futures en fonction de l'avantage qu'elles procurent aux individus qui les portent. Ce phénomène est connu sous le nom de « survie du plus fort » (*survival of the fittest*). Le moine Gregor Mendel découvre en 1866 les lois probabilistes qui régissent les processus de croisement et de mutation en réalisant des expériences sur des pois. Plus tard, en 1953, Watson et Crick identifient la structure de l'ADN et son alphabet. Les algorithmes d'EA sont inspirés de ces découvertes et, plus généralement, du concept de sélection naturelle. L'idée d'appliquer les principes de la théorie de l'évolution au développement de techniques d'optimisation numérique date des années 1940, d'après (D. B. Fogel 1998). Bien que les mécanismes réels de l'évolution soient beaucoup plus riches que la plupart des algorithmes évolutionnaires ne le permettent, il est usuel de les introduire au moyen de la métaphore darwinienne.

Le tableau 3.2 présente certains éléments récurrents de cette métaphore dans laquelle l'évolution peut être vue comme la résolution du problème d'optimisation. Un vecteur de décision x est assimilé à un **individu**. Un individu est codé selon un codage spécifié appelé **génotype**. Une composante du code de l'individu est appelée **gène**. L'évolution s'effectue, généralement, sur un ensemble de vecteurs de décisions, nommé **population d'individus**, $\mathcal{P} = \{x^1, ..., x^\mu\}$ de taille $\mu = |\mathcal{P}|$, et progresse en **génération**, représentant le passage de la population $\mathcal{P}^t$ à l'instant t à $\mathcal{P}^{t+1}$ à l'instant $t+1$. Les individus de la population évoluent en étant soumis à deux mécanismes : les **variations** (croisement ou recombinaison et mutation) lors de la reproduction et la **sélection naturelle** favorisant les individus les plus adaptés. L'objectif est de permettre, à terme, l'émergence d'individus adaptés à l'**environnement**. Dans le contexte de l'optimisation, l'environnement correspond à l'espace de recherche $\mathcal{S}$. L'adaptation d'un individu à son environnement est quantifiée par la valeur de la fonction objectif f, usuellement appelée **fonction de fitness** (ou de performance ou d'adaptation).

Algorithme d'EA	Méthode d'optimisation
individu	vecteur de décision x
génotype	codage de x (binaire, flottant, ...)
population	ensemble de vecteurs de décision $\mathcal{P} = \{x^1, ..., x^\mu \in \mathcal{S}^\mu\}$
génération	passage de la population $\mathcal{P}^t$ à l'instant t à $\mathcal{P}^{t+1}$ à l'instant $t+1$
environnement	espace de recherche
fitness	valeur de la fonction objectif $f(x)$
évolution	optimisation de la fonction objectif

Tableau 3.2 – Le vocabulaire de la métaphore darwinienne.

La figure 3.6 présente une illustration d'une population d'individus (points noirs) générés sur un paysage d'espace de recherche, appelé aussi **paysage de fitness**.

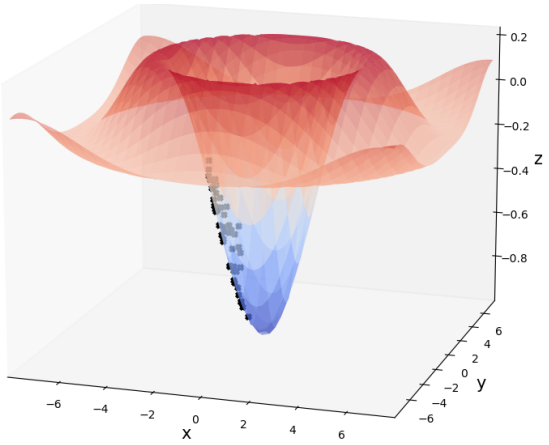


FIGURE 3.6 – Les algorithmes d'EA utilisent une population d'individus (points noirs) à chaque génération. La fonction de fitness illustrée est une fonction à deux dimensions : $f : (x, y) \mapsto -\frac{sin(\sqrt{x^2+y^2})}{\sqrt{x^2+y^2}}$ avec $x \in [-8, 8]$ et $y \in [-8, 8]$. La surface de f est saturée en rouge pour des valeurs maximales et en bleu pour des valeurs minimales. L'objectif d'un algorithme d'EA est de faire évoluer la population vers le *minimum* global se situant dans le fond de la cuve (en bleu).

3.2.2 Structure générale

La figure 3.7 introduit l'organigramme d'un algorithme d'EA commun. Il s'agit d'une représentation très limitée, car elle ne rend pas compte des spécificités des algorithmes plus récents, mais permet d'introduire conceptuellement les briques de bases.

Une population d'individus est d'abord initialisée, puis évaluée. Un sous-ensemble de cette population initiale (que l'on appelle les parents) est sélectionné pour générer de nouveaux individus (les enfants) au moyen d'opérateurs de variation (croisement, mutation, *etc.*). Les individus enfants sont évalués et remplacent les individus parents les moins adaptés (avec la valeur de fitness la plus haute) de la population. Le processus itère sur cette boucle qui forme une génération. L'évolution termine lorsque l'un des critères d'arrêt est atteint. La solution gardée est usuellement le meilleur individu de la population de la dernière génération.

Il existe un grand nombre de formes d'algorithmes d'évolution artificielle. De ce fait, pour en définir un en particulier, il faut détailler ses composants. Comme listé dans (RUSSELL et NORVIG 2016), les facteurs qui varient sont :

— la représentation ou définition d'un individu. Dans les algorithmes génétiques, on considère souvent une chaîne de booléens, dans les stratégies évolutionnaires, un vecteur de décision est une séquence de nombres réels et, en programmation génétique, un programme informatique ;

— la taille de la population. Il s'agit d'un hyperparamètre important, car sa valeur impacte fortement la diversité : l'algorithme peut soit trop explorer ou au contraire trop exploiter ;

— l'initialisation de la première population. Le plus souvent, cela est fait de manière aléatoire, mais des heuristiques spécifiques au problème peuvent être utilisées pour l'améliorer. Plus simplement, une initialisation biaisée intégrant des solutions pro-

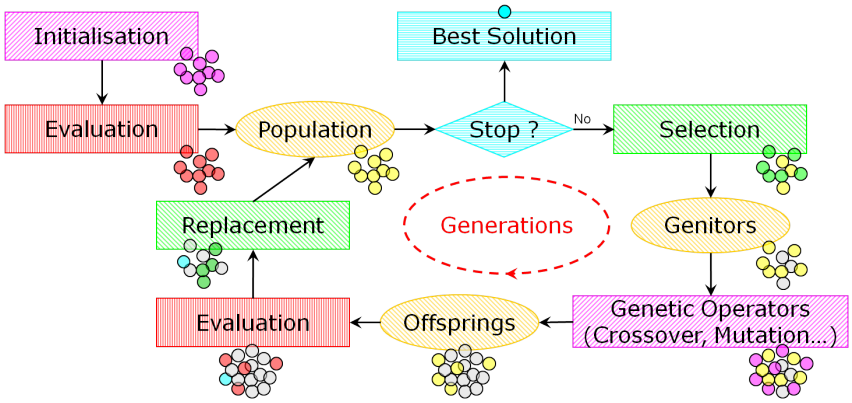


FIGURE 3.7 – Schéma évolutionnaire extrait de https://eodev.sourceforge.net/eo/tutorial/html/eoTutorial.html et modifié par Denis Pallez.

metteuses issues d'une ancienne expérimentation ou de connaissances *a priori* peut être implémentée ;

— le mécanisme de sélection des parents. Dans le but de générer de nouveaux enfants plus adaptés, on peut considérer une sélection parmi tous les individus d'une population avec une probabilité proportionnelle à leur score de fitness, ou encore, un choix aléatoire de i individus (avec $i < |\mathcal{P}|$) puis garder les $|\mathcal{P}| - i$ meilleurs ;

— les opérateurs de variations : mutation et croisement. Une approche courante du croisement est de considérer deux individus parents et de choisir de manière aléatoire un point de croisement afin de générer deux enfants formés des deux parties distinctes des parents. Pour la mutation, le taux de mutation, qui détermine la fréquence à laquelle les génotypes des enfants subissent des changements aléatoires, peut varier d'une implémentation à l'autre ;

— le mécanisme de sélection des survivants. Lors du passage d'une population à l'autre (changement de génération), la taille de la population doit rester la même (à quelques exceptions près). Cela demande de choisir quels individus sont gardés. On peut choisir de ne garder que les enfants de la population précédente, ou d'y intégrer des parents ayant les meilleures valeurs de fitness (cette notion d'élitisme permet de garantir la monotonicité de l'évolution de la fonction de fitness). On peut aussi choisir de supprimer les individus en dessous d'un seuil de valeur choisi (cette méthode est connue sous le nom d'abattage) ;

— le critère de terminaison. Cela peut être le temps CPU maximum autorisé, le nombre d'évaluations de la fonction (*number of function evluations*, NFEs), un seuil minimum de diversité, une valeur de fitness à atteindre, ou encore le seuil de progression minimal entre la meilleure valeur de fitness de la population précédente et la meilleure valeur de l'actuelle.

3.2.3 Métaheuristiques classiques

Il existe une diversité d'algorithmes d'EA. Dans cette partie, nous ne décrivons que les variants historiques sur lesquels repose le développement de nouvelles versions, à savoir

l'algorithme génétique, les stratégies d'évolution, l'évolution différentielle et l'optimisation par essaims particulaires.

3.2.3.1 Algorithme génétique

L'algorithme génétique, *genetic algorithm* (GA), est sûrement l'algorithme d'EA le plus connu. Il a été initialement conçu par Holland (Holland 1975) et utilisé comme un optimiseur dans (De Jong 1975). Dans sa version canonique, le déroulement de l'évolution par GA est le suivant : (i) une population de μ individus est initialisée ; (ii) chaque individu est évalué à l'aide de la fonction de fitness ; (iii) parmi les individus les plus adaptés, un sous-ensemble d'individus est choisi ; (iv) à partir de ces individus appelés parents, des paires aléatoires sont créées et, pour chacune de ces paires, l'opérateur de croisement est appliqué avec une probabilité p_c formant ainsi un nouvel enfant ; (v) les μ enfants générés subissent une étape de mutation, où chacun des gènes d'un individu est modifié avec une probabilité p_m. La population finale qui en résulte devient alors la population initiale de la génération suivante.

La figure 3.8 illustre ce procédé sur le problème du One Max ; ce problème est considéré comme la *drosophilia* des problèmes combinatoires dont l'objectif est de trouver la chaîne de bits de taille τ ayant la somme maximale.

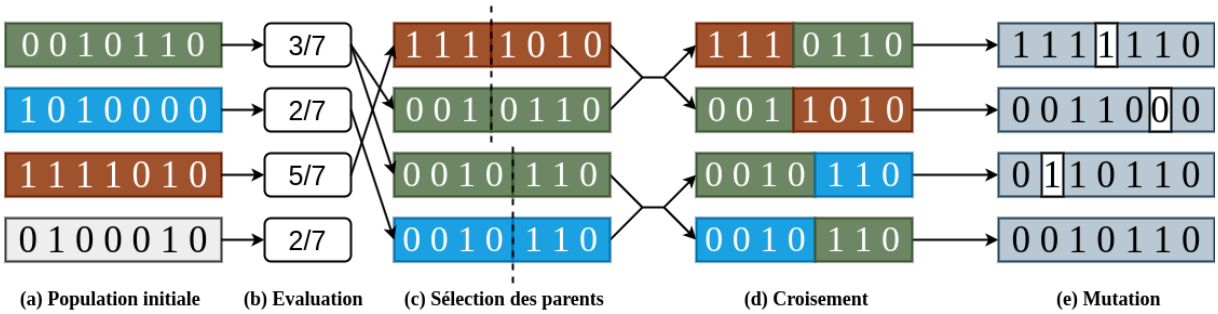


Figure 3.8 – Un GA illustré dans lequel un individu est représenté en binaire pour le problème du One Max. La population initiale de taille $|\mathcal{P}| = 4$ (a) est classée en fonction de sa valeur de fitness (b) et une partie seulement de ces individus est sélectionnée comme parents (c). Un point de croisement (ligne verticale pointillée) est choisi aléatoirement pour chacun des parents, qui deux à deux produisent deux nouveaux individus (d). Ces enfants sont soumis à une étape de mutation aléatoire (e). Illustration inspirée de (Russell et Norvig 2016).

La solution globale de ce problème est une chaîne ne contenant que des 1. Chaque individu de la population initiale de taille $|\mathcal{P}| = 4$ est représenté en chaîne de bits $x \in [\![0, 1]\!]^\tau$ (figure 3.8.a). En (figure 3.8.b), chaque individu est évalué lui appliquant une fonction de fitness. Ici, nous choisissons par exemple la somme des bits divisé par la taille de la chaîne $f(x) = \frac{\sum_{i=0}^{n} x_i}{\tau}$. Les valeurs les plus élevées sont donc les meilleures (le problème du One Max est traité comme un problème de maximisation). En figure 3.8.c, les individus sont triés par valeur de fitness et deux paires de parents sont sélectionnées. On remarque qu'un individu est choisi deux fois (en vert) et un autre pas du tout (en gris clair). Un point de croisement (ligne de croisement) est choisi aléatoirement. En figure 3.8.d, les chaînes sont croisées donnant naissance à de nouveaux individus. Enfin, en figure 3.8.e, une mutation

aléatoire (bit entouré d'un carré à fond blanc) est appliquée à chaque emplacement de chaque individu. On remarque qu'aucune mutation n'est appliquée au quatrième individu. Ce processus est itéré jusqu'à ce qu'un critère d'arrêt soit atteint ou qu'une solution.

Bien que la représentation classique d'un individu soit une chaîne de bits, les algorithmes génétiques sont aussi utilisés comme optimiseur stochastique de paramètres réels depuis (A. H. Wright 1991). Dans ce contexte, chaque individu est un vecteur x composé de variables continues et bornées : $\forall x_i \in x, x_i \in [x_i^{max}, x_i^{min}]$. Les étapes d'évaluation et de sélection restent identiques. Avec une probabilité p_m, l'étape de mutation est réalisée en ajoutant une perturbation aléatoire telle que : $x'_i \sim x_i + \sigma \cdot \mathcal{N}(0,1)$ où x'_i est le nouvel individu muté à partir de x_i, σ représente l'intensité de la mutation (une valeur élevée augmente l'importance de la mutation, favorisant l'exploration, et à l'inverse, une valeur plus faible favorise l'exploitation) et $\mathcal{N}(0,1)$ est une variable aléatoire suivant une loi normale centrée autour de 0 avec une dispersion unitaire.

3.2.3.2 Stratégies d'évolution

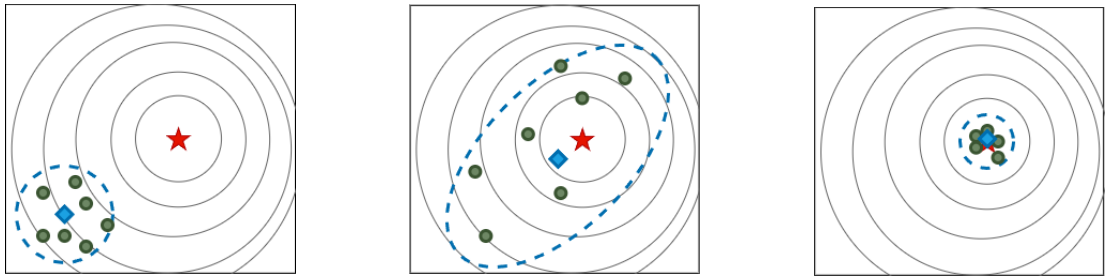
Les stratégies d'évolution, *evolution strategies* (ES), ont été initialement proposées par Rechenberg (Rechenberg 1965) et Schwefel (Schwefel 1981) qui travaillaient à l'Université Technique de Berlin sur une application concernant l'optimisation des formes.

Les stratégies d'évolution utilisent généralement ρ parents de la population $\mathcal{P}$ de taille μ pour produire λ enfants. Une itération peut être découpée en deux étapes. La première est appelée étape de recombinaison. Elle consiste à choisir aléatoirement ρ individus parents de $\mathcal{P}$, (ii) à les recombiner, à l'aide d'un opérateur de croisement choisi, pour former un unique individu enfant, et (iii) à muter cet enfant. Cette mutation est généralement réalisée par ajout d'une variable aléatoire suivant une loi normale centrée en 0 et de déviation standard σ : $x'_i \sim x_i + \mathcal{N}(0,\sigma)$ où x_i est un individu enfant, x'_i l'individu enfant x_i muté et σ est appelé la taille du pas de mutation. Lorsque λ individus sont produits itérativement, une dernière étape a lieu : l'étape de sélection. Cette étape consiste à sélectionner les μ meilleurs individus et peut s'appliquer : soit uniquement aux λ nouveaux enfants mutés, on note la stratégie d'évolution $(\mu/\rho, \lambda)$, soit à l'ensemble des parents et des enfants $(\rho + \lambda)$, dans ce cas la stratégie est notée $(\mu/\rho + \lambda)$. La notation (μ, λ) ou $(\mu + \lambda)$ signifie que l'étape de recombinaison n'est pas présente : il n'y a que l'étape de sélection.

Tout l'intérêt des stratégies d'évolution réside dans le caractère « auto-adaptatif » de leurs paramètres, comme notamment le pas de mutation σ. Par exemple, la technique « 1/5 *success rule* » utilise le nombre de mutations réussies pour adapter sa valeur, où une mutation réussie est un individu muté dont la valeur de fitness est meilleure que celle du même individu avant mutation.

Un des algorithmes les plus efficaces pour des problèmes difficiles d'optimisation continu et en scénario boîte noire est CMA-ES (Nikolaus Hansen et Ostermeier 2001). Il s'agit d'une stratégie d'évolution qui repose sur l'adaptation de la matrice de covariance de la distribution normale. Une illustration en figure 3.9 dépeint les étapes principales de l'évolution de CMA-ES[1]. L'*optimum* global de la fonction à optimiser est indiqué par l'étoile rouge sur le paysage de fitness (cercles noirs concentriques). Lors de l'initialisation, λ

1. La référence pédagogique suivante contient de nombreuses visualisations intéressantes : https://blog.otoro.net/2017/10/29/visual-evolution-strategies/



(a) Début de l'optimisation ($\sigma = 0.2$)

(b) Recherche de l'*optimum* ($\sigma = 0.5$)

(c) Convergence ($\sigma = 0.02$)

FIGURE 3.9 – Illustration de CMA-ES inspirée de (Z. LI et al. 2023).

individus (points verts) sont échantillonnés à partir de la distribution de CMA-ES. La ligne bleue pointillée marque le profil de la distribution de CMAES vue du dessus. La moyenne de la distribution gaussienne est indiquée par le losange bleu en son centre. Lors de l'évolution, la déviation standard σ de CMA-ES est mise à jour et permet de diriger et d'étendre la recherche vers une nouvelle partie de l'espace de recherche. Finalement, lors de la convergence, la distribution se concentre autour de l'*optimum*.

3.2.3.3 Évolution différentielle

L'évolution différentielle, *differential evolution* (DE), est née en 1995 (STORN 1995). C'est un algorithme évolutionnaire basé sur des vecteurs à valeurs réelles, que l'on note $\bar{x}$. Sa particularité réside dans la définition de l'opérateur de reproduction appelé *mutation différentielle*. Étant donné une population de solutions candidates $\mathcal{P}$, un nouvel individu $\bar{x'}$ () est généré en ajoutant une *perturbation vectorielle* $\bar{p}$ à un individu existant $\bar{x}$, $\bar{x'} = \bar{x} + \bar{p}$. Cette perturbation vectorielle est la différence vectorielle entre deux individus ($\bar{y}$ et $\bar{z}$) de la population, pondérée par un facteur réel strictement positif noté F : $\bar{p} = F \cdot (\bar{y} - \bar{z})$. Ce paramètre F contrôle la rapidité d'évolution de la population. L'opérateur de croisement par défaut est le croisement uniforme dépendant du paramètre $Cr \in [0, 1]$. Un exemple de mutation différentielle est fourni en figure 3.10 dans un espace de paramètres en deux dimensions (X_1, X_2). Le nouvel individu $\bar{x'}$ (en orange) est généré en appliquant une perturbation vectorielle $\bar{p}$ qui n'est autre que la distance vectorielle entre $\bar{y}$ et $\bar{z}$ multipliée par un facteur $F > 0$ ($\bar{p} = F \cdot (\bar{y} - \bar{z})$) et ajoutée à l'individu $\bar{x}$.

Afin de classifier les différents variants de DE, la notation $DE/a/b/c$ a été introduite. a précise la méthode de sélection du vecteur de base de la mutation différentielle : *rand* permet de choisir un vecteur de décision aléatoire et *best* d'en choisir un ayant la meilleure valeur de fitness. b est le nombre de différences vectorielles utilisées pour calculer la perturbation vectorielle, *i.e.*, le nombre de vecteurs divisé par deux. En effet, il peut y avoir plus d'une seule différence vectorielle : $\bar{p} = F.(\bar{y} - \bar{z} + \bar{y'} - \bar{z'})$ avec $\bar{y}, \bar{z}, \bar{y'}, \bar{z'}$ quatre individus de la population. Et c correspond à la méthode de croisement choisie : *bin* pour le croisement aléatoire suivant une loi binomiale ou *exp* pour une loi exponentielle. La version la plus basique de DE est $DE/rand/1/bin$.

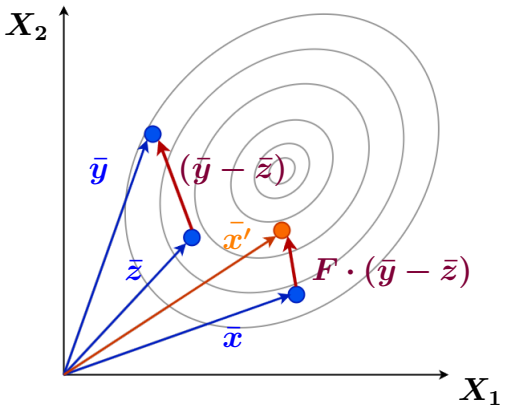


FIGURE 3.10 – Illustration de l'opérateur de mutation dans un espace de paramètres en deux dimensions (X_1, X_2) et un paysage de fitness (lignes grisées formant des ovales concentriques) vu du dessus, inspirée de (DAS et SUGANTHAN 2011).

3.2.3.4 Optimisation par essaims particulaires

L'optimisation par essaims particulaires, *particle swarm optimisation* (PSO), (KENNEDY et EBERHART 1995) est une métaheuristique qui se base sur la simulation du comportement social des oiseaux. L'algorithme utilise un ensemble d'individus, renommés particules, pour guider la recherche dans l'espace des états. Chaque particule possède une vitesse et est influencée par les meilleures solutions localement et globalement. On note x^i la position de la i-ème particule composée de n variables de décision ($x^i = (x^i_1, ..., x^i_n)$ et v^i sa vitesse. g est la position de la meilleure solution trouvée (globalement) dans l'essaim, tandis que p^i est la meilleure position historiquement trouvée par la i-ème particule.

La vitesse de la i-ème particule pour la génération suivante $t+1$, notée $v^{i,t+1}$ est mise à jour par la formule suivante :

$$v^{i,t+1} = \underbrace{\omega \times v^{i,t}}_{\text{inertie}} + \underbrace{c_1 \times r_1 \times (p^{i,t} - x^{i,t})}_{\text{influence cognitive}} + \underbrace{c_2 \times r_2 \times (g^t - x^{i,t})}_{\text{influence sociale}} \tag{3.7}$$

où $v^{i,t}$ est la vitesse de la i-ème particule de la génération t, $p^{i,t}$ est la meilleure position de la particule i jusqu'à la génération t, $x^{i,t}$ est la position de la particule i à la génération t et g^t la position de la meilleure solution trouvée dans l'essaim jusqu'à la génération t. ω est la variable d'inertie qui sert de pondération à la vitesse de la particule, c_1 et c_2 sont deux poids de valeur positive qui permettent de balancer entre exploitation de p^i et g respectivement. Par exemple, plus la valeur de c_1 est haute, plus l'importance sera donnée à p^i. r_1 et r_2 sont deux valeurs aléatoires qui renforcent c_1 et c_2.

La position de la particule i à la génération suivante $t+1$ est ensuite mise à jour à l'aide de la vitesse et de la position de la particule de la génération actuelle t :

$$x^{i,t+1} = x^{i,t} + v^{i,t} \tag{3.8}$$

L'analogie bio-inspirée provient du fait que l'équilibre entre exploration et exploitation dans PSO est contrôlé par l'influence sociale et l'influence cognitive dans l'équation (3.7). L'influence sociale provient de l'utilisation de la meilleure solution trouvée globalement

dans l'essaim pour la mise à jour de la vitesse. L'influence cognitive de l'essaim est déterminée par la meilleure solution de la particule. La détermination des valeurs appropriées pour c_1 et c_2 est donc primordiale.

Un exemple de mise à jour de la position est illustré en figure 3.11. Chaque particule est représentée par un cercle. La nouvelle position du i-ème individu (cercle jaune) $x^{i,t+1}$ dépend de la position de la particule $x^{i,t}$ (cercle noir), de sa vélocité $v^{i,t}$ (flèche noire partant de $x^{i,t}$), de sa meilleure position historiquement mémorisée $p^{i,t}$ (cercle violet) et de la position de la meilleure particule de l'essaim g^t (cercle rouge). Les trois flèches jaunes, de gauche à droite, représentent l'inertie, l'influence cognitive et l'influence sociale, voir l'équation (3.7), qui, ajoutées à la position initiale de la particule, permettent de calculer sa nouvelle position, voir l'équation (3.8).

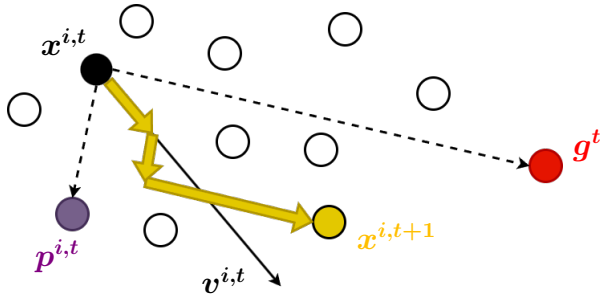


FIGURE 3.11 – Illustration, inspirée de https://www.baeldung.com/cs/pso, d'un essaim particulaire.

3.2.3.5 Revue partielle des autres variantes et critique

Il existe de nombreuses autres variantes majeures que nous ne considérons pas dans ce manuscrit comme la programmation génétique (KOZA 1994) ou la programmation évolutionnaire (L. J. FOGEL 1962), visant à faire évoluer des programmes, les algorithmes d'estimation par distribution (BALUJA 1994) visant à résoudre des problème d'optimisation en construisant et échantillonnant explicitement les modèles probabilistes des variables de décision, la neuroévolution (K. O. STANLEY et MIIKKULAINEN 2002) dont l'objectif est d'optimiser la structure et les poids d'un réseau de neurones artificiel, ou encore les algorithmes de qualité-diversité comme MAP-Elites (MOURET et CLUNE 2015) visant à identifier plusieurs solutions optimales et diverses au cours d'une unique exécution.

Les références à des concepts de biologie, de génétique et de l'évolution sont fréquentes. On peut aussi citer d'autres exemples comme les *niches de population* (MAHFOUD 1995) pour maintenir une diversité de solutions au sein d'une même population, les interactions écologiques comme le « troc biologique » (OLLERTON 2006) (*biological barter*), pour les mécanismes de coévolution traitant des problèmes large échelle, ou encore le phénomène d'*épistasie* (REEVES et C. C. WRIGHT 1995) pour décrire les problèmes d'optimisation dans lesquels certaines variables de décision sont liées, rendant l'optimisation plus difficile. Bien que les métaphores puissent être de puissants outils pour faciliter l'explication ou même stimuler l'intuition, l'émergence et la prolifération de centaines de variantes algorithmiques (à peine discernables d'un point de vue algorithmique) sous différentes étiquettes ont été contre-productives pour le progrès scientifique du domaine[2]. Il a été

2. *The Evolutionary Computation Bestiary* (https://fcampelo.github.io/EC-Bestiary/) est un

montré qu'une vaste majorité d'entre elles n'est composée que des variantes des algorithmes classiques (Molina et al. 2020) et non de nouvelles méthodes d'optimisation globales révolutionnaires. Dans ces cas de figure, la métaphore n'améliore pas la capacité à comprendre et à simuler les systèmes biologiques et n'apporte pas de connaissances généralisables ou de principes de conception pour les approches d'optimisation globale, comme énoncé dans (Sörensen 2015 ; Campelo et Aranha 2023). Des travaux récents critiquant ce phénomène de la métaphore se sont concentrés sur la manière d'améliorer la solidité expérimentale, la reproductibilité et la normalisation des nouvelles approches. On peut citer un exemple récent qui propose une procédure de tests statistiques pour évaluer les propriétés d'invariance de translation et de rotation intrinsèques aux métaheuristiques fiables, que la plupart de ces métaphores trompeuses ne possèdent pas (Walden et Buzdalov 2024).

Cette introduction générale du domaine de l'évolution par EA peut être approfondie grâce aux travaux de (Michalewicz 1996 ; Le Riche, Schoenauer et Michèle Sebag 2007 ; Eiben et J. E. Smith 2015b ; Kenneth De Jong 2017).

3.3 Résolution du problème d'optimisation

Cette section s'attache à montrer comment le problème d'optimisation, formulé en section 3.1.2, peut être résolu avec l'utilisation d'algorithmes d'EA. Plus particulièrement, l'intérêt de cette section porte, d'abord, sur les spécifications techniques de la mise œuvre de la fonction de fitness. À la suite de quoi, une description des études expérimentales menées pour valider empiriquement cette approche est fournie. La validation empirique est un point essentiel, car elle permet de montrer la pertinence de la formulation du problème et des choix de sa mise en pratique à travers l'obtention de résultats probants. Une critique de ces résultats sera aussi menée *a posteriori*.

3.3.1 Prérequis des expérimentations

Nous apportons quelques précisions pour la réalisation de l'étude expérimentale présentée en section 3.3.2.

Représentation d'un vecteur de décision et espace de recherche. Comme énoncé dans l'équation (3.1), un vecteur de décision est défini par l'état hybride initial h_i de la trace, l'ensemble des vecteurs célérités C et l'état hybride final h_f de la trace. Cependant, nous avons des connaissances *a priori* sur les réseaux qui permettent de réduire l'espace de recherche et ainsi de potentiellement améliorer le processus d'évolution. Dans les trois réseaux étudiés, nous connaissons l'état hybride initial et pour chacun de ces réseaux, la connaissance biologique spécifie que la trace doit suivre un cycle (le départ de la trace correspond à son arrivée) : $h_i = h_f$. De plus, nous bornons les valeurs admissibles des paramètres dynamiques entre $[-r, r]$ avec $r = 2$, pour les réseaux 2G et 3G, et $r = 7$ pour le réseau 5G.

site web listant plus de 260 de ces extravagantes variantes, dont voici quelques exemples : « Alibaba et les 40 voleurs », « Brainstorming », « Mobilité des spermatozoïdes », ou encore « les oiseaux d'Hitchcock »...

Ces connaissances sont intégrées dans la définition du problème d'optimisation. Un vecteur de décision est ainsi réduit à

$$x = \{C_{v,\omega,\eta} \mid v \in V, \eta \in \mathbb{S}\}$$

avec la fonction de fitness $f(x)$ définie sur $[-r, r]^{|C|}$ et à valeurs dans $\mathbb{R}^+$. Cette modification apporte un changement de classe du problème d'optimisation, devenant un problème continu dans lequel toutes les variables de décision sont continues. Ainsi, pour chacun des réseaux, le lecteur peut se rapporter à la section 1.3, afin de connaître l'ensemble des paramètres dynamiques à identifier. Un vecteur de décision est évalué en simulant sa trace hybride correspondante et en évaluant, état discret par état discret, chacun des trois critères : d_{Δ_t}, d_a et d_{dpa}.

Proposition d'une seconde fonction de fitness. Dans l'équation (3.6), nous avons agrégé ces critères pour former une distance globale sous la forme d'une fonction additive. Cette fonction est notée $f_+(x)$ et est une somme non pondérée des trois critères [3].

Néanmoins, une seconde version est proposée. Elle correspond à une fonction multiplicative : $f_\times(x) = (1 + d_{\Delta t}) \times (1 + d_b) \times (1 + d_e) - 1$. Le paysage de fitness correspondant est différent, car le taux de convergence est fortement impacté, amplifiant les erreurs. Grâce à une pente plus raide, les incréments peuvent être mieux contrôlés : la différence de valeur de fitness entre deux points de l'espace de recherche, où le premier est mieux choisi que le second, sera plus grande.

Nous souhaitons étudier comment le processus d'optimisation stochastique de l'algorithme d'EA est impacté par cette modification.

Méthodes d'optimisation choisies et paramètres. Nous avons choisi un algorithme d'optimisation aléatoire (Matyas et al. 1965), noté RO, en tant qu'algorithme de référence. Quatre métaheuristiques ont été choisies :

- DE avec l'opérateur de croisement suivant une loi binomiale et l'opérateur de mutation $DE/rand/1/bin$. Les paramètres de contrôles sont $Cr = 0.3$ et F est sélectionné de manière adaptative dans l'intervalle $[0.5, 1.0]$ pour le calcul du vecteur différence grâce à la technique *dither* (Dawar et Ludwig 2014) ;
- $(\mu + \lambda)$ GA utilisé avec une sélection par tournoi binaire, l'opérateur de croisement *Simulated Binary* et la mutation polynomiale. De plus, chaque solution dupliquée est supprimée ;
- PSO adaptatif (PSO) (Zhan et al. 2009) initialisé avec les paramètres suivants : $w = 0.9, c_1 = 2.0, c_2 = 2.0$ et $max\_velocity\_rate = 0.2$. Cette version adaptative cherche à déterminer des valeurs c_1 et c_2 dynamiquement au lieu de les considérer comme des constantes ;
- CMA-ES avec pour valeur initiale de déviation standard $\sigma = 0.1$.

Chacune de ces implémentations provient de la librairie `pymoo` (Blank et Deb 2020). On notera $\mathcal{A} = \{\text{RO, DE, GA, PSO, CMA-ES}\}$ l'ensemble des algorithmes d'optimisation testés.

3. La recherche de poids adéquats ne fait pas l'objet d'une étude particulière. Il pourrait être intéressant d'analyser la variabilité des performances en termes de qualité des solutions trouvées et de vitesse de convergence en fonction de chacune des métaheuristiques.

Comme il s'agit d'algorithmes impliquant des processus stochastiques, deux exécutions d'un même algorithme d'EA sont susceptibles de ne pas donner le même résultat [4]. Ainsi, il est nécessaire de reproduire la même expérience plusieurs fois pour s'assurer qu'en moyenne les performances d'une méthode d'optimisation m_1 sont supérieures à celles d'une seconde méthode m_2, et ce, sur un échantillon représentatif. En réalité, il n'existe pas de règle absolue sur le nombre d'expériences, cela dépend de l'étude statistique à mener. En essence, plus grand est l'échantillon, plus précise sera la conclusion, afin de réduire les erreurs de Type I (faux négatif). Dans notre cas d'étude, nous verrons qu'un faux négatif correspond à prédire qu'il existe des différences significatives en termes de performances entre m_1 et m_2, alors que ce n'est pas le cas : l'hypothèse nulle énonçant que les performances de m_1 et m_2 sont identiques est réfutée alors qu'elle ne devrait pas. En évitant les erreurs de Type I, on évite de conclure de manière erronée sur la supériorité d'une méthode sur une autre.

3.3.2 Étude expérimentale

L'efficacité des métaheuristiques (en termes de qualité des solutions trouvées et de vitesse de convergence) est très dépendante du problème considéré (*cf.* les théorèmes No Free Lunch (Wolpert et Macready 1997)). C'est pourquoi une étude expérimentale comparant plusieurs métaheuristiques est nécessaire pour (i) identifier le meilleur algorithme candidat pour le problème, dans le but de (ii) trouver une solution optimale plus efficacement.

3.3.2.1 Preuve de concept sur le cycle négatif (2G).

(Michelucci, Jean-Paul Comet et Pallez 2022) se focalise sur la validation de la preuve de concept consistant à montrer que la résolution du CSP peut être reformulé en problème d'optimisation et que cette reformulation est pertinente au regard des résultats obtenus. Dans cette publication, une étude expérimentale a été menée sur le cycle négatif à deux variables, comportant plusieurs objectifs :

1. évaluer la performance des algorithmes testés sur notre problème,
2. identifier la fonction de fitness la plus adéquate entre f_+ et $f_\times$, et
3. valider qu'une solution optimale peut-être identifiée.

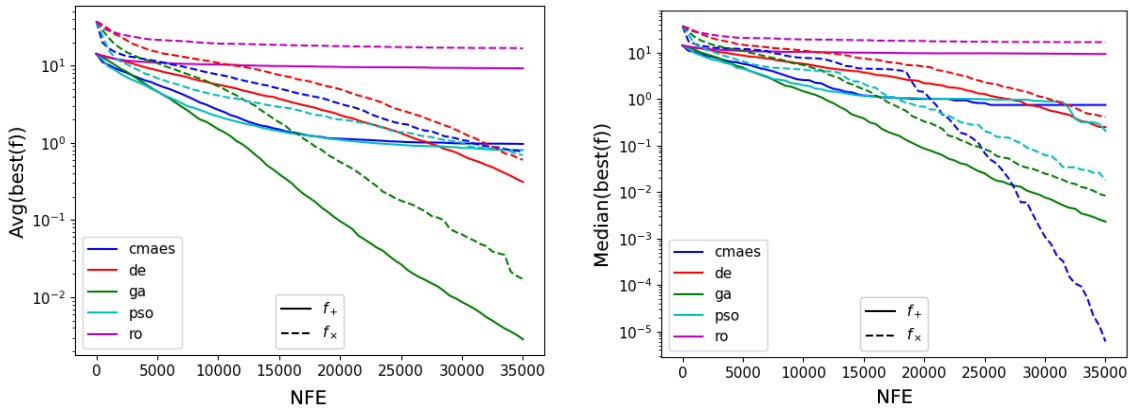
On notera $\mathcal{F}$ l'ensemble des fonctions objectif testées ($\mathcal{F} = \{f_+, f_\times\}$). L'atteinte de ces objectifs permettra d'exhiber, de concert, l'algorithme et la fonction de fitness les plus pertinents pour notre problème, et ainsi trouver une solution (globale, dans le meilleur des cas, ou locale) au problème.

Les performances sont évaluées sur 100 exécutions indépendantes. Chaque métaheuristique comprend une population initiale de 500 individus, et un budget de 35 000 fonction d'évaluations leur est attribué. L'acronyme « NFE » est utilisé pour décrire le nombre de fonction d'évaluation. Ce nombre a été empiriquement choisi afin de trouver un compromis entre le coût computationnel et la qualité des résultats. En d'autres termes, il n'est

4. D'ailleurs, ils peuvent être modélisés comme des chaînes de Markov où les opérateurs (sélection, croisement, mutation) définissent des transitions en probabilités entre les états, que sont les populations (Nix et Vose 1992).

pas désirable de gaspiller de la ressource et de l'énergie computationnelle, mais cela ne doit pas être fait au détriment des résultats. En effet, si le budget est trop faible, le comportement asymptotique des algorithmes peut ne pas être atteint et donc, de fait, rendre des conclusions erronées.

Analyse des résultats. Pour chaque algorithme et chaque fonction de fitness ($\mathcal{A} \times \mathcal{F}$), à chaque génération, nous gardons le meilleur vecteur de décision obtenu jusqu'à présent, et calculons la moyenne (resp. la médiane) sur les 100 exécutions. Les évolutions monotones de tous les algorithmes sont représentées sur la figure 3.12a où les lignes continues représentent f_+ et celles en pointillés $f_\times$.



(a) (b)

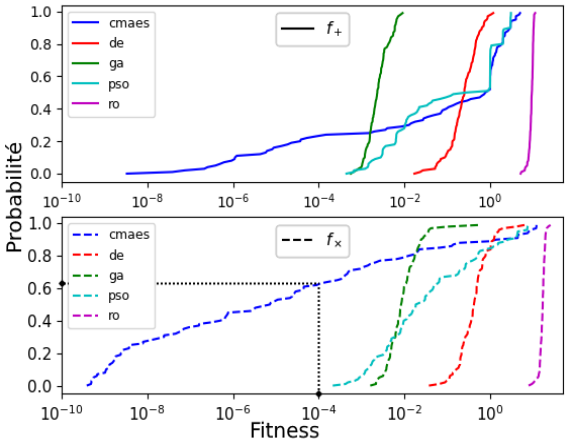
FIGURE 3.12 – Comparaison des évolutions monotones de (a) la moyenne et (b) la médiane des meilleures valeurs de fitness obtenues par configuration ($\mathcal{A} \times \mathcal{F}$) pour 100 exécutions sur le RRGH à deux dimensions. L'axe des ordonnées est à l'échelle logarithmique.

On peut observer que (i) comme prévu, les métaheuristiques sont bien plus performantes que RO, (ii) les diminutions des valeurs de f_+ et $f_\times$ se font au même rythme (les courbes sont à peu près parallèles), sauf pour CMA-ES dont l'évolution médiane de $f_\times$ a un meilleur taux de convergence qu'avec f_+, et mis à part ce cas, (iii) la convergence de GA est l'une des meilleures, à la fois avec f_+ et $f_\times$, que ce soit en moyenne ou en valeur médiane.

Le tableau en 3.13a résume les statistiques des résultats obtenus après 100 exécutions des cinq algorithmes considérés. Le meilleur résultat (colonne par colonne) pour f_+ (resp. $f_\times$) est en gras. Le minimum (min), la moyenne (avg) et l'écart-type (stdev) sont indiqués, ainsi que le taux de réussite biologique *BSR* (pour *biological success rate*, en anglais). Cette mesure est définie par le nombre de fois où un algorithme trouve une solution avec une valeur de fitness proche de 0 prenant en compte une erreur de précision ϵ égale à 10^{-2}. *BSR* est basé sur le score appelé *success rate*. Le *success rate* est défini comme le pourcentage d'exécutions où une solution a été identifiée. Le *BSR* introduit une erreur de précision cohérente avec l'expertise biologique. Par exemple, une trajectoire qui glisserait dans $\eta = (0, 0)$ pendant une fraction de secondes ($< \epsilon$) tout près du point de sortie $e(v_1) = 1$ avant de passer à l'état discret suivant est une trajectoire acceptable bien que *CB* indique $noslide(v)$. Une trace reste donc valide à ϵ près.

$\mathcal{A}$	$\mathcal{F}$	avg ± stdev	min	BSR
CMA-ES	f_+	0.9644 ± 1.18	**3e-9**	0.41
	$f_\times$	0.7661 ± 2.51	**4e-10**	0.86
DE	f_+	0.3102 ± 0.23	0.0171	0.13
	$f_\times$	0.6004 ± 0.77	0.0373	0.04
GA	f_+	**0.0029 ± 2e-3**	6e-4	**1.**
	$f_\times$	**0.0172 ± 0.05**	0.0016	**0.98**
PSO	f_+	0.8053 ± 0.98	4e-4	0.48
	$f_\times$	0.6938 ± 1.71	2e-4	0.68
RO	f_+	9.1934 ± 1.11	5.1679	0.
	$f_\times$	16.6763 ± 2.5	7.9144	0.

(a)



(b)

FIGURE 3.13 – (a) Table et (b) courbes de fonction de répartition des meilleurs résultats obtenus sur 100 exécutions.

En outre, les courbes de la fonction de répartition sont construites dans la figure 3.13b pour f_+ (en haut) et $f_\times$ (en bas). Chaque courbe décrit la probabilité qu'une solution soit trouvée à, ou en dessous de, un score de fitness donné. Par exemple, dans l'expérience $f_\times$, il y a environ 60% de probabilité qu'un utilisateur obtienne une solution avec un score de fitness inférieur ou égal à 10^{-4} avec CMA-ES, étant donné 35 000 NFEs (lignes pointillées noires). Deux algorithmes ressortent des deux diagrammes : GA et CMA-ES. GA a la probabilité la plus élevée d'obtenir de bons résultats. C'est la courbe qui arrive proche de 1.0 en ordonnée pour des valeurs plus petites de fitness (en abscisse). CMA-ES a une probabilité non nulle d'obtenir les meilleurs résultats ($< 10^{-8}$). En effet, avec $f_\times$, il y a plus de 20% de probabilité d'obtenir un solution avec un score de 10^-8.

Tests statistiques. Pour valider statistiquement les différences observées entre les algorithmes, nous avons mené une campagne de validation statistique sur les valeurs de performance rapportées pour les deux scénarios suivants :

— (H_0^1) les performances des algorithmes avec f_+ sont identiques ;

— (H_0^2) les performances des algorithmes avec $f_\times$ sont identiques ;

— (H_0^3) les performances des algorithmes obtenues avec f_+ sont identiques à celles obtenues avec $f_\times$.

Par valeurs de performance, on entend prendre pour chaque exécution la valeur de fitness du meilleur individu de la dernière population d'un algorithme.

Tout d'abord, nous testons les conditions d'indépendance, de normalité et d'homoscédasticité des variances requises pour l'application des tests paramétriques ou non-paramétriques (HOLLANDER, WOLFE et CHICKEN 2013 ; EFTIMOV et KOROŠEC 2021). La condition d'indépendance est satisfaite *de facto* par le choix de graines aléatoires, *seeds*. Le test de Shapiro-Wilk évalue si l'échantillon est issu d'une population normalement distribuée. Le test de Levene est utilisé pour évaluer l'égalité de variance pour une variable calculée pour plusieurs groupes. Les conditions n'étant pas remplies, nous utilisons des tests statistiques non-paramétriques. Tout d'abord, l'ANOVA de Friedman par

rangs (*Friedman rank-sum test*) est appliqué pour évaluer si au moins deux algorithmes présentent des différences significatives dans les valeurs de performance observées. Les p-valeurs calculées pour les hypothèses nulles sont H_0^1 et H_0^2 sont $p_+ = 5\text{e-}56$ et $p_\times = 2\text{e-}64$ (pour f_+ et $f_\times$ respectivement). Au niveau de confiance $\alpha = 0.05$, on peut dire que les différences entre les algorithmes sont significatives.

L'analyse statistique procède à une analyse pour déterminer quelles paires d'algorithmes présentent des différences significatives en termes de performances (pour les trois scénarios considérés ci-dessus). Dans cette étape, nous procédons au test des rangs signés de Wilcoxon (*Wilcoxon signed-rank test*) sur l'échantillon de performance de chaque paire d'algorithmes. En outre, la correction de Bonferroni est appliquée. Cette méthode est utilisée pour corriger le seuil de significativité lors de comparaisons multiples. Elle permet de réduire le risque d'erreurs de Type I. Pour rappel, une erreur de Type I arrive lorsque l'hypothèse nulle est rejetée alors qu'elle est vraie en réalité.

Pour tous les scénarios, le tableau 3.3 présente des diagrammes en mosaïque afin d'illustrer toutes les différences par paire dans les performances observées, à un niveau de confiance α. Plus précisément, les résultats des tests des rangs signés de Wilcoxon, sans et avec l'application de la méthode de correction de Bonferroni, sont fournis à gauche et à droite des graphiques, respectivement. Chaque tuile correspond à un test entre les algorithmes de la ligne et de la colonne correspondantes. La couleur de la tuile indique si les différences de performance observées sont suffisantes pour rejeter l'hypothèse nulle au niveau α : $p < 0.05$. Les carreaux gris clair indiquent des différences significatives entre les paires d'algorithmes, tandis que les carreaux gris foncé indiquent qu'aucune différence significative n'a été observée.

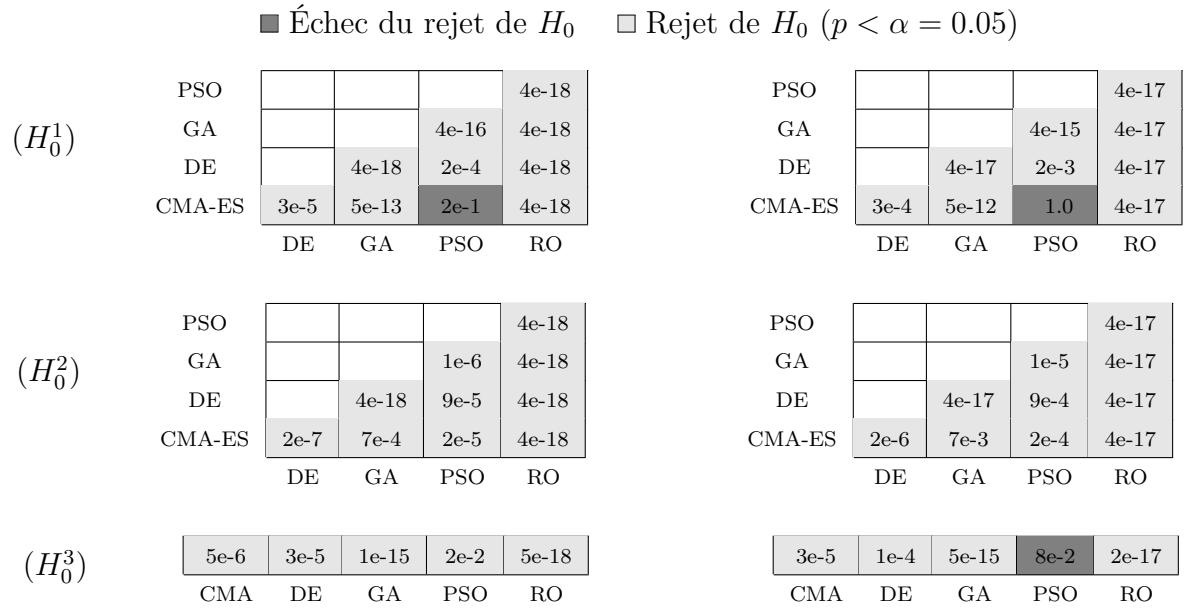


Tableau 3.3 – Test des rangs signés de Wilcoxon (gauche) avec l'analyse *post hoc* Bonferroni (droite) pour les trois scénarios concernés : H_0^1 (f_+), H_0^2 ($f_\times$) et H_0^3 (f_+ vs. $f_\times$).

En analysant ces résultats, si nous nous reposons sur l'acceptation ou le rejet des hypothèses, les conclusions suivantes peuvent être déduites :

— dans le scénario f_+, avec ou sans la correction de Bonferroni, les performances de

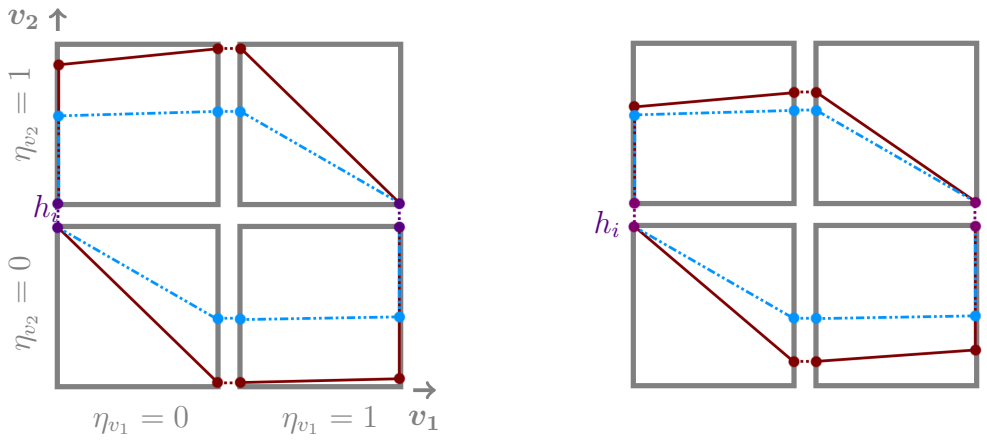
PSO et CMA-ES ne sont pas significativement différentes ;

— la correction de Bonferroni révèle qu'il n'y a pas d'évidence suffisante pour rejeter l'hypothèse nulle H_0^3 : les performances de PSO seraient donc les mêmes quelle que soit la fonction de fitness choisie. Néanmoins, les performances des autres algorithmes dépendent de la fonction de fitness choisie.

Ainsi, selon l'algorithme considéré, le choix de la fonction de fitness a un impact certain sur les performances : f_+ est préféré dans le cas de DE et GA, tandis que $f_\times$ l'est dans le cas de CMA-ES et PSO.

Finalement, en conclusion des expériences et des tests menés, GA et CMA-ES sont les métaheuristiques les plus performantes. Le premier donne de bons résultats avec une forte probabilité (figure 3.13b) et ses courbes d'évolution montrent une convergence intéressante (figure 3.12). Le second donne de meilleures performances globales, mais est sujet à l'instabilité (probablement en raison des phases d'exploration locale).

Visualisation des résultats. L'application d'algorithmes d'EA nous permet de présenter différentes solutions compatibles avec *CB* et elles semblent complémentaires à celles de l'approche CSP. Les deux diagrammes de la figure 3.14 présentent en rouge la meilleure trajectoire globale obtenue par GA (figure 3.14a) et CMA-ES (figure 3.14b) ainsi que celle en bleu, obtenue par l'approche CSP en utilisant le solveur AbSolute combiné avec une stratégie de découpage de l'espace de recherche utilisée dans (J. Behaegel, J.-P. Comet et M. Pelleau 2018). Les solutions fournies par GA et CMA-ES illustrent la diversité des solutions acceptables qui sont conformes à *CB*.



(a) (b)

Figure 3.14 – Visualisation des meilleures traces (en rouge) obtenues par (a) GA et (b) CMA-ES avec f_+ ayant obtenues les valeurs de fitness 6e-4 et 3e-9 respectivement. En bleu est représentée une des solutions obtenue avec l'approche CSP.

Suite aux résultats obtenus sur le cycle négatif (2G), nous avons décidé d'évaluer les performances des algorithmes CMA-ES et GA avec la fonction de fitness f_+ sur les deux autres RRGHs étudiés. Nous commençons d'abord par le cycle circacdien (3G).

3.3.2.2 Validation sur le cycle circadien (3G)

L'idée est de poursuivre la validation de la preuve de concept et de montrer qu'il est possible de trouver une solution globale au problème d'identification des paramètres dynamiques des RRGHs, mais en s'attaquant maintenant au cycle circadien à trois variables. Cette partie des expérimentations est extraite de l'étude expérimentale présentée dans (Michelucci, Callegari et al. 2024). Les performances sont évaluées sur 50 exécutions indépendantes. Chaque métaheuristique possède une population initiale de 500 individus. Le budget attribué est de 100 000 NFEs.

