Développement d'un algorithme de zonage piloté par un modèle de performance d'itinéraire technique : application à la gestion durable de l'eau d'irrigation

Antoine Régnier

Ministère de l'Agriculture et de la Souveraineté alimentaire

ECOLE NATIONALE SUPÉRIEURE

des SCIENCES AGRONOMIQUES de BORDEAUX

AQUITAINE

1, cours du Général de Gaulle - CS 40201 – 33175 GRADIGNAN cedex

MEMOIRE de fin d'études

pour l'obtention du titre

d'Ingénieur de Bordeaux Sciences Agro

Développement d'un algorithme de zonage piloté par un modèle de performance d'itinéraire technique : Application à la gestion durable de l'eau d'irrigation

Présenté par : Antoine Régnier **Spécialisation : AgroTIC**

Maître de stage : Patrice LOISEL **Tutrice : Mathilde BEAUCHESNE**

Membres du jury : Lionel BOMBRUN, Natalie TOULON, Christian GERMAIN, Jérôme STEFFE

-2024-

1. Ce travail a bénéficié d'une aide de l'État gérée par l'Agence Nationale de la Recherche au titre de France 2030 portant la référence ANR-16-CONV-0004. This work was supported by the French National Research Agency under France 2030 program, referred as ANR-16-CONV-0004.

Remerciements

Je tiens à exprimer ma gratitude envers mes maîtres de stage de l'INRAE : Patrice Loisel, Sébastien Roux et Hazaël Jones pour leur agréable accompagnement. Je remercie également Mathilde Beauschenes, ma tutrice côté école, pour nos échanges qui m'ont apporté un point de vue élargi. J'aimerais aussi remercier l'ensemble des stagiaires de l'UMR MISTEA, ainsi que ceux d'Eco&Sols, sans qui ce stage aurait été un peu moins gai. Plus généralement, j'ai une pensée tous les gens que j'ai rencontré durant ce stage et qui ont enrichi cette expérience.

Résumé

Le changement climatique, en augmentant la fréquence des aléas et perturbant les cycles des cultures, a amené de nouveaux enjeux dans le monde agricole. Les contrôle des intrants est devenu une problématique majeure, que ce soit pour des raisons économique ou environnementales. Ainsi, de nombreux instituts de recherche, dont l'INRAE, se sont penchés sur l'agriculture de précision pour tenter de répondre à ces enjeux.

Le travail présenté dans ce mémoire s'inscrit dans la continuité de deux autres projets de l'UMR MISTEA. Le premier projet, *geozoning*, propose un algorithme de zonage se basant sur la variabilité des données parcellaires. Le deuxième projet porte sur l'optimisation des performances d'itinéraires techniques en irrigation à partir d'un modèle de culture simplifié. Durant ce stage, une nouvelle méthode de zonage a été développée, où on se sert du modèle de performance d'itinéraire technique pour calculer un critère de qualité du zonage. Ensuite, grâce à une méthode d'optimisation, on maximise ce critère de qualité, tout en respectant des contraintes spatiales définies. Le contexte de l'irrigation intra-parcellaire va servir d'étude de faisabilité de la méthode.

Dans un premier temps, une analyse du fonctionnement de *geozoning* et du modèle de performance d'itinéraire technique a été réalisée. Dans un deuxième temps, on s'est intéressé à un cas d'optimisation aspatial, où on ne tient pas compte de la spatialisation des données. Ce problème a permis de tester différentes méthodes d'optimisation, qui ont servi pour le reste du projet. Finalement, un algorithme de zonage piloté par le modèle de performance d'itinéraire technique a été développé.

La comparaison des rendements calculés par cette méthode et ceux obtenus en appliquant une irrigation homogène d'une parcelle générée montrent l'intérêt de cet algorithme de zonage piloté par un modèle de performance d'itinéraire technique. La méthode est assez générique pour être adaptée à d'autres domaines de l'agriculture de précision. Néanmoins, des analyses sur des données terrains réelles devront être menées pour consolider les conditions d'usage de la méthode.

MOTS CLÉS : carte de modulation - zonage - modèle de performance - agriculture de précision - optimisation - irrigation - étude de faisabilité

Abstract

Climate change, by increasing the frequency of hazards and disrupting crop cycles, has brought new challenges to the world of agriculture. Controlling inputs has become a major issue, for both economic and environmental reasons. Numerous research institutes, including INRAE, have therefore turned their attention to precision agriculture in an attempt to meet these challenges.

The work presented in this dissertation is a continuation of two other UMR MISTEA projects. The first project, *geozoning*, proposes a zoning algorithm based on the variability of plot data. The second project involves optimising the performance of technical irrigation itineraries using a simplified crop model. During this internship, a new zoning method was developed, using the technical itinerary performance model to calculate a zoning quality criterion. Then, using an optimisation method, this quality criterion is maximised, while respecting defined spatial constraints. The context of intra-parcel irrigation will serve as a feasibility study for the method.

Firstly, an analysis of the operation of *geozoning* and the technical itinerary performance model was carried out. Secondly, we looked at a case of aspatial optimisation, where the spatialisation of the data is not taken into account. This problem made it possible to test different optimisation methods, which were used for the rest of the project. Finally, a zoning algorithm driven by the technical itinerary performance model was developed.

A comparison of the yields calculated by this method and those obtained by applying homogeneous irrigation to a generated plot shows the value of this zoning algorithm driven by a technical itinerary performance model. The method is generic enough to be adapted to other areas of precision agriculture. Nevertheless, analyses based on real field data will have to be carried out to consolidate the conditions of use of the method.

KEY WORDS : modulation map - zoning - performance model - precision agriculture - optimization - irrigation - feasibility study

Table des matières

1 Contexte

1.1 Cadre et fondations du stage

1.1.1 Présentation générale de la structure d'accueil

L'Institut national de recherche pour l'agriculture, l'alimentation et l'environnement (INRAE) est un organisme de recherche public placé sous la tutelle du ministère de l'enseignement supérieur et de la recherche ainsi que du ministère de l'agriculture et de la souveraineté alimentaire.

Les recherches de l'INRAE sont axées sur 5 orientations scientifiques (OS) :

— OS 1 : Répondre aux enjeux environnementaux et gérer les risques associés
— OS 2 : Accélérer les transitions agroécologique et alimentaire, en tenant compte des enjeux économiques et sociaux
— OS 3 : Une bioéconomie basée sur une utilisation sobre et circulaire des ressources
— OS 4 : Favoriser une approche globale de la santé
— OS 5 : Mobiliser la science des données et les technologies du numérique au service des transitions

L'INRAE compte près de 8200 agents titulaires, dont environ 2000 chercheurs, 3200 ingénieurs et 3000 techniciens, répartis dans 18 centres de recherche, au sein de 14 départements différents.

J'ai réalisé mon stage dans l'UMR MISTEA (Mathématiques, Informatique et STatistique pour l'Environnement et l'Agronomie), qui est une unité mixte de recherche de l'INRAE au sein du département MathNum, spécialisé sur les mathématiques et le numérique. Elle est installée sur le campus de La Gaillarde, à Montpellier. Elle est structurée en 3 axes de recherche :

— L'axe Systèmes Dynamiques
— L'axe Probabilités Statistiques
— L'axe Informatique

1.1.2 Contexte du stage

Près de 70% de la consommation d'eau douce est dédiée à l'agriculture, et plus particulièrement à son irrigation. Ainsi, l'agriculture est le premier secteur touché par les sécheresses et l'épuisement des nappes phréatiques. Or, avec le changement climatique, les sécheresses vont devenir de plus en plus récurrentes. Dans ce contexte, il est nécessaire d'adopter des méthodes d'irrigation plus durables et moins coûteuses. Pour cela, on se tourne vers l'agriculture de précision, qui vise à optimiser les intrants à l'échelle parcellaire. En agriculture de précision, bien connaître sa parcelle permet d'économiser sur son apport d'intrant, et d'augmenter ses chances d'avoir un meilleur rendement. L'enjeu est de pouvoir s'adapter à une hétérogénéité intra-parcellaire, en définissant des itinéraires techniques qui sont différenciés spatialement.

Pour cela, des méthodes permettant de synthétiser les données intra-parcellaire ont été crées. En général, on définit un zonage, qui consiste à associer les données obtenues à partir d'une ou plusieurs cartes sous la forme de zones, puis à définir des traitements pour les zones. Les cartes résultantes associant zones et traitement par zones sont appelées carte de modulation (figure 1). Ces cartes se servent donc de données géoréférencées afin de rendre compte de l'état de la parcelle par une approche spatiale. Les cartes de modulation sont utilisées pour de nombreuses applications :

en général, elles permettent la modulation d'intrant, en vue de réaliser des économies sur le coût des traitements (ex : modulation de dose, modulation d'engrais). Dans notre cas, on va s'intéresser plus particulièrement à l'irrigation d'une parcelle.

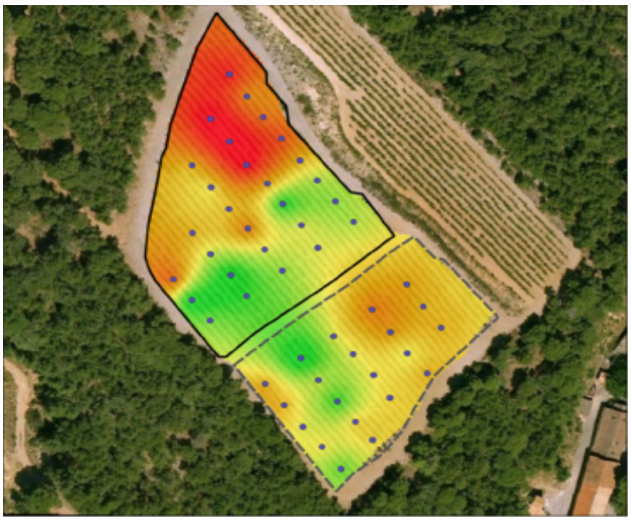


FIGURE 1 – exemple d'une carte de modulation

Afin d'évaluer la qualité d'un zonage, il est essentiel de prendre en compte trois contraintes :

— L'homogénéité intra-zone : les données au sein d'une même zone doivent être de valeur assez proches
— L'hétérogénéité inter-zone : les données d'une zone doivent être éloignées de celles d'une autre zone de label différent
— La prise en compte des contraintes opérationelles : les zones doivent respecter des contraintes de tailles et de forme qui doivent prendre en compte les contraintes du travail agricole effectué sur la parcelle. Par exemple, pour une carte de modulation de pesticide, les zones ne doivent pas être plus fines que l'envergure du pulvérisateur.

Le travail que j'ai réalisé pendant ce stage s'inscrit dans la continuité de deux projets mis en place par l'équipe "Systèmes Dynamiques" de l'UMR MISTEA. Le premier projet, *Geozoning*, a porté sur la conception d'un algorithme de zonage nommé Geozoning [10] qui prend en compte les trois contraintes citées ci-dessus. Le deuxième projet porte sur l'optimisation des performances d'itinéraires techniques en irrigation à partir d'un modèle de culture simplifié [6, 1]. Les modèles de performance d'itinéraires technique sont des outils d'aide à la décision qui permettent de choisir un itinéraire technique pour une culture donnée. Ici, cette approche calcule un plan d'irrigation respectant une quantité q d'eau apportée. À l'issue de ce plan d'irrigation, le modèle prévoit qu'une quantité B de biomasse sera récoltée.

L'objectif de ce stage est de réaliser un algorithme de zonage piloté par un modèle de performance d'itinéraire technique, sur cas d'étude d'irrigation de précision. On veut créer des cartes de modulations, mais ici, la méthode va différer des approches classiques : au lieu de définir des zones puis de leur associer un traitement, il va y avoir un couplage entre la définition

des zones et l'optimisation des traitements par zone. Le cas de l'irrigation parcellaire permet de réaliser une étude de faisabilité de la méthode. Dans notre cas d'étude, on va notamment comparer cette approche avec une approche d'irrigation homogène, où on irrigue uniformément la parcelle.

Pour réaliser ce couplage, on suppose que l'hétérogénéité spatiale de la sortie du modèle de performance d'itinéraire technique (ici, il s'agit de la biomasse récoltée B) est due à l'hétérogénéité d'un paramètre du modèle de culture (ici, la réserve totale transpirable du sol) dont on dispose d'une carte haute résolution des mesures. Les autres paramètres du modèle sont identiques dans la parcelle.

La structure générale de l'algorithme sera similaire à celle de *geozoning*, à la différence qu'ici, au lieu d'utiliser les critères d'homogénéité intra-zone et d'hétérogénéité inter-zones, on se sert du modèle de performance d'itinéraire technique pour faire ressortir un critère de qualité. L'idée est que la méthode générique développée puisse être utilisable avec n'importe quel modèle de performance d'itinéraire technique pourvu que l'hétérogénéité spatiale ne soit due qu'à un seul paramètre du modèle. Néanmoins, pour ce stage, c'est le modèle de performance d'irrigation conçu au sein de l'UMR MISTEA qui va être utilisé.

La réalisation de ce travail a été faite en trois étapes clés :
— Compréhension de l'existant : compréhension du fonctionnement de *geozoning* et du modèle de performance de culture
— Problème d'optimisation aspatiale : Utilisation du modèle de performance de culture afin d'obtenir une carte labellisée qui servira de point de départ à l'algorithme de zonage
— Réalisation d'un processus de zonage inspiré de *geozoning*, modifiant la carte labellisée afin de respecter les contraintes spatiales.

A Retenir...

— L'objectif du stage est de réaliser des cartes de modulation parcellaires basées sur des données intraparcellaires géospatialisées
— Pour cela, on va s'inspirer de l'algorithme de *geozoning*, en le couplant à un modèle de performance d'itinéraire technique en irrigation

1.2 Analyse de l'existant

1.2.1 Analyse de *geozoning*

Geozoning [10] est une méthode permettant d'obtenir un zonage d'une parcelle selon les valeurs d'une variable agronomique d'intérêt. Les données peuvent être sous la forme d'une grille ou de cellules. L'objectif du zonage est de maximiser la variabilité inter zones, tout en minimisant la variabilité intra zone. Pour cela, un critère combinant ces deux objectifs est mis en place. Pour chaque paire de zones voisines I et J, on définit un critère de contraste entre I et J comme le rapport entre l'hétérogénéité inter-zones M_{IJ} et l'hétérogénéité intra-zone moyenne $\frac{M_{II}+M_{JJ}}{2}$. On a donc :

$$C_{I,J} = \frac{2M_{IJ}}{M_{II}+M_{JJ}}$$

En simplifiant l'expression, on obtient :

$$C_{I,J} = 1 + \frac{(\mu_I - \mu_J)^2}{\sigma_I^2 + \sigma_J^2}$$

où μ_I et μ_J correspondent à la moyenne des données dans la zone I et dans la zone J, σ_I et σ_J correspondent à l'écart-type des données dans la zone I et dans la zone J. En calculant les contrastes de chaque zone avec chacun de ses voisins, et en conservant la valeur minimale, on se retrouve avec un indicateur de contraste entre une zone et son voisinage. On obtient donc un critère qui définit la qualité d'un ensemble des zones définies dans un zonage Z :

$$\text{Crit}(Z) = \min_{I \in Z} \min_{J \in N(I)} C_{I,J}$$

où $N(I)$ est l'ensemble des zones voisines de la zone I.

Pour obtenir des zonages, on utilise des quantiles. Les quantiles sont des valeurs qui permettent de diviser un jeu de données en intervalles de même probabilités. Pour simplifier, dans le reste de cette étude, on nomme vecteur de quantiles les probabilités associées aux quantiles. Par exemple, pour 3 labels, et en utilisant le vecteur (0.1, 0.8), on a les 10% des données de plus faibles valeurs qui sont associées au label 1, les 70% suivants sont associés au label 2, et le reste est associé au label 3. Ainsi, chaque point est associé à un label, et les points adjacents de même label forment une zone.

Sur la figue 2, on voit que les zones de label A contiennent les données aux valeurs les plus faibles, représentées en marron, qui ne représentent que 10% des données totales de la parcelles. Les 20% de données de plus grande valeur, représentées d'une couleur plus claire, sont associées au label C. Les 70% des données restantes sont associées au label B.

On peut voir que certaines zones, comme les trois zones de label *A* encadrées dans la figure, sont de trop dans ce zonage. Que ce soit pour une question de compréhension de la carte ou dans le cadre de certains travaux agricoles sur la parcelle qui ne permettraient pas une telle précision, on ne peut pas les considérer valides. Ces zones trop petites vont donc être traitées, au travers d'un arbre de décision.

