Prediction, neural network, and optimization : applications to the fields of agro-materials and telecommunication

Shohre Sadeghsa

Thèse de Doctorat

Mention : Informatique
Spécialité : Recherche Opérationnelle et Optimisation

Présentée à ***l'École Doctorale en Sciences Technologie et Santé (ED 585)***

De l'Université de Picardie Jules Verne

Par

Shohre Sadeghsa

Pour obtenir le grade de Docteur de l'Université de Picardie Jules Verne

Prédiction, réseau de neurones et optimisation : Applications aux domaines des agro-matériaux et de la télécommunication

Soutenue le 15/12/2021, après avis des rapporteurs, devant le jury d'examen :

M. Kaçem Imed, Professeur, Université de Lorraine	**Rapporteur**
M. Aîder Méziane, Professeur, Université des Sciences et de la Technologie Houari Boumediene	**Rapporteur**
M. Vassilis Zissimopoulos, Professeur, University of Athens	**Examinateur**
M. Gérard-Michel Cochard, Professeur, Université de Picardie Jules Verne	**Examinateur**
M. Mhand Hifi, Professeur, Université de Picardie Jules Verne	**Directeur de thèse**
M^{me} Adeline Goullieux, Maître de Conférences HDR, Université de Picardie Jules Verne	**Directrice de thèse**

Acknowledgements

I would like to thank the following people, without whom I would not have been able to complete this research.

The *EPROAD* team at the *University of Picardie Jules Verne*, especially to my supervisors professor *Mhand Hifi* and doctor *Adeline Goullieux*, whose insight and knowledge into the subject matter steered me through this research.

To the referees of my thesis, professor *Méziane Aider* and professor *Imed Kaçem* for reviewing this manuscript.

To professor *Gérard-Michel Cochard* and professor *Vassilis Zissimopoulos* for their support.

Thanks to *the minister and the region "Haut de France"* which financed my PhD for 3 years.

Special thanks to *IMaP* for the experimental data of the first part of this study.

Thanks to Professor *Lôys Thimonier*, Professor *Reza Tavakkoli-Moghaddam* for being so encouraging and doing everything to keep me motivated.

I wish to acknowledge the support of *my mother*, *Matthieu Antoniol* and his family. They kept me going on, and this work would not have been possible without their input.

Contents

List of figures

List of tables

List of terms and abbreviations

AI Artificial Intelligence

ANN Artificial Neural Networks

ANS Adaptive Neighborhood Search

Avg Average

BD Bulk Density

BP Back Propagation

BRP Basic Rounding Procedure

CCS Composite Compressive Strength

CCSPP Composite Compressive Strength Prediction Problem

CP Constuctive Procedure

CS(part1) Coating Substance

CS(part2) Complementary Solution

CST Composite Strength Test

DP Descent Procedure

DS Drying Shrinkage

EBRP Enhanced Basic Rounding Procedure

EDV Extreme Dimensional Variation

EPROAD Eco-PRocédés, Optimisation & Aide à la Décision

EQM Erreur Quadratique Moyenne

FAOSTAT Food and Agriculture Organization Corporate Statistical Database

FP Forward Propagation

GA Genetic Algorithm

GD Gradient Descent

GEP Gen Expression Programming

GRASP Greedy Random Adaptive Search Procedure

HSC High Strength Concrete

IBRP Iterative Basic Rounding Procedure

IMap Ingénierie des Matériaux et des Procédés

It Iteration

KCMBC K-Clustering Minimum Bi-clique Completion

LB Local Branching

LNS Large Neighborhood Search

Max Maximum

Min Minimum

MIP Mixed-Integer Programming

MSE Mean Squared Error

Nb Number

NNs Neural Networks

NSBH Neighborhood Search-Based Heuristic

NSH Neighborhood Search Heuristic

OPC Ordinary Portland Cement

PAB Procédure d'Arrondi de Base

PC Procédure Constructive

PD Procédure de Descente

PSO Particle Swarm Optimisation

RCC Résistance à la Compression des Composites

RCL Restricted Candidate List

RELU Rectified Linear Unit

RINS Relaxation Induced Neighborhood Search

RNA Réseaux neuronaux artificiels

RSBA Rounding Strategy-Based Algorithm

RSLB Rounding Strategy combined with Local Branching

SABL La Stratégie d'Arrondi combinée au Branchement Local

SCT Soft Computing Technics

St Dev Standard Deviation

Tanh Tangent Hyperbolic

Chapter 1

Introduction générale (*en français*)

Table des matières

1.1 Introduction

Dans le contexte des changements planétaires, l'utilisation des ressources végétales dans la production de matériaux composites constitue une solution alternative à l'exploitation des ressources fossiles et contribue à une gestion plus raisonnée de ces dernières [1]. Cependant, le développement d'un composite incorporant des granulats végétaux nécessite de prendre en compte la disponibilité dans le temps et dans l'espace des matières premières bio-sourcées, leur interchangeabilité, et leurs conséquences sur les caractéristiques fonctionnelles du matériau obtenu.

L'optimisation d'un composite végétal est confrontée à la grande variété des caractéristiques des matières premières et des paramètres opératoires, à la difficulté d'établir les relations existantes entre matières premières, procédés et produits finis, et à la nécessité de gérer les événements inattendus (tels que les catastrophes, les crises, les pannes...). Ainsi, un système fiable et durable est nécessaire afin de produire des agro-composites avec une efficacité continue [2].

Les méthodes d'intelligence artificielle doivent répondre (i) à la diversité des données disponibles, (ii) à la difficulté d'établir des relations de cause à effet, et (iii) au besoin particulier de prédire la production afin de maintenir une efficacité continue. Ces méthodes améliorent la compréhension et le contrôle de la production relative aux matériaux considérés.

Face à cette incertitude et à la variabilité des données, nous avons appliqué des

méthodes d'apprentissage automatique et d'intelligence artificielle pour prédire les résistances à la compression et à la flexion de certains agro-matériaux. L'intelligence artificielle permet notamment de prédire les paramètres de manière dynamique. Un modèle pourra ainsi s'adapter en fonction des données d'apprentissage. Les résultats en découlant finissent par être orientés vers les données desquelles il apprend.

Pour résoudre les problèmes mentionnés, la méthode des réseaux neuronaux artificiels (RNA) est recommandée afin de trouver une solution qui apprend à partir de données existantes afin de converger vers de meilleurs optima locaux [3].

Cette thèse se compose de deux parties. La première partie présente une étude de la résistance à la compression des composites en utilisant des méthodes d'intelligence artificielle. La deuxième partie présente l'optimisation de la chaîne logistique par regroupement des centres d'approvisionnement à l'aide de l'optimisation combinatoire.

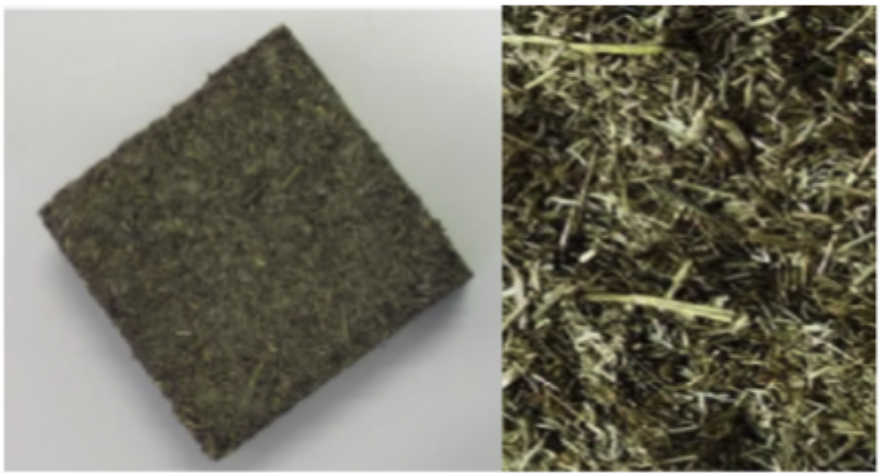
1.2 Etude de la résistance à la compression d'un composite végétal (Partie 1)

Dans cette étude, les expériences de test de résistance à la compression des composites (RCC) sont réalisées dans l'équipe *IMaP* du laboratoire EPROAD sur l'enrobage minéral d'anas de lin avant leur incorporation dans une matrice cimentaire.