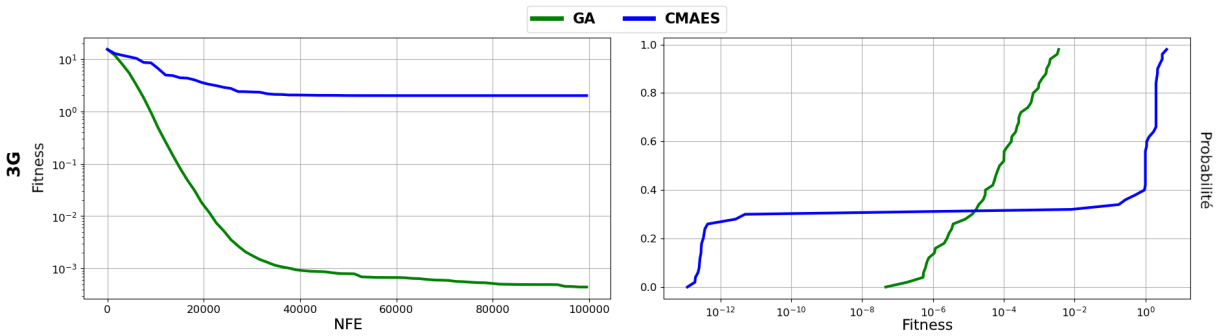
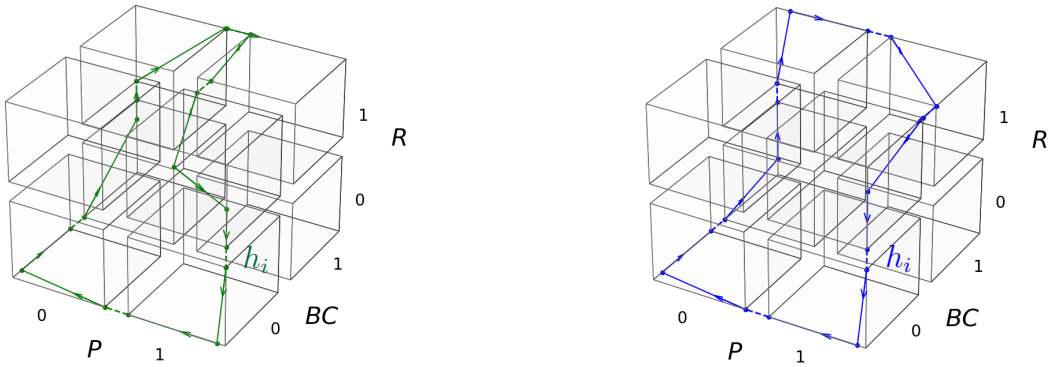


Figure 3.15 – Comparaison des évolutions monotones de la moyenne des meilleures valeurs de fitness f_+ au cours de l'exécution (à gauche), et les courbes de répartitions (à droite) obtenues pour CMA-ES et GA sur 50 exécutions pour le cycle circadien. L'axe des ordonnées est à l'échelle logarithmique.

$\mathcal{A}$	$\mathcal{F}$	avg ± stdev	min	BSR
CMA-ES	f_+	1.08 ± 1.02	**1.17e-13**	0.34
GA	f_+	**4.42e-4 ± 7.99e-4**	4.66e-8	**1.0**

Tableau 3.4 – Table des meilleurs résultats obtenus sur les 50 exécutions sur le cycle circadien.



(a) (b)

Figure 3.16 – Meilleures traces obtenues par GA (a) et CMA-ES (b) avec f_+, ayant pour valeur de fitness 1.39e-9 et 4.53e-14 respectivement.

Analyse et visualisation des résultats. Les résultats des expérimentations réalisées sur le cycle négatif sont analogues à ceux extraits des expérimentations sur le cycle circadien. D'après le tableau 3.4, GA obtient en moyenne de meilleurs résultats et dans 100% des cas, ces résultats sont probants au regard de la mesure *BSR*. CMA-ES est capable de trouver d'excellents résultats quand son évolution ne reste pas bloquée dans un *minimum* local : les solutions dont la valeur de fitness est proche de 1 sont des solutions qui présentent un blocage (*cf.* équation (3.2)). Ces résultats peuvent être extraits visuellement avec les courbes de la figure 3.15 : à gauche, l'évolution monotone moyenne de chaque algorithme et, à droite, la fonction de répartition des résultats.

La figure 3.16 représente une trace valide obtenue à partir d'une solution optimale, pour GA (figure 3.16a) et CMA-ES (figure 3.16b).

3.3.2.3 Résultats sur le cycle cellulaire (5G)

Finalement, nous proposons de nouvelles autres expérimentations dont le but est d'identifier un modèle valide dans le cycle cellulaire avec cinq variables. Cette partie des expérimentations est aussi extraite de (Michelucci, Callegari et al. 2024). Les performances sont évaluées sur 50 exécutions indépendantes. Chaque métaheuristique possède une population initiale de 500 individus. Le budget attribué est de 200 000 NFEs.

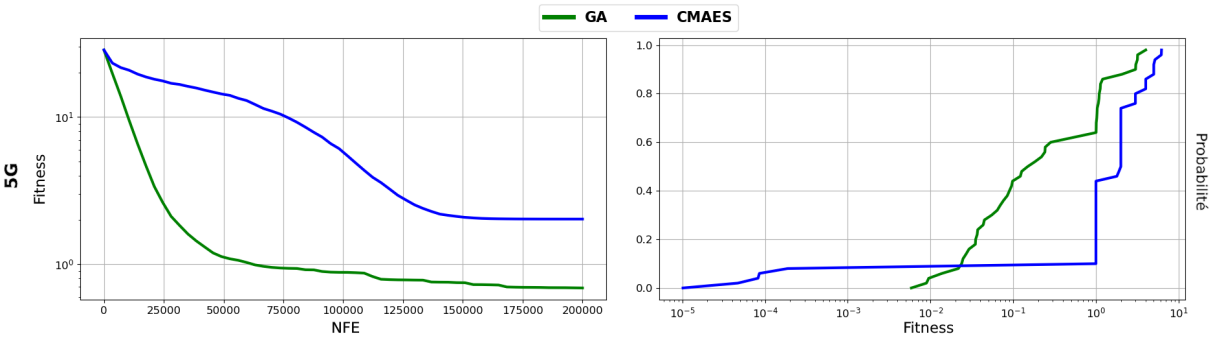


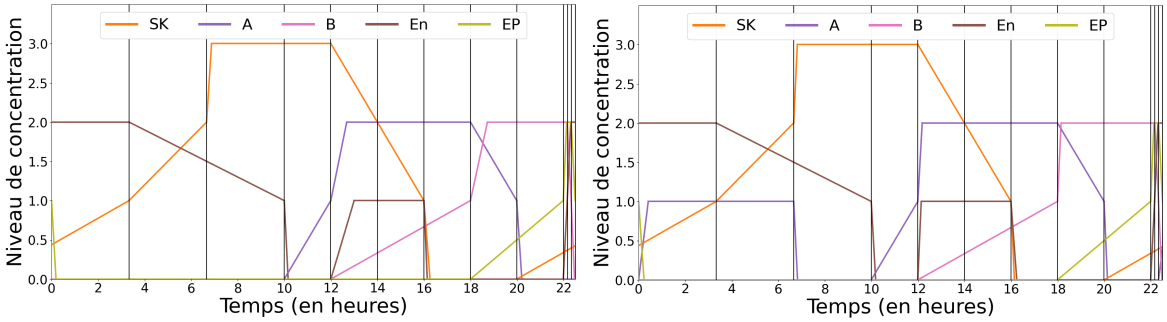
Figure 3.17 – Comparaison des évolutions monotones de la moyenne des meilleures valeurs de fitness f_+ au cours de l'exécution (à gauche), et les courbes de répartitions (à droite) obtenues pour CMA-ES et GA avec sur 50 exécutions sur le cycle cellulaire. L'axe des ordonnées est à l'échelle logarithmique.

$\mathcal{A}$	$\mathcal{F}$	avg $\pm$ stdev	min	BSR
CMA-ES	f_+	2.02 ± 1.58	**1.02e-5**	**0.1**
GA	f_+	$\mathbf{0.69 \pm 0.99}$	5.87e-3	0.06

Tableau 3.5 – Table des meilleurs résultats obtenus sur les 50 exécutions pour le cycle cellulaire.

Analyse et visualisation des résultats. D'après la mesure *BSR* du tableau 3.5, GA et CMA-ES sont capables d'identifier une solution optimale (dans peu de cas). Les courbes de répartitions de la figure 3.17 montrent que les algorithmes restent bloqués le plus souvent dans des *optima* locaux : dans plus de 40% des cas pour CMA-ES et

60% pour GA. On peut noter cependant qu'avec CMA-ES, on obtient une plus haute probabilité d'obtention de solutions ($< 10^{-2}$) que celle de GA. Ce résultat est confirmé par l'obtention du plus haut score de *BSR* par CMA-ES : 0.1 pour CMA-ES contre 0.06 pour GA. On peut aussi observer sur la figure 3.17, à gauche, que les évolutions moyennes des métaheuristiques atteignent un plateau (asymptote horizontale) démontrant que ces observations ne résultent pas d'un budget insuffisant. La figure 3.18 présente deux solutions optimales obtenues sur le cycle cellulaire, l'une avec GA (figure 3.18a) et l'autre avec CMA-ES (figure 3.18b).



(a) (b)

FIGURE 3.18 – Meilleures traces obtenues par (a) GA et (b) CMA-ES avec f_+, ayant pour valeur de fitness 1.16e-3 et 1.44e-6 respectivement. L'évolution de la concentration au cours du temps de chaque variable du graphe d'interaction est représentée par une ligne colorée.

Dans la partie suivante, nous décrivons les différentes pistes envisagées pour se défaire des limitations rencontrées.

3.3.2.4 Conclusions, remarques et critiques des résultats obtenus

La transformation du CSP en un problème d'optimisation a permis de montrer à travers une étude expérimentale qu'il est possible d'obtenir une solution globale sur les trois RRGHs . La fonction objectif et les algorithmes ont bien permis d'identifier au moins une solution dans chacun des cas. Cependant, on note que les performances décroissent à mesure que le nombre de paramètres dynamiques à identifier augmente. Pour tenter de pallier ce problème, on pourrait :

— modifier les opérateurs de mutation et de croisement en injectant des connaissances du domaine. Par exemple, on pourrait augmenter la probabilité de mutation p_m d'un vecteur célérité dont on sait qu'il n'est pas valide ou augmenter la probabilité de croisement entre des solutions qui sont valides sur une partie antagoniste de la trace dans le but d'augmenter la probabilité de générer un vecteur de décision valide après les variations stochastiques du processus d'évolution ;

— intégrer un opérateur de réparation afin de chercher à ne générer que des traces qui passent par la bonne séquence d'états discrets. Cela pourrait enlever le problème du poids du blocage dans l'équation (3.2) : les métaheuristiques pourraient éviter plus facilement les *minima* locaux provoqués par le blocage de la trace ;

— utiliser des algorithmes et procédures adéquates pour gérer la montée en charge du nombre de paramètres (passage à l'échelle) ;

— pousser l'exploration à travers l'utilisation de mécanismes de préservation de diversité pour échapper aux *optima* locaux et diriger la recherche vers des zones du paysage de fitness plus prometteuses (Preuss, Michael G Epitropakis et al. 2021).

Les deux derniers points ont été envisagés dans cette thèse : le premier, dans la section suivante (section 3.4), et le second, indirectement, dans un chapitre ultérieur (chapitre 5).

3.4 Passage à l'échelle

Les performances d'un algorithme peuvent se détériorer rapidement lorsque la dimension du problème, c'est-à-dire le nombre de variables de décision, augmente. Ce phénomène est connu sous le nom de « malédiction de la dimension » (« *curse of dimensionality* », expression introduite par Richard E. Bellman). Les problèmes impliquant un grand nombre de variables de décision sont considérés comme des problèmes d'optimisation globale à grande échelle (*large-scale global optimization problems*). Il existe pléthore de méthodes pour adresser ce problème : soit en le décomposant ou en réduisant l'espace de recherche, soit en améliorant l'efficacité des méthodes de recherche en concevant par exemple de nouvelles stratégies plus adaptées à large échelle (J. Liu et al. 2024).

La **coévolution** (Popovici et al. 2012), ou optimisation coévolutive, consiste à décomposer le vecteur de décision d'un problème d'optimisation en le distribuant entre plusieurs individus. L'évaluation d'un individu devient fonction de son interaction avec d'autres individus. Ce domaine découle de l'observation biologique montrant que l'évolution concurrente d'un certain nombre d'espèces, définies comme des groupes d'individus ayant un phénotype similaire, est plus conforme à la réalité que la simple évolution d'un groupe d'individus représentant une seule espèce (Paredis 1995).

D'un point de vue algorithmique, au lieu de faire évoluer une population d'individus à la recherche d'une solution globale d'un problème d'optimisation donné, comme c'est le cas avec les algorithmes d'EA traditionnels, la coévolution consiste en l'évolution d'un ou plusieurs groupes d'individus dans lesquels chaque individu représente une partie différente du problème (un sous-problème). Dans une recherche coévolutive qui implique l'utilisation de plusieurs groupes d'individus similaires, ces groupes sont appelés espèces ou **sous-populations**. On distingue ainsi la coévolution **mono-populationnelle** de la coévolution **multi-populationnelle**.

Dans tous les cas, l'évaluation de la fonction de fitness s'appuie sur l'agrégation de plusieurs individus. Si, dans le cas classique, la fonction de fitness associe un ensemble d'arrivée (généralement $\mathbb{R}$) à l'ensemble des individus : $f : \mathcal{S} \to \mathbb{R}$, dans le cas de la coévolution, le domaine d'entrée de la fonction est un produit cartésien de plusieurs ensembles de vecteurs de décisions $f : \mathcal{S}_1 \times ... \times \mathcal{S}_m \to \mathbb{R}$, avec m dépendant du problème. Le tuple $(\mathcal{S}_1 \times ... \times \mathcal{S}_m)$ est appelé interaction car il représente l'interaction entre plusieurs individus. Dans le cas d'un algorithme d'EA traditionnel, il est possible de mesurer la valeur de fitness d'un individu indépendamment, alors que dans la coévolution la fitness n'est calculée qu'à travers une interaction de plusieurs individus appelés **coindividus** $(x_1 \in \mathcal{S}_1, ..., x_m \in \mathcal{S}_m)$.

Historiquement, il existe deux types d'interactions formant deux classes de coévolution : la coévolution **compétitive** et **coopérative**. Dans le cadre de la coévolution compétitive (Rosin et Belew 1997), les coindividus sont récompensés aux dépens de ceux avec lesquels ils interagissent. D'autre part, dans le cadre de la coévolution coopérative (CC), les coindividus sont évalués en fonction du résultat de leur interaction avec d'autres coindividus : ils sont récompensés lorsqu'ils obtiennent de bons résultats et sont punis dans le cas contraire (Ma et al. 2019).

Le mécanisme inhérent de la CC est de décomposer le problème initial en plusieurs sous-problèmes afin que chacun de ces sous-problèmes soit résolu en faisant évoluer un algorithme d'EA. Grâce à la coévolution coopérative d'une ou de plusieurs sous-populations, un **individu complet** du problème initial est obtenu en regroupant des **coindividus représentatifs**. Dans le schéma mono-populationnel, tous les (ou un ensemble de) coindividus de la même population forment ensemble un individu complet, comme dans l'évolution parisienne (*Parisian evolution*) (Collet et al. 2000). Dans le schéma multi-populationnel, la combinaison du meilleur individu de chaque sous-population génère un individu complet, comme dans le travail pionier de (Potter et Kenneth A. De Jong 1994).

Les mécanismes sous-jacents de « diviser pour mieux régner » et de collaboration permettent aux algorithmes de CC de s'attaquer efficacement au problème du passage à l'échelle (Yang, Tang et Yao 2008 ; X. Li 2014 ; Miguel Antonio et Coello Coello 2018), et plus généralement à des problèmes complexes (Krawiec et Heywood 2019). C'est pour cette raison que nous étudions la CC comme moyen de faire face à la diminution des performances des résultats d'EA classiques sur notre problème d'identification des paramètres dynamiques dans les RRGHs.

3.4.1 État de l'art de la coévolution coopérative

Cette partie de manuscrit s'inspire de la classification proposée dans l'étude de (Ma et al. 2019) pour présenter les notions importantes de la CC dans le contexte des problèmes d'optimisation mono-objectif. Elle établit les bases permettant de s'intéresser aux questions suivantes :

1. le problème est-il **séparable**, *i.e.* est-il possible de diviser le problème initial en sous-problèmes ?
2. quelle stratégie employer pour décomposer le problème initial ?
3. quels coindividus choisir et comment les faire collaborer pour l'évaluation de la fonction de fitness ?
4. que choisir entre un schéma mono ou multi-populationnel ?

3.4.1.1 Séparabilité

La CC repose sur la décomposition du problème initial en plusieurs sous-problèmes de taille inférieure. Or, cette décomposition n'est pas forcément triviale et impacte la performance de la méthode utilisée. De ce fait, le concept de séparabilité est central dans la résolution d'un problème d'optimisation avec un algorithme de CC (Yang, Tang et Yao 2008). Les problèmes complètement séparables sont les problèmes les plus faciles à résoudre :

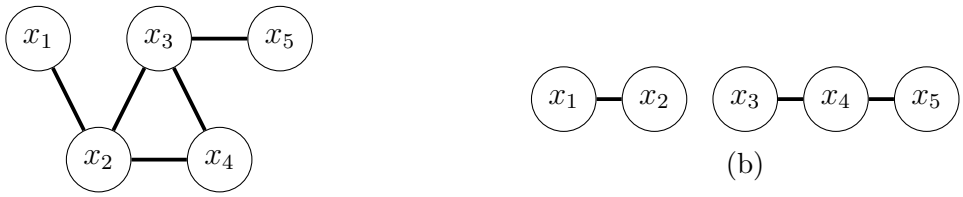
Séparabilité d'un problème d'optimisation

Définition 17. Un problème d'optimisation $f : \mathcal{S} \rightarrow \mathbb{R}$ est dit *complètement séparable* si et seulement si

$$\text{argmin}_{x_1,...,x_n} f(x) = (\text{armgin}_{x_1} f(x_1, ...), ..., \text{argmin}_{x_n} f(..., x_n))$$

où $\mathcal{S}$ est un espace de recherche de dimension n : il y a n variables de décisions dans un vecteur de décision x. Un problème d'optimisation complètement séparable peut être résolu en optimisant chaque variable de décision de manière indépendante. Si ce n'est pas le cas, on parle de problème d'optimisation non séparable.

Les problèmes d'optimisation complètement séparables sont connus pour être efficacement solvables par CC et, plus généralement, par des techniques mettant en place la stratégie de « diviser pour mieux régner », car il suffit d'optimiser les différentes variables de décision indépendamment. À l'opposé, les problèmes non séparables sont plus complexes à résoudre, car il existe une chaîne de dépendance impliquant toutes les variables de décisions : la modification de la valeur d'une variable de décision a un impact sur toutes les autres. La figure 3.19a illustre un exemple de problème non séparable dans lequel il existe une chaîne de dépendance reliant toutes les variables de décision. Dans cet exemple, le vecteur de décision formé de cinq variables de décision ($x = (x_1, ..., x_5)$) est représenté sous la forme d'un graphe dans lequel un nœud matérialise une variable de décision et une arrête indique une interaction.



(a) (b)

FIGURE 3.19 – Chaque nœud représente une variable de décision $\{x_1, ..., x_5\}$ et une arrête indique une dépendance entre la variable source et la variable cible. (a) : Un problème non-séparable et (b) : Un problème k non séparable avec $k = 3$. k correspond au nombre de variables de décision dépendantes de la sous-composante ayant la cardinalité maximale. Il y a deux sous-graphes et la cardinalité maximale est de trois.

Il existe aussi des problèmes d'optimisation partiellement séparables comme les problèmes k-séparables (COLSON et TOINT 2005) dans lesquels au plus k variables sont dépendantes. Une illustration est proposée en figure 3.19b dans laquelle le vecteur de décision est k-séparable. En effet, les deux chaînes de dépendance forment deux sous-composantes, les composantes connexes $G_1 = \{x_1, x_2\}$ et $G_2 = \{x_3, x_4, x_5\}$ de cardinalité deux et trois, respectivement. k correspond au nombre de variables de décision dépendantes du sous-graphe ayant la cardinalité maximale : ici, $k = 3$. Plus k est grand, plus le problème se rapproche d'un problème complètement non séparable et donc plus il est complexe à résoudre.

Lors de la conception d'un algorithme de coévolution, il est donc primordial de faire le choix de la décomposition du vecteur de décision en fonction des dépendances entre ses

variables. Une décomposition optimale est entièrement basée sur la minimisation des dépendances entre les sous-composantes du problème (Omidvar et al. 2014). Il a été montré que si les variables liées ne sont pas groupées dans la même sous-composante, alors la méthode d'optimisation a plus de chance d'être bloquée dans un pseudo-*minimum*, qui n'est pas un *minimum* local de la fonction objectif, mais un *minimum* local introduit à cause d'une mauvaise décomposition (Bergh et A.P. Engelbrecht 2004). Nous présentons différentes stratégies de décomposition ci-après.

3.4.1.2 Stratégies de décomposition

Il existe de nombreuses stratégies de décomposition qui donnent lieu à des types de groupements de variables de décision différents. Le groupement statique détermine une décomposition *a priori* selon certaines règles générales et s'y tient tout au long du processus d'optimisation. La figure 3.20 illustre deux stratégies de groupement statique. Dans cet exemple, le vecteur de décision est composé de huit variables ($x = (x_1, ..., x_8)$) qui sont décomposées en quatre sous-composantes de taille deux. La première décomposition statique (figure 3.20a) est séquentielle : les variables sont regroupées deux à deux dans l'ordre des indices des variables ($\forall i \in [\![1, 7]\!], (x_i, x_{i+1})$). La seconde décomposition statique (figure 3.20b) est aléatoire.

(a) (b)

Figure 3.20 – Illustration de deux types de groupements de variables statiques. Décomposition (a) séquentielle et (b) aléatoire d'un vecteur de décision à $n = 8$ dimensions en $m = 4$ coindividus de taille fixe $s = 2$.

La stratégie basée sur l'apprentissage des dépendances peut apprendre à déterminer de manière automatique le nombre des sous-composantes et leur taille. Le groupement aléatoire suit la même stratégie que celui illustré en figure 3.20b mais il peut, en plus, faire varier le placement des variables au cours du processus d'évolution. Il existe aussi une longue liste de stratégies mêlant plusieurs méthodes énoncées ci-dessus, appelées hybrides (Ma et al. 2019).

Il est important de noter que la connaissance spécifique au domaine peut être introduite pour choisir une meilleure découpe en sous-composantes.

Lorsque le choix d'une stratégie de décomposition est réalisée, il reste à choisir comment les coindividus collaborent entre eux. L'évaluation de la fitness de chaque coindividu nécessite la collaboration d'autres coindividus. Dans le cas multi-populationnel, la manière dont les représentants des autres sous-populations sont sélectionnés pour former un individu complet joue un rôle important dans les performances des algorithmes de CC. Nous présentons ensuite différentes stratégies de collaboration.

3.4.1.3 Stratégies de collaboration

La stratégie, probablement la plus répandue et la plus utilisée, est la stratégie du *single best* qui consiste tout simplement à évaluer le j-ème coindividu de la i-ème sous-population avec les meilleurs coindividus de chacune des autres sous-populations. La figure 3.21 (a)

représente trois sous-populations (lignes pointillées) de trois coindividus (rectangles). Ces coindividus sont classés par valeur de fitness : du pire au meilleur, de bas en haut. Le coindividu en vert est évalué avec la stratégie du *single best* représentée par les lignes noires. À l'opposé, la stratégie du *single worst*, combine les coindividus avec les moins bons coindividus de chaque sous-population, voir la figure 3.21 (b). La stratégie *linked* nécessite de lier des coindividus au début du processus d'évolution : un dans chaque sous-population. Ensuite, un individu complet est tout simplement formé en combinant les coindividus liés. Dans la figure 3.21 (c), tous les j-ème coindividus sont liés à tous les autres j-ème individus des autres sous-populations. Finalement, la stratégie *random* sélectionne aléatoirement des coindividus dans les sous-populations pour former un individu complet du problème à évaluer, comme le montre la figure 3.21 (d). Il est aussi possible d'utiliser ces stratégies de sélection pour évaluer un coindividu en fonction de sa participation à la formation de plusieurs individus complets, comme c'est le cas dans (Panait, Wiegand et Luke 2003). (Ma et al. 2019) fournit une liste plus complète de la diversité des stratégies existantes.

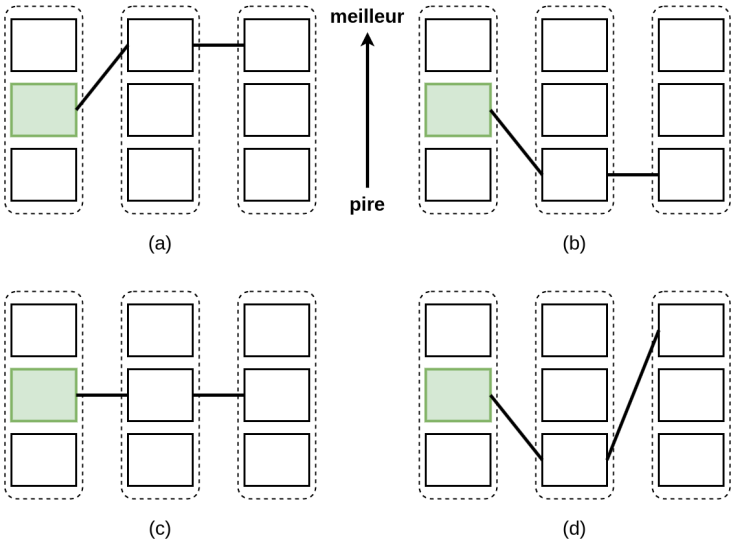


Figure 3.21 – Stratégies de collaboration : pour évaluer chaque coindividu de la sous-population i (en vert), on sélectionne (a) le meilleur coindividu, (b) le pire coindividu, (c) le coindividu lié ou (d) un coindividu choisi aléatoirement dans chaque autre sous-population.

Les stratégies de collaboration sont le plus souvent mises en place lorsque la CC suit un schéma multi-populationnel. Cependant, la CC peut être réalisée à partir d'une ou de plusieurs populations. Nous détaillons les différences ci-dessous.

3.4.1.4 Schéma mono ou multi-populationnel

Dans un algorithme de CC à population unique, les coindividus sont évalués en interagissant avec d'autres coindividus de cette même population. Dans (É. Lutton et al. 2023), l'objectif est de placer de manière optimale un ensemble de cercles (lampes) d'un rayon donné, de manière à couvrir entièrement un champ carré. Chaque coindividu correspond à un placement, à une coordonnée dans ce champ. Ici, la contribution de chaque coindividu de la population est relative à celle des autres.

Dans un algorithme de CC multi-populationnel (deux populations ou plus), les coindividus d'une population interagissent avec les coindividus d'une ou plusieurs autres populations. (Bugajska et Schultz 2002) introduit un algorithme de CC à deux sous-populations pour faire évoluer la morphologie et le comportement d'un drone dans le but de réaliser une navigation sans collision dans un environnement virtuel. Un coindividu de la première sous-population est un vecteur représentant le choix des capteurs (leur nombre et la largeur de chaque faisceau), tandis qu'un coindividu de la seconde sous-population est un vecteur représentant les poids d'un réseau de neurones artificiels.

Nous avons fait le choix d'investiguer ces deux types de CC.

CC avec GA. L'algorithme génétique basé sur la coévolution coopérative (CCGA) est le premier algorithme d'EA proposé par (Potter et Kenneth A. De Jong 1994). Dans sa procédure, un problème d'optimisation de n variables est d'abord décomposé en n sous-problèmes unidimensionnels. Chaque sous-problème est ensuite optimisé par un GA distinct. Un individu complet est obtenu en combinant les coindividus représentatifs de chaque sous-population. La valeur de fitness d'un coindividu dans une sous-population particulière est évaluée en fonction du ou des individus complets auxquels il participe. Les sous-populations évoluent selon le principe du *round-robin*, c'est-à-dire qu'elles se voient attribuer les mêmes ressources computationnelles.

Évolution groupée. L'algorithme d'évolution groupée (GE) a été proposé par (Sanchez, Squillero et Tonda 2011). Cette approche de CC utilise une population unique de coindividus et exploite des ensembles d'individus complets appelés *groupes*. Au cours du processus de recherche, une population de coindividus évoluent comme dans un algorithme d'EA traditionnel, et chaque coindividu fait partie d'un ou plusieurs groupes où un groupe représente un individu complet.

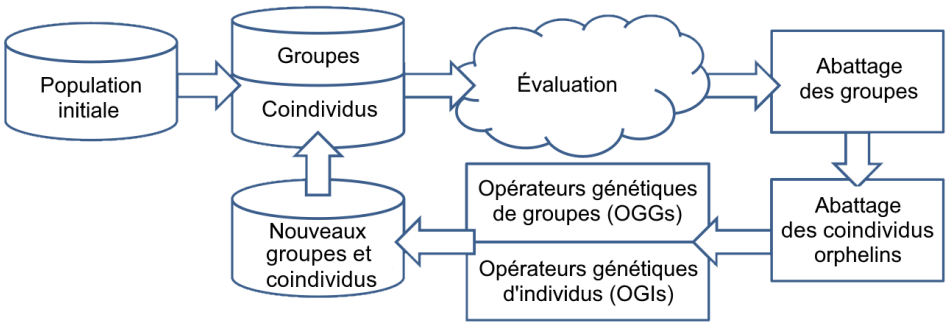


Figure 3.22 – Schéma évolutionnaire extrait de (Tonda, E. Lutton et Squillero 2012) de l'évolution groupée. Les groupes sont un ensemble de coindividus. À chaque étape, de nouveaux groupes et coindividus sont produits.

Le schéma de la figure 3.22 résume les principales étapes de GE. La première étape du processus d'évolution consiste à créer une population initiale de taille fixe de coindividus et à produire des groupes formés de coindividus choisis de manière aléatoire. Chaque coindividu est inclus, au moins, dans un groupe. La seconde étape attribue une valeur de fitness à chaque coindividu en fonction de l'évaluation des groupes auxquels il participe. Une étape d'abattage (*slaughtering*) des groupes les moins bons est ensuite réalisée. Cette

étape provoque un abattage des coindividus qui ne font parti d'aucun groupe : les coindividus orphelins. S'ensuit une étape de génération de nouveaux coindividus et de nouveaux groupes. Ces coindividus sont produits avec des opérateurs classiques (mutation, reproduction, ...) appelés, ici, opérateurs génétiques d'individus (OGIs). Les groupes, quant à eux, sont générés à partir d'opérateurs spécifiquement conçus : les opérateurs génétiques de groupes (OGGs). Trois OGGs sont présentés :

— l'opérateur de croisement de deux groupes (A et B) est un opérateur générant deux nouveaux groupes en échangeant un coindividu du groupe A avec un coindividu du groupe B ;

— l'opérateur d'union génère un seul nouveau groupe en sélectionnant des coindividus de deux groupes parents ;

— l'opérateur de remplacement crée un seul nouveau groupe en remplaçant un coindividu d'un groupe parent par un coindividu de la population, pris au hasard.

Comme les résultats de l'étude expérimentale précédente (section 3.3.2) diminuent à mesure que le nombre de variables du vecteur de décision augmente, nous cherchons à évaluer ces méthodes de CC sur notre problème d'identification de paramétrisations des RRGHs. Nous commençons par introduire des stratégies de décomposition spécifiques au domaine d'application.

3.4.2 Contribution

Nous proposons deux stratégies pour décomposer le problème d'identification de paramétrisations des RRGHs. Elles appartiennent toutes les deux à la catégorie des groupements de variables statiques qui utilisent des connaissances limitées, mais spécifiques au domaine.

3.4.2.1 Décomposition par état discret ($\mathcal{D}_{etat}$).

La première stratégie propose une découpe du vecteur de décision par état discret. Les variables de décision appartenant au vecteur célérité d'un même état discret sont regroupées entre elles pour former le génotype d'un coindividu. De manière générale, les variables de décision d'un coindividu dans l'état discret η^i forment l'ensemble suivant :

$$x^{\eta^i} = \{C_{v,\omega,\eta_v} \mid \eta = \eta^i\}$$

avec ω l'ensemble des prédécesseurs de v.

L'application de cette stratégie au cycle négatif (2G) résulte en la décomposition d'un individu complet en quatre coindividus possédant le génotype suivant : $x^{(0,0)} = \{C_{v_1,\{m_1\},0}, C_{v_2,\emptyset,0}\}$, $x^{(1,0)} = \{C_{v_1,\{m_1\},1}, C_{v_2,\{m_2\},0}\}$, $x^{(0,1)} = \{C_{v_1,\emptyset,0}, C_{v_2,\emptyset,1}\}$ et $x^{(1,1)} = \{C_{v_1,\emptyset,1}, C_{v_2,\{m_2\},1}\}$. Le nombre de sous-problèmes correspond au nombre d'états discrets par lesquels passe la trace hybride.

3.4.2.2 Décomposition par composante ($\mathcal{D}_{comp}$).

La deuxième stratégie consiste à découper le vecteur de décision original par variable du graphe d'interaction $v \in V$. Chaque variable de décision codant pour les célérités d'une

même variable du graphe d'interaction composent le génotype du même coindividu. De manière générale, on a :

$$x^{v_i} = \{C_{v,\omega,n} \mid v = v_i\}$$

Si on applique cette stratégie au RRGH représentant un cycle négatif (2G), un individu complet est décomposé en deux coindividus :
$x^{v_1} = \{C_{v_1,\{m_1\},0}, C_{v_1,\{m_1\},1}, C_{v_1,\emptyset,0}, C_{v_1,\emptyset,1},\}$ et $x^{v_2} = \{C_{v_2,\emptyset,0}, C_{v_2,\{m_2\},0}, C_{v_2,\emptyset,1}, C_{v_2,\{m_2\},1}\}$.
Ici, le nombre de sous-problèmes à optimiser est égal au nombre de variables du graphe d'interaction.

Nous proposons de mener une étude expérimentale sur le problème d'optimisation étudié en combinant chacune des deux stratégies de décomposition avec un algorithme de CC mono-populationnel, puis un multi-populationnel.

3.4.3 Étude expérimentale

Avant de réaliser l'étude expérimentale, nous détaillons d'abord les spécificités techniques relatives à l'adaptation des algorithmes à notre problème.

3.4.3.1 Adaptation des algorithmes à notre problème

CCGA Chaque sous-problème est optimisé à l'aide d'une sous-population de GA. À chaque génération, un individu complet du problème original est obtenu en combinant le coindividu d'une sous-population donnée avec le coindividu représentant de chacune des autres sous-populations. Le représentant est choisi comme étant le coindividu avec la valeur de fitness minimale calculée lors de la génération précédente (*single-best*). Lors de la première génération, comme il n'y a aucune valeur de fitness, les représentants sont élus de manière aléatoire. La valeur de fitness d'un coindividu est donnée par l'évaluation de l'individu complet auquel le coindividu participe. Les sous-populations évoluent selon le principe *round-robin* et un même budget leur est assigné. Le parallélisme des sous-populations est implémenté pour accélérer le processus d'optimisation.

GE Un des avantages de GE est de pouvoir définir un nombre dynamique de coindividus par groupe : deux groupes peuvent avoir un nombre de coindividus différent. Dans notre problème, ce nombre est fixe et connu à l'avance. Le vecteur de décision de l'individu complet est de taille 8 pour le RRGH 2G, 16 pour 3G et 56 pour 5G. Pour 2G, dans le cas de la décomposition $\mathcal{D}_{etat}$, il y a donc forcément 4 coindividus par groupe. Dans le cas de la décomposition $\mathcal{D}_{comp}$, il y a forcément 2 individus. La liste complète est fournie dans le plan d'expérimentations de la section 3.4.3.2. GA est choisi comme algorithme d'optimisation de la population de coindividus. L'évaluation d'un coindividu est basée sur sa meilleure valeur parmi les groupes auxquels il participe. Enfin, en raison du schéma à population unique, le parallélisme ne se produit que lors des étapes d'évaluation et de calcul des opérateurs.

3.4.3.2 Expérimentations et résultats

Plan d'expérimentations. Le but de cette étude est d'évaluer les performances de chaque algorithme avec différentes stratégies de décomposition. On note l'ensemble des

algorithmes $\mathcal{A} = \{\text{CCGA}, \text{GE}\}$ et l'ensemble des stratégies $\mathcal{D} = \{\mathcal{D}_{etat}, \mathcal{D}_{comp}\}$. On évalue ainsi le produit cartésien : $\mathcal{A} \times \mathcal{D}$. GA est gardé comme algorithme de référence pour comparer les résultats obtenus par les méthodes de CC. L'implémentation de chacun des algorithmes de $\mathcal{A}$ est basée sur celle de GA de `pymoo`. Une taille initiale de 500 individus est allouée à l'ensemble des sous-populations. 25 expérimentations indépendantes sont exécutées. Le critère de terminaison est de 40 000 NFEs pour 2G, 52 500 pour 3G et 82 500 pour 5G.

Le tableau 3.6 présente le nombre de coindividus pour former un individu complet ou un groupe en fonction du réseau et de la stratégie de décomposition choisie. Pour CCGA, il s'agit du nombre de sous-populations et pour GE, il s'agit du nombre de codindividus requis pour former un groupe. La première ligne correspond à la décomposition par état discret. Chaque élément représente le nombre d'états par lesquels la trace doit passer et qui concorde à la longueur du triplet de Hoare au réseau. La seconde ligne correspond au nombre de variables du graphe d'interaction.

	2G	3G	5G
$\mathcal{D}_{etat}$	4	6	12
$\mathcal{D}_{comp}$	2	3	5

Tableau 3.6 – Nombre de coindividus pour former un individu complet en fonction de la dimension du RRGH et de la stratégie de décomposition choisie.

Analyse des résultats. Le tableau 3.7 présente les statistiques des résultats obtenus pour chaque combinaison d'un algorithme et d'une décomposition ($\mathcal{A} \times \mathcal{D}$) sur 25 exécutions indépendantes. Le minimum, la moyenne et l'écart-type sont indiqués ainsi que le taux de réussite biologique (BSR). Pour rappel, le BSR peut être considéré comme une modification du *success rate* traditionnel, mais avec une erreur de précision ϵ. Le meilleur résultat colonne par colonne est en gras : plus il est bas, mieux c'est, sauf pour le BSR. D'après les résultats en gras, on observe que GA obtient de meilleurs résultats que les algorithmes de CC quelle que soit la stratégie de décomposition employée. Pour les réseaux 2G et 3G, CCGA avec la stratégie de décomposition $\mathcal{D}_{comp}$ obtient des résultats proches de GA. Pour 5G, GA est le seul algorithme à obtenir des solutions.

Les courbes de répartition sont présentées sur la figure 3.23. Chaque courbe décrit la probabilité (en ordonnée) qu'une solution soit trouvée à un niveau égal ou inférieur à un score de fitness (en abscisse) compte tenu d'un budget spécifique. Cette représentation permet de tirer des conclusions sur l'algorithme à privilégier dans chaque expérience. Par exemple, dans l'expérience sur le cycle cellulaire (5G), la probabilité qu'un utilisateur obtienne une solution à un score de fitness inférieur ou égal à 1 (10^0) avec GA (courbe bleue) est d'environ 70 % pour un NFE de 82 500. Les figures montrent que CCGA et GE ne présentent pas d'intérêt par rapport à GA dans toutes les expériences considérées. La probabilité de GA d'obtenir les meilleurs résultats est supérieure pour chaque configuration.

Finalement, la figure 3.24 représente les valeurs de fitness des meilleures solutions obtenues en fonction de la taille du RRGH (le nombre de variables du graphe d'interaction). Plus un point est bas, meilleure est la solution obtenue. Cette visualisation illustre la capacité d'un algorithme à prévenir le problème de l'augmentation de la dimension

RRGH	$\mathcal{A}+\mathcal{D}$	min	avg	std	BSR
2G	GA	**2e-4**	**9e-4**	**5e-4**	**1.**
	CCGA + $\mathcal{D}_{comp}$	3e-3	3e-2	6e-2	0.96
	CCGA + $\mathcal{D}_{etat}$	8e-2	3.9354	5.1962	0.12
	GE + $\mathcal{D}_{comp}$	2e-2	0.6819	0.5857	0.12
	GE + $\mathcal{D}_{etat}$	7e-2	0.8725	0.7074	0.04
3G	GA	**3e-8**	**3e-7**	**2e-7**	**1.**
	CCGA + $\mathcal{D}_{comp}$	2e-5	1e-4	9e-5	**1.**
	CCGA + $\mathcal{D}_{etat}$	1e-3	0.5956	1.7852	0.72
	GE + $\mathcal{D}_{comp}$	6e-2	0.8272	0.7919	0.1
	GE + $\mathcal{D}_{etat}$	0.1515	1.1267	1.1318	0.0
5G	GA	**1e-4**	**0.5795**	**1.0594**	**0.64**
	CCGA + $\mathcal{D}_{comp}$	0.1212	1.0956	1.3684	0.0
	CCGA + $\mathcal{D}_{etat}$	2.6257	16.0334	7.981	0.0
	GE + $\mathcal{D}_{comp}$	2.7263	5.5219	1.5198	0.0
	GE + $\mathcal{D}_{etat}$	6.1689	23.2717	10.0134	0.0

Tableau 3.7 – Résultats obtenus par les différents algorithmes couplés aux stratégies de décomposition proposées sur les trois réseaux.

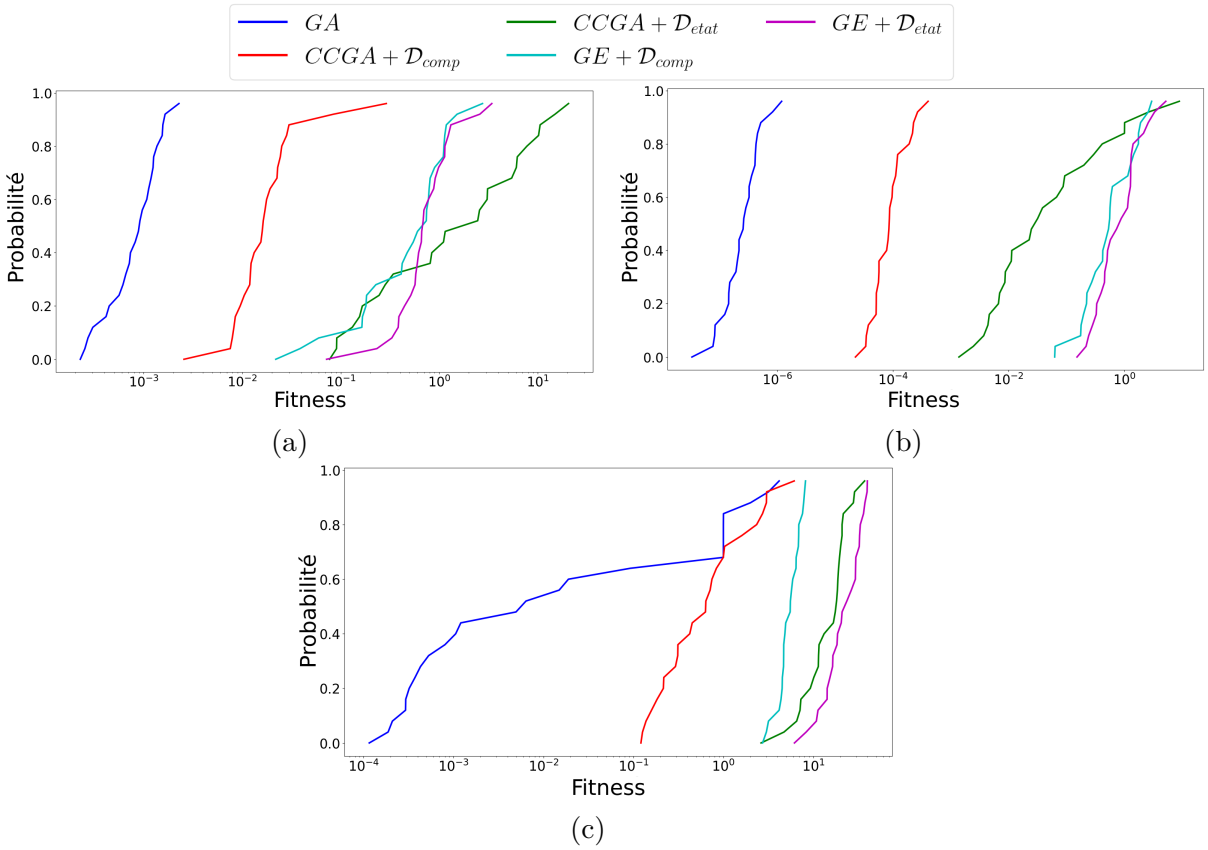


FIGURE 3.23 – Courbes de répartitions pour les réseaux (a) 2G , (b) 3G et (c) 5G.

du problème. Dans le contexte d'un problème de minimisation, une pente négative entre deux points du graphique permet d'illustrer que plus la taille du problème augmente, plus l'algorithme est performant. Or, aucun algorithme ne présente vraiment de propriété intéressante, car entre 3G et 5G les performances diminuent : la pente est positive.

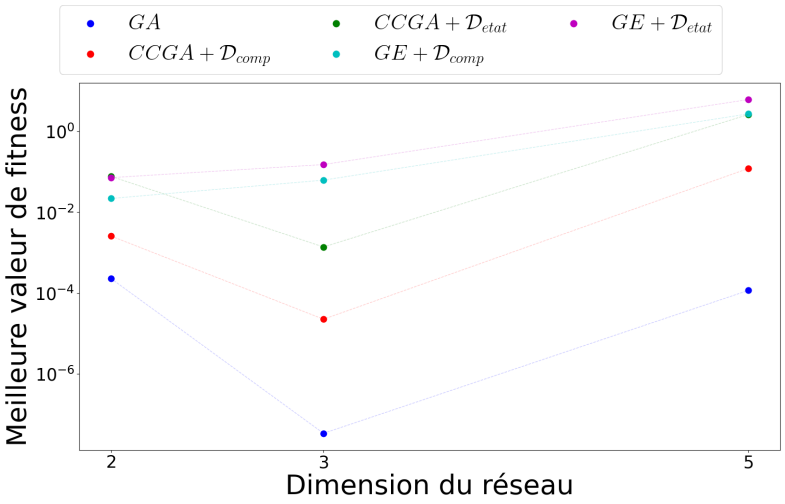


FIGURE 3.24 – Comparaison des meilleurs résultats obtenus par les différentes stratégies et algorithmes par dimension. Les points représentent les meilleures valeurs de fitness obtenues pour toutes les exécutions indépendantes réalisées.

Visualisation des résultats Les illustrations de la figure 3.25 montrent la meilleure trace obtenue par chacune des configurations (algorithme et stratégie de décomposition) pour 2G et 3G. Il est à noter que pour $GE+\mathcal{D}_{etat}$ appliqué au cycle circadien (3G), aucune trace n'est illustrée, car aucune solution n'a été identifiée. C'est aussi pour cette raison que seule la meilleure trace de GA a été intégrée dans la représentation de la figure 3.26. Les quatre autres stratégies de CC n'ont pas permis d'extraire de solutions, *cf.* tableau 3.7.

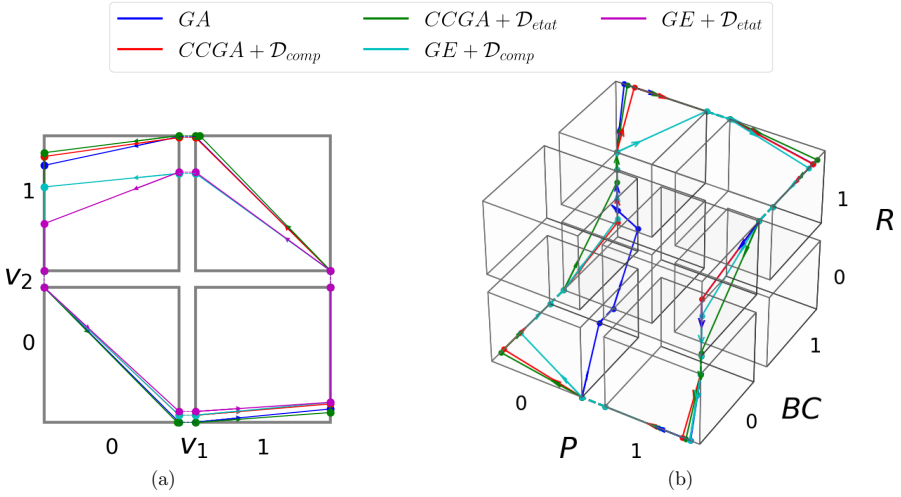


FIGURE 3.25 – Visualisation des meilleures traces obtenues avec les différentes paires algorithme-décomposition sur le réseau (a) 2G et (b) 3G.

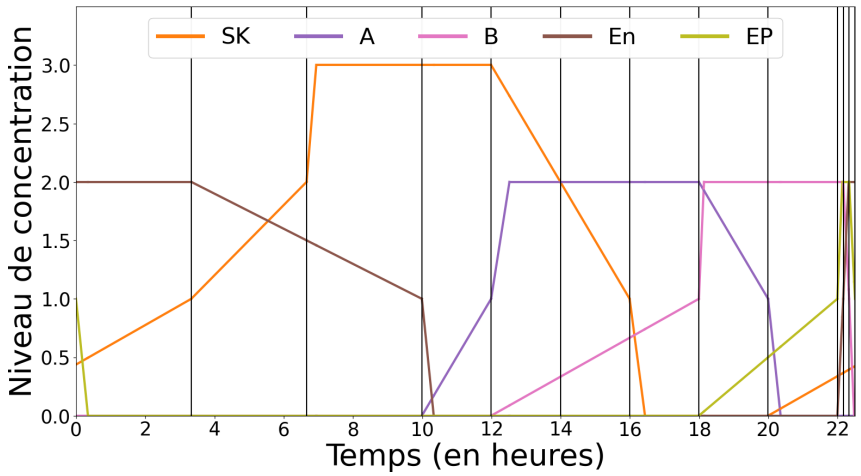


FIGURE 3.26 – Visualisation de la meilleure trace obtenue par GA sur le cycle cellulaire (5G).

3.4.3.3 Limites et discussion

Ces résultats montrent que les performances de GA sont meilleures que celles des méthodes de CC, quelle que soit la stratégie de décomposition utilisée. Il est important de s'intéresser aux raisons de ces sous-performances en formulant des hypothèses afin de proposer des possibilités d'amélioration. Les différents points mentionnés ci-dessous sont extraits d'un échange avec Évelyne Lutton :

— la taille de la sous-population est inadéquate. Une population plus petite pourrait mieux fonctionner, en divisant par exemple la taille initiale de la population de GA par le nombre de variables de décisions ($\frac{|\mathcal{P}|}{n}$). Pour un même budget, une taille plus petite permettrait de prolonger le processus d'optimisation ;

— les stratégies de décomposition ne sont pas efficaces. Les représentations ne sont peut-être pas pertinentes. D'autres stratégies peuvent être envisagées : représenter le vecteur célérité en coordonnées polaires ou comme un angle et une intensité et découper ces deux sous-composantes. On pourrait aussi travailler sur l'identification de connaissances spécifiques au domaine avec une expertise biologique plus poussée, voire choisir une décomposition dynamique ;

— les instances du problème sont trop petites pour traiter de la coévolution. Le nombre de variables du vecteur de décision du problème initial est peut-être trop faible, ce qui expliquerait la supériorité de GA par rapport aux méthodes de coévolution impliquant les stratégies de décomposition. Si nous avions à disposition des réseaux de régulation de taille plus importante, la tendance pourrait alors peut-être s'inverser ;

— le problème est non-séparable. Ce que tendrait d'ailleurs à montrer la dynamique selon laquelle une décomposition à plus gros grain est plus performante qu'une à grain plus fin (pas de décomposition $> \mathcal{D}_{comp} > \mathcal{D}_{etat}$) ;

3.5 Synthèse

Dans ce chapitre, nous montrons comment nous reformulons le CSP comme un PO. Dans ce PO, la fonction objectif introduite permet d'évaluer la distance entre un vecteur de décision simulé et les informations de la connaissance biologique. L'objectif est de la minimiser. Nous démontrons qu'en utilisant cette fonction objectif, il est équivalent de résoudre le CSP ou le PO : il y a une garantie d'obtenir des solutions au problème si et seulement si la fonction objectif est minimisée. Nous faisons le choix d'utiliser des algorithmes d'EA pour résoudre ce problème. Deux variantes de fonction de fitness, une additive et une multiplicative, sont proposées. À travers une étude expérimentale, nous comparons les performances obtenues en fonction du choix de la fonction de fitness et de la métaheuristique. Nous obtenons une solution optimale, une paramétrisation qui est en accord avec la connaissance biologique pour chaque RRGH à l'aide de la fonction de fitness additive combinée à GA et à CMA-ES.

Cependant, à mesure que le nombre de variables du vecteur de décision augmente, les résultats des algorithmes diminuent. Pour faire face à ce problème de passage à l'échelle, nous étudions la coévolution coopérative. Nous proposons deux stratégies pour décomposer le problème d'optimisation en sous-problèmes de dimensions inférieures. Lors de cette seconde étude expérimentale, deux algorithmes de CC sont testés, l'un mono-populationnel et l'autre multi-populationnel. Les résultats obtenus ne parviennent pas à surpasser ceux extraits de la première étude expérimentale. Nous discutons des limites de cette approche.

Dans le chapitre suivant, nous proposons une nouvelle formulation du problème. La recherche d'une paramétrisation d'un RRGH est envisagée comme un problème de décision séquentiel dans lequel l'identification des paramètres dynamiques est réalisée séquentiellement.

4

Problème de décision séquentiel avec la recherche arborescente Monte Carlo

I grant, that good and evil, reward and punishment, are the only motives to a rational creature ; these are the spur and reins whereby all mankind are set on work and guided [...].