Pour chaque zone, 2 traitements algorithmiques possibles vont être essayés :

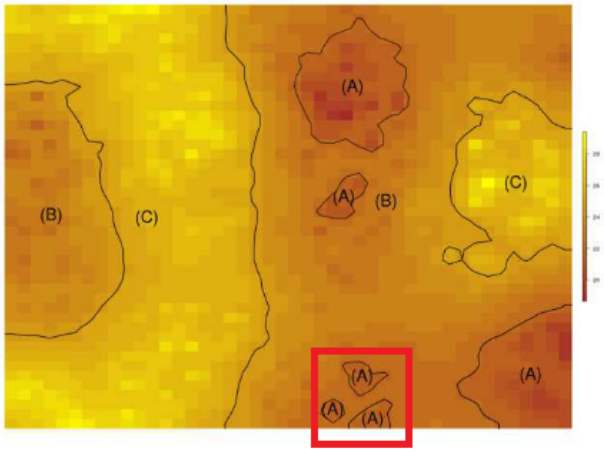


FIGURE 2 – exemple de carte issue d'un vecteur de quantiles $(0.1, 0.8)$

— Soit la zone est assimilée par la zone environnante, et les points de la zone sont associés au label de la zone environnante.
— Soit la zone est agrandie afin d'obtenir une taille suffisante. Si elle est proche d'une autre zone de même label, les deux zones sont fusionnées.

Chaque choix dans l'arbre de décision va aboutir à un zonage final différent. Le critère $Crit(Z)$ permet de comparer ces zonages, et de conserver le meilleur.

La méthode décrite ci-dessus permet d'obtenir le meilleur zonage possible selon le critère $Crit(Z)$ pour un vecteur de quantiles, soit pour un zonage initial. Pour obtenir le meilleur zonage possible, il faut aussi comparer les meilleurs zonages pour chaque vecteur de quantiles dans un ensemble discrétisé. Par exemple, si on fixe le nombre de label à 3, un pas de 0.1 et un écart de 0.2 entre deux quantiles, on obtient l'ensemble $\mathscr{P}(3) = \{(0.1, 0.3), (0.1, 0.4), ..., (0.7, 0.9)\}$

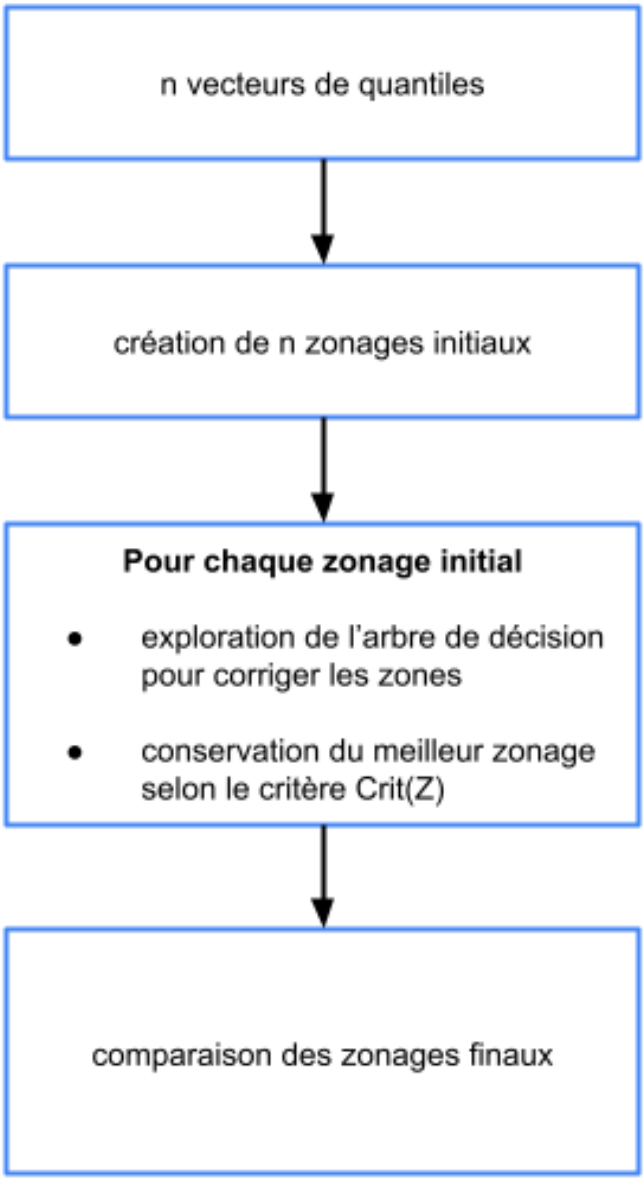


FIGURE 3 – résumé de la méthode *geozoning*

1.2.2 Description du modèle de performance agronomique utilisé

Le modèle de performance agronomique utilisé a été développé par des membres de l'UMR MISTEA. Il s'appuie sur un modèle de croissance d'une culture impactée par le stress hydrique [6]. Il se base sur l'évolution de deux variables au cours du temps :

— $S(t)$ correspond à l'humidité relative du sol. Elle est influencée par l'évapotranspiration de la plante, la pluie, l'irrigation, le drainage et l'évaporation du sol.

— $B(t)$ correspond à la biomasse récoltable. Elle est influencée par la transpiration de la plante. Son calcul est basé sur le modèle Aquacrop de la FAO (Organisation des Nations unies pour l'alimentation et l'agriculture)

A l'aide de ce modèle de culture et d'une méthode d'optimisation, il est possible de trouver un planning d'irrigation qui distribue une quantité totale Q d'eau à la plante au fil du temps, tout en maximisant la biomasse B, qui correspond à la biomasse de la plante lors de la récolte.

Les entrées spatialisées du modèle de performance agronomique vont être les suivantes :

— $v(x)$: entrée spatialisée du modèle de culture. Elle correspond à la réserve totale transpirable au point x.

— $q(x)$: quota d'irrigation. Il correspond à la quantité d'eau que le modèle peut distribuer à la plante au point x. Le quota conditionne le planning d'irrigation de la plante.

En sortie, le modèle donne $B(v,q)$. Il s'agit de la biomasse obtenue en appliquant le quota d'irrigation q sur un point de réserve transpirable v. Pour réaliser le calcul de $B(v,q)$, il est également nécessaire de connaître les valeurs des autres entrées du modèle de culture. Comme on fait l'hypothèse que ces autres entrées sont spatialement invariantes, on ne va pas les préciser ici.

Dans le reste de ce stage, la fonction $B(v,q)$ va être utilisée dans le problème d'optimisation. Comme on va avoir besoin de nombreuses fois de $B(v,q)$, cela signifie qu'il faudra faire tourner un grand nombre de fois le modèle, ce qui pourrait avoir de lourds impacts sur la performance du code. Pour ces raisons, on va se servir d'un tableau de correspondance de B selon les valeurs de v et de q (figure 4).

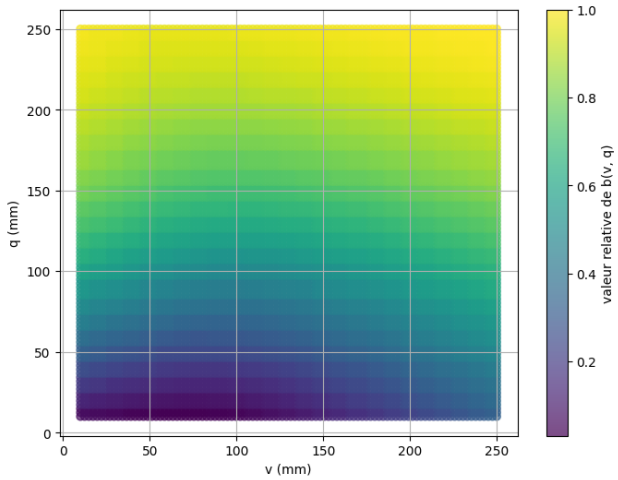


FIGURE 4 – tableau de correspondance de B selon les valeurs de v et de q

A Retenir...

- *geozoning* utilise un critère qui combine la variabilité intra-zone et la variabilité inter zone
- *geozoning* explore de façon quasi systématique les quantiles de la variable d'intérêt
- *geozoning* pour chaque vecteur de quantile part d'une carte initiale, et explore un arbre de décision afin de corriger les zones trop petites
- Le modèle de performance d'itinéraire technique renvoie une quantité $B(v,q)$ de biomasse récoltée pour une réserve totale transpirable v et un quota d'irrigation q

2 Matériels et méthodes

Le travail réalisé lors de ce stage s'est fait en deux temps :

— L'optimisation aspatiale, où on ne tient pas compte de la spatialisation des données.

— L'optimisation spatiale ou zonage, où des contraintes spatiales ont été ajoutées.

L'optimisation aspatiale a permis de me familiariser avec les méthodes d'optimisation. L'idée était de tester et développer des outils qui me serviront dans l'obtention d'un zonage optimal en résolvant une optimisation spatiale, qui constitue l'objectif de ce stage. Dans la suite, après avoir présenté dans un premier temps plus en détail le problème, nous aborderons le cas aspatial et le cas aspatial.

2.1 Choix de logiciel

Tout d'abord, la question du choix du langage de programmation est la première à avoir été posée. Les candidats les plus pertinents étaient Python et Julia.

— Julia est un langage de programmation conçu pour le calcul scientifique. Il est notamment particulièrement adapté pour des problèmes d'optimisation. Aussi, il possède de nombreuses librairies qui permettent de manipuler des objets spatiaux.

— Python est un langage plus généraliste, mais qui dispose d'un grand nombre de bibliothèques, que ce soit pour l'optimisation ou la gestion d'objets spatiaux. C'est un langage mieux connu de moi-même et de mon équipe encadrante.

Le choix final s'est donc porté sur Python, qui possède une communauté plus active, et qui m'est déjà familier.

2.2 Présentation du problème

Ici, on s'intéresse à une parcelle agricole. L'objectif de l'optimisation, est, à partir des données de la réserve totale transpirable v sur une parcelle et d'un modèle de performance d'itinéraires techniques, d'être capable :

— d'associer un label à chaque point de la parcelle

— d'affecter à chaque point associé au label l une quantité d'eau q_l

tout en maximisant la biomasse totale de la parcelle B qui est calculée grâce au modèle de performance d'itinéraire technique. On devra également faire en sorte de ne pas dépenser une quantité d'eau totale supérieur à un quota Q. Chaque label est associé à un plan d'irrigation. Dans l'ensemble des exemples, nous nous limiterons à 3 labels différents.

Les données parcellaires utilisées lors de ce stage seront générées par géostatistiques. Cela a certes les inconvénients d'être moins réaliste et de ne pas prendre en compte les aléas du terrain, mais des données générées ont deux avantages non négligeables. Elles sont faciles à générer et à modifier, ce qui est plus pratique dans une phase de développement, pour par exemple tester des cas particuliers.

Dans le cadre de ce stage, les données générées sont des grilles de points des valeurs de v, l'exemple repris dans cette partie sera celui de la figure 5. Les lignes rouges représentent les iso-

contours, c'est à dire les points de même valeur de v. Cela donne une idée du gradient de v sur la carte.

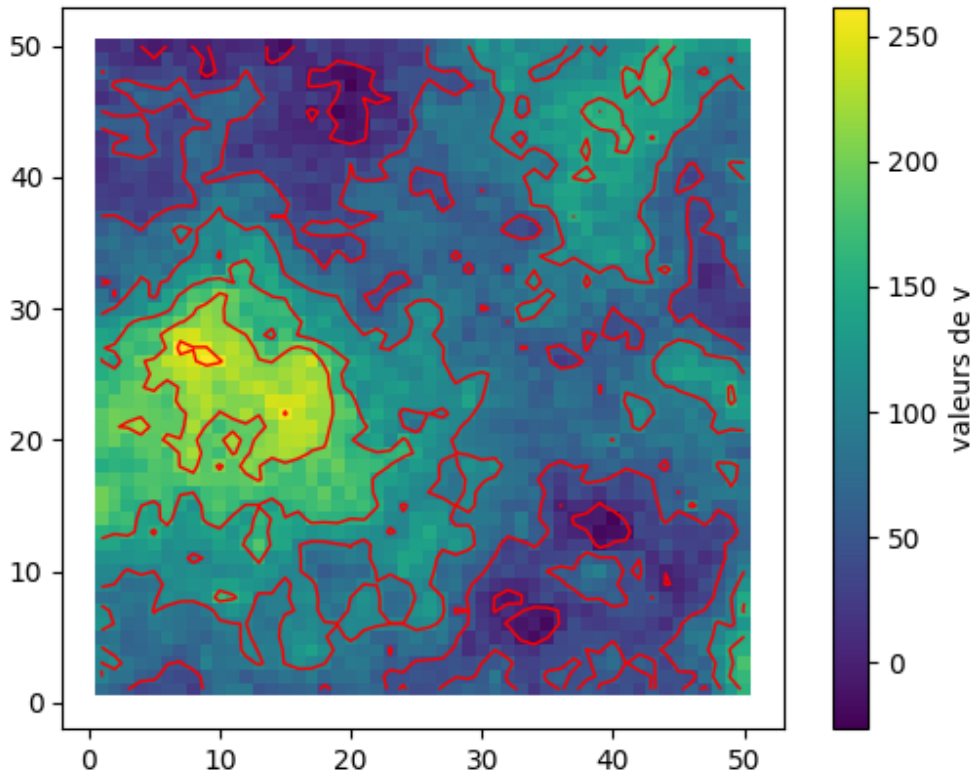


FIGURE 5 – exemple de carte parcellaire générée par géostatistique

Comme dans la méthode *geozoning*, pour obtenir un zonage optimal il est raisonnablement clair que les frontières des zones doivent suivre ces lignes de niveau ou isocontours de la variable d'intérêt. On va donc également utiliser des vecteurs de quantiles. Pour un vecteur de quantiles fixé, celui-ci nous permet de diviser le jeu de données selon le nombre de labels. En figure 6, on prend l'exemple d'une liste ordonnée des valeurs de v de taille 100, et d'un vecteur de quantiles $(0.3, 0.8)$. Les 30 premières valeurs seront associées au label 1, les 50 suivantes seront associées au label 2, et les 20 dernières seront associées au label 3. Aussi, le modèle de performance d'itinéraire technique prend en compte un quota d'irrigation q. Comme chaque label correspond à un plan d'irrigation différent, on considère le vecteur $(q_1, ..., q_L)$ qui associe à chaque label un quota d'irritation q pour un nombre de label L. La figure 6 montre comment les quotas d'irrigation q sont associés aux données, en reprenant l'exemple du vecteur de quantiles $(0.3, 0.8)$.

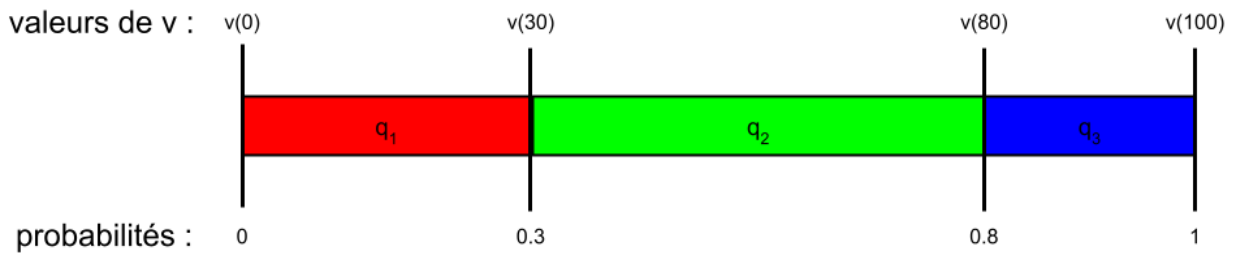


FIGURE 6 – exemple d'attribution des labels aux données

Voici plus précisément les notations et la structure du problème mathématique à résoudre :

Données

- Une grille : x_{ij}. Nous notons $X = \#\{x_{ij}\}$.
- Un tableau de valeurs de réserve transpirable : $v_{ij} = v(x_{ij})$ avec $\underline{v} \leq v_{ij} \leq \overline{v}$.
- Une fonction : $B(v,q)$ qui évalue la biomasse obtenue pour la valeur v et le quota d'irrigation q. Cette fonction est donnée sous la forme d'une matrice évaluée pour des valeurs v_m et des quotas q_l.

Dans notre cas, les valeurs de biomasse $B(v_m, q_n)$ sont fournies en sortie du modèle [1].
Soit $v_{ij} = v(x_{ij})$ la réserve totale transpirable au point x de coordonnées (i,j), on définit $B(v_{ij}, q)$ la biomasse obtenue en x_{ij} en appliquant un quota q.

Le but étant de trouver un zonage (c'est-à-dire attribuer un label à chaque point de la parcelle) qui maximise la biomasse sur la grille, on définit la fonction $\mathscr{L}$ qui associe à chaque point x_{ij} son label $l = \mathscr{L}((i,j))$.