La matrice cimentaire détermine principalement les propriétés d'un matériau multiphase et lui confère une résistance aux contraintes mécaniques. Deux substances d'enrobage (Coating substance, CS) ont été sélectionnées, la chaux et le ciment. Au cours de l'expérience, le rapport entre la quantité d'eau et la quantité de CS ainsi que le rapport entre la quantité de chaux et la quantité d'anas de lin varient. Les caractéristiques des anas de lin après enrobage et les caractéristiques des composites d'anas de lin obtenus sont mesurées.

Les caractéristiques mesurées pour les pailles sont la masse volumique, la capacité d'absorption d'eau et la diminution de la capacité d'absorption d'eau (par rapport aux pailles non traitées) et pour les composites à base d'anas, les caractéristiques mesurées sont la résistance à la flexion, la résistance à la compression, la masse volumique apparente, la variation dimensionnelle au séchage et la variation dimensionnelle extrême.

L'objectif de l'expérience est d'obtenir des propriétés mécaniques élevées et une faible variation dimensionnelle du composite. La figure 1.1(a) met en évidence un exemple de composite végétal (Al-Mohamadawi, Benhabib, Dheilly, and Goullieux, 2020) et la figure 1.1 (b) montre un exemple d'anas de lin (Barnat-Hunek, Smarzewski, and Brzyski, 2017).



(a) Composite de ciment (b) Anas de lin

Figure 1.1: Exemples (a) d'un composite végétal (Al-Mohamadawi, Benhabib, Dheilly, and Goullieux, 2020) et (b) d'anas de lin(Barnat-Hunek, Smarzewski, and Brzyski, 2017)

Les données expérimentales

L'ensemble des données utilisées dans cette étude provient d'un total de 123 expériences. Au total, 60 expériences ont été menées sur la chaux en tant que substance d'enrobage, en faisant varier le ratio d'anas de lin par chaux et le ratio d'eau par chaux.

De la même manière, un total de 60 expériences ont été menées sur le ciment comme substance d'enrobage, en variant le ratio d'anas par ciment et le ratio d'eau par ciment. Trois expériences avec des anas non enrobés ont également été conduites.

Le ratio d'anas par substance d'enrobage est dans la gamme $\{1/3, 1/2, 2/3, 1, 2\}$ et pour le ratio d'eau par substance d'enrobage dans la gamme $\{1/2, 3/4, 1, 2\}$.

Pour chaque expérience, avec ou sans substance d'enrobage, la capacité d'absorption d'eau et la masse volumique apparente des anas ont été mesurées. Certaines caractéristiques des composites d'anas de lin sont mesurées pour les anas non enrobés et enrobés. Ces caractéristiques sont : la résistance à la flexion, la résistance à la compression, la masse volumique apparente, le retrait au séchage et les variations dimensionnelles extrêmes.

Les expériences avec de la chaux et du ciment comme substances d'enrobage sont composées de 20 combinaisons de 5 groupes d'anas par substance d'enrobage et de 4 groupes d'eau par substance d'enrobage. Chacune de ces expériences est répétée 3 fois pour un total de 60 expériences pour chaque substance d'enrobage.

1.3 Une méthodologie pour prédire la résistance à la compression d'un composite végétal en utilisant la méthode de descente de gradient

Il existe plusieurs études dans la littérature pour mettre en œuvre divers algorithmes d'optimisation. La descente de gradient est l'une des méthodes d'optimisation les plus populaires, utilisée dans les études sur la science de l'apprentissage automatique (par exemple, Bengio, Boulanger-Lewandowski, and Pascanu, 2013).

La descente de gradient a trois variantes : (I) descente de gradient par lot, (II) descente de gradient stochastique et (III) descente de gradient par mini-batch. La différence entre ces méthodes réside dans la quantité de données disponibles.

Dans cette étude, la méthode de descente de gradient est appliquée pour trouver la solution optimale pour le modèle linéaire proposé.

La fonction objectif appelée fonction de coût peut être formulée dans l'équation 1.1.

$$C(X) = \frac{1}{2m} \sum_{j=1}^{m} (\upsilon(X)_j - Y_j)^2 \tag{1.1}$$

Où $C(X)$ représente la fonction de coût, m représente le nombre des échantillons d'apprentissage et Y représente la valeur réelle de la résistance à la compression du composite. Pour chaque exemple j, $\upsilon(X)$ représente la valeur prédite de la résistance à la compression du composite. En fait, $\upsilon(X)$ est la fonction d'hypothèse qui est définie dans 1.2.

$$\upsilon(X) = \theta_0 + X\theta \tag{1.2}$$

Où X représente un vecteur des caractéristiques $X = \{X_1, X_2, ..., X_n\}$ et $\theta = \{\theta_1, \theta_2, ..., \theta_n\}$ représente leurs coefficients, et θ_0 représente la valeur fixe de l'équation en conséquence.

Nous pouvons reformuler l'équation 1.1 en modèle de minimisation dans 1.3.

$$\theta_{It+1} := \theta_{It} - \frac{\lambda}{m} \sum_{j=1}^{m} (P(\theta)_j - Y_i) X_j \tag{1.3}$$

La méthode de descente de gradient va minimiser la fonction objectif au cours de It itérations en mettant à jour les coefficients des paramètres du modèle. Dans cette équation, $-\frac{1}{m} \sum_{j=1}^{m} (P(\theta)_j - Y_j) X_j$ représente le gradient de la fonction de coût décrite

dans l'équation 1.1. À chaque itération It de la descente de gradient, θ sera mis à jour dans la direction opposée du gradient de la fonction objectif. Le paramètre λ détermine le taux d'apprentissage. Il détermine la taille des étapes pour atteindre un minimum local. En d'autres termes, à chaque étape, nous nous déplaçons dans la direction du gradient de la fonction objectif vers le bas de la pente, jusqu'à ce que nous atteignions un minimum local. Nous rappelons que la fonction objectif ainsi que le modèle mathématique est convexe[1], donc un minimum local est aussi le minimum global. Par conséquent, les solutions sont garanties comme étant optimales.

Principales étapes de l'application du modèle d'optimisation:

- Mélange des données: Le mélange est effectué de manière à ce que seul l'ordre des données soit permuté au hasard, et non les paramètres des expériences.
- Séparer l'ensemble de données en un ensemble d'apprentissage et d'évaluation : L'analyse statistique sera effectuée dans l'ensemble d'apprentissage et la qualité de la méthode sera examinée dans l'ensemble d'évaluation.
- Normalisation de la moyenne: La mise à l'échelle des caractéristiques par une normalisation moyenne permet de guider l'algorithme de descente de gradient vers une convergence. Nous avons normalisé les données en utilisant la formulation 1.4 :

$$\frac{X - \mu}{\sigma} \tag{1.4}$$

  Où X est la valeur du paramètre à normaliser, μ et σ représentent la moyenne et les écarts types.

- Initialiser aléatoirement theta : Theta (θ)est le coefficient des paramètres décrits dans la section1.3. Avant d'appliquer la descente par gradient, thêta doit être initialisé.
- Appliquer l'algorithme de descente de gradient pour trouver un thêta optimisé. En initialisant les deux paramètres (nombre d'itérations et taux d'apprentissage), l'algorithme de descente de gradient peut être appliqué aux données.
- L'équation de prédiction :

$$Y = X \times \theta \quad \forall \theta \in \{\theta_0, \theta_1, \theta_2, ..., \theta_n\} \quad and \quad \forall X \in \{X_0, X_1, X_2, ..., X_n\} \tag{1.5}$$

  La valeur cible Y de chaque échantillon peut être prédite à l'aide de l'équation 1.5. Où X est la donnée normalisée de l'ensemble de test et n est le nombre de paramètre d'entrée (features). Rappelons que θ_0 est le coefficient de X_0 qui représente la variable de biais (bias unit).