Extrait de (Locke 1996)

En informatique, et plus particulièrement dans le domaine de l'intelligence artificielle, la prise de décision séquentielle (*sequential decision making*) est une tâche fondamentale à laquelle fait face un agent qui interagit avec son environnement, d'après (Russell et Norvig 2016). L'agent joue le rôle de décideur et l'environnement fait référence à tout ce qui est externe à l'agent. Selon (Littman 1996), la prise de décision séquentielle est l'action de répondre à la question « Que dois-je faire maintenant ? », où « je » est l'agent, « faire » correspond au choix d'une action par l'agent, « maintenant » fait référence à l'état dans lequel se trouve l'agent en cet instant et « dois » est la maximisation d'une récompense sur le long terme. Dans ce contexte, il apparaît que l'objectif de la résolution d'un problème de décision séquentiel est de développer un agent capable d'interagir de manière optimale avec son environnement, c'est-à-dire, en maximisant une récompense à long terme. Un tel problème peut se modéliser comme un processus de décision markovien (PDM) où ce dernier représente un modèle de l'interface agent-environnement.

Dans ce chapitre, la section 4.1 débute sur une introduction des concepts relatifs à la définition d'un problème de décision séquentiel. Sur ces bases, la section 4.1.2 propose une formulation du problème d'identification des paramètres dynamiques d'un RRGH en un PDM. Pour résoudre ce PDM, nous choisissons la recherche arborescente Monte Carlo (*Monte Carlo Tree Search*, MCTS), parmi l'ensemble des techniques d'apprentissage par renforcement. Ensuite, nous présentons un état de l'art de MCTS en section 4.2 qui se concentre d'abord sur la résolution de problèmes combinatoires dans lesquels le domaine des états et celui des actions sont discrets pour arriver au cas qui nous intéresse où les domaines sont continus. La section 4.3 propose des contributions heuristiques pour améliorer les performances de MCTS dans ce cadre. Finalement, en section 4.4, nous

montrons, à travers les résultats d'une étude expérimentale, que ces contributions nous permettent de résoudre le problème d'identification des paramètres dynamiques.

4.1 D'un problème d'optimisation à un problème de décision séquentiel

Dans cette première section, nous réalisons d'abord une introduction aux problèmes de décision séquentiels (PDS) en utilisant le cadre formel du processus de décision markovien. Nous reformulons ensuite le problème d'identification des paramètres dynamiques dans les RRGHs. Le PDS est énoncé sur les bases de la formulation du problème d'optimisation du chapitre précédent (section 3.1.2.1). Nous introduisons, finalement, la recherche arborescente Monte Carlo parmi l'ensemble des techniques d'apprentissage.

4.1.1 Introduction aux problèmes de décision séquentiels

Une grande variété de tâches dans le domaine de l'intelligence artificielle peuvent être formalisées sous la forme d'un processus de décision markovien. Dans ce contexte, nous appelons le décideur l'agent, un algorithme d'apprentissage par exemple, et tout ce qui se trouve à l'extérieur de l'agent son environnement. À chaque **pas de temps**, l'agent reçoit des observations de son environnement, que l'on nomme **état**, et exécute une **action** conformément à sa **politique** (qui détermine son comportement dans l'environnement). L'environnement fournit alors en retour un signal sous la forme d'une **récompense** et place l'agent dans un nouvel état en suivant une **fonction de transition**. Cette série temporelle d'actions, d'états et de récompenses définit l'expérience de l'agent. En fonction de cette expérience, l'objectif d'un agent est d'identifier la séquence d'actions qui maximise une récompense cumulée espérée.

Les différentes notions en gras composent un problème de décision séquentiel et sont introduites ci-après, en suivant les références (Otterlo et Wiering 2012 ; Russell et Norvig 2016 ; Sutton et Barto 2018). Pour illustrer ces concepts, nous nous appuierons sur l'exemple fourni en figure 4.1, dans lequel un agent interagit dans un environnement représenté par une grille 4×3. L'objectif de l'agent est de partir de la cellule « Départ » en A1 (état initial) pour atteindre « Arrivée » (état terminal) en C4, en évitant l'obstacle (cellule grisée). L'ensemble des actions est $\{Haut, Bas, Gauche, Droite\}$ et est matérialisé par l'ensemble des flèches directionnelles en figure 4.1b. En figure 4.1c, une fonction de transition stochastique est illustrée. Si l'agent choisit l'action $Haut$, alors le résultat prévu (aller en haut) a une probabilité de 0.6 de se produire, mais il y a une probabilité de 0.3 que l'agent se déplace à gauche et une probabilité de 0.1 pour qu'il aille à droite (0.1).

4.1.1.1 Composants d'un problème de décision séquentiel

Pas de temps. Le pas de temps t représente l'instant auquel une décision (le choix d'une action) doit être prise par l'agent. Dans certains cas, comme dans (Doya 1995 ; Bertsekas et Tsitsiklis 1996), l'agent peut décider à n'importe quel moment de prendre une décision : la prise de décision est réalisée en continu. Dans ce manuscrit, nous considérons que le temps est discrétisé et est ainsi appelé pas de temps (*timestep*). Attention, cela ne signifie pas pour autant que le temps dans l'environnement évolue de

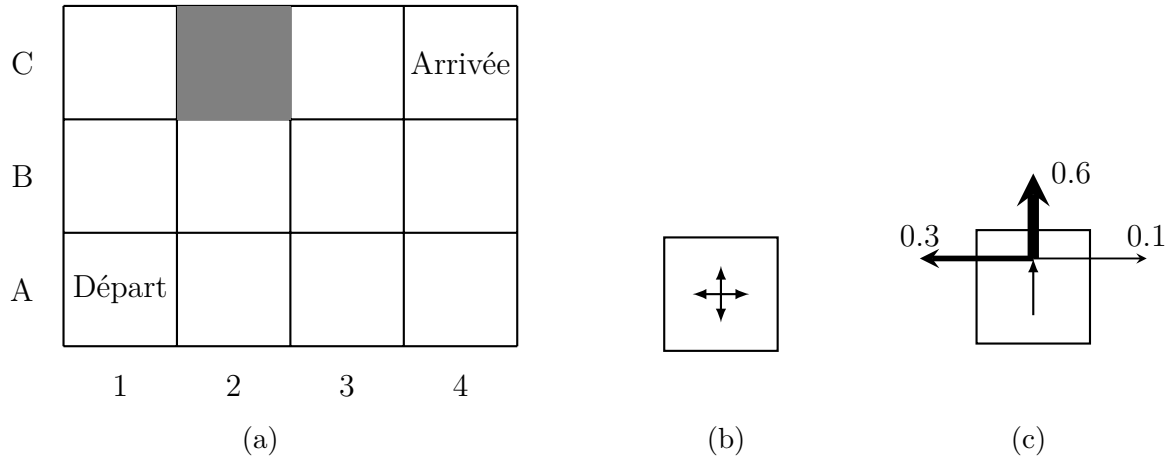


FIGURE 4.1 – (a) : Problème de décision séquentiel pour un agent dans un environnement simple composé d'une grille de 4×3, inspiré de (RUSSELL et NORVIG 2016). (b) : L'ensemble des actions possibles d'un agent dans cet environnement est matérialisé par les flèches directionnelles et correspond à $\{Haut, Bas, Gauche, Droite\}$. (c) : Exemple d'une fonction de transition stochastique où le choix de l'action $Haut$ a une probabilité de 0.6 d'arriver sur l'état du haut, 0.3 d'arriver sur l'état à gauche et 0.1 d'arriver sur l'état à droite.

manière discrète. Le temps « réel » entre deux décisions de l'agent, *i.e.*, pas de temps, n'est pas nécessairement le même. L'exemple du jeu de golf permet d'illustrer ce concept. Le but du golfeur (l'agent) consiste à envoyer une balle dans un trou à l'aide de clubs en effectuant le moins de coups possible sur un parcours défini. Au début du jeu ($t = 0$), le golfeur utilise son club pour envoyer la balle dans le trou. La balle effectue une trajectoire qui dure un certain temps, par exemple 10 secondes. Si la balle n'a pas atteint le trou, le golfeur utilise de nouveau son club ($t = 1$) et la balle effectue une trajectoire différente qui dure plus ou moins de temps que la précédente, 3 secondes par exemple.

Quand il est nécessaire de distinguer plusieurs états ou actions en fonction de leur choix dans le temps, les variables sont indexées par t. Dès lors que l'on parle d'un état (resp. une action) parmi l'ensemble des états (resp. actions) possibles, les variables ne sont pas indexées.

État. Un état s est l'information utilisée par l'agent pour déterminer la prochaine action à effectuer. Un état peut contenir des informations sur l'état physique de l'agent, un signal de l'environnement, ou encore des informations sur l'historique de l'agent ou de l'environnement. Le problème peut considérer que l'environnement est entièrement observable (l'agent sait toujours où il se trouve) ou partiellement observable (l'agent n'est pas un observateur omniscient, il lui manque certaines informations). Dans l'exemple de la figure 4.1, si l'environnement est entièrement observable, on peut définir l'état de l'agent s comme sa position xy avec $x \in \{A, B, C\}$ et $y \in \{1, 2, 3, 4\}$, la position de l'obstacle (cellule grisée) et la position de l'état terminal (cellule notée « Arrivée »). Si l'environnement est partiellement observable, on peut définir l'état de l'agent comme sa seule position. Dans le cas d'un environnement entièrement observable, si une information est nécessaire pour la prise de décision, c'est-à-dire calculer le prochain état, alors elle doit être ajoutée.

Dans le cas contraire, elle ne doit pas être incluse [1]. Dans ce manuscrit, nous considérons seulement le cas où l'environnement est entièrement observable. L'ensemble des états est noté S. On distingue souvent deux types d'états. Un **état terminal** est un état où le processus de décision prend fin : aucune action ne peut être entreprise, car le processus ne peut pas évoluer vers un autre état. À l'inverse, un **état non terminal** est un état où le processus peut continuer à évoluer vers un ou plusieurs autres états.

Action. Une action a correspond à une prise de décision. Cette action permet à l'agent d'interagir avec son environnement. Le but d'un agent est de sélectionner la meilleure séquence d'actions possible. L'ensemble des actions est noté A. Dans l'exemple de la figure 4.1, on a $A = \{Haut, Bas, Gauche, Droite\}$ correspondant à l'action de se déplacer vers la case adjacente en haut ($Haut$), en bas (Bas), à gauche ($Gauche$) et à droite ($Droite$). Il est courant de parler d'actions légales, pour se référer à des actions qui peuvent être choisies par l'agent dans un état particulier à un pas de temps donné $s_t \in S$. L'ensemble des actions légales est ainsi dénoté $A(s_t)$ et est un sous-ensemble de $A : A(s_t) \subseteq A$. Dans la figure 4.1, on pourrait considérer que lorsque l'agent est proche d'un mur, on retire de l'ensemble des actions légales l'action qui ferait se collisionner l'agent dans le mur extérieur de la grille. Par exemple, en C1, on aurait $A(s = C1) = \{Bas\}$. Si le problème de décision possède un seul espace d'actions réalisables pour chaque état $s_t \in S$, on le note A.

Fonction de transition. La fonction de transition décrit les issues de chaque action dans chaque état. En appliquant une action $a_t \in A$ dans un état $s_t \in S$, une transition est réalisée de s_t vers un nouvel état $s_{t+1} \in S$ grâce à la fonction de transition. On peut notamment distinguer deux types de transition. Le premier type considère que l'environnement est stochastique, c'est-à-dire que lorsque l'agent décide d'une action, l'état d'arrivée est incertain et cette incertitude dépend de l'environnement. Dans ce cas, la fonction de transition T est définie par $T : S \times A \times S \to [0, 1]$. Elle représente la probabilité que l'agent arrive dans un état s_{t+1} après avoir choisi l'action a_t dans l'état s_t. T définit une distribution sur les prochains états possibles : autrement dit $\forall s_t \in S, a_t \in A, \sum_{s_{t+1} \in S} T(s_t, a_t, s_{t+1}) = 1$.

Dans le second type de transition, lorsque les transitions sont déterministes, il n'y a pas d'incertitude. Étant donné un état s_t et une action a_t, la fonction de transition retourne un unique état d'arrivée. On définit la fonction $T : S \times A \to S$ qui associe un nouvel état s_{t+1} résultant de l'action a_t dans l'état s_t. Dans l'exemple de la figure 4.1, si l'environnement est déterministe, alors lorsque l'agent choisit l'action $Haut$ dans la position A3, sa nouvelle position est forcément B3. À l'inverse, si la fonction de transition est stochastique (figure 4.1c), alors il n'y a plus qu'une probabilité de 0.6 que l'agent atteigne la position B3 (0.3 pour atterrir en A2 et 0.1 pour A4).

Fonction de récompense. Bien que le terme de récompense soit usuellement utilisé, une récompense est représentée par une valeur numérique $r_t \in R$ et peut donc être négative (punition) ou positive (récompense) en fonction du problème. La fonction de récompense joue un rôle important dans la définition d'un processus de décision séquentiel, car elle

1. Dans un jeu de plateau comme le jeu de Go par exemple, informer l'agent que l'adversaire porte un t-shirt bleu ne servirait à rien, mis à part ajouter du temps de calcul gratuitement.

spécifie implicitement l'objectif d'apprendre. Le but d'un agent est de maximiser la récompense cumulée (long terme), pas seulement la récompense immédiate (court terme). Souvent, cette fonction est utilisée par l'agent pour diriger son apprentissage. L'idée qui sous-tend cette notion d'apprentissage est appelée « hypothèse de récompense » (« *reward hypothesis* »). Cette dernière énonce que tous les objectifs d'un agent peuvent être décrits par la maximisation de l'espérance de la récompense cumulative [2]. Utiliser le signal de récompense pour atteindre un objectif implique que la fonction de récompense est un aspect essentiel du mécanisme d'apprentissage et qu'elle doit être conçue avec attention. Dans le cas de la grille 4×3, il existe plusieurs manières de concevoir la fonction de récompense. Par exemple, on peut considérer que pour chaque déplacement vers un état non terminal, la récompense est de -1, et lorsque l'agent atteint l'état terminal, la récompense est de $+1$. L'idée est ainsi de pousser l'agent à minimiser son chemin pour maximiser la récompense cumulée.

La fonction de récompense est définie en fonction du problème. Trois choix sont possibles :

— $R : S \to \mathbb{R}$ est la fonction état-récompense, car elle spécifie la récompense de l'agent pour être arrivé dans un état ;

— $R : S \times A \to \mathbb{R}$ récompense le choix d'une action étant donné un état ;

— $R : S \times A \times S \to \mathbb{R}$ évalue la transition complète (après avoir choisi une action a_t dans un état s_t et être arrivé dans un nouvel état s_{t+1}).

4.1.1.2 Un processus de décision markovien

Nous considérons le problème de décision séquentiel comme un processus de décision markovien (Puterman 2014 ; Boutilier, Dean et Hanks 1999), c'est-à-dire que les transitions sont markoviennes : le prochain état s_{t+1} dépend seulement de l'état actuel s_t et de l'action choisie a_t. La fonction de transition suit la propriété de Markov spécifiant qu'un état est markovien si et seulement si on a la relation suivante : $\mathbb{P}(s_{t+1} \mid s_t, a_t, s_{t-1}, a_{t-1}, ..., s_0) = \mathbb{P}(s_{t+1} \mid s_t, a_t)$ où $\mathbb{P}(a \mid b)$ est la probabilité conditionnelle de a sachant b et l'état s_t contient toute l'expérience de l'agent et est décrit comme un état markovien. En substance, l'idée est qu'étant donné le présent, le futur est indépendant du passé ou, en d'autres termes, peu importe l'historique de l'agent, l'état courant donne assez d'information pour faire le choix d'une action optimale.

Un PDM peut être défini comme une interface entre l'agent et l'environnement. Plus spécifiquement, l'agent et l'environnement interagissent à chaque pas de temps $t = 0, 1, 2, ...$ Lors de ces interactions, l'agent reçoit une représentation de l'état de l'environnement s_t, et choisit, en fonction de l'état courant, une action $a_t \in A$. Au temps $t + 1$, l'agent, en conséquence de son action, reçoit une récompense r_{t+1} et une représentation de l'état de l'environnement qui vient de changer s_{t+1}. Le processus décrit ci-dessus est illustré par le diagramme de la figure 4.2. L'objectif de l'agent est de choisir les actions qui maximisent la récompense cumulée (espérée, si la notion d'incertitude entre en jeu) et non pas uniquement la récompense immédiate.

2. « *That all of what we mean by goals and pruposes can be well thought of as the maximization of the expected value of the cumulative sum of a recieved scalar signal (called reward).* » (Sutton et Barto 2018)

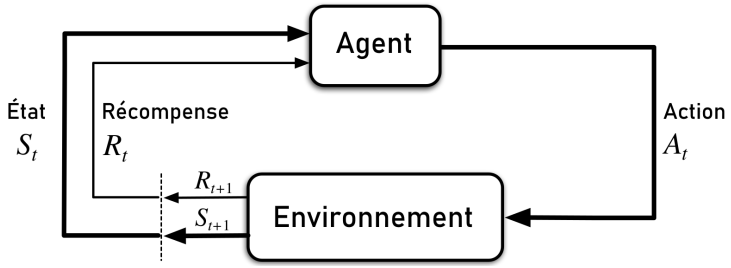


FIGURE 4.2 – L'interaction entre un agent et un environnement comme un processus de décision markovien, extrait de (SUTTON et BARTO 2018).

La définition 18 assemble les différents éléments introduits et un exemple de PDM est illustré en figure 4.3.

Processus de décision markovien (PDM)

Définition 18. Un processus de décision markovien est un tuple $\Pi = \langle S, A, T, R \rangle$ dans lequel S est un ensemble d'états, A un ensemble d'actions, T une fonction de transition et R une fonction de récompense. La fonction de transition T et la fonction de récompense R composent ensemble ce qui est appelé le **modèle du PDM**.
Si l'état initial n'est pas déterminé, il est possible de définir la distribution de l'état initial, notée I. Il s'agit d'une fonction $I : S \to [0, 1]$ qui spécifie la probabilité pour l'agent de débuter dans un état plutôt qu'un autre.

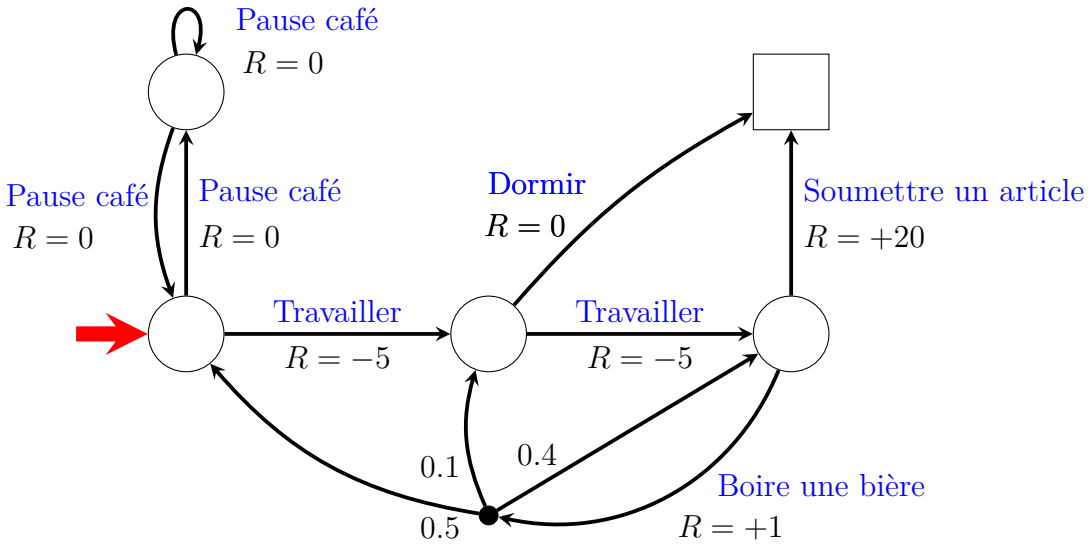


FIGURE 4.3 – Le processus de décision markovien du doctorant, inspiré de (SILVER 2015).

S est composé des états correspondant aux nœuds du graphe. Les états non-terminaux sont représentés par les cercles et l'état terminal par un carré ; A, l'ensemble des actions, est composé des flèches labellisées en bleu dont les labels dénotent les actions : {Pause café, Travailler, Dormir, Boire une bière, Soumettre un article}. La récompense associée à une action est renseignée sous le label de l'action et correspond à la fonction de

récompense R. L'action Boire une bière mène à une transition stochastique. Le point noir indique que plusieurs transitions sont possibles, chacune des flèches sortantes est labellisée par la probabilité de transitionner donnée par la fonction de transition T (la somme des probabilités équivaut à 1).

On distingue deux types de PDMs en fonction de la présence (ou de l'absence) :

— d'états terminaux : un état terminal est un état à partir duquel le processus de décision prend fin ;

— de la notion d'**épisode** : lorsqu'on peut découper l'interaction de l'agent et de son environnement en séquences d'états débutant d'un état initial et finissant dans un état terminal, comme par exemple une partie d'un jeu de plateau ou encore le parcours de la grille 4×3, un épisode correspond à une de ces séquences.

Dans le cas d'une **tâche épisodique** (*episodic task*), un épisode est une séquence d'états de taille finie. L'objectif de l'agent est de démarrer d'un état initial donné et d'arriver à un état terminal. Il existe deux types de tâches finies (*finite episodic task*). La tâche épisodique à horizon fixé (*fixed horizon*) dans laquelle un épisode est de taille finie et fixe. À l'inverse, la tâche pour laquelle un épisode est de taille finie mais arbitraire est à horizon indéfini (*indefinite horizon*). Dans ce dernier type de tâche, lorsque les épisodes sont de taille infinie (*infinite horizon*), on précise que la tâche est **continue** (*continuing task*). La figure 4.3 illustre un PDM de tâche épisodique à horizon indéfini : il existe un état terminal, mais les épisodes ne sont pas de taille fixée. Sans état terminal, le PDM serait considéré comme une tâche continue.

Gain. Un agent ne cherche pas à maximiser la récompense à chaque pas de temps t (objectif à court terme) mais plutôt à maximiser les récompenses accumulées à plus long terme (pour une séquence d'actions). La notion de gain permet de répondre à la question suivante : l'objectif de l'agent étant de maximiser la récompense cumulée, comment définir cet objectif ? En général, cela signifie que l'agent cherche à maximiser la séquence de récompenses reçues après un pas de temps t : $r_{t+1}, r_{t+2}, r_{t+3}, ...$

Dans le cas d'une tâche épisodique à horizon fini, le gain est défini comme étant la somme des récompenses accumulées à partir du pas de temps $t+1$ jusqu'à ce qu'un épisode se termine à l'horizon H (lorsqu'un état terminal est atteint) : $g_t = r_{t+1}, r_{t+2}, ..., r_H = \sum_{k=t+1}^{H} r_k$. En reprenant l'exemple de la figure 4.1a, si on considère que pour chaque pas de temps $t < H$, $r_t = 0$ et que $r_H = 1$, alors le gain sera simplement de 1 quelle que soit la longueur de l'épisode.

Dans le cas d'une tâche continue, l'horizon est infini ($H = +\infty$). Il est commun d'appliquer un facteur de dévaluation (*discount factor*) $\gamma \in]0, 1]$, à la somme des récompenses cumulées : $g_t = \sum_{k=t+1}^{H} \gamma^{k-t-1} r_k$. Ce facteur de dévaluation quantifie dans quelle mesure un agent donne de l'importance aux récompenses actuelles par rapport aux récompenses futures. Quand γ est proche de 0, les récompenses distantes dans le temps ont un faible impact sur le calcul du gain. À l'inverse, quand γ est proche de 1, l'agent aura tendance à autant valoriser les récompenses proches que les récompenses distantes. Dans le cas $\gamma = 1$, il s'agit du cas particulier des récompenses additives.

Le gain est souvent la valeur que l'agent va chercher à maximiser, c'est pourquoi nous choisissons de travailler dans ce cadre.

Politique. Le comportement décisionnaire de l'agent peut être décrit par une politique π. Étant donné un PDM, une politique π est une distribution de probabilités sur les actions étant donné les états. On distingue une politique stochastique qui renvoie la probabilité de choisir une action a en fonction d'un état s à un pas de temps t, d'une politique déterministe qui renvoie directement l'action choisie en fonction d'un état donné. Dans le premier cas, la politique stochastique est notée $\pi : S \times A \to [0, 1]$ avec $\pi(a|s) = \mathbb{P}(a_t = a|s_t = s)$. Dans le second cas, on a $\pi : S \to A$ tel que $\pi(s) = a$. Intuitivement, le résultat d'une politique est l'action recommandée pour l'état courant.

L'application d'une politique à un PDM déterministe peut être résumée comme suit : d'abord un état initial s_0 est généré à partir de la distribution I, ensuite une action est donnée par la politique $a_0 = \pi(s_0)$. À partir de l'action a_0 choisie dans l'état s_0, la fonction de transition T fournit à l'agent un nouvel état $s_1 = T(s_0, a_0)$ et la récompense associée est fournie $r_1 = R(s_0, a_0, s_1)$. Si la tâche est épisodique, le processus termine dans un état terminal. Si la tâche est continue, il s'étend indéfiniment.

Dans un PDM, il existe une politique optimale $\pi^*(s, a)$ qui maximise le gain espéré pour chaque état du PDM : $\pi^*(s, a) = \underset{\pi}{\operatorname{argmax}}\ \mathbb{E}_\pi[g_t \,|\, s_t = s, a_t = a]$, où $\mathbb{E}_\pi$ indique l'espérance suivant la politique π. C'est l'espérance d'obtenir le gain g_t en partant de l'état s, en sélectionnant l'action a et en suivant l'épisode généré par la politique π.

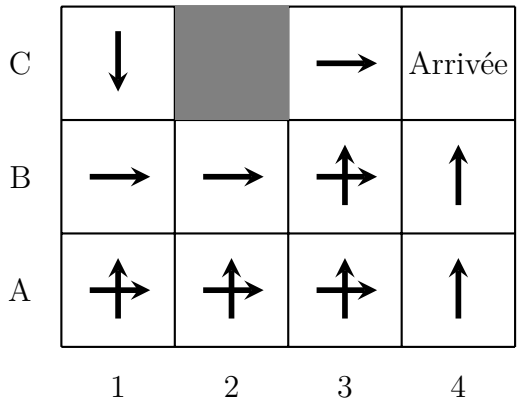


FIGURE 4.4 – Politiques optimales pour l'environnement déterministe, quand $\forall t < H, r_t = -1$ et $r_H = +1$, pour les états non terminaux. Ces politiques optimales sont tous les chemins de longueur égale à la distance de Manhattan séparant l'état actuel s_t de l'état terminal s_H en évitant l'obstacle.

En utilisant les termes et concepts introduits jusqu'à maintenant, nous formulons le problème d'identification de paramètres dynamiques d'un RRGH comme un PDS en définissant le PDM associé.

4.1.2 Formulation du PDS

L'idée de formuler le problème d'identification des paramètres dynamiques de RRGHs comme un problème de décision séquentiel peut paraître moins intuitive, de prime abord, que de le formuler comme un problème d'optimisation numérique (comme introduit dans le chapitre précédent). On considérait jusqu'alors la trace « comme un bloc », dans son ensemble, où l'ensemble est le vecteur de décision. Dans la dernière partie section 3.4, nous

avons considéré des stratégies de décomposition, mais une trace était toujours simulée et évaluée à partir d'un individu complet. Ici, la trace est décomposée séquentiellement : elle est simulée et évaluée étape par étape. Ce choix est notamment motivé par le fait qu'il existe une analogie claire entre la dynamique d'une trace hybride et une dynamique de PDM dans laquelle :

— une décision est prise pour chaque état discret par lequel la trace hybride doit passer (ces derniers sont spécifiés par la connaissance biologique). Entre deux décisions de l'agent, le temps « réel » s'écoule différemment. Dans le cas du cycle négatif (équation (2.3)), cinq heures s'écoulent après la première décision ($t = 0$), sept heures après la seconde ($t = 1$), huit heures entre la troisième et la quatrième ($t = 2$), puis quatre heures après la dernière ($t = H = 3$) ;

— l'état hybride d'entrée d'un état discret correspond à un état du PDM ;

— le choix d'un vecteur célérité correspond à une action multidimensionnelle du PDM ;

— une récompense immédiate correspond à l'évaluation de cette action dans l'état discret actuel.

Pour affiner cette intuition, une analogie détaillée entre la dynamique d'un RRGH et la dynamique d'un PDM est fournie dans le tableau 4.1 et est illustrée dans la figure 4.5 où les états du PDM représentent les états d'entrées de la trace dans chaque état discret ($S = \{s_0, s_1, s_2\} = \{h_0, h_1, h_2\}$) et les actions correspondent aux vecteurs célérités dans chaque état discret ($A = \{a_0, a_1\} = \{\left(C_{v_1,\rho(h_0,v_1),0}, C_{v_2,\rho(h_0,v_2),0}\right), \left(C_{v_1,\rho(h_1,v_1),1}, C_{v_2,\rho(h_1,v_2),0}\right)\}$).

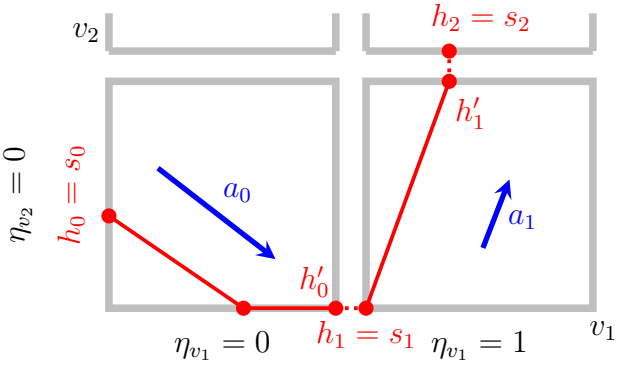


FIGURE 4.5 – Illustration de l'analogie entre la dynamique d'une trace hybride et du PDM.

t	État/Action	Dynamique de la trace hybride	Dynamique du PDM
0	État Action	h_0 $C_{v,\rho(h_0,v),\eta_v}$	s_0 $\pi(s_0) = a_0$
1	État Action	$h_1 = f_{simulateur}(h_0, C_{v,\rho(h_1,v),\eta_v})$ $C_{v,\rho(h_1,v),\eta_v}$	$s_1 = T(s_0, a_0)$ $\pi(s_1) = a_1$
$t < H$	...	...	...
H	État	h_f, $h_{blocked}$ ou h_{error}	s_H

Tableau 4.1 – Analogie entre la dynamique d'une trace hybride et la dynamique du PDM.

Pour définir la fonction de récompense, nous utilisons les éléments de la fonction objectif définie dans la section 3.1.2. $tr(x)$ faisait alors référence à la trace hybride complète simulée à partir d'un vecteur de décision. Ici, $tr(s_t, a_t, s_{t+1})$ est un sous-ensemble de $tr(x)$ et définit une partie de la trace, plus précisément, la trajectoire continue suivie de la trajectoire discrète partant de s_t et arrivant à s_{t+1} en suivant le vecteur célérité de l'action a_t. Par exemple sur la figure 4.5, $tr(s_0, a_0, s_1)$ correspond à la trajectoire continue $h_0 \to h'_0$ et à la trajectoire discrète $h'_0 \to h_1$. La fonction de récompense utilise donc la distance entre la connaissance biologique de l'état courant et la partie de la trace de l'état courant composée de la trajectoire continue et trajectoire discrète. Elle correspond à $d(tr(s_t, a_t, s_{t+1}), CB) = d_\eta(tr(x), CB) = d_{\Delta t,\eta}(tr(x), CB) + d_{a,\eta}(tr(x), CB) + d_{dpa,\eta}(tr(x), CB)$.

Comme il est usuel de chercher à maximiser le gain, la fonction de récompense est définie comme l'opposé de cette distance. Une récompense égale ou très proche de 0 traduit le fait que la trajectoire respecte la connaissance biologique. À l'inverse, une récompense négative signifie que cette partie de la trace est très éloignée de la connaissance biologique. Pour définir le gain, nous choisissons la somme des récompenses : $\sum_{k=t+1}^{H} r_k$. Ainsi, si un gain est optimal, il a une valeur proche de ou égale à 0, sinon sa valeur est comprise dans l'intervalle ouvert $]-\infty, 0[$.

Nous définissons plus formellement le PDM du problème d'identification des paramètres dynamiques dans un RRGH.

PDM du problème d'identification des paramètres dynamiques dans un RRGH

Définition 19. Le PDM du problème d'identification des paramètres dynamiques est un tuple $\langle S, A, T, R \rangle$ où :

- S est l'ensemble des états hybrides d'entrée de chaque état discret avec $s_t = h_t$ l'état hybride d'entrée du t-ème état discret (le t-ème tuple du triplet de Hoare hybride de longueur $H + 1$),
- A est l'ensemble des vecteurs célérités de chaque état discret avec a_t la t-ème action choisie par l'agent correspondant au vecteur célérité du t-ème état discret,
- $T : S \times A \to S$ est la fonction de transition fournie par le simulateur (section 1.4) telle que $s_{t+1} = T(s_t, a_t)$,
- $R : S \times A \times S \to \mathbb{R}$ est la fonction de récompense définie par :

$$\begin{aligned} R : S \times A \times S &\to \mathbb{R}^+ \\ : s_t \times a_t \times s_{t+1} &\mapsto -d(tr(s_t, a_t, s_{t+1}), CB) \end{aligned} \tag{4.1}$$

Le PDM de la définition 19 est un PDM dans lequel l'environnement, ici le simulateur (section 1.4), est déterministe. La fonction de transition est donc déterministe. Le PDM correspond à une tâche épisodique, car il existe des états terminaux. Il existe deux types d'états terminaux. Les premiers forment l'ensemble des états hybrides finaux où $s_H = h_f$ (la trace revient à l'état hybride initial formant un cycle). Dans ce cas, l'horizon est égal au nombre de triplets dans la connaissance biologique, *i.e.*, la longueur maximale de la trace hybride pour qu'elle puisse former un cycle. Dans le cas du cycle négatif : $H = 3$, du cycle circadien : $H = 5$, et du cycle cellulaire : $H = 11$. Le second type d'état terminal

correspond à un état hybride final de la trace bloquée $s_H = h_{blocked}$ ou $s_H = h_{error}$ (*cf.* section 3.1.2.2).

4.1.3 Fonctions valeur et équations de Bellman

La plupart des algorithmes d'apprentissage pour les PDMs calculent les politiques optimales en apprenant une fonction valeur (*value function*) qui estime l'intérêt (ou l'utilité) pour l'agent d'être dans un état particulier. Nous présentons cette notion ci-dessous.

On distingue deux types de fonction valeur : la fonction valeur pour un état et la fonction valeur pour une paire état-action. La fonction valeur d'un état s ($V : S \to \mathbb{R}$) suivant la politique π est notée $V_\pi(s)$. Elle est définie comme étant l'espérance du gain g_t, en partant de l'état s puis en suivant la politique π :

$$V_\pi(s) = \mathbb{E}_\pi[g_t \mid s_t = s],\ \forall s \in S \tag{4.2}$$

Notons que la valeur d'un état terminal est toujours de 0.

De manière analogue, la valeur d'une action a choisie à partir d'un état s et suivant la politique π est notée $Q_\pi(s, a)$. La fonction valeur de la paire état-action (s, a), notée $Q : S \times A \to \mathbb{R}$, est définie comme l'espérance d'obtenir le gain g_t, partant de l'état s, choisissant l'action a et suivant la politique π ensuite :

$$Q_\pi(s, a) = \mathbb{E}_\pi[g_t \mid s_t = s, a_t = a],\ \forall s \in S, a \in A(s) \tag{4.3}$$

Les fonctions valeur V_π et Q_π peuvent être estimées à partir de l'expérience de l'agent. Par exemple, si un agent suit une politique π et maintient une statistique moyenne, pour chaque état rencontré s, du gain obtenu en visitant cet état s, alors la moyenne empirique convergera vers la valeur théorique $V_\pi(s)$ à mesure que le nombre de visites de l'état approche l'infini. Comme nous le verrons dans la prochaine section, ces méthodes d'estimation sont des méthodes de Monte Carlo.

Une propriété fondamentale des fonctions de valeur est qu'elles satisfont certaines propriétés récursives. Pour toute politique π et tout état s, l'expression de l'équation (4.2) peut être définie récursivement en termes d'équation dite de Bellman (Bellman 1957) :

$$\begin{aligned} V_\pi(s) &= \mathbb{E}_\pi[g_t \mid s_t = s] \\ &= \mathbb{E}_\pi[r_{t+1} + r_{t+2} + ... + r_T \mid s_t = s] \\ &= \mathbb{E}_\pi[r_{t+1} + V_\pi(s_{t+1}) \mid s_t = s] \end{aligned} \tag{4.4}$$

L'équation (4.4) indique que la valeur attendue d'un état est définie en termes de (i) la récompense immédiate et (ii) des valeurs des états suivants possibles, pondérées par leurs probabilités de transition.

L'objectif d'un PDM donné est de trouver la meilleure politique, c'est-à-dire la politique qui reçoit le plus grand gain. Cela signifie maximiser la fonction valeur (équation (4.2)) pour tous les états $s \in S$. Une politique optimale, notée π^*, est telle que $V^*_\pi(s) \geq V_\pi(s)$ pour tous les états et toutes les politiques π. On a l'équation suivante :

$$V_*(s) = \max_\pi v_\pi(s),\ \forall s \in S \tag{4.5}$$

De manière similaire, on a :

$$Q_*(s,a) = \max_{\pi} Q_\pi(s,a),\ \forall s \in S, a \in A(s) \tag{4.6}$$

Dans la prochaine section, nous présentons la recherche arborescente Monte Carlo (MCTS) que nous utilisons pour résoudre le PDM décrivant le problème d'identification des paramètres dynamiques dans les RRGHs.

4.2 État de l'art recherche arborescente Monte Carlo

Le développement de MCTS remonte à (Abramson 1990) démontrant que les simulations de Monte Carlo peuvent être utilisées pour évaluer la valeur d'un état, et à (Brügmann 1993) appliquant des méthodes Monte Carlo au jeu de Go. Nous faisons le choix de ne pas retracer l'historique de MCTS, mais plutôt de le présenter à l'aide des termes et des notations introduits dans (Silver 2009) et (Gelly et Silver 2011). Cet état de l'art s'inspire des travaux de (Browne et al. 2012) et plus récemment de (Świechowski et al. 2023), mais aussi de (Silver 2009), (Couetoux 2013) et (Sutton et Barto 2018).

MCTS est décrit comme une méthode de recherche heuristique utilisée dans le cadre de la prise de décision. L'idée principale est de réaliser des simulations aléatoires à partir de l'état actuel s. Les nouveaux états sont ajoutés à un arbre de recherche et chaque nœud de l'arbre contient la valeur d'une paire état-action $Q(s,a)$ permettant de prédire l'utilité de choisir l'action a dans l'état s. Ces prédictions sont mises à jour à l'aide des simulations de Monte Carlo : la valeur est simplement le résultat moyen de toutes les parties simulées qui visitent la position s et choisissent l'action a ensuite.

4.2.1 Recherche basée sur les simulations

MCTS est un **algorithme de recherche** basé sur la construction d'un **arbre de recherche**. MCTS utilise des simulations pour construire l'arbre de recherche, évaluer les états et les actions de l'arbre et orienter la recherche : il s'agit d'un **algorithme de recherche par simulation**. Plus précisément, MCTS utilise la **simulation de Monte Carlo** pour approximer la valeur théorique d'une action étant donné un état. Nous décrivons pas à pas les différents éléments en gras.

Arbre de recherche. Un arbre est une structure de graphe hiérarchique qui est utilisée pour modéliser et analyser les problèmes de décision séquentiels. On peut penser à l'arbre de jeu en théorie des jeux. Dans un arbre, chaque nœud représente un état du problème et une arête représente une transition entre deux états successifs. Un arbre est complet lorsqu'il comprend tous les états possibles accessibles à partir de l'état initial (placé à la racine de l'arbre). Avec un arbre complet, l'espace de recherche d'un problème de décision est représenté dans son ensemble. En pratique, le facteur de branchement, ou nombre de successeurs d'un nœud, est souvent beaucoup trop grand pour construire un tel arbre. Un arbre partiel ou **arbre de recherche**, noté $\mathcal{T}$, est utilisé à la place. Il s'agit d'un sous-arbre de l'arbre complet qui est construit au fur et à mesure de l'exécution d'un algorithme de recherche.

Algorithme de recherche. Les états de l'arbre de recherche peuvent être développés de manière sélective, en développant les nœuds feuilles selon un certain critère, ou de manière exhaustive, en développant tous les nœuds jusqu'à une certaine profondeur fixe. Ainsi, l'expansion peut se faire en profondeur (*depth-first*), en largeur (*breadth-first*) ou par le meilleur (*best-first*). Ce dernier cas est possible lorsque la stratégie d'exploration est informée. Elle utilise une fonction heuristique, par exemple, pour guider la recherche vers les états les plus prometteurs. Une présentation détaillée de ces différentes stratégies de résolution de problèmes peut être trouvée dans (Russell et Norvig 2016) comprenant des exemples d'algorithmes de recherche bien connus comme A^*, ou encore alpha-beta.

Algorithme de recherche par simulation. L'idée de la recherche par simulation, *simulation-based search* (Silver 2009), est d'évaluer les états à partir de simulations. Une simulation débute à la racine de l'arbre s_0 et échantillonne séquentiellement les états et actions jusqu'à atteindre un état terminal. À chaque pas de temps t, la politique de simulation π est utilisée pour sélectionner une action et la fonction de transition génère le nouvel état. Le résultat de la simulation est utilisé pour mettre à jour les valeurs des états ou des actions rencontrées lors de la simulation.

La fonction d'évaluation de la recherche par simulation se distingue des algorithmes de recherche traditionnels en ne dépendant que des résultats observés à la fin des simulations. La récompense n'est obtenue que lorsque la simulation termine, c'est-à-dire qu'un état terminal est atteint.

Simulation de Monte Carlo. La simulation de Monte Carlo est un algorithme de recherche par simulation qui utilise la méthode de Monte Carlo pour évaluer les actions légales à partir d'un état racine s_0.

La simulation de Monte Carlo permet d'approcher la valeur d'un état. La fonction valeur de chaque état s est estimée par la moyenne des résultats obtenus de toutes les simulations dans lesquelles s a été visité :

$$V(s) = \frac{1}{n(s)} \sum_{i=1}^{n(s)} z_i \tag{4.7}$$

avec z_i le résultat ou le gain (récompense cumulée) obtenu lors de la i-ème simulation contenant l'état s et $n(s)$ le nombre total d'épisodes contenant s.

De manière analogue, l'évaluation de Monte Carlo permet d'approximer la valeur d'une paire état-action. La fonction valeur de chaque paire état-action est estimée par la moyenne des résultats de toutes les simulations dans lesquelles a a été choisie à partir de l'état visité s :

$$Q(s,a) = \frac{1}{n(s,a)} \sum_{i=1}^{n(s)} \mathbb{1}_i(s,a) \times z_i \tag{4.8}$$

avec z_i le résultat ou la récompense cumulée obtenue lors de la i-ème simulation contenant l'état s, $\mathbb{1}_i(s,a)$ la fonction indicatrice ayant la valeur 1 si l'action a a été choisie à partir de l'état, et $n(s,a) = \sum_{i=1}^{n(s)} \mathbb{1}_i(s,a)$ est le compteur du nombre total de simulation dans lesquelles l'action a a été sélectionné dans l'état s.

La recherche se déroule en effectuant des simulations de s_0 jusqu'à un état terminal s_H, en utilisant une politique de simulation aléatoire fixée, par exemple en sélectionnant

les actions uniformément parmi toutes les actions légales. Les travaux de (Abramson 1990) mettent en évidence que la simulation Monte Carlo permet bien d'estimer la valeur théorique d'une action $Q^*(s,a)$ en l'appliquant sur des variantes des jeux du morpion, d'Othello, et des échecs.

Dans notre cas d'étude, les espaces des actions et des états sont continus. Avant de nous intéresser à ce cadre continu, nous présentons en premier lieu MCTS dans un cadre combinatoire.

4.2.2 Recherche arborescente Monte Carlo : cadre combinatoire

Comme nous venons de l'introduire, MCTS est un algorithme de recherche basé sur la simulation de Monte Carlo pour évaluer les nœuds d'un arbre de recherche, noté $\mathcal{T}$, construit itérativement (Coulom 2007b).

Cet arbre de recherche est un sous-ensemble de toutes les paires état-action possibles $\mathcal{T} \subseteq S \times A$ dans lequel un nœud représente un état $s \in S$ et une arrête entre deux nœuds représente une action $a \in C_{\text{action}}(s) \subseteq A(s)$, où $A(s)$ est l'ensemble des actions possibles à partir de s et $C_{\text{action}}(s) = \{a \,|\, (s,a) \in \mathcal{T}\}$ est l'ensemble des actions présentes dans l'arbre à partir de s et est un sous-ensemble $A(s)$. Une arrête relie un état parent s à un état enfant $s' \in C_{\text{etat}}(s)$, où $C_{\text{etat}}(s) = \{s \,|\, (s,a) \in \mathcal{T}\}$ représente l'ensemble des états enfants de s. Chaque état est généré grâce à la fonction de transition T. Pour chaque paire (s,a) dans l'arbre, les statistiques maintenues *a minima* sont : un compteur total du nombre de fois que l'état s a été visité ($n(s)$) et la valeur $Q(s,a)$ estimée par la moyenne des résultats des simulations de Monte Carlo dans lesquelles a a été choisie à partir de s.

Nous décrivons ici certains *variants* de la famille des algorithmes de MCTS. Chacun de ces variants se repose sur deux hypothèses fondamentales :

— étant donné un état $s \in S$, la valeur théorique d'une action $a \in A$, $Q^*(s,a)$, peut être approximée en utilisant des simulations aléatoires, où une simulation correspond à une séquence aléatoire d'actions appliquées à un état initial et atteignant un état terminal et générant un résultat,

— la politique de l'agent est ajustée en fonction des statistiques des simulations en appliquant la recherche *best-first*, favorisant les régions de l'espace de recherche les plus prometteuses.

À mesure que des simulations sont réalisées, l'arbre de recherche est construit à partir des actions choisies et des états visités et la recherche dans l'arbre est guidée par les statistiques obtenues à la fin des simulations. Plus leur nombre augmente, plus la moyenne empirique $Q(s,a)$ devient une bonne estimation de $Q^*(s,a)$. En résumé, MCTS se concentre sur la recherche de l'action la plus prometteuse, en développant un arbre de recherche à partir d'un échantillonnage aléatoire de l'espace de recherche.

Nous détaillons dans un premier temps la version de base de MCTS où il n'y a qu'un seul agent dans l'environnement.

4.2.2.1 Description de l'algorithme de base

MCTS est un algorithme itératif *anytime*, c'est-à-dire qu'il s'agit d'un algorithme capable de rendre une solution à n'importe quelle itération : des solutions sont trouvées au fur et à mesure de son exécution. L'avantage est de pouvoir produire un résultat à tout

moment, mais ce résultat est une approximation, dont la qualité dépend de la quantité de calculs effectués (produire un résultat de meilleure qualité, en échange de temps de calcul supplémentaire). Un budget computationnel est prédéfini : cela peut être le nombre d'itérations, un temps d'exécution ou encore une capacité de mémoire.

Étant donné ce budget, les différentes étapes de MCTS sont illustrées sur la figure 4.6. L'étape (a) correspond à l'initialisation de l'arbre de recherche où la racine de l'arbre est définie comme l'état initial du PDM. Ensuite, chaque itération de MCTS permet de construire l'arbre de recherche. Une itération se décompose en quatre étapes : (b) la sélection, (c) l'expansion, (d) le *rollout* et (e) la rétropropagation. Ces différentes étapes sont décrites ci-dessous :

— (b) à partir du nœud racine s_0, une fonction de sélection est appliquée récursivement pour descendre dans l'arbre de recherche jusqu'à ce que le nœud feuille le plus prometteur (on parle aussi de nœud le plus urgent) pouvant être développé soit atteint. Un nœud feuille peut être développé s'il représente un état non terminal et possède au moins un enfant qui n'a pas encore été développé ;

— (c) à partir du nœud sélectionné dans l'étape (b), un ou plusieurs nœuds enfants sont ajoutés en choisissant une ou plusieurs actions qui n'ont pas été développées ($\tilde{A}(s) = \{a \,|\, (s,a) \notin \mathcal{T}, a \in A(s)\}$) et en appliquant la fonction de transition T ;

— (d) un *rollout* est exécuté à partir du ou des nouveaux nœuds développés de l'étape (c), ce *rollout* permet d'atteindre un état terminal et de recevoir un résultat (une récompense cumulée, par exemple). Par défaut, une action est choisie aléatoirement (parmi l'ensemble des actions légales) donnant lieu à un nouvel état, puis une action est choisie aléatoirement à partir de ce nouvel état donnant lieu à un nouvel état et ce processus est itéré jusqu'à ce qu'un état terminal soit atteint ;

— (e) le résultat z est rétropropagé à travers le nœud développé (étape (c)) et les nœuds sélectionnés de l'étape (b) pour mettre à jour leurs statistiques. En fonction de la politique de sélection, ces statistiques peuvent varier, mais les informations minimales enregistrées sont celles qui composent la moyenne empirique : $Q(s,a)$ et $n(s)$.

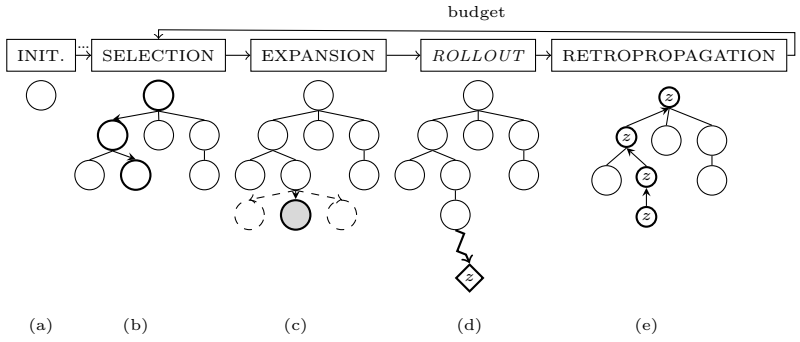


FIGURE 4.6 – Illustration des différentes étapes de la recherche arborescente Monte Carlo : (a) initialisation, (b) sélection, (c) expansion et (d) rétropropagation, réalisée par Denis Pallez.

Afin d'éviter toute ambiguïté, nous avons fait le choix de nommer l'étape (d) « *rollout* » alors que la notion de « simulation » est souvent utilisée. En effet, le terme « simulation »

utilisé dans les algorithmes basés sur les simulations fait référence à une séquence d'actions appliquées à un état initial atteignant ainsi un état terminal. Or, l'étape (d) correspond à une séquence d'actions appliquées à un nœud feuille et non terminal de l'arbre de recherche qui permet d'atteindre un état terminal. Il faut comprendre le terme *rollout* comme « dérouler la simulation jusqu'à son terme » (« *rollout a simulation to completion* »). La simulation est divisée en deux étapes qui forment deux politiques distinctes utilisées pour construire l'arbre de recherche :

— la politique de l'arbre (*tree policy*), combine la fonction de sélection (étape (b)) et l'étape d'expansion (étape (c)) : elle permet de sélectionner les actions pour les états présents dans l'arbre de recherche et d'étendre un nouveau nœud, et

— la politique par défaut (*default policy*), notée $\pi_{default}$, intervient dans l'étape (d). Cette politique est généralement une politique purement aléatoire, c'est-à-dire que pour un état non terminal, la probabilité de choisir entre les différentes actions possibles est uniforme.

MCTS est utilisé pour la prise de décision en ligne et le résultat de la recherche est généralement l'action unique qui mène au « meilleur » enfant du nœud racine s_0. Ensuite, MCTS génère une nouvelle recherche à partir de ce nouvel état (nouveau nœud racine). Cependant, MCTS peut également être utilisé pour retourner un arbre de recherche. De cet arbre est extraite une séquence d'action de l'arbre de recherche (chemin allant de la racine de l'arbre à un nœud feuille). C'est ce deuxième cas qui est envisagé ici.

Les différentes étapes de la construction de l'arbre de recherche sont décrites dans le pseudo-code de l'algorithme 1, où s_0 correspond à l'état initial du PDM, *budget* est un budget prédéfini (temps, mémoire ou nombre d'itérations) et renvoie un arbre de recherche noté $\mathcal{T}$. L'algorithme comprend quatre fonctions faisant référence aux quatre étapes de MCTS illustrées dans la figure 4.6 :

— `Selection` est une fonction qui prend en entrée la racine de l'arbre MCTS (s_0) et renvoie un nœud qui est soit un nœud feuille, soit un nœud pour lequel toutes les actions n'ont pas été étendues, s_{select} ;

— `Expansion` est une fonction qui prend en entrée le nœud sélectionné et choisit une action qui n'a jamais été testée, puis renvoie l'état résultant de la fonction de transition. Cet état est un nœud feuille, noté ici s_{feuille} ;

— `Rollout` est une fonction qui prend en entrée le nouveau nœud développé, applique la *default policy* et renvoie le résultat obtenu z ;

— `Retropropagation` est une fonction qui prend en entrée le résultat de `Rollout` et le nœud nouvellement étendu s_{feuille} et rétropropage le résultat le long du chemin à la racine formé des nœuds sélectionnés dans `Selection` jusqu'à s_0.

Cette version de base de MCTS est appelée *greedy* MCTS dans laquelle la *tree policy* fait le choix de l'action avec la plus haute valeur ($\text{argmax}_{a \in A(s)} Q(s,a)$) et choisit les actions aléatoirement de manière uniforme dans la *default policy*. MCTS peut être amélioré en modifiant la politique de l'arbre ou en améliorant la politique par défaut au cours de l'étape de *rollout*.

Nous présentons l'*Upper Confidence Bounds for Trees* (UCT) qui utilise une autre fonction de sélection pour améliorer les performances de cette version de base.

Algorithm 1 Pseudo-code de la recherche arborescente Monte Carlo.