Formulation mathématique du problème

Pour un nombre de labels L fixé, le critère à maximiser est le suivant :
En agissant sur $\mathscr{L}(.)$ et $\mathbf{q}= (q_l)l$, on cherche à maximiser :

$$(\mathscr{P}) \quad \max_{\mathscr{L}(.)\in\mathscr{C},\ (q_l)_l} S(Q, \mathscr{L}(.), \mathbf{q}) = \sum_{l=1}^{L} \sum_{\mathscr{L}((i,j))=l} B(v_{ij}, q_l)$$

sous les contraintes portant sur les quotas : $q_l \geq 0,\ l = 1,..,L$ et

$$\sum_{l=1}^{L} \#\{(i,j) | \mathscr{L}((i,j)) = l\} q_l \leq Q$$

et où $\mathscr{C}$ est l'ensemble des fonctions $(.)$ qui génèrent des zonages admissibles c.a.d qui respectent un certain nombre de contraintes spatiales.

Ce problème d'optimisation se décompose en deux problèmes emboités d'optimisation :

$$\max_{\mathscr{L}(.)\in\mathscr{C},\ (q_l)_l} S(Q,\mathscr{L}(.),\mathbf{q}) = \max_{\mathscr{L}(.)\in\mathscr{C}} J(\mathscr{L}(.))$$

où :

$$J(\mathscr{L}(.)) = \max_{(q_l)_l} \sum_{l=1}^{L} \sum_{\mathscr{L}((i,j))=l} B(v_{ij},q_l)$$

sous les contraintes : $q_l \geq 0,\ l = 1,..,L$ et

$$\sum_{l=1}^{L} \#\{(i,j)|\mathscr{L}((i,j)) = l\} q_l \leq Q$$

A Retenir...

- L'objectif du problème est de maximiser une biomasse totale S calculée grâce à au modèle de performance d'itinéraire technique
- On peut agir $\mathscr{L}(.)$, qui attribue un label à chaque point, et sur q_l, le quota d'irrigation attribué à chaque point de label l
- Il y a une contrainte de quantité d'eau totale Q apportée à la parcelle à ne pas dépasser

2.3 Optimisation des quotas à label fixé

2.3.1 Problème aspatial

Dans ce cas précis, il n'y a plus de contrainte spatiale (c'est-à-dire pas de contrainte sur $\mathscr{L}$) et le quota appliqué ne dépend plus de la position d'un point x mais uniquement de la valeur de la réserve totale respirable en ce point $v(x)$. Les labels sont de ce fait très simples à caractériser, ils sont délimité par des valeurs de coupure de v : $p_l, l = 0,..,L$ avec $p_0 = \underline{v}$ et $p_L = \bar{v}$. La fonction $\mathscr{L}$ est alors caractérisée par $\mathscr{L}((i,j)) = l$ si et seulement si $p_{l-1} \leq v_{ij} < p_l$ et le problème $(\mathscr{P})$ se réécrit en terme de $(p_l)_l$:

$$(\mathscr{P}_0) \quad \max_{(p_l)_l} J_0(p_0,...,p_L)$$

où :

$$J_0(p_1,..,p_l) = \max_{(q_l)_l} \sum_{l=1}^{L} \sum_{p_{l-1}\leq v_{ij}<p_l} B(v_{ij},q_l)$$

sous les contraintes : $q_l \geq 0, l = 1,..,L$ et

$$\sum_{l=1}^{L} \#\{(i,j)|p_{l-1} \leq v_{ij} < p_l\} q_l \leq Q$$

Définissant θ_l comme la proportion de points pour lesquels $v_{ij} \leq p_l$ c'est-à-dire $\dfrac{\#\{(i,j)|v_{ij} \leq p_l\}}{X}$, la dernière contrainte devient :

$$\sum_{l=1}^{L} (\theta_l - \theta_{l-1}) q_l \leq \frac{Q}{X}$$

Maximisation de J_0

La résolution de ce problème utilise les conditions nécessaires d'optimalité du premier ordre dite de Karush-Kuhn-Tucker (fournies en Annexe A). Ici f (indiquée en Annexe A) à maximiser correspond à la somme des valeurs de B aux valeurs de v et q et les contraintes sont $q_l \geq 0, l = 1,..,L$ et $\sum_{l=1}^{L}(\theta_l - \theta_{l-1})q_l \leq \frac{Q}{X}$, on a donc pour les dérivées par rapport à q_l :

$$\sum_{p_{l-1} \leq v_{ij} \leq p_l} \frac{\partial B}{\partial q}(v_{ij}, q_l) + \lambda_l - \lambda(\theta_l - \theta_{l-1}) = 0$$

En multipliant par q_l et utilisant (1) :

$$\sum_{p_{l-1} \leq v_{ij} \leq p_l} q_l \frac{\partial B}{\partial q}(v_{ij}, q_l) = \lambda(\theta_l - \theta_{l-1})q_l$$

On en déduit que pour tout l : $\frac{1}{\theta_l - \theta_{l-1}} \sum_{p_{l-1} \leq v_{ij} \leq p_l} \frac{\partial B}{\partial q}(v_{ij}, q_l) = \lambda$ ou $q_l = 0$.

En discrétisant q par pas de 10 on peut tester toutes les configurations de q pour chaque label en calculant numériquement la dérivée. Comme la contrainte est unique, λ est le même pour chaque quota correspondant à une solution optimale. On peut donc sélectionner les listes de quotas pour lesquelles λ est identique pour tous les labels (ou un sous-ensemble dans le cas où certains quotas seraient nuls), et retenir celles qui respectent la contrainte $\sum_{l=1}^{L}(p_l - p_{l-1})q_l \leq \frac{Q}{X}$. Pour ce faire on représente (figure 7) dans une table (quota q, label l) les isocontours de λ. A chaque isocontour correspond une quantité totale utilisée et on sélectionne l'isocontour qui correspond à la quantité totale Q. La liste de quotas par label se déduit en parcourant l'isocontour sélectionné.

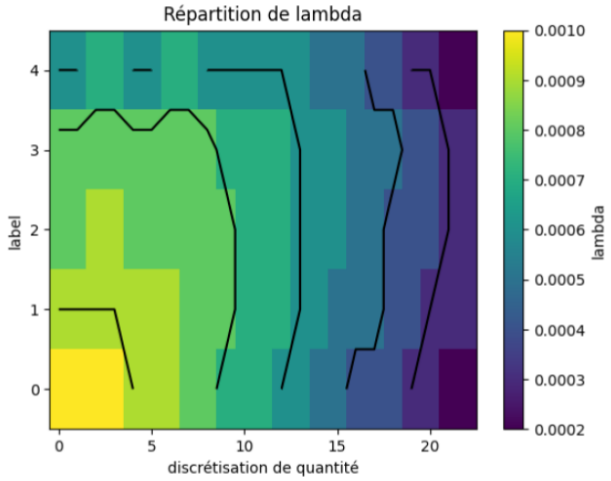


FIGURE 7 – exemple de répartition des valeurs de lambda pour 5 labels

Puisque cette méthode est quasiment analytique, on s'attend à ce que ce soit celle qui fonctionne le mieux. Pour cette raison, c'est celle là qui va être utilisée dans l'optimisation spatiale.

2.3.2 Comparaisons de différentes méthodes d'optimisation

Pour répondre à ce problème d'optimisation avec contrainte à deux vecteurs de variables, il existe différentes solutions qui ont été testées.

Aussi, pour les trois méthodes d'optimisation qui vont être comparées, on va optimiser le vecteur de quantiles $(p_0, ..., p_L)$ de la même manière que dans *geozoning*. C'est à dire que plutôt d'essayer d'optimiser les deux vecteurs de variables en même temps, on va d'abord fixer les valeurs du vecteur de quantile, puis on va réaliser l'optimisation sur le vecteur de quotas d'irrigation $(q_1, ..., q_L)$. Pour cela, on définit des règles, afin d'obtenir l'ensemble des vecteurs de quantile qui respectent ces règles. Par exemple, si on fixe le nombre de label à 3, un pas de 0.1 et un écart de 0.2 entre deux quantiles, on obtient l'ensemble $P(3) = \{(0.1, 0.3), (0.1, 0.4), ..., (0.7, 0.9)\}$.
Dans ce cas, on va procéder à l'optimisation du vecteur $(q_1, ..., q_L)$ pour $(0.1, 0.3)$, puis une nouvelle fois pour $(0.1, 0.4)$ et ainsi de suite... À la fin, on va pouvoir conserver la meilleure optimisation et ainsi obtenir le meilleur couple de vecteurs $(p_0, ..., p_L)$ et $(q_1, ..., q_L)$.

Pour l'ensemble de ces optimisations, deux librairies ont été envisagées : SciPy [15] et Pyomo [3] [5]. Les deux sont des librairies qui proposent d'utiliser des solveurs pour résoudre des problèmes d'optimisation. L'avantage de SciPy est que les solveurs sont directement intégrés à la librairie, tandis que Pyomo nécessite d'installer chaque solveur indépendemment. Un solveur est un programme qui va prendre en entrée des variables, une fonction objectif et éventuellement des contraintes, afin de trouver les valeurs de variable qui maximisent ou minimisent la fonction objectif. Chaque méthode ici va nécessiter différents solveurs, et pour chaque méthode, des solveurs vont être comparés. Pour cette raison, on a préféré utiliser uniquement SciPy pour un soucis de gain de temps. Aussi, le package Scipy est plus reconnu par la communauté, avec environ 13000 recommandations contre 2000 pour Pyomo.

Pour l'optimisation aspatiale, on va comparer différents solveurs pour chaque méthode d'optimisation vue plus haut. On retiendra la méthode et le solveur qui présente le meilleur critère de biomasse totale, tout en limitant le temps d'éxecution.

Optimisation sous contraintes

L'optimisation avec contrainte est la méthode la plus simple à utiliser. L'idée est que la contrainte est directement intégrée au solveur. Pour faire fonctionner cette méthode d'optimisation, il suffit donc de donner 3 éléments à SciPy :

— la fonction objectif, qui est la somme S que l'on cherche à maximiser
— le vecteur de variables $(q_1, ..., q_L)$
— la fonction de contrainte

Pour l'optimisation avec contrainte, trois solveurs différents on été testés :

— COBYLA (Constrained Optimisation BY Linear Approximation) [12]
— SLSQP (Sequential Least Squares Programming Algorithm) [8]

— trust-constr [14]

Optimisation avec élimination de contrainte

L'idée est de réaliser un changement de variables sur q_l, de sorte à ce que la contrainte soit respectée, quelles que soient les valeurs de la nouvelle variable. Ainsi, on n'aura pas besoin d'intégrer la contrainte au solveur.

A θ_l fixé, on cherche à définir q_l afin de maximiser S. On définit a_l tel que :

$$\forall l \in 1,..,L, \quad q_l = \frac{1}{\theta_l - \theta_{l-1}} \frac{a_l^2}{1 + \sum_{i=1}^{L} a_i^2} \frac{Q}{X}$$

Ainsi, on a :

$$\sum_{l=1}^{L} (\theta_l - \theta_{l-1}) q_l = \sum_{l=1}^{L} \frac{a_l^2}{1 + \sum_{i=1}^{L} a_i^2} \frac{Q}{X} \leq \frac{Q}{X}$$

Les contraintes sont donc respectées pour toutes valeurs de a_l, et on doit maximiser :

$$J_1 = \sum_{l=1}^{L} \sum_{\mathscr{L}((i,j))=l} B(v_{ij}, \frac{a_l^2}{1 + \sum_{k=1}^{L} a_k^2} \frac{Q}{(\theta_l - \theta_{l-1})X})$$

L'optimisation avec élimination de contrainte va s'exécuter de la même manière que pour l'optimisation avec contrainte, à la différence qu'il n'y a pas besoin de donner une fonction de contrainte puisqu'elle a déjà été gérée par le changement de variable. L'absence de contraintes permet également d'utiliser d'autres solveurs que ceux cités précedemment. Trois solveurs ont donc été testés :

— Nelder-Mead [11]
— Powell [13]
— BFGS (Broyden–Fletcher–Goldfarb–Shanno) [2]
— CG (Conjugate Gradients) [4]

A Retenir...

— Le but de l'optimisation aspatiale est de comparer différentes méthodes d'optimisation.
— Pour cela, la librairie SciPy du langage Python est utilisée. Elle permet d'utiliser des solveurs, qui permettent de résoudre des problèmes d'optimisation.
— Différents solveurs vont être testés et comparées.

2.3.3 Problème spatial

Revenons au problème général avec contraintes :

$$(\mathscr{P}_1) \quad S_1 = \max_{\mathscr{L}(.)\in\mathscr{C}} J_1(\mathscr{L}(.))$$

La maximisation du critère J_1 peut se faire en appliquant comme pour le problème aspatial les conditions de Karish-Kuhn-Tucker. Ces conditions pour le problème spatial sont obtenues en remplaçant θ_l de la section 2.3.1 par μ_l qui est définit par : $\mu_l = \frac{\#\{(i,j)|\mathscr{L}((i,j)) \leq l\}}{X}$, la proportion de points de label l. Concrètement cette optimisation est un peu différente que celle pour le problème aspatial puisqu'au lieu d'utiliser le vecteur de quantiles $(p_0,...,p_L)$, on utilise le label associé à chaque point.

Afin d'affiner l'optimisation avec les conditions de Karush-Kuhn-Tucker, on peut la faire suivre d'une optimisation avec élimination de contrainte, en se servant des résultats de l'optimisation avec les conditions de Karush-Kuhn-Tucker comme conditions initiales d'une optimisation sans contraintes. C'est ce qui va être fait ici, en utilisant le solveur "Nelder-Mead".

Comme pour le problème aspatial, pour utiliser une méthode d'optimisation sans contraintes nous effectuons un changement de variable similaire en remplaçant θ_l par μ_l définis ci-dessus.

Ainsi, on réalise une optimisation seulement sur le vecteur de quotas d'irrigation $(q_1,...,q_L)$, mais on peut tout de même utiliser les mêmes méthodes que celles vues précédemment. L'optimisation aspatiale nous permet de ressortir la biomasse totale S_1 récoltée sur la parcelle, qui va pouvoir nous servir de critère de qualité du zonage.

2.4 Obtention d'un zonage cohérent grâce à une optimisation spatiale

2.4.1 Création de la carte initiale de zonage

La première étape du zonage consiste à transformer les résultats de l'optimisation aspatiale en une carte de la parcelle labellisée. Pour ce faire, nous avons besoin d'une des sorties de l'optimisation aspatiale : le vecteur de la variable d'intérêt $(q_1, ..., q_L)$. Ce vecteur permet d'attribuer un label à chaque point, ce qui nous permet de construire une carte labellisée.

A partir de cette carte, on va pouvoir créer des zones. Une zone est un ensemble de points qui sont voisins et qui partagent le même label. Pour cela, on peut utiliser un algorithme à composante connectée : ces algorithmes permettent de parcourir des graphes et d'en lister les composantes connexes, qui sont les sous-ensembles de noeuds qui sont connectés les uns aux autres. Dans notre cas, les composantes connexes sont les zones. Ces algorithmes sont de complexité $O(n)$, où n est le nombre de points à parcourir. Plusieurs librairies, dont SciPy, mettent à disposition ce genre d'algorithmes. Mais il a été codé à la main avec la complexité $O(n)$ dans notre cas, car on n'a besoin de le faire tourner qu'une seule fois. Ainsi, le temps d'exécution gagné serait minime et ne justifierait pas le temps de travail passé à réajuster les données en entrée pour les rendre compatibles avec les librairies

En ce qui concerne le voisinage, deux manières de l'aborder ont été envisagées. Soit seulement deux pavés (centrés sur les points de mesure) partageant une face sont considéré comme voisins (4-voisins), soit deux pavés partageant une face ou une arête sont considérés comme voisins (8-voisins).