[1] n'ayant aucun angle intérieur supérieur à 180∘

Dans cette étude, nous avons étudié les effets d'un cas réel expérimental lié à l'enrobage de la paille de lin avant son incorporation dans une matrice cimentaire. Un modèle simple a été proposé pour aborder le problème, où une méthode de gradient de descente par lots a été employée pour prédire la résistance à la compression d'un composite végétal.

Étant donné que le problème étudié est lié à une situation pratique, il existe de nombreuses possibilités d'études plus poussées impliquant des méthodes efficaces capables de prédire les paramètres. Premièrement, les réseaux neuronaux peuvent être appliqués pour prédire certains paramètres et donc, dans certains cas, ils peuvent être utilisés pour fournir des estimations précises. Deuxièmement, l'hybridation entre les réseaux neuronaux et les techniques de recherche opérationnelle peut également être envisagée pour obtenir de meilleures prédictions des paramètres, en particulier dans le cas de fonctions multi-objectifs, comme la prédiction des paramètres et la minimisation du prix global de la matière première requise, pour concevoir le composite cible.

1.4 Réseaux neuronaux artificiels pour prédire la résistance à la compression et à la flexion des composites

Les réseaux neuronaux artificiels (RNA) ou, plus simplement, les réseaux neuronaux (RN) sont issus d'une compréhension à la fois mathématique et biologique. Les RNs s'inspirent du cerveau humain et imite le processus de résolution de solutions à l'aide de neurones. Le'RNA est connu comme une méthode capable de modéliser des processus complexes et non linéaires. Elle est largement utilisée dans de nombreuses disciplines, y compris le génie civil et la modélisation des composites cimentaires(Asteris, Kolovos, Douvika, and Roinos, 2016; Malagavelli and Manalel, 2014; Mansouri, Ozbakkaloglu, Kisi, and Xie, 2016; Plevris and Asteris, 2014; Topcu and Sarıdemir, 2008).

Dans cette étude, l'objectif est de concevoir des réseaux afin de prédire la résistance à la compression et la résistance à la flexion du composite. Les données expérimentales d'un test de résistance à la compression de 28 jours sont disponibles dans Goullieux, Hifi, and Sadeghsa, 2020a. La structure théorique d'un réseau est représentée sur la figure 1.2: Dans la couche d'entrée, 6 caractéristiques d'un mélange composite biosourcé sont fournies au réseau.

Les neurones et leurs poids sont les principaux composants des RNAs. Les RNAs sont des systèmes multicouches avec un minimum de trois couches : la couche d'entrée, la ou les couches cachées et la couche de sortie. Les étapes de la création des RNAs sont présentées ci-dessous :

- Prétraitement des données : séparer les données en un ensemble d'apprentissage (80%) et un ensemble de test (20%) puis appliquer la normalisation. Nous avons

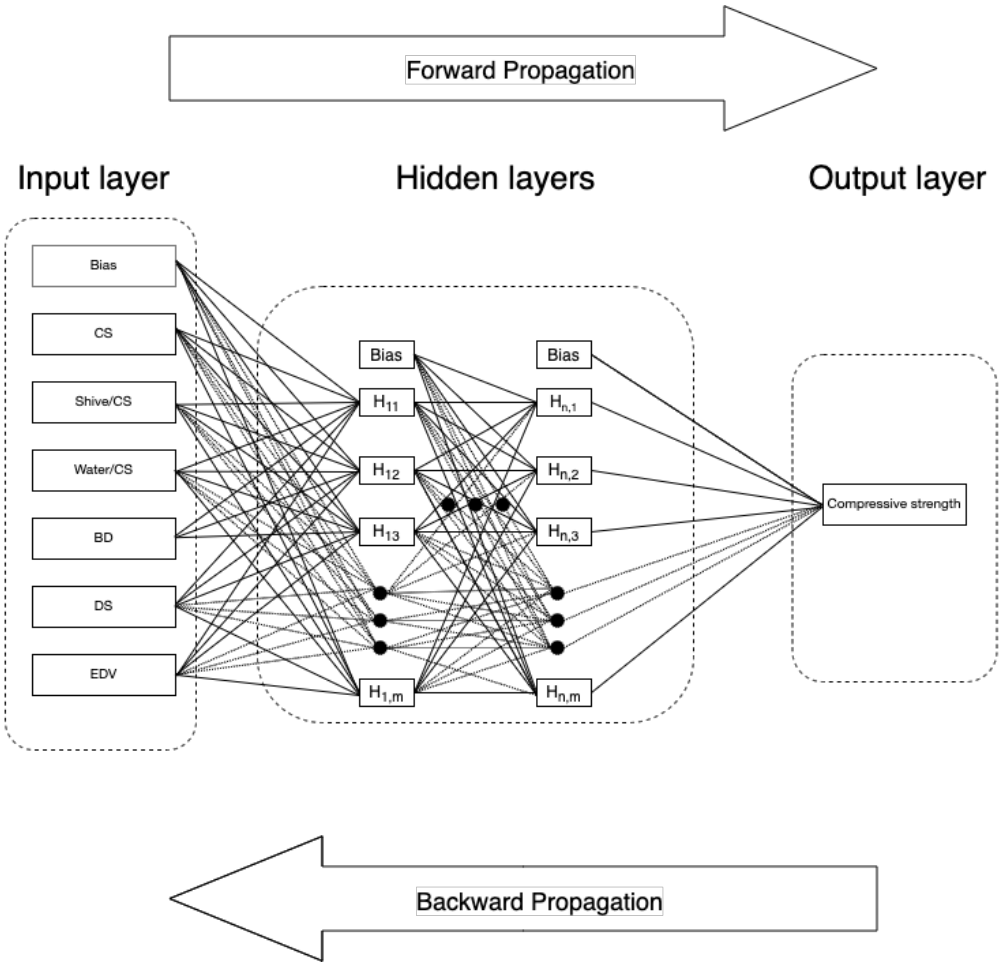


Figure 1.2: Représentation globale des réseaux neuronaux artificiels (RNAs); La description des abréviations utilisées dans cette figure est la suivante; CS: substances d'enrobage (coating substance), Shive/CS: proportion de paille de lin/substance d'enrobage (ratio of the shive per coating substance), water/CS: proportion d'eau par substance d'enrobage (ratio of the water per coating substance), BD: résistance à la flexion (bending strength), DS: retrait au séchage (drying shrinkage), EDV: variation dimensionnelle extrême (extreme dimensional variation), $H_{n,m}$: m^{eme} élément de n^{eme} couche cachée

appliqué quatre méthodes de normalisation : 1) moyenne, 2) min-max, 3) divide-max, et 4) aucune normalisation.

- Configuration du réseau : déterminer le nombre de couches cachées et le nombre de neurones dans chaque couche cachée.
- Initialisation du réseau (propagation vers l'avant) : initialiser les poids du réseau et les neurones en direction avant.
- Optimisation du réseau : actualiser et optimiser les poids à l'aide d'une méthode de descente de gradient dans une direction vers l'arrière.
- Modèle d'évaluation : évaluer la méthode et calculer les erreurs de prédiction
- Résultats et conclusions

Un ensemble de 123 données expérimentales a été utilisé. Les composites étudiés sont constitués d'une matrice cimentaire et d'anas de lin enrobés avec du ciment ou de la chaux selon différents paramètres d'enrobage (rapports substance d'enrobage/eau et anas/substance d'enrobage). Les composites sont caractérisés par leur masse volumique apparente, leur retrait au séchage, leur variation dimensionnelle extrême et leurs résistances à la compression et à la flexion.

Cette étude présente une application des réseaux neuronaux pour la prédiction de la résistance à la compression et à la flexion de composites d'anas de lin. Les valeurs prédites sont comparées aux résultats expérimentaux évalués à partir d'expériences conduites dans l'unité de recherche d'EPROAD.

