Couplage d'algorithmes d'optimisation par un système multi-agents pour l'exploration distribuée de simulateurs complexes : Application à l'épidémiologie

Rapport d'avancement à mi-parcours

HO The Nhan

24 juin 2014

Direction :	Jean-Daniel Zucker	UMMISCO, IRD, FRANCE
Co-Direction :	Nguyen Hong Quang	IFI-VNU, VIETNAM
Encadrements :	Nicolas Marilleau	UMMISCO, IRD, FRANCE
	et Laurent Philippe	DISC, FEMTO-ST, FRANCE

1 Introduction

Cette thèse s'inscrit dans les domaines de l'exploration de paramètres de simulations complexes dédié à l'aide à la décision environnementale.Pour cela, nous souhaitons profiter des ressources de calcul haute performance et des techniques de couplage d'algorithmes d'optimisation par une approche agent.
L'étude des systèmes complexes tels que des systèmes écologiques ou urbains, nécessite le plus souvent l'utilisation d'outils informatiques de simulation. Ces derniers reposent à la fois sur des approches de modélisation (ex. Systèmes dynamiques, systèmes multi-agents), et sur des données réels (ex. Système d'Information Géographique, Enquêtes de terrain). L'enjeu est de reproduire, de façon réaliste, des phénomènes complexes (ex. congestion routière, la raréfaction des sols ou la salinisation des estuaires). Les simulateurs deviennent alors des outils de compréhension et/ou de prospection [1], qui par la réalisation de scénarios virtuels mais réalistes donnent des résultats d'un réalisme croissant.
Le crédit donné aux résultats d'une simulation dépend fortement de la confiance qui est accordée au simulateur, et donc de la qualité de sa validation. Cette confiance ne s'obtient qu'au travers d'une étude poussée du modèle, d'une analyse de sensibilité aux paramètres et d'une confrontation des résultats de simulation aux des données de terrain. Lorsqu'une résolution mathématique n'est pas possible, il est alors nécessaire d'explorer l'espace des paramètres du simulateur afin d'établir le lien entre les paramètres d'entrée du simulateur et les résultats de simulation. Pour cela, pléthore de simulations (devant être répliquées) est nécessaire, ce qui coûteux du point de vue des ressources mobilisés (temps de calcul, processeurs et mémoire). Ceci est particulièrement vrai pour les simulateurs à base d'agents, qui nécessitent des temps de simulation parfois longs (minutes, plusieurs heures voire journées) et présentent une forte sensibilité aux paramètres [2].
Dans ce contexte où les modèles et les simulateurs présentent une complexité croissante, l'utilisation de moyens de calcul haut performance (HPC) s'avère être une nécessité. Afin de facilité leur accès, des plate-formes spécialisées comme Epis [3], Kepler [4], OpenMole [5] ou SimExplorer [6] ont été développés pour la communauté des systèmes complexes. Ces dernières affichent des fonctionnalités intéressantes, rendant accessible et transparent la parallélisation et le déploiement de plans d'expériences sur clusters, grilles ou serveurs isolés de calcul. Ainsi, la soumission d'une multitude d'expérimentations sur un cluster est possible par des chercheurs venant d'autres horizons [7]. Malgré ces avancées, une exploration complète de l'espace des paramètres d'un simulateur s'avère généralement impossible compte tenu du nombre de simulations à réaliser et de la durée de chacune d'elle : la réduction de l'espace des paramètres est un passage obligé.
L'une des particularités des *simulateurs complexes* (simulateurs représentatifs d'un système complexe) est d'avoir un espace des paramètres dont la nature et la forme est fonction : (i) des objectifs scientifiques ; (ii) de la nature des paramètres manipulés ; et (iii) surtout du systèmes complexes étudiés. Ainsi, le choix d'une stratégie d'exploration, qui peut être classifiée en la stratégie d'exploration systématiques

de l'espace de recherche (les méthodes exactes [8], les approaches de discrétisation [9], les techniques d'échantillonnages [10]) et la stratégie algorithmes d'exploration stochastiques et heuristiques de l'espace de recherche (approche de Monté-Carlo [11], les approches par voisinage, les approches constructives [12, 13]), s'effectue au cas par cas selon les attentes du scientifique et les données de terrain en présence. C'est pourquoi quelques approches interactives d'exploration ont vu le jour. Elles reposent sur l'expérience du scientifique et sur sa participation tout au long du processus d'exploration d'un simulateur. D'autres approches tentent de combiner, au cas par cas, des algorithmes traditionnels d'exploration pour constituer un nouvel algorithme adapté à la problématique abordée par le simulateur. Cependant cette association d'algorithme entraine un important travail de mise au point.
Compte tenu de la singularité des simulateurs complexes, des difficultés rencontrées lors de l'exploration de leur espace de paramètres, nous envisageons de guider cette tâche d'exploration en proposant un protocole distribué, à base d'agent, appelé GRADEA. Il s'agit d'une approche hybride reposant sur trois stratégies de recherche de différents classements dans un même environnement : la recherche en criblage pour zones d'intérêt, la recherche globale et la recherche locale. Ainsi différentes stratégies de recherche vont en parallèle parcourir l'espace de recherche pour trouver l'optimum globale d'un problème et également pour établir partiellement la cartographie de l'espace des solutions pour comprendre le comportement émergent du modèle.
Une première version du protocole d'exploration est appliquée à un cas d'application concrêt : l'aide à la conception de la planification des politiques de vaccination de la rougeole au Vietnam. En considérant ce cas d'étude et le simulateur complexe développé pour l'occasion, nous avons présélectionné des algorithmes d'exploration que nous avons couplés grâce au protocole GRADEA.
Le protocole GRADEA est associé à une infrastructure logicielle modulaire, à base d'agents. Cette dernière interagit avec des ressource de calcul haute performance afin d'exécutés les simulations nécessaire à l'exploration d'un modèle. L'enjeu résolu jusqu'à ce temps est de proposer, à la communauté, un environnement optimisé où l'utilisateur sera en mesure : (i) de combiner des algorithmes d'exploration adaptés à son cas d'étude ; (ii) et de tirer partie des ressources disponibles de calcul haute performance pour réaliser l'exploration.