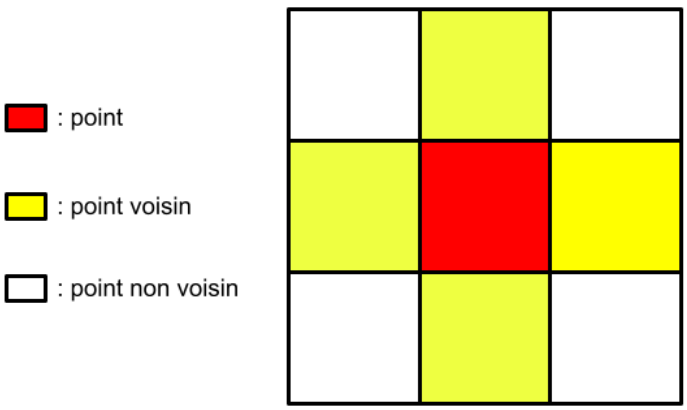
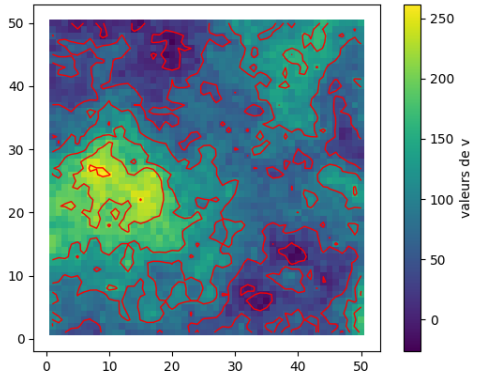
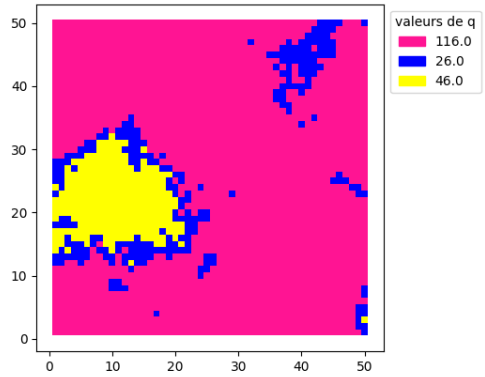


FIGURE 8 – schéma du système 4 voisins

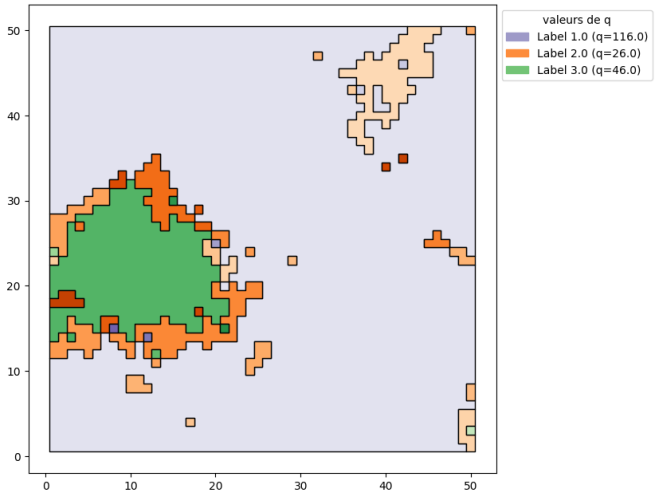
Finalement, c'est le système à 4 voisins qui a été retenu, car le système à 8 voisins a tendance à créer des zones filiformes. De plus, les voisins en diagonale sont plus éloignés d'un point que ses voisins directs, ce qui entraîne des voisins de deux types qui devraient sans doute être différentiés.



(a) carte des valeurs de v



(b) carte des labels pour 3 niveaux



(c) carte des zones

FIGURE 9 – exemple de cartes de zonage obtenues

Les mêmes données sont représentées sur les trois cartes de la figure 9. On peut voir une correspondance directe entre la carte des valeurs de *v* et la carte des labels. Si on regarde la carte 9c, on peut voir qu'on est encore loin d'un zonage efficace qui répondrait aux contraintes opérationnelles de travaux agricole. En effet, le zonage actuel ne respecte aucune contrainte spatiale. On se retrouve donc avec des zones filiformes ou trop petites, parfois même constituées d'un seul point, ce qui n'est en général pas applicable pour la réalisation de travaux agricoles. En clair, le zonage est

trop précis pour son utilité, ce qui le rend confus et impratique. Pour corriger ces problèmes, on va devoir réaliser un traitement sur les zones.

2.4.2 Définition des zones invalides

Afin de corriger le zonage, on doit être capable de détecter les zones indésirables. Une zone indésirable est une zone qui ne respecte pas des contraintes spatiales définies. Ces contraintes dépendent du contexte applicatif, et peuvent donc varier selon la nature des travaux agricoles. Pour détecter ces zones indésirables, il est nécessaire de mettre en place des contraintes de validité de la zone.

L'imposition des contraintes doit permettre de satisfaire au mieux un certain nombre de propriétés :

— Ne pas avoir de trop petites zones
— Eviter les zones comportant une ou des parties filiformes et étroites.
— Disposer de zones permettant d'être gérées par une empreinte

Concernant plus spécifiquement la prise en compte d'une empreinte, l'idée est d'invalider les zones si elles ne correspondent pas à l'empreinte de l'engin agricole utilisé. Cette contrainte est donc spécifique de l'engin utilisé pour un type de travaux agricoles. Par exemple, en irrigation, on pourrait imaginer que les zones doivent suivre un traitement unidirectionnel, qui correspond à l'empreinte d'une rampe d'irrigation. Le critère devrait encore être différent si on utilisait des pivots d'irrigation, qui fonctionnent de façon circulaire. Cette méthode pourrait également être utilisée pour la création de zones, comme dans l'article [9], où on applique un filtre de l'empreinte sur la carte pour redéfinir les zones. Étant donné que l'objectif de cette étude était de développer une méthode générique qui puisse être utilisable avec n'importe quel modèle de performance d'itinéraire technique, la prise en compte d'une empreinte a été écartée puisqu'elle est trop spécialisée.

Deux différentes contraintes ont été envisagées pour satisfaire les deux premières propriétés :

— Une contrainte de **Taille** : une zone vérifie cette contrainte si le nombre de points qui constituent la zone est strictement supérieur à un entier n. C'est la contrainte qui a été utilisée pour *geozoning*. Il a l'avantage d'être simple à exécuter et facilement compréhensible.
— Une contrainte portant sur la **Morphologie** : on applique une érosion sur la zone [voir annexe] et on considère $A_{erosion}$ l'aire de la zone après érosion. Une zone vérifie cette contrainte si les aires avant et après érosion vérifient l'inégalité :

$$A_{erosion} \geq \frac{A_{zone}}{k}$$

où A_{zone} est l'aire de la zone avant érosion, et k est un coefficient positif avec $2 \leq k < 4$. Plus k est faible, plus la contrainte **Morphologie** est discriminante. L'idée est de se débarrasser des zones filiformes, qui sont plus sujettes à l'érosion que des zones en forme de bloc. L'érosion est réalisée avec le package Shapely.

La contrainte **Taille** exclue efficacement les petites zones mais limite peu ou pas du tout la présence de zones filiformes. Par contre, la contrainte **Morphologie** permet d'une part d'éliminer efficacement les petites zones et d'autre part limite la présence de zones en partie filiforme, c'est

pourquoi c'est cette dernière contrainte qui a été choisie pour le code. Il est également possible d'utiliser une combinaison de plusieurs de ces contraintes, mais ces options n'ont pas été explorées dans ce stage. Aussi, augmenter le nombre de contraintes augmente également le temps de traitement des zones, ce qui peut avoir des conséquences sur le temps de calcul de l'algorithme. En reprenant les cartes de la figure 9b, on détecte ces zones invalides quand la contrainte **Morphologie** n'est pas satisfaite.

2.4.3 Traitements des zones invalides

Le traitement appliqué aux zones invalides peut se faire de deux manières :

— Suppression de la zone : les points de la zone sont alors assignés à la zone voisine ayant le plus de points de contact avec la zone d'intérêt. Pour ces points, seuls les attributs "zone" et "label" changent.

— Agrandissement de la zone : on réalise une boucle dans laquelle on ajoute les points voisins de la zone dont la valeur v est la plus proche des valeurs v des points de la zone. En quelque sorte, on agrandit la zone en suivant les quantiles. Cela permet une meilleure homogénéité des données de la zone dilatée, au prix de rendre sa forme plus complexe, comme on le voit sur la figure 10.

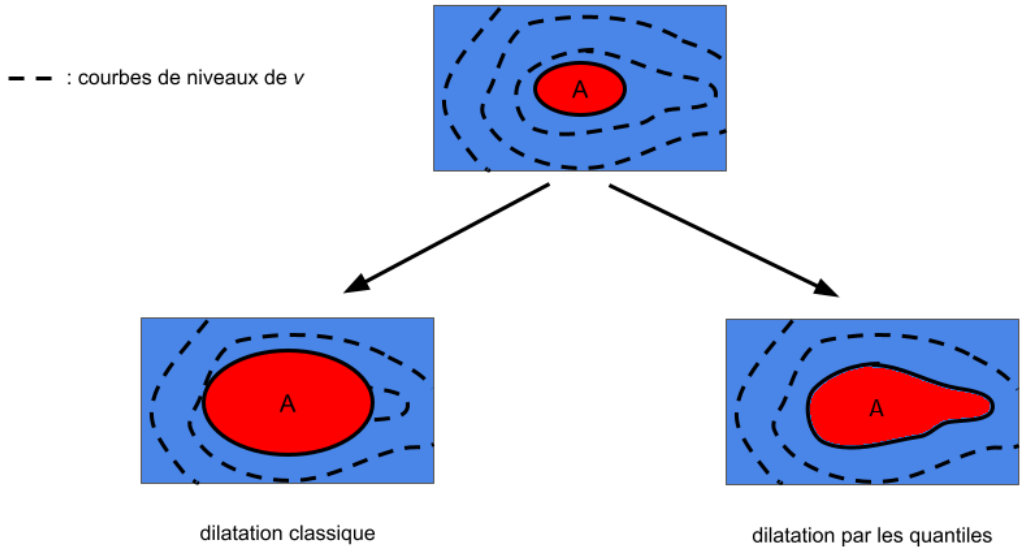


FIGURE 10 – comparaison entre une dilatation classique et une dilatation en suivant les quantiles

A chaque fois qu'on ajoute des points, on vérifie la contrainte de validité de la zone (ici la contrainte **Morphologie**). Si la zone est validée par la contrainte, alors on peut arrêter le processus d'agrandissement.
On aurait pu utiliser une approche morphologique, où on se contenterait de dilater les zones pour les agrandir, mais ce serait oublier le deuxième objectif du zonage, qui est de maximiser la biomasse totale prévue par le modèle.

Ce processus d'agrandissement est aussi une des raisons pour lesquelles la contrainte **Taille** n'a pas été utilisé, car en suivant le gradient, on a tendance à plus facilement créer des structures filiformes. c'est pour cela qu'une contrainte prenant en compte la morphologie de la zone est nécessaire afin d'éviter ces cas gênants.

Pour chaque zone, on va donc appliquer les deux traitements, du fait du changement d'affection des points dans les labels, donc de la fonction $\mathscr{L}$ on doit relancer l'algorithme d'optimisation de $J_1(\mathscr{L})$. Pour faire le choix entre la suppression ou l'agrandissement de la zone invalide, on va donc comparer pour chaque traitement les valeurs du critère S_1 et choisir le traitement qui donne la plus grande valeur de S_1.

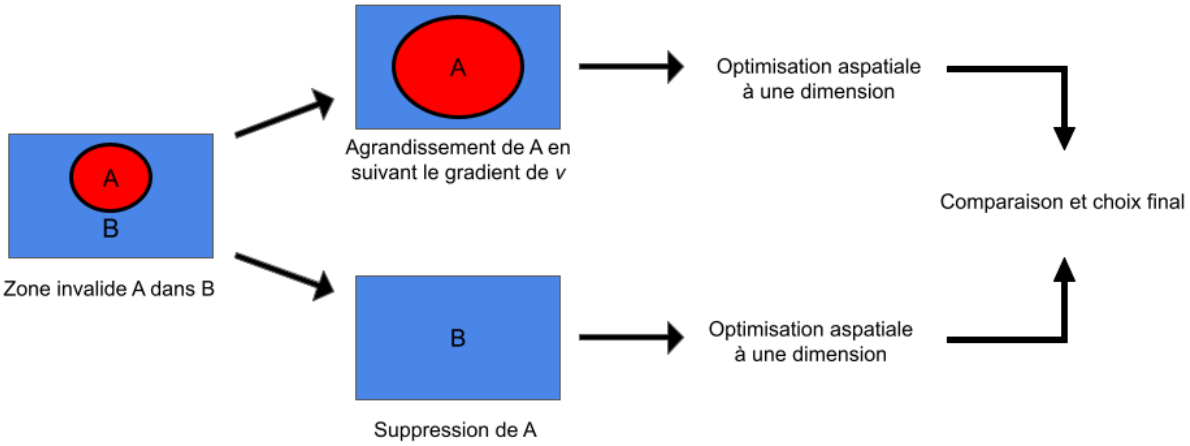


FIGURE 11 – procédure générique de traitement d'une zone invalide

Cas particuliers :

- Traitement de 2 zones très proches et de même label

Lors du processus d'agrandissement, il est possible que la zone traitée rentre en contact avec d'autres zones de même label. Par définition, deux zones voisines ne peuvent pas avoir le même label, c'est pourquoi un processus de fusion peut être déclenché. Ce processus de fusion transfère l'ensemble des points d'une zone dans l'autre, puis supprime la zone nouvellement vide. On se retrouve donc avec une nouvelle zone composée de l'ensemble des points des deux anciennes zone. En appelant A la zone traitée et B la zone rencontrée, il y a deux possibilités :

— La zone B est invalide (figure 12) : on peut alors soit supprimer une zone et agrandir l'autre, soit supprimer les deux zones, soit fusionner les deux zones en une nouvelle zone

C. Éventuellement, on continue le processus d'agrandissement si la zone C ne respecte toujours pas la contrainte **Morphologie**.

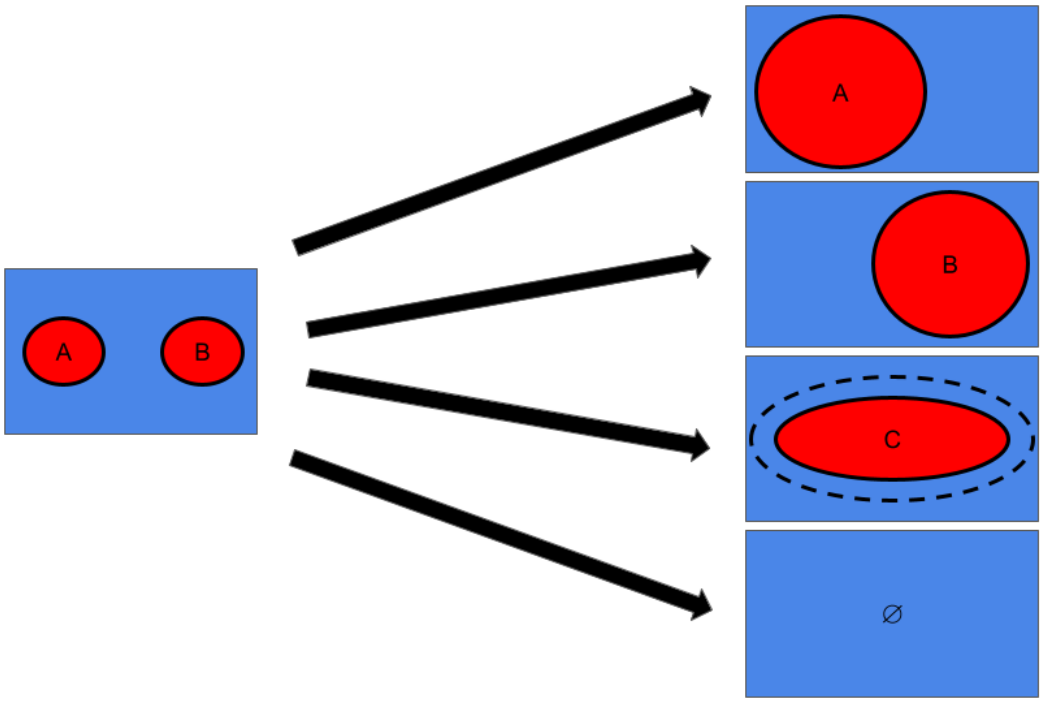


FIGURE 12 – possibilités de traitement dans le cas d'une fusion de zone où la zone rencontrée est invalide

— La zone B est valide (figure 13) : on peut soit fusionner la zone A avec, créant ainsi une nouvelle zone C, soit la supprimer. Cependant, fusionner les deux zones directement pourrait amener la zone C a être valide, peu importe la forme de la zone traitée. Cela peut amener à créer des structures ou une grande zone présente des excroissances filiformes. Afin d'éviter cela, la zone B et la zone A fusionnent seulement quand la contrainte **Morphologie** est respectée pour la zone A. L'agrandissement de la zone A continue en ignorant la zone B, et c'est seulement à la fin de l'agrandissement qu'on fusionne les deux, créant une nouvelle zone C.

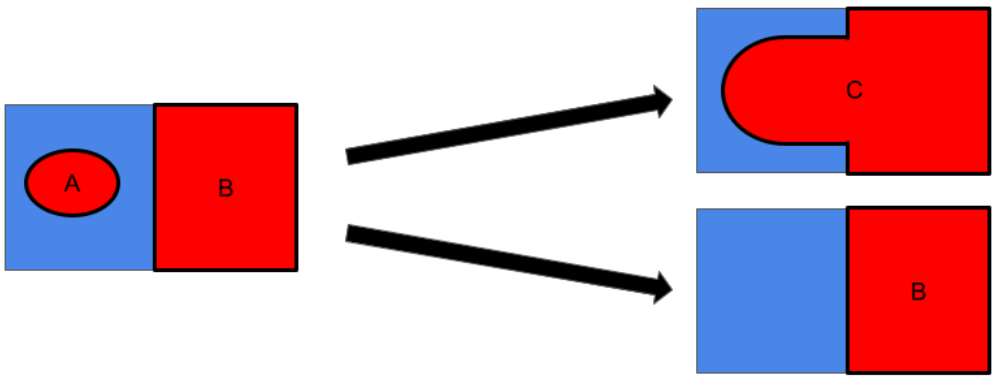


FIGURE 13 – possibilités de traitement dans le cas d'une fusion de zone où la zone rencontrée est valide

- **Fusion entraînant de nouvelles petites zones**

Il est également possible qu'en agrandissant une zone, on se retrouve à créer de nouvelles zones trop petites. Dans la figure 14, on voit que l'agrandissement et la fusion des zones cause la création d'un grand nombre de zones de label 2 (vert).