Au total, 67360 modèles différents à une, deux et trois couches avec de 1 à 20 neurones dans les couches cachées ont été créés pour chaque paramètre cible. Les réseaux conçus sont testés en utilisant quatre méthodes de normalisation et deux fonctions d'activation (tanh et relu). Les résultats prédits sont validés à l'aide de mesures statistiques : l'erreur quadratique moyenne (EQM) et le score R2. Les informations détaillées des meilleures configurations de chaque modèle sont résumées dans le Tableau 1.

Table 1.1: Les meilleures configurations de chaque modèle

Paramètres de l'analyse	**Paramètre cible**					
	Résistance à la compression			Résistance à la flexion		
	1-couche	2-couches	3-couches	1-couche	2-couches	3-couches
Normalisation	devide-max	moyenne	moyenne	min-max	min-max	devide-max
Activation	relu	relu	tanh	tanh	relu	relu
Format du réseau	11	9-10	11-16-6	10	10-10	19-2-20
EQM (apprentissage)	0.05	0.02	1.03E-05	0.003	0.022	0.02
R2 (apprentissage)	0.99	1.00	1.00	0.99	0.95	0.96
EQM (test)	0.16	0.14	0.09	0.06	0.03	0.02
R2 (test)	0.97	0.97	0.98	0.86	0.89	0.93

Comme indiqué dans le Tableau 1, pour prédire la résistance à la compression sur l'ensemble des tests, la valeur la plus élevée de R2 a été obtenue pour le modèle a trois

couches cachées. Les données ont été normalisées en utilisant la méthode de normalisation de la moyenne et la fonction d'activation tanh. Cette architecture présente une EQM de 1.03E-5 et 0.09 et un R2 de 1.00 et 0.98 sur les ensembles d'apprentissage et de test respectivement.

De même, pour prédire la résistance à la flexion, le meilleur modèle, selon la valeur la plus élevée de R2, possède trois couches cachées. Les données ont été normalisées avec la méthode divide-max et relu comme fonction d'activation. Les valeurs EQM pour les ensembles d'apprentissage et de test sont respectivement égales à 0.017 et 0.025. Les scores de R2 sont respectivement de 0.96 et 0.93.

En conclusion, nous avons développé des modèles de réseaux neuronaux efficaces pour prédire et optimiser des agro-matériaux de manière dynamique et adaptative vis à vis de l'incertitude des données.

1.5 Modèle mathématique et optimisation pour prédire les paramètres de test de résistance à la compression

Un matériau est souvent caractérisé par sa résistance à la compression qui est étroitement liée à sa masse volumique apparente.

Comme indiqué dans le document de Siqueira Tango, 1998, l'essai de résistance à la compression RCC a été introduit pour la première fois pour analyser le comportement du matériau cible, qui prend conventionnellement 28 jours pour une expérience réelle.

La prédiction du matériau cible avec ses paramètres estimés en utilisant des méthodes basées sur l'optimisation et/ou l'apprentissage (de Siqueira Tango, 1998 et Snell, Van Roekel, and Wallace, 1989) peut être considérée comme une méthode alternative ou complémentaire pour réaliser une expérience basée sur la RCC.

Les essais ont été menés dans le laboratoire EPROAD sur à base d'anas enrobés (i) 60 expériences avec la chaux comme substance d'enrobage, (ii) 60 expériences avec le ciment comme substance d'enrobage.

Comme la chaux et le ciment sont tous deux utilisés, nous avons ensuite étudié deux expériences différentes:

1. *L'étude de la chaux.* L'objectif de la première étude est de prédire la quantité d'eau par rapport à la quantité de CS et, le ratio de la quantité de paille par rapport à la quantité de CS pour la chaux en ayant d'une part la masse volumique apparente et la capacité d'absorption d'eau des pailles, et d'autre part la résistance à la flexion, la résistance à la compression, la masse volumique apparente, la variation dimensionnelle au séchage (drying shrikage ou DS) et la variation dimensionnelle

Table 1.2: Représentation des variables utilisées

Symbole	Paramètre
X_1	Absorption de l'eau de paille de lin (%)
X_2	Masse volumique apparente d'anas de lin (kg/m^3)
X_3	Résistance à la flexion des composites (MPa)
X_4	Résistance à la compression du composite(MPa)
X_5	Masse volumique apparente du composite (kg/m^3)
X_6	Retrait au séchage du composite(mm/m)
X_7	Variation dimensionelle extrême du composite(mm/m)
Y_1	Paille de lin/Substance d'enrobage (S/CS)
Y_2	Eau/Substance d'enrobage (W/CS)

extrême (Extreme dimensional variations ou EDV) des composites. Cette expérience va induire deux équations où chacune d'entre elles est utilisée pour prédire un rapport.

2. *L'étude du ciment.* En plus de la première étude, la deuxième expérience permettra de calculer les deux ratios liés au ciment.

Le tableau 1.2 illustre les paramètres utilisés dans ces deux études. Par conséquent, en utilisant tous les symboles introduits dans le tableau 1.2,

- D'une part, pour l'expérience avec la chaux, les paramètres sont représentés par l'ensemble $\{X_1, X_2, X_3, X_4, X_5, X_6, X_7\}$ tel que Y_1 désigne la variable à prédire pour la chaux.
- D'autre part, pour la deuxième équation, les paramètres sont caractérisés par l'ensemble $\{X_1, X_2, X_3, X_4, X_5, X_6, X_7\}$ tel que Y_2 désigne la variable à prédire.

Nous notons que pour la deuxième expérience, les paramètres similaires sont sélectionnés en utilisant les données correspondant au ciment comme substance d'enrobage.

Analyse de régression

L'analyse de régression est une approche statistique puissante qui peut être appliquée pour supprimer une relation entre les variables dépendantes et indépendantes. Elle peut généralement aider à modéliser une certaine relation entre les variables utilisées, où plusieurs types de régressions peuvent être considérés, comme la régression linéaire, la régression non linéaire et la régression linéaire multiple. Comme la littérature mentionne la préférence pour l'utilisation de régressions linéaires ou multi-linéaires, nous avons décidé d'appliquer la régression linéaire dans cette étude.

De plus, l'utilisation d'une régression linéaire nécessite le recours à une des méthodes de descente de gradient. La descente de gradient est un algorithme d'optimisation utilisé pour minimiser (ou maximiser) la fonction de coût (profit) dans divers algorithmes d'apprentissage automatique. Elle est essentiellement utilisée pour mettre à jour les paramètres du modèle d'apprentissage considéré. Comme discuté dans la littérature (Bengio et al., 2013 et Ruder, 2016), il existe trois types de méthodes basées sur la descente de gradient:

- Méthode de descente de gradient par batch : A chaque itération du processus de recherche, tous les exemples d'entraînement sont traités. Cependant, cette version est intéressante lorsque la taille des données reste faible. Par conséquent, plus la taille des exemples d'apprentissage (données) est grande, plus la convergence de la méthode est lente.
- Méthode de descente de gradient stochastique : A chaque itération du processus de recherche, un seul échantillon d'entraînement est traité et tous les paramètres sont mis à jour. Dans ce cas, on peut observer que cette version devient intéressante lorsque la taille des données (échantillons d'entraînement) reste faible. Elle induit une accélération par rapport à la version précédente, même si elle reste lourde lorsque le nombre d'échantillons d'entraînement augmente.
- Méthode de descente de gradient en mini-batch : Cette version est souvent plus rapide que les deux précédentes, même si la manipulation de cette approche reste subtile pour une bonne parallélisation.

Comme nous l'avons déjà mentionné, les données sont chères et leur collecte prend du temps. Par conséquent, la quantité de données disponibles étant limitée, la descente de gradient par batch qui calcule le gradient pour l'ensemble des données peut être utiliser en toute sécurité.