2 Approches usités pour l'exploration de Simulateurs Complexes

L'exploration d'un simulateur consiste à exécuter une multitude de simulations avec différents jeux de paramètres en vue de trouver le jeu de paramètres donnant des résultats de simulation qui répondent, au mieux, à la problématique de terrain. L'une des pratiques des plus usités consiste à réaliser une exploration exhaustive, c'est-à-dire combiner l'ensemble des valeurs possibles des paramètres, et réaliser les simulations associées. Afin d'obtenir une confiance suffisante dans les résultats, chaque jeu de paramètres doit être testé plusieurs fois (environ 30 fois), surtout lorsque la stochasticité du simulateur est importante.
Compte tenue du nombre de paramètres, du domaine à parcourir (parfois infini lorsqu'il s'agit d'une combinaison de paramètres réels) et des temps de simulation, l'exploration exhaustive des simulateurs complexes est souvent impossible. Ceci est particulièrement vrai lorsque nous nous intéressons à des simulateurs comme MobiSim [14], SimPop [15], MatSim [16] : simulateurs particulièrement descriptifs de la réalité qui affiche une importante sensibilité aux paramètres.
Des méthodes d'exploration plus évoluées ont alors vu le jour comme : [12, 13, 17] , (nous en faisons une taxonomie par la suite). De nombreux travaux tentent alors de les associer afin d'en tirer le meilleur. Ce point est abordé dans un second temps.

2.1 Taxonomie des algorithmes d'exploration usités

La réalisation de simulations reproduisant un phénomène réel vise à répondre à une problématique scientifique de terrain, par exemple en écologie, en urbanisme et en épidémiologie. L'exploration d'un simulateur complexe est un problème d'optimisation : nous tentons de trouver un jeu de paramètres qui minimise (ou maximise) une fonction d'utilité, fonction nécessaire pour évaluer la réponse du modèle vis à vis de la question scientifique.
Nombreux sont les algorithmes d'exploration. Ils s'organisent dans 3 catégories [12, 13] :

— *Recherche locale (ex. HillClimbing, Recuit Simulé, Tabu Search, Pattern Search)* : A partir d'un jeu de paramètre de départ, cette stratégie vise à parcourir de proche en proche l'espace des paramètres tout en maximisant (ou minimisant) une fonction d'utilité. il s'agit donc d'approches dirigées par

le voisinage ou la granulation discrétisée si bien que l'exploration peut s'enfermer et s'achever dans un optimum local. Dans un contexte favorable, ces approches sont rapides et efficaces.

— *Recherche globale (ex. Algorithme génétique, Stratégie d'évolution, Optimisation par essaims particulaires)* : Cette stratégie vise à couvrir la totalité de l'espace de recherche et déterminer par raffinement successif une solution optimale. Ces approches sont particulièrement performantes lorsque l'espace de recherche est multimodale. Une stratégie de recherche globale peut être réalisée de divers manières : par un algorithme de recherche globale par lui-même ; par de multiples réplications d'un algorithme de recherche locale à partir de différents points initiaux ; une hybridisation des algorithmes de recherche (tous deux types globale et local).

— *Recherche par criblage (ex. Latin Hypercube Samling, DIviding RECTangle, Recherche Dichotomique)* : la recherche par criblage est une stratégie de recherche systématique visant à découper l'espace des paramètres en vue de détecter des zones prometteuses. Une solution simple de criblage vise à échantillonner l'espace des paramètres à intervalles réguliers. D'autres stratégies s'appuient sur des heuristiques pour diviser et fusionner afin de focaliser la recherche dans des zones optimales. Ce type d'approche est efficace et donne avec peu d'itération un résultat mais qui reste imprécis.

En résumé, les 3 stratégies précédemment citées sont complémentaires. Les approches de recherche locales sont précises et efficaces mais tombent dans le piège de l'optimum local. Inversement, les approches de recherche global s'affranchissent des optimums locaux mais restent imprécises. Les approches de criblages sont quant-à elles adaptées pour réaliser une cartographie de l'espace des paramètres. Cette complémentarité des approches nous laisse imaginer l'intérêt de les associer.

2.2 Combiner pour mieux explorer

Combiner différentes stratégies d'exploration permet tirer partie de chacune d'elles voire d'iniber leurs lacunes. De nombreux travaux proposent d'associer, par exemple, les algorithmes évolutionnaires avec d'autres approches comme DIRECT [10]. Leur association offre une cartographie de l'espace des paramètres [18, 12] et de mieux l'explorer ensuite.

Par exemple, l'algorithme Mémétique combine un Algorithme Evolutionaire (AE) avec un algorithme de recherche local. Cet algorithme s'exécute en deux temps : (i) sélection d'un jeu de paramètres à l'aide d'un AE ; (ii) utilisation d'un algorithme de recherche locale pour rafiner les solutions trouvées. L'interaction entre l'AE et l'algorithme de recherche locale est permise : (i) par une mémoire partagée [19] ; (ii) ou par des interactions entre agents [20].

Dans les travaux de *Hiwa et al.* [10], l'algorithme de criblage (DIRECT) a été associé à la fois avec un algorithme de recherche globale (Algorithme génétique) et un algorithme de recherche locale **(SQP)** [21]. Cette solution offre une meilleure couverture de l'espace des paramètres lorsque ce dernier n'est pas homogène.