Pour traiter ce cas, deux solutions ont possibles :

— fusionner les nouvelles zones invalides

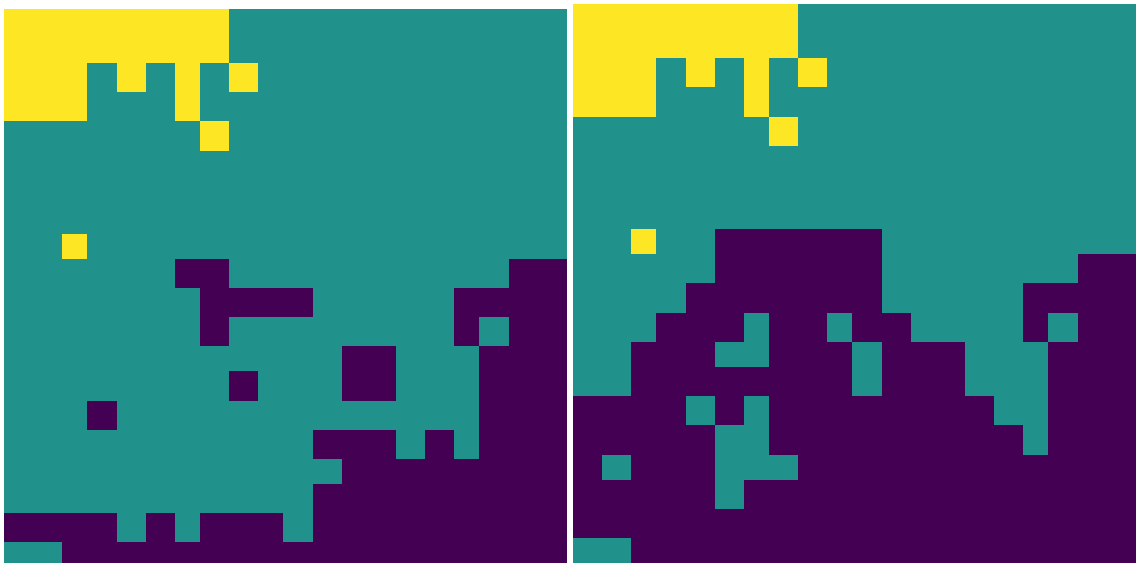
— supprimer la zone parce que l'agrandissement de la zone ne peut pas se faire dans de bonnes conditions.

Étant donné que la première solution risque de créer de nouvelles petites zones pour au final avoir un risque de bouclage de l'algorithme, il a été choisi de supprimer la zone au lieu de l'agrandir. La meilleure solution serait de tester les deux traitements, mais cela n'a pu être fait par manque de temps.

2.4.4 Résumé et choix de la méthode

Dans l'ensemble, les méthodes utilisées vont être d'une structure similaire. Néanmoins, ici, deux types d'organisation sont proposées : une méthode dite séquentielle et une méthode dite parallèle.

Comme le montrent les figures 15 et 16, les deux méthodes commencent de la même manière. On se sert de l'optimisation aspatiale afin de créer une carte labellisée, et à partir de cette carte



(a) carte avant agrandissement (b) carte après agrandissement

FIGURE 14 – exemple d'agrandissement provoquant la création de nouvelles zones invalides

labellisée, on peut utiliser un algorithme à composantes connectées, qui va parcourir chaque point de la carte et l'assigner à une zone, selon les règles de voisinnage, afin de créer les zones de la carte. On choisit ensuite l'ordre dans lequel les zones seront traitées. Ici, on va procéder par taille, afin que les zones plus petites soient traitées en premier. En effet on préfère nettoyer la carte le plus possible avant de modifier les plus grandes zones, dont on suppose que le traitement modifie le plus la carte. À partir de là, les deux méthodes divergent :

— Dans la méthode séquentielle, présentée par la figure 16, lorsqu'il s'agit de choisir entre l'agrandissement et la suppression d'une zone invalide, le traitement sélectionné est appliqué immédiatement avant de passer à la zone invalide suivante. Ainsi, la carte de zonage évolue progressivement au fur et à mesure des décisions prises. Bien que cette méthode soit censée produire de meilleurs résultats, elle est également plus exigeante en termes de ressources, ce qui pourrait poser des problèmes pour les grands ensembles de données.

— Dans la méthode parallèle, présentée par la figure 15, les phases de décision et de traitement sont séparées. C'est à dire qu'on fait le choix entre agrandissement et suppression indépendemment pour chaque zone invalide. On se base donc uniquement sur le zonage initial pour prendre les décisions. On applique ensuite les traitements correspondant aux décisions pour chaque zone invalide.

D'un point de vue algorithmique, cela donne deux structures différentes comme on peut le voir sur les algorithmes 1 et 2. La principale différence est que contrairement à l'algorithme séquentiel, l'algorithme parallèle réalise deux boucle sur les zones invalides : une première fois pour prendre les décisions, une deuxième fois pour les traiter. L'algorithme parallèle nécessite également de faire tourner une optimisation à la fin de l'algorithme, afin de calculer les valeurs finales de quotas d'irrigation.

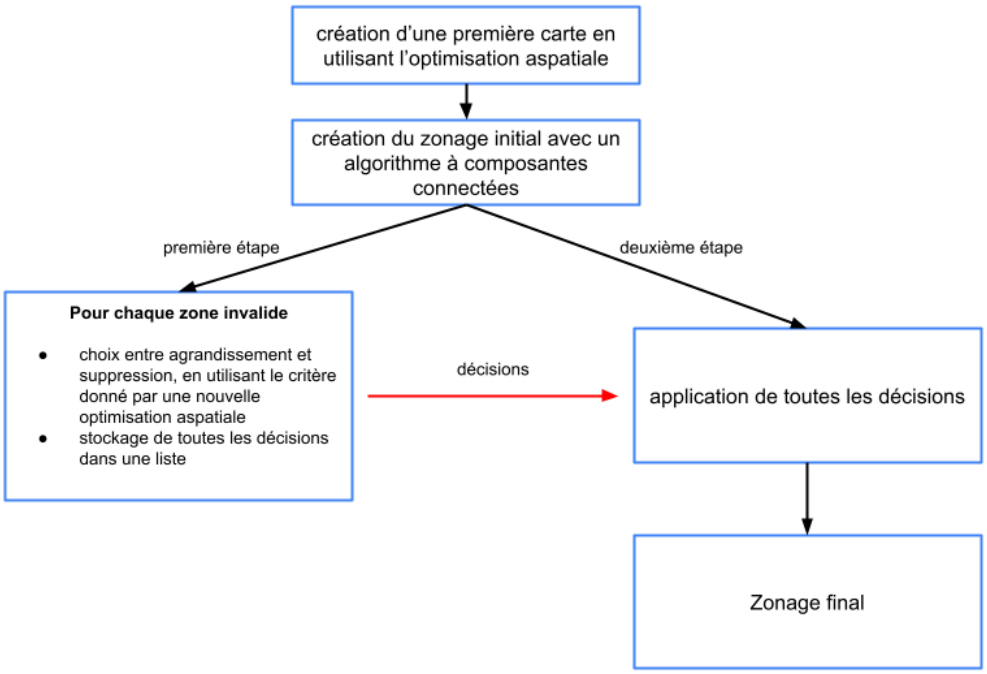


FIGURE 15 – résumé de la méthode parallèle

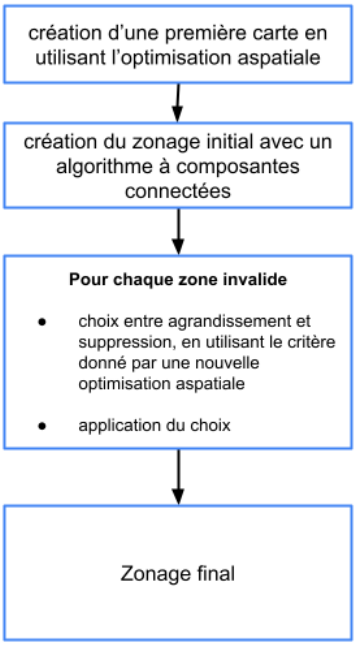


FIGURE 16 – résumé de la méthode séquentielle

Input : Q, L, k
Output: $\mathscr{L}(i,j)_{i,j}$, $\mathbf{p}=(q_l)_l$, $\mathbf{q}=(q_l)_l$
Data : $(v_{ij)_{i,j}}$, $B(v_{ij},q)_{i,j}$

denote $\mathscr{P}(L)$ as the set of available quantile sets.
foreach $\mathbf{p}_n \in \mathscr{P}(L)$ **do**
    evaluate $\mathscr{L}_n(.)$ associated to $\mathbf{p}_n$
    evaluate $\mathbf{q}_n$ which maximizes $S(Q,\mathscr{L}_n(.),\mathbf{q})$
end
select n^* which maximizes $S(Q,\ \mathscr{L}_n(.),\mathbf{q}_n)$
store $\mathscr{L}_{n^*}(.)$, $\mathbf{p}_{n^*}$ and $\mathbf{q}_{n^*}$

define the associated zones using the connected components algorithm;

foreach zone $\in$ zones **do**
    if zone does not respect the morphological_constraint(k) **then**
        add zone to dictionary_invalid_zones
    end
end

sort dictionary invalid_zones by increasing zone size

$\mathscr{L}(.)=\mathscr{L}_{n^*}(.)$
foreach invalid_zone $\in$ dictionary_invalid_zones **do**
    evaluate $\mathscr{L}_s(.)$ and $S_{1,\text{suppress}}=\max_{\mathbf{q}} S(Q,\mathscr{L}_s(.),\mathbf{q})$ for suppression of invalid_zone
    evaluate $\mathscr{L}_e(.)$ if enlargement of invalid_zone
    if enlargement of invalid_zone creates new invalid zones **then**
        suppress invalid_zone and update $\mathscr{L}(.)=\mathscr{L}_s(.)$
    else
        evaluate $S_{1,\text{enlarge}}=\max_{\mathbf{q}} S(Q,\mathscr{L}_e(.),\mathbf{q})$ for enlargement of invalid_zone
        if $S_{1,\text{enlarge}} \leq S_{1,\text{suppress}}$ **then**
            suppress invalid_zone :
            update $\mathscr{L}(.)=\mathscr{L}_s(.)$ and $\mathbf{q}^*$ which maximizes $S(Q,\mathscr{L}_s(.),\mathbf{q})$
        else
            enlarge invalid_zone :
            update $\mathscr{L}(.)=\mathscr{L}_e(.)$ and $\mathbf{q}^*$ which maximizes $S(Q,\mathscr{L}_e(.),\mathbf{q})$
        end
    end
end

construct and save final map

Algorithm 1: Algorithme global de zonage séquentiel

Input : Q, L, k
Output: $\mathscr{L}(i,j)_{i,j}$, $\mathbf{p}=(q_l)_l$, $\mathbf{q}=(q_l)_l$
Data : $(v_{ij})_{i,j}$, $B(v_{ij},q)_{i,j}$

denote $\mathscr{P}(L)$ as the set of available quantile sets.
foreach $\mathbf{p}_n \in \mathscr{P}(L)$ **do**
  evaluate $\mathscr{L}_n(.)$ associated to $\mathbf{p}_n$
  evaluate $\mathbf{q}_n$ which maximizes $S(Q,\mathscr{L}_n(.),\mathbf{q})$
end
select n^* which maximizes $S(Q,\ \mathscr{L}_n(.),\mathbf{q}_n)$
store $\mathscr{L}_{n^*}(.)$, $\mathbf{p}_{n^*}$ and $\mathbf{q}_{n^*}$

define the associated zones using the connected components algorithm

foreach zone $\in$ zones **do**
  if zone does not respect the morphological_constraint(k) **then**
    add zone to dictionary_invalid_zones;
  end
end

sort dictionary invalid_zones by increasing zone size

$\mathscr{L}(.) = \mathscr{L}_{n^*}(.)$
foreach invalid_zone $\in$ dictionary_invalid_zones **do**
  evaluate $\mathscr{L}_s(.)$ and $S_{1,\text{suppress}} = \max_{\mathbf{q}} S(Q,\mathscr{L}_s(.),\mathbf{q})$ for suppression of invalid_zone
  evaluate $\mathscr{L}_e(.)$ for enlargement of invalid_zone
  if enlargement of invalid_zone creates new invalid zones **then**
    $\mathscr{D}$(invalid_zone) = S
  else
    evaluate $S_{1,\text{enlarge}} = \max_{\mathbf{q}} S(Q,\mathscr{L}_e(.),\mathbf{q})$ for enlargement for invalid_zone
    if $S_{1,\text{enlarge}} \leq S_{1,\text{suppress}}$ **then**
      $\mathscr{D}$(invalid_zone) = S
    else
      $\mathscr{D}$(invalid_zone) = E
    end
  end
end
foreach invalid_zone $\in$ dictionary_invalid_zones **do**
  if $\mathscr{D}$(invalid_zone) = S **then**
    suppress invalid_zone
  else
    enlarge invalid_zone
  end
  update $\mathscr{L}(.)$
end

evaluate $\mathbf{q}^*$ which maximizes $S(Q,\mathscr{L}(.),\mathbf{q})$
construct and save final map

Algorithm 2: Algorithme global de zonage parrallèle

2.4.5 Choix techniques

Python étant un langage permettant la programmation objet, la question a été posée d'utiliser des objets ou non. Les avantages de la programmation objet sont multiples : on gagne en lisibilité sur le code, on peut attribuer plus facilement les traitements que l'on fait sur les objets dans des méthodes... Les désavantages pouvaient être la complexité à gérer ces objets. Par exemple, pour supprimer un objet, il faut supprimer toutes les références à cet objet dans la mémoire, ce qui peut rendre cette tâche fastidieuse ou créer des beugues difficiles à identifier. Dans notre algorithme, les zones sont parfois amenées à être supprimées.

Les deux choix ont été testés (avec et sans programmation objet), et il a finalement été décidé d'utiliser des objets dans le code, avec deux classes d'objets :

— la classe **Point** : Un point est associé à une donnée de v sur l'emplacement de coordonnées (i, j).

— la classe **Zone** : Une zone correspond à un ensemble de points voisins de même label.

Classe	**Attribut**	**Description**
Point	`index`	Identifiant unique du point
	`i`	Indice de l'abscisse du point
	`j`	Indice de l'ordonnée du point
	`label`	Label du point
	`value`	Valeur de v associée au point
	`zone`	Référence à la zone à laquelle le point appartient
	`geometry`	Représentation géométrique du point
Zone	`id`	Identifiant unique de la zone
	`label`	Label de la zone
	`points`	Liste des points appartenant à la zone
	`geometry`	Représentation géométrique de la zone
	`voisins`	Liste des zones voisines

TABLE 1 – attributs des classes `Point` et `Zone`

Ces objets permettent donc de faire rapidement un lien entre les points et les zones. En effet, on peut voir sur la table 1 que les points font référence à la zone auquelle ils appartiennent avec l'attribut "zone", et les zones font référence aux points qui les constitue avec l'attribut "points". En ce qui concerne les attributs "geometry", ils sont crées grâce à la librairie Shapely. Cela permet de réaliser des traitements morphologiques sur les zones, notamment l'érosion de zones, et simplifie l'affichage graphique des points. La géométrie d'un point est un carré de côté un, avec pour centre

les coordonnées du point. La géométrie d'une zone est la concaténation de la géométrie des points qui la constituent.

On peut remarquer que les attributs de la classe zone ne traitent pas de la validité de la zone. C'est parce qu'il a été préféré de le faire à part, en construisant un dictionnaire qui recense les zones invalides. On a fait ce choix car on a très souvent besoin d'avoir accès à ces zones dans le code, et revérifier les attributs de chaque objet Zone dès qu'on veut faire le moindre traitement serait fastidieux.

A Retenir...

- L'optimisation aspatiale permet d'obtenir une carte labellisée en regroupant les points voisins par label en zones
- Une contrainte de morphologie permet de détecter des zones invalides qu'il faut traiter
- Chaque zone invalide est soit agrandie, soit supprimée, selon les résultats donnés après réutilisation de l'optimisation aspatiale.
- La gestion de certains cas particuliers est nécessaire pour garantir la cohérence du zonage

3 Résultats

Conformément à l'approche adoptée pour ce stage, les résultats seront présentés en commençant par l'optimisation aspatiale. Ensuite, nous aborderons l'approche spatiale, avant de nous pencher sur les résultats agronomiques.