Input: un état initial du PDM : s_0, un budget computationnel : *budget*
Output: un arbre de recherche : $\mathcal{T}$

1: Création d'un nœud racine avec l'état s_0 ▷ Étape (a)
2: **while** *budget* n'est pas atteint **do**
3: $s_{\text{select}} \leftarrow \texttt{Selection}(s_0)$ ▷ Étape (b)
4: $s_{\text{feuille}} \leftarrow \texttt{Expansion}(s_{\text{selection}})$ ▷ Étape (c)
5: $z \leftarrow \texttt{Rollout}(s_{\text{feuille}})$ ▷ Étape (d)
6: $\texttt{Retropropagation}(z, s_{\text{feuille}})$ ▷ Étape (e)
7: **end while**

4.2.2.2 *Upper Confidence Bounds for Trees*

Le variant le plus populaire de la famille MCTS est l'algorithme UCT (Kocsis et Szepesvári 2006). Il tire son nom de l'utilisation de la formule de sélection UCB1 (Auer, Cesa-Bianchi et Fischer 2002) dans la *tree policy*. Le choix de l'action menant au nœud enfant le plus prometteur est alors considéré comme un problème de bandit manchot dans lequel la valeur d'un nœud enfant est la récompense espérée approximée par les simulations Monte Carlo et les récompenses sont les variables aléatoires de distribution inconnues. UCB1 adresse le dilemme d'exploration *versus* exploitation en suivant le principe d'optimisme face à l'incertitude (*optimism in the face of uncertainty*). Ce principe favorise l'action la plus prometteuse, *i.e.*, l'action dont la valeur de la moyenne empirique (exploitation) est haute et qui n'a pas encore été beaucoup sélectionnée (exploration). Ainsi, pendant l'étape de sélection, à partir d'un état s, l'action $a \in A(s)$ est choisie en appliquant la formule d'UCB1 :

$$Q^{\oplus}_{\text{UCT}}(s,a) = \begin{cases} \overbrace{Q_{\text{UCT}}(s,a)}^{\text{terme d'exploitation}} + \overbrace{c \times \sqrt{\frac{log(n(s))}{n(s,a)}}}^{\text{terme d'exploration}} & \text{, si } n(s,a) > 0 \\ +\infty & \text{, sinon.} \end{cases} \tag{4.9}$$

$$\pi_{\text{UCT}}(s) = \underset{a \in A(s)}{\text{argmax}}\, Q^{\oplus}_{\text{UCT}}(s,a) \tag{4.10}$$

avec

- $n(s,a)$ est le nombre de fois que a a été choisie après que s ait été visité ;
- $n(s) = \sum_{a \in A(s)} n(s,a)$ est le nombre de fois que l'état s est sélectionné (compteur de visite de s) ;
- $Q_{\text{UCT}}(s,a) = \frac{\sum_{i=1}^{n(s)} \mathbb{1}_i(s,a) \times z_i}{n(s,a)}$ est la moyenne empirique des récompenses obtenues lorsque l'action a est choisie à partir du nœud s (équation (4.8)) ; et
- c est une constante positive, appelée facteur d'exploration, car si sa valeur est importante, une action ayant été peu visitée obtiendra un score important (et réciproquement si sa valeur est faible).

Lorsqu'une action est choisie à partir de l'état s, le dénominateur du terme d'exploration augmente, diminuant ainsi sa contribution. À l'inverse, si l'action n'est pas choisie (au

profit d'une action sœur), le numérateur augmente, mais pas le dénominateur : sa contribution augmente ainsi que celle des autres actions sœurs n'ayant pas été choisies. Lorsque deux actions ont la même valeur, le choix est réalisé aléatoirement de manière uniforme. Pour chaque action a qui n'a pas encore été choisie $\tilde{A}(s)$, il est usuel d'initialiser la valeur de $Q^{\oplus}(s,a)$ à $+\infty$.[3]

Les quatre fonctions du pseudo-code de l'algorithme 1 sont développées dans l'algorithme 2, l'algorithme 3, l'algorithme 4 et l'algorithme 5. Le pseudo-code d'UCT repose sur quelques hypothèses :

— l'ensemble des états et celui des actions sont finis, auquel cas l'arbre de recherche s'élargira uniquement en largeur et jamais en profondeur : la profondeur maximale de l'arbre sera de 1 ;

— le PDM décrit une tâche épisodique à horizon fini au bout duquel un résultat est calculé : l'étape de *rollout* termine et peut être évaluée par une fonction `evaluer` : $S \to \mathbb{R}$ qui prend en entrée un état terminal et renvoie une valeur réelle $z \in \mathbb{R}$;

— l'algorithme dispose d'une fonction d'échantillonnage *sampler* qui, étant donné un ensemble d'actions retourne une action : $a \leftarrow sampler(A(s))$ signifie qu'une action a est échantillonnée aléatoirement parmi un ensemble d'actions ($A(s)$ ici) ;

— la fonction $P : S \to S \times A$ existe et prend en entrée un état s' et renvoie son état parent s et l'action a qui l'ont mené à lui (en appliquant $s' \leftarrow T(s,a)$).

Algorithm 2 Pseudo-code de la fonction `Selection` de UCT.

Input: un état initial du PDM : s_0, une constante d'exploration : c
Output: le nœud sélectionné de $\mathcal{T}$: $s_{\text{selection}}$

1: $s \leftarrow s_0$
2: **while** s est non terminal ou $n(s) == 0$ **do**
3: $\quad a \leftarrow \pi_{\text{UCT}}(s)$ ▷ Équation (4.10)
4: $\quad s \leftarrow T(s,a)$
5: **end while**
6: $s_{selection} \leftarrow s$
7: **return** $s_{selection}$

L'un des avantages les plus significatifs de MCTS est qu'il ne nécessite pas de connaissances spécifiques au domaine, ce qui le rend facilement applicable à tout domaine pouvant être modélisé comme un PDM à l'aide d'un arbre. Cependant, appliqué tel quel, il obtient rarement les meilleures performances. Pour décrire ce phénomène, le terme *aheuristic* est utilisé dans (Browne et al. 2012). Des modifications heuristiques ont été apportées pour améliorer son efficacité *i.e.*, sa capacité à trouver une solution optimale avec un *minimum* de budget. Nous présentons certaines d'entre elles qui visent à améliorer la *tree policy*.

3. Dans le cas d'un facteur de branchement trop important, initialiser $Q^{\oplus}$ pose problème. En effet, comme chaque nœud non visité devra d'abord être testé, l'exploitation en profondeur dans l'arbre de recherche n'est pas possible ou trop lente. La méthode *First Play Urgency* (Gelly et Y. Wang 2006) propose d'assigner un score (à ajuster) pour les nœuds jamais visités et d'utiliser UCB1 pour les nœuds visités. Cela permet ainsi d'encourager l'exploitation malgré le facteur de branchement important.

Algorithm 3 Pseudo-code de la fonction `Expansion` de UCT.

Input: le nœud sélectionné renvoyé par l'algorithme 2 : $s_{\text{selection}}$
Output: le nœud développé de $\mathcal{T}$: s_{feuille}

if $s_{\text{selection}}$ est non terminal **then**
  $a \leftarrow sampler(\tilde{A}(s_{\text{selection}}))$ ▷ $\tilde{A}(s_{\text{selection}}) = \{a \mid (s_{\text{selection}}, a) \notin \mathcal{T}, a \in A(s_{\text{selection}})\}$
  $C_{\text{action}}(s_{\text{selection}}) \leftarrow C_{\text{action}}(s_{\text{selection}}) \cup \{a\}$
  $s_{\text{feuille}} \leftarrow T(s_{\text{selection}}, a)$
end if
return s_{feuille}

Algorithm 4 Pseudo-code de l'étape de `Rollout` de UCT.

Input: le nœud développé renvoyé par l'algorithme 3 : $s_{feuille}$
Output: un résultat, gain ou récompense cumulée : z

$s, g \leftarrow s_{feuille}, 0$
while s est non terminal **do**
  $a \leftarrow \pi_{default}(s)$
  $s \leftarrow T(s, a)$
end while
$z \leftarrow$ `evaluer`(s)
return z

Algorithm 5 Pseudo-code de la fonction `Retropropagation` de UCT.

Input: le nœud développé de algorithme 3 : $s_{feuille}$, un résultat : z

$s \leftarrow s_{feuille}$
while s n'est pas nul **do**
  $s, a \leftarrow P(s)$
  $n(s) \leftarrow n(s) + 1$
  $n(s, a) \leftarrow n(s, a) + 1$
  $Q_{\text{UCT}}(s, a) \leftarrow Q_{\text{UCT}}(s, a) + \frac{1}{n(s,a)}[z - Q_{\text{UCT}}(s, a)]$
end while

4.2.3 Améliorations

Le but de ces améliorations heuristiques est de permettre d'améliorer l'efficacité de MCTS sur un ensemble de problèmes donnés. Nous en présentons trois.

4.2.3.1 *All Moves As First*

Initialement proposée dans le contexte du Go, *All moves as first* (AMAF) (Brügmann 1993) est une amélioration heuristique visant à modifier l'étape de rétropropagation. L'idée est de considérer les actions choisies par la *default policy* comme si elles avaient été sélectionnées par la *tree policy*. En plus de mettre à jour les nœuds sélectionnés par la politique de l'arbre (comme pour UCT), AMAF considère aussi les nœuds qui résultent d'une des actions choisies pendant l'étape de *rollout*. Il est usuel de considérer deux types de statistiques par nœud en ajoutant celles d'AMAF, distinctes de celles d'UCT.

La figure 4.7 illustre l'heuristique AMAF sur l'exemple de la grille 4×3. Dans cette

situation, à partir de l'état initial A1, UCT sélectionne les actions *Droite* et *Droite* menant d'abord à l'état A2, puis A3 (les nœuds bleus). À partir de A3, un *rollout* choisit les actions *Haut*, *Droite* puis *Haut* menant à l'état « Arrivée » en C4. L'idée de l'heuristique AMAF est de considérer qu'UCT aurait pu faire un autre choix que celui qui a été fait en choisissant des actions sélectionnées lors de l'étape de *rollout*. Dans notre cas, cela aurait pu être *Haut* d'abord et *Droite* ensuite, par exemple. Le score AMAF du nœud de l'arbre qui résulte du choix de l'action *Haut* à partir du nœud racine A1, *i.e.* le nœud B1, est mis à jour lors de l'étape de rétropropagation. Ce processus est itéré pour chacun des nœuds choisis par UCT : dans l'exemple, c'est donc aussi le cas du nœud B2 à partir de A2. Ainsi, les nœuds dont le score UCT est mis à jour sont en bleu, tandis que les nœuds dont le score AMAF est mis à jour sont en rouge.

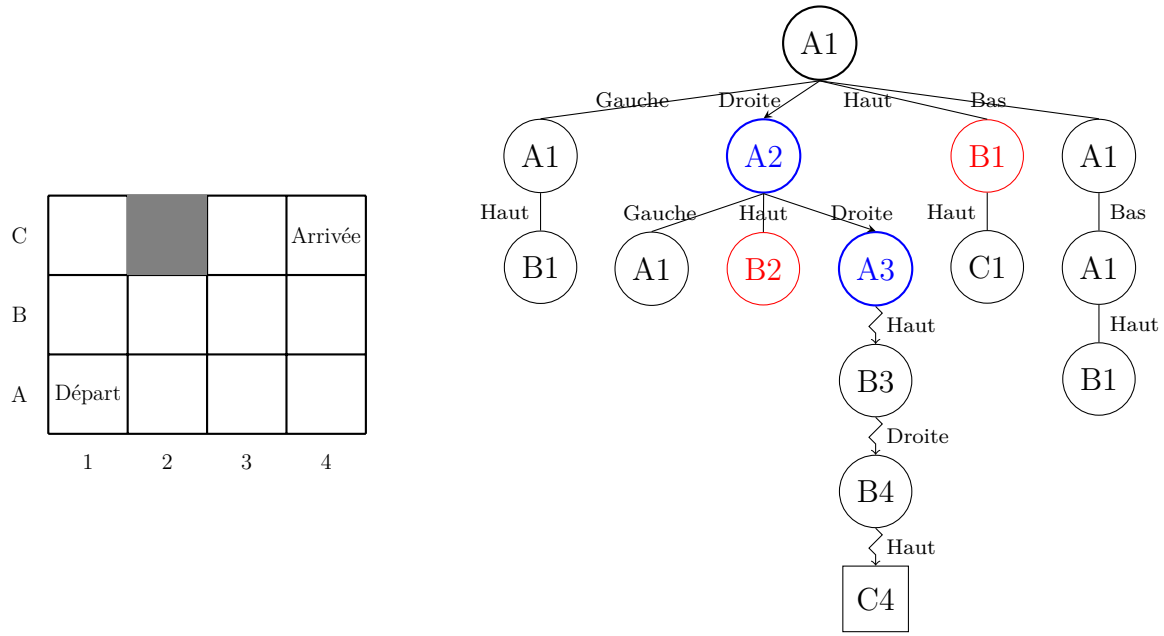


FIGURE 4.7 – Illustration de l'heuristique AMAF sur l'exemple de la figure 4.1, inspirée de (HELMBOLD et PARKER-WOOD 2009).

Dans cet exemple simpliste de la grille 4×3, l'intuition derrière la stratégie d'AMAF est assez claire : les actions *Droite* et *Haut* permettent de se diriger rapidement vers l'état terminal, alors que *Gauche* et *Bas* ne font que ralentir l'agent. Ainsi, AMAF renforce cette stratégie en favorisant les actions *Droite* et *Haut* dès le début. Lorsque le nombre d'itérations augmente, on préfère l'estimation non biaisée du score UCT pour, par exemple, éviter l'obstacle.

Plus formellement, d'après les notations de (GELLY et SILVER 2011), la fonction valeur AMAF d'une paire état-action notée Q_{AMAF} est le résultat z espéré à partir de l'état s et en suivant la politique π, étant donné que a ait été sélectionnée à un moment dans la simulation :

$$Q_{\text{AMAF}}(s,a) = \mathbb{E}_\pi[z \mid s_t = s, \exists k \geq t \ \text{ t.q. } \ a_k = a] \tag{4.11}$$

Ainsi, la fonction valeur AMAF est une estimation biaisée de la fonction valeur $Q(s,a)$. Le biais $B(s,a)$ dépend de l'état s et l'action a :

$$Q_{\text{AMAF}}(s,a) = Q(s,a) + B(s,a) \tag{4.12}$$

La valeur $Q_{\text{AMAF}}(s,a)$ est estimée à l'aide des simulations de Monte Carlo. Il s'agit de la moyenne des résultats de toutes les simulations dans lesquelles l'action a est sélectionnée après que s soit rencontré :

$$Q_{\text{AMAF}}(s,a) = \frac{1}{m(s,a)} \sum_{i=1}^{n(s)} \tilde{\mathbb{1}}_i(s,a) z_i \tag{4.13}$$

où $\tilde{\mathbb{1}}_i(s,a)$ est la fonction indicatrice qui retourne 1 si l'état s est rencontré lors de la i-ème simulation et que l'action a est sélectionnée plus tard, 0 sinon ; et $m(s,a)$ correspond au nombre total de simulations utilisées pour estimer la valeur AMAF : $m(s,a) = \sum_{i=1}^{n(s)} \tilde{\mathbb{1}}_i(s,a)$.

L'heuristique AMAF combinée à UCT donne lieu à un nouveau variant de MCTS : *Rapid Action Value Estimate* (RAVE).

4.2.3.2 *Rapid Action Value Estimate*

RAVE (Gelly et Silver 2007 ; Gelly et Silver 2011) est une modification de la fonction de sélection basée sur l'heuristique AMAF : elle combine le score UCT et le score AMAF. Comme énoncé dans (Gelly et Silver 2011), l'hypothèse de RAVE est que la valeur de l'action a dans l'état s sera similaire dans tous les états du sous-arbre de s. Ainsi, la valeur de a est estimée à partir de toutes les simulations commençant par s, quel que soit le moment exact où a est choisie. La motivation est de minimiser le problème des estimations précoces, *i.e.* l'estimation de la valeur d'une action $Q(s,a)$ lorsque le nombre d'itérations est faible. Lorsqu'il n'y a pas beaucoup de simulations, RAVE sert à une évaluation plus robuste des actions en partageant les récompenses recueillies le long des différents sous-arbres de l'arbre de recherche.

Comme précisé dans (Sironi et Winands 2016), il existe de nombreuses variantes de la fonction de sélection RAVE comme (Gelly et Silver 2007), (Gelly et Silver 2011), (F. Teytaud et O. Teytaud 2010), ou encore (Finnsson et Björnsson 2010), nous utilisons dans nos travaux celle utilisée dans (Cazenave 2015) et définie ci-dessous :

$$Q^{\oplus}_{\text{RAVE}}(s,a) = (1.0 - \beta(s,a)) \times Q_{\text{UCT}}(s,a) + \beta(s,a) \times Q_{\text{AMAF}}(s,a) \tag{4.14}$$

dans laquelle $\beta(s,a)$ correspond à :

$$\beta(s,a) = \frac{m(s,a)}{m(s,a) + n(s,a) + \text{bias} \times m(s,a) \times n(s,a)} \tag{4.15}$$

où la valeur de la constante *biais* permet d'introduire un biais d'exploration. Cette valeur doit être choisie en fonction du problème. La politique de RAVE

$$\pi_{\text{RAVE}}(s) = \underset{a \in A(s)}{\text{argmax}}\, Q^{\oplus}_{\text{RAVE}}(s,a) \tag{4.16}$$

Il s'agit d'une combinaison linéaire de l'estimation d'UCT et de celle d'AMAF avec une pondération dynamique. Lorsqu'il n'y a pas beaucoup de simulations ($n(s)$ est faible), la valeur Q_{AMAF} est utilisée pour choisir une action. Lorsque le nombre de simulations augmente, la pondération $\beta(s,a)$ diminue, donc le poids de la valeur AMAF diminue en proportion et c'est la valeur Q_{UCT} qui prend plus d'importance. À mesure que le nombre

de simulations augmente, Q_{UCT} est une estimation plus fiable, car elle ne comporte pas de biais.

Dans (Gelly et Silver 2011), l'idée d'utiliser les valeurs AMAF d'un nœud ancêtre pour améliorer l'estimation de la valeur Q_{AMAF} lorsque le nombre de simulation est trop faible est évoquée. Cette idée est expérimentée dans *Generalized Rapid Action Value Estimate* (GRAVE).

4.2.3.3 *Generalized Rapid Action Value Estimate*

GRAVE (Cazenave 2015) est une modification de RAVE permettant de surmonter le problème suivant : pour les nœuds proches des feuilles de l'arbre de recherche, l'estimation AMAF, $Q_{\text{AMAF}}(s,a)$, est basée sur un trop faible nombre de simulations, et c'est d'autant plus le cas pour $Q_{\text{UCT}}(s,a)$. Dans ces nœuds, les estimations sont beaucoup moins précises que pour celles dont les nœuds sont placés plus haut (profondeur plus petite) dans l'arbre de recherche. GRAVE utilise les scores AMAF d'un nœud ancêtre pour les nœuds dont le nombre de visites est inférieur à une valeur *ref* fournie en entrée. Chaque nœud de l'arbre mémorise ses propres scores AMAF, mais conserve également une référence à l'ancêtre le plus proche, appelé *sref* qui a un nombre suffisant de visites *ref* pour que ses scores AMAF soient considérés comme fiables. *ref* est une constante spécifique au problème. Lorsqu'un nœud s a suffisamment de visites ($n(s) > ref$), il commence à utiliser ses propres valeurs AMAF au lieu de celles d'un ancêtre, et la fonction de sélection appliquée à ce nœud commence à se comporter comme celle de RAVE. Notons que si $ref = 0$, GRAVE se comporte exactement comme RAVE. La stratégie GRAVE permet d'augmenter la précision des estimations pour les nœuds les moins visités.

Nous décrivons la fonction de sélection pour GRAVE dans l'algorithme 6.

Algorithm 6 Pseudo-code de la fonction `Selection` de GRAVE.

Input: un état initial du PDM : s_0, une constante d'exploration : c, une valeur de référence de GRAVE : *ref*

Output: le nœud sélectionné : $s_{\text{selection}}$

$s, sref \leftarrow s_0, s_0$
while s est non terminal OU $n(s) == 0$ **do**
  if $n(s) > ref$ OU $n(sref) \leq ref$ **then**
    $sref \leftarrow s$
  end if
  $a \leftarrow \underset{a \in C_{\text{action}}(s)}{\text{argmax}} \; (1.0 - \beta(s,a)) \times Q_{\text{UCT}}(s,a) + \beta(s,a) \times Q_{\text{AMAF}}(sref,a)$
  $s \leftarrow T(s,a)$
end while
$s_{selection} \leftarrow s$
return $s_{selection}$

En résumé, le principe de GRAVE est d'utiliser les valeurs AMAF d'un état situé plus haut dans l'arbre de recherche que n'est placé l'état actuel (cet état est souvent une feuille ou proche d'une feuille). Même si les statistiques d'un ancêtre se réfèrent à des actions effectuées antérieurement, cet état, situé plus haut dans l'arbre, a une meilleure précision, car il a plus de simulations associées. GRAVE montre que l'utilisation des statistiques des

nœuds ancêtres, lorsque le nombre de simulations est trop faible, rend plus précises les estimations des statistiques du nœud plus bas dans l'arbre.

La figure 4.8 illustre la différence lors de l'étape de sélection de GRAVE et de RAVE. Cet exemple représente l'étape de sélection appliquée au nœud gris pour le choix d'un de ses enfants (nœuds avec un point d'interrogation). La valeur de *ref* vaut 30. Chaque nombre dans les nœuds indique le nombre de visites ($n(s)$). Dans le cas de RAVE, ce sont les valeurs AMAF du nœud gris qui sont utilisées pour sélectionner un des nœuds enfants, tandis que dans le cas de GRAVE, il s'agit de celles du nœud parent (avec $n(s) = 60$).

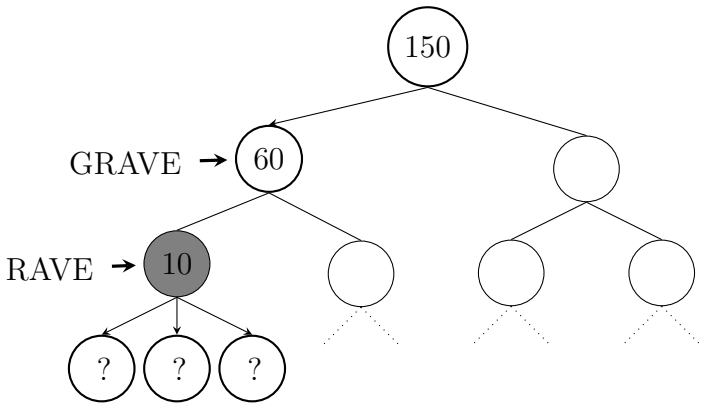


FIGURE 4.8 – Quel nœud enfant choisir ? Comparaison des informations utilisées pour la fonction de sélection par RAVE et par GRAVE. L'illustration est inspirée de (SIRONI et WINANDS 2016).

Dans le PDM des RRGHs, les espaces des actions et des états sont continus. Nous nous intéressons donc maintenant à l'état de l'art de MCTS dans le cadre continu.

4.2.4 Recherche arborescente Monte Carlo : cadre continu

Une hypothèse fondamentale pour l'application des politiques telles que celle d'UCT ou RAVE est qu'elles exigent que chaque action soit essayée au moins une fois. Elles ne sont pas applicables dans les cas où la cardinalité de l'espace d'actions est très grande par rapport au nombre d'itérations i ($|A| >> i$), ou lorsque le facteur de branchement est trop grand. Nous considérons ici le cadre continu où le domaine des états S et celui des actions A sont continus. Ainsi, les ensembles sont infinis, qu'ils soient bornés ou non.

Dans cette partie, nous présentons certains des travaux existants visant à étendre MCTS aux états et actions continues, sur lesquels se fondent certaines de nos contributions.

4.2.4.1 *Progressive Widening*

Alors que dans un cadre fini, il est possible d'explorer au moins une fois toutes les actions légales d'un état, cela n'est pas possible lorsque le nombre d'actions devient trop important (en particulier, lorsque ce nombre est infini). Puisqu'on ne peut pas explorer toutes les actions à la fois, l'idée est plutôt d'explorer progressivement de nouvelles actions. Cependant, il reste à déterminer à quel rythme explorer les nouvelles actions.

Des réponses ont été apportées sous le nom de *progressive strategies* dans (Coulom 2007a ; G. M. J. Chaslot et al. 2008). Nous nous intéressons à la plus référencée d'entre elles, l'heuristique *progressive widening* (PW). L'idée est de limiter le nombre d'actions à considérer d'un état s en fonction du nombre de fois que cet état a été visité ($n(s)$). Cette limite est contrôlée par un coefficient $\zeta \in]0, 1[: n(s)^{\zeta}$. Plus ce coefficient est proche de 0, plus le nombre d'actions à considérer sera faible. À l'inverse, plus ζ sera proche de 1, plus le nombre d'actions échantillonnées sera haut. (Y. Wang, Audibert et Munos 2008) propose une analyse théorique de cette stratégie. Les valeurs par défaut sont $\frac{1}{2}$ ou $\frac{1}{4}$, cependant il convient d'optimiser ce paramètre en fonction du problème.

Intuitivement l'équation (4.17) permet de faire un choix entre profondeur (descendre dans l'arbre de recherche) et largeur (élargir l'arbre de recherche en échantillonnant une nouvelle action). Plus précisément, si l'équation est vérifiée, alors cela signifie que le nombre de fois que s a été visité est plus grand que le nombre d'actions qu'il possède, $C_{\text{action}}(s)$. Cela implique que les actions ont été suffisamment sélectionnées (exploitation) et qu'il est temps d'agrandir l'ensemble $C_{\text{action}}(s)$ en échantillonnant une nouvelle action (exploration). À l'inverse, si l'équation n'est pas vérifiée, alors s n'a pas été assez visité, signifiant que les actions doivent être encore testées.

$$n(s)^{\zeta} > |C_{\text{action}}(s)| \tag{4.17}$$

L'algorithme 7 propose un pseudo-code du choix réalisé dans l'équation (4.17). Le point essentiel est que le choix de l'action se limite à un sous-ensemble fini de $A(s)$: $C_{\text{action}}(s)$. Et la taille de ce sous-ensemble augmente progressivement en y ajoutant des actions échantillonnées selon l'échantillonneur choisi (*sampler*). L'ajout d'une action est de plus en plus rare à mesure que le nombre de visites de s augmente.

Algorithm 7 Pseudo-code de l'heuristique PW.

Input: un coefficient : ζ, un nœud de $\mathcal{T}$: s, un échantillonneur : *sampler*
Output: une action a de s

1: **if** $n(s)^{\zeta} > |C_{\text{action}}(s)|$ **then**
2: $\quad a \leftarrow sampler(A(s))$
3: $\quad C_{\text{action}}(s) \leftarrow C_{\text{action}}(s) \cup \{a\}$
4: **else**
5: $\quad a \leftarrow \pi(s)$ avec $\pi(s) = \underset{a \in C_{\text{action}}(s)}{\text{argmax}} \; Q(s, a)$
6: **end if**
7: **return** a

Pour remédier à la limitation du nombre d'actions infinies, le nombre de bras à considérer dans le problème du bandit manchot d'un état est restreint à l'aide de l'heuristique PW, tandis que le choix des bras est contrôlé par la politique de l'arbre π. En d'autres termes, la fonction de sélection garantit que l'arbre s'étend dans les régions prometteuses de l'espace de recherche en alternant entre exploration et exploitation. La stratégie PW garantit qu'il s'élargit progressivement dans ces régions.

La stratégie PW appliquée à UCT est appelée *continuous upper confidence bound for tree* (cUCT).

4.2.4.2 *Continuous upper confidence bound for tree*

Appliquer la stratégie PW à UCT implique de modifier la fonction de sélection comme proposé dans l'algorithme 8. Le pseudo-code combine l'algorithme 2 et l'algorithme 7 et l'étape d'expansion est maintenant intégrée dans cette combinaison.

Algorithm 8 Pseudo-code de l'heuristique PW intégrée à la fonction `Selection` de UCT.

Input: un coefficient : ζ, un nœud racine de $\mathcal{T}$: s_0, un échantillonneur : *sampler*
Output: un nœud sélectionné : $s_{\text{selection}}$

$s \leftarrow s_0$
while s est non terminal ou $n(s) == 0$ **do**
  if $n(s)^{\zeta} > |C_{\text{action}}(s)|$ **then**
    $a \leftarrow sampler(A(s))$
    $C_{\text{action}}(s) \leftarrow C_{\text{action}}(s) \cup \{a\}$
  else
    $a \leftarrow \pi_{\text{UCT}}(s)$
  end if
  $s \leftarrow T(s, a)$
end while
return a

De la même manière que RAVE est souvent préférée à UCT, *continuous rapid action value estimation* (cRAVE) est souvent préférée à cUCT.

4.2.4.3 *Continuous rapid action value estimation*

cRAVE a été introduit dans (Couëtoux et al. 2011), motivé par des applications en gestion et en robotique. Ces travaux prennent racine dans l'observation suivante : bien que l'approche RAVE permette une estimation rapide des valeurs d'action $Q_{\text{RAVE}}(s, a)$, sa fiabilité diminue à mesure que le nombre d'actions légales d'un état $A(s)$ augmente, *ceteris paribus* (toutes choses étant égales par ailleurs). En effet, dans un espace d'actions continues A, le nombre de fois qu'une action donnée $a \in A$ est essayée est égal à 0 en espérance, ce qui rend RAVE inefficace. Les formules introduites ci-après suivent les notations de (Couëtoux et al. 2011).

cRAVE propose de considérer une estimation lissée des valeurs d'action en utilisant une convolution Gaussienne. Étant donné un ensemble $D = \{(x_j, y_j)_{j=1,..,n}\}$ tel que $x_j \in \mathbb{R}^m$ avec $m \in \mathbb{N}^*$ et $y_j \in \mathbb{R}$, une estimation gaussienne d'une valeur y associée à $x \in \mathbb{R}^m$ est définie comme :

$$\hat{y}_\sigma(x) = \frac{1}{\sum_{j=1}^{n} e^{-\frac{1}{\sigma^2} d(x_j, x)^2}} \sum_{j=1}^{n} e^{-\frac{1}{\sigma^2} d(x_j, x)^2} \times y_j \tag{4.18}$$

où σ est un paramètre de lissage qui contrôle l'importance relative des plus proches voisins de x et d est une fonction de dissimilarité sur $\mathbb{R}^m$.

Dans le cas de cRAVE, D correspond à un ensemble de paires état-action sélectionnées lors d'une simulation et du résultat associé z (ou le gain, la récompense cumulative). On note $w = \{(s_t, a_t)_{t=0,...,H}\}$ l'ensemble de ces paires et $z(w)$ le résultat (c'est l'équivalent du résultat de la i-ème simulation noté z_i de l'équation (4.8)). La valeur d'un état s suivie

d'une action a est définie par :

$$Q_{\text{cRAVE}_a}(s,a) = \frac{1}{\sum_{w,a_t \in w} e^{-logN_a \frac{d(a_t,a)^2}{\alpha_{action}}}} \sum_{w,a_t \in w} e^{-logN_a \frac{d(a_t,a)^2}{\alpha_{action}}} \times z(w) \tag{4.19}$$

où α_{action} correspond à σ et est un paramètre dépendant du problème, N_a correspond au nombre total d'actions dans toutes les simulations et le terme $log(N_a)$ est introduit pour augmenter la valeur de Q_{cRAVE_a} à mesure que le nombre d'itérations augmente.

Une différence importante entre Q_{RAVE} et Q_{cRAVE_a} est que le nombre de simulations w à considérer est beaucoup plus grand dans le second cas. En effet, dans RAVE seules les simulations contenant l'action a sont utilisées, tandis que dans cRAVE toutes les simulations sont considérées avec un poids qui décroît exponentiellement en fonction de la distance entre les actions exécutées dans les simulations et l'action considérée a. Cependant, Q_{cRAVE_a} n'est calculé que pour un nombre fini d'actions en raison de la stratégie PW (section 4.2.4.1). Comme pour RAVE, la mise à jour est effectuée après chaque simulation.

Q_{cRAVE_a} ne prend en compte que le cas où le domaine des actions est continu. Dans l'optique où l'espace des états est aussi continu, cRAVE propose de pondérer cette contribution relative en intégrant aussi une estimation lissée en fonction de la distance entre l'état considéré s et les états s_t de la simulation w :

$$Q_{\text{cRAVE}_{s,a}}(s,a) = \frac{1}{\sum_{w,a_t \in w} e^{-logN_{s,a}\{\frac{d(s_t,s)^2}{\alpha_{state}} + \frac{d(a_t,a)^2}{\alpha_{action}}\}}} \sum_{w,a_t \in w} e^{-logN_{s,a}\{\frac{d(s_t,s)^2}{\alpha_{state}} + \frac{d(a_t,a)^2}{\alpha_{action}}\}} \times z(w) \tag{4.20}$$

où α_{state} correspond à σ et est un paramètre dépendant du problème, $N_{s,a}$ est le nombre d'états et d'actions dans toutes les simulations.

Il est intéressant de noter que $Q_{\text{cRAVE}_{s,a}}$ est une généralisation de Q_{cRAVE_a} avec $\alpha_{state} = \infty$ qui est elle-même une généralisation de Q_{RAVE} avec $\alpha_{action} = \infty$. Le choix des paramètres α_{state} et α_{action} ainsi que la fonction de dissimilarité sont des paramètres à ajuster en fonction du problème.

4.3 Contributions

L'heuristique GRAVE permet d'augmenter l'efficacité de MCTS sur de nombreux problèmes (Cazenave 2015). À notre connaissance, GRAVE n'a été appliqué jusqu'à présent que sur des problèmes combinatoires. Motivés par notre problème d'identification des paramètres dynamiques dans les RRGHs, une première contribution consiste en l'extension de GRAVE au domaine continu, *continuous* GRAVE (cGRAVE), en utilisant un lissage basé sur la convolution gaussienne. Nous proposons deux autres contributions qui visent à améliorer l'efficacité de cGRAVE. L'approche proposée est validée expérimentalement sur le cycle cellulaire à cinq variables (5G). Le contenu de cette section peut être retrouvé en grande partie dans (Michelucci, Pallez et al. 2024).

L'algorithme 9, noté $\text{cGRAVE}_{\text{AD,CSP}}$, synthétise les différentes contributions. Pour en faciliter la lecture, la boucle principale est divisée : l'algorithme 10 contient le *progressive widening* et l'algorithme 11 intègre le *rollout* et la rétropropagation.

Algorithm 9 Pseudo-code de cGRAVE$_{\text{AD,CSP}}$.

Input: un état initial s_0, un budget computationnel *budget*
Output: un arbre de recherche $\mathcal{T}$

1: Initialiser les contraintes du module CSP
2: Création d'un nœud racine avec l'état s_0
3: **while** *budget* n'est pas atteint **do**
4: $s \leftarrow \texttt{PW}(s_0)$ ▷ algorithme 10
5: $z \leftarrow \texttt{Rollout}(s)$ ▷ algorithme 11
6: $\texttt{Retropropagation}(z, s)$ ▷ algorithme 11
7: **end while**

Algorithm 10 Pseudo-code de l'étape de `PW` dans cGRAVE$_{\text{AD,CSP}}$.

Input: le nœud racine de l'arbre $\mathcal{T}$ contenant l'état s_0, un coefficient d'élargissement ζ, un échantillonneur *sampler*, et une valeur de référence ref et un facteur de biais *bias* pour GRAVE
Output: un nœud s de $\mathcal{T}$

1: $s, sref \leftarrow s_0, s_0$
2: **while** s est non terminal et $n(s) == 0$ et s n'est pas simulable **do**
3: **if** $n(s)^{\zeta} > |C_{\text{action}}(s)|$ **then** ▷ équation (4.17)
4: $a \leftarrow sampler(A_{\text{CSP}}(s))$
5: $C_{\text{action}}(s) \leftarrow C_{\text{action}}(s) \cup \{a\}$
6: **else**
7: **if** $n(sref) \leq ref$ OU $n(s) > ref$ **then** ▷ GRAVE
8: $sref \leftarrow s$
9: **end if**
10: **for all** $a \in C_{\text{action}}(s)$ **do**
11: $\beta(s,a) \leftarrow \frac{m(sref,a)}{m(sref,a)+n(s,a)+bias \times m(sref,a) \times n(s,a)}$
12: **end for**
13: $a \leftarrow \underset{a \in C_{\text{action}}(s)}{\text{argmax}} \; (1.0 - \beta(s,a)) \times Q_{\text{UCT}}(s,a) + \beta(s,a) \times Q_{\text{AMAF}}(sref, a)$
14: **end if**
15: $s \leftarrow T(s,a)$
16: **end while**
17: **return** s

4.3.1 GRAVE dans le domaine continu (cGRAVE)

GRAVE utilise les valeurs AMAF d'un état moins profond dans l'arbre de recherche que ne l'est l'état actuel. Le but est d'obtenir des estimations plus précises de la fonction valeur au niveau de ses feuilles. Cependant, de la même manière que RAVE, sa fiabilité diminue à mesure que le facteur de branchement augmente : dans un espace d'actions continu, le nombre de fois qu'une action donnée est choisie est nul en espérance. Une estimation de la fonction valeur état-action doit être envisagée. Nous choisissons le lissage par convolution gaussienne. L'algorithme qui en résulte étend GRAVE au domaine continu (cGRAVE). La seule différence avec son équivalent discret est que les statistiques AMAF sont mises à jour à l'aide d'un lissage par convolution gaussienne (la ligne 9 de

Algorithm 11 Pseudo-code de `Rollout` et de `Retropropagation` dans cGRAVE$_{\text{AD,CSP}}$.

Input: le nœud s renvoyé par l'algorithme 10

1: $S_{visite}, A_{visite} \leftarrow \{\}, \{\}$
2: **while** s est non terminal **do**
3: $\quad a \leftarrow \pi_{default}(s)$ tel que $a \in A_{\text{CSP}}(s)$
4: $\quad s \leftarrow T(s, a)$
5: $\quad S_{visité}, A_{visité} \leftarrow S_{visité} \cup \{s\}, A_{visité} \cup \{a\}$
6: **end while**
7: $z \leftarrow$ `evaluer`(s)
8: `Retropropagation`(z, s) ▷ algorithme 5
9: Mettre à jour $Q_{\text{cRAVE}_{a,s}}(s, a)$, selon $\forall s \in S_{visite}, \forall a \in A_{visite}$ ▷ équation (4.20)

l'algorithme 11 suit l'équation (4.20)). Pour le calcul de la *tree policy* de cGRAVE (lignes 10 à 13 de l'algorithme 10), l'état ancêtre le plus proche ayant été visité par plus de simulations qu'une constante *ref* donnée est conservé comme état de référence, appelé *sref* dans le pseudo-code, et mis à jour dans les lignes 7 à 9 de l'algorithme 10).

4.3.2 Stratégie de décomposition des actions

De nombreux problèmes du monde réel requièrent des espaces d'actions continus, $a \in \mathbb{R}^d, d \in \mathbb{N}^*$. En outre, la dimension d de l'espace d'actions peut également être élevée, ce qui rend les variants de MCTS dans le cadre continu moins efficaces.

Pour résoudre ce problème, nous proposons une stratégie de décomposition des actions (AD) qui consiste à décomposer chaque action formée de d composantes (un d-uplet $a = (x_1, ..., x_d)$) en d actions unidimensionnelles $(a_1 = x_1, ..., a_d = x_d)$. Par conséquent, le choix de chaque composante de l'action est laissé à la *tree policy*.

Pour plus de clarté, nous distinguons ici les nœuds d'état qui correspondent aux nœuds déjà introduits, des nœuds d'action qui représentent la d-ème composante de a. Dans la figure 4.9, les premiers sont représentés par des cercles, les seconds par des points noirs. Lors de l'étape d'expansion, à partir d'un nœud d'état, un premier composant d'action est échantillonné et un nœud d'action est ajouté à l'arbre de recherche. Puis, à partir de ce nœud d'action, un nouveau nœud d'action est développé. En itérant d fois, le choix du composant final conduit à un nouveau nœud d'état à partir duquel l'étape de simulation a lieu.

En utilisant cette stratégie, l'arbre de recherche contient des nœuds à partir desquels aucune simulation ne peut être effectuée (les nœuds d'action). La ligne 2 de l'algorithme 10 (« [...] s n'est pas simulable ») vérifie si s_t est un état simulable ou, en d'autres termes, s'il s'agit bien d'un nœud d'état. Si s n'est pas simulable, une nouvelle composante de l'action doit être échantillonnée. Enfin, si la stratégie AD n'est pas choisie, s est toujours simulable puisque l'action choisie qui mène à s comprend déjà d composantes.

L'avantage d'une telle stratégie est que chaque nœud d'action est considéré comme un nœud de l'arbre de recherche lors de l'étape de sélection, ce qui conduit à une *tree policy* qui renforce progressivement les meilleures composantes d'action au détriment des mauvaises. La figure 4.9 illustre cette stratégie : les actions $a_i = (a_{i,1}, ..., a_{i,d})$, $i \in [\![1, n]\!]$ et $n \in \mathbb{N}^*$ sont décomposées de telle sorte que la recherche arborescente qui en résulte est

composée des composantes d'action distinctes.

L'efficacité de cette stratégie dépend de l'ordre dans lequel les composantes d'action sont développées. Dans un cadre général, il est souvent impossible d'obtenir un ordre optimal des actions, soit parce que les composantes sont dépendantes entre elles, soit parce que la fonction d'ordre des composantes est inconnue. Si aucune information n'est disponible, un ordre aléatoire des composantes peut être choisi. Sinon, une heuristique ou même un calcul dynamique de l'ordre des composantes, par exemple, peut augmenter la vitesse de convergence et conduire à un cas de complexité plus favorable.

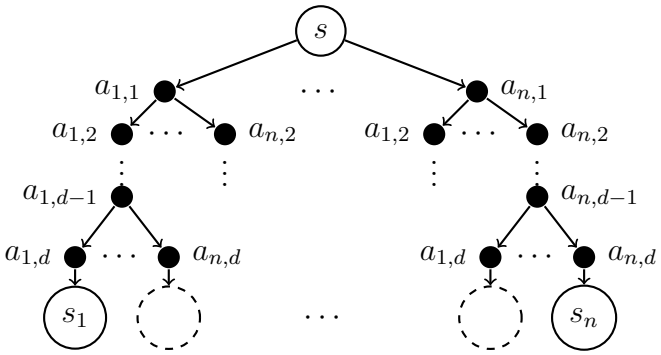


FIGURE 4.9 – Illustration de la stratégie de décomposition des actions. Les actions sont des actions multidimensionnelles, *i.e.*, des uplets de d composantes. Chacune des composantes est représentée par un cercle noir. Les actions sont décomposées, composante par composante, en suivant un ordre et formant une structure d'arbre.

4.3.3 Politique sélective basée sur les contraintes

Bien que les méthodes de MCTS soient agnostiques par rapport au domaine, des améliorations significatives de la performance peuvent souvent être obtenues en utilisant des connaissances spécifiques au domaine. Ces connaissances n'ont pas besoin d'être exhaustives tant qu'elles sont capables de biaiser la sélection des actions de manière favorable, *i.e.*, d'orienter le choix vers des actions prometteuses.

La proposition que nous faisons est une *politique sélective basée sur les contraintes* (CSP). Elle prend ses racines dans les politiques sélectives (*selective policies*) de (CAZENAVE 2017) introduites avec succès dans un variant de MCTS, *Nested Rollout Policy Adaptation*. Le principe d'une politique sélective consiste à obtenir plus de sélectivité lors de ces simulations. Cela peut être appliqué à n'importe quel problème en modifiant l'ensemble des actions légales ($A(s)$) de manière à ce que les actions qui ont peu de chances d'être bonnes soient élaguées au cours des simulations. L'idée du CSP est d'extraire automatiquement des contraintes à partir de la définition de l'environnement, de certaines données d'entrée ou de connaissances d'experts. L'objectif est de réduire les espaces d'actions légales de sorte que les valeurs d'action qui ne sont pas susceptibles d'être bonnes, voire infaisables, au regard d'une contrainte extraite soient élaguées. Cet élagage peut arriver soit avant, soit pendant l'exécution.

Un module est construit pour extraire les contraintes qui peuvent être utilisées pendant l'étape d'expansion pour choisir uniquement les actions légales et supprimer celles

qui conduiraient à un état impossible (un état dans lequel il y a un ensemble vide d'actions légales). Ce module est donc d'abord utile dans l'arbre pour sélectionner les actions prometteuses, mais il peut également être utilisé pendant l'étape de simulation, où le partage d'informations (comme l'estimation par noyau ou les statistiques AMAF) n'a pas lieu. Il est donc utile tout au long de la simulation : pendant l'étape de sélection et pour biaiser la *default policy*.

Dans les applications réelles, l'espace des actions $A(s)$ d'un état s est borné, de sorte que chaque action a_i est définie à l'intérieur d'un domaine spécifié D_i. Chaque domaine D_i consiste en un ensemble de valeurs possibles. Le CSP permet d'extraire un ensemble de contraintes C_i, qui réduit son domaine D_i correspondant. Un tel espace d'action est noté $A_{CSP}(s)$ (ligne 4 dans l'algorithme 10 et ligne 3 dans l'algorithme 11). L'extraction de ces contraintes permet de réduire la dimension de D_i non seulement aux actions légales, mais donc aussi aux actions ayant une plus grande probabilité de faire partie d'une solution. Elle peut être effectuée *a priori* et pendant l'exécution de la recherche. Le module permet d'atténuer le problème selon lequel certaines actions d'un état s peuvent ne jamais être atteintes en raison d'une action précédemment exécutée, même si cette action appartient à l'ensemble légal $A(s)$. La construction d'un tel module d'extraction peut être considérée comme un cadre généralisé applicable à de multiples applications distinctes au prix de la conception du module : les contraintes dépendent du domaine d'application. Un cas d'utilisation est présenté dans la section suivante qui présente l'étude expérimentale sur le problème étudié.

4.4 Étude expérimentale

L'étude expérimentale, présentée en section 4.4.1, suit un plan d'expérimentation dans lequel les performances des différents variants (cRAVE et cGRAVE) sont testés en ajoutant ou retirant les différentes contributions (la stratégie de décomposition des actions et le module CSP) afin de comprendre la contribution de chaque contribution (*ablative study*). Nous introduisons ensuite quelques explications techniques au niveau du module CSP (section 4.4.2) et de la fonction d'évaluation d'une simulation (section 4.4.3). Finalement, une analyse des résultats obtenus (section 4.4.4), une analyse statistique (section 4.4.5) et une visualisation des modèles extraits (section 4.4.6) sont proposées.

4.4.1 Plan d'expérimentation

L'objectif des expériences à venir est d'évaluer les performances des différentes contributions sur notre problème. Nous comparons par incrément les différentes contributions avec pour référence cRAVE. Tout d'abord, cRAVE est comparé à cRAVE auquel la politique sélective basée sur les contraintes a été ajoutée ($\text{cRAVE}_{\text{CSP}}$). Dans l'étape suivante, nous cumulons la stratégie de décomposition des actions ($\text{cRAVE}_{\text{CSP-AD}}$). De la même manière, nous remplaçons l'heuristique cRAVE par cGRAVE.

Pour éviter les inconvénients du réglage manuel et *ad hoc* des paramètres des algorithmes, nous avons décidé de mettre en place une procédure de configuration automatique des algorithmes. Nous avons utilisé le package `irace` (López-Ibáñez et al. 2016) et nous avons conservé la meilleure configuration (élite) obtenue après 1 000 itérations pour déterminer les valeurs des paramètres dépendant du problème. Pour configurer cRAVE,

l'espace des paramètres et les valeurs obtenues sont présentées dans le section 4.4.1. Pour configurer l'heuristique cGRAVE, les paramètres de cRAVE sont gardés et *ref* est ajouté. Parmi les valeurs $[\![1, 100]\!]$, la valeur obtenue de ref est 29. La distance euclidienne est toujours choisie pour les espaces d'actions et d'états. Chaque expérience est exécutée 30 fois pour obtenir des résultats statistiquement significatifs. Les différentes solutions obtenues et leur récompense associée sont comparées pour le même budget computationnel, c'est-à-dire 200 000 itérations. Ce chiffre provient des expériences réalisées dans (Michelucci, Callegari et al. 2024).

Paramètres	$bias$	α_{state}	α_{action}	ζ
Domaine	$\{10^{-15}, ..., 10^{-2}, 0.1\}$	$[\![1, 100]\!]$	$[\![1, 100]\!]$	$\{0.1, 0.11, ..., 0.9\}$
Valeurs obtenues	0.1	47	87	0.61

Tableau 4.2 – Espace des paramètres et valeurs obtenues pour la configuration de cRAVE par `irace`.

Avant de présenter les résultats, deux spécificités nécessaires à la réalisation des expérimentations sont introduites : le module CSP (présenté en section 4.3.3) et la fonction `evaluer` utilisée dans l'algorithme 11.

4.4.2 Module CSP pour les paramètres dynamiques

Le module CSP utilise des connaissances du domaine pour extraire des contraintes sur les paramètres dynamiques (les actions de l'arbre de recherche). Elles sont extraites avant l'exécution et pendant la construction de l'arbre de recherche. Le module est utilisé pour améliorer la convergence de MCTS.

Connaissance du domaine. Pour développer le module CSP, nous avons utilisé certaines connaissances sur le cadre hybride. En effet, certaines célérités, c'est-à-dire des composantes du d-uplet d'action, sont identiques dans différents états discrets qui seront rencontrés par la trajectoire, voir tableau 1.2. Cela provient des interactions du graphe d'interaction. Pour illustrer le fonctionnement du module, nous prenons l'exemple d'une sous-partie du triplet de Hoare hybride du réseau 5G, *cf.* équation (4.21).

$$\{h_i\} \underbrace{\begin{pmatrix} 3.33 \\ slide^-(EP) \\ SK+ \end{pmatrix}; \begin{pmatrix} 3.33 \\ \top \\ SK+ \end{pmatrix}; \begin{pmatrix} 3.33 \\ slide^+(SK) \\ En- \end{pmatrix}}_{G1}; \underbrace{\begin{pmatrix} 2.0 \\ slide^-(En) \\ A+ \end{pmatrix}; ... ; ...}_{S}; \{h_f\} \quad (4.21)$$

Avant l'exécution de MCTS. Le module a propagé certaines informations de la connaissance biologique entre les états qui partagent au moins une composante de célérité commune. Prenons l'exemple du troisième état discret (le dernier de la phase G1), où $\eta = (2, 0, 0, 1, 0)$. Le module a automatiquement extrait que la célérité de SK, $C_{SK,\{m_1,m_2\},2}$, est contrait par $slide^+(SK)$. Mais il a également extrait que l'état discret suivant partage la même célérité. Ainsi, le module a permis de déterminer que (i) la célérité de SK dans le troisième état est strictement positive en raison de la connaissance

de $slide^+(SK)$ (réduisant l'espace de recherche pour cette action à des valeurs positives uniquement : $A(s) =]0, 7]$) et (ii) que lorsque cette valeur est connue, elle est conservée dans l'état discret suivant.

Au cours de l'exécution. Le module a également contribué à l'éviction de certaines valeurs d'action en ligne. Dans le même exemple de $slide^+(SK)$ dans le troisième état discret, toute valeur positive pour C_{SK} est considérée comme un mouvement légal. Cependant, la valeur de C_{SK} est influencée par son point d'entrée, c'est-à-dire l'état hybride lors de l'entrée dans le troisième état discret. Par conséquent, en fonction des coordonnées du point d'entrée, les valeurs de C_{SK} sont ajustées en ligne.

Nous ne disposons d'aucune connaissance sur la fonction d'ordre des célérités dans la stratégie de décomposition des actions, c'est pourquoi nous avons défini un ordre arbitraire entre les variables du graphe d'interaction (d'abord SK, ensuite A, puis B, En, et enfin EP) et l'avons conservé pour chaque décomposition. Cependant, faire varier l'ordre des célérités pourrait avoir, dans certains cas, un impact sur le taux de convergence.

4.4.3 Évaluation d'une simulation et construction de l'arbre de recherche.

La fonction de récompense introduite en équation (4.1) est utilisée pour évaluer le score d'une simulation (fonction `evaluer` ligne 7 de l'algorithme 11), mais aussi pour contrôler (i) le nombre de nœuds dans l'arbre de recherche $\mathcal{T}$ et (ii) la longueur de la simulation H. En effet, si un nœud échantillonné n'obtient pas une récompense inférieure à $\epsilon = -10^{-2}$, alors le nœud n'est pas sauvegardé dans l'arbre de recherche, car l'état produit est invalide (selon la connaissance biologique). Cela permet d'éviter une accumulation de nœuds « morts » qui ne servent à rien mis à part utiliser de l'espace mémoire.

Dans l'étape de *rollout*, si un état est atteint et que son score est inférieur à ϵ, alors la simulation est stoppée. Le résultat d'un *rollout* est calculé comme une récompense cumulée, un gain de la manière suivante :

$$z = \sum_{i=0}^{H} \mathbb{1}_{R(s_t^i, a_t^i, s_{t+1}^i) \geq \epsilon} \tag{4.22}$$

où s_t^i est l'état s obtenu au pas de temps t de la i-ème simulation, R est la fonction récompense de l'équation (4.1), $\mathbb{1}_{R(s_t^i, a_t^i, s_{t+1}^i) \geq \epsilon}$ est la fonction indicatrice qui renvoie 1 si la récompense est supérieure à ϵ, 0 sinon ; et H correspond à l'état terminal de la simulation (soit l'épisode atteint son horizon, soit la trace atteint un état bloquant). Le score d'une simulation correspond à sa longueur ou, autrement dit, à la profondeur de l'état terminal de la simulation dans $\mathcal{T}$. Un résultat z optimal est donc égal à 12 dans le cas du cycle cellulaire (5G).

4.4.4 Analyse des résultats.

La figure 4.10 compare l'évolution monotone des gains obtenus, calculés selon l'équation (4.22), par les différents algorithmes testés. On peut observer que le module CSP est largement, mais sans surprise, bénéfique pour la convergence des différents variants

de MCTS : en collectant des contraintes lors des simulations. Il permet d'éviter un blocage précoce et réduit l'espace de recherche (visuellement, il s'agit d'une asymptote horizontale). On note ensuite que cRAVE$_{\text{CSP-AD}}$ et cGRAVE$_{\text{CSP}}$ contribuent tous deux à améliorer les résultats comparativement à cRAVE$_{\text{CSP}}$. Finalement, en combinant les trois contributions, cGRAVE$_{\text{CSP-AD}}$ garantit de toujours trouver une solution au problème, c.-à-d., une évaluation admissible des vecteurs de célérité permettant à la trace de satisfaire la connaissance biologique. Le seuil maximum de gain est atteint vers 180 000 itérations.