Le travail de *Griffin et al.* [22] tente de capitaliser des algorithmes pour les combiner sous la formes de composants logiciels. Plusieurs composants de recherche sont alors disponibles (LHS, DIRECT, APPS) et rendent la plate-forme flexible, distribuable sur plusieurs noeuds de calcul. Cependant la stratégie d'exploration reste fixé.

3 Protocole d'exploration stratifiée coopérative GRADEA : une approche orientée agent

Comme nous l'avons vu précédemment, une exploration efficace d'un modèle complexe doit tenir compte des objectifs scientifiques et du système étudié. Dans ce cadre, l'utilisation de stratégies hybrides est nécessaire pour explorer des espaces de paramètres singuliers dans leur nature et leur forme. L'utilisation de stratégies hybrides d'exploration, établies au cas par cas en tenant compte des ressources de calcul, s'avère nécessaire. Mais la mise en place de tels algorithmes est coûteuse en développement et en mise au point.

Ainsi, nous proposons une architecture modulaire et distribuée qui assure le couplage de plusieurs algorithmes d'exploration, selon les besoins. L'enjeu est de permettre aux utilisateurs de définir simplement des stratégies d'exploration qui s'appuient sur des algorithmes existants. Par la combinaison et l'ordonnancement de ces algorithmes, nous souhaitons faciliter la définition de stratégies personnalisées et adaptées à l'exploration de simulateurs complexes.

3.1 Exploration stratifiée coopérative

L'approche GRADEA s'appuie sur une architecture originale qui repose sur le paradigme agent [23]. Les agents sont des entités autonomes qui agissent et interagissent entre elles. Ils accomplissent ainsi un objectif individuel selon un comportement prédéfini. Leur association au sein d'un *Système Multi-Agents (SMA)* favorise la réalisation d'un objectif global, qui, dans notre contexte, vise à explorer un simulateur complexe, de manière intelligente et optimisée.

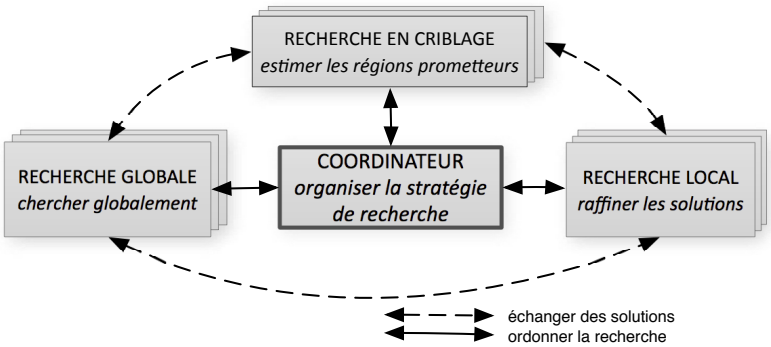

FIGURE 1 – La stratégie de recherche multi-étage

Deux types d'agents sont considérés dans l'approche GRADEA (voir dans la figure 1) :

- *les agents de recherche* réalisent l'exploration d'un espace (ou sous espace) de paramètres selon un algorithme de recherche défini dans son comportement (ex. Monte-Carlo, évolutionnaire, etc) ;
- *l'agent coordinateur* ordonnance, quant à lui, les différentes explorations en interagissant avec les *agents de recherche* pour ordonner une exploration et/ou récupérer et analyser des résultats (même partielle). La stratégie d'ordonnancement est définie par l'utilisateur au sein du comportement de l'agent coordinateur.

Compte tenu de la taxonomie des algorithmes d'exploration présent dans la littérature (cf section 2.1), nous avons identifiés 3 catégories d'*agents de recherche* : (i) les Agents de Recherche Globale (ARG) ; (ii) les Agents de Recherche par Criblage (ARC) ; (iii) et les Agents de Recherche Locale (ARL). Ces agents ont la particularité d'implémenter un algorithme d'exploration de sa catégorie.

3.2 Fonctionnement général

Le système multi-agents issue de GRADEA se compose d'**1** instance d'*agent de coordination* et de **n** instances d'*agents de recherche*. Le déroulement de l'exploration est fortement conditionné par la stratégie développée dans le comportement de l'agent de coordination. Les algorithmes de recherche sont disponibles sur une base de donnée (figure 3).
En générale, le processus de control de l'agent de coordination est comme dans la figure (figure 2) :

a. L'agent de coordination (AC) inscrit et initialise les algorithmes ordonnés sur GRADEA à base de la stratégie d'adaptation (SAd) et la stratégie d'analyse (SAn) prédéfinie du modélisateur à partir de la base d'algorithmes.

b. Au commencement, l'agent de coordination lance au moins un instance d'un algorithme de recherche selon le stratégie.

c. Pendant la recherche, l'agent de recherche (AR) partager les régions d'intérêt ou les solutions prometteuses ensemble et avec l'agent de coordinateur.

d. L'agent de l'algorithme se termine selon ses conditions de termination s'il est activé...

A titre d'exemple, un déroulement très simple d'une exploration d'une simulation complexe serait :

a. L'agent de coordination ordonne à une instance d'ARC de cribler l'espace des paramètres.

b. L'agent de coordination ordonne le transfert des 20 meilleurs résultats de simulation et des jeux de paramètres associés à une instance d'ARG.