Méthode de descente du gradient par batch

Pour cette version de la descente de gradient, la fonction objectif est définie comme la somme moyenne des erreurs quadratiques de la prédiction pour la régression linéaire. Sous une forme différente, cette fonction objectif peut être écrite comme la fonction de coût de minimisation suivante(Equation 1.6):

$$C(\theta) = \frac{1}{2m} \sum_{i=1}^{m} \big(P(\theta)_i - Y_i\big)^2, \tag{1.6}$$

Où C(θ) représente la fonction de coût pour chaque θ, m désigne le nombre d'échantillons d'apprentissage tandis que Y est la valeur réelle de la variable, qui est prédite à l'aide de la fonction de coût qui doit être minimisée.

La méthode tente de minimiser la fonction objectif en utilisant une procédure itérative, où les coefficients des paramètres du modèle sont mis à jour comme suit (Equation 1.7):

$$\theta_j := \theta_j - \frac{\alpha}{m}\sum_{i=1}^{m}(P(\theta)_i - Y_i)X_i \tag{1.7}$$

A chaque itération j de la descente de gradient, θ sera mis à jour dans la direction opposée du gradient de la fonction objectif. Le paramètre α est le taux d'apprentissage. Il détermine la taille des étapes pour atteindre un minimum local. En d'autres termes, à chaque étape, nous nous déplaçons dans la direction du gradient de la fonction objectif vers le bas d'une pente, jusqu'à ce qu'un minimum local soit atteint.

Méthode de descente de gradient stochastique

La différence entre les méthodes de descente de gradient batch et stochastique réside dans la forme de la fonction objectif utilisée et dans le processus d'optimisation. Comme souligné ci-dessus, il est préférable d'utiliser cette méthode lorsque la taille des données devient importante.

Dans ce cas, le processus d'apprentissage est effectué sur chaque échantillon d'apprentissage tandis que tous les paramètres sont mis à jour à chaque itération. La fonction de coût utilisée est représentée comme suit :

$$C(\theta,(x^i,y^i)) = \frac{1}{2}\big(P(\theta,x^i) - Y^i\big)^2 \tag{1.8}$$

et

$$C_{train} = \frac{1}{m}\sum_{i=1}^{m}(C(\theta,(x^i,y^i))), \tag{1.9}$$

Où C(θ) désigne la fonction de coût pour chaque θ, m représente le nombre d'échantillons d'apprentissage tandis que Y est la valeur réelle de la variable. Cette valeur est prédite en utilisant la fonction de coût qui doit être minimisée. Enfin, (x^i, y^i) représente les caractéristiques et la valeur réelle d'un échantillon d'apprentissage.

De plus, la descente de gradient stochastique applique de manière répétée l'équation suivante :

$$\theta_j := \theta_j - \alpha(P(\theta,x^i) - y^i)x_j^i, \tag{1.10}$$

Où $(P(\theta,x^i) - y^i)x_j^i$ est la dérivée des fonctions de coût telles que l'équation (1.10) sera pour chaque $i = 1, 2, \ldots, m$ et $j = 0, 1, \ldots, n$, où m est le nombre d'exemples et n le nombre de caractéristiques.

La méthode proposée est une version étendue de la méthode proposée dans Goullieux et al., 2020a. La méthode de descente de gradient par lots et les étapes d'optimisation sont décrites dans la section précédente 1.3.

Dans ce document, nous avons étudié les effets d'un cas réel d'enrobage de la paille de lin avant son incorporation dans une matrice cimentaire. Un modèle simple a été proposé pour aborder le problème, où une méthode de descente de gradient par lots a été employée pour prédire les paramètres liés à la résistance à la compression du composite. Un tel problème nécessite 28 jours pour obtenir un matériau final, tandis que la méthode proposée converge vers de bonnes solutions approximatives en moins d'une minute.

1.6 La fusion d'établissements d'approvisionnement par optimisation combinatoire : K-clustering (partie 2)

La deuxième partie de cette thèse propose de consolider les installations d'approvisionnement afin d'assurer une production et une distribution continues du matériau à base de végétal enrobé étudié. Dans cette partie, le problème de l'achèvement minimum par k-clustering est étudié pour consolider les installations d'approvisionnement.

La production des composites végétaux est très sensible. Les matériaux d'approvisionnement doivent être entièrement disponibles à un certain endroit (x) à un certain moment (t). Comme les matériaux végétaux sont très sensibles aux conditions climatiques, il est très important d'envisager une stratégie de secours pour choisir les centres de la chaîne d'approvisionnement.

La consolidation des centres d'approvisionnement, de production, de distribution et de vente au détail est une solution alternative pour réduire les coûts d'inventaire et de vente au détail (Teo, Ou, and Goh, 2001; van der Vlist and Broekmeulen, 2006).

Le regroupement, la consolidation ou le clustering font référence à l'action de fusionner deux ou plusieurs unités (Koch and Wäscher, 2016). C'est l'acte de réduire le nombre de centres à un nombre plus petit (ou clusters) en fusionnant les centres existants. Afin de maintenir l'efficacité continue des chaînes d'approvisionnement et de production, chaque site doit offrir l'ensemble des services qui étaient auparavant fournis par les sites remplacés (Mansour, Tabbara, and Dana, 2004).

Les exemples de regroupement peuvent concerner non seulement les centres d'approvisionnement mais aussi les sites éducatifs pour regrouper les écoles publiques en un plus petit nombre d'écoles existantes, les entrepôts, les systèmes de télécommunication, les magasins, les centres de distribution, etc.

La chaîne d'approvisionnement se compose de trois sections : (i) centre(s) d'approvisionnement, (ii) centre(s) de production, (ii) et centre(s) de distribution. Le problème du regroupement peut se poser dans n'importe quelle partie de cette chaîne. Le problème du k-clustering peut être défini dans n'importe laquelle des deux parties de la chaîne d'approvisionnement qui sont en relation directe.

La durabilité dans la chaîne logistique

Pour assurer une production continue et une commercialisation en ligne, il est nécessaire de garantir la fiabilité de tous les éléments de la chaîne d'approvisionnement. La figure8.1 représente un exemple de chaîne logistique. En général, un système de chaîne logistique se compose de trois parties principales : les installations d'approvisionnement, de production et de distribution.

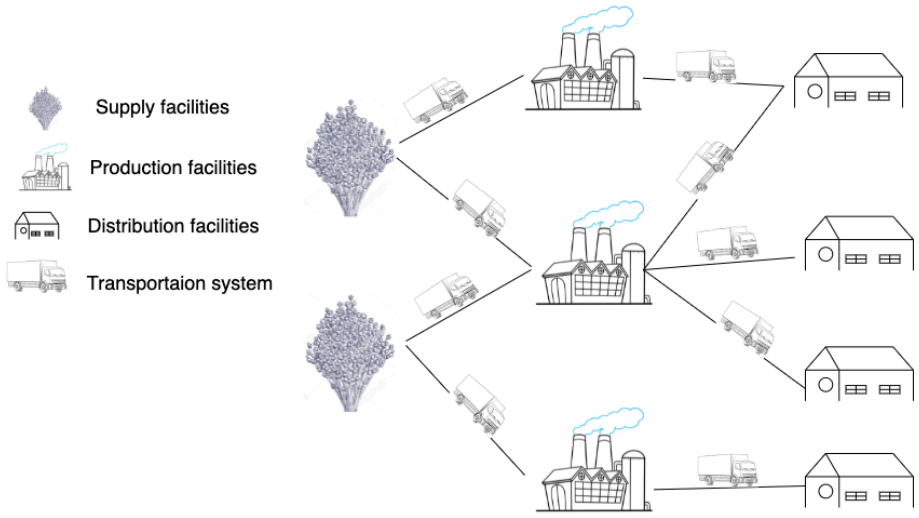


Figure 1.3: Un exemple de système de chaîne logistique

Les décisions relatives à la chaîne d'approvisionnement comprennent les décisions stratégiques (à long terme) visant à créer un système, ainsi que les décisions à moyen et à court terme visant à maintenir un système existant. Les principaux problèmes de la chaîne d'approvisionnement sont la localisation des sites de production, le routage et l'ordonnancement, le stockage des matières premières, la gestion des stocks, l'optimisation des systèmes de transport, la distribution et le marketing.