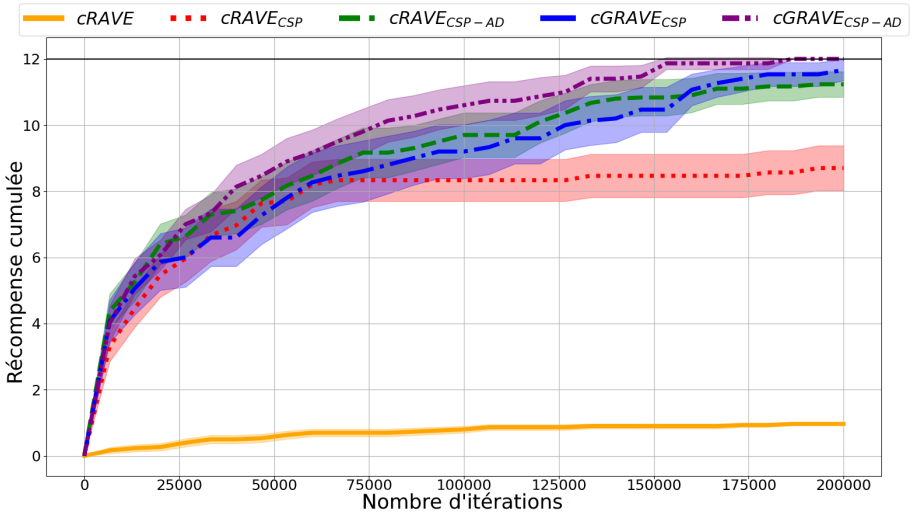


FIGURE 4.10 – Performances comparatives des récompenses cumulées, voir équation (4.22) obtenues par les différents variants en fonction du budget (nombre d'itérations) sur le cycle cellulaire à cinq variables. Plus la récompense est élevée, mieux c'est. Une récompense de 12 signifie qu'une solution a été trouvée (ligne horizontale noire).

Le tableau 4.3 résume les statistiques de la meilleure récompense cumulée obtenue pour chaque exécution. La moyenne et l'écart-type des résultats sont aussi indiqués, ainsi que les récompenses cumulées maximales et minimales (les meilleurs résultats, colonne par colonne, sont en gras). Pour retrouver la valeur *maximum* sur la figure 4.11, il faut identifier la valeur en abscisse du point le plus à droite de chaque courbe qui a une probabilité supérieure à 0% (en ordonnée).

Variant	avg±std	max	min	% de solutions
cRAVE	0.97 ± 0.18	1	0	0
cRAVE$_{\text{CSP}}$	8.7 ± 2.72	**12**	6	20
cRAVE$_{\text{CSP-AD}}$	11.2 ± 1.54	**12**	6	70
cGRAVE$_{\text{CSP}}$	11.6 ± 1.3	**12**	6	93.33
cGRAVE$_{\text{CSP-AD}}$	**12.0 ± 0.0**	**12**	**12**	**100**

Tableau 4.3 – Statistiques des performances obtenues par les différents variants testés. Les valeurs en gras indiquent les meilleurs résultats colonne par colonne.

Les courbes de répartition sont intégrées dans la figure 4.11. Chaque courbe décrit la probabilité de trouver une solution égale ou inférieure à un score de récompense cumulée donné. Par exemple, avec cRAVE$_{\text{CSP}}$, il y a 100% de probabilité qu'à la fin d'une exécution,

un utilisateur obtienne une récompense cumulée inférieure ou égale à 6, 50% de probabilité que la récompense cumulée soit comprise entre 6 et 11 et moins de 20% pour pour 12.

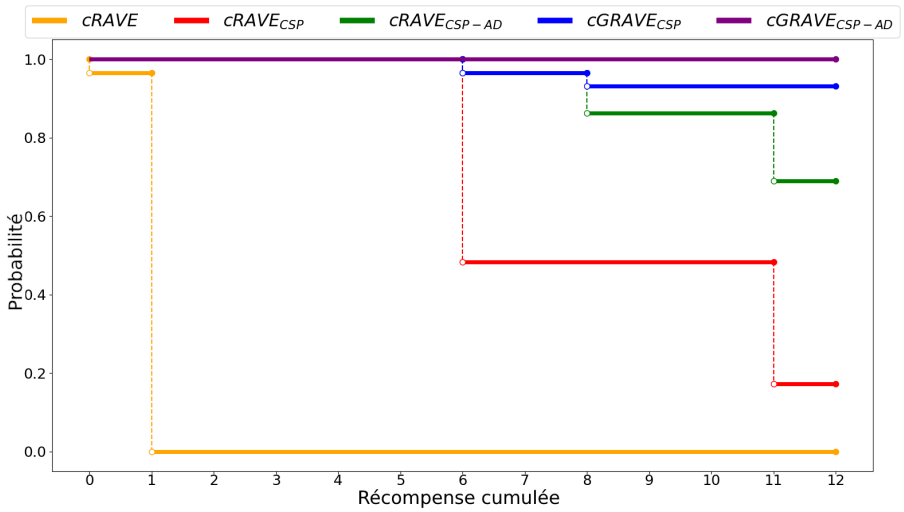


FIGURE 4.11 – Courbes de répartition montrant les meilleurs résultats des variants.

La figure 4.11 et la tableau 4.3 permettent de se rendre compte, quantitativement, de l'intérêt des différentes contributions. Sans CSP, aucune solution ne peut être trouvée. En cumulant le module CSP, la stratégie AD et l'heuristique cGRAVE, MCTS obtient une probabilité d'obtenir une meilleure récompense cumulative et augmente le pourcentage de solutions trouvées. La combinaison de nos propositions (cGRAVE$_{\text{CSP-AD}}$) nous permet d'obtenir une solution avec une probabilité de 100%.

4.4.5 Analyse statistique

Une campagne de validation statistique a été menée pour évaluer les différences observées dans les valeurs de performance de toutes les paires d'algorithmes afin de présenter le meilleur variant de MCTS sur le problème étudié. Nous considérons l'hypothèse nulle H_0 selon laquelle les scores de performance observés sont égaux. Tout d'abord, le choix entre les tests paramétriques et non paramétriques se fait en fonction de l'indépendance des échantillons (les graines sont différentes), de la normalité ou non de la distribution des échantillons de données (test de Kolmogorov-Smirnov) et de l'homoscédasticité des variances (test de Levene) (EFTIMOV et KOROŠEC 2021). Comme les conditions de normalité et d'homoscédasticité requises pour l'application des tests paramétriques ne sont pas remplies (au niveau de confiance $\alpha = 5\%$), le test non paramétrique ANOVA de Friedman (*Friedman rank-sum test*) est utilisé pour déterminer si au moins deux algorithmes présentent des différences significatives dans les valeurs de performance observées. La valeur p obtenue est environ égale à 7×10^{-21}, ce qui montre que les différences entre les algorithmes sont significatives. Cependant, nous ne savons toujours pas quelles paires d'algorithmes obtiennent des performances significativement différentes. C'est pourquoi le test non paramétrique des rangs signés de Wilcoxon (*Wilcoxon signed-rank test*) a été effectué. De manière complémentaire, pour réduire le problème des erreurs de Type I dans les comparaisons multiples, la méthode de correction de Bonferroni a été appliquée. Comme introduit dans la section 3.3.2, les erreurs de Type I correspondent au rejet de

l'hypothèse nulle qui, en réalité, n'aurait pas dû être rejetée. La correction de Bonferroni permet de réduire ce problème en corrigeant le seuil de significativité α lors de comparaisons multiples. Cela permet d'éviter de conclure de manière erronée sur la supériorité d'un algorithme par rapport à un autre.

■ Échec du rejet de H_0 □ Rejet de H_0

cRAVE	1.17e-6	6.94e-7	1.45e-7	6.79e-8
$\text{cRAVE}_{\text{CSP}}$		9.84e-5	5.98e-5	8.89e-6
$\text{cRAVE}_{\text{CSP-AD}}$			0.176	6.46e-3
$\text{cGRAVE}_{\text{CSP}}$				0.179

cRAVE	1.17e-5	6.94e-6	1.45e-6	6.8e-7
$\text{cRAVE}_{\text{CSP}}$		9.84e-4	5.98e-4	8.9e-5
$\text{cRAVE}_{\text{CSP-AD}}$			1.0	6.46e-2
$\text{cGRAVE}_{\text{CSP}}$				1.0
	$\text{cRAVE}_{\text{CSP}}$	$\text{cRAVE}_{\text{CSP-AD}}$	$\text{cGRAVE}_{\text{CSP}}$	$\text{cGRAVE}_{\text{CSP-AD}}$

Tableau 4.4 – Tests statistiques de Wilcoxon (en haut) et l'analyse *post hoc* Bonferroni (en bas) pour l'hypothèse nulle H_0.

Le tableau 4.4 montre, en haut, les valeurs p obtenues avec le test de Wilcoxon et, en bas, celles calculées avec la correction de Bonferroni.

Sur la base de l'acceptation ou du rejet des hypothèses nulles, nous arrivons aux conclusions suivantes : cRAVE est largement surpassé par les autres variants testés. En outre, $\text{cRAVE}_{\text{CSP}}$ est moins performant que $\text{cRAVE}_{\text{CSP-AD}}$, $\text{cGRAVE}_{\text{CSP}}$ et $\text{cGRAVE}_{\text{CSP-AD}}$. Les résultats obtenus par $\text{cGRAVE}_{\text{CSP}}$ ne sont pas statistiquement différents de ceux obtenus par $\text{cRAVE}_{\text{CSP-AD}}$ et $\text{cGRAVE}_{\text{CSP-AD}}$ (cases foncées dans le tableau 4.4). On peut toutefois souligner que les performances de $\text{cRAVE}_{\text{CSP}}$ sont significativement inférieures à celles de $\text{cGRAVE}_{\text{CSP}}$, et qu'il en va de même entre $\text{cRAVE}_{\text{CSP-AD}}$ et $\text{cGRAVE}_{\text{CSP-AD}}$.

4.4.6 Visualisation des résultats

La figure 4.12 montre la meilleure solution obtenue par $\text{cGRAVE}_{\text{CSP-AD}}$ pour chaque exécution. Les lignes verticales noires illustrent les seuils qui séparent les différents états discrets. Cette illustration confirme que les contributions proposées ont permis d'extraire des solutions, toutes cohérentes avec la connaissance biologique.

Elle illustre aussi qu'il existe une diversité de traces qui satisfont la connaissance biologique. Du point de vue du modélisateur, avoir à disposition un tel ensemble de traces biologiques permet de l'aider dans sa démarche exploratoire. Cependant, rappelons que ces 30 traces différentes sont obtenues en lançant MCTS 30 fois. Cette approche n'est pas envisageable si le nombre de traces requises augmente de manière significative.

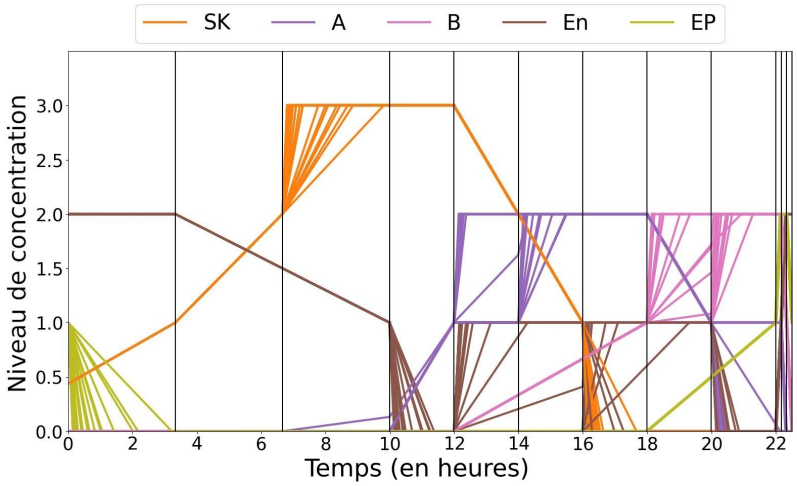


FIGURE 4.12 – Visualisation des 30 solutions (une pour chaque exécution) obtenues par $\text{cGRAVE}_{\text{CSP-AD}}$ pour le cycle cellulaire à cinq variables.

4.5 Synthèse

Le but de ce chapitre était de montrer qu'en formulant le problème comme une séquence de décisions, il était aussi possible d'obtenir un modèle valide de RRGH. Nous avons fait le choix de traiter ce problème à l'aide de la recherche arborescente Monte Carlo. Ce chapitre adapte le variant GRAVE, performant dans le cadre combinatoire, à un cadre continu. Deux autres améliorations heuristiques dans le cadre continu sont également introduites. La première consiste à décomposer une action multidimensionnelle en plusieurs actions unidimensionnelles pour que la stratégie de la *tree policy* s'applique aussi au choix de chacune des composantes d'actions. La seconde amélioration consiste à intégrer des connaissances du problème au cours de l'exécution de MCTS en extrayant des contraintes pour améliorer le choix des actions lors des simulations. L'étude expérimentale menée sur le cycle cellulaire a montré que la combinaison de ces améliorations permet d'identifier à chaque fois une trace qui satisfait la connaissance biologique.

Dans ce manuscrit, nous proposons deux approches différentes : l'optimisation stochastique avec EA et l'apprentissage par renforcement avec MCTS. Ces deux approches se concentrent sur la recherche d'une unique solution optimale à notre problème : un ensemble de paramètres dynamiques générant un modèle valide. Or, comme ces approches sont développées afin d'être utiles pour le modélisateur en biologie, la recherche d'un seul modèle n'est pas suffisante. En effet, le but du modélisateur est de raffiner les connaissances du système qu'il étudie. L'extraction d'un modèle unique est utile, mais elle a un impact limité sur l'apport de ces nouvelles connaissances sur le système. Du point de vue du modélisateur, il est intéressant d'avoir à sa disposition une diversité de modèles valides. Cela lui permet de raisonner non pas sur une seule identification possible, mais sur un ensemble d'identifications sensiblement différentes, mais toujours en accord avec la connaissance biologique.

Les travaux présentés dans la partie suivante ont pour objectif d'étendre les expérimentations réalisées dans le cadre d'optimisation avec EA et celui d'apprentissage par renforcement avec MCTS pour être capable de proposer un ensemble de solutions diverses au problème étudié.

Troisième partie

Recherche d'un ensemble de paramétrisations diverses

5 Optimisation multimodale

We consider optimization without derivatives one of the most important, open, and challenging areas in computational science and engineering, and one with enormous practical potential.

Extrait de (Conn, Scheinberg et Vicente 2009)

Dans le processus de résolution d'un problème, la possibilité de disposer de plusieurs solutions optimales, ou proches de l'optimal, avant de prendre une décision finale est désirable. En effet, si une solution ne convient pas, une autre solution peut ainsi être adoptée immédiatement sans avoir besoin de relancer le processus de résolution dans l'espoir d'obtenir une solution différente de la précédente. Le *Second Toyota Paradox* (Ward et al. 1995) démontre que dans le cadre du processus de production d'un constructeur automobile, le fait d'adopter simultanément plusieurs solutions potentielles permet de produire de meilleures voitures plus rapidement et à moindre coût.

En recherche opérationnelle, l'optimisation multimodale (*multimodal optimisation*, MMO) traite des tâches d'optimisation qui impliquent de trouver toutes ou la plupart des solutions multiples, au moins localement optimales, d'une fonction objectif en une seule exécution. Étant donné qu'il n'est pas garanti qu'en partant de différents points initiaux de l'espace de recherche, la technique d'optimisation aboutisse à des solutions différentes lorsque plusieurs exécutions sont lancées, les méthodes d'optimisation classiques ne sont pas adaptées à la recherche de solutions multiples. Les métaheuristiques d'évolution artificielle dotées de mécanismes de préservation de la diversité spécifiquement conçus pour localiser plusieurs solutions optimales, communément appelées méthodes de *niching*, se sont révélées particulièrement efficaces pour résoudre les problèmes de MMO.

Nous présentons dans un premier temps (section 5.1), un état de l'art sur les techniques de *niching* ainsi que les mesures de performance pour évaluer ces méthodes. Dans la section suivante (section 5.2), nous adaptons deux métaheuristiques à la résolution du problème d'identification de plusieurs paramétrisations des RRGHs. Après quoi, l'étude

expérimentale est détaillée dans la section 5.3. Enfin, la section 5.4 souligne l'originalité du problème étudié au regard des fonctions de référence (*benchmark*) et des cas d'étude de la communauté d'EA. Nous proposons d'introduire de nouvelles fonctions de référence et un axe de recherche futur.

5.1 État de l'art

Comparée à l'optimisation globale, l'**optimisation multimodale** vise à chercher plusieurs *optima* au cours d'une unique exécution et présente plusieurs bénéfices. D'un point de vue algorithmique, d'abord, cela permet d'éviter l'effet de la dérive génétique (*genetic drift*) menant la population à converger prématurément vers un *optimum* local. Comme le budget n'est pas concentré sur une unique zone de l'espace de recherche, cela permet d'augmenter la probabilité de trouver la ou les solutions optimales. D'un point de vue applicatif, ensuite, obtenir plusieurs solutions de haute qualité donne au décideur plusieurs options à considérer et ainsi peut fournir de nouvelles informations sur le problème (J.-P. Li et al. 2002).

5.1.1 Multimodalité

D'après (Wessing 2015) et (Preuss, M. Epitropakis et al. 2021), un problème d'optimisation multimodal (MMO), considérant la minimisation d'une fonction objectif f, peut être défini de la manière suivante :

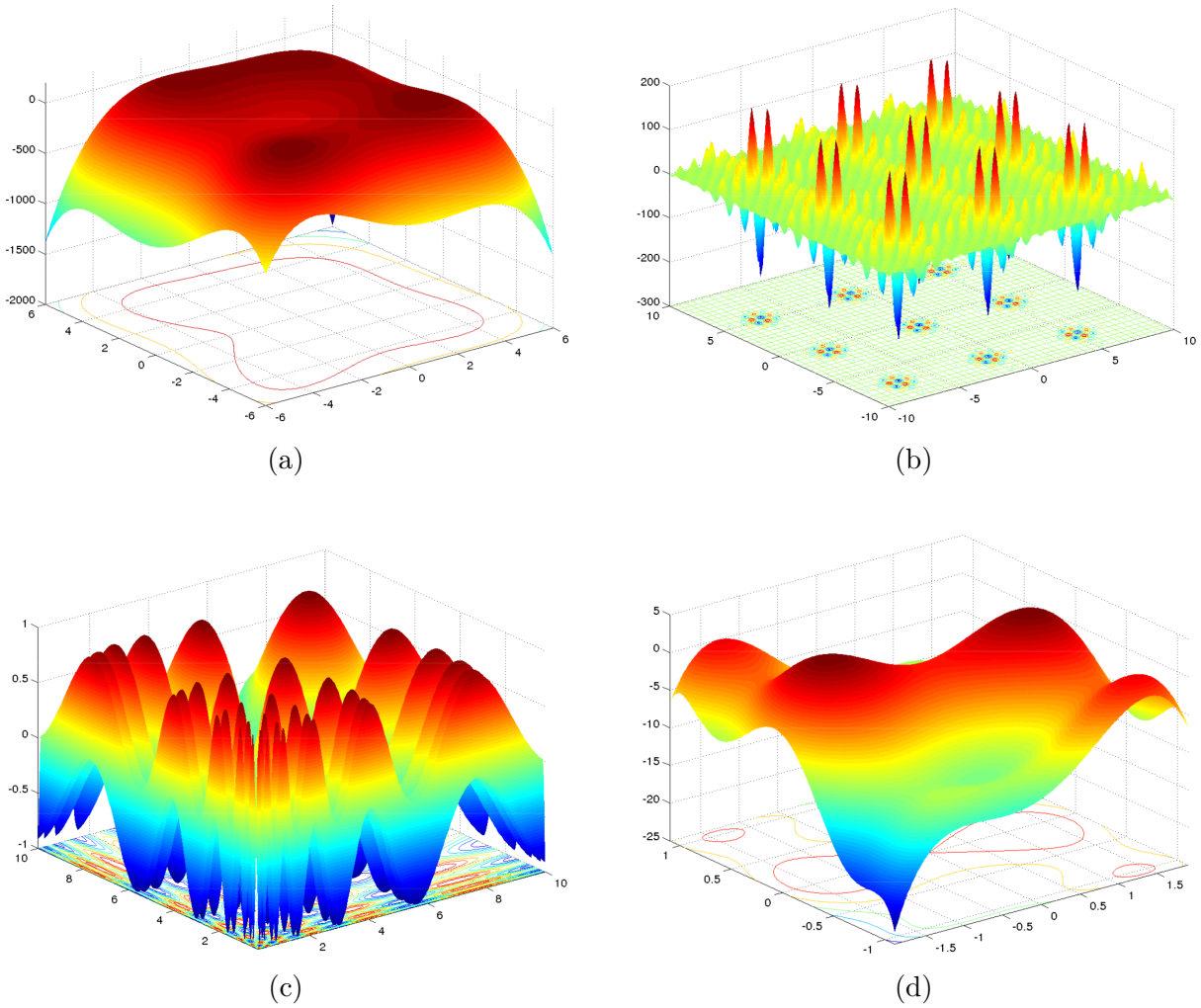
Problème de minimisation multimodal

Définition 20. Soit υ *optima* locaux $x_1^*, ..., x_\upsilon^*$ appartenant à un espace de recherche $\mathcal{S}$, une fonction objectif f et $h \in \mathbb{R}$ un critère de qualité, si l'ordre des *optima* est $f(x_1^*) \leq ... \leq f(x_l^*) \leq h \leq f(x_{l+1}^*) \leq ... \leq f(x_\upsilon^*)$, un problème de minimisation multimodal consiste à approximer l'ensemble $\mathcal{O} = \bigcup_{i=1}^{l} x_i^*$.
La variable h correspond au seuil permettant d'exclure certains *optima* : ceux ayant une valeur en dessous de h sont considérés comme étant des *optima* de qualité. Si $h = \infty$, l'objectif est d'identifier tous les *optima* locaux. Si $h = f(x_1^*)$, alors l'objectif est d'identifier uniquement le ou les *optima* globaux.

Des contraintes additionnelles peuvent être ajoutées à la définition 20. Soit $\tilde{A}$ l'ensemble d'approximation de l'ensemble $\mathcal{O}$ obtenu par un algorithme d'optimisation, la cardinalité de $\tilde{A}$ peut être utilisée comme contrainte, comme par exemple $|\tilde{A}| \leq k$. Si $k = 1$, on se ramène au problème d'optimisation global dans lequel une unique solution est recherchée. Une autre contrainte sur la diversité peut être formulée comme : $\forall x, y \in \tilde{A}, x \neq y, d(x, y) > \kappa$, signifiant que la distance entre deux solutions de $\tilde{A}$ ne doit pas être inférieure à un seuil noté κ.

La figure 5.1 montre des exemples de fonctions multimodales à maximiser, provenant de la suite de fonctions de référence la plus récente utilisée par la communauté d'EA pour évaluer les différentes techniques développées (X. Li, Andries Engelbrecht et Michael G Epitropakis 2013). La fonction Himmelblau inversée, en figure 5.1a, possède quatre *optima* globaux et il n'y a aucun *optimum* local. La fonction Shubert, en figure 5.1b, pré-

sente neuf paires de pics globaux regroupés, chaque paire est très proche l'une de l'autre, mais la distance entre les paires est beaucoup plus grande. La fonction Vincent en deux dimensions (2D), visible en figure 5.1c, présente un paysage de fitness composé de plusieurs pics globaux avec des largeurs de bassin très différentes. Et SixHump, en figure 5.1d, a deux *optima* globaux et deux *optima* locaux.



(a) (b)

(c) (d)

FIGURE 5.1 – Exemples de paysages de fitness avec plusieurs *optima* extraits de la suite de fonctions multimodales CEC'2013 (X. LI, Andries ENGELBRECHT et Michael G EPITROPAKIS 2013). (a) : la fonction Himmelblau inversée, (b) : la fonction Shubert inversée en 2D, (c) : la fonction Vincent inversée en 2D, et (d) : la fonction Six-Hump Camel Back.

5.1.2 Techniques employées

À l'instar de nombreux concepts de l'évolution artificielle, le *niching* est également un principe bio-inspiré. Les notions de niches et d'espèces font référence à la littérature sur les écosystèmes naturels. Dans ce contexte, une **niche** est définie comme un sous-espace de l'environnement dans lequel une espèce est une classe ou un type d'individus capable d'évoluer. Dans le contexte d'EA, (HORN 1997) introduit les niches comme des divisions

d'un environnement, le paysage de fitness, tandis que les espèces sont des divisions de la population en **sous-populations**.

Les études sur les méthodes de *niching* remontent aux prémisses du développement des algorithmes d'EA. Elles sont d'abord développées pour promouvoir la diversité de la population afin d'améliorer les capacités de recherche globale dans le but d'éviter la convergence prématurée et d'augmenter la probabilité de localiser l'unique *optimum* global, comme c'est le cas dans (Cavicchio 1970 ; Holland 1975 ; De Jong 1975). Toutefois, il s'est avéré que ces techniques de préservation de la diversité pouvaient également être utilisées pour localiser des solutions optimales multiples au cours d'une unique exécution d'un algorithme. Nous présentons trois méthodes classiques de *niching* qui ont eu une influence significative sur le développement des méthodes plus récentes.

Sharing. La technique de *fitness sharing* (Goldberg, Richardson et al. 1987) vise à maintenir une population diversifiée en dégradant la valeur de fitness d'un individu en fonction de la présence d'autres individus voisins. Un individu est considéré dans le voisinage d'un autre, si la distance les séparant est inférieure à un seuil, noté σ_{share}. Au cours de l'étape de sélection, si de nombreux individus sont dans un même voisinage, leur valeur de fitness est dégradée. Ce mécanisme provoque la diminution du nombre d'individus occupant le même voisinage à la génération suivante. L'idée est de permettre de récompenser ou plutôt de ne pas punir les individus qui explorent d'autres zones de l'espace de recherche peu denses en individus. La valeur de *sharing* d'un individu x est calculée selon l'expression suivante :

$$\tilde{f}(x) = \frac{f(x)}{\sum_{y \in P} sh(d(x,y))} \tag{5.1}$$

avec

$$sh(w) = \begin{cases} 1 - (\frac{w}{\sigma_{share}})^{\alpha} & \text{si } w < \sigma_{share} \\ 0 & \text{sinon.} \end{cases} \tag{5.2}$$

où $sh : \mathbb{R}+ \rightarrow [0,1]$ est la fonction de *sharing*, $d(x,y)$ est la distance entre les deux individus x et y de la population $\mathcal{P}$, α une constante à déterminer (égale à 1 la plupart du temps) et σ_{share} le seuil de distance à ne pas dépasser pour faire partie d'un même voisinage. Dans le processus évolutionnaire, la technique de *sharing* intervient juste après l'évaluation de la fitness et avant l'opérateur de sélection. C'est aussi le cas de la procédure de *clearing* introduite ci-après.

Clearing. Comme la méthode de *sharing*, la procédure de *clearing* (Petrowski 1996) utilise une mesure de dissimilarité entre les individus pour déterminer s'ils sont dans le même voisinage. La valeur de cette dissimilarité peut être une distance comme la distance de Hamming pour les génotypes binaires, la distance euclidienne pour les génotypes à valeurs réelles, ou elle peut être définie au niveau du phénotype. Les individus d'un même voisinage forment ainsi une sous-population.

Cette technique est illustrée en figure 5.2. Un individu appartient à une sous-population lorsque sa distance avec un individu est inférieure à un seuil appelé le rayon de *clearing* (σ_{clear}). Chaque sous-population est représentée par un cercle de rayon σ_{clear} et est constituée des individus présents dans un même cercle. L'idée est de préserver la fitness de l'individu dominant et de remettre à 0 la fitness de tous les autres individus de la même

sous-population. Ainsi, la procédure attribue la totalité des ressources d'une même sous-population à un seul individu. Cet individu est l'individu dominant, c'est celui qui a la meilleure valeur de fitness (en rouge dans l'illustration). Plutôt que de partager ses ressources, comme dans la technique du *sharing*, il prend tout : c'est la stratégie du« *winner takes all* ».

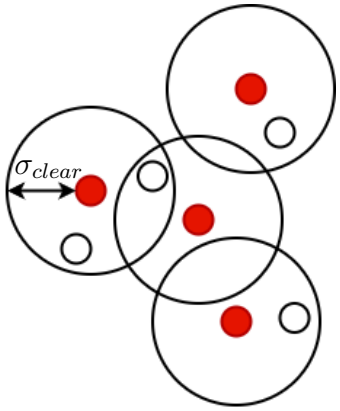


FIGURE 5.2 – Illustration de la technique de *clearing* simplifiée dans laquelle il n'y a qu'un seul individu dominant (rouge) par voisinage (cercle).

.

Il est intéressant de noter que cette procédure peut être généralisée en acceptant plusieurs individus dominants par sous-population. Dès lors que le nombre d'individus dominants est égal à la taille de la population, la méthode de recherche se ramène à une métaheuristique standard.

Crowding. La méthode de *crowding* (DE JONG 1975) repose sur un mécanisme de compétition entre un individu et une partie de la population de ses parents. L'idée est de jouer sur la pression de sélection en favorisant les individus éloignés les uns des autres et possédant une bonne valeur de fitness. Un individu nouvellement généré est comparé à un petit échantillon aléatoire prélevé dans la population actuelle (celle dont proviennent les parents du nouvel individu), et l'individu le plus similaire de l'échantillon est remplacé. Le facteur de *crowding*, *crowding factor* noté CF, est généralement utilisé pour déterminer la taille de cet échantillon. (MAHFOUD et al. 1992) propose une version revisitée appelée *deterministic crowding* qui est capable de localiser et de maintenir de nombreux *optima* sans avoir besoin de connaître le nombre d'*optima* ou de définir un paramètre seuil (comme le rayon σ dans la procédure de *clearing* ou de *sharing*). Elle est présentée sous forme de pseudo-code dans l'algorithme 12.

(MENGSHOEL et GOLDBERG 1999) propose une technique de *crowding* probabiliste dans laquelle l'étape de remplacement est aléatoire.

Il existe de nombreuses autres méthodes de *niching* comme celle de *clustering* (YIN et GERMAY 1993) ou de *speciation* (J.-P. LI et al. 2002). Elles peuvent être intégrées à des algorithmes d'EA ou combinées à des méthodes de recherche locale (X. LI, Michael G. EPITROPAKIS et al. 2017). Selon (STOEAN et al. 2010), ces stratégies sont classifiées en deux groupes : celles basées sur un rayon (*radius-based*) et celles qui ne le sont pas (*non-radius-based*). Comme il existe de nombreuses techniques de *niching*, il est nécessaire de

Algorithm 12 Pseudo-code du *deterministic crowding* de (Mahfoud et al. 1992).

Sélectionner deux parents p_1, p_2 aléatoirement (sans remplacement)
Générer deux enfants c_1 et c_2 par croisement et mutation
if $d(p_1, c_1) + d(p_2, c_2) \leq d(p_1, c_2) + d(p_2, c_1)$ **then**
    if $f(c_1) < f(p_1)$ **then**
        Remplacer p_1 avec c_1
    end if
    if $f(c_2) < f(p_2)$ **then**
        Remplacer p_2 avec c_2
    end if
else
    if $f(c_2) < f(p_1)$ **then**
        Remplacer p_1 avec c_2
    end if
    if $f(c_1) < f(p_2)$ **then**
        Remplacer p_2 avec c_1
    end if
end if

pouvoir les comparer. Il existe plusieurs mesures de comparaisons que nous présentons dans ce qui suit.

5.1.3 Mesures de performance

Les mesures de performance peuvent être divisées en deux groupes : celles qui supposent que le nombre et la position des *optima* à identifier sont des informations connues *a priori* et les autres. Les mesures du premier groupe sont plus nombreuses et plus utilisées, car elles servent notamment à l'évaluation des performances d'un algorithme sur les fonctions de référence de (X. Li, Andries Engelbrecht et Michael G Epitropakis 2013).

Nous commençons par introduire les mesures les plus connues du premier groupe, celles qui nécessitent la connaissance des *optima*.

5.1.3.1 Connaissance des *optima*

Succes rate. Le *succes rate* (SR) mesure le pourcentage d'exécutions dans lesquelles tous les *optima* globaux ont été localisés (à ϵ près dans le cas continu).

$$SR = \frac{\sum_{i=1}^{N} T^i}{N} \tag{5.3}$$

où T^i vaut 1 si tous les *optima* de $\mathcal{O}$ ont été identifiés au cours de la i-ème exécution et 0 sinon, et N est le nombre total d'exécutions.

Peak ratio. Le *peak ratio* (PR) mesure le pourcentage moyen des *optima* globaux identifiés par un algorithme sur plusieurs exécutions. Étant donné un niveau de précision fixé

ϵ, le calcul de PR pour un algorithme donné est :

$$PR = \frac{\sum_{i=1}^{N} |\tilde{A}^i|}{|\mathcal{O}| \times N} \tag{5.4}$$

où $\tilde{A}^i$ est l'ensemble des *optima* globaux trouvés, à ϵ près, à la fin de la i-ème exécution de l'algorithme, $\mathcal{O}$ est l'ensemble des *optima* globaux connus et N est le nombre total d'exécutions.

F1. Le score F1, mesure inspirée du domaine de la recherche d'information (*information retrieval*), ne s'intéresse pas seulement à la capacité d'un algorithme à trouver tous les *optima* globaux, mais il mesure aussi la précision (*precision*) et le rappel (*recall*) :

- la précision mesure la fraction du nombre d'*optima* globaux identifiés par un algorithme sur le nombre total de solutions potentielles identifiées par l'algorithme (le nombre d'individus dans la population finale $\mathcal{P}$ de l'algorithme, par exemple) : $precision = \frac{|\tilde{A}|}{|\mathcal{P}|}$;
- le rappel mesure la fraction du nombre d'*optima* globaux qu'un algorithme identifie sur le nombre d'*optima* globaux à identifier (c'est la mesure du PR de l' équation (5.4) pour une exécution) : $recall = \frac{|\tilde{A}|}{|\mathcal{O}|}$.

Le F1 score d'un algorithme pour une exécution est ainsi calculé :

$$F1 = 2 \times \frac{precision \times recall}{precision + recall} = \frac{2 \times |\tilde{A}|}{|\mathcal{O}| + |\mathcal{P}|} \tag{5.5}$$

Cette mesure quantifie à la fois la qualité des objectifs ainsi que leur diversité.

Il existe une mesure dynamique du score F1 qui s'intéresse à la rapidité avec laquelle un algorithme identifie les *optima* globaux souhaités (Preuss, M. Epitropakis et al. 2021). La mesure du *Chi-square-like performance statistic* (Deb et Goldberg 1989) vise à calculer l'écart type de la distribution des génotypes des individus de la population finale et de la distribution moyenne des *optima* globaux à identifier.

5.1.3.2 *Optima* inconnus

Les mesures présentées ci-dessus supposent que le nombre et la position des *optima* globaux sont connus *a priori* signifiant que l'ensemble $\mathcal{O}$ est connu. En pratique, il est très improbable que cette hypothèse soit valide.

Une mesure de performance qui ne repose pas sur cette hypothèse a été proposée dans (Kronfeld et Zell 2010). À la place, ce calcul de score requiert la définition d'un intervalle de seuil $[\theta_l, \theta_u]$. Cet intervalle est spécifique au domaine d'application et recouvre toutes les valeurs de score de fitness considérées comme intéressantes. Ces valeurs peuvent être déterminées *a priori* par un expert du problème en question, par exemple. θ_l est le point idéal, c'est-à-dire la valeur de fitness idéale ou atteignable idéalement, tandis que θ_u est une limite supérieure en dessous (resp. au-dessus) de laquelle les valeurs de fitness sont jugées satisfaisantes (resp. insatisfaisantes). Cette mesure de performance est évaluée sur la population finale d'un algorithme $\mathcal{P}$ et calculée comme :

$$sc(\mathcal{P}, \theta_l, \theta_u) = \sum_{\{x_i \in \mathcal{P} | f(x_i) < \theta_l\}} \frac{\theta_u - f(x_i)}{\theta_u - \theta_l} \tag{5.6}$$

L'inconvénient de cette mesure est lié à un problème de redondance lorsqu'on compare le score de deux populations finales $\mathcal{P}_1$ et $\mathcal{P}_2$. Dans le cas de figure où tous les individus de $\mathcal{P}_1$ sont regroupés autour d'un même *optimum*, tandis que les individus de $\mathcal{P}_2$ identifient plusieurs *optima*, le score de $\mathcal{P}_1$ peut être plus élevé que celui de $\mathcal{P}_2$. Ce qui est contradictoire avec l'idée selon laquelle l'algorithme générant $\mathcal{P}_2$ devrait être favorisé. En effet, $\mathcal{P}_2$ contient un plus grand nombre d'*optima* différents, alors que dans $\mathcal{P}_1$ il n'y a qu'un seul *optimum* trouvé.

Une mesure alternative, aussi introduite dans (Kronfeld et Zell 2010), propose d'utiliser un algorithme de regroupement basé sur la densité (*density-based clustering*) avec un paramètre σ pour supprimer la redondance entre les individus regroupés autour du même *optimum* local. Cette mesure est calculée selon l'équation (5.7) :

$$sc(\mathcal{P}, \theta_l, \theta_u) = \sum_{B_j \in Bin_k(clust_\sigma(\mathcal{P}), \theta_l, \theta_u)} w_j |B_j| \tag{5.7}$$

Chaque *cluster* identifié par l'algorithme de regroupement $clust_\sigma$ est appliqué aux individus de la population $\mathcal{P}$ ayant une valeur de fitness comprise entre θ_l et θ_u. En général, le nombre d'*optima* de qualité inférieure (proche de θ_u) est plus important que le nombre d'*optima* de qualité supérieure (proche de θ_l). Si cette distribution est connue à l'avance, les poids peuvent être adaptés en conséquence. Sinon, un *equidistant binning* noté Bin_k est appliqué où $w_j = \frac{k-j+1}{k}$ pour $1 \leq j \leq k$. La contribution de chaque *bin* B_j est pondérée par un facteur w_j pour rendre compte de la qualité des *optima* trouvés. Ainsi, toute solution dans B_1, qui est la meilleure *bin*, a une valeur de 1, tandis que chaque solution dans la pire *bin* (B_k) contribue à hauteur de $\frac{1}{k}$ au score.

Dans l'équation (5.6), plus l'intervalle $[\theta_l, \theta_u]$ est large, plus les individus dont la valeur de fitness est proche de θ_u sont ignorés. En effet, dans ce cas, la valeur du dénominateur augmente et celle du numérateur diminue. C'est moins le cas dans l'équation (5.7) car leur contribution reste proportionnelle à k. Cette modification permet de respecter l'idée selon laquelle même les *optima* proches de θ_u doivent être considérés comme intéressants et contribuer au score final.

Cette introduction sur l'optimisation multimodale peut être approfondie à l'aide d'ouvrages tels que (Das, Maity et al. 2011 ; Preuss 2015 ; X. Li, Michael G. Epitropakis et al. 2017 ; Preuss, M. Epitropakis et al. 2021).

5.2 Investigation de deux stratégies de *niching*

La plupart des algorithmes d'EA utilisent une seule population d'individus. On parle alors de **panmixie**. Un algorithme panmictique applique des opérateurs génétiques sur l'ensemble de cette population. Par contraste, il existe également des méthodes d'EA dans lesquelles l'ensemble des individus est décomposé en plusieurs sous-populations, ou que la population est structurée, comme avec les algorithmes distribués. Ces méthodes reposent sur l'idée que l'isolement de ces sous-populations ou sous-structures permet d'obtenir une plus grande diversité génétique (S. Wright 1943). Dans de nombreux cas,

investigués par (Alba et José M Troya 2002), ces algorithmes fournissent un meilleur échantillonnage de l'espace de recherche et améliorent ainsi la qualité des solutions pour un algorithme panmictique équivalent. Elles permettent surtout d'identifier une diversité de solutions au cours d'une unique exécution. Parmi l'ensemble des méthodes existantes, nous en présentons deux en détail.

5.2.1 RS-CMSA-ESII

(Ali Ahrari, Deb et Preuss 2017) proposent l'algorithme *covariance matrix self-adaptation evolution strategy with repelling subpopulations* (RS-CMSA-ES) qui a été désigné comme la méthode de *niching* la plus performante pour la suite de tests MMO de CEC'2013 (X. Li, Andries Engelbrecht et Michael G Epitropakis 2013). La stratégie de *niching* principale de RS-CMSA-ES est de faire évoluer plusieurs sous-populations en parallèle pour que chacune d'elles identifie un *optimum*. Plus précisément, la taille de la population initiale est divisée en N_s sous-populations de taille λ qui explorent l'espace de recherche en parallèle. Chaque sous-population a ses propres paramètres de mutation qui correspondent à la taille de pas de la distribution (σ_{mean_i}), la matrice de covariance associée (C_i), son centre (x_{mean_i}) et le meilleur individu de la sous-population (x_{best_i}).

L'algorithme combine plusieurs concepts et techniques existants, dont voici les principaux :

- une archive est utilisée pour stocker les *optima* identifiés ;
- CMSA-ES (Beyer et Sendhoff 2008) est utilisée comme processus d'optimisation pour chacune des sous-populations. CMSA-ES est une version de CMA-ES qui nécessite moins d'hyperparamètres à configurer, ce qui permet de diminuer la complexité du processus d'adaptation pour des performances équivalentes ;
- la stratégie de *restart* (Auger et N. Hansen 2005) consiste à stopper le processus d'évolution d'une sous-population lorsqu'un critère d'arrêt est atteint. Un exemple de critère d'arrêt utilisé est la stagnation : la valeur médiane des κ_1 nouveaux individus évalués ne doit pas être plus petite que la valeur médiane des κ_2 derniers. κ_1 et κ_2 sont deux hyperparamètres à définir. Ensuite, la sous-population est redémarrée dans une autre zone de l'espace de recherche. Le calcul de son nouveau point initial est réalisé en fonction de la dernière position initiale de la sous-population, de sa dernière position finale (avant d'être stoppée) et de la position des autres sous-populations. La taille de la sous-population est aussi augmentée par un certain facteur (généralement 2 ou 3) ;
- une initialisation biaisée, introduite dans (A. Ahrari, Shariat-Panahi et Atai 2009), vise à initialiser une sous-population à la position qui maximise la distance avec les autres sous-populations et les *optima* stockés dans l'archive ;
- un point tabou est un individu à partir duquel les individus d'une sous-population donnée doivent maintenir une distance suffisante. L'ensemble des points tabous est composé des centres des sous-populations les plus adaptées et des *optima* stockés dans l'archive. Une région taboue est soit une sous-population centrée en un point tabou, soit une niche précédemment identifiée dont le centre est un *optimum* de l'archive. Une illustration de ces concepts est proposée en figure 5.3. Dans la figure 5.3 (a), la sous-population $\mathcal{P}_1$ représentée par la distribution normale centrée

en x_{mean_1} et de déviation standard σ_{mean_1} perçoit deux régions tabous formées par les points tabous y_1 et y_2 qui sont des régions déjà identifiées. Dans la figure 5.3 (b), la sous-population $\mathcal{P}_2$ représentée par la distribution normale centrée en x_{mean_2} perçoit trois régions taboues : les régions identifiées par y_1 et y_2 (similairement à la sous-figure (a)), mais aussi la sous-population $\mathcal{P}_1$ qui est une sous-population dont x_{mean_1}, le point tabou, possède une meilleure valeur de fitness ;

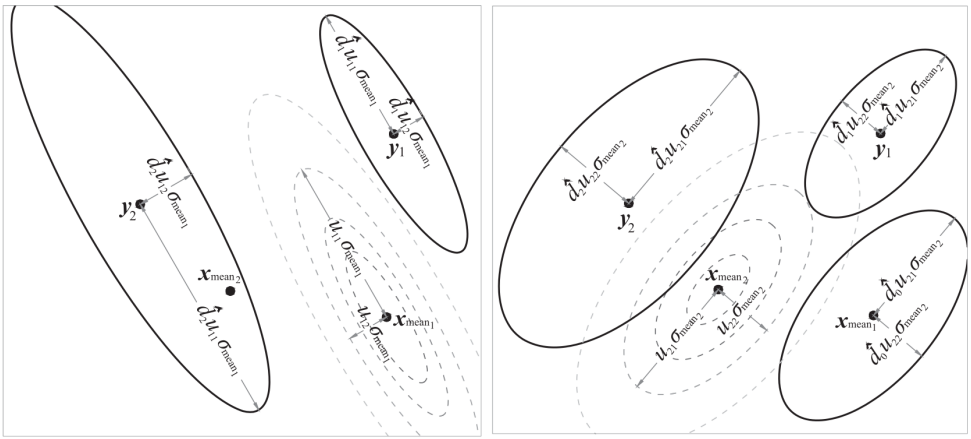


FIGURE 5.3 – Illustrations de régions taboues, extraites de (Ali AHRARI, DEB et PREUSS 2017).

— la distance de Mahalanobis normalisée est préférée à la distance euclidienne, car elle permet de mieux estimer la forme des régions tabous.

— le test de *Hill-Valley* (URSEM 1999) est utilisé pour déterminer si deux individus appartiennent à la même niche. Ici, la niche est une région spéciale de l'espace de recherche formant un creux. Le test est présenté dans le pseudo-code de l'algorithme 13 et est illustré en figure 5.4. L'idée est d'échantillonner N_t points de l'espace de recherche entre les deux individus considérés : si un des points a une valeur de fitness supérieure à celle des deux individus, alors ils ne font pas partie de la même niche. La figure illustre l'échec du test ;

Algorithm 13 Pseudo-code du test de *Hill-Valley* de (URSEM 1999).

Input: deux vecteurs de décision x et y, un nombre de points de test N_t, une fonction objectif f

Output: un booléen répondant à la question : x et y appartiennent à la même niche ?

1: **for** $k = 1, ..., N_t$ **do**
2: $x^{test} = y + \frac{k}{N_t+1}(x - y)$
3: **if** $max(f(x), f(y)) < f(x_{test})$ **then**
4: **return** Faux
5: **end if**
6: **end for**
7: **return** Vrai

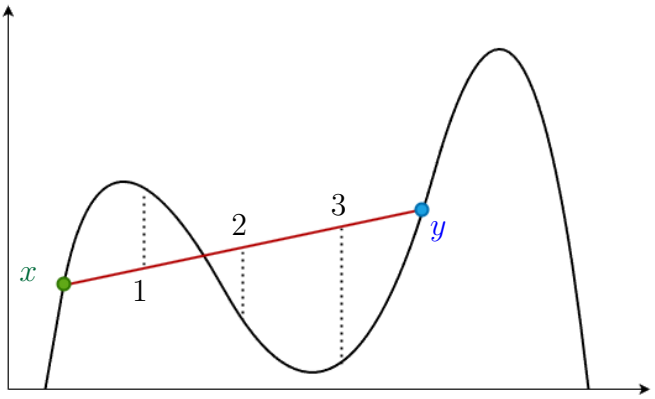


FIGURE 5.4 – Illustration du test de *Hill-Valley* sur une fonction à une dimension, inspirée de (DELLA CIOPPA, MARCELLI et NAPOLI 2011). Les chiffres font référence aux $N_t = 3$ différents x^{test} de l'algorithme 13. Le test renvoie Faux : x et y ne font pas partie de la même niche.

Dans RS-CMSA-ES, la diversité est préservée en forçant les membres de chaque sous-population à maintenir une distance suffisante par rapport aux points tabous et aux *optima* déjà identifiés. L'évolution de ces sous-populations suit le processus de CMSA-ES avec une sélection élitiste. Lorsque l'évolution d'une sous-population termine, soit parce qu'elle a convergé, stagné ou divergé, le meilleur individu est comparé *via* le test de *Hill-Valley* avec les *optima* présents dans l'archive. Les nouveaux *optima* sont ajoutés à cette archive. Puis l'évolution des sous-populations est redémarrée *via* la stratégie *restart* : une taille de population plus importante leur est attribuée et elles sont réinitialisées à une distance maximale les unes des autres. Ce processus est itéré jusqu'à ce que le budget d'évaluation soit atteint.

La nouvelle variante RS-CMSA-ESII (Ali AHRARI, ELSAYED et al. 2022) garde les mêmes éléments introduits précédemment et propose notamment :

— d'introduire de nouveaux critères de terminaison pour stopper l'évolution des sous-populations qui ont peu de probabilité de converger vers un *optimum*. Ce qui permet ainsi de sauvegarder du budget ;

— de modifier l'étape d'initialisation d'une sous-population pour augmenter la rapidité de temps de la stratégie *restart*,

— d'ajouter une mesure plus précise pour la détermination des régions taboues critiques basée sur les propriétés de la distance de Mahalanobis.

RS-CMSA-ESII a été comparé aux méthodes d'optimisation multimodale les plus performantes sur la suite de tests de CEC'2013. La comparaison des résultats a montré la supériorité de RS-CMSAESII sur les méthodes existantes, y compris son ancienne variante RS-CMSA-ES et HillValleyEA (MAREE et al. 2018), un autre algorithme de référence de la communauté MMO.

5.2.2 Algorithmes cellulaires génétiques

Les algorithmes cellulaires génétiques (*cellular genetic algorithms*, cGAs) sont des méthodes d'optimisation bien connues pour traiter des problèmes d'optimisation multimodale

et des problèmes d'épistasie (problèmes dans lesquels les variables de décision dépendent les unes des autres) (Alba et José Ma Troya 2000 ; Alba et Bernabé Dorronsoro 2008). Il s'agit d'une sous-classe des algorithmes génétiques dans laquelle la population est structurée selon une topologie spécifique, permettant aux individus d'interagir uniquement avec leurs voisins selon cette topologie. La structure topologique définit un graphe connecté où un sommet représente un individu et une arête représente la possibilité d'interaction entre deux individus : dans ce graphe, chaque individu ne peut générer un individu enfant qu'avec ses voisins. La figure 5.5 présente une population toroïdale d'un algorithme cellulaire génétique de 36 individus placés sur une grille.

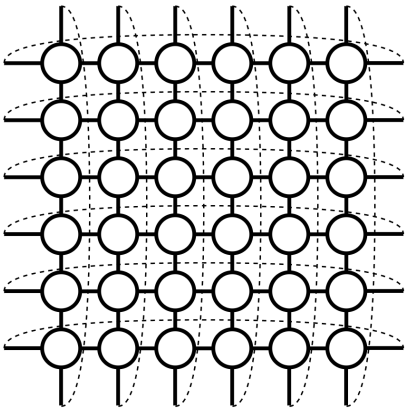


Figure 5.5 – Illustration d'une population toroïdale d'un algorithme cellulaire génétique de 36 individus placés sur une grille, inspirée de (Alba et Bernabé Dorronsoro 2008).

Le voisinage est une fonction qui, pour chaque individu x d'une population $\mathcal{P}$ renvoie un sous-ensemble de la population $V(x) \subset \mathcal{P}$. En général, la population est organisée avec une structure en maillage. Le voisinage d'un individu est alors défini en utilisant un motif sur ce maillage et est composé de tous les individus à l'intérieur du motif choisi centré sur l'individu. Le nom donné à un voisinage suit la notation classique : Ln (L pour linéaire) pour les voisinages composés des n plus proches voisins dans chacune des directions axiales données (nord, sud, ouest et est) tandis que le label Cn (C pour compact) désigne les voisinages contenant les $n-1$ individus les plus proches de l'individu considéré dans les directions horizontale, verticale et diagonale.

Le chevauchement des voisinages fournit un mécanisme indirect de propagation des solutions. Avec un voisinage plus petit, les meilleures solutions se répandent plus lentement dans l'ensemble de la population, car il y a plus d'étapes intermédiaires entre deux individus éloignés sur la grille. La diversité de la population est ainsi préservée plus longtemps que dans le cas des GAs ou des cGAs avec une taille de voisinage plus importante. La définition de la taille de voisinage est ainsi un des facteurs clés du compromis entre exploration et exploitation. La figure 5.6 montre le degré de chevauchement de deux voisinages différents pour les mêmes individus (en bleu et en rouge). Sur la partie gauche, on peut voir le voisinage L5, tandis que sur la partie droite, c'est le voisinage C9 qui est représenté. Les individus sur fond violet appartiennent aux deux voisinages, tandis que les individus sur les deux autres fonds de couleur n'appartiennent qu'à un seul voisinage.

Les voisinages qui se chevauchent aident donc à explorer l'espace de recherche de par la lente diffusion des solutions à travers la grille de la population. (Alba et Bernabé

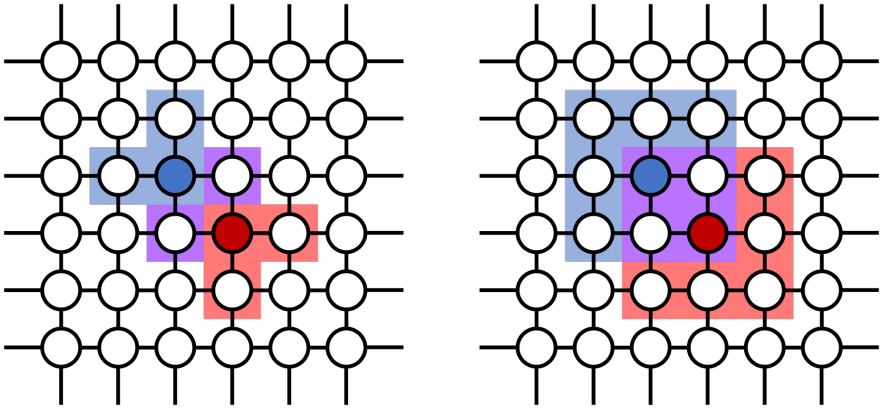


FIGURE 5.6 – Illustration de la structure de voisinage L5 (à gauche) et C9 (à droite), inspirée de (ALBA et Bernabé DORRONSORO 2008).

DORRONSORO 2004 ; ALBA et Bernabé DORRONSORO 2008) montrent que ce chevauchement motive l'exploration et préserve la diversité. Cela conduit ainsi les cGAs à trouver plusieurs *optima*. Mais cela se fait souvent au détriment d'une convergence plus lente que les GAs. Le choix de la topologie de la population et du voisinage sont deux paramètres qui guident la recherche et contrôlent la vitesse de diffusion des solutions le long de la grille.