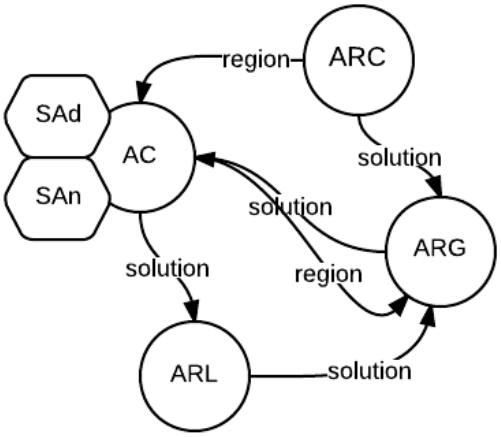

FIGURE 2 – Une stratégie de coordination

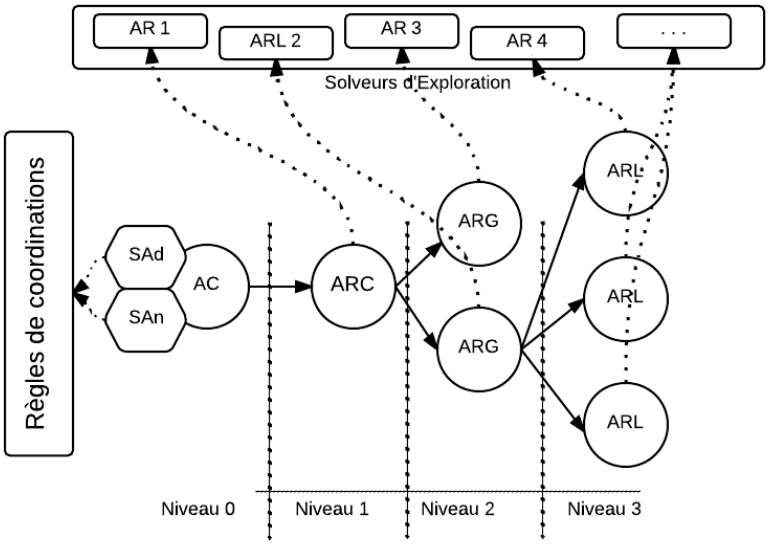

FIGURE 3 – Connexité du réseau d'exploration

c. L'ARG choisi, implémentant un algorithme évolutionnaire, explore le simulateur pendant un certain nombre de générations et prévient l'agent de coordination de la découverte d'une solution intéressante. L'ARG continue ensuite à itérer.

d. Pendant ce temps là, l'agent de coordination choisit une instance d'ARL et lui transmet le jeu de paramètres correspondant à la solution précédemment découverte par l'ARG.

e. et ainsi de suite jusqu'à l'obtention d'une solution satisfaisante...

L'exemple présenté ci-dessus est un cas d'école. Le processus d'exploration, autrement dit la stratégie de coordination, est fonction des objectifs scientifiques et doit être définie par le modélisateur. Ceci est rendu possible dans l'approche GRADEA par la description de l'agent de coordination, ce qui est un atout majeur.
Ainsi, l'utilisation d'un système multi-agents comme moteur d'exploration présente de nombreux avantages :

— *Modularité* : Cette caractéristique est hérité du paradigme agent. Chaque instance d'algorithme de recherche étant indépendante et autonome, il est possible d'en réduire le nombre ou d'en ajouter de nouvelles sans impacter le système et être obligé de modifier son architecture et son

fonctionnement.

— *Versatilité* : Les algorithmes pouvant être choisis à la demande, la stratégie de coordination pouvant être déterminé par l'utilisateur, GRADEA offre une importante versatilité si bien qu'elle peut s'appliquer dans de nombreux domaines.
— *Parallelisme* : L'ensemble de la plate-forme a été conçue afin de profiter des ressources de calcul intensif. Pour cela nous avons conçu une architecture distribuée qui repose sur un systèmes multi-agents, un serveurs d'application et un cluster. Ce point est abordée plus en détail dans la partie suivante.

4 Architecture logicielle distribuée

4.1 Architecture logicielle

Afin de réduire les temps d'exploration, l'infrastructure logicielle de GRADEA s'appuie sur les techniques et les outils de calculs intensifs, à savoir (cf figure 4) : (i) un serveur de calcul pour réaliser les nombreuses simulations complexes ; (ii) un serveur d'applications JEE pour l'exécution du système multi-agents et le contrôle des soumissions. Cette infrastructure se décompose en 3 couches :

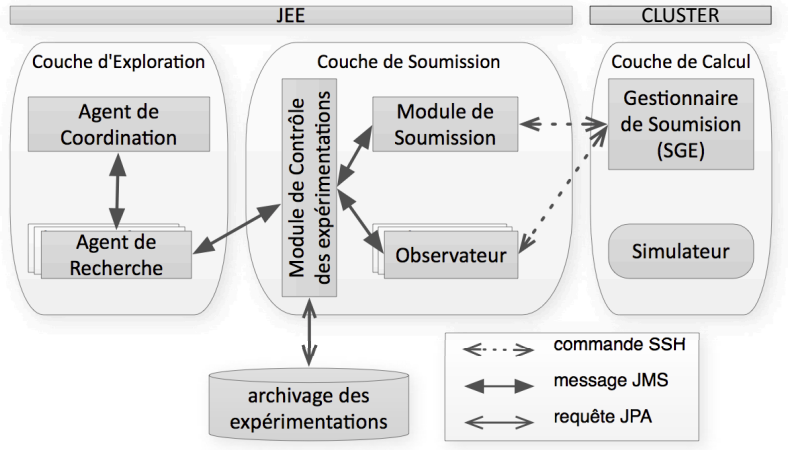

FIGURE 4 – La connection entre les agents de recherche et le cluster

— *Couche de Calcul :* Cette couche vise à tirer partie des ressources d'un cluster pour réaliser les centaines, les milliers voire les dizaines de milliers de simulations nécessaires pour explorer un simulateur complexe. Les simulations sont pilotés par le serveur d'application et son Système Multi-Agents, au travers des gestionnaires de soumission (couche de Soumission et Gestionnaire de Soumission du cluster).
— *Couche d'Exploration :* cette couche détermine les jeux de paramètres de simulations qui sont à tester. Pour cela, elle exécute le Système Multi-Agents dont l'architecture est présentée dans la section 3. Le SMA est en étroite interaction avec le *module de contrôle des expérimentations* à la fois pour ordonner l'exécution de simulations et pour recevoir les résultats de simulations antérieures.
— *Couche de Soumission :* Cette couche est une interface de haut niveau qui permet : (i) de soumettre des simulations à la *couche de calcul* (module de soumission), (ii) d'écouter et recevoir les résultats des simulations en cours (Observateur) ; (iii) d'archiver les résultats et les rendre disponibles pour la couche d'exploration.