Face à ces problèmes, il est important pour les entreprises de construire un modèle économique fiable et une stratégie d'achat et de marketing compétitive. La modélisation des flux d'information liés aux problèmes de localisation et de planification d'une unité de production sur un réseau de transport urbain et rural, génère des systèmes très complexes (Daskin, Snyder, and Berger, 2005).

Les problèmes de localisation et de consolidation relèvent de la recherche opérationnelle et de la prise de décision. D'une part, un problème de localisation conduit à une modélisation monocritère et multicritère ainsi qu'à la proposition de méthodes de résolution efficaces. D'autre part, l'extension du modèle à une variante robuste peut être tentée en fonction de la rareté des matériaux nécessaires à la fabrication du composite végétal et des critères liés aux décideurs.

La production de composites végétaux est très sensible. Les matériaux d'approvisionnement doivent être entièrement disponibles à un certain endroit (x) et à un certain moment (t). En raison de la grande sensibilité des matériaux végétaux, il est très important de considérer une stratégie de secours pour choisir les centres de la chaîne d'approvisionnement.

Un élément qui peut être considéré pour augmenter la robustesse d'une chaîne d'approvisionnement existante est d'envisager la possibilité de consolider le système d'approvisionnement. Envisager de fusionner les centres d'approvisionnement est un élément qui peut augmenter la durabilité et la robustesse du système en cas de perte d'un centre en raison d'une fermeture d'urgence, d'une catastrophe ou de raisons économiques.

Problème de complétion minimale des graphes bi-cliques par K-clustering

Étant donné un graphe biparti G(S, C, E), le problème de "K-clustering minimum bi-clique completion" (kCMBC) consiste à trouver k clusters bipartis (sous-graphes) tels que :

- $\forall i \in S$; S indique le premier ensemble de sommets de G, il apparaît une fois dans un cluster,
- $\forall j \in C$; C indique le deuxième ensemble de sommets de G, il apparaît dans chaque cluster dans lequel au moins un de ses voisins apparaît,
- le nombre total d'arêtes à ajouter à chaque sous-graphe pour devenir un sous-graphe bipartite complet est minimisé

Cette problématique est également décrite comme une application du monde réel liée au domaine des télécommunications. En effet, afin de limiter certains canaux donnés dans les transmissions multicast, en fonction d'un ensemble de demandes de clients liées à des services, k sessions multicast doivent être construites pour inclure des demandes indépendantes. k sessions de multidiffusion doivent être construites pour inclure des demandes indépendantes. D'une part, un service est contenu dans une seule session multicast et, d'autre part, un client peut apparaître dans plusieurs sessions (Hifi and Sadeghsa, 2020a).

Dans la littérature, des heuristiques déterministes et stochastiques sont souvent utilisées pour résoudre approximativement le problème du clustering. L'établissement d'un modèle adaptatif aux données avec des modèles heuristiques et méta-heuristiques dans les optimisations combinatoires assurera la durabilité et la fiabilité de la production de composites végétaux. De plus, les résultats de la convergence conduiront à un modèle commercial fiable et à une stratégie d'achat et de marketing compétitive.

Un exemple du problème de K-clustering

Nous considérons un exemple de problème de K-clustering. Dans cet exemple, nous considérons un graphe bipartite avec 4 sommets dans les services (S) et 5 sommets dans les clients (C). Il est nécessaire de fusionner les centres de services de manière à obtenir 2 clusters ($K = 2$). Chaque nouveau cluster doit servir les clients sans perdre personne. Pour cela, de nouvelles arêtes doivent être ajoutées afin que les nouveaux clusters soient un graphe bipartite complet. La figure 1.4(i) présente un exemple de graphe bipartite $G(S, C, K)$ avec $S = \{a, b, c, d\}$, $C = \{1, 2, 3, 4, 5\}$, et $K = 2$.

Les figures 1.4(ii)et(iii) représentent les deux solutions différentes avec des valeurs objectives égales. Les boîtes autour de chaque nœud de S, représentent les clusters. Comme dans la figure 1.4, pour obtenir la valeur objective de chaque solution, il suffit d'affecter chaque nœud $i \in S$ dans un cluster k. Le nombre près d'un sommet dans S représente le nombre d'arêtes partant de ce sommet.

Dans la solution (ii), le nombre (2+1) près du nœud b dans S signifie 2 arêtes qui quittent le nœud b du graphe initial plus une arête supplémentaire pour faire un sous-graphe complet.

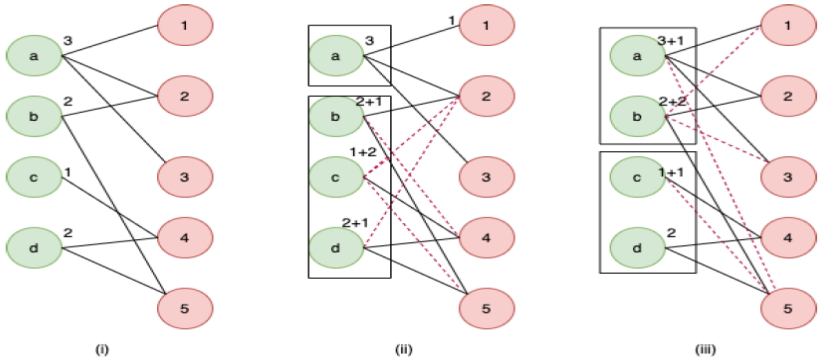


Figure 1.4: Exemple du problème de complétion minimale des graphes bi-cliques par K-clustering : (i) est un réseau initial avec 4 sommets dans S, 5 sommets dans C, (ii) et (iii) sont les solutions égales avec la valeur objective égale à 4.

Formulations mathématiques

Le problème $KCMBC$ décrit est présenté avec des formulations mathématiques. La formulation mathématique initiale pour $KCMBC$ est proposée dans Gualandi, Maffioli, and Magni, 2013. Le modèle étant non linéaire, un modèle linéaire est proposé.

Le modèle quadratique

Tout d'abord, definir la variable x_{ik} égale à 1 si le nœud i dans la partition $k, k \in S$, 0 sinon. Deuxièmement, fixer y_{ik} égal à 1 si le nœud $j,\ j \in C$, est affecté au cluster k, 0 sinon. Ainsi, la description formelle de kCMBC (notée $\mathrm{Q}_{k\mathrm{CMBC}}$) est décrite comme suit (Faure *et al.* Faure, Chrétienne, Gourdin, and Sourd, 2007) :

$$\sum_{k \in K} \sum_{(i,j) \in \bar{E}} x_{ik} y_{ik} \tag{1.11}$$

$$\text{s.t.} \quad \sum_{k \in K} x_{ik} = 1, \forall i \in S \tag{1.12}$$

$$x_{ik} \leq y_{jk}, \forall (i,j) \in E, \forall k \in K, \tag{1.13}$$

$$x_{ik} \in \{0,1\},\ y_{jk} \in \{0,1\}, \forall i \in S, \forall j \in C, \forall k \in K, \tag{1.14}$$

Où la fonction objectif (1.11) minimise le nombre d'arêtes ajoutées au sous-graphe bipartite qui induit un sous-graphe bipartite complet. Chaque contrainte (1.12) assure l'affectation d'un noeud i appartenant à S à un seul cluster et, chaque seconde contrainte (1.13) indique que chaque noeud j appartenant à C est affecté au même cluster que son noeud correspondant dans S s'il existe une arête de E qui relie le couple de noeuds (i,j).

Optimisation linéaire en nombres entiers (Integer programming)

La formulation quadratique (*1a - 1d*) peut être résolue avec des solveurs de programmation quadratique, mais le modèle peut être converti en formulation linéaire en définissant $z_{ijk} = x_{ik} y_{ik}$.