L'ensemble des résultats a été généré à partir de deux cartes générées par géostatistiques : 'map1.csv' et 'map2.csv'. Les paramètres qui ont amené à la création des deux cartes sont les mêmes pour les deux fichiers, mais leur structure globale est très différente (détails en annexe). Pour l'optimisation aspatiale, seuls les résultats pour le fichier 'map2.csv' sont présentés.

Aussi, pour pouvoir présenter les résultats, un certain nombre de paramètres ont du être fixés.

— **La contrainte totale d'irrigation Q a été fixée a 100**. Cette valeur a été choisie car c'est autour d'une valeur de q de 100 que l'on observe la plus grande variabilité de $B(v,q)$ selon v. Ainsi, cela permet plus facilement de rendre compte d'une variabilité spatiale intraparcellaire.

— **Le nombre de labels a été fixé à 3**. Cela permet une meilleur lisibilité de la carte dans l'approche spatiale. Des tests ont également été réalisés avec un plus grand nombre de labels.

Enfin, les résultats de biomasse correspondent à une biomasse relative. En effet, la biomasse totale récoltée B est divisée par B_{opt}, qui correspond à la biomasse B calculée par le modèle dans le cas où on a aucune contrainte Q. En quelque sorte, B_{opt} correspond à la biomasse potentielle de la parcelle, et ne dépend que des valeurs de v.

3.1 Résultats de l'optimisation aspatiale

3.1.1 Comparaison des différentes méthodes d'optimisation

Méthode	**Quantiles** $(p_l)_l$	**Quotas locaux** $(q_l)_l$	**Biomasse** B/B_{opt}	**Temps (s)**
Conditions de Karush-Kuhn-Tucker	[0.8, 0.9]	[116, 26, 46]	0.5597	24
Élimination de contrainte (Powell)	[0.2, 0.7]	[107, 105, 86]	0.5593	68
Élimination de contrainte (Nelder-Mead)	[0.5, 0.9]	[106, 95, 87]	0.5589	46
Avec contrainte (trust-constr)	[0.3, 0.9]	[95, 106, 76]	0.5588	11
Avec contrainte (SLSQP)	[0.7, 0.8]	[115, 62, 66]	0.5582	4.3
Avec contrainte (COBYLA)	[0.7, 0.8]	[116, 55, 65]	0.5582	34
Élimination de contrainte (BFGS)	[0.5, 0.8]	[105, 95, 92]	0.5545	2.1
Élimination de contrainte (CG)	[0.2, 0.5]	[99, 106, 96]	0.5531	2.1

TABLE 2 – résultats des différents algorithmes d'optimisation avec biomasse normalisée

Le tableau 2 représente les résultats de chaque optimisation pour le fichier map2, ordonnés par la biomasse relative obtenue. On voit que l'optimisation avec les conditions de Karush-Kuhn-Tucker donne les meilleurs résultats de B, suivi par l'optimisation avec élimination de contrainte,

avec les méthodes Powell et Nelder-Mead. De plus, l'optimisation avec les conditions de Karush-Kuhn-Tucker est assez rapide. Puisqu'elle présente le meilleur rapport performance/temps, c'est cette méthode d'optimisation qui va être utilisée.

On remarque aussi que le vecteur de quantiles et le vecteur de quotas d'irrigation (q_1, q_2, q_3) proposé varie beaucoup d'une méthode à l'autre. D'un point de vue agronomique, cela signifie qu'il y a des manières d'irriguer très différentes qui permettent d'obtenir un rendement satisfaisant.

3.1.2 Détails de l'optimisation avec les conditions de Karush-Kuhn-Tucker

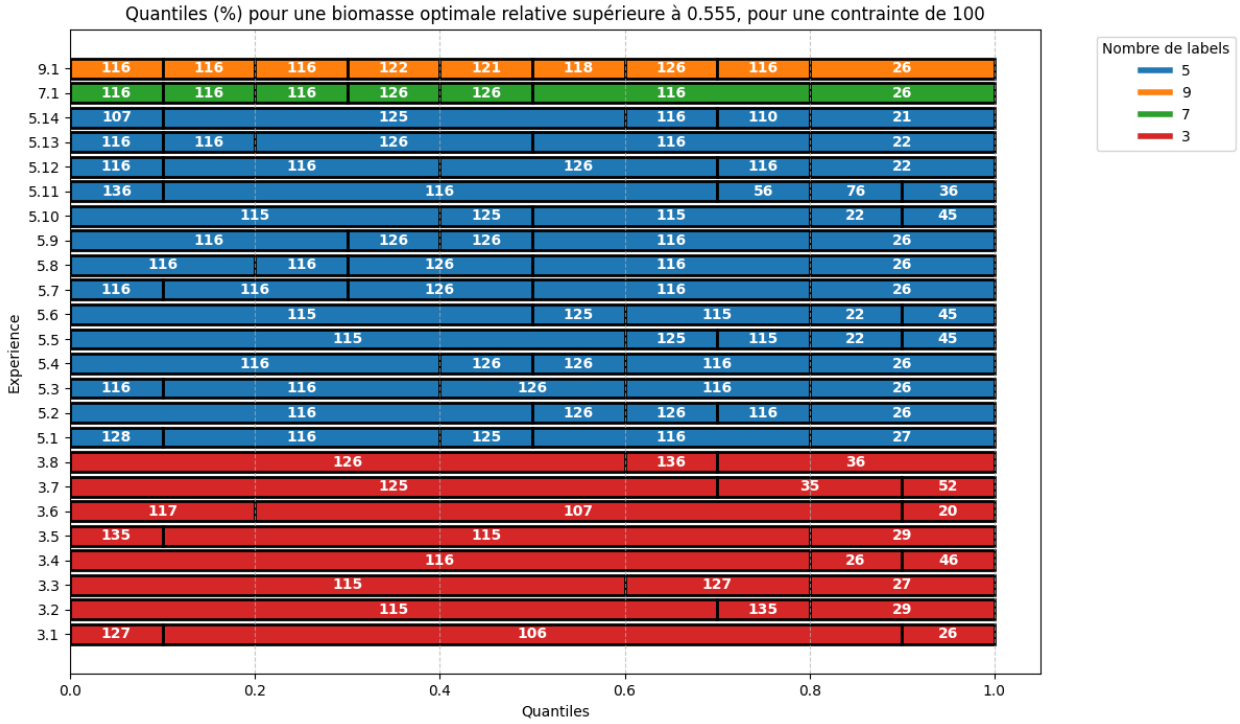


FIGURE 17 – résultats de l'optimisation avec les conditions de Karush-Kuhn-Tucker

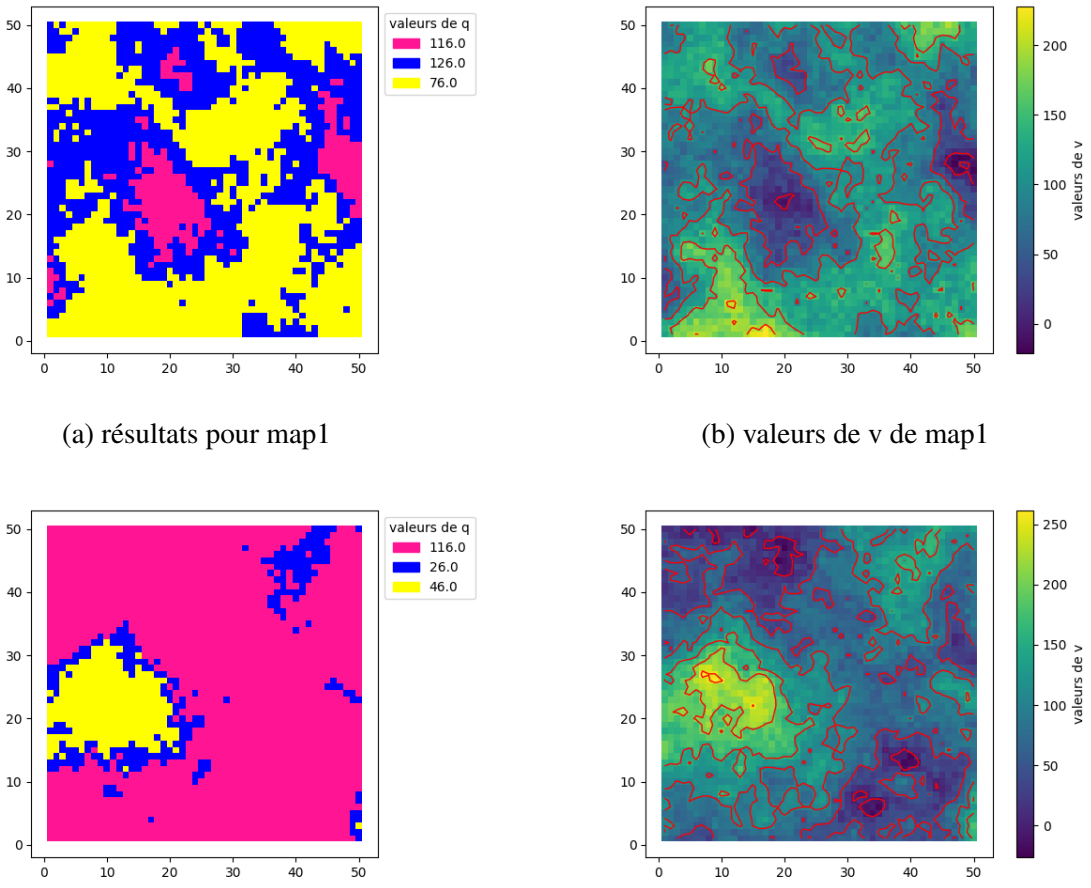
On peut voir sur la figure 17 les vecteurs de quantiles et les vecteurs de quotas d'irrigation $(q_1, ..., q_L)$ qui donnent les meilleurs résultats, après avoir réalisé une optimisation avec les conditions de Karush-Kuhn-Tucker. Les résultats sont présentés pour un nombre de labels impairs, les résultats pour un nombre de label pair ont été écartés. Ici aussi, on peut voir une grande variation dans le vecteur de quantiles et le vecteur de quotas d'irrigation $(q_1, ..., q_L)$ proposé. Néanmoins, on peut voir une tendance se dégager. En effet, l'algorithme d'optimisation à tendance à donner un quota d'irrigation plus faible pour les plus grandes valeurs de v. Cela peut s'expliquer par le fait la valeur de $B(v, q)$ selon q augmente très peu pour des fortes valeurs de v. Ainsi, négliger les points de valeur de v élevée à peu d'impact sur leur rendement, mais permet d'économiser l'eau pour augmenter les quotas d'irrigation ailleurs.

On peut remarquer que le nombre de vecteurs de quantiles qui donnent un bon résultat de biomasse B baisse quand on augmente le nombre de labels. En effet, quand on augmente le nombre

de labels, on augmente le nombre de dimension du problème, qui devient plus complexe. Dès lors, les méthodes d'optimisation utilisées ici sont moins efficaces. Si on voulait travailler avec un grand nombre de labels, il faudrait reconsidérer le choix de la méthode d'optimisation. Aussi, d'un point de vue agronomique, cela peut être difficile d'irriguer de manière différente pour un grand nombre de labels. Comme on préfère travailler avec un petit nombre de labels pour la partie spatiale, cela ne présentera pas de problème.

Parmi tous ces résultats, pour un nombre de labels donné, on ne va prendre que le vecteur de quantiles et le vecteur de quotas d'irrigation $(q_1, ..., q_L)$ qui donne la meilleure biomasse B. Ce résultats va nous permettre d'obtenir une carte labellisée, qui va être traitée dans l'optimisation spatiale.

3.1.3 Zonage initiaux résultant de l'optimisation aspatiale



(a) résultats pour map1

(b) valeurs de v de map1

(c) résultats pour map2

(d) valeurs de v de map2

FIGURE 18 – cartes obtenues après l'optimisation aspatiale

La figure 18 correspond aux cartes obtenus grâce à l'optimisation aspatiale avec les conditions de Karush-Kuhn-Tucker pour les fichiers map1 et map2. Quand on compare les cartes labellisées

à gauche, avec les cartes des valeurs de v à droite, on peut remarquer que les zones créées suivent le gradient de v.

Carte	**Quantiles**	**Biomasse**	**Temps (s)**
map1	[0.1, 0.5]	0.5395	25
map2	[0.8, 0.9]	0.5597	34

TABLE 3 – résultats de l'optimisation aspatiale pour les cartes map1 et map2

On remarque bien une correspondance entre les vecteurs de quantiles présenté dans le tableau 3 et les cartes obtenues. Notamment pour map2 (figure 18c), où on voit bien les données associées au label 1 représentent 80% des données.

On rappelle que pour construire cette carte, on ne se soucie pas de la spatialisation des données. Elle correspond donc à un plan d'irrigation idéal de la parcelle. Comme l'étape de zonage va rajouter une contrainte au problème d'optimisation, on s'attend à ce que la biomasse B obtenue soit plus faible à la fin du traitement.

A Retenir...

— Après comparaison des différentes méthodes d'optimisation, c'est l'optimisation avec les conditions de Karush-Kuhn-Tucker qui est choisie.
— Cette optimisation permet d'obtenir une ou plusieurs cartes labellisées, qui vont ensuite être traitées.

3.2 Résultats du zonage

3.2.1 Étapes du zonage

Le zonage se fait en traitant les zones qui ne respectent pas un critère de validité une par une, en appliquant une décision sur la zone selon les résultats d'une optimisation aspatiale. Soit la zone est supprimé, soit elle est agrandie.

En utilisant le critère de validité **Morphologie**, on obtient les zones invalides suivantes pour la carte map2 montrées sur la figure 19. Comme il y a 40 zones invalides, le traitement se fera en 40 étapes, par ordre croissant de taille des zones. Dans un contexte de travail agricole, ces zones seraient particulièrement gênantes, car cela demanderait une trop grande précision de s'adapter à la forme de chacune de ces zones pour y appliquer un traitement différent.

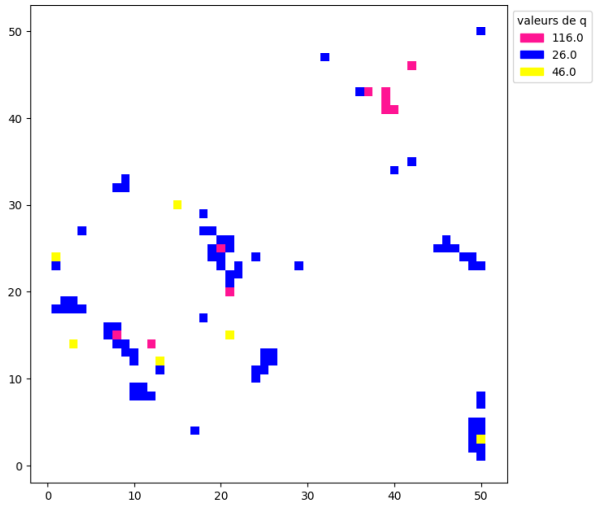


FIGURE 19 – zones invalides pour la carte map2

La figure 26 présente toutes les étapes du processus de zonage pour la carte map2 et la méthode **séquentielle**. Le grand nombre d'étapes rend le processus confus, ainsi, seules certaines étapes clés ont été représentées (le reste a été mis en annexe). À l'itération 6, le point de coordonnées ($x = 3, y = 14$) est agrandit. Après son agrandissement, il est fusionné à la grande zone de même label (jaune). Un bon exemple de suppression est la zone de coordonnées (25, 12), qui est supprimée à l'itération 39. La dernière étape consiste en l'agrandissement de la zone en bas à droite.