4.2 Principe de fonctionnement

Les différentes couches communiquent au moyen de message JMS au sein du serveur d'application et via SSH entre le serveur d'application et le cluster. Ainsi nous proposons un fonctionnement décentralisé des modules avec un exécution distribuée et asynchrone.

a. Les Agents de Recherche du système multi-agents s'exécutant dans la couche d'exploration génèrent des jeux de paramètres qui sont envoyés par JMS au module de Contrôle des expérimentations. Ils traitent les résultats de simulations lorsqu'ils sont réceptionnés.

b. Le module de controle des expérimentations recherche dans la base d'archivage si ces jeux de paramètres ont été testés antérieurement. Si ce n'est pas le cas, le module de contrôle soumet les jeux de paramètres au cluster via le module de Soumission. Ce dernier génère les scripts (script SGE) et les communique en SSH au gestionnaire de soumission du cluster.

c. Pour chaque soumission, un observateur est créé. Son rôle est d'avertir, par message JMS, le module de contrôle d'un changement d'état d'une simulation.

d. Ainsi, lorsqu'une simulation s'achève, le module de contrôle est en mesure d'avertir la couche d'exploration et son système multi-agents.

Cette infrastructure logiciel a été déployée et éprouvée sur un serveur d'application JONAS [1] associé au cluster du mésocentre de calcul de l'Université de Franche-Comté.

5 Application

Afin d'illustrer ce travail, nous avons choisi de nous intéresser à une problématique concrète et d'actualité : la propagation d'une épidémie en Asie du Sud Est en vue d'identifier des stratégies de vaccination de la rougeole. Dans ce cadre, un modèle stochastique a été développé à partir du modèle SEIR [24]. Il permet d'évaluer des stratégies de vaccination à réaliser en tenant compte des coûts. Ce modèle introduit une politique de vaccination par impulsion, c'est à dire par des campagnes ciblées (spatialement et temporellement) de vaccination. Ces campagnes sont déclenchées par le nombre d'individus infecté (I) et le pourcentage vacciné de la population. Quand ce nombre atteint un seuil, une campagne sanitaire est menée visant à vacciner une partie des susceptibles (S). La politique est défini selon trois paramètres (w_1, w_2, w_3) : si $w_1 x(\mathrm{I}) + w_2 x(\mathrm{E}) > 100.0 * w_3$, faire vacciner 80% de (S).
Compte tenu du nombre de stratégies spatiales de vaccination possibles, une exploration complète du modèle est impossible d'où la nécessité d'utiliser des approches comme GRADEA. L'enjeu est de minimiser l'espace des paramètres à explorer, noté (w_1, w_2, w_3), tout en conservant des résultats satisfaisants.
Nous suivrons donc pas à pas l'approche GRADEA en présentant les algorithmes d'exploration mis en place, et la stratégie d'exploration dans un second temps. Dans un dernier temps, nous ferons une analyse des résultats.

5.1 Choix des algorithmes d'exploration

Comme nous l'avons vu dans la section 3, GRADEA est une approche modulaire qui permet de construire des algorithmes hybrides d'exploration, en fonction des besoins et de la question scientifique. Afin de répondre au mieux à notre problématique de modélisation, nos avons choisi d'associer 3 algorithmes d'exploration : (i) DIRECT (algorithme de recherche par criblage) ; (ii) l'algorithme évolutionnaire (algorithme de recherche globale) ; (iii) et APPS (un algorithme de recherche locale). A l'image des préconisations de GRADEA, un agent a été créé pour chaque algorithme couplé :

— *Agent GrDIRECT pour la recherche par criblage :* DIRECT (DIviding RECTangle) [25] est un algorithme déterministe de recherche par motifs pour l'optimisation de fonctions d'utilité multivariées. Il considère l'espace des paramètres comme des hyper-rectangle (boxs) multidimensionnelles et divisibles. DIRECT est caractérisé par 2 opérateurs : *la division* pour diviser des boxs potentiellement optimales en trois sous boxs ; *la sélection* pour choisir les boxs potentiellement optimales devant être explorées à l'itération suivante. Chaque *box* est représentée par son centre qui est évalué pour déterminé le prochain découpage. L'agent de Recherche par Criblage dont le comportement est animé par une variante de l'algorithme DIRECT proposé dans [26]. Nous avons développé une variante de l'opérateur de division qui s'applique à un ensemble de boxs dès que ils sont évaluées au lieu d'attendre tous les boxs sélectionnées. Cela réduit les temps d'attente des évaluations mais peut provoquer un changement d'ordre du partitionnement.

— *Agent GrEA pour la recherche globale :* Pour la recherche globale, nous faisons le choix d'utiliser un algorithme évolutionnaire (AE) présenté dans [27]. Cette algorithme a été adapté à notre

1. url :http ://jonas.ow2.org

contexte et implémenté dans le comportement de l'Agent de Recherche Globale. Le comportement de GrEA maintient μ jeux de paramètres dans sa population. AE est équipé des opérateurs *Selection, CrossOver, Mutation* commun à tous les algorithmes de la même famille. Par contre, notre implémentation présente l'avantage de s'executer de façon asynchrone. Il n'a pas besoin d'attendre l'évaluation de tous les candidats de la population pour passer à l'itération suivante.

— *Agent GrPS pour la recherche Locale :* L'algorithme intégrée pour l'Agent de Recherche Local est GrPS qui est une version de l'algorithme APPS (« Asynchronous Parallel Pattern Search » en anglais) dans [28]. GrPS explore l'espace de paramètres à partir d'un point unique et cherche l'optimum local autour du point.