Plus formellement, il existe des mesures permettant de faire varier ces paramètres. Le *radius* introduit dans (ALBA et José Ma TROYA 2000) oriente la force de dispersion en fonction du voisinage choisi : plus le *radius* est élevé, plus le voisinage est étendu, et donc plus une bonne solution atteindra facilement d'autres individus de la population, car il y aura moins d'individus intermédiaires pour atteindre l'individu le plus éloigné. Formellement, le *radius* est calculé comme étant égal à la dispersion des n^* points de la structure de voisinage centrée en $(\overline{a}, \overline{b})$:

$$radius = \sqrt{\frac{\sum_{i=1}^{n^*}(a_i - \overline{a})^2 + \sum_{i=1}^{n^*}(b_i - \overline{b})^2}{n^*}}, \overline{a} = \frac{\sum_{i=1}^{n^*} a_i}{n^*}, \overline{b} = \frac{\sum_{i=1}^{n^*} b_i}{n^*} \tag{5.8}$$

où a_i (resp. b_i) est la coordonnée discrète en abscisse (resp. ordonnée) du i-ème individu sur la grille. Sur la figure 5.7, à gauche, la structure de voisinage choisie en exemple est L5 sur une grille de 4×4. Il y a $n^* = 5$ individus et $\overline{a}$ (resp. $\overline{b}$) est égal à la somme des valeurs en abscisse (resp. ordonnée) des individus : $\sum_{i=1}^{n^*} 1 + 3 \times 2 + 3 = 2$. Si on calcule $\sum_{i=1}^{n^*}(a_i - \overline{a})^2 = (1-2)^2 + 3 \times (2-2)^2 + (3-2)^2$, on obtient 2. Et il en est de même pour $\sum_{i=1}^{n^*}(b_i - \overline{b})^2$. On obtient donc $radius_{L5} = \sqrt{\frac{2+2}{5}} \approxeq 0,8944$.

Le radius peut aussi être appliqué à la grille : on considère la structure de voisinage équivalente à la structure de la grille. Si on considère la grille 4×4 au centre de la figure 5.7, le radius de la grille $radius_{4\times4}$ est égal au radius du voisinage C16. À ce propos, (SARMA et Kenneth DE JONG 1996) introduit la mesure appelée *ratio* définie comme le résultat de la fraction du *radius* du voisinage sur le *radius* de la grille :

$$ratio = \frac{radius_{\text{voisinage}}}{radius_{\text{grille}}} \tag{5.9}$$

Le ratio permet de contrôler l'équilibre entre l'exploration et l'exploitation : la réduction du ratio favorise l'exploration. Visuellement, on le comprend en observant la différence

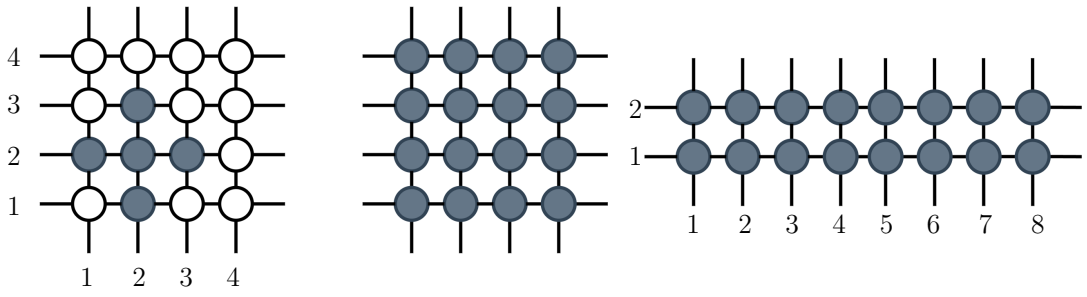


FIGURE 5.7 – À gauche : la structure de voisinage L5 (individus en gris) sur une grille 4×4 et le calcul du *radius*, noté $radius_{L5}$. Au centre : une grille 4×4 d'individus. À droite : une grille 2×8 d'individus.

entre la grille 4×4 et la grille 2×8 (à droite de la figure 5.7). En effet, la distance Manhattan maximale qui sépare deux individus est plus grande dans le cas de la grille 2×8. On note que dans cet exemple, les grilles sont de structure différente et ont donc un *radius* différent. Cela implique aussi qu'elles ont un *ratio* différent. Le choix de la structure de voisinage est donc primordial lors de la conception d'un cGA pour un problème donné.

Un autre aspect des algorithmes cellulaires est la possibilité d'appliquer l'étape de reproduction de deux manières différentes. D'une part, si l'étape est appliquée à tous les individus simultanément, la politique de mise à jour est dite synchrone, car les individus de la population de la génération suivante sont créés en même temps, de manière concomitante. D'autre part, si cette mise à jour des nouveaux individus est faite de manière séquentielle selon une politique d'ordre particulière, on parle de mise à jour asynchrone. Il existe de nombreuses politiques d'ordre (OLARIU et ZOMAYA 2005) qui peuvent être déterministes comme *Line Sweep* dont la mise à jour séquentielle est réalisée ligne par ligne ou stochastiques comme *Uniform Choice* (MCINNES, HEALY et MELVILLE 2020) qui choisit aléatoirement le prochain individu à remplacer. Dans ce manuscrit, nous faisons le choix de ne considérer que des mises à jour synchrones.

La prochaine section propose une étude expérimentale mettant en jeu les deux algorithmes présentés. Le but est d'observer quelle stratégie de préservation de la diversité est la plus adéquate à notre problème.

5.3 Étude expérimentale

Dans cette section, nous présentons les expérimentations réalisées. L'objectif est de comparer les performances de RS-CMSA-ESII (présenté dans la section 5.2.1) avec celles des algorithmes cellulaires génétiques (section 5.2.2) pour exhiber la méthode la mieux adaptée à notre tâche multimodale : identifier plusieurs paramétrisations des RRGHs étudiés. Différentes valeurs de structure de voisinage et de ratio pour les cGAs ont été testées. Les résultats obtenus sont comparés avec des métaheuristiques standards (panmictiques) sur les trois réseaux RRGHs afin d'évaluer la pertinence des différents mécanismes de diversité. Le contenu de cette section peut être retrouvé en grande partie dans (MICHELUCCI, CALLEGARI et al. 2024).

5.3.1 Expérimentations

L'étude expérimentale porte sur la comparaison entre GA, CMA-ES, six cGAs (de mise à jour synchrone) avec différents ratios et structures de voisinage, et RS-CMSA-ESII. Ces techniques sont évaluées sur les trois modèles hybrides de GRN. Les différentes implémentations des métaheuristiques panmictiques proviennent de la librairie `pymoo` (Blank et Deb 2020), et chacun des hyperparamètres choisis est identique à ceux détaillés dans la section 3.3.2. La taille de chaque population est également de 500. Comme nous souhaitons observer l'influence des hyperparamètres des cGAs pour trouver ceux qui conviennent le mieux à la résolution des différents RRGHs, plusieurs ensembles d'hyperparamètres ont été testés : ils sont énumérés dans le tableau 5.1.

nom	taille de la population	structure de voisinage	ratio
cGAL5	5x10	L5	0.279
cGAL9	10x10	L9	0.367
cGAL29	15x15	L29	0.719
cGAL41	21x21	L41	0.851
cGAL13	7x7x7	L13	0.607
cGAC9	7x7x7	C9	0.408

Tableau 5.1 – Description des hyperparamètres des cGAs testés.

La taille de la population et la structure du voisinage varient de manière à ce que nous puissions tester (i) des cGAs à faible ratio avec une petite taille de population et, inversement, (ii) des cGAs à ratio élevé avec une plus grande population, à la fois dans une grille carrée toroïdale de deux dimensions et (iii) une structure de voisinage en trois dimensions. Pour garantir des résultats équitables, leur implémentation est également basée sur l'implémentation GA standard fournie dans `pymoo`. La mise en œuvre de RS-CMSA-ESII est tirée de (Ali Ahrari, Elsayed et al. 2022) et les paramètres de contrôle auto-adaptatifs sont initialisés à leurs valeurs par défaut.

Chaque expérience est exécutée 50 fois afin d'obtenir des résultats statistiquement significatifs. Le critère de terminaison choisi est le nombre d'évaluations de fonctions (NFEs) : 100 000 pour 2G et 3G et 200 000 pour 5G. Ces valeurs ont été choisies empiriquement en fonction de la complexité relative et de la longueur du vecteur de décision.

5.3.2 Résultats

Pour chaque algorithme, chaque dimension du problème et à chaque génération, nous calculons la meilleure solution candidate obtenue jusqu'à présent, répétons les exécutions 50 fois et calculons la moyenne de la meilleure valeur de fitness (*mean best fitness*, MBF).

L'évolution monotone de tous les algorithmes est présentée dans la colonne de gauche de la figure 5.8. On peut notamment observer que :

1. cGAL13, cGAL29 et cGAL41 se distinguent des algorithmes testés, car, d'une part, leur convergence est plus lente et, d'autre part, même lorsque le budget maximal est atteint, leurs courbes montrent que le processus de recherche aurait pu poursuivre sa convergence ;

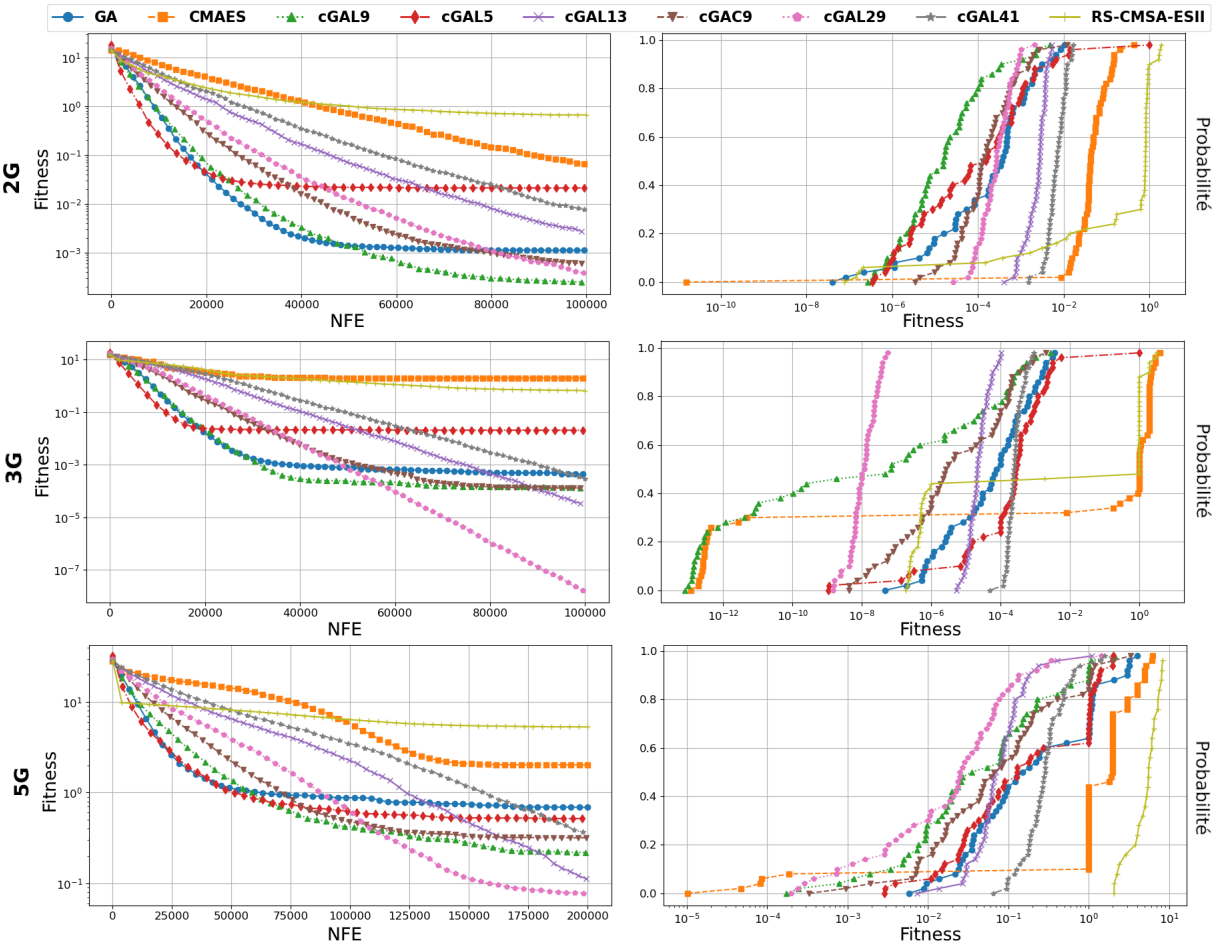


FIGURE 5.8 – Évolution monotone des valeurs MBF (à gauche) et des courbes de répartition des meilleurs résultats (à droite) pour les trois RRGHs.

2. comme prévu, les métaheuristiques panmictiques (GA et CMA-ES) sont moins performantes que les cGAs dans tous les cas. En effet, leurs courbes d'évolution moyenne atteignent une asymptote horizontale plus rapidement et elles obtiennent un score de fitness plus élevé à la fin de la convergence : le point le plus à droite de ces courbes est plus bas ;
3. et RS-CMSA-ESII et CMA-ES obtiennent en moyenne les moins bons résultats.

En outre, les courbes de la fonction de répartition sont construites à droite de la figure 5.8, pour chaque réseau considéré. Chaque courbe décrit la probabilité de trouver une solution à un niveau inférieur ou égal à un score de fitness donné. Par exemple, en 3G, la probabilité qu'un utilisateur obtienne une solution avec un score de fitness inférieur ou égal à 10^{-4} avec cGAL9 est de 80% pour 100 000 NFEs. D'après ces graphiques, on remarque que :

1. les cGAs ne trouvent pas souvent la meilleure solution. En effet, le point le plus à gauche des courbes de répartition appartient dans deux cas sur trois à CMA-ES. Cependant, dans chacun des cas, ils ont une plus haute probabilité d'obtenir une solution avec un score de fitness inférieur à 10^{-2} comparé aux autres algorithmes

testés. Pour rappel, 10^{-2} est considéré comme l'erreur de précision cohérente avec l'expertise biologique, introduite dans la section 3.3.2.1) ;

2. dans tous les cas, CMA-ES a une faible probabilité de fournir les meilleurs résultats sur certaines exécutions, mais il est peu performant dans la majorité des cas : par exemple, en 3G, il n'y a qu'environ 30% de probabilité d'obtenir un score inférieur à $10^{-}2$;
3. RS-CMSA-ESII a des performances mitigées sur les trois réseaux et ne trouve aucune solution satisfaisante en 5G.

Après avoir comparé la convergence et la qualité des solutions obtenues, nous avons besoin de comparer la diversité des solutions obtenues. Pour cela, on utilise la mesure de performance de l'équation (5.7). Elle ne requiert aucune connaissance sur le nombre et la localisation des *optima*, ce qui est un avantage, car nous ne les connaissons pas. Cette mesure suggère la sélection d'un intervalle seuil $[\theta_l, \theta_u]$. Dans notre cas, nous choisissons $\theta_l = 0$ et $\theta_u = 10^{-2}$. La mesure du score de performance utilise une méthode de regroupement basée sur la densité avec le paramètre σ pour supprimer la redondance entre les solutions candidates regroupées étroitement autour du même *optimum* local. Dans cette étude, DBSCAN (Ester et al. 1996) est paramétré avec $\sigma = 10^{-1}$ qui est la distance euclidienne maximale entre deux échantillons pour que l'un soit considéré comme étant dans le voisinage de l'autre. La méthode de l'*equidistant binning* est ensuite utilisée pour adapter les poids de distribution : l'accent est mis davantage sur les *optima* de qualité supérieure que sur les *optima* de qualité inférieure. Le nombre k est maintenu à 16 comme dans (Kronfeld et Zell 2010). Le tableau 5.2 présente les valeurs numériques des scores moyens, les résultats en gras mettant en évidence les meilleures performances pour chaque dimension du modèle. La comparaison indique que les cGAs à faible ratio (cGAL9 et cGAL5) sont à privilégier pour le réseau 2G, tandis que les cGAs à ratio plus élevé sont plus performants dans les cas des réseaux 3G et 5G, comme le montrent cGAL41 et cGAL29. Il convient également de noter que, dans le cas du réseau 5G, les valeurs de ratio extrêmes (cGAL5 et cGAL41) sont pénalisées parce qu'elles sont trop exploratoires ou à l'inverse trop exploitantes. cGAC9 a des résultats intéressants dans les trois cas, mais ne se démarque jamais.

Algorithmes	2G	3G	5G
GA	12.13	148.09	8.29
CMA-ES	3e-3	0.275	29.32
cGAL9	**19.99**	63.47	21.11
cGAL5	16.82	30.07	2.04
cGAL13	7.33	290.59	0.03
cGAC9	13.78	162.28	34.11
cGAL29	13.90	182.18	**49.81**
cGAL41	1.56	**311.07**	0.0
RSCMSAII	1e-2	0.275	0.0

Tableau 5.2 – Mesures de performance moyennées sur 50 exécutions.

Le tableau 5.3 résume les statistiques de la dernière population regroupée : elle contient uniquement les valeurs de fitness des meilleurs individus, inférieures à θ_u, rassemblées

autour de chaque *optimum* distinct trouvé par méthode de regroupement. Les meilleurs résultats (colonne par colonne) sont indiqués en gras. La moyenne et l'écart-type des résultats regroupés sont indiqués, ainsi que les scores minimaux de fitness (le point le plus à gauche de chaque courbe de répartition correspondante). Lors de l'examen d'une exécution particulière, il se peut qu'un algorithme ne trouve aucune solution en dessous de θ_u. Dans ce cas, la valeur maximale θ_u est prise en compte : il en résulte une moyenne normalisée, la valeur idéale étant θ_l, et θ_u la valeur seuil. On peut observer que cGAL9 trouve, en moyenne, des *optima* de meilleure qualité que les autres algorithmes en 2G et 5G. En 3G, cGAL29 identifie des solutions satisfaisantes avec un score de fitness inférieur en moyenne.

Algorithmes	2G		3G		5G	
	mean ± std	min	mean ± std	min	mean ± std	min
GA	1e-3 ± 2e-4	4e-8	4e-4 ± 2e-6	5e-8	99e-4 ± 94e-4	6e-3
CMA-ES	98e-4 ± 96e-4	**2e-11**	7e-3 ± 2e-13	1e-13	9e-3 ± 9e-3	**1e-5**
cGAL9	**8e-4 ± 8e-4**	3e-7	1e-4 ± 7e-6	**8e-14**	**85e-4 ± 7e-3**	2e-4
cGAL5	1e-3 ± 9e-4	3e-7	9e-4 ± 2e-4	1e-9	95e-4 ± 94e-4	3e-3
cGAL13	6e-3 ± 2e-3	4e-4	3e-4 ± 4e-4	5e-6	1e-2 ± 98e-4	7e-3
cGAC9	1e-3 ± 1e-3	3e-6	1e-4 ± 8e-6	4e-9	9e-3 ± 8e-3	3e-4
cGAL29	3e-3 ± 2e-3	3e-5	**9e-7 ± 1e-5**	1e-9	79e-4 ± 7e-3	2e-4
cGAL41	8e-3 ± 3e-3	2e-3	2e-3 ± 16e-4	5e-5	1e-2 ± 0	1e-2
RSCMSAII	8e-3 ± 1e-20	8e-8	7e-3 ± 2e-13	1e-13	1e-2 ± 0	1e-2

Tableau 5.3 – Résumé des valeurs de fitness après application de la méthode de regroupement sur 50 exécutions.

5.3.3 Analyse statistique

Une campagne de validation statistique a été menée pour évaluer les différences observées dans les valeurs de performance déclarées de toutes les paires d'algorithmes pour chaque RRGH différent. Nous considérons deux hypothèses nulles : H_0^P qui stipule que les scores de performance observés sont égaux, et H_0^F qui stipule que les scores de fitness moyens obtenus par regroupement sont similaires. Ces hypothèses nulles sont dupliquées pour chacune des dimensions des RRGHs considérées. Pour les tester, nous suivons la procédure proposée dans (Eftimov et Korošec 2021). Nous avons d'abord utilisé l'ANOVA de Friedman par rangs (*Friedman rank-sum test*) afin de déterminer si au moins deux méthodes présentent des différences significatives de résultats. Les valeurs pour les hypothèses nulles montrent, à un niveau de confiance de $\alpha = 5\%$, que les différences sont significatives. Ensuite, pour comparer les méthodes deux à deux, il faut faire un choix entre l'utilisation de tests paramétriques ou non paramétriques. Ce choix se fait en fonction de l'indépendance des échantillons, de la normalité ou non de la distribution des échantillons et de l'homoscédasticité de leur variance. L'indépendance est assurée par le choix de graines (*seeds*) différentes. La condition de normalité est vérifiée avec le test de Kolmogorov-Smirnov et celle d'homoscédasticité par le test de Levene. Les conditions requises pour l'application des tests paramétriques ne sont pas réunies, car les hypothèses

nulles pour les conditions de normalité et d'homoscédasticité des variances sont rejetées à un niveau de confiance de $\alpha = 0.05$. Le test non paramétrique des rangs signés de Wilcoxon (*Wilcoxon signed-rank test*) a donc été appliqué. De manière complémentaire, pour réduire le problème des erreurs de Type I dans les comparaisons multiples, la méthode de correction de Bonferroni a été appliquée.

(Kronfeld, Dräger et al. 2009) donne le score +1 (resp. -1) pour l'algorithme dont les performances sont supérieures (resp. inférieures) chaque fois que l'hypothèse nulle considérée a pu être rejetée de manière significative. Un score de 0 est attribué lorsqu'aucun algorithme n'est significativement meilleur que l'autre. Comme nous avons trois études de cas différentes (2G, 3G, 5G), pour chaque paire d'algorithmes et chaque hypothèse nulle, nous additionnons les trois scores obtenus pour estimer lequel est globalement le meilleur, compte tenu des trois RRGHs. Le tableau 5.4 (resp. le tableau 5.5) montre ces sommes selon les tests de Wilcoxon par paire (resp. la correction de Bonferroni) : un nombre positif pour l'algorithme dans la ligne l montre qu'il était significativement meilleur que l'algorithme de la colonne c (en considérant les trois RRGHs). Par exemple, selon la correction de Bonferroni appliquée sur H_0^P, nous pouvons affirmer que cGAL29 est significativement meilleur que RS-CMSA-ESII pour les trois cas d'étude, mais par rapport à cGAL41, nous pouvons seulement dire qu'il est globalement meilleur : cGAL29 peut avoir obtenu +2 et cGAL41 +1 ou cGAL29 peut avoir obtenu +1 et cGAL41 0.

	CMA-ES		cGAL9		cGAL5		cGAL13		cGAC9		cGAL29		cGAL41		RSCMSAII	
GA	+2	+2	0	−2	0	+1	0	+1	−2	−1	−2	−1	0	+2	+2	+2
CMA-ES			−2	−2	−2	−2	−1	−1	−2	−2	−2	−2	−1	−1	0	+1
cGAL9					+3	+2	+1	+3	0	+1	0	0	+1	+3	+3	+3
cGAL5							0	0	−1	0	−1	−1	0	+2	+2	+2
cGAL13									−1	−3	−1	−3	0	+2	+2	+2
cGAC9											−1	−1	+1	+3	+3	+3
cGAL29													+1	+3	+3	+3
cGAL41															+2	+1

Tableau 5.4 – Tests statistiques *Pairwise Wilcoxon Rank Sum* de H_0^P (à gauche) et de H_0^F (à droite).

Si nous analysons les conclusions étayées par les tests, sur la base de l'acceptation ou du rejet des hypothèses, nous arrivons aux résultats suivants : sur les différentes tâches, cGAL9 et cGAL29 sont plus compétitifs en trouvant plus d'*optima* que les autres algorithmes avec de meilleures valeurs de fitness en moyenne. RS-CMSA-ESII n'est pas intéressant pour notre problème, car les algorithmes panmictiques maintiennent une plus grande diversité dans leur population à travers les différents paysages de RRGHs.

5.3.4 Visualisation

La figure 5.9 montre la diversité des solutions obtenues avec cGAL9 sur les différents RRGHs. Sur la figure 5.9a, on observe que les traces extraites sont très différentes dans certains états discrets, comme $\eta = (\eta_{v_1}, \eta_{v_2}) = (1, 0)$ et $(0, 1)$. C'est moins le cas pour les états discrets $(0, 0)$ et $(1, 1)$. En effet, les transitions continues dans ces états sont très

	CMA-ES		cGAL9		cGAL5		cGAL13		cGAC9		cGAL29		cGAL41		RSCMSAII	
GA	+2	+2	0	**0**	**+1**	**0**	0	+1	**0**	**0**	**−1**	**0**	0	+2	+2	**+1**
CMA-ES			−2	−2	−2	−2	**−2**	**−2**	−2	−2	−2	−2	**−2**	**−2**	0	**0**
cGAL9					**+1**	**+1**	**0**	**+2**	0	**0**	0	**+1**	**0**	**+2**	**+2**	**+2**
cGAL5							0	**+1**	−1	**−1**	−1	**0**	0	+2	+2	**+1**
cGAL13									**0**	**−2**	−1	−3	0	+2	+2	**0**
cGAC9											**0**	**0**	**0**	**+2**	**+2**	**+2**
cGAL29													+1	+3	+3	+3
cGAL41															+2	**0**

Tableau 5.5 – Analyse *post hoc* Bonferroni de H_0^P (à gauche) et H_0^F (à droite). Les valeurs en gras désignent la correction des erreurs de Type I comparées aux valeurs du tableau 5.4.

proches les unes des autres, signifiant que les valeurs des paramètres dynamiques dans ces états sont proches.

Une observation similaire peut être faite sur la figure 5.9b. Dans le premier état de la trace $\eta = (\eta_P, \eta_{BC}, \eta_R) = (1, 0, 0)$, ainsi que dans le second $(0, 0, 0)$, les traces sont semblables. Dans les états suivants, comme dans l'état discret $(1, 1, 1)$, la diversité est plus intéressante, car on observe des transitions avec ou sans glissement. Cette diversité est cohérente avec la connaissance biologique spécifiée par le triplet $\begin{pmatrix} 3.44 \\ \top \\ BC- \end{pmatrix}$ énoncé dans la section 2.3.2.

En figure 5.9c, on remarque que la diversité des traces est due, en grande partie, aux paramètres dynamiques de :

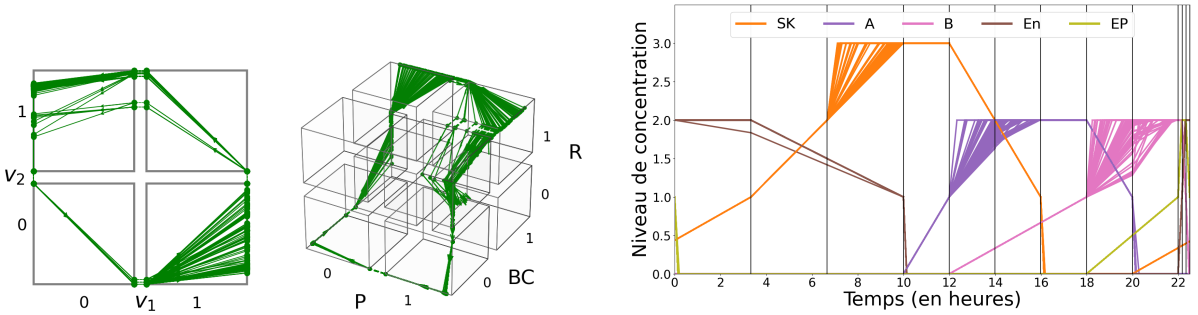
— SK dans le troisième état discret $\eta = (\eta_{SK}, \eta_A, \eta_B, \eta_{En}, \eta_{EP}) = (2, 0, 0, 1, 0)$ entre 6 heures 40 minutes et 10 heures environ ;

— A dans les cinq et sixièmes états discrets ($(2, 1, 0, 0, 0)$ et $(1, 1, 0, 0, 0)$) entre 12 et 16 heures environ ;

— B dans les huit et neuvièmes états discrets ($(0, 1, 1, 0, 0)$ et $(0, 0, 1, 0, 0)$) entre 18 et 22 heures environ.

5.3.5 Conclusion de l'étude expérimentale

Les cGAs ont montré leur supériorité sur RS-CMSA-ESII, en maintenant de la diversité génotypique au sein de la structure de la population. De manière surprenante, RS-CMSA-ESII ne garantit pas de diversité dans les résultats : lorsqu'une solution est trouvée, elle est unique. Dans notre cas, nous faisons l'hypothèse que les *optima* sont situés sur un *minimum* global plat. En effet, il existe un nombre infini de solutions formant un ensemble de mesure nulle. Cet ensemble est composé de solutions qui ne sont pas forcément disjointes. Il s'agit d'une surface de dimension inférieure à la taille du problème plutôt qu'à des points situés dans des vallées, comme c'est le cas dans la figure 5.1. Par conséquent, pour l'échantillonnage de solutions sur un tel paysage de fitness, les mécanismes employés par RS-CMSA-ESII ne sont pas adaptés. Comme le test de *Hill-Valley*



(a) (b) (c)

FIGURE 5.9 – Traces admissibles obtenues avec cGAL9 sur les réseaux 2G (a), 3G (b), et 5G (c).

échoue, il garantit qu'une seule sous-population à la fois évolue, conduisant à une solution unique. Cela entraîne une utilisation dégénérée de la métaheuristique et cela expliquerait les résultats décevants de RS-CMSA-ESII. Nous approfondissons cette hypothèse dans la section suivante. Dans le cas des cGAs, le maintien de la diversité par la structure de la population aide à préserver la diversité dans l'espace des paramètres et nous permet donc d'obtenir une diversité de paramétrisations menant à une diversité de modèles valides.

5.4 Originalité du problème multimodal

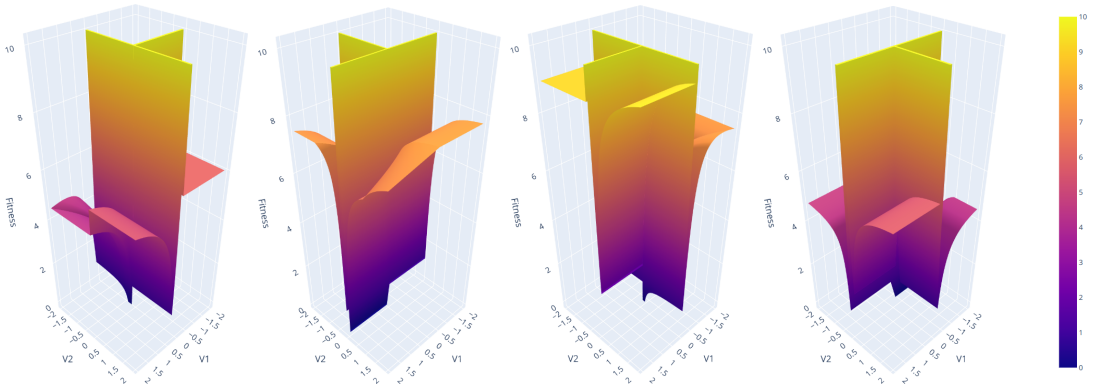
En partant des conclusions de l'étude expérimentale, nous formulons une hypothèse sur une des raisons pour laquelle RS-CMSA-ESII obtient des résultats sous-optimaux. Au regard de cette hypothèse, nous envisageons une approche de résolution.

5.4.1 Une hypothèse

Les résultats de l'étude expérimentale ont montré que la méthode de *niching* la plus performante (jusqu'à maintenant) sur les fonctions de benchmarks multimodales de CEC'2013 n'a pas été à la hauteur sur notre problème. Une hypothèse qui a émergé de cette conclusion est que le paysage de fitness d'un RRGH possède des particularités qui ne sont pas présentes sur les fonctions multimodales étudiées dans la communauté d'EA. Pour vérifier cette hypothèse, nous avons cherché à visualiser le paysage de fitness pour un RRGH donné.

Dans le cas du cycle négatif à deux variables, il y a un total de huit célérités à identifier : visualiser le paysage de fitness associé n'est pas possible à moins d'utiliser des techniques de réduction de la dimensionnalité comme l'analyse en composantes principales ou l'algorithme *Uniform Manifold Approximation and Projection* (McInnes, Healy et Melville 2020). Nous proposons de considérer un sous-ensemble de ce problème. La figure 5.10 illustre le paysage de fitness du problème d'identification d'une paramétrisation du cycle négatif (2G), décomposé état discret par état discret. Chaque problème décomposé revient à identifier le vecteur célérité d'un état discret particulier. La figure 5.10a représente le paysage de fitness associé à la recherche du vecteur célérité de l'état discret $(0, 0)$. Le paysage de fitness de la figure 5.10b est associé au vecteur célérité du second

état $\eta = (1, 0)$. Et il en va de même pour l'état $(1, 1)$ avec la figure 5.10c et l'état $(0, 1)$ avec la figure 5.10d. L'axe des abscisses correspond à la célérité de v_1 de l'état discret $(C_{v_1,\rho(h_0,v_1),\eta_{v_1}})$, dont la valeur est comprise dans $[-2, 2]$. L'axe des ordonnées correspond à la célérité de v_2 dans l'état $(C_{v_2,\rho(h_0,v_2),\eta_{v_2}})$, bornée dans $[-2, 2]$, dont la valeur est aussi comprise dans $[-2, 2]$. La valeur de fitness associée à chaque vecteur célérité est représentée dans la troisième dimension (l'axe des cotes) et un seuil est fixé à 10 pour un souci de représentation (les valeurs de fitness peuvent théoriquement atteindre $+\infty$). L'état hybride initial pour chaque état discret est fixé arbitrairement, mais de manière cohérente avec la connaissance biologique.



(a) (b) (c) (d)

FIGURE 5.10 – Visualisation des paysages de fitness du problème d'identification des vecteurs célérités du cycle négatif, décomposé état discret par état discret. Paysage de fitness associé à l'état discret (a) : $(0, 0)$, (b) : $(1, 0)$, (c) : $(1, 1)$, (d) : $(0, 1)$.

Nous observons que la particularité de chacun de ces paysages de fitness se situe au niveau de l'emplacement des solutions optimales. En effet, contrairement aux *optima* des fonctions multimodales de CEC'2013 qui se situent sur un ensemble de creux, les solutions optimales sont placées sur un plateau, un *minimum* plat. Usuellement, une solution d'un problème de minimisation est une fonction Dirac, un point de l'espace de recherche pour lequel tous ses voisins ont une valeur de fonction objectif strictement supérieure. Dans notre problème, l'inégalité stricte est relaxée : il peut exister des points voisins qui ont une valeur de fonction objectif égale.

Cette observation peut se vérifier analytiquement. Considérons l'exemple de la figure 5.10a où les solutions de ce sous-problème sont les vecteurs célérités dont les valeurs sont :

— $C_{v_1,\rho(h_0,v_1),0} = \frac{1}{5}$: il faut 5 heures avant d'atteindre un seuil (la distance que la trajectoire continue doit parcourir est de 1),

— $C_{v_2,\rho(h_0,v_2),0} \in]-\frac{1}{5}, 0[$: toute valeur négative différente de 0 (sinon il n'y aurait pas d'évolution de trajectoire), inférieure à $\frac{1}{5}$ est satisfaisante pour que la trajectoire ne glisse pas. Dans le cas contraire, la trajectoire glisse sur le bord inférieur de l'état discret et c'est incohérent avec la connaissance biologique.

Ainsi, en analysant les solutions de ce sous-problème, on remarque qu'elles forment (i) une infinité de solutions et (ii) ces solutions sont placées sur une ligne dans un plan. La figure 5.11 présente un zoom de la figure 5.10a sur l'ensemble des solutions qui forme un

minimum global plat. Cet ensemble est représenté par la ligne verte pointée par le curseur. Une solution de cet ensemble est d'ailleurs sélectionnée et visible dans le rectangle gris : $x = C_{v_1,\rho(h_0,v_1),0} = \frac{1}{5}$, $y = C_{v_2,\rho(h_0,v_2),0} = -0.04 \in]-\frac{1}{5}, 0[$ et la valeur de fitness associée z est bien égale à 0. Les courbes de niveau (lignes colorées sur le plan) ont été ajoutées pour montrer que la partie verte (qui correspond à l'ensemble des solutions) est bien une ligne dans un plan. Cela peut être aussi observé pour les autres sous-problèmes.

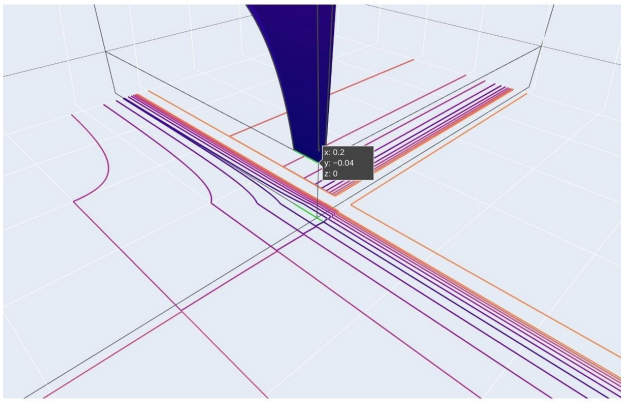


FIGURE 5.11 – L'ensemble des solutions du paysage de fitness de la figure 5.10a forme un *minimum* global plat (la ligne verte). La boite grise représente une solution de cet ensemble. Les lignes colorées sur le plan en deux dimensions représentent les courbes de niveau de la fonction.

Ainsi, si on considère le problème dans son ensemble, il existe un ensemble de solutions, dont les solutions sont placées sur un (ou plusieurs) *minimum* (ou *minima*) plat(s).

5.4.2 Une proposition de solution

Les techniques de recherche pour les fonctions multimodales les plus performantes comme HillValleyEA (MAREE et al. 2018) ou RS-CMSA-ES (I ou II) font usage du test de *Hill-Valley* et utilisent une unique sous-population d'un algorithme panmictique dans chacune des niches ou des bassins identifié(e)s. Or, si une niche est, comme dans notre cas d'étude, un ensemble de solutions au lieu d'être une solution unique, ces méthodes ne sont plus appropriées. Il existe, selon nous, deux possibilités pour traiter ce problème :

1. utiliser des mécanismes de préservation de diversité similaires à ceux employés par les cGAs ;
2. développer des techniques capables d'échantillonner un minimum plat.

Cette seconde option reste à être explorée. Il existe de nombreuses méthodes pour échapper à un plateau, car cela ralentit le processus d'optimisation, comme c'est le cas avec la variante du hill-climbing avec redémarrage aléatoire, *Shotgun hill climbing*, présentée dans (RUSSELL et NORVIG 2016). Ici, au contraire, dès lors qu'un *minimum* plat est identifié, une méthode d'échantillonnage ou de pavage de cet espace permettrait de trouver des solutions diversifiées. Ce pourrait être un algorithme de recherche locale avec un budget fixé dont le but serait de ne pas sortir du plateau.

Une fois la méthode développée, cette dernière devra ensuite être évaluée sur une suite de fonctions de *benchmark* multimodales adéquates. Ces dernières restent à définir. L'objectif de la section 5.4.3 est de présenter cette perspective de recherche.

5.4.3 Des exemples de fonctions de benchmarks

La suite de fonctions multimodales devra alors être composée de segments de fonctions plates. L'exemple trivial est une fonction constante sur $\mathbb{R}$ qui est plate partout. Ci-dessous, se trouve une liste de fonctions simples à une dimension qui pourraient être envisagées par la communauté scientifique comme nouvelles fonctions de *benchmark*.

Valeur absolue avec un *minimum* global plat.

$$F_1(x) = \begin{cases} 0 & \text{si } x \in [-c, c] \\ |x| - c & \text{sinon.} \end{cases} \tag{5.10}$$

avec $|.|$ la valeur absolue, $c > 0$ un paramètre à déterminer et $x \in [-c - n, c + n]$ avec $n \geq 0$. Si $n = 0$, l'espace de recherche est borné de telle sorte qu'il s'agit de la fonction constante, sinon il existe un unique *minimum* global plat. Elle est illustrée en figure 5.12a.

Valeur absolue avec un *minimum* global plat et plusieurs *minima* locaux plats.

$$F_2(x) = \begin{cases} 0 & \text{si } x \in [-1, 1] \\ 2 \times |x| - 2 & \text{si } x \in [-2, -1] \cup [1, 2] \\ 2 & \text{si } x \in [-3, -2] \cup [2, 3] \\ -|x| + 5 & \text{si } x \in [-4, -3] \cup [3, 4] \\ 1 & \text{si } x \in [-5, -4] \cup [4, 5] \\ |x| - 4 & \text{sinon.} \end{cases} \tag{5.11}$$

Il existe un unique *minimum* global plat, 2 *minima* locaux plats et 2 paliers, voir figure 5.12b.

Valeur absolue avec plusieurs *minima* globaux plats.

$$F_3(x) = \begin{cases} -|x| + 1 & \text{si } x \in [-1, 1] \\ 0 & \text{si } x \in [-2, -1] \cup [1, 2] \\ |x| - 2 & \text{sinon.} \end{cases} \tag{5.12}$$

Il existe deux *minima* globaux plats, comme on peut observer en figure 5.12c.

On peut modifier la fonction précédente pour qu'elle présente un palier. $F_4(x)$ est illustrée en figure 5.12d.

$$F_4(x) = \begin{cases} 1 & \text{si } x \in [-0.5, 0.5] \\ -2|x| + 2 & \text{si } x \in [-1, -0.5] \cup [0.5, 1] \\ 0 & \text{si } x \in [-2, -1] \cup [1, 2] \\ |x| - 2 & \text{sinon.} \end{cases} \tag{5.13}$$

Ces fonctions sont évidemment généralisables en n'importe quelle dimension. On peut aussi envisager une topologie plus spécifique lorsqu'on est en plus grande dimension, comme par exemple, trouver les points formés par l'intersection d'un tore posé sur un plan, de telle sorte à obtenir un ensemble de solutions optimales placées sur un cercle.

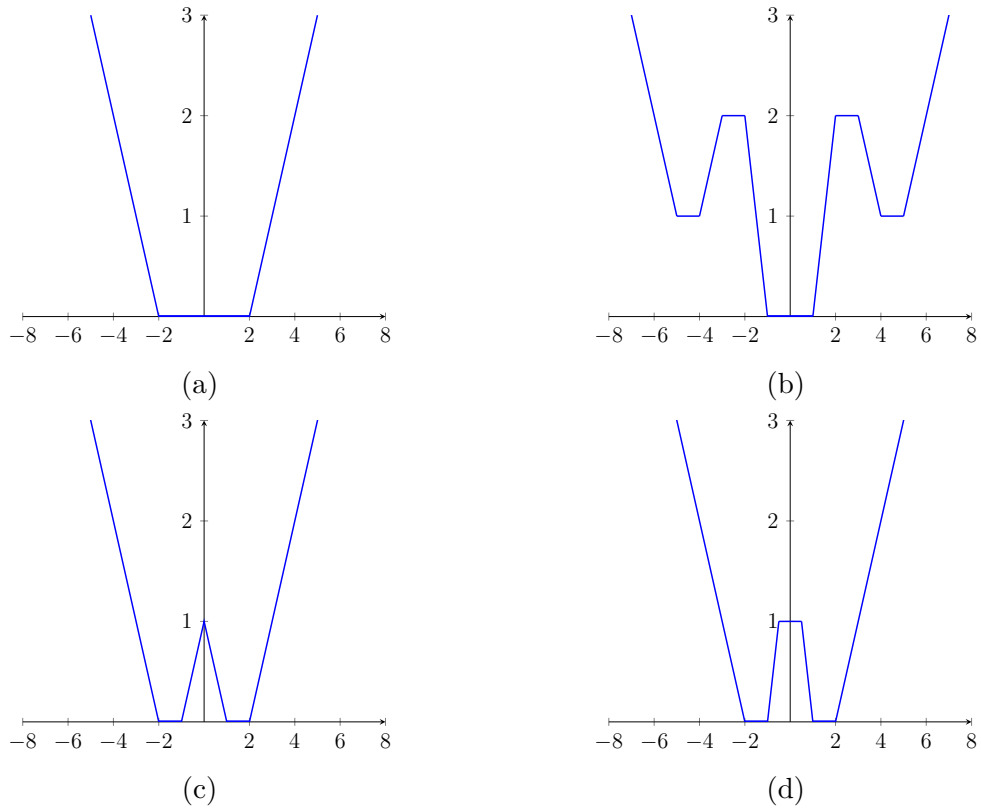


FIGURE 5.12 – Visualisation de fonctions en deux dimensions avec un ou plusieurs *minima* globaux plats.

5.5 Synthèse

Dans ce chapitre, la recherche d'un ensemble de solutions au cours d'une unique exécution d'un algorithme d'EA est traitée comme un problème d'optimisation multimodale. L'étude expérimentale réalisée met en concurrence deux stratégies de préservation de la diversité à travers deux algorithmes différents : RS-CMSA-ESII utilise une sous-population différente par niche identifiée et les algorithmes cellulaires génétiques intègrent la diversité au sein d'une même population en conditionnant la génération des nouveaux individus à un voisinage spécifique. Les résultats de l'étude montrent que le second mécanisme, avec une structure de voisinage adéquate, est celui qui permet d'extraire l'ensemble de solutions plus diverses et de cardinalité plus grande. L'issue de cette étude nous a aussi amenés à émettre une hypothèse quant à l'originalité du problème. Cette hypothèse tente de mettre en évidence la raison pour laquelle la première stratégie n'est pas efficace sur notre problème : il s'agit de la spécificité topologique de l'ensemble des solutions sur le paysage de fitness. Finalement, des pistes de recherche pour traiter ce type de fonction objectif sont évoquées. Elles pourraient mener à l'étude d'un nouvel aspect des problèmes multimodaux pour la communauté d'EA.

Le prochain chapitre traite du problème de la multimodalité dans le domaine de la prise de décision séquentielle avec MCTS. Le terme de planification diverse est employé dans cette communauté pour décrire la recherche d'un ensemble de solutions au cours de la construction de l'arbre de recherche de MCTS.

6 Planification diversifiée

The subject of optimization is a fascinating blend of heuristics and rigour, of theory and experiment.

Extrait de (Fletcher 2000)

Les méthodes capables de générer un ensemble de solutions optimales d'un problème au cours d'une unique exécution sont désirables pour un décideur. Dans notre cas, l'approche du modélisateur exige de produire un ensemble de solutions diverses dans le but d'appréhender la diversité des comportements issus des données biologiques disponibles.

Dans ce chapitre, nous nous concentrons sur les méthodes qui s'appliquent aux problèmes de décision séquentiels formulés comme des processus de décisions markoviens (PDMs). Ce domaine de recherche est très fécond dans le cadre des problèmes qui requièrent un modèle du PDM : la planification. Les planificateurs qui produisent des ensembles de solutions optimales, appelés plans, ont des applications dans un large éventail de domaines comme l'analyse de vulnérabilités des réseaux informatiques (Boddy et al. 2005), la recherche de préférences utilisateurs avec informations incomplètes ou inconnues (Nguyen et al. 2012), la qualité de service (Deng et al. 2013) ou encore la prédiction de scénarios pour la gestion des risques en entreprise (Sohrabi et al. 2018).

Dans ce chapitre, nous investiguons une approche récemment développée qui propose d'utiliser la recherche arborescente Monte Carlo pour la recherche de plusieurs solutions. La section 6.1 commence par fournir un état de l'art sur les méthodes du domaine de la planification qui génèrent un ensemble de solutions optimales. Dans l'ensemble de ces méthodes, une partie nous intéresse plus particulièrement : la recherche d'un ensemble de solutions optimales et diversifiées. Nous détaillons ensuite dans la section 6.1.2 comment cette approche particulière de la planification a été adaptée dans le cas de la recherche arborescente Monte Carlo pour des problèmes combinatoires. À partir de cet état de l'art, nous proposons dans la section 6.2 plusieurs contributions heuristiques dans le cadre continu. Dans la section 6.3, nous les évaluons sur notre problème d'identification des paramètres dynamiques dans les RRGHs. Finalement, la section 6.4 offre une synthèse du chapitre.

6.1 État de l'art

Cette section est documentée à partir du travail de (Benke et al. 2023) qui introduit le problème de la recherche d'un ensemble de solutions optimales et diverses avec MCTS. Les idées et concepts présentés dans ces travaux proviennent du domaine de la planification, c'est pourquoi nous choisissons de les introduire à travers ce prisme. La planification est une approche de résolution de problèmes de décision séquentiels basée sur la connaissance complète d'un modèle du PDM, noté $\Pi = \langle S, A, T, R \rangle$ (définition 18), dans lequel la fonction de récompense R et la fonction de transition T sont connues à l'avance. Un planificateur génère une stratégie unique ou une séquence d'actions, appelée **plan**, pour résoudre une tâche de planification spécifique. Nous nous concentrons ici sur les planificateurs permettant d'extraire un ensemble de plans optimaux. Nous choisissons d'utiliser la notation ν pour un plan.

6.1.1 Planification top-k, top-qualité ou diversifiée

La génération d'un ensemble de plans peut faire référence à plusieurs types de planification : la **planification top-k**, **top-qualité** ou **diversifiée** lorsque l'ensemble est limité respectivement par la cardinalité, la qualité ou la diversité. La planification top-k limite l'ensemble des plans à une taille fixe, la planification de qualité supérieure impose une qualité de solution minimale et la planification diversifiée ajoute une contrainte sur la similarité entre les différents plans. Plus spécifiquement, la planification top-k restreint l'ensemble des plans à un ensemble borné par sa cardinalité. La planification top-qualité requiert que la qualité d'un plan soit supérieure à un seuil fixé. La planification diversifiée possède la contrainte que chaque plan de l'ensemble soit « significativement » distinct des autres. Nous verrons par la suite comment définir cette significativité.

Planification top-k La planification top-k traite du problème de la recherche d'un ensemble fini de plans de cardinalité k, tel qu'il n'existe aucun plan de qualité supérieure ne faisant pas partie dudit ensemble. La notion de qualité d'un plan est déterminée par une mesure fournie par l'utilisateur. Il est intéressant de noter que la planification top-k peut être considérée comme une forme de généralisation de la planification simple dans laquelle on a $k = 1$. En s'inspirant de la définition proposée dans (Speck, Mattmüller et Nebel 2020), un problème de planification top-k peut être introduit comme suit :

Problème de planification top-k

Définition 21. Étant donnés un PDM noté $\Pi = \langle S, A, T, R \rangle$, une mesure de qualité d'un plan ν, $Q_{plan}(\nu)$, et un entier naturel k, le problème de planification top-k consiste à trouver un sous-ensemble P inclus dans l'ensemble des plans P_Π tel que :

- $\forall \nu \in P, \nexists \nu' \in P_\Pi \setminus P$ tel que $Q_{plan}(\nu') > Q_{plan}(\nu)$,
- $|P| = k$ si $|P_\Pi| \geq k$, $|P| = |P_\Pi|$ sinon.

Planification top-qualité La planification top-qualité peut être vue comme une variante de la planification top-k dans laquelle la contrainte de *quantité* est remplacée par

une contrainte de *qualité*. Cette approche est intéressante pour les applications qui ne nécessitent pas un nombre défini de solutions.

D'après (Katz, Sohrabi et Udrea 2020), un problème de planification top-qualité peut être défini comme :

Problème de planification top-qualité

Définition 22. Étant donnés un PDM $\Pi = \langle S, A, T, R \rangle$, une mesure de qualité d'un plan ν, $Q_{plan}(\nu)$, et une contrainte de qualité q, un problème de planification top-qualité consiste à trouver un ensemble de plans $P \subset P_\Pi$ tel que $P = \{\nu \in P_\Pi \mid Q_{plan}(\nu) \geq q\}$.

Planification diversifiée La planification diversifiée considère le problème de génération d'un ensemble de plans significativement différents ou diversifiés. Cette notion de diversité requiert la définition d'une mesure de dissimilarité entre plans et cette mesure est usuellement définie comme la distance *minimum* ou moyenne entre deux plans. Différentes mesures ont été introduites dans la littérature. Certaines, appelées mesures qualitatives, sont dépendantes du domaine d'application, comme dans (Myers et T. J. Lee 1999), d'autres, appelées mesures quantitatives, sont indépendantes du domaine d'application (Srivastava et al. 2007). Il existe aussi des mesures hybrides (Coman et Munoz-Avila 2011) basées sur des mesures de distance de plan personnalisables et se prêtant ainsi à la définition d'une notion de diversité à la fois quantitative et qualitative.

D'après (Benke et al. 2023), un problème de planification diversifiée peut être défini comme :

Problème de planification diversifiée

Définition 23. Étant donné un PDM $\Pi = \langle S, A, T, R \rangle$, une mesure de qualité d'un plan ν, $Q_{plan}(\nu)$, une contrainte de qualité q, une distance minimale d'un plan à un ensemble de plans $D(\nu, P)$ et une contrainte de distance δ, résoudre un problème de planification diversifiée, c'est trouver un sous-ensemble maximal $P \subset P_\Pi$ tel que :

- $\forall \nu \in P, \nexists \nu' \in P_\Pi \setminus P$ tel que $D(\nu', P) \geq \delta$ et $Q_{plan}(\nu') > Q_{plan}(\nu)$;
- $\forall \nu \in P, D(\nu, P \setminus \nu) \geq \delta$;
- $\forall \nu \in P, Q_{plan}(\nu) \geq q$;
- $|P| \leq k$.

Dans ce contexte, notons que si la contrainte de distance est initialisée à 0, la planification diversifiée peut être considérée comme une généralisation de la planification top-k et top-qualité.

Nous présentons maintenant comment résoudre un problème de planification diversifiée lorsque le modèle du MDP n'est pas complètement connu, plus spécifiquement, à travers l'utilisation de MCTS.

6.1.2 Planification diversifiée avec MCTS

Les travaux de (Benke et al. 2023) montrent que MCTS peut être utilisé pour générer des plans en extrayant des séquences d'actions de l'arbre de recherche $\mathcal{T}$. Dans ce contexte, un plan correspond à l'ensemble des nœuds (paires état-action) sélectionnés dans le chemin menant de la racine de l'arbre à un nœud feuille. On le note $\nu = \{s_0, a_0, ..., s\}$ où s_0 est l'état initial du nœud racine et s est l'état d'un nœud feuille de l'arbre de recherche.

Dans le cas le plus simple (top-k avec $k = 1$, comme dans le chapitre 4), le plan optimal peut être extrait en commençant par le nœud racine et en sélectionnant récursivement le nœud enfant ayant la valeur d'une action $Q(s, a)$ maximale jusqu'à ce qu'un nœud feuille soit atteint. De là, on peut facilement extraire la séquence d'actions correspondante. La génération d'un ensemble de plans optimaux est plus complexe. Elle nécessite une méthode d'extraction des plans dans un arbre de recherche se reposant sur une méthode de comparaison des plans (qui peuvent être de taille variable). Nous présentons une mesure de la qualité d'un plan d'un arbre de recherche, ainsi qu'un processus d'extraction d'un ensemble de plans soumis aux contraintes de cardinalité, de qualité et de diversité, tous deux introduits dans (Benke et al. 2023).