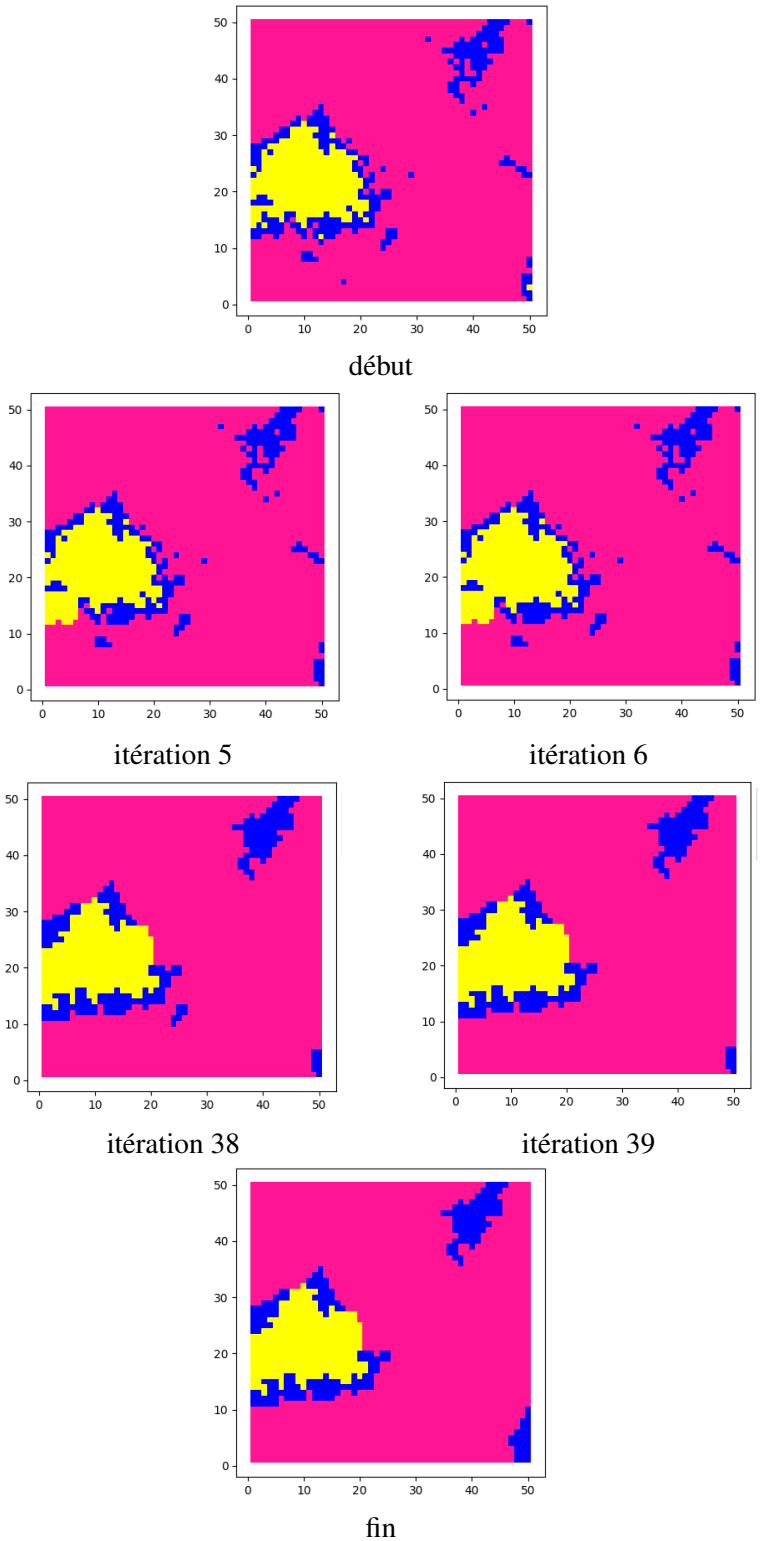
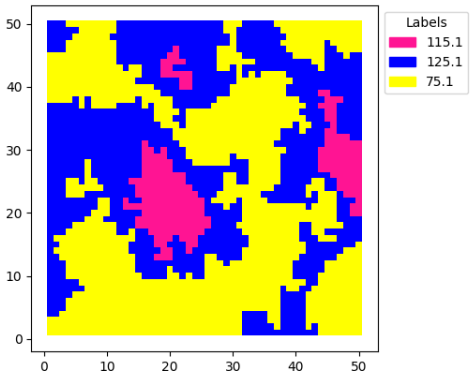


FIGURE 20 – étapes clés du processus de zonage

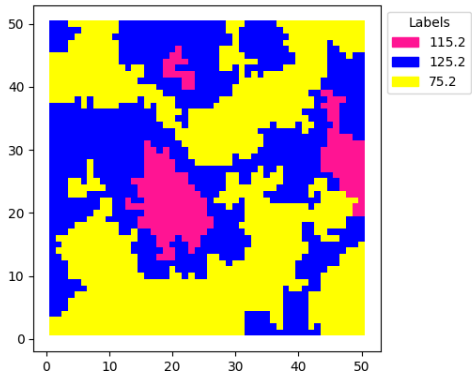
Après la 40^{eme} étape, le zonage est terminé. Les zones obtenus sur la carte vérifient la contrainte morphologique. A chaque zone est associé un traitement défini par l'optimisation.

3.2.2 Comparaison des méthodes séquentielles et parallèles

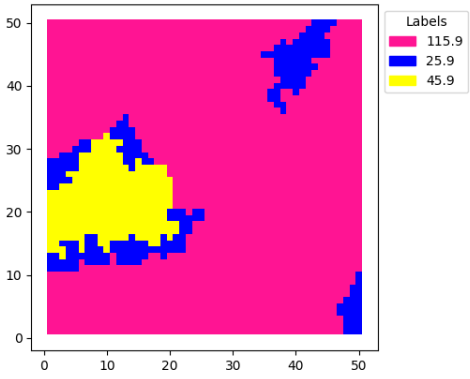
Comme expliqué précédemment, deux approches différentes ont été testées : une approche séquentielle, où on applique le traitement pour une zone juste après la prise de décision sur la zone ; et une approche parallèle, où on prend toutes les décisions de traitement sur les zones invalides dans un premier temps, puis on applique l'ensemble des décisions.



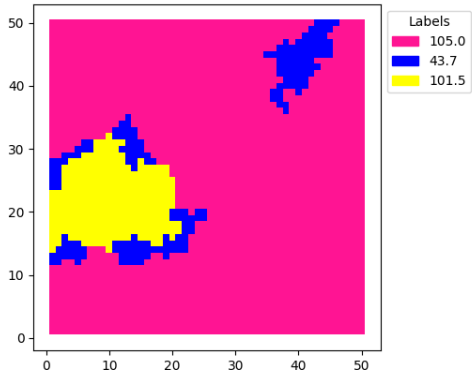
(a) méthode séquentielle pour map1



(b) méthode parallèle pour map1



(c) méthode séquentielle pour map2



(d) méthode parallèle pour map2

FIGURE 21 – comparaison entre la méthode séquentielle et la méthode parallèle

Visuellement, on peut voir des différences entre la carte finale des deux méthodes sur la figure 21. Par exemple, pour map1, la zone de label 3 (jaune) en haut à droit est isolée pour l'approche séquentielle, tandis qu'elle est a été reliée à d'autres zones de même label dans l'approche parallèle. Sur map2, la zone en bas à droite de label 2 (bleu) a été agrandie dans l'approche séquentielle, tandis qu'elle a été supprimée dans l'approche parallèle.

Carte	Méthode	Temps total (s)	Biomasse
map2	Séquentiel	614	0.5600
	Parallèle	382	0.5565
map1	Séquentiel	1443	0.5438
	Parallèle	647	0.5431

TABLE 4 – résumé des temps de traitement et de la biomasse pour les cartes map1 et map2 avec espace vertical augmenté

En ce qui concerne la performance des deux approches, on voit sur la table 4 que pour les deux cartes, l'approche séquentielle donne un meilleur résultat de biomasse relative que l'approche parallèle. Cela s'explique par le fait que dans l'approche séquentielle, des conflits entre les décisions peuvent arriver, puisqu'il sont traités indépendemment. Dans ces cas là, c'est la zone qui est traitée en premier qui prend la priorité sur la décision d'agrandissement. Avec l'approche séquentielle, on est certain de prendre la meilleure décision à chaque itération, ce qui aboutit à un meilleur résultat.

Cependant, le temps total de la méthode parralèle est environ 2 fois plus court que celle de la méthode séquentielle. La méthode parallèle pourrait donc être envisagée dans le cas où on doit traiter une grande quantité de données.

A Retenir...

- Le processus de zonage traite toutes les zones invalides pour obtenir un zonage adéquat.
- Les approches séquentielles et parallèles aboutissent à des cartes différentes
- Parmi les deux approches essayées, l'approche séquentielle est la plus performante.

3.3 Application agronomique

3.3.1 Comparaison entre l'optimisation aspatiale et une distribution uniforme

Ici, on compare les résultats données par l'optimisation aspatiale avec ceux donnés par une distribution uniforme. Pour une contrainte Q, la distribution uniforme correspond à apporter un quota d'irrigation $q == Q$. En clair, on ne pratique pas d'agriculture de précision, et on se contente d'irriguer uniformément sur la parcelle.

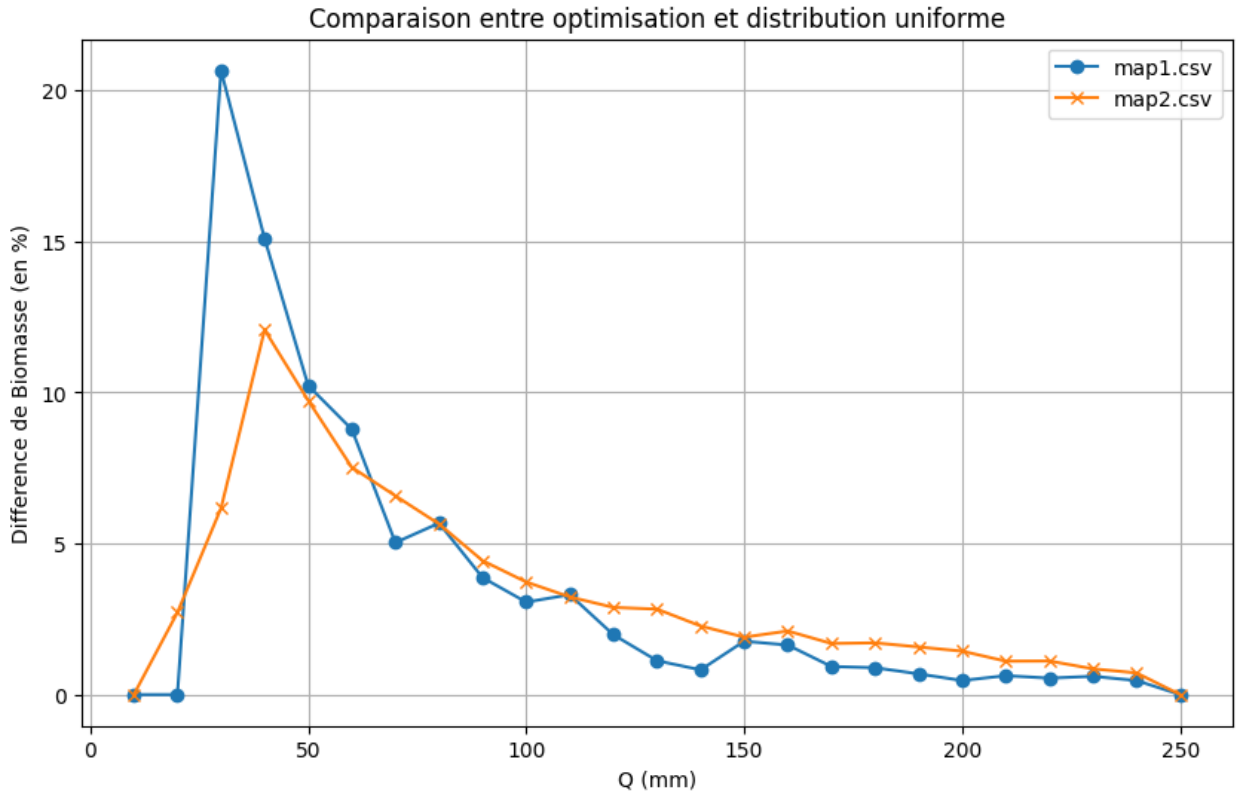


FIGURE 22 – comparaison entre optimisation aspatiale et distribution uniforme

La figure 22 représente la différence de biomasse donnée par l'optimisation aspatiale et la distribution uniforme pour différentes valeurs de contraintes Q, avec les cartes map1 et map2. On peut voir que cette différence est positive pour toutes les valeurs de Q, donc à minima dans le cas de ce modèle, il y a un gain dans le fait de pratiquer de l'agriculture de précision.

Aussi, on peut voir que cette différence diminue lorsque la contrainte Q augmente. Cela est dû au fait que comme on peut le voir sur la figure 4, c'est pour des valeurs de quota d'irrigation q faibles qu'on observe la plus grande variabilité de $B(v,q)$ en fonction de q. Ainsi, les différences de quota d'irrigation q données par l'optimisation ont plus de conséquence sur la variabilité de $B(v,q)$. On aurait donc pu utiliser une contrainte Q encore plus faible dans l'optimisation. On peut noter que pour les valeurs extrêmes, la distribution optimale et la distribution uniforme donnent la même valeur de biomasse B.

4 Discussions

L'objectif de ce stage était de s'inspirer de la méthode *geozoning*, développée par l'équipe "Systèmes Dynamiques" de l'UMR MISTEA. La méthode présentée correspond à un zonage piloté par un modèle de performance d'itinéraire technique, où on se sert des résultats fournis par le modèle pour construire un zonage initial, puis pour prendre des décisions dans le traitement de ce zonage. Cet algorithme, en plus de proposer un zonage adapté aux données de la parcelle, propose donc un traitement différencié associé au zonage proposé.

Sur le tableau 5, on peut voir les différences entre *geozoning* et le zonage piloté par un modèle, dans le contexte de l'irrigation. Comme le modèle $B(v,q)$ utilise le vecteur de quotas d'irrigation $(q_1,...,q_L)$, ce vecteur s'ajoute aux contrôles et aux sorties de l'algorithme dans le zonage piloté par un modèle. Ce vecteur joue également un rôle dans la contrainte globale, ici la quantité totale d'eau qu'on apporte à la parcelle. Cette contrainte est totalement absente pour *geozoning*. En ce qui concerne les contraintes sur les zones, un plus grand nombre d'options a été exploré que pour *geozoning*. Piloter un zonage par un modèle est novateur, et ce type de zonage n'existe pas à notre connaissance. En effet, la plupart des zonages définissent des zones en se basant uniquement sur la spatialisation des données, puis proposent un traitement adapté à chaque zone.Dans notre approche, les deux étapes sont couplées.

	Geozoning	**Zonage piloté par un modèle**
Entrées	variable v_{ij}	variable v_{ij}, sortie du modèle $B(v_{ij},q)$
Sorties	$(z_m)_m, \mathscr{L}(i,j)$	$(z_m)_m, \mathscr{L}(i,j), (q_l)_l$
Contrôles	$\mathscr{L}(i,j)(\sim$ quantiles de $v)$	$\mathscr{L}(i,j),\ (q_l)_l$
Critère	$\max\limits_{z} \min\limits_{z' \in N(z)} C(z,z')$	$\max \sum\limits_{l} \sum\limits_{\mathscr{L}((i,j))=l} B(v_{ij},q_l)$
Contraintes globales	aucune	$\sum\limits_{l} \#\{(i,j) \mid \mathscr{L}((i,j)) = l\} q_l \leq Q, q_l \geq 0$
Contraintes locales sur les zones	taille	(taille), forme, (empreinte)

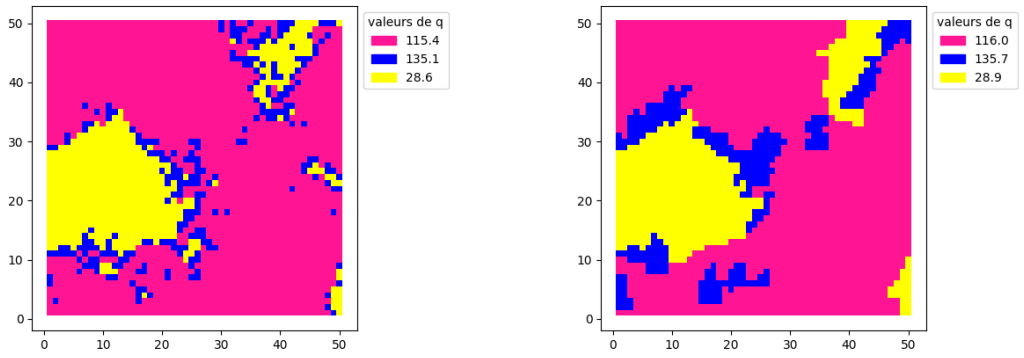
TABLE 5 – comparaison entre *geozoning* et le zonage piloté par un modèle de performance d'itinéraire technique

4.1 Voies d'amélioration

Des pistes d'améliorations ont été identifiées :

Dans l'algorithme, on ne prend en compte que le meilleur vecteur de quantiles $(q_1,...,q_L)$ afin d'obtenir un zonage initial. Or, on a vu sur la figure 17 qu'il existe différents arrangements de quantiles qui permettent d'obtenir un résultat satisfaisant à label fixé. Ces arrangements de

quantiles donnent des zonages initiaux très différents les uns des autres, comme on peut le voir sur la figure 23. Ainsi, il serait possible qu'un vecteur de quantiles légèrement moins bon dans l'optimisation aspatiale, permette d'obtenir un meilleur zonage final après traitement. Dans l'idéal, il faudrait tester un plus grand nombre de vecteur de quantiles, mais cela multiplierait le temps d'exécution de l'algorithme par le nombre de vecteurs testés. Également, il aurait été intéressant de tester d'autres librairies d'optimisations que SciPy, notamment Pyomo dont l'utilisation a été discutée dans la partie 2.3.2.