5.2 Choix de l'algorithme coordination

Les trois agents définis précédemment (Agent GrPS, Agent GrEA et Agent GrDIRECT) sont cadencés selon l'approche GRADEA par un agent de coordination. Ce dernier décompose l'exploration du modèle d'épidémiologie en 3 étapes :

a. L'agent de coordination faire l'inscription des agents GrDIRECT, GrEA, GrPS pour s'executer en même temps.
b. L'agent GrDIRECT s'arrête lorsque les *box* ont une taille inférieure à un seuil.
c. L'agent GrEA s'achève lorsqu'aucune modification de la population de jeux de paramètre n'évolues pas pendant quelques itérations.
d. L'agent GrPS s'exécute tant que la fonction d'utilité n'attend pas un seuil avec une précision établie.
e. Agent GrPS et Agent GrDIRECT partage son meilleur jeu de paramètres trouvé à l'agent GrEA.

Dans ce contexte, l'agent Coordinateur est seulement un agent de transmission. Mais dans l'avenir, nous envisageons de lui conférer des capacités cognitives d'analyse de l'espace de paramètres afin de mieux orienter l'exploration par une utilisation parallèle des différentes méthodes de recherche.

5.3 Conditions initiales

Nous recherchons à réduire le nombre d'infection sur une population de 1000000 durant une période de 5 ans avec 600000 doses disponibles. L'évaluation de la performance de GRADEA sera réalisée au travers de l'exploration des paramètres (w_1, w_2, w_3) dont le domaine est compris entre 0 et 1 avec une précision de 0,01.
Le système GRADEA est intégré dans la plate-forme EPIS [3] afin d'utiliser son système de gestion des simulateurs. Ainsi, GRADEA sera en mesure de lancer les simulations sur un cluster basé sur SGE, système de soumission supporté par la version courante de l'EPIS. chaque simulation est ainsi soumise dans une file d'attente spécifique du cluster et répliquée 3000 fois.
Les simulations soumissent dans une file d'attente pour laquelle sera affectée 8, 16 et 32 coeurs, cadencés à 2.4 Ghz, et doté de 64go de mémoire vive. Nous testons ces différentes configurations afin d'évaluer les performances et la monté en charge du système GRADEA en fonction de sa solicitation.
Nous avons testé différentes configuration d'utilisation de GRADEA. Les compositions en mode parallèle que nous avons implémenté sur GRADEA se composent : GrEA (E), GrEA + GrDIRECT (ED), GrEA + GrDIRECT + GrPS (EDP)

5.4 Résultats numériques

L'approche GRADEA est évalué par deux métriques : le résultat thématique qui montre le capacité de convergence et son performance qui montre son scalabilité.
La figure 5 affiche la convergence de E, ED et EDP réalisé sur 8, 16 et 32 coeurs de calcul. La forme de la convergence a montré que EDP consomme moins nombre des évaluations que ED même que ED utilise moins que E. Autre part, la convergence des hybrides est plus velouté. Ce dernière est utile en cas d'explorer le comportement du modèle plutôt que trouver une solution optimale.
.
Les résultats présentés par la figure 6 montre l'efficacité de GRADEA et l'intérêt de combiner des algorithmes d'exploration. En effet, l'algorithme EDP se termine avec moins d'évaluations que ED tout en conservant une précision dans ses résultats. Pour cela, il tire le meilleur des trois algorithmes de

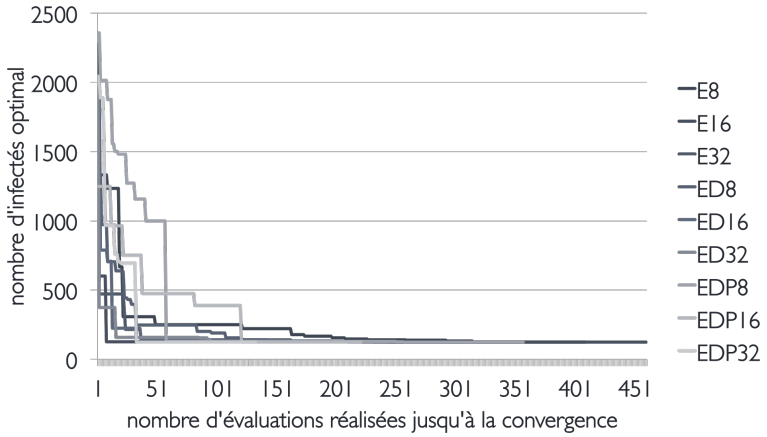

FIGURE 5 – convergence des différents stratégies réalisées sur 8, 16 et 32 coeurs de calcul

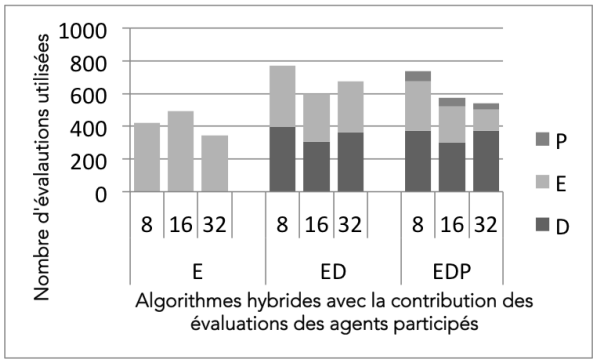

FIGURE 6 – nombre d'évaluations (nb_{eval}) (T(ms)/nb_{eval})

recherche. E semble plus performant que ED et EDP. Mais ces derniers profitent des capacités de calcul offertes par le cluster et favorisent une montée en charge, ce qui réduit les temps de simulation.