Mesure de qualité relative d'une branche. Dans le cas d'un problème de planification, la qualité d'un plan est usuellement définie comme la somme des coûts des actions (Katz, Sohrabi et Udrea 2020). Dans le cas de MCTS, les récompenses sont supposées n'être reçues que dans les états terminaux. Ainsi, la mesure de la qualité est basée sur le rapport entre la valeur du nœud et la valeur du nœud du plan optimal. Cette mesure relative est aussi nécessaire pour comparer des plans de longueurs différentes.

Mesure de qualité relative d'un plan

Définition 24. Étant donné un chemin d'un arbre de recherche $\mathcal{T}$, *i.e.*, un plan $\nu = \{s_0, a_0, s_1, a_1, ..., a_{l-1}, s_l\}$ formé d'une séquence d'états et d'actions partant de l'état initial s_0 (nœud racine) jusqu'à un état s_l (nœud feuille), la qualité relative de ν est définie comme :

$$Q_{plan}(\nu) = \prod_{t=0}^{l-1} \frac{Q(s_t, a_t)}{\max_{a' \in A(s_t)} Q(s_t, a')} \tag{6.1}$$

Il s'agit du produit des rapports à chaque instant t entre la valeur obtenue résultant du choix de l'action a_t à partir de s_t (le numérateur) et la valeur obtenue si le choix de l'action avait été optimal à partir de s_t (le dénominateur).

La figure 6.1 illustre, à gauche, un arbre de recherche contenant cinq plans $\{\alpha, \beta, \gamma, \epsilon, \delta\}$. Chaque nœud de l'arbre de recherche possède une valeur Q correspondant à la paire état-action qu'il représente et cette valeur est précisée à l'intérieur de chaque nœud. La figure présente, à droite, les calculs détaillés de la qualité de chacun des plans $\{Q_\alpha, Q_\beta, Q_\gamma, Q_\epsilon, Q_\delta\}$. Le plan optimal est α dont la qualité relative est égale à 1. Prenons l'exemple de δ, sa qualité relative est un produit qui peut être décomposé de la manière suivante :

— le fils gauche du nœud racine (s_1) a été choisi alors que le fils droit (s_2) relève du choix optimal, car l'action qu'il a choisi fait partie du plan optimal α. La fraction

correspond donc à la valeur du choix réalisé ($\frac{5}{10}$) sur la valeur du choix optimal ($\frac{9}{14}$). Le résultat du premier produit est donc égal à $\frac{5/10}{9/14}$;

— s_3 (fils gauche de s_1) a été choisi alors que s_4 (fils droit de s_1) relève du choix optimal. Ce rapport correspond donc à $\frac{1/5}{4/5}$.

On obtient bien $Q_\delta = \frac{5/10}{9/14} \times \frac{1/5}{4/5} \approxeq 0.19$.

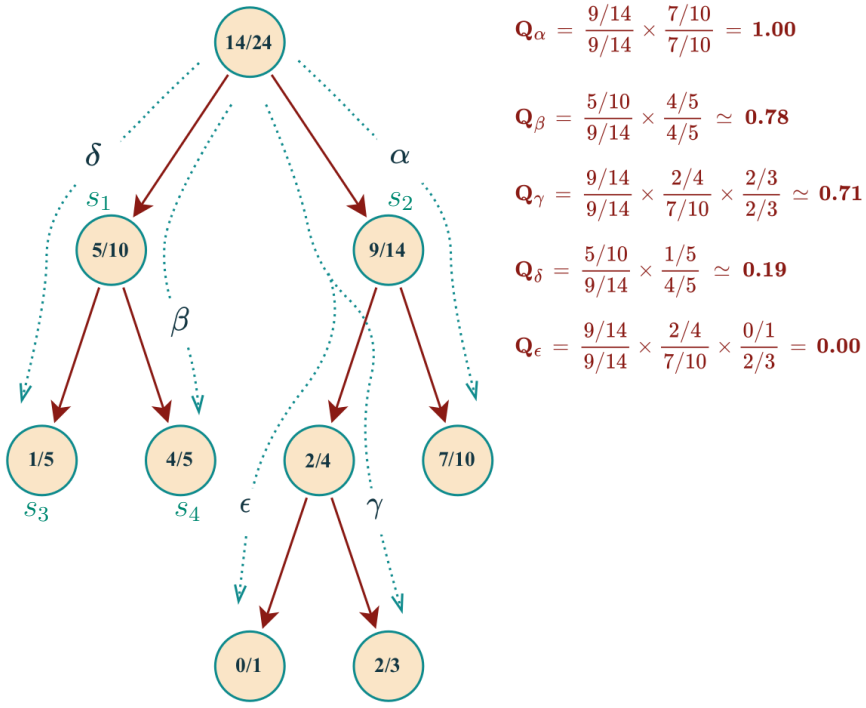


FIGURE 6.1 – Illustration du calcul de la qualité relative d'un plan dans un arbre de recherche partiellement construit. La valeur Q de chaque paire état-action est précisée à l'intérieur de chaque nœud. α est le plan optimal et obtient la valeur $Q_\alpha = 1$. Les autres plans font des choix d'actions sous-optimaux par rapport à α. Figure extraite de (BENKE et al. 2023) et annotée.

Algorithme d'extraction d'un ensemble de plans diversifiés. L'algorithme 14 présente un processus d'extraction d'un ensemble de plans optimaux d'un arbre de recherche. L'algorithme peut être appliqué à des problèmes de planification diversifiée, top-k ou top-qualité en ajustant respectivement δ la contrainte de diversité, k la contrainte de cardinalité et q la contrainte de qualité (*cf.* les entrées de l'algorithme). L'arbre de recherche est parcouru en développant progressivement les plans par ordre de qualité (*best-first order*). Pendant l'exécution de l'algorithme, la file de priorité $\mathcal{F}$ contient une liste de séquences de nœuds. Ces séquences sont ordonnées par qualité, calculée en fonction de la mesure introduite en définition 24. Ces séquences, à profondeur partielle, s'étendent du nœud racine jusqu'à un nœud qui n'est pas une feuille de l'arbre de recherche : il s'agit de tiges de plan (*plan stems*), concept introduit dans (ZIMMERMAN et KAMBHAMPATI 2002). À chaque itération de la boucle (ligne 5), la tige de plan ν la plus prioritaire est retirée de la liste $\mathcal{F}$ (ligne 6) et un ensemble de plans est généré, chacun de longueur

$|\nu| + 1$, en ajoutant successivement chaque enfant du nœud final de ν (lignes 7-10). La qualité de ces nouvelles tiges est évaluée et toutes celles qui satisfont à la contrainte de qualité minimale q sont ajoutées à $\mathcal{F}$ (lignes 11-15). Si ν n'a pas été étendu, le plan a atteint un nœud feuille et, s'il répond à la contrainte de diversité δ, le plan est ajouté à l'ensemble final de plans complets et optimaux P (lignes 17-19). La diversité permet de trancher les égalités entre plans de même qualité (lignes 20-28). Le processus est répété jusqu'à ce que $\mathcal{F}$ soit vide ou que le nombre souhaité de plans k ait été généré.

Algorithm 14 Pseudo-code de l'algorithme pour extraire un ensemble borné de k plans dont la qualité est inférieure à q (critère d'optimalité) et respectant une contrainte de diversité δ (critère de diversité).

Input: s_0, $Q_{plan}(\nu)$, $D(\nu, P)$, $k \in \mathbb{Z}^+, q \in [0,1], \delta \in [0,1]$
Output: P un ensemble de plans optimaux et diverses

1: $P \leftarrow \emptyset$
2: $\nu_{s_0} \leftarrow \{s_0\}$
3: $\mathcal{F} \leftarrow FileDePriorite()$
4: $\mathcal{F}.push(\nu_{s_0}, priorite = 1)$
5: **while** $\mathcal{F} \neq \emptyset$ **do**
6: $\quad \nu \leftarrow \mathcal{F}.pop\_max()$
7: $\quad s \leftarrow \nu.get\_last()$
8: $\quad etendu \leftarrow False$
9: $\quad$ **for all** $a \in C_{action}(s)$ **do**
10: $\quad\quad \nu' \leftarrow \nu \cup \{T(s,a)\}$
11: $\quad\quad q_{\nu'} \leftarrow Q_{plan}(\nu')$
12: $\quad\quad$ **if** $q_{\nu'} \geq q$ **then**
13: $\quad\quad\quad \mathcal{F}.push(\nu', priorite = q_{\nu'})$
14: $\quad\quad\quad etendu \leftarrow True$
15: $\quad\quad$ **end if**
16: $\quad$ **end for**
17: $\quad$ **if** $\neg etendu \wedge (D(\nu, P) \geq \delta)$ **then**
18: $\quad\quad$ **if** $|P| < k$ **then**
19: $\quad\quad\quad P.push(\nu)$
20: $\quad\quad$ **else**
21: $\quad\quad\quad q_{min} \leftarrow \min_{\xi \in P} Q_{plan}(\xi)$
22: $\quad\quad\quad$ **if** $(\delta = 0) \vee (Q_{plan}(\nu) < q_{min})$ **then**
23: $\quad\quad\quad\quad$ **break**
24: $\quad\quad\quad$ **end if**
25: $\quad\quad\quad P_{q_{min}} \leftarrow \{\xi |\ \xi \in P, Q_{plan}(\xi) = q_{min}\}$
26: $\quad\quad\quad \nu_{\delta_{min}} \leftarrow \text{argmin}_{\xi \in P_{q_{min}}} D(\xi, P \setminus \xi)$
27: $\quad\quad\quad$ **if** $D(\nu, P) > D(\nu_{\delta_{min}}, P - \nu_{\delta_{min}})$ **then**
28: $\quad\quad\quad\quad P.replace(\nu_{\delta_{min}}, \nu)$
29: $\quad\quad\quad$ **end if**
30: $\quad\quad$ **end if**
31: $\quad$ **end if**
32: **end while**
33: **return** P

Dans (Benke et al. 2023), cet algorithme a été prouvé :

— correct : aucun plan de qualité supérieure ne peut exister dans l'arbre en dehors de l'ensemble P, et

— complet : P contient tous les plans optimaux de l'arbre qui satisfont aux contraintes de qualité q et de diversité δ.

Pour assurer la diversité de l'ensemble des plans générés par l'algorithme, chaque plan est évalué par rapport à l'ensemble des meilleurs plans P, et tous ceux qui ne satisfont pas la contrainte de diversité requise δ sont rejetés (test de la ligne 17). La diversité d'un plan ν, $D(\nu, P)$, est une mesure de sa dissimilarité par rapport à l'ensemble des plans P. Cette mesure est calculée comme la distance minimale par paire (*minimum pairwise distance*) par rapport à P, qui peut être basée sur des mesures qualitatives, spécifiques au domaine, ou sur des mesures quantitatives, indépendantes du domaine. Pour calculer $D(\nu, P)$, une mesure de dissimilarité entre deux plans doit donc être fournie par l'utilisateur, on la note $d(\nu, \nu')$. La mesure de diversité utilisée dans l'algorithme 14 est calculée comme la distance minimale d'un plan ν à n'importe quel autre plan de P :

$$D(\nu, P) = \min_{\nu' \in P} d(\nu, \nu') \tag{6.2}$$

Les plans extraits de l'algorithme sont des *optima* de Pareto. La figure 6.2 illustre ce concept. Le problème consiste à maximiser la qualité et la diversité d'un plan. Chaque point correspond à la qualité (en abscisse) et à la diversité (en ordonnée) d'une solution de P. Les points rouges représentent des plans dominés : il existe un autre plan dont la valeur de qualité ou de diversité est supérieure à la sienne. Selon cette définition, les plans en bleu ne sont pas dominés. Ils sont appelés *optima* de Pareto et forment un front de Pareto.

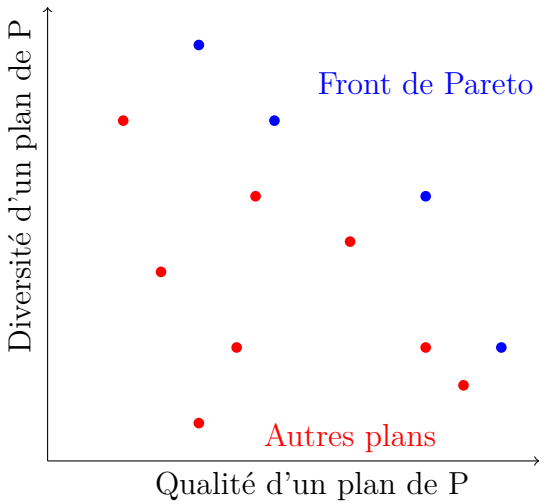


Figure 6.2 – Illustration d'un front de Pareto sur le problème de maximisation de la qualité et de la diversité d'un plan. Chaque point représente la qualité d'un plan de P (en abscisse) et sa diversité (en ordonnée). Chaque point rouge est dominé : pour chaque point rouge, il existe un point bleu qui a une meilleure valeur de qualité ou de diversité. Les points bleus forment le front de Pareto du problème.

Stratégie de bandit diversifiée. À l'aide de sa fonction de sélection, MCTS oriente l'exploration vers les zones les plus prometteuses de l'espace de recherche. Ainsi, les branches éloignées de celle correspondant au plan optimal sont peu visitées. Il en résulte une réduction possible de la diversité de l'ensemble des plans finaux.

Pour y remédier, deux approches ont été proposées, toujours dans (Benke et al. 2023), pour accroître la diversité de l'arbre de recherche de MCTS. La première solution, la plus évidente, consiste à augmenter la constante d'exploration (c dans UCB1, équation (4.9)) ou de biais (*bias* dans RAVE, équation (4.14)). Cette constante permet d'équilibrer l'exploration et l'exploitation dans la recherche de solutions lors de la construction de l'arbre : l'augmenter permet d'encourager l'exploration des zones les moins prometteuses. Cette approche peut néanmoins être pénalisante en termes de performances, car elle sacrifie la recherche en profondeur au profit d'une recherche en largeur et nécessite donc un plus grand nombre d'itérations pour que MCTS produise un arbre de recherche avec des estimations fiables. En outre, l'exploration supplémentaire n'est pas dirigée et n'oriente pas nécessairement la recherche vers des plans différents de haute qualité.

Pour orienter l'exploration de l'arbre vers des solutions diverses, une deuxième approche, appelée *diverse multi-armed bandit* (DMAB) (Benke et al. 2023), consiste à modifier la politique de l'arbre en tenant compte explicitement de la diversité. On modifie la politique de sélection en ajoutant un terme propre à la diversité. Par exemple, la formule UCB1 peut être modifiée pour inclure ce terme supplémentaire :

$$Q^{\oplus}_{\text{divUCT}}(s,a) = \begin{cases} \overbrace{Q_{\text{UCT}}(s,a)}^{\text{terme d'exploitation}} + c \times \overbrace{\sqrt{\frac{log(n(s))}{n(s,a)}}}^{\text{terme d'exploration}} + \overbrace{D(\nu_{s,a},P)}^{\text{terme de diversité}} & \text{, si } n(s,a) > 0 \\ +\infty & \text{, sinon.} \end{cases} \tag{6.3}$$

où $D(\nu_{s,a}, P)$ est la diversité de la tige de plan $\nu_{s,a}$ par rapport à l'ensemble des plans P. La tige de plan est formée des états et actions de l'état initial jusqu'à l'état résultant de l'action a à partir de l'état s.

Ainsi, la politique de l'arbre devient :

$$\pi_{\text{divUCT}}(s) = \text{argmax}_{a \in A(s)} Q^{\oplus}_{\text{divUCT}}(s,a) \tag{6.4}$$

6.2 Contributions

Cette section présente des contributions, extraites de (Michelucci, Jean-Paul Comet et Pallez 2024), qui traitent de la recherche d'un ensemble de paramétrisations diversifiées dans les RRGHs, au cours d'une seule exécution de MCTS. Dans ce problème, le domaine des états et des actions est continu. Nous introduisons des heuristiques dans ce cadre.

6.2.1 Stratégies d'inhibition

Tout d'abord, nous proposons deux stratégies, que l'on nomme Verrou (*Lock*) et DivPW (*Diverse Progressive Widening*). Leur but est de renforcer le biais de diversité

dans l'ensemble des plans de qualité optimale P, lors de la construction de l'arbre de recherche. L'idée sous-jacente de ces stratégies est d'inhiber certaines zones de l'arbre de recherche qui conduisent déjà à des plans optimaux pour renforcer l'exploration de zones prometteuses. Nous les nommons stratégies d'inhibition.

Verrou (*Lock*). La première approche consiste simplement à verrouiller les branches de l'arbre $\mathcal{T}$ dans lesquelles un plan ν est stocké dans P. Lorsqu'un plan est sauvegardé dans P, la séquence d'actions sélectionnées $(a_0, ..., a_l)$ est retirée de $\mathcal{T}$. Cela signifie que toute la branche de l'arbre est inhibée, pas uniquement le plan.

***Diverse Progressive Widening* (DivPW).** Notre deuxième approche consiste à modifier la formule du *progressive widening*, PW (équation (4.17)), afin de promouvoir l'échantillonnage de nouvelles actions à chaque profondeur de $\mathcal{T}$. Nous ajoutons à PW la contrainte selon laquelle les actions à partir d'un état s_t ($C_{\text{action}}(s_t)$) ne doivent pas appartenir à l'ensemble des actions sauvegardées dans les plans de P à cette même profondeur : $P^t_{\text{action}} = \{a_i \,|\, (s_i, a_i) \ \in P, i = t\}$. Au lieu de considérer le nombre d'actions à partir de s_t ($C_{\text{action}}(s_t)$), nous ne considérons que le sous-ensemble des actions qui ne sont présentes dans aucun des plans à cette profondeur : $C^{\text{DivPW}}_{\text{action}}(s_t) = C_{\text{action}}(s_t) \setminus \{a \,|\, a \in C_{\text{action}}(s_t) \cap P^t_{\text{action}}\}$.

La condition DivPW est ainsi formulée :

$$n(s_t)^{\zeta} > |C^{\text{DivPW}}_{\text{action}}(s_t)| \tag{6.5}$$

Si la condition DivPW (de l'équation (6.5)) est remplie et que $C^{\text{DivPW}}_{\text{action}}(s_t)$ n'est pas un ensemble vide, la fonction de sélection est appliquée sur le sous-ensemble de nœuds dont les actions ne sont pas dans P à cette profondeur. Cela empêche l'action d'être à nouveau sélectionnée à la même profondeur, alors qu'un plan de P la contient déjà. Si la condition n'est pas satisfaite, on évite de rééchantillonner une action qui existe déjà dans P^t_{action}, si cela est possible. Comme nous sommes dans un cadre continu, on considère une action à ϵ près ($a \pm \epsilon$) qui est un hyperparamètre à déterminer en fonction du problème.

Ces heuristiques, en inhibant des plans, poussent l'exploration vers des zones de l'espace de recherche non visitées ou moins prometteuses. Elles sont testées et comparées dans la section suivante.

6.2.2 Diversité comme objectif dans MCTS multi-objectif

Nous proposons une troisième stratégie qui considère le problème de planification diversifiée comme un problème d'apprentissage par renforcement multi-objectif. Dans la stratégie DMAB (équation (6.3)), la diversité des solutions n'est pas considérée comme un objectif à part entière. En fait, la planification diversifiée est principalement considérée comme un problème intermédiaire, représenté comme une combinaison linéaire entre le résultat de la simulation et le score de diversité. Étant donné que la solution d'un problème de planification diversifiée est un ensemble de plans Pareto optimal entre la diversité et la qualité des plans, nous envisageons d'utiliser le variant multi-objectif de MCTS.

MO-MCTS (W. Wang et Michèle Sebag 2012) est l'extension multi-objectif de MCTS qui a été démontré plus performant que les approches reposant sur la combinaison linéaire des objectifs. MO-MCTS considère un vecteur de récompenses $r \in \mathbb{R}^d$ où d

est le nombre d'objectifs. Plusieurs concepts ont besoin d'être introduits pour comprendre le fonctionnement de MO-MCTS :

— une archive, notée χ, conserve toutes les récompenses vectorielles non dominées, (évaluées dans les itérations précédant l'utilisation de l'archive) ;

— l'indicateur d'hypervolume (Zitzler et Thiele 1998), noté HV, est une mesure scalaire qui permet d'évaluer l'intérêt d'un front de Pareto. Plus le score de HV est grand, meilleur est le front de Pareto. Cette mesure prend en entrée un ensemble de vecteurs de récompenses et un point de référence et renvoie une valeur réelle. La figure 6.3 (a) représente 10 récompenses vectorielles en deux dimensions (les croix) dont les deux composantes r_1 et r_2 sont à maximiser. Les récompenses en rouge sont dominées et les récompenses en noir forment le front de Pareto. Le point w (dans le coin inférieur gauche) est le point de référence pour le calcul de l'indicateur d'hypervolume. La valeur du score de HV du front de Pareto correspond à la surface de la région ombrée ;

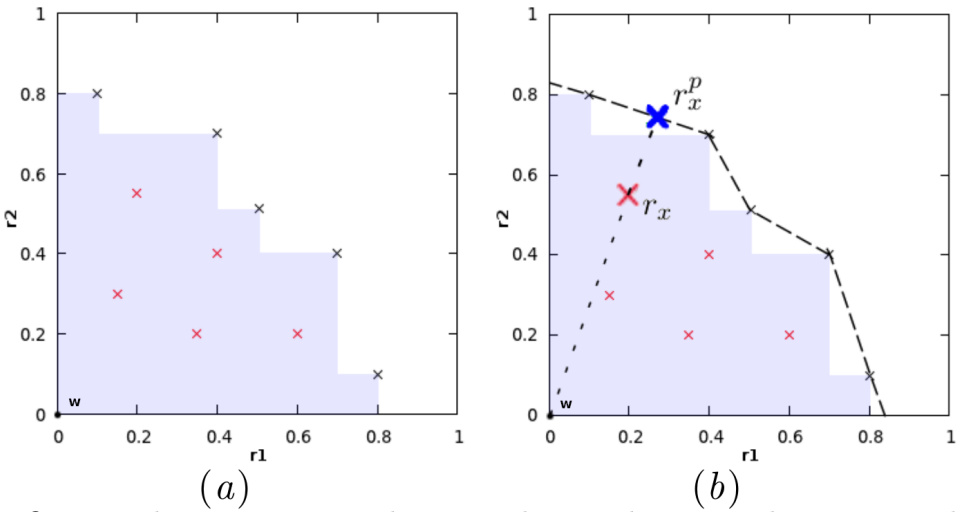


Figure 6.3 – Image provenant de (W. Wang et Michèle Sebag 2012). (a) : Représentation graphique de 10 récompenses vectorielles en deux dimensions (les croix) dont les deux composantes r_1 et r_2 sont à maximiser.

— la projection en perspective d'une récompense vectorielle sur l'enveloppe du front de Pareto permet de calculer le score de l'hypervolume d'une récompense dominée. Sur la figure 6.3 (b), la projection en perspective du vecteur de récompense r_x (croix rouge) est calculée par rapport à w et à l'enveloppe, qui est la surface linéaire par morceaux représentée en traits pointillés. Il s'agit de l'intersection entre la ligne formée en partant de w et passant par r_x et l'enveloppe. Le résultat de la projection est noté r_x^p (croix bleue). La distance euclidienne entre r_x et r_x^p soustraite au calcul de l'hypervolume du front de Pareto permet d'obtenir la valeur de la récompense dominée.

En s'appuyant sur ces concepts, MO-MCTS diffère de MCTS sur quelques aspects seulement. D'abord, à chaque fin d'itération, après l'étape de rétropropagation, la nouvelle récompense vectorielle est comparée aux éléments de l'archive χ. Si elle est non

dominée, alors elle est sauvegardée dans l'archive et les récompenses vectorielles dominées par cette nouvelle récompense sont supprimées de l'archive. Ensuite, la fonction de sélection est traduite en une nouvelle fonction, introduite dans l'équation (6.6). L'indicateur d'hypervolume est utilisé pour calculer cette nouvelle fonction de sélection :

$$Q^{\oplus}_{MO}(s,a) = \begin{cases} HV(\chi \cup \{\overline{r}_{s,a}\}, w) & \text{, si } \overline{r}_{s,a} \text{ est non dominé dans } \chi \\ HV(\chi \cup \{\overline{r}_{s,a}\}, w) - \|\overline{r}^{p}_{s,a} - \overline{r}_{s,a}\|_2 & \text{, sinon} \end{cases} \tag{6.6}$$

où χ est l'archive utilisée pour stocker les récompenses, $w \in \mathbb{R}^d$ est le point de référence de HV, $\overline{r}_{s,a}$ est un vecteur composé de la fonction de sélection usuelle (UCB1, RAVE, *etc.*) appliquée à a dans l'état s pour chaque objectif, et $\overline{r}^{p}_{s,a}$ est la projection en perspective de $\overline{r}_{s,a}$ sur l'enveloppe de χ.

La *tree policy* devient donc :

$$\pi_{MO}(s) = \underset{a \in A(s)}{\operatorname{argmax}}\, Q^{\oplus}_{MO}(s,a) \tag{6.7}$$

Dans ce contexte, nous introduisons le score de diversité en tant qu'objectif supplémentaire. Les deux objectifs sont donc la qualité d'un plan $Q_{plan}(\nu)$, et sa diversité $D(\nu, P)$. Le score de diversité est calculé à l'aide de la mesure de dissimilarité des plans fournie par l'utilisateur, telle que la distance minimale par rapport à n'importe quel plan de l'ensemble existant P. Il peut également être basé sur des mesures qualitatives ou quantitatives. Dans la section suivante, nous détaillons les choix réalisés pour l'étude expérimentale.

6.3 Validation expérimentale

Les contributions proposées sont évaluées sur les RRGHs étudiés dans le but d'obtenir un ensemble de paramétrisations diverses. Le contenu de cette section peut être retrouvé en grande partie dans (Michelucci, Jean-Paul Comet et Pallez 2024).

6.3.1 Cadre expérimental

Dans les expériences réalisées, nous testons quatre planificateurs diversifiés basés sur MCTS (DP-MCTS) :

— DMAB utilise la stratégie basée sur la *tree policy* de l' équation (6.4) ;

— Lock utilise la stratégie verrou présentée dans la section 6.2.1 ;

— DivPW utilise la stratégie qui consiste à modifier l'équation de *progressive widening* introduite dans la section 6.2.1 ;

— DPMO utilise MO-MCTS présenté dans la section 6.2.2.

Ils génèrent un ensemble de plans pour un *maximum* de cinq solutions ($k = |P| \leq 5$), imposent un *minimum* de 100% de la qualité du plan optimal ($q = 1.0$) et une distance minimale entre les plans de $\delta = 10^{-1}$. La mesure de distance entre deux plans choisie est la distance euclidienne et est appliquée sur les actions sélectionnées. Nous faisons le choix de cette mesure, car un ensemble diversifié d'actions conduit toujours à un ensemble diversifié de traces biologiques. Une mesure de dissimilarité basée sur la distance entre états n'est pas gardée, car la mesure peut attribuer une valeur similaire à un plan nouvellement

extrait, même si sa séquence d'actions est différente. Ainsi, la mesure de diversité pour un nouveau plan ν est la distance minimale par rapport à n'importe quel plan de P :

$$D(\nu, P) = inf\{||\nu, \nu_j||_2 \mid \nu_j \in P\} \tag{6.8}$$

Nous avons choisi d'utiliser l'heuristique *continuous* GRAVE (cGRAVE) sur la base des recherches antérieures présentées dans le chapitre 4.

Comme nous l'avons déjà évoqué dans la section 4.4.1, le réglage manuel et *ad hoc* des paramètres d'un algorithme présente de nombreux inconvénients, comme le fait d'être soumis à des biais humains, de prendre du temps et d'être limité par le nombre d'instances du problème. Nous utilisons une procédure automatique pour la configuration des algorithmes : la librairie `irace` (López-Ibáñez et al. 2016). La configuration élite obtenue après 1 000 itérations est conservée pour déterminer les valeurs des paramètres dépendant du problème. Pour configurer cGRAVE, l'espace des paramètres et les valeurs obtenues sont présentées dans le tableau 6.1. La distance euclidienne est choisie pour les espaces d'action et d'état. Chaque expérience est réalisée 30 fois pour obtenir des résultats statistiquement significatifs. Les performances des différents planificateurs sont comparées pour le même budget de calcul, c'est-à-dire 50 000 itérations en 3G et 200 000 en 5G.

Paramètres	$bias$	α_{state}	α_{action}	ζ	ref
Domaine	$\{10^{-15}, ..., 10^{-2}, 0.1\}$	$[\![1, 100]\!]$	$[\![1, 100]\!]$	$\{0.3, 0.31, ..., 0.9\}$	$[\![1, 100]\!]$
Valeurs obtenues	0.1	47	81	0.61	15

Tableau 6.1 – Espace des paramètres et valeurs obtenues pour la configuration de cGRAVE par `irace`.

6.3.2 Analyse des résultats

Tout d'abord, nous nous concentrons sur l'analyse et donc la comparaison de la diversité de l'ensemble des solutions P générées par chaque variant de DP-MCTS. Cette performance est évaluée grâce à la mesure de la somme des distances (SD) définie dans le tableau 6.2. Elle détermine la dissimilarité entre les plans en calculant la racine carrée de la somme de leurs distances respectives :

$$SD(P) = \sqrt{\sum_{\nu_i \in P} \sum_{\substack{\nu_j \in P \\ i<j}} ||\nu_i - \nu_j||_2} \tag{6.9}$$

où ν_i et ν_j sont deux plans distincts dans P.

Le tableau 6.2 représente la moyenne, l'écart-type et l'*extremum* des mesures SD obtenues par chaque planificateur sur les deux instances du problème. Les meilleurs résultats sont en gras. Nous pouvons déduire de ce tableau que, tant en 3G qu'en 5G, DivPW maintient un ensemble plus diversifié de solutions, et Lock trouve la meilleure (compte tenu des 30 exécutions). Néanmoins, la stratégie naïve employée par Lock est opportuniste : les performances moyennes sont inférieures à celles des autres planificateurs pour la 5G. DPMO et DMAB obtiennent des statistiques similaires dans les deux cas. DMAB montre une plus grande variabilité des performances (visible grâce à l'écart-type).

Alg	3G			5G		
	mean ± std	max	min	mean ± std	max	min
DMAB	29.33 ± 3.41	34.9	24.19	93.29 ± 57.35	225.87	18.87
DPMO	23.12 ± 3.31	28.36	16.26	97.91 ± 16.85	115.14	77.02
Lock	33.95 ± 2.96	**39.84**	26.77	50.03 ± 48.68	**226.61**	15.91
DivPW	**34.48 ± 2.42**	39.35	**29.59**	**126.4 ± 24.92**	182.53	**87.6**

Tableau 6.2 – Statistiques de diversité (suivant la mesure de l'équation (6.9)) des ensembles de plans générés par les différents planificateurs sur les instances du problème 3G et 5G. Les valeurs en gras indiquent les meilleurs résultats colonne par colonne.

Dans une seconde étape, nous nous concentrons sur un aspect différent : que se passerait-il si l'ensemble des plans P n'était pas limité ? En fait, du point de vue du modélisateur, le fait d'avoir le plus large choix possible peut être instructif et constituer une caractéristique précieuse. En outre, le nombre de solutions n'est pas nécessairement connu à l'avance. Dans notre contexte, présenter le plus grand échantillon de solutions serait pertinent pour analyser les dynamiques les plus intéressantes parmi les solutions extraites. Plus le nombre de solutions est grand (et diversifié), mieux c'est. C'est la raison pour laquelle, au cours de nos expériences, un second ensemble de plans P' a été introduit pour rassembler tous les plans qui sont à la fois de qualité supérieure et différents ($D(\nu, P') \neq 0$), mais cet ensemble n'est pas borné. On se ramène à un problème top-qualité et nous évaluons les performances de chaque planificateur en termes de diversité. Nous cherchons à déterminer quel planificateur trouve l'ensemble de solutions le plus grand et le plus diversifié. Pour évaluer la diversité de chaque ensemble de plans extrait par un planificateur, nous avons utilisé la mesure de performance suivante, inspirée de celle de (Kronfeld et Zell 2010) :

$$sc(P') = |clust_{\sigma}(P')| \tag{6.10}$$

Ce score utilise une technique de regroupement basée sur la densité qui renvoie le nombre de *clusters*. Chaque plan représentatif des *clusters* trouvés (avec la valeur de qualité la plus élevée) est conservé comme solution optimale, tandis que les autres sont éliminés. Dans cette étude, OPTICS (Ankerst et al. 1999) est paramétré avec $\sigma = \delta = 10^{-1}$ qui est la distance maximale définie qui permet de supprimer la redondance entre deux plans proches.

La figure 6.4 présente les statistiques de diversité calculées avec l'équation (6.9) des ensembles de plans générés par les différents planificateurs sur les instances du problème 3G et 5G. Les valeurs en gras indiquent les meilleurs résultats colonne par colonne. Nous pouvons affirmer qu'en 3G, DivPW surpasse largement les autres planificateurs testés. DPMO et Lock obtiennent de meilleures performances en matière de diversité que DMAB, et leurs performances ne semblent pas différentes. En 5G, la stratégie DMAB est également à la traîne par rapport aux autres stratégies proposées. Cependant, la stratégie DivPW est surpassée par DPMO. Dans l'ensemble, la figure 6.4 démontre l'intérêt d'introduire

les nouveaux variants de DP-MCTS évalués sur les deux instances du problème lors de la recherche d'un ensemble de plans diversifié non borné.

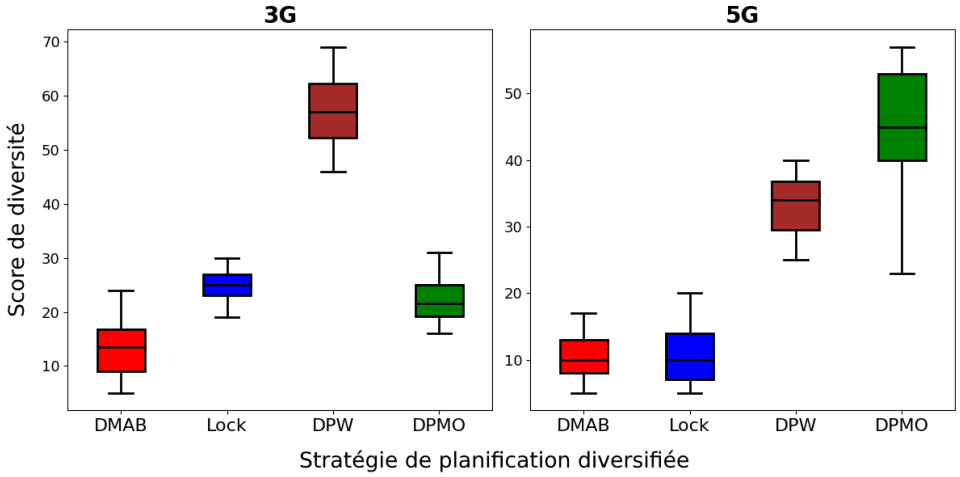


FIGURE 6.4 – Diagrammes en boîte du score de diversité moyen (calculé à partir de l'équation (6.10)) de l'ensemble de plans généré par les planificateurs. Le plus haut, le meilleur.

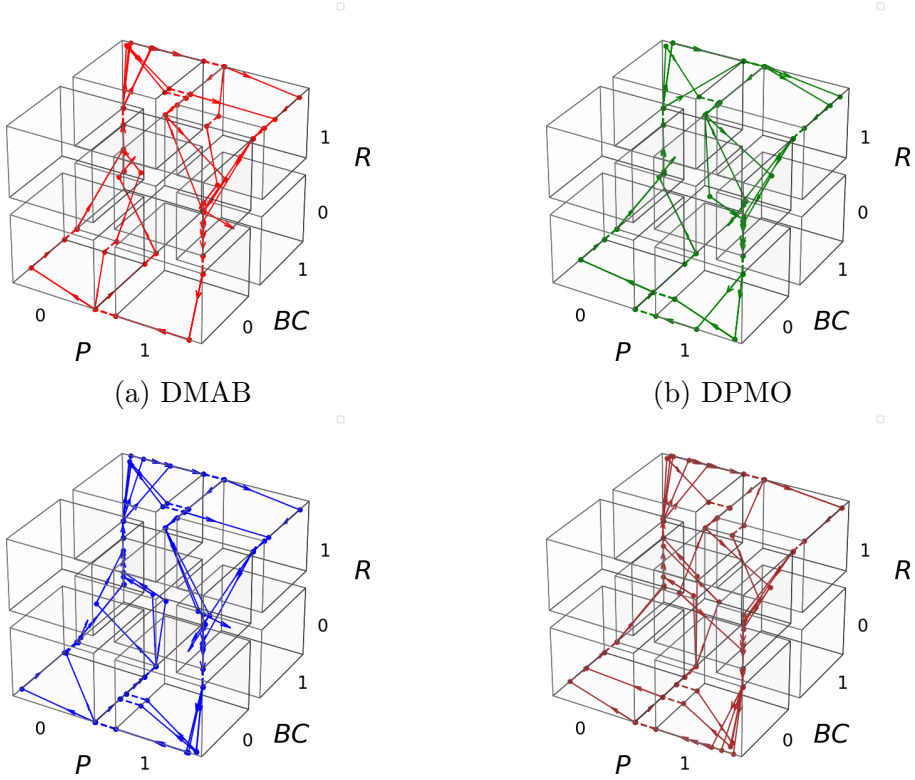
Discussion. D'après les expériences menées sur notre problème, les deux meilleurs planificateurs sont DivPW et DPMO. Comme pour DMAB, l'un des inconvénients de ces planificateurs basés sur des heuristiques est que rien ne prouve que l'exploration oriente la recherche vers des plans diversifiés de haute qualité. Cependant, sur le problème testé, ils ont montré qu'ils présentaient un ensemble de plans avec une plus grande diversité que la méthode de base DMAB.

Suivant la figure 6.4, les résultats obtenus par les planificateurs en 5G sont inférieurs à ceux affichés en 3G, mais ils ne sont pas comparables puisque la dimension de l'espace d'action est différente. En outre, la complexité de l'instance du problème augmente à mesure que le nombre de paramètres à identifier augmente. De même, à partir du tableau 6.2, les scores SD augmentent naturellement entre les deux instances puisqu'il y a un plus grand nombre d'actions à identifier, ce qui conduit à un espace d'action plus grand. Cela expliquerait également la variabilité des résultats des différents planificateurs basés sur des heuristiques.

Il convient de noter que MO-MCTS est connu pour souffrir d'un coût de calcul plus élevé en raison du calcul de l'hypervolume (HV) : le modélisateur pourrait préférer obtenir un ensemble de plans moins diversifiés au profit d'un calcul plus rapide en utilisant DivPW.

6.3.3 Visualisation

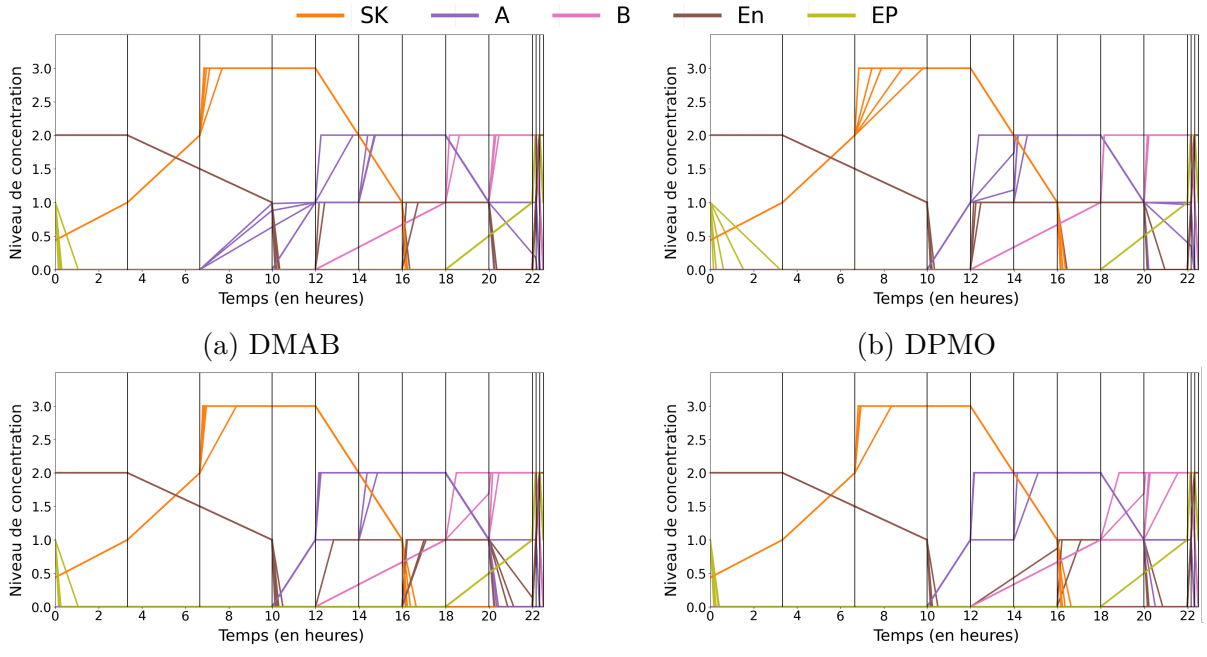
Les figures 6.5 et 6.6 montrent la diversité de l'ensemble des solutions trouvées dans l'ensemble des plans P de chaque planificateur testé sur les deux instances du problème. Elle illustre l'ensemble des solutions générées par chaque planificateur lors d'une seule exécution. Les différents planificateurs proposés ont contribué à l'obtention de solutions



(a) DMAB (b) DPMO

(c) Lock (d) DivPW

FIGURE 6.5 – Ensemble des solutions extraites des différentes stratégies de planification diversifiée sur le cycle circadien (3G).



(a) DMAB (b) DPMO

(c) Lock (d) DivPW

FIGURE 6.6 – Ensemble des solutions extraites avec MCTS sur le cycle cellulaire (5G) en une unique exécution.

diverses, chacune étant cohérente avec la connaissance biologique. Entre les différents ensembles de plans présentés dans la 5G, l'avantage de DPMO en termes de diversité peut être observé. En effet, les célérités des variables SK et EP couvrent mieux l'espace des solutions, en particulier : (i) entre 0 et 4 heures, les pentes des trajectoires de EP (courbes jaunes) sont plus diversifiées, et (ii) entre 6h30 et 10 heures, il en va de même pour les pentes de SK (courbes orange).

6.4 Synthèse

Ce chapitre propose différentes heuristiques dans le but d'orienter la recherche de solutions pour trouver des solutions optimales et diverses avec MCTS. Ces heuristiques sont appliquées dans un contexte où les états et les actions sont continus. Nous les évaluons dans le but d'identifier un ensemble de paramétrisations des modèles hybrides du cycle circadien et du cycle cellulaire. L'ensemble de solutions obtenues est composé d'une diversité de dynamiques cohérentes avec l'expertise biologique.

Le modélisateur peut s'emparer des résultats obtenus avec cette approche pour analyser les différents comportements possibles d'un RRGH et explorer de nouvelles hypothèses de modélisation sur d'autres réseaux. Cette approche est complémentaire à celle introduite dans le chapitre 5 : la combinaison des solutions obtenues permet au modélisateur d'avoir plus de choix. Quelques pistes de réflexion sont à envisager au sujet de la combinaison de certaines heuristiques (DPMO + DivPW, par exemple) pour obtenir un ensemble de solutions plus diverses. Appliquer ces stratégies à des problèmes de la communauté plus typiques permettrait de valider ces résultats expérimentaux. Et finalement, tenter de mettre à profit les techniques de l'optimisation multimodale dans le développement de nouvelles stratégies de planification diversifiée est un axe de recherche à explorer.

Conclusion

La modélisation de réseaux de régulation est une aide précieuse pour l'étude des systèmes biologiques. Elle se présente comme une approche complémentaire aux approches expérimentales en laboratoire. En effet, une meilleure compréhension d'un système biologique passe par l'étude et par la modélisation de ce système.

Modéliser la dynamique d'un système biologique, et plus particulièrement, dans cette thèse, de réseaux de régulation biologique, requiert un cadre de modélisation. Ce cadre de modélisation est choisi en fonction des connaissances biologiques disponibles et du niveau d'abstraction nécessaire. Les travaux entrepris dans cette thèse se placent dans un cadre de modélisation hybride.

L'analyse de la dynamique d'un modèle permet au modélisateur d'extraire des interprétations. Ce dernier peut ainsi émettre des hypothèses et les vérifier en confrontant ses connaissances aux modèles extraits. La dynamique associée au cadre de modélisation hybride choisi repose sur des paramètres continus et se présente comme une trajectoire linéaire par morceau. Cette trajectoire doit rester fidèle à un ensemble d'observations biologiques. Cet ensemble d'observations biologiques est spécifié sous la forme d'un triplet de Hoare hybride. De précédents travaux ont introduit une logique de Hoare hybride dont le calcul de la plus faible précondition permet de construire un ensemble de contraintes minimal. Extraire un modèle valide, c'est satisfaire ces contraintes.

Cette méthode formelle permet de construire automatiquement des contraintes sur les paramètres dynamiques, ce qui a permis de formuler un problème de satisfaction de contraintes. Bien que sa résolution à l'aide de solveurs par contraintes ait été démontrée pour des réseaux de régulation comme le cycle négatif ou le cycle circadien, l'approche est rapidement limitée par la taille du CSP. Dès lors que le nombre de paramètres qui gouvernent la dynamique d'un RRGH augmente, il devient nécessaire de se tourner vers des méthodes heuristiques.

Contributions

L'objectif de cette thèse est de pouvoir obtenir un échantillonnage de l'ensemble des solutions du problème d'identification des paramètres dynamiques d'un RRGH. Nous avons fait le choix dans un premier temps de reformuler le CSP comme un problème d'opti-

misation numérique en boîte noire. Dans ce cadre, la recherche d'une solution globale est équivalente à l'identification d'une paramétrisation respectant les contraintes du CSP et générant un modèle valide. Nous avons introduit une formulation du problème d'optimisation qui se ramène à la minimisation de la distance entre une trace simulée et la connaissance biologique. Dans le but de valider empiriquement cette approche, une étude expérimentale impliquant différentes métaheuristiques d'EA a été menée. Cette démonstration de faisabilité a pris pour exemple le cycle négatif, le cycle circadien et le cycle cellulaire. Bien qu'une solution optimale soit identifiée pour chacun de ces réseaux, il est visible que l'efficacité des optimiseurs stochastiques diminue à mesure que le nombre de paramètres dynamiques augmente. C'est pourquoi une seconde étude expérimentale a été menée dans laquelle des métaheuristiques du domaine de la coévolution font usage de mécanismes spécifiques pour faire face à la malédiction de la dimension.

Du point de vue du modélisateur, extraire une unique paramétrisation optimale n'est pas suffisant. Il est préférable d'avoir à sa disposition un ensemble significativement diversifié. L'objectif du modélisateur est de pouvoir confronter les modèles obtenus *in silico* à ses hypothèses dans le but de les réfuter et de pouvoir raffiner les connaissances du système biologique étudié. Nous avons donc envisagé la résolution du problème d'optimisation comme un problème d'optimisation multimodale. À partir des résultats obtenus lors des précédentes expérimentations, une nouvelle étude expérimentale a été menée impliquant des mécanismes de préservation de la diversité dans le but d'identifier plusieurs *optima* au cours d'une unique exécution. Les résultats extraits ont montré que les algorithmes cellulaires génétiques sont plus efficaces que les métaheuristiques traditionnelles et permettent bien d'extraire plusieurs modèles valides. Les cGAs ont même permis d'obtenir de meilleurs résultats qu'un algorithme vainqueur d'une compétition spécifique à la résolution des problèmes multimodaux. Nous posons l'hypothèse que la spécificité du paysage de fitness de notre problème en est la raison. Nous proposons donc d'introduire de nouvelles fonctions de *benchmarks* ayant cette même spécificité. Ces fonctions sont à généraliser à une dimension quelconque et une réflexion sur la topologie de l'ensemble des solutions est à mener.

Une seconde partie de ce travail de thèse réside dans la formulation du problème d'identification des paramètres dynamiques des réseaux de régulation génétique comme un problème de décision séquentiel. Ce problème est ainsi représenté comme un processus de décision markovien. Nous utilisons la recherche arborescente Monte Carlo dans un cadre où le domaine des états et des actions est continu. Nous proposons des améliorations heuristiques : *continuous generalized rapid action value estimation* (cGRAVE), une politique sélective basée sur les contraintes et une décomposition du vecteur d'action multidimensionnel. Ces différentes améliorations ont été évaluées dans une étude expérimentale impliquant le cycle cellulaire et nous montrons que l'agrégat de ces heuristiques permet de trouver une paramétrisation optimale à chaque exécution.

La recherche d'un ensemble de solutions diverses avec la recherche arborescente Monte Carlo a été introduite très récemment. Ces travaux prennent racine dans le domaine de la planification diversifiée et introduisent notamment un algorithme de bandit permettant de biaiser l'exploration vers des zones prometteuses de l'espace de recherche tout en introduisant un critère de diversité. Nos contributions reposent sur ces avancées et proposent des heuristiques dans le domaine continu : une modification de la stratégie du *progressive widening*, une stratégie d'inhibition et une version de MCTS multi-objectif qui considère

la diversité comme un objectif à part entière. L'étude expérimentale mise en place valide l'intérêt de ces heuristiques pour l'identification d'un ensemble de solutions diverses au cours d'une unique exécution sur les cycles circadien et cellulaire.

Perspectives

Différents axes de recherche sont proposés. Les premiers se placent du point de vue du cadre applicatif en proposant d'étendre certains des travaux entrepris durant cette thèse. Les suivants suggèrent des contributions plus théoriques.

Chronothérapie. Une première perspective à dimension applicative à long terme consisterait à répondre à la question : « À quelle heure dois-je prendre mon médicament ? ». La chronobiologie met en avant l'importance de l'heure de la prise du médicament en considération des rythmes biologiques chez les êtres vivants. La chronothérapie est une branche de la chronobiologie dont l'objectif est d'administrer un médicament donné au meilleur moment de la journée, en fonction des variations temporelles de l'efficacité et de la tolérance des substances médicamenteuses. Il s'agit d'une optimisation temporelle de l'administration des médicaments dans la volonté d'améliorer son effet et de diminuer les effets indésirables sur le patient. En cancérologie, notamment, la chronothérapie permettrait de cibler spécifiquement les cellules cancéreuses en évitant de détruire les cellules saines. Lorsque les connaissances biologiques sur les réseaux de régulation de taille plus importante, comme le couplage du cycle cellulaire et du cycle circadien ou le métabolisme du pancréas, seront disponibles sous la forme d'un triplet de Hoare hybride, il est envisageable d'appliquer les méthodes développées dans le but d'extraire des traces valides et d'étudier l'effet de l'heure d'administration d'un médicament sur le modèle en l'administrant *in silico* à différents moments de la journée.

Génération de nouvelles instances. Seulement trois réseaux sont étudiés. L'espace d'instances de ce problème est donc très faible. La génération de nouvelles instances demande un long travail qui requiert une expertise conjointe en biologie et en modélisation. Cependant, en acceptant de s'abstraire de la connaissance biologique, de nouvelles instances du problème pourraient être générées. L'idée est de transposer le problème dans un cadre général : la recherche d'une trajectoire continue par morceau dans une mosaïque d'hypercubes. Cette trajectoire serait aussi soumise à des contraintes formulées sous la forme d'un triplet de Hoare hybride. Spécifié d'une telle manière, le problème requiert, *a minima*, un nombre de dimensions et un triplet de Hoare contenant des informations générées aléatoirement (de manière cohérente). L'ajout d'un graphe d'interaction avec un nombre de variables et des interactions générées aléatoirement rendrait le problème plus complexe à résoudre, car il serait plus contraint. En effet, comme c'est le cas dans les réseaux 3G et 5G, des contraintes d'égalités peuvent intervenir entre des variables de décisions appartenant à différents hypercubes.

Création d'opérateur génétique spécifique au RRG. Comme présenté dans la section 3.3.2.4, une piste prometteuse serait de travailler sur la création d'un opérateur de réparation spécifique à notre problème. Cet opérateur permettrait de n'échantillonner que

des paramétrisations qui génèrent des traces suivant la bonne séquence d'états discrets. Cela permettrait d'éviter le problème du blocage dans l'équation (3.2). Une partie du budget serait alors économisée pour permettre de prolonger le processus d'optimisation.

Variants de MCTS dans le continu. Dans le domaine de la prise de décision séquentielle, *Nested Monte Carlo Search* et plus particulièrement *Nested Rollout Policy Adaptation* (NRPA) sont indéniablement des variants de MCTS ultra-performants qui ont été appliqués sur de nombreux problèmes combinatoires. De la même manière que GRAVE a été adapté au cadre continu dans le chapitre 4, il serait intéressant d'envisager une version continue de NRPA. On pourrait imaginer y intégrer un mécanisme de *progressive widening* et une politique de *rollout* continue.

Alternatives aux cGAs. Un axe de recherche trivial serait de s'emparer des travaux réalisés dans le contexte d'algorithmes cellulaires génétiques auto-adaptatifs, *self-adaptive cellular genetic algorithms* (sACGAs) (Alba et Bernabè Dorronsoro 2008), dans le but d'améliorer les performances obtenues par les cGAs dans le chapitre 5. L'idée serait d'utiliser les résultats de l'étude réalisée pour initialiser un sACGA avec la structure de voisinage et le ratio obtenus par cGAL9, par exemple.

On pourrait aussi considérer le problème de la recherche de diversité en EA comme un problème multi-objectif, similairement à l'approche de MO-MCTS dans le chapitre 6, où le second critère serait un critère de diversité.

L'utilisation d'algorithmes de *quality diversity* (QD) comme MAP-Elites (Mouret et Clune 2015) a été envisagée au cours de cette thèse. La principale différence avec les algorithmes d'optimisation multimodale réside dans le fait que la diversité des individus n'est pas définie dans un espace génotypique, mais dans un nouvel espace, appelé espace des caractéristiques. Dans le cas d'un robot, une caractéristique pourrait être sa rapidité, une seconde, son poids et une troisième, sa taille. Le but de l'algorithme est alors d'identifier le meilleur individu possible pour chaque combinaison de caractéristiques. Le résultat est une archive de solutions diversifiées et de haute qualité, obtenue en une seule exécution. Cependant, la question de la définition d'un espace des caractéristiques (*feature space*) pour notre problème reste pleinement ouverte.