L'expression non linéaire du modèle quadratique (équation 1.11) se produit dans la fonction objectif à laquelle les deux variables binaires $x_{ik} y_{jk}$ sont multipliées. Une méthode classique de linéarisation pour un tel modèle non linéaire consiste à remplacer les deux variables de décision x_{ik} et y_{jk} par la nouvelle variable binaire z_{ijk} ($z_{ijk} = x_{ik} y_{ik}$). Dans ce cas, nous pourrions avoir besoin d'ajouter les contraintes linéaires suivantes:

$$z_{ijk} \leq x_{ik} \quad \text{and} \quad z_{ijk} \leq y_{jk} \quad \forall (i,j) \in \bar{E}, \forall k \in K \tag{1.15}$$

$$x_{ik} + y_{jk} \leq 1 + z_{ijk}, \quad \forall (i,j) \in \bar{E}, \forall k \in K \tag{1.16}$$

Cependant, en considérant les contraintes existantes dans le modèle 1.11, l'équation 1.15 est toujours satisfaite. La reformulation formelle, notée P'_{KCMBC}, est décrite comme

suit :

$$\omega(x) = \min \sum_{k\in K} \sum_{(i,j)\in \bar{E}} z_{ijk} \tag{1.17}$$

$$x_{ik} + y_{jk} \leq 1 + z_{ijk}, \quad \forall (i,j) \in \bar{E}, \forall k \in K \tag{1.18}$$

$$\text{s.t.} \sum_{k\in K} x_{ik} = 1, \quad \forall i \in S$$

$$x_{ik} \leq y_{jk}, \quad \forall (i,j) \in E, \forall k \in K \tag{1.19}$$

$$x_{ik}, y_{jk} \in \{0,1\}, \quad \forall i \in S, \forall j \in C, \forall k \in K, \tag{1.20}$$

$$z_{ijk} \in \{0,1\}, \quad \forall (i,j) \in \bar{E}, \forall k \in K,$$

Notez que $\overline{E}$ indique l'ensemble des liens complémentaires entre les services de S et les clients de C. Cela signifie qu'il est composé avec les arêtes qui ne sont pas dans l'ensemble E. On peut observer que P'_{KCMBC} reste difficile à résoudre jusqu'à l'optimalité et les solveurs exacts peuvent converger très lentement (surtout pour les instances de petite et moyenne taille, il est impossible de résoudre les instances plus grandes avec un temps d'exécution raisonnable).

Par conséquent, au lieu de résoudre les versions quadratiques ou linéaires du modèle, on peut commencer par résoudre leurs relaxations linéaires respectives pour construire une solution quasi-optimale par rapport à la structure fournie par n'importe quel solveur de boîte noire ou heuristique sur mesure. Notez que le modèle de relaxation lié au modèle linéaire en nombres entiers est le problème où toutes les variables de décision sont fixées dans l'intervalle continu $[0,1]$. Cependant, ici, la relaxation des variables liées aux x et y reste suffisante pour la méthode proposée, qui induit le modèle relaxé suivant (noté MIP_{kCMBC}) :

$$\overline{z}(x) = \quad \sum_{k\in K} \sum_{(i,j)\in \bar{E}} z_{ijk} \tag{1.21}$$

$$\text{s.t.} \quad x_{ik} + y_{jk} \leq 1 + z_{ijk}, \quad \forall (i,j) \in \bar{E}, \forall k \in K \tag{1.22}$$

$$\sum_{k\in K} x_{ik} = 1, \quad \forall i \in S \tag{1.23}$$

$$x_{ik} \leq y_{jk}, \quad \forall (i,j) \in E, \forall k \in K \tag{1.24}$$

$$0 \leq x_{ik} \leq 1, \quad 0 \leq y_{jk} \leq 1, \quad \forall i \in S, \forall j \in C, \forall k \in K \tag{1.25}$$

$$z_{ijk} \in \{0,1\}, \quad \forall (i,j) \in \bar{E}, \forall k \in K. \tag{1.26}$$

1.7 La procédure d'arrondi de base (PAB)

La procédure d'arrondi de base [2] peut être décrite comme suit :

[2] The basic rounding procedure (BRP)

1. Définir $\mathrm{P}_{reduce} = \mathrm{MIP}_{KCMBC}$.
2. Résoudre P_{reduce} en utilisant le solveur Cplex, et $(\underline{x}, \underline{y}, \underline{z})$ soit la solution atteinte.
3. Fixer la valeur de chaque$\underline{x}_i$, si $\underline{x}_i = 1$ à $\underline{x}$.
4. Sélectionner parmi les variables non entières, la variable $\underline{x}_i$ ayant la plus grande valeur fractionnaire, et l'arrondir à un.
5. Vérifier que les valeurs fixées respectent les contraintes de (11.1 à (11.3) sinon passer à l'étape 4 et sélectionner la plus grande valeur fractionnelle suivante
6. Supprimer les variables fixes du problème actuel et laisser P_{reduce} être le nouveau problème réduit.
7. Répéter les étapes de (2) à (6) jusqu'à fixer tous les éléments du problème original.
8. Récupérer tous les éléments satisfaisant les contraintes de (11.1) à (11.3), et les fixer à 1, 0 sinon.

Enfin, la PAB fournit une solution entière faisable pour P'_{KCMBC}. En raison de la nature avide de PAB et de la complexité des contraintes de précédence de P'_{KCMBC} (Inégalités de type (11.3)), PAB peut fournir des solutions de qualité moyenne. En effet, fixer une variable de décision y_{jk} (service j appartenant aux partitions k) n'induit pas une fixation directe de ses éléments contenants (clients) alors que fixer x_{ik} à un (client i appartenant aux partitions k), impose la fixation de ses services.

Ainsi, la PAB peut éliminer certains éléments en fixant leurs variables de décision correspondantes à un. Par conséquent, la PAB n'est guère utilisée comme une procédure autonome, mais plutôt comme une partie d'une approche computationnelle élaborée.

Akeb, Hifi, and Mounir, 2011 ont proposé une méthode de solution "générique" basée sur la stratégie d'arrondi combinée à une solution exacte restreinte. Ils ont mentionné qu'un solveur exact tel que Cplex n'est pas un moyen approprié pour résoudre des instances de grande taille. En guise d'observation, le fait de fixer certaines variables et de compléter le problème réduit en utilisant la recherche locale peut améliorer la recherche locale, bien que l'amélioration dépende fortement des procédures de recherche locale et de la limite de temps d'exécution imposée. De plus, le processus nécessite une méthode d'accélération. Dans le cas présent, nous avons appliqué des stratégies d'élimination et de fixation pour disperser les solutions. Ceci est appliqué pour découvrir de nouveaux (sous-)espaces de recherche.

Amélioration de la solution actuelle

Pour affiner la qualité d'une solution, quatre voisinages sont considérés. Les deux premiers opérateurs introduisent des déplacements avec un léger degré de liberté tandis

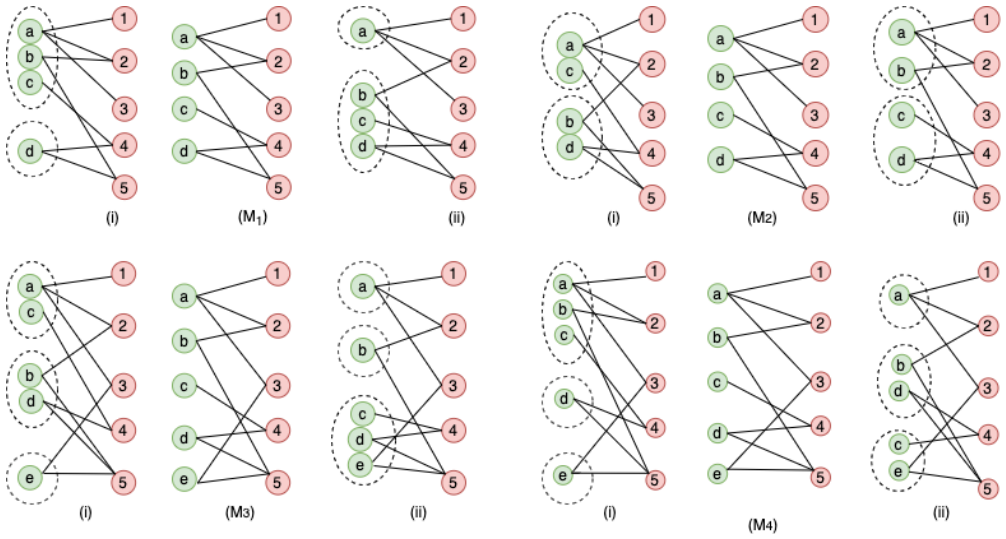


Figure 1.5: Illustration des quatre opérateurs locaux (M_1, M_2, M_3 et M_4) sur deux instances de P_{KCMBC}

que les deux derniers opérateurs tentent d'étendre le voisinage de recherche à plusieurs clusters. Les opérateurs de déplacement sont décrits comme suit :