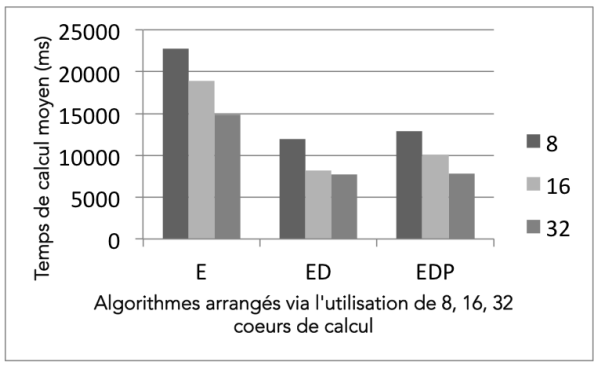

FIGURE 7 – temps de calcul moyen (T(ms)/nb_{eval})

Les résultats présentés par la figure 7 comparent les temps nécessaires pour l'exploration du modèle en fonction des différentes stratégies. Les algorithmes hybrides comme EDP ou ED sont plus rapides que E malgré un nombre d'itérations plus grand.
Les figures 6 et 7 montrent que l'algorithme DIRECT donne une solution avec moins d'itérations que les deux autres algorithmes. Par contre, le temps de calcul nécessaire pour atteindre la solution est plus important. Ceci vérifie que les approches hybrides sont hautement parallélisables et assurent une meilleure montée en change.
Malgré un temps moyen d'exécution plus lent que ED, l'algorithme hybride EDP donne en contrepartie des résultats plus précis. Ces temps de calcul et cette précision est le fruit de l'algorithme de recherche local (APPS), dernière étape de la stratégie implémentée dans EDP. Cet algorithme raffine la solution

mais ne soumet pas suffisamment d'évaluations dans la file d'attente. En effet il génère au plus de 2n (n=3) évaluations à chaque itération. Une des améliorations possibles viserait à distribuer plusieurs exécutions d'APPS, en le considérant, dans la stratégie d'exploration, comme une partie de l'algorithme évolutionnaire.

6 Conclusion

Au travers d'une analyse des algorithmes de la littérature nous remarquons que les stratégies d'exploration de simulateur s'organisent en trois courants majeurs : (i) la recherche par criblage ; (ii) la recherche locale ; (iii) et la recherche globale. Chacune de ces méthodes ont leurs avantages et inconvénients. En ce sens, elles peuvent devenir complémentaires lorsqu'elles sont combinées.
L'exploration d'un simulateur complexe nécessite la mise en place d'un algorithme d'exploration adapté et singulier. En effet, la nature et la texture de l'espace des paramètres sont fortement conditionnées par le système complexe étudier et la question scientifique. C'est pourquoi nous proposons une approche originale, GRADEA, qui permet de coupler facilement et rapidement des algorithmes d'exploration en fonction du cas d'étude.
GRADEA repose sur une architecture multi-agents où chaque agent est un algorithme d'exploration autonome qui collabore avec d'autres agents algorithme, sous le couvert d'un coordinateur. Grâce à cette approche, l'algorithme d'exploration devient interchangeable et combinable, en fonction des cas d'étude et sans modification de l'architecture existante.
Couplée à un serveur d'application JEE et un cluster, GRADEA permet de déployer les nombreuses simulations sur des moyens de calcul intensif. La coordination de l'exploration (choix des simulations à réaliser) est exécuter au sein du serveur d'application. Cette infrastructure logicielle a été déployée sur un serveur JEE JONAS jumelé au mésocentre de calcul de l'université de Franche-Comté (France).
Nous tirons partie de notre expérience en modélisation de phénomènes de propagation de maladie rougeole au Vietnam. Les tests ont montré les performances de GRADEA et sa scalabilité.
En amont de GRADEA, nous envisageons de proposer des mécanismes de clustering qui permettront d'identifier la nature de l'espace des paramètres d'un modèle complexe. Cet outil facilitera le choix des algorithmes d'exploration à utiliser pour étudier un phénomène réel.

Remerciements

Ce travail est réalisé avec le Département d'Informatique des Systèmes Complexes de l'institut Femto-ST et le Mésocentre de calcul de l'Université de Franche-Comté.

Publications

[29] The-Nhan HO, Nicolas MARILLEAU, Laurent PHILIPPE, Hong-Quang NGUYEN et Jean-Daniel ZUCKER. "A grid-based multistage algorithm for parameter simulation-optimization of complex system". In : *Computing and Communication Technologies, Research, Innovation, and Vision for the Future (RIVF), 2013 IEEE RIVF International Conference on*. 2013, p. 221–226.

[30] The-Nhan HO, Hong-Quang NGUYEN, Nicolas MARILLEAU et Jean-Daniel ZUCKER. *Couplage d'algorithmes d'optimisation par un système multiagent pour l'exploration distribuée de simulateurs complexes : Application à l'épidémiologie*. Conférence d'informatique en Parallélisme, Architecture et Système. Neuchâtel, Suisse. Avr. 2014.

Contributions

[3] Eric BLANCHART, Christophe CAMBIER, C. CANAPE, Benoit GAUDOU, The-Nhan HO, Tuong-Vinh HO, Christophe LANG, Fabien MICHEL, Nicolas MARILLEAU et Laurent PHILIPPE. "EPIS : A Grid Platform to Ease and Optimize Multi-agent Simulators Running". In : *Advances on Practical Applications of Agents and Multiagent Systems (PAAMS)*. T. 88. Springer, 2011, p. 129–134.

Références

[1] S. F. RAILSBACK et V. GRIMM. *Agent-Based and Individual-Based Modeling : A Practical Introduction*. Princeton Univ Press, 2011.

[2] Eric DAUDÉ et Patrice LANGLOIS. "Modélisation et simulation multi-agents application pour les Sciences de l'Homme et de la Société". In : Paris : Hermès, 2006. Chap. Comparaison de trois implémentations du modèle de Schelling, p. 411–441.