Multimodalité sur *minimum* global plat. Une piste de recherche dans le domaine de l'évolution artificielle consisterait à investiguer le problème de la multimodalité à la lumière de l'originalité du problème présenté en section 5.4. Le développement d'une suite de fonctions de benchmark, complémentaire à celle introduite dans (X. Li, Andries Engelbrecht et Michael G Epitropakis 2013), permettrait de rendre compte d'une nouvelle facette des problèmes qui peuvent être rencontrés dans des applications de la vie réelle. Leur résolution requiert le développement de nouvelles techniques de *niching*, dont on a vu qu'elles n'étaient pas adaptées à cette nature de fonction objectif, qui pourront potentiellement être bénéfiques pour les sciences de l'ingénieur.

Arbres MCTS en parallèles. Une idée qui n'a pas été expérimentée dans le chapitre 6 serait de considérer le problème de recherche de plusieurs solutions à travers le prisme d'un paradigme multi-agents. En s'inspirant des travaux sur les techniques de parallélisation de MCTS (Cazenave et Jouandeau 2007 ; G. M. J. .-B. Chaslot, Winands et

Herik 2008), une première idée pourrait être simplement d'initialiser une forêt d'arbres indépendants dotés d'une des heuristiques précédemment détaillées. Les n arbres exécutés en parallèle pourraient proposer des ensembles différents de plans optimaux et diversifiés $\mathcal{E} = \{P_1, ..., P_n\}$. À la fin du budget attribué, les différents ensembles générés seraient à fusionner en un unique ensemble Pareto-optimal. Au lieu de réaliser l'opération de fusion à la fin de l'exécution des arbres, une seconde idée pourrait être de l'exécuter de manière périodique. À chaque pas de temps, l'opération de fusion met à jour chaque ensemble de $\mathcal{E}$. Comme spécifié dans (Fern et Lewis 2011), cette méthode peut être parallélisée facilement en exécutant simplement les n appels à UCT sur des processus indépendants et en renvoyant les ensembles générés à un processus central. Cela signifie que si une forêt d'arbres obtient une récompense plus élevée qu'un arbre unique avec le même budget, la parallélisation permettra d'obtenir un ensemble de solutions plus diversifiées sans augmenter le temps d'exécution.

Hybridation solveur par contraintes et heuristique. Une autre piste de recherche pourrait hybrider la méthode de résolution de contraintes avec l'évolution artificielle ou MCTS dans le but de résoudre le problème d'extraction d'un ensemble de paramétrisations diverses dans les RRGHs. L'idée consiste à utiliser le solveur par contraintes pour découper l'espace de recherche. Ce découpage permet d'obtenir un pavage de l'espace en boîtes (*cf.* section 2.4.3). Lorsque le solveur obtient des boîtes sûres, on sait que toute paramétrisation est valide, donc l'utilisation d'une méthode statistique pour l'échantillonnage quasi-aléatoire suffit, comme l'échantillonnage par hypercube latin. Néanmoins, lorsque le solveur renvoie des boîtes non-sûres, il existe au moins une solution, mais toutes les valeurs contenues dans ces boîtes ne sont pas des solutions. Une méthode heuristique développée dans la thèse (algorithmes cellulaires génétiques ou planification diversifiée avec MCTS) peut ainsi être appliquée pour extraire une ou plusieurs solutions par boîte non sûre. Cette approche présente l'avantage de pouvoir pallier le problème d'échantillonnage du solveur tout en réduisant drastiquement le temps computationnel des heuristiques du fait de la réduction de l'espace de recherche. On pourrait d'ailleurs envisager une recherche parallélisée dans laquelle chaque heuristique est appliquée dans une boîte non sûre différente.

Hybridation de MCTS et des métaheuristiques pour les problèmes mixtes. Dans la définition du problème d'optimisation numérique, il est question de résoudre un problème mixte dans lequel on a des variables de type entier et d'autres de type continu. Du fait des connaissances à disposition, nous nous sommes ramenés à un problème continu. Mais si l'on devait considérer le problème mixte, on pourrait travailler sur le développement de méthodes hybrides entre les métaheuristiques et MCTS. Une perspective de recherche consisterait en l'utilisation de MCTS pour l'aspect combinatoire du problème et d'une métaheuristique pour l'aspect continu. Ainsi, on pourrait remplacer l'étape de *rollout* de MCTS par l'évolution d'une population d'un algorithme d'EA. Un nœud de l'arbre posséderait des statistiques liées à cette évolution (en gardant par exemple le meilleur individu). La *tree policy* de MCTS permettrait de guider la recherche dans l'arbre qui ne serait constitué que des variables entières et la métaheuristique permettrait d'aider à identifier les variables continues.

Bibliographie

Abramson, B. (1990). « Expected-Outcome: A General Model of Static Evaluation ». In : *IEEE Transactions on Pattern Analysis and Machine Intelligence* 12.2, p. 182-193. doi : 10.1109/34.44404.

Ahmad, Jamil, Gilles Bernot, Jean-Paul Comet, Didier Lime et Olivier Roux (2007). « Hybrid Modelling and Dynamical Analysis of Gene Regulatory Networks with Delays ». In : *Complexus* 3.4, p. 231-251. doi : 10.1159/000110010.

Ahrari, A., M. Shariat-Panahi et A.A. Atai (2009). « GEM: A Novel Evolutionary Optimization Method with Improved Neighborhood Search ». In : *Applied Mathematics and Computation* 210.2, p. 376-386. doi : 10.1016/j.amc.2009.01.009.

Ahrari, Ali, Kalyanmoy Deb et Mike Preuss (2017). « Multimodal Optimization by Covariance Matrix Self-Adaptation Evolution Strategy with Repelling Subpopulations ». In : *Evolutionary Computation* 25.3, p. 439-471. doi : 10.1162/evco_a_00182.

Ahrari, Ali, Saber Elsayed, Ruhul Sarker, Daryl Essam et Carlos A. Coello Coello (2022). « Static and Dynamic Multimodal Optimization by Improved Covariance Matrix Self-Adaptation Evolution Strategy With Repelling Subpopulations ». In : *IEEE Transactions on Evolutionary Computation* 26.3, p. 527-541. doi : 10.1109/TEVC.2021.3117116.

Alarie, Stéphane, Charles Audet, Aïmen E. Gheribi, Michael Kokkolaras et Sébastien Le Digabel (2021). « Two Decades of Blackbox Optimization Applications ». In : *EURO Journal on Computational Optimization* 9. doi : 10.1016/j.ejco.2021.100011.

Alba, Enrique et Bernabé Dorronsoro (2004). « Solving the Vehicle Routing Problem by Using Cellular Genetic Algorithms ». In : *Evolutionary Computation in Combinatorial Optimization*. Springer, p. 11-20. doi : 10.1007/978-3-540-24652-7_2.

— (2008). « Introduction to Cellular Genetic Algorithms ». In : *Cellular Genetic Algorithms*. Springer. doi : 10.1007/978-0-387-77610-1_1.

Alba, Enrique et Bernabè Dorronsoro (2008). « Design of Self-Adaptive cGAs ». In : *Cellular Genetic Algorithms*. Springer, p. 83-99. doi : 10.1007/978-0-387-77610-1_6.

Alba, Enrique et José M Troya (2002). « Improving Flexibility and Efficiency by Adding Parallelism to Genetic Algorithms ». In : *Statistics and Computing* 12.2, p. 91-114. doi : 10.1023/A:1014803900897.

Alba, Enrique et José Ma Troya (2000). « Cellular Evolutionary Algorithms: Evaluating the Influence of Ratio ». In : *Parallel Problem Solving from Nature PPSN VI*. Springer, p. 29-38. doi : 10.1007/3-540-45356-3_3.

Anafi, Ron C, Yool Lee, Trey K Sato, Anand Venkataraman, Chidambaram Ramanathan, Ibrahim H Kavakli, Michael E Hughes, Julie E Baggs, Jacqueline Growe, Andrew C Liu et al. (2014). « Machine Learning Helps Identify CHRONO as a Circadian Clock Component ». In : *PLoS biology* 12.4. doi : 10.1371/journal.pbio.1001840.

Ankerst, Mihael, Markus M. Breunig, Hans-Peter Kriegel et Jörg Sander (1999). « OPTICS: Ordering Points to Identify the Clustering Structure ». In : *SIGMOD Rec.* 28.2, p. 49-60. doi : 10.1145/304181.304187.

Audet, Charles et Warren Hare (2017). « Introduction: Tools and Challenges in Derivative-Free and Blackbox Optimization ». In : *Derivative-Free and Blackbox Optimization*. Springer, p. 3-14. doi : 10.1007/978-3-319-68913-5_1.

Auer, Peter, Nicolo Cesa-Bianchi et Paul Fischer (2002). « Finite-time Analysis of the Multiarmed Bandit Problem ». In : *Machine learning* 47, p. 235-256. doi : 10.1023/A:1013689704352.

Auger, A. et N. Hansen (2005). « A Restart CMA Evolution Strategy with Increasing Population Size ». In : *2005 IEEE Congress on Evolutionary Computation*. T. 2, p. 1769-1776. doi : 10.1109/CEC.2005.1554902.

Baluja, Shummet (1994). *Population-Based Incremental Learning: A Method for Integrating Genetic Search Based Function Optimization and Competitive Learning*. Rapp. tech.

Behaegel, J., J.-P. Comet et F. Folschette (2017). « Constraint Identification Using Modified Hoare Logic on Hybrid Models of Gene Networks ». In : *Proceedings of the 24th International Symposium on Temporal Representation and Reasoning*. T. 90. doi : 10.4230/LIPIcs.TIME.2017.5.

Behaegel, J., J.-P. Comet et M. Pelleau (2018). « Identification of Dynamic Parameters for Gene Networks ». In : *Proceedings of the 30th IEEE International Conference on Tools with Artificial Intelligence*, p. 122-129. doi : 10.1109/ICTAI.2018.00028.

Behaegel, Jonathan (2018). « Modèles Hybrides de Réseaux de Régulation: Étude du Couplage des Cycles Cellulaire et Circadien ». Thèse de doct. Université Côte d'Azur. url : https://theses.hal.science/tel-01962051.

Behaegel, Jonathan, Jean-Paul Comet, Gilles Bernot, Emilien Cornillon et Franck Delaunay (2016). « A Hybrid Model of Cell Cycle in Mammals ». In : *Journal of Bioinformatics and Computational Biology* 14.01. doi : 10.1142/S0219720016400011.

Bellman, R.E. (1957). *Dynamic Programming*. Princeton University Press.

Benke, Lyndon, Tim Miller, Michael Papasimeon et Nir Lipovetzky (2023). « Diverse, Top-k, and Top-Quality Planning Over Simulators ». In : *26th European Conference on Artificial Intelligence*, p. 231-238. doi : 10.3233/FAIA230275.

Bergh, F. van den et A.P. Engelbrecht (2004). « A Cooperative Approach to Particle Swarm Optimization ». In : *IEEE Transactions on Evolutionary Computation* 8.3, p. 225-239. doi : 10.1109/TEVC.2004.826069.

Bernot, Gilles, Jean-Paul Comet et Olivier Roux (2015). « A Genetically Modified Hoare Logic that Identifies the Parameters of a Gene Network ». In : *Computational Methods in Systems Biology*. Springer, p. 8-12. doi : 10.1007/978-3-319-23401-4_2.

Bertsekas, Dimitri et John N Tsitsiklis (1996). *Neuro-Dynamic Programming*. Athena Scientific.

Beyer, Hans-Georg et Bernhard Sendhoff (2008). « Covariance Matrix Adaptation Revisited – The CMSA Evolution Strategy – ». In : *Parallel Problem Solving from Nature – PPSN X*. Springer, p. 123-132. doi : 10.1007/978-3-540-87700-4_13.

Blank, Julian et Kalyanmoy Deb (2020). « Pymoo: Multi-Objective Optimization in Python ». In : *IEEE Access* 8, p. 89497-89509. doi : 10.1109/ACCESS.2020.2990567.

Blum, Christian et Andrea Roli (2003). « Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison ». In : *ACM Comput. Surv.* 35.3, p. 268-308. doi : 10.1145/937503.937505.

Boddy, Mark, Johnathan Gohde, Tom Haigh et Steven Harp (2005). « Course of Action Generation for Cyber Security using Classical Planning ». In : *Proceedings of the Fifteenth International Conference on International Conference on Automated Planning and Scheduling*. AAAI Press, p. 12-21.

Boutilier, Craig, Thomas Dean et Steve Hanks (1999). « Decision-Theoretic Planning: Structural Assumptions and Computational Leverage ». In : *Journal of Artificial Intelligence Research* 11, p. 1-94. doi : 10.1613/jair.575.

Brailsford, Sally C., Chris N. Potts et Barbara M. Smith (1999). « Constraint Satisfaction Problems: Algorithms and Applications ». In : *European Journal of Operational Research* 119.3, p. 557-581. doi : 10.1016/S0377-2217(98)00364-6.

Browne, Cameron B., Edward Powley, Daniel Whitehouse, Simon M. Lucas, Peter I. Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis et Simon Colton (2012). « A Survey of Monte Carlo Tree Search Methods ». In : *IEEE Transactions on Computational Intelligence and AI in Games* 4.1, p. 1-43. doi : 10.1109/TCIAIG.2012.2186810.

Brügmann, Bernd (1993). *Monte Carlo Go*. Rapp. tech. Physics Department, Syracuse University, NY.

Buchet, Samuel, Francesco Carbone, Morgan Magnin, Mickaël Ménager et Olivier Roux (2021). « Inference of Gene Networks from Single Cell Data through Quantified Inductive Logic Programming ». In : *The 12th International Conference on Computational Systems-Biology and Bioinformatics*. CSBio2021. Association for Computing Machinery, p. 48-63. doi : 10.1145/3486713.3486746.

Bugajska, M.D. et A.C. Schultz (2002). « Coevolution of Form and Function in the Design of Micro Air Vehicles ». In : *Proceedings 2002 NASA/DoD Conference on Evolvable Hardware*, p. 154-163. doi : 10.1109/EH.2002.1029881.

Campelo, Felipe et Claus Aranha (2023). « Lessons from the Evolutionary Computation Bestiary ». In : *Artificial Life* 29.4, p. 421-432. doi : 10.1162/artl_a_00402.

Cavicchio, Daniel Joseph (1970). *Adaptive Search using Simulated Evolution*. Rapp. tech.

Cazenave, Tristan (2015). « Generalized Rapid Action Value Estimation ». In : *24th International Joint Conference on Artificial Intelligence*, p. 754-760.

— (2017). « Nested Rollout Policy Adaptation with Selective Policies ». In : *Computer Games*. Springer, p. 44-56. doi : 10.1007/978-3-319-57969-6_4.

Cazenave, Tristan et Nicolas Jouandeau (2007). « On the Parallelization of UCT ». In : *Computer Games Workshop*.

Chaslot, Guillaume M Jb, Mark HM Winands, H Jaap van den Herik, Jos WHM Uiterwijk et Bruno Bouzy (2008). « Progressive Strategies for Monte-Carlo Tree Search ». In : *New Mathematics and Natural Computation* 04.03, p. 343-357. doi : 10.1142/S1793005708001094.

Chaslot, Guillaume M. J. -B., Mark H. M. Winands et H. Jaap van den Herik (2008). « Parallel Monte-Carlo Tree Search ». In : *Computers and Games*. Springer, p. 60-71. doi : 10.1007/978-3-540-87608-3_6.

Collet, Pierre, Evelyne Lutton, Frédéric Raynal et Marc Schoenauer (2000). « Polar IFS+Parisian Genetic Programming=Efficient IFS Inverse Problem Solving ». In : *Genetic Programming and Evolvable Machines* 1, p. 339-361. doi : 10.1023/A:1010065123132.

Colson, Benoît et Philippe L. Toint (2005). « Optimizing Partially Separable Functions Without Derivatives ». In : *Optimization Methods and Software* 20.4-5, p. 493-508. doi : 10.1080/10556780500140227.

Coman, Alexandra et Hector Munoz-Avila (2011). « Generating Diverse Plans Using Quantitative and Qualitative Plan Distance Metrics ». In : *Proceedings of the AAAI Conference on Artificial Intelligence* 25.1, p. 946-951. doi : 10.1609/aaai.v25i1.8006.

Comet, Jean-Paul, Gilles Bernot, Aparna Das, Francine Diener, Camille Massot et Amélie Cessieux (2012). « Simplified Models for the Mammalian Circadian Clock ». In : *Procedia Computer Science* 11. Proceedings of the 3rd International Conference on Computational Systems-Biology and Bioinformatics, p. 127-138. doi : 10.1016/j.procs.2012.09.014.

Conn, Andrew R, Katya Scheinberg et Luis N Vicente (2009). *Introduction to Derivative-Free Optimization*. SIAM.

Cornillon, Emilien (2017). « Modèles Qualitatifs de Réseaux Génétiques: Réduction de Modèles et Introduction d'un Temps Continu ». Thèse de doct. Université Côte d'Azur. url : https://theses.hal.science/tel-01682212.

Couetoux, Adrien (2013). « Monte Carlo Tree Search for Continuous and Stochastic Sequential Decision Making Problems ». Thèse de doct. Université Paris Sud-Paris XI. url : https://theses.hal.science/tel-00927252.

Couëtoux, Adrien, Mario Milone, Mátyás Brendel, Hassan Doghmen, Michèle Sebag et Olivier Teytaud (2011). « Continuous Rapid Action Value Estimates ». In : *Proceedings of the Asian Conference on Machine Learning*. T. 20. Proceedings of Machine Learning Research. PMLR, p. 19-31.

Coulom, Rémi (2007a). « Computing Elo Ratings of Move Patterns in the Game of Go ». In : *ICGA journal*, p. 198-208.

— (2007b). « Efficient Selectivity and Backup Operators in Monte-Carlo Tree Search ». In : *Computers and Games*. Springer, p. 72-83. doi : 10.1007/978-3-540-75538-8_7.

Cousot, Patrick et Radhia Cousot (1992). « Abstract Interpretation Frameworks ». In : *Journal of Logic and Computation* 2.4, p. 511-547. doi : 10.1093/logcom/2.4.511.

Coutinho, Ricardo, Bastien Fernandez, Ricardo Lima et Arnaud Meyroneinc (2006). « Discrete Time Piecewise Affine Models of Genetic Regulatory Networks ». In : *Journal of Mathematical Biology* 52, p. 524-570. doi : 10.1007/s00285-005-0359-x.

Darwin, Charles (1859). *On the Origin of Species by Means of Natural Selection, or the Preservation of Favoured Races in the Struggle for Life.* London: John Murray.

Darwin, Charles et Alfred Wallace (1858). « On the Tendency of Species to form Varieties; and on the Perpetuation of Varieties and Species by Natural Means of Selection. » In : *Journal of the Proceedings of the Linnean Society of London. Zoology* 3.9, p. 45-62. doi : 10.1111/j.1096-3642.1858.tb02500.x.

Das, Swagatam, Sayan Maity, Bo-Yang Qu et P.N. Suganthan (2011). « Real-Parameter Evolutionary Multimodal Optimization — A Survey of the State-Of-The-Art ». In : *Swarm and Evolutionary Computation* 1.2, p. 71-88. doi : 10.1016/j.swevo.2011.05.005.

Das, Swagatam et Ponnuthurai Nagaratnam Suganthan (2011). « Differential Evolution: A Survey of the State-of-the-Art ». In : *IEEE Transactions on Evolutionary Computation* 15.1, p. 4-31. doi : 10.1109/TEVC.2010.2059031.

Davidich, Maria et Stefan Bornholdt (2008). « The Transition from Differential Equations to Boolean Networks: A Case Study in Simplifying a Regulatory Network Model ». In : *Journal of Theoretical Biology* 255.3, p. 269-277. doi : 10.1016/j.jtbi.2008.07.020.

Dawar, Deepak et Simone A. Ludwig (2014). « Differential Evolution with Dither and Annealed Scale Factor ». In : *IEEE Symposium on Differential Evolution (SDE)*, p. 1-8. doi : 10.1109/SDE.2014.7031528.

De Jong, KA (1975). « An Analysis of the Behavior of a Class of Genetic Adaptive Systems ». Thèse de doct. University of Michigan.

De Jong, Kenneth (2017). « Evolutionary Computation: A Unified Approach ». In : *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. Association for Computing Machinery, p. 373-388. doi : 10.1145/3067695.3067715.

Deb, Kalyanmoy et David E Goldberg (1989). « An Investigation of Niche and Species Formation in Genetic Function Optimization ». In : *Proceedings of the third international conference on Genetic algorithms*, p. 42-50.

Della Cioppa, Antonio, Angelo Marcelli et Prisco Napoli (2011). « Speciation in Evolutionary Algorithms: Adaptive Species Discovery ». In : *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation*. Association for Computing Machinery, p. 1053-1060. doi : 10.1145/2001576.2001719.

Deng, Shuiguang, Bin Wu, Jianwei Yin et Zhaohui Wu (2013). « Efficient Planning for Top-K Web Service Composition ». In : *Knowledge and Information Systems* 36, p. 579-605. doi : 10.1007/s10115-012-0541-6.

Dijkstra, Edsger W. (1975). « Guarded Commands, Nondeterminacy and Formal Derivation of Programs ». In : *Commun. ACM* 18.8, p. 453-457. doi : 10.1145/360933.360975.

Doya, Kenji (1995). « Temporal Difference Learning in Continuous Time and Space ». In : *Advances in Neural Information Processing Systems*. T. 8. MIT Press.

Eftimov, Tome et Peter Korošec (2021). « Statistical Analyses for Meta-Heuristic Stochastic Optimization Algorithms ». In : *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. Association for Computing Machinery, p. 770-785. doi : 10.1145/3449726.3461438.

Eiben, A. E. et J. E. Smith (2015a). « Constraint Handling ». In : *Introduction to Evolutionary Computing*. Springer, p. 203-213. doi : 10.1007/978-3-662-44874-8_13.

Eiben, A. E. et J. E. Smith (2015b). *Introduction to Evolutionary Computing*. Springer. doi : 10.1007/978-3-662-44874-8.

Ester, Martin, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu et al. (1996). « A Density-based Algorithm for Discovering Clusters in Large Spatial Databases with Noise ». In : *AAAI*. T. 96. 34, p. 226-231.

Fern, Alan et Paul Lewis (2011). « Ensemble Monte-Carlo Planning: An Empirical Study ». In : *Proceedings of the International Conference on Automated Planning and Scheduling* 21.1, p. 58-65. doi : 10.1609/icaps.v21i1.13458.

Finnsson, Hilmar et Yngvi Björnsson (2010). « Learning Simulation Control in General Game-Playing Agents ». In : t. 24. 1, p. 954-959. doi : 10.1609/aaai.v24i1.7651.

Fletcher, Roger (2000). *Practical Methods of Optimization*. John Wiley & Sons.

Floudas, Christodoulos A et Panos M Pardalos (2008). *Encyclopedia of Optimization*. Springer Science & Business Media.

Fogel, David B (1995). « Toward a New Philosophy of Machine Intelligence ». In : *IEEE Evolutionary Computation* 1080.

— (1998). « Evolutionary Computation. The Fossil Record. Selected Readings on the History of Evolutionary Computation ». In : *Classifier Systems*.

Fogel, Lawrence J (1962). « Autonomous Automata ». In : *Industrial research* 4, p. 14-19.

Gelly, Sylvain et David Silver (2007). « Combining Online and Offline Knowledge in UCT ». In : *Proceedings of the 24th International Conference on Machine Learning*. Association for Computing Machinery, p. 273-280. doi : 10.1145/1273496.1273531.

— (2011). « Monte-Carlo Tree Search and Rapid Action Value Estimation in Computer Go ». In : *Artificial Intelligence* 175.11, p. 1856-1875. doi : 10.1016/j.artint.2011.03.007.

Gelly, Sylvain et Yizao Wang (2006). « Exploration Exploitation in Go: UCT for Monte-Carlo Go ». In : *NIPS: Neural Information Processing Systems Conference On-line trading of Exploration and Exploitation Workshop*.

Gérard, Claude et Albert Goldbeter (2012). « Entrainment of the Mammalian Cell Cycle by the Circadian Clock: Modeling Two Coupled Cellular Rhythms ». In : *PLOS Computational Biology* 8.5, p. 1-21. doi : 10.1371/journal.pcbi.1002516.

Glass, Leon et Stuart A. Kauffman (1972). « Co-operative Components, Spatial localization and Oscillatory Cellular Dynamics ». In : *Journal of Theoretical Biology* 34.2, p. 219-237. doi : 10.1016/0022-5193(72)90157-9.

Glover, Fred (1986). « Future Paths for Integer Programming and Links to Artificial Intelligence ». In : *Computers & Operations Research* 13.5, p. 533-549. doi : 10.1016/0305-0548(86)90048-1.

Glover, Fred et Manuel Laguna (1998). « Tabu Search ». In : *Handbook of Combinatorial Optimization: Volume1–3*. Springer, p. 2093-2229. doi : 10.1007/978-1-4613-0303-9_33.

Goldberg, David E, Jon Richardson et al. (1987). « Genetic Algorithms with Sharing for Multimodal Function Optimization ». In : *Genetic Algorithms and their Applications: Proceedings of the Second International Conference on Genetic Algorithms*. T. 4149, p. 414-425.

Gotoh, Tetsuya, Jae Kyoung Kim, Jingjing Liu, Marian Vila-Caballer, Philip E Stauffer, John J Tyson et Carla V Finkielstein (2016). « Model-Driven Experi-

mental Approach Reveals the Complex Regulatory Distribution of p53 by the Circadian Factor Period 2 ». In : *Proceedings of the National Academy of Sciences* 113.47, p. 13516-13521. DOI : 10.1073/pnas.1607984113.

GRANGER, Gilles Gaston et Henri ATLAN (1991). « Science, Idéologie, Philosophie ». In : *Raison présente* 97.1, p. 89-119.

GRATALOUP, G., D. PALLEZ, G. BERNOT et J.-P. COMET (2024). « Single-objective Constrained Optimization for Gene Regulatory Networks Modeling ». In : *Proceedings of the 16th International Conference on Artificial Evolution*.

HAN, Yixing, Shouguo GAO, Kathrin MUEGGE, Wei ZHANG et Bing ZHOU (2015). « Advanced Applications of RNA Sequencing and Challenges ». In : *Bioinformatics and Biology Insights* 9s1, BBI.S28991. DOI : 10.4137/BBI.S28991.

HANSEN, Nikolaus et Andreas OSTERMEIER (2001). « Completely Derandomized Self-Adaptation in Evolution Strategies ». In : *Evolutionary Computation* 9.2, p. 159-195. DOI : 10.1162/106365601750190398.

HELMBOLD, David P et Aleatha PARKER-WOOD (2009). « All-Moves-As-First Heuristics in Monte-Carlo Go. » In : t. 2. Citeseer, p. 605-610.

HOARE, C. A. R. (1969). « An Axiomatic Basis for Computer Programming ». In : *Commun. ACM* 12.10, p. 576-580. DOI : 10.1145/363235.363259.

HOLLAND, John H (1975). *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. The University of Michigan Press.

HOLLANDER, Myles, Douglas A WOLFE et Eric CHICKEN (2013). *Nonparametric Statistical Methods*. John Wiley & Sons Inc.

HORN, Jeffrey (1997). *The Nature of Niching: Genetic Algorithms and the Evolution of Optimal, Cooperative Populations*. University of Illinois at Urbana-Champaign.

HUGHEY, Jacob J (2017). « Machine Learning Identifies a Compact Gene Set for Monitoring the Circadian Clock in Human Blood ». In : *Genome Medicine* 9.19, p. 1-11. DOI : 10.1186/s13073-017-0406-4.

HUYNH-THU, Vân Anh et Guido SANGUINETTI (2019). « Gene Regulatory Network Inference: An Introductory Survey ». In : *Gene Regulatory Networks: Methods and Protocols*. Springer, p. 1-23. DOI : 10.1007/978-1-4939-8882-2_1.

ISHIBUCHI, Hisao, Noritaka TSUKAMOTO et Yusuke NOJIMA (2008). « Evolutionary Many-Objective Optimization: A Short Review ». In : *IEEE Congress on Evolutionary Computation*, p. 2419-2426. DOI : 10.1109/CEC.2008.4631121.

JAMSHIDI, Shahrad (2013). « Comparing Discrete, Continuous and Hybrid Modelling Approaches of Gene Regulatory Networks ». Thèse de doct. Freie Universität Berlin. DOI : 10.17169/refubium-8209.

KATZ, Michael, Shirin SOHRABI et Octavian UDREA (2020). « Top-Quality Planning: Finding Practically Useful Sets of Best Plans ». In : *Proceedings of the AAAI Conference on Artificial Intelligence* 34.06, p. 9900-9907. DOI : 10.1609/aaai.v34i06.6544.

KAUFFMAN, S.A. (1969). « Metabolic Stability and Epigenesis in Randomly Constructed Genetic Nets ». In : *Journal of Theoretical Biology* 22.3, p. 437-467. DOI : 10.1016/0022-5193(69)90015-0.

KAUR, Manpreet et Rajni BALA (2013). « Chronotherapy: A Review ». In : *International Journal of Pharmaceutical Sciences and Research* 4.1, p. 90. DOI : 10.13040/IJPSR.0975-8232.4(1).90-02.

Kennedy, J. et R. Eberhart (1995). « Particle Swarm Optimization ». In : *Proceedings of ICNN'95 - International Conference on Neural Networks*. T. 4, p. 1942-1948. doi : 10.1109/ICNN.1995.488968.

Khalis, Zohra, Jean-Paul Comet, Adrien Richard et Gilles Bernot (2009). « The SMBioNet Method for Discovering Models of Gene Regulatory Networks ». In : *Genes, genomes and genomics* 3.1, p. 15-22.

Khoodeeram, Rajeev, Gilles Bernot et Jean-Yves Trosset (2017). « An Ockham Razor Model of Energy Metabolism ». In : *Advances in Systems and Synthetic Biology*, p. 1925.

Kocsis, Levente et Csaba Szepesvári (2006). « Bandit Based Monte-Carlo Planning ». In : *Machine Learning: ECML 2006*. Springer, p. 282-293. doi : 10.1007/11871842_29.

Koza, John R (1994). « Genetic Programming as a Means for Programming Computers by Natural Selection ». In : *Statistics and computing* 4, p. 87-112. doi : 10.1007/BF00175355.

Krawiec, Krzysztof et Malcolm Heywood (2019). « Solving Complex Problems with Coevolutionary Algorithms ». In : *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. Association for Computing Machinery, p. 975-1001. doi : 10.1145/3319619.3323384.

Kronfeld, Marcel, Andreas Dräger, Moritz Aschoff et Aandreas Zell (2009). « On the Benefits of Multimodal Optimization for Metabolic Network Modeling ». In : *German Conference on Bioinformatics*. Gesellschaft für Informatik e.V.

Kronfeld, Marcel et Andreas Zell (2010). « Towards Scalability in Niching Methods ». In : *IEEE Congress on Evolutionary Computation*, p. 1-8. doi : 10.1109/CEC.2010.5585916.

Latha, K, MU Uhumwangho, SA Sunil, MV Srikanth et KV Ramana Murthy (2010). « Chronobiology and Chronotherapy of Hypertension–A review ». In : *International Journal of Health Research* 3.3, p. 121-131. doi : 10.4314/ijhr.v3i3.70276.

Le, Hai-Son, Marcel H. Schulz, Brenna M. McCauley, Veronica F. Hinman et Ziv Bar-Joseph (2013). « Probabilistic Error Correction for RNA Sequencing ». In : *Nucleic Acids Research* 41.10, e109-e109. doi : 10.1093/nar/gkt215.

Le Riche, Rodolphe, Marc Schoenauer et Michèle Sebag (2007). « Un état des lieux de l'optimisation évolutionnaire et de ses implications en sciences pour l'ingénieur ». In : *Modélisation numérique. 2, Défis et perspectives. Traité MIM, série Méthodes numériques et éléments finis*, p. 187-259.

Leloup, Jean-Christophe et Albert Goldbeter (2008). « Modeling the Circadian Clock: From Molecular Mechanism to Physiological Disorders ». In : *BioEssays* 30.6, p. 590-600. doi : 10.1002/bies.20762.

Lévi, Francis (2006). « Chronotherapeutics: The Relevance of Timing in Cancer Therapy ». In : *Cancer causes & control* 17, p. 611-621. doi : 10.1007/s10552-005-9004-7.

Li, Jian-Ping, Marton E. Balazs, Geoffrey T. Parks et P. John Clarkson (2002). « A Species Conserving Genetic Algorithm for Multimodal Function Optimization ». In : *Evolutionary Computation* 10.3, p. 207-234. doi : 10.1162/106365602760234081.

Li, Xiaodong (2014). « Decomposition and Cooperative Coevolution Techniques for Large Scale Global Optimization ». In : *Proceedings of the Companion Publication of the*

2014 Annual Conference on Genetic and Evolutionary Computation. Association for Computing Machinery, p. 819-838. DOI : 10.1145/2598394.2605360.

LI, Xiaodong, Andries ENGELBRECHT et Michael G EPITROPAKIS (2013). « Benchmark Functions for CEC'2013 Special Session and Competition on Niching Methods for Multimodal Function Optimization ». In : *RMIT University, Evolutionary Computation and Machine Learning Group, Australia, Tech. Rep.*

LI, Xiaodong, Michael G. EPITROPAKIS, Kalyanmoy DEB et Andries ENGELBRECHT (2017). « Seeking Multiple Solutions: An Updated Survey on Niching Methods and Their Applications ». In : *IEEE Transactions on Evolutionary Computation* 21.4, p. 518-538. DOI : 10.1109/TEVC.2016.2638437.

LI, Zengcong, Tianhe GAO, Kuo TIAN et Bo WANG (2023). « Elite-Driven Surrogate-Assisted CMA-ES Algorithm by Improved Lower Confidence Bound Method ». In : *Engineering with Computers* 39, p. 2543-2563. DOI : 10.1007/s00366-022-01642-5.

LITTMAN, Michael Lederman (1996). « Algorithms for Sequential Decision-Making ». Thèse de doct. Brown University.

LIU, Jing, Ruhul SARKER, Saber ELSAYED, Daryl ESSAM et Nurhadi SISWANTO (2024). « Large-scale Evolutionary Optimization: A Review and Comparative Study ». In : *Swarm and Evolutionary Computation* 85, p. 101466. DOI : 10.1016/j.swevo.2023.101466.

LOCKE, John (1996). *Some Thoughts Concerning Education: and, of the Conduct of the Understanding*. Hackett Publishing.

LÓPEZ-IBÁÑEZ, Manuel, Jérémie DUBOIS-LACOSTE, Leslie PÉREZ CÁCERES, Thomas STÜTZLE et Mauro BIRATTARI (2016). « The irace package: Iterated Racing for Automatic Algorithm Configuration ». In : *Operations Research Perspectives* 3, p. 43-58. DOI : 10.1016/j.orp.2016.09.002.

LUTTON, É., S. AL-MALIKI, J. LOUCHET, A. TONDA et F. P. VIDAL (2023). « Fine-Grained Cooperative Coevolution in a Single Population: Between Evolution and Swarm Intelligence ». In : *Artificial Evolution*. Springer, p. 103-117. DOI : 10.1007/978-3-031-42616-2_8.

MA, Xiaoliang, Xiaodong LI, Qingfu ZHANG, Ke TANG, Zhengping LIANG, Weixin XIE et Zexuan ZHU (2019). « A Survey on Cooperative Co-Evolutionary Algorithms ». In : *IEEE Transactions on Evolutionary Computation* 23.3, p. 421-441. DOI : 10.1109/TEVC.2018.2868770.

MAHFOUD, Samir W et al. (1992). « Crowding and Preselection Revisited. » In : *PPSN*. T. 2, p. 27-36.

MAHFOUD, Samir W (1995). *Niching Methods for Genetic Algorithms*. University of Illinois at Urbana-Champaign.

MAREE, S. C., T. ALDERLIESTEN, D. THIERENS et P. A. N. BOSMAN (2018). « Real-Valued Evolutionary Multi-Modal Optimization driven by Hill-Valley Clustering ». In : *Proceedings of the Genetic and Evolutionary Computation Conference*. Association for Computing Machinery, p. 857-864. DOI : 10.1145/3205455.3205477.

MATYAS, J et al. (1965). « Random Optimization ». In : *Automation and Remote control* 26.2, p. 246-253.

MCINNES, Leland, John HEALY et James MELVILLE (2020). *UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction*. URL : https://arxiv.org/abs/1802.03426.

Mengshoel, Ole J et David E Goldberg (1999). « Probabilistic Crowding: Deterministic Crowding with Probabilistic Replacement ». In : doi : 10.1184/R1/6710189.v1.

Michalewicz, Zbigniew (1996). « GAs: Why Do They Work? » In : *Genetic Algorithms + Data Structures = Evolution Programs*. Springer, p. 45-55. doi : 10.1007/978-3-662-03315-9_4.

Michelucci, Romain, Vincent Callegari, Jean-Paul Comet et Denis Pallez (2024). « Cellular Genetic Algorithms for Identifying Variables in Hybrid Gene Regulatory Networks ». In : *Applications of Evolutionary Computation*. Springer, p. 131-145. doi : 10.1007/978-3-031-56852-7_9.

Michelucci, Romain, Jean-Paul Comet et Denis Pallez (2022). « Evolutionary Continuous Optimization of Hybrid Gene Regulatory Networks ». In : *Artificial Evolution*. Springer, p. 159-172. doi : 10.1007/978-3-031-42616-2_12.

— (2024). « Comparing Diverse Planning Strategies with Continuous Monte Carlo Tree Search applied to hybrid Gene Regulatory Networks ». In : *Proceedings of the 36th IEEE International Conference on Tools with Artificial Intelligence*. accepted.

Michelucci, Romain, Denis Pallez, Tristan Cazenave et Jean-Paul Comet (2024). « Improving Continuous Monte Carlo Tree Search for Identifying Parameters in Hybrid Gene Regulatory Networks ». In : *Parallel Problem Solving from Nature – PPSN XVIII*. Springer, p. 319-334. doi : 10.1007/978-3-031-70085-9_20.

Miguel Antonio, Luis et Carlos A. Coello Coello (2018). « Coevolutionary Multiobjective Evolutionary Algorithms: Survey of the State-of-the-Art ». In : *IEEE Transactions on Evolutionary Computation* 22.6, p. 851-865. doi : 10.1109/TEVC.2017.2767023.

Molina, Daniel, Javier Poyatos, Javier Del Ser, Salvador García, Amir Hussain et Francisco Herrera (2020). « Comprehensive Taxonomies of Nature-and Bio-inspired Optimization: Inspiration Versus Algorithmic Behavior, Critical Analysis Recommendations ». In : *Cognitive Computation* 12, p. 897-939. doi : 10.1007/s12559-020-09730-8.

Morgat, Anne et François Rechenmann (2002). « Modélisation des Données Biologiques - Bio-Informatique (2) ». In : *médecine/sciences* 18, p. 366-374. doi : 10.1051/medsci/2002183366.

Mouret, Jean-Baptiste et Jeff Clune (2015). « Illuminating Search Spaces by Mapping Elites ». In : *arXiv preprint arXiv:1504.04909*.

Myers, Karen L. et Thomas J. Lee (1999). « Generating Qualitatively Different Plans through Metatheoretic Biases ». In : *Proceedings of the Sixteenth National Conference on Artificial Intelligence and the Eleventh Innovative Applications of Artificial Intelligence Conference Innovative Applications of Artificial Intelligence*. American Association for Artificial Intelligence, p. 570-576.

Nguyen, Tuan Anh, Minh Do, Alfonso Emilio Gerevini, Ivan Serina, Biplav Srivastava et Subbarao Kambhampati (2012). « Generating Diverse Plans to Handle Unknown and Partially Known User Preferences ». In : *Artificial Intelligence* 190, p. 1-31. doi : 10.1016/j.artint.2012.05.005.

Nix, Allen E et Michael D Vose (1992). « Modeling Genetic Algorithms with Markov Chains ». In : *Annals of Mathematics and Artificial Intelligence* 5.1, p. 79-88. doi : 10.1007/BF01530781.

Olariu, Stephan et Albert Y Zomaya (2005). *Handbook of Bioinspired Algorithms and Applications*.

Ollerton, Jeff (2006). « Biological Barter": Patterns of Specialization Compared Across Different Mutualisms ». In : *Plant-pollinator interactions: from specialization to generalization. University of Chicago Press*, p. 411-435.

Omidvar, Mohammad Nabi, Xiaodong Li, Yi Mei et Xin Yao (2014). « Cooperative Co-Evolution With Differential Grouping for Large Scale Optimization ». In : *IEEE Transactions on Evolutionary Computation* 18.3, p. 378-393. doi : 10.1109/TEVC.2013.2281543.

Oshlack, Alicia et Matthew J Wakefield (2009). « Transcript Length Bias in RNA-seq Data Confounds Systems Biology ». In : *Biology direct* 4, p. 1-10. doi : 10.1186/1745-6150-4-14.

Otterlo, Martijn van et Marco Wiering (2012). « Reinforcement Learning and Markov Decision Processes ». In : *Reinforcement Learning: State-of-the-Art*. Springer, p. 3-42. doi : 10.1007/978-3-642-27645-3_1.

Panait, Liviu, R Paul Wiegand et Sean Luke (2003). « Improving Coevolutionary Search for Optimal Multiagent Behaviors ». In : *IJCAI*, p. 653-660.

Paredis, Jan (1995). « Coevolutionary Computation ». In : *Artificial Life* 2.4, p. 355-375. doi : 10.1162/artl.1995.2.4.355.

Pelleau, Marie, Antoine Miné, Charlotte Truchet et Frédéric Benhamou (2013). « A Constraint Solver Based on Abstract Domains ». In : *International Workshop on Verification, Model Checking, and Abstract Interpretation*. Springer, p. 434-454. doi : 10.1007/978-3-642-35873-9_26.

Peterson, James L. (1977). « Petri Nets ». In : *ACM Comput. Surv.* 9.3, p. 223-252. doi : 10.1145/356698.356702.

Petrowski, A. (1996). « A Clearing Procedure as a Niching Method for Genetic Algorithms ». In : *Proceedings of IEEE International Conference on Evolutionary Computation*, p. 798-803. doi : 10.1109/ICEC.1996.542703.

Pincus, Diane J, William R Beam et Richard J Martin (1995). « Chronobiology and Chronotherapy of Asthma ». In : *Clinics in Chest Medicine* 16.4, p. 699-713. doi : 10.1016/S0272-5231(21)01172-2.

Popovici, Elena, Anthony Bucci, R. Paul Wiegand et Edwin D. De Jong (2012). *Coevolutionary Principles*. doi : 10.1007/978-3-540-92910-9_31.

Potter, Mitchell A. et Kenneth A. De Jong (1994). « A Cooperative Coevolutionary Approach to Function Optimization ». In : *Parallel Problem Solving from Nature*. Springer, p. 249-257. doi : 10.1007/3-540-58484-6_269.

Preuss, Mike (2015). *Multimodal Optimization by Means of Evolutionary Algorithms*. Springer.

Preuss, Mike, Michael Epitropakis, Xiaodong Li et Jonathan E. Fieldsend (2021). « Multimodal Optimization: Formulation, Heuristics, and a Decade of Advances ». In : *Metaheuristics for Finding Multiple Solutions*. Springer, p. 1-26. doi : 10.1007/978-3-030-79553-5_1.

Preuss, Mike, Michael G Epitropakis, Xiaodong Li et Jonathan E Fieldsend (2021). *Metaheuristics for Finding Multiple Solutions*. Springer.

Puterman, Martin L (2014). *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons.

Rechenberg, Ingo (1965). « Cybernetic Solution Path of an Experimental Problem ». In : *Roy. Aircr. Establ., Libr. transl.* 1122.

Reeves, Colin R et Christine C Wright (1995). « Epistasis in Genetic Algorithms: An Experimental Design Perspective. » In : *ICGA*, p. 217-224.

Rigden, Daniel J et Xosé M Fernández (2017). « The 2018 Nucleic Acids Research Database Issue and the Online Molecular Biology Database Collection ». In : *Nucleic Acids Research* 46.D1, p. D1-D7. doi : 10.1093/nar/gkx1235.

Rosin, Christopher D. et Richard K. Belew (1997). « New Methods for Competitive Coevolution ». In : *Evolutionary Computation* 5.1, p. 1-29. doi : 10.1162/evco.1997.5.1.1.

Russell, Stuart J et Peter Norvig (2016). *Artificial Intelligence: A Modern Approach*. Pearson.

Sanchez, Ernesto, Giovanni Squillero et Alberto Tonda (2011). « Group Evolution: Emerging Synergy through a Coordinated Effort ». In : *IEEE Congress of Evolutionary Computation (CEC)*, p. 2662-2668. doi : 10.1109/CEC.2011.5949951.

Sarma, Jayshree et Kenneth De Jong (1996). « An Analysis of the Effects of Neighborhood Size and Shape on Local Selection Algorithms ». In : *Parallel Problem Solving from Nature — PPSN IV*. Springer, p. 236-244. doi : 10.1007/3-540-61723-X_988.

Schwefel, Hans-Paul (1981). *Numerical Optimization of Computer Models*. John Wiley & Sons, Inc.

Selman, Bart et Carla P Gomes (2006). « Hill-Climbing Search ». In : *Encyclopedia of cognitive science* 81, p. 82.

Siebert, Heike et Alexander Bockmayr (2006). « Incorporating Time Delays into the Logical Analysis of Gene Regulatory Networks ». In : *Computational Methods in Systems Biology*. Springer, p. 169-183. doi : 10.1007/11885191_12.

Silver, David (2009). « Reinforcement Learning and Simulation-based Search in the Game of Go ». Thèse de doct. University of Alberta.

— (2015). *Lectures on Reinforcement Learning*. url: https://www.davidsilver.uk/teaching/.

Sironi, Chiara F. et Mark H. M. Winands (2016). « Comparison of Rapid Action Value Estimation Variants for General Game Playing ». In : *2016 IEEE Conference on Computational Intelligence and Games (CIG)*, p. 1-8. doi : 10.1109/CIG.2016.7860429.

Smolensky, Michael H (1996). « Chronobiology and Chronotherapeutics Applications to Cardiovascular Medicine ». In : *American Journal of Hypertension* 9.4, Supplement 2, 11S-21S. doi : 10.1016/0895-7061(95)00405-X.

Snoussi, El Houssine (1989). « Qualitative Dynamics of Piecewise-Linear Differential Equations: A Discrete Mapping Approach ». In : *Dynamics and Stability of Systems* 4.3-4, p. 565-583. doi : 10.1080/02681118908806072.

Sohrabi, Shirin, Anton Riabov, Michael Katz et Octavian Udrea (2018). « An AI Planning Solution to Scenario Generation for Enterprise Risk Management ». In : *Proceedings of the AAAI Conference on Artificial Intelligence* 32.1. doi : 10.1609/aaai.v32i1.11304.

Sörensen, Kenneth (2015). « Metaheuristics—The Metaphor Exposed ». In : *International Transactions in Operational Research* 22.1, p. 3-18. doi : 10.1111/itor.12001.

Speck, David, Robert Mattmüller et Bernhard Nebel (2020). « Symbolic Top-k Planning ». In : *Proceedings of the AAAI Conference on Artificial Intelligence* 34.06, p. 9967-9974. doi : 10.1609/aaai.v34i06.6552.

Srivastava, Biplav, Tuan Anh Nguyen, Alfonso Gerevini, Subbarao Kambhampati, Minh Binh Do et Ivan Serina (2007). « Domain Independent Approaches for Finding Diverse Plans. » In : *Proceedings of the 20th International Joint Conference on Artifical Intelligence*, p. 2016-2022.

Stanley, David A, Sachin S Talathi et Paul R Carney (2014). « Chronotherapy in the Treatment of Epilepsy ». In : *ChronoPhysiology and Therapy* 4, p. 109-123. doi : 10.2147/CPT.S54530.

Stanley, Kenneth O. et Risto Miikkulainen (2002). « Evolving Neural Networks through Augmenting Topologies ». In : *Evolutionary Computation* 10.2, p. 99-127. doi : 10.1162/106365602320169811.

Stoean, Catalin, Mike Preuss, Ruxandra Stoean et D. Dumitrescu (2010). « Multimodal Optimization by Means of a Topological Species Conservation Algorithm ». In : *IEEE Transactions on Evolutionary Computation* 14.6, p. 842-864. doi : 10.1109/TEVC.2010.2041668.

Storn, Rainer (1995). « Differrential Evolution-A Simple and Efficient Adaptive Scheme for Global Optimization over Continuous Spaces ». In : *Technical report, International Computer Science Institute* 11.

Sun, Honglu (2023). « Identifying and Analyzing Long-term Dynamical Behaviors of Gene Regulatory Networks with Hybrid Modeling ». Thèse de doct. École centrale de Nantes. url : https://theses.hal.science/tel-04472607.

Sun, Jianyong, Jonathan M. Garibaldi et Charlie Hodgman (2012). « Parameter Estimation Using Metaheuristics in Systems Biology: A Comprehensive Review ». In : *IEEE/ACM Transactions on Computational Biology and Bioinformatics* 9.1, p. 185-202. doi : 10.1109/TCBB.2011.63.

Sutton, Richard S et Andrew G Barto (2018). *Reinforcement Learning: An Introduction*. MIT press.

Świechowski, Maciej, Konrad Godlewski, Bartosz Sawicki et Jacek Mańdziuk (2023). « Monte Carlo Tree Search: A Review of Recent Modifications and Applications ». In : *Artificial Intelligence Review* 56.3, p. 2497-2562. doi : 10.1007/s10462-022-10228-y.

Teytaud, Fabien et Olivier Teytaud (2010). « Creating an Upper-Confidence-Tree Program for Havannah ». In : *Advances in Computer Games*. Springer, p. 65-74. doi : 10.1007/978-3-642-12993-3_7.

Thomas, R. et M. Kaufman (2001). « Multistationarity, the Basis of Cell Differentiation and Memory. II. Logical Analysis of Regulatory Networks in Terms of Feedback Circuits ». In : *Chaos: An Interdisciplinary Journal of Nonlinear Science* 11.1, p. 180-195. doi : 10.1063/1.1349893.

Thomas, René (1973). « Boolean Formalization of Genetic Control Circuits ». In : *Journal of Theoretical Biology* 42.3, p. 563-585. doi : 10.1016/0022-5193(73)90247-6.

Thomas, René et Richard d'Ari (1990). *Biological Feedback*. CRC press.

Tonda, Alberto, Evelyne Lutton et Giovanni Squillero (2012). « A Benchmark for Cooperative Coevolution ». In : *Memetic Computing* 4, p. 263-277. doi : 10.1007/s12293-012-0095-x.

Tuerk, Andreas, Gregor Wiktorin et Serhat Güler (2017). « Mixture Models Reveal Multiple Positional Bias Types in RNA-Seq Data and Lead to Accurate Transcript Concentration Estimates ». In : *PLOS Computational Biology* 13.5, p. 1-25. doi : 10.1371/journal.pcbi.1005515.

Ursem, R.K. (1999). « Multinational Evolutionary Algorithms ». In : *Proceedings of the 1999 Congress on Evolutionary Computation*. T. 3, p. 1633-1640. doi : 10.1109/CEC.1999.785470.

Voss, Stefan (2001). « Meta-heuristics: The State of the Art ». In : *Local Search for Planning and Scheduling*. Springer, p. 1-23. doi : 10.1007/3-540-45612-0_1.

Walden, Aidan et Maxim Buzdalov (2024). « A Simple Statistical Test Against Origin-Biased Metaheuristics ». In : *Applications of Evolutionary Computation*. Springer, p. 322-337. doi : 10.1007/978-3-031-56852-7_21.

Wang, Weijia et Michèle Sebag (2012). « Multi-objective Monte-Carlo Tree Search ». In : *Proceedings of the Asian Conference on Machine Learning*. Sous la dir. de Steven C. H. Hoi et Wray Buntine. T. 25. Proceedings of Machine Learning Research. PMLR, p. 507-522.

Wang, Yizao, Jean-yves Audibert et Rémi Munos (2008). « Algorithms for Infinitely Many-Armed Bandits ». In : *Advances in Neural Information Processing Systems*. T. 21. Curran Associates, Inc.

Ward, Allen, Jeffrey K Liker, John J Cristiano et Durward K Sobek II (1995). « The Second Toyota Paradox: How delaying decisions can make better cars faster ». In : *MIT Sloan Management Review*.

Wessing, Simon (2015). « Two-stage Methods for Multimodal Optimization ». Thèse de doct. Dortmund, Technische Universität.

Wolpert, D.H. et W.G. Macready (1997). « No Free Lunch Theorems for Optimization ». In : *IEEE Transactions on Evolutionary Computation* 1.1, p. 67-82. doi : 10.1109/4235.585893.

Wright, Alden H. (1991). « Genetic Algorithms for Real Parameter Optimization ». In : t. 1. Foundations of Genetic Algorithms. Elsevier, p. 205-218. doi : 10.1016/B978-0-08-050684-5.50016-1.

Wright, Sewall (1943). « Isolation by Distance ». In : *Genetics* 28.2, p. 114.

Yang, Zhenyu, Ke Tang et Xin Yao (2008). « Large Scale Evolutionary Optimization using Cooperative Coevolution ». In : *Information Sciences* 178.15, p. 2985-2999. doi : 10.1016/j.ins.2008.02.017.

Yin, Xiaodong et Noël. Germay (1993). « A Fast Genetic Algorithm with Sharing Scheme Using Cluster Analysis Methods in Multimodal Function Optimization ». In : *Artificial Neural Nets and Genetic Algorithms*. Springer, p. 450-457. doi : 10.1007/978-3-7091-7533-0_65.

Zhan, Zhi-Hui, Jun Zhang, Yun Li et Henry Shu-Hung Chung (2009). « Adaptive Particle Swarm Optimization ». In : *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 39.6, p. 1362-1381. doi : 10.1109/TSMCB.2009.2015956.

Zhang, Ray, Nicholas F Lahens, Heather I Ballance, Michael E Hughes et John B Hogenesch (2014). « A Circadian Gene Expression Atlas in Mammals: Implications for Biology and Medicine ». In : *Proceedings of the National Academy of Sciences* 111.45, p. 16219-16224. doi : 10.1073/pnas.1408886111.

Zimmerman, Terry et Subbarao Kambhampati (2002). « Generating Parallel Plans Satisfying Multiple Criteria in Anytime Fashion ». In : *Proceedings Workshop on Planning and Scheduling with Multiple Criteria*, p. 56-66.

Zitzler, Eckart et Lothar Thiele (1998). « Multiobjective Optimization using Evolutionary Algorithms — A Comparative Case Study ». In : *Parallel Problem Solving from Nature — PPSN V*. Springer, p. 292-301. doi : 10.1007/BFb0056872.