(a) zonage initial

(b) zonage final

FIGURE 23 – exemple de zonage séquentiel en se basant sur le vecteur de quantiles : $(0, 0.7, 0.8, 1)$

Le modèle utilisé ici a entraîné certains problèmes. Premièrement, le modèle présente des variations assez faibles de $B(v,q)$ pour différentes valeurs de q. C'est pour cela qu'il est difficile de faire apparaître des différences de biomasse totale vraiment significatives dans la partie résultats. En outre, le modèle est assez simple, et ne prend pas en compte les processus de stress hydriques. Dans un cadre applicatif, utiliser un modèle de culture plus précis concernant les effets de stress hydriques sur les cultures ajouterait de la richesse au modèle de performance d'itinéraire technique, avec un $B(v,q)$ moins lisse, et les résultats des optimisations en seraient très différents.

Les données utilisées sont des cartes générées par géostatistiques. Le problème de ces cartes est qu'elles ne correspondent pas vraiment à la réalité terrain. Deux solutions sont envisageables : il faudrait donc tester cette méthode sur des données parcellaires réelles. Puisque ces données seraient plus irrégulières dans l'espace, il faudrait les associer à des cellules de Voronoi [7], qui permettent de partitionner des données spatiales. Cela nécessiterait quelques changements dans le code, puisqu'en l'état, l'algorithme traite des données sous la forme de grilles. C'est la solution la plus efficace pour tester la méthode, mais également celle demandant le plus d'efforts.

En ce qui concerne les contraintes morphologiques sur les zones, on a fait le choix ici d'explorer une seule contrainte : la contrainte morphologique. Avec plus de temps, les autres contraintes auraient pu être testées. On aurait pu également utiliser une combinaison de ces contraintes, mais il faut aussi prendre en compte le fait que plus on augmente le nombre de contraintes, plus la combinatoire de l'algorithme augmente.

4.2 Suites possibles du projet

A la suite de ce stage, on pourrait envisager une évolution du projet. Tout d'abord, le travail réalisé lors de ce stage a donné lieu à la création d'un code Python. Il est envisageable, à la suite de ce projet, de transformer ce code en librairie Python.

Enfin, l'algorithme de zonage piloté par un modèle d'itinéraire technique, dans sa méthode générale, a été envisagé de manière à pouvoir être utilisé pour d'autres modèles. Ainsi, il est envisageable d'adapter le code à d'autres modèles de performance d'itinéraire technique, dans d'autres éléments d'agriculture de précision, comme la modulation de dose d'engrais ou de produits phytosanitaires. Pour cela, deux conditions :

— L'hétérogénéité de la parcelle doit pouvoir être expliquée par une seule variable.
— On ne peut agir que sur un vecteur de quotas. Par exemple, cela pourrait correspondre à une quantité en viticulture q de pesticides pulvérisés.

5 Conclusion

Ce stage, réalisé au sein de l'UMR MISTEA de l'INRAE, a été réalisé dans un contexte scientifique et technique, visant à répondre aux enjeux de l'agriculture de précision. Le contexte de l'irrigation parcellaire, fortement impacté par les pénuries d'eau provoquées par le changement climatique, a servi d'étude de faisabilité.

Le projet s'est articulé autour de la conception d'un algorithme de zonage couplé à un modèle de performance d'itinéraire technique. L'objectif était de créer des cartes de modulation permettant d'adapter les pratiques d'irrigation en fonction des caractéristiques intra-parcellaires. Le modèle de performance d'itinéraire technique permet d'incorporer un critère de qualité de la carte, qui est pris en compte lors des choix de traitement sur les zones.

Les résultats obtenus dans le cas d'étude de l'irrigation ont montré la faisabilité et l'efficacité de cette approche, notamment en comparaison avec les méthodes d'irrigation homogène. Il faut néanmoins nuancer ces résultats. L'agriculture de précision permet certes d'économiser sur les intrants tout en maximisant ses rendements, mais elle est également très coûteuse sur d'autres aspects, notamment la collecte de données qui peut nécessiter des capteurs spécifiques et des compétences particulières. Ainsi, il est difficile de dire si ces solutions sont pérennes, ou si on devrait se tourner vers un autre type d'agriculture plus durable.

En conclusion, ce stage a permis de consolider des compétences techniques en optimisation et algorithmique, en développant un outil de zonage piloté par un modèle de performance d'itinéraire technique. Les perspectives d'amélioration et les suites possibles de ce projet ouvrent la voie à une généralisation de cette méthodologie à d'autres pratiques agricoles.

Références

[1] Kenza BOUMAZA et al. “Optimal control of a crop irrigation model under water scarcity”. In : Optimal Control Applications and Methods 42.6 (2021), p. 1612. DOI : 10.1002/oca.2749. URL : https://hal.inrae.fr/hal-03226630.

[2] Broyden-Fletcher-Goldfarb-Shanno method - Encyclopedia of Mathematics. URL : https://encyclopediaofmath.org/wiki/Broyden-Fletcher-Goldfarb-Shanno_method.

[3] Michael L. BYNUM et al. Pyomo–optimization modeling in python. Third. T. 67. Springer Science & Business Media, 2021.

[4] T. GINSBURG. “The Conjugate Gradient Method”. In : Handbook for Automatic Computation : Volume II : Linear Algebra. Sous la dir. de J. H. WILKINSON et al. Berlin, Heidelberg : Springer, 1971, p. 57-69. ISBN : 978-3-642-86940-2. DOI : 10.1007/978-3-642-86940-2_5. URL : https://doi.org/10.1007/978-3-642-86940-2_5.

[5] William E HART, Jean-Paul WATSON et David L WOODRUFF. “Pyomo : modeling and solving mathematical programs in Python”. In : Mathematical Programming Computation 3.3 (2011), p. 219-260.

[6] Nesrine KALBOUSSI et al. “About modeling and control strategies for scheduling crop irrigation ∗”. In : IFAC-PapersOnLine. 1st IFAC Workshop on Control Methods for Water Resource Systems CMWRS 2019 52.23 (1er jan. 2019), p. 43-48. ISSN : 2405-8963. DOI : 10.1016/j.ifacol.2019.11.007. URL : https://www.sciencedirect.com/science/article/pii/S2405896319309425.

[7] James M. KANG. “Voronoi Diagram”. In : Encyclopedia of GIS. Sous la dir. de Shashi SHEKHAR et Hui XIONG. Boston, MA : Springer US, 2008, p. 1232-1235. ISBN : 978-0-387-35973-1. DOI : 10.1007/978-0-387-35973-1_1461. URL : https://doi.org/10.1007/978-0-387-35973-1_1461.

[8] D. KRAFT. A software package for sequential quadratic programming. Forschungsbericht / Deutsche Forschungs- und Versuchsanstalt für Luft- und Raumfahrt, DFVLR. OCLC : 471799986. 1988. 33 p.

[9] Corentin LEROUX et al. “A New Approach for Zoning Irregularly-Spaced, Within-Field Data”. In : Computers and Electronics in Agriculture 141 (4 août 2017), p. 196-206. DOI : 10.1016/j.compag.2017.07.025.

[10] Patrice LOISEL et al. “An optimisation-based approach to generate interpretable within-field zones”. In : Precision Agriculture 20.1 (1er fév. 2019), p. 101-117. ISSN : 1573-1618. DOI : 10.1007/s11119-018-9584-3. URL : https://doi.org/10.1007/s11119-018-9584-3.

[11] J. A. NELDER et R. MEAD. “A Simplex Method for Function Minimization”. In : The Computer Journal 7.4 (1er jan. 1965), p. 308-313. ISSN : 0010-4620. DOI : 10.1093/comjnl/7.4.308. URL : https://doi.org/10.1093/comjnl/7.4.308.

[12] M. J. D. POWELL. “A Direct Search Optimization Method That Models the Objective and Constraint Functions by Linear Interpolation”. In : Advances in Optimization and Numerical Analysis. Sous la dir. de Susana GOMEZ et Jean-Pierre HENNART. Dordrecht : Springer Netherlands, 1994, p. 51-67. ISBN : 978-94-015-8330-5. DOI : 10.1007/978-94-015-8330-5_4. URL : https://doi.org/10.1007/978-94-015-8330-5_4.

[13] M. J. D. POWELL. “An efficient method for finding the minimum of a function of several variables without calculating derivatives”. In : The Computer Journal 7.2 (1er jan. 1964), p. 155-162. ISSN : 0010-4620. DOI : 10.1093/comjnl/7.2.155. URL : https://doi.org/10.1093/comjnl/7.2.155.

[14] Antonio Horta RIBEIRO. antonior92/ip-nonlinear-solver. original-date : 2017-05-28T20 :20 :51Z. 12 mars 2024. URL : https://github.com/antonior92/ip-nonlinear-solver.

[15] Pauli VIRTANEN et al. “SciPy 1.0 : Fundamental Algorithms for Scientific Computing in Python”. In : Nature Methods 17 (2020), p. 261-272. DOI : 10.1038/s41592-019-0686-2.

A Annexes

1 : Conditions de Karush-Kuhn-Tucker

Soit $f : \mathbb{R} \to \mathbb{R}$, une fonction appelée fonction objectif, et des fonctions $g_j : \mathbb{R}^n \to \mathbb{R}$, $1 \leq j \leq m$, appelées contraintes. On suppose que f et les g_j sont de classe C^1.

Si f admet un maximum en x^* sous les contraintes $g_j(x^*) \geq 0$ pour tout j, alors il existe des multiplicateurs de Lagrange $(\lambda_j)_{1 \leq j \leq m} \in \mathbb{R}^m$ vérifiant les conditions suivantes, dites conditions de Karush-Kuhn-Tucker :

$$\begin{aligned} &\frac{\partial f}{\partial x_k}(x^*) + \sum_{j=1}^{m} \lambda_j \frac{\partial g_j}{\partial x_k}(x^*) = 0, && 1 \leq k \leq n \\ &\lambda_j . g_j(x^*) = 0, \lambda_j \geq 0, && 1 \leq j \leq m \end{aligned} \tag{1}$$

2 : Détails des cartes générées par géostatistique

Les cartes sont des grilles de dimension 50×50. Elles sont générées en suivant un variogramme. Un variogramme représente la corrélation des données selon leur distance les une aux autres. On peut le résumer en trois paramètres :

- **L'effet pépite** : c'est la variabilité qu'on observe pour de courtes distances. Sa valeur est de 0.0002.
- **Le palier** : c'est la variabilité qu'on observe pour des grandes distances. Sa valeur est de 0.25.
- **La portée** : c'est la distance à partir de laquelle la variabilité observée est celle du palier. Au delà de cette distance, les données sont indépendantes. Sa valeur est de 25.

On obtient ainsi deux cartes, dont les distributions des valeurs de v sont représentées figure 24 et 25.

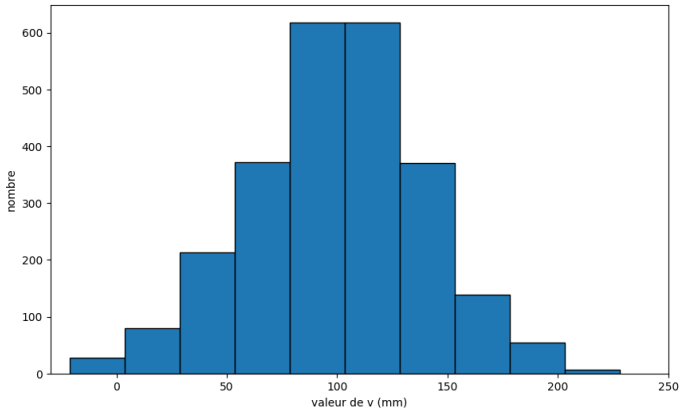


FIGURE 24 – répartition des valeurs de v pour la carte map1

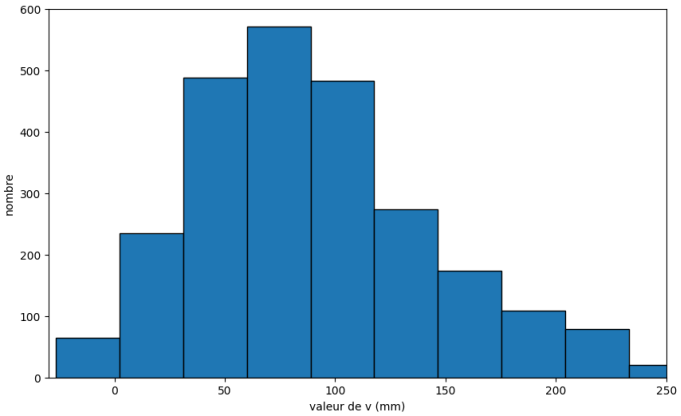


FIGURE 25 – répartition des valeurs de v pour la carte map2

3 : Etapes du processus de zonage

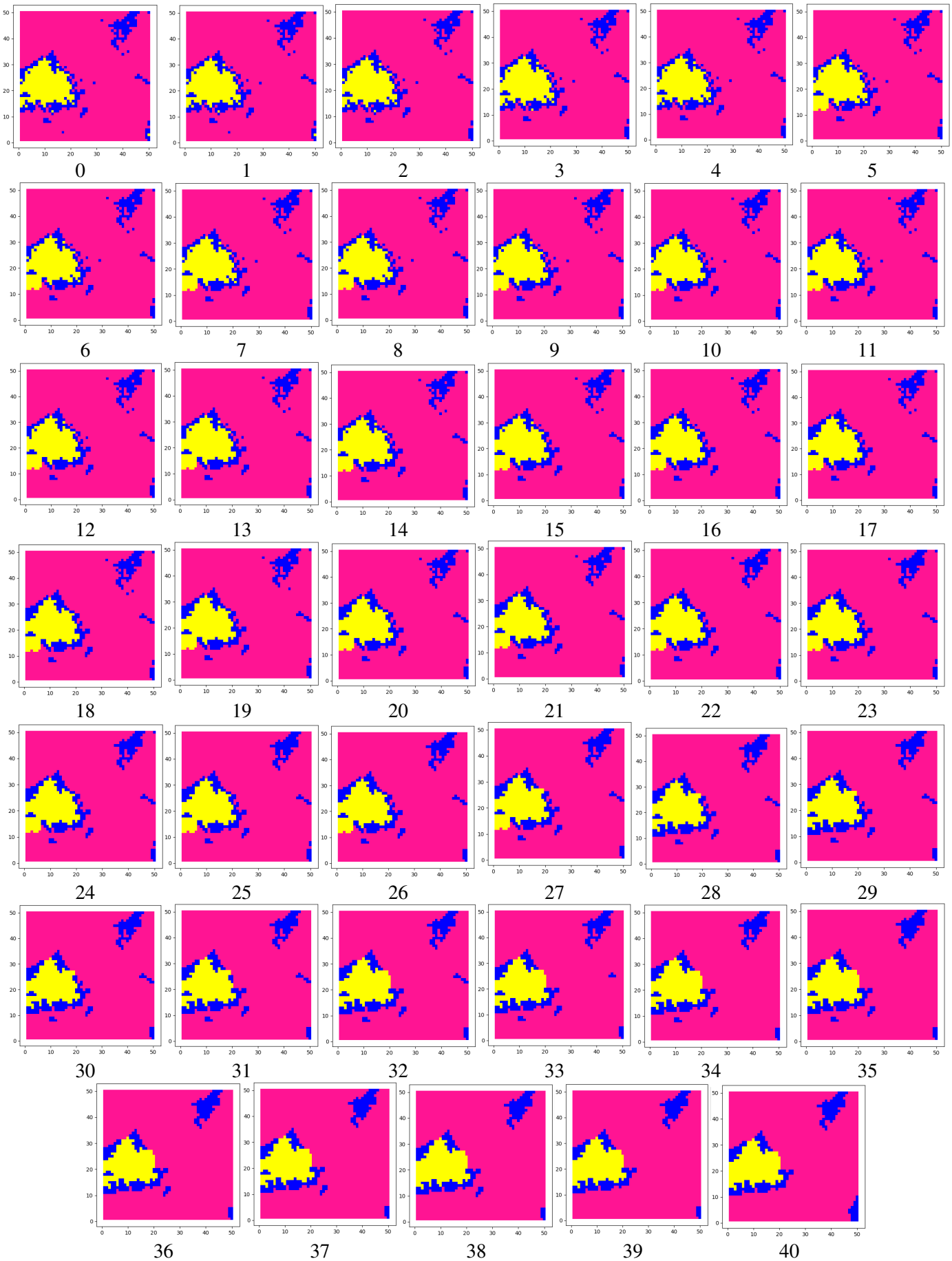


FIGURE 26 – étapes du processus de zonage