[4] David ABRAMSON, Blair BETHWAITE, Colin ENTICOTT, Slavisa GARIC et Tom PEACHEY. "Parameter Space Exploration Using Scientific Workflows". In : *Proceedings of the 9th International Conference on Computational Science (ICCS '09)*. Springer, 2009, p. 104–113.

[5] Romain REUILLON, Florent CHUFFART, Mathieu LECLAIRE, Thierry FAURE, Nicolas DUMOULIN et David R. C. HILL. "Declarative task delegation in OpenMOLE". In : *Proceedings of the 2010 International Conference on High Performance Computing & Simulation, (HPCS)*. T. 3. IEEE, 2010, p. 55–62.

[6] Florent CHUFFART, Nicolas DUMOULIN, Thierry FAURE et Guillaume DEFFUANT. "Simexplorer : Programming experimental designs on models and managing quality of modelling process". In : *International Journal of Agricultural and Environmental Information Systems (IJAEIS)* 1.1 (2010), p. 55–68.

[7] Clara SCHMITT et Sébastien REY COYREHOURCQ. "Exploring geographical agent based models : a new protocol using automated procedures". In : *European Colloquium on Quantitative and Theoritical Geography*. Athène, Grèce, 2011.

[8] Jose M. TROYA et M ORTEGA. "A study of parallel branch-and-bound algorithms with best-bound-first search". In : *Parallel Computing* 11.1 (1989), p. 121–126.

[9] Patrick TAILLANDIER et Frédéric AMBLARD. "Cartography of Multi-Agent Model Parameter Space through a reactive Dicotomous Approach". In : *European Simulation and Modelling Conference*. Guimaraes, Portugal, 2011, p. 38–42.

[10] S. HIWA, T. HIROYASU et M. MIKI. "Hybrid optimization using DIRECT, GA, and SQP for global exploration". In : *Evolutionary Computation, 2007. CEC 2007. IEEE Congress on*. 2007, p. 1709–1716.

[11] Michael MITZENMACHER et Eli UPFAL. *Probability and computing : Randomized algorithms and probabilistic analysis*. Cambridge University Press, 2005.

[12] Benoît CALVEZ et Guillaume HUTZLER. "Automatic tuning of agent-based models using genetic algorithms". In : *Proceedings of the 6th international conference on Multi-Agent-Based Simulation*. MABS'05. Springer, 2006, p. 41–57.

[13] Forrest J STONEDAHL et Uriel J ADVISER-WILENSKY. "Genetic algorithms for the exploration of parameter spaces in agent-based models". In : (2011).

[14] Jean-Philippe ANTONI, Gilles VUIDEL et Pierre FRANKHAUSER. "The MobiSim project : a multi-scalar agent-based Land-Use and Transport Integrated model". Anglais. In : *16 th European Colloquium of Quantitative and Theoretical Geography*. Maynooth, Irlande, sept. 2009.

[15] J.-M. FAVARO et D. PUMAIN. "Gibrat Revisited : An Urban Growth Model including Spatial Interaction and Innovation Cycles". In : *Geographical Analysis* 43.3 (2011), p. 261–286.

[16] Marianne HATZOPOULOU, Jiang Y. HAO et Eric J. MILLER. "Simulating the impacts of household travel on greenhouse gas emissions, urban air quality, and population exposure". In : *Transportation* 38 (2011).

[17] L. Jeff HONG et Barry L. NELSON. "A brief introduction to optimization via simulation". In : *Winter Simulation Conference*. WSC '09. Winter Simulation Conference, 2009, p. 75–85.

[18] Forrest STONEDAHL et Uri WILENSKY. "Finding forms of flocking : Evolutionary search in abm parameter spaces". In : *In Proceedings of the MABS workshop at the 9th international conference on Autonomous Agents and Multi-Agent Systems*. 2010.

[19] El-Ghazali TALBI et Vincent BACHELET. "COSEARCH : A Parallel Cooperative Metaheuristic". In : *Journal of Mathematical Modelling and Algorithms* 5.1 (2006), p. 5–22.

[20] David MEIGNAN, Abderrafiaa KOUKAM et Jean-Charles CREPUT. "Coalition-based metaheuristic : a self-adaptive metaheuristic using reinforcement learning and mimetism". In : *Journal of Heuristics* 16.6 (2010), p. 859–879.

[21] K. SCHITTKOWSKI. "NLPQL : A fortran subroutine solving constrained nonlinear programming problems". English. In : *Annals of Operations Research* 5.1-4 (1986), p. 485–500.

[22] Joshua D. GRIFFIN et Tamara G. KOLDA. "Asynchronous parallel hybrid optimization combining DIRECT and GSS". In : *Optimization Methods and Software* 25.5 (2010), p. 797–817.

[23] J. FERBER. *Les systèmes multi-agents. Vers une intelligence collective*. InterEditions, 1995.

[24] Matthew James KEELING et Pejman ROHANI. *Modeling infectious diseases in humans and animals*. Princeton : Princeton University Press, 2008.

[25] D.R. JONES, C.D. PERTTUNEN et B.E. STUCKMAN. "Lipschitzian optimization without the Lipschitz constant". English. In : *Journal of Optimization Theory and Applications* 79.1 (1993), p. 157–181.

[26] Noam GOLDBERG, Tamara G. KOLDA et Ann S. YOSHIMURA. *Concurrent optimization with DUET : Direct Using External Trial Points*. Rapp. tech. Sandia National Laboratories, 2008.

[27] Sean LUKE. *Essentials of Metaheuristics*. second. Lulu, 2013.

[28] Patricia D. HOUGH, Tamara G. KOLDA et Virginia J. TORCZON. "Asynchronous Parallel Pattern Search for Nonlinear Optimization". In : *SIAM Journal on Scientific Computing* 23.1 (jan. 2001), p. 134–156.