1. Opérateur M_1: Soit C_i et C_k, $i \neq k$, sont deux clusters, où $|C_i| \geq 2$ et, $y_i^{(1)}, y_i^{(2)} \in C_i$. Déplacer les deux $y_i^{(1)}$ et $y_i^{(2)}$ de C_i à C_k et évaluer la nouvelle solution obtenue.
2. Opérateur M_2: Soit C_i anetd C_k, $i \neq k$, sont deux clusters, où $|C_i| \geq 1,\ C_k| \geq 1$ et, $y_i^{(1)} \in C_i$ et $y_k^{(1)} \in C_k$. Faire un échange entre les deux $y_i^{(1)}$ et $y_k^{(1)}$, c'est-à-dire, déplacer $y_i^{(1)}$ de C_i à C_k et, déplacer $y_k^{(1)}$ de C_k à C_i et évaluer la nouvelle solution obtenue.
3. Opérateur M_3: Soit $C_\ell,\ \ell \in \{1, \ldots, n_K\}$ et (i, j, k) les indices caractérisant trois grappes différentes, de sorte qu'au moins deux grappes contiennent au moins deux services, par exemple C_i et C_j. Imaginons que $y_i^{(1)} \in C_i$ et $y_j^{(1)} \in C_j$; Déplacer les deux $y_i^{(1)}$ (de C_i) et $y_j^{(1)}$ (de C_j) à C_k et, évaluer la nouvelle solution obtenue
4. Opérateur M_4: Soit $C_\ell,\ \ell \in \{1, \ldots, n_K\}$ et (i, j, k) les indices caractérisant trois grappes différentes, de sorte qu'au moins une grappe contient au moins trois services, par exemple C_i. Imaginons que $y_i^{(1)} \in C_i$ et $y_i^{(2)} \in C_i$; soit (i) déplacer $y_i^{(1)}$ de C_i à C_j et $y_i^{(2)}$ de C_i à C_k ou soit, (ii) movdéplacer $y_i^{(2)}$ de C_i à C_j et $y_i^{(1)}$ de C_i à C_k. Évaluer la nouvelle solution fournie.

La figure 1.5 illustre les transformations induites par les quatre opérateurs locaux sur deux instances de P_{KCMBC}.

1. En utilisant les opérateurs M_1 et M_2 sur la première instance − (côté supérieur de la Figure 1.5):

- Premièrement (côté supérieur gauche), l'instance est composée de deux ensembles {a, b, c, d } et {1, 2, 3, 4, 5} désignant respectivement les services et les clients. La solution actuelle (côté supérieur gauche (i)) est composée de 2 clusters C_1 = {a, b, c} et C_2 = {d} : les autres membres liés à chaque cluster sont les clients qui sont connectés aux services avec leurs arêtes correspondantes. Notez que nous négligeons de représenter les clients dans les clusters afin d'obtenir une meilleure visualisation dans les graphes. Le coût de cette solution est la somme des coûts pour chaque cluster ; ensuite, C_1 est évalué à 9 et C_2 à 0. Ainsi, la valeur objective de la solution actuelle est égale à 9. L'utilisation du premier opérateur M_1 induit une nouvelle solution (côté supérieur gauche (ii)) avec une valeur objective égale à 4.
- Deuxièmement (côté supérieur droit), la solution actuelle (côté supérieur droit (i)) est composée de deux clusters $C_1\{a, c\}$ et $C_2 = \{b, d\}$, où sa valeur objective est égale à 6. En appliquant l'opérateur local M_2 sur la solution susmentionnée, on obtient la solution affichée en haut à droite (ii) de la figure, c'est-à-dire une solution dont la valeur de l'objectif est égale à 4 (puisque pour le premier cluster C_1 son coût est égal à 3 tandis que pour C_2 il est égal à 1).

2. Utilisation des opérateurs M_3 et M_4 sur la deuxième instance − (côté inférieur de la Figure 1.5):

   - Le côté inférieur gauche : les deux ensembles de la deuxième instance (services et clients) sont composés de cinq noeuds : {a, b, c, d, e } et {1, 2, 3, 4, 5}, respectivement. La solution courante (côté inférieur gauche (i)) est composée de trois grappes $C_1 = \{a, c\}$,$C_2 = \{b, d\}$ et $C_3 = \{e\}$, avec une valeur objective égale à 6 (sinus le coût de C_1 est égal à 4, celui de C_2 est égal à 2 et celui de C_3 est égal à 0). En appliquant l'opérateur local M_3 à la solution courante, on obtient une nouvelle solution, comme le montre la figure 1.5 (partie inférieure gauche (ii)) ; c'est-à-dire une solution atteignant une valeur objective de 4.
   - La partie inférieure droite (i) : la solution actuelle est maintenant composée de trois clusters $C_1 = \{a, b, c\}$,$C_2 = \{d\}$ et $C_3 = \{e\}$, avec une valeur objective égale à 9. En utilisant le dernier opérateur, M_4 a construit une nouvelle solution telle qu'affichée sur la Figure 1.5 (côté inférieur droit (ii)) ; c'est une solution atteignant une valeur objective de 5.

La stratégie de Hill-climbing

L'application d'un solveur exactpeut pénaliser la complexité et le temps d'exécution moyen. Par conséquent, nous proposons une *stratégie de Hill-climbing* au lieu de considérer le voisinage entier. Cette procédure est dérivée d'une méthode de descente, où les clusters et les services sont choisis aléatoirement. La procédure de Hill-Climbing utilisée est décrite comme suit :

1. Définir la solution actuelle Λ comme l'état initial. Soit S_N (initialement $z(S_N) = +\infty$) la meilleure solution liée au voisinage global (contenant tous les opérateurs).
2. Répéter
   a) Appliquer un opérateur, parmi les quatre opérateurs de M_1 à M_4, à la solution actuelle Λ et faire de la solution Λ' la nouvelle solution.
   Définir $S_N := S'$ (si $z(S_N) > z(S')$).
   b) Mise à jour S avec S' si $z(S) > z(S')$; dans ce cas, redémarrer le processus de recherche avec la nouvelle solution.
3. Jusqu'à ce que le critère d'arrêt soit réalisé.

Bien évidemment, au lieu d'utiliser tous les opérateurs à chaque étape de l'algorithme, nous ne faisons qu'imiter une méthode de descente, où les clusters et les services sont sélectionnés aléatoirement (comme décrit dans l'Algorithme 1).

Algorithm 1 − Une Procédure de Descente (PD)

1. Soit S la solution à disposition (initialement atteinte par PAB)..
   Définir S_N (initialement $z(S_N) = +\infty$) comme la meilleure solution liée au voisinage global (contenant tous les opérateurs).
2. Répéter
   a) Générer aléatoirement ρ à partir de l'intervalle discret $\{1, \ldots, 4\}$.
   b) Appliquer l'opérateur M_ρ à la solution actuelle S et laissez S' être la nouvelle solution fournie.
   c) si $z(S_N) > z(S')$, définir $S_N := S'$, puis actualiser S avec S'.
   d) Relancer le processus de recherche avec la nouvelle solution.
3. Répéter jusqu'à ce que le critère d'arrêt soit réalisé.
4. Sortir avec S_N avec la valeur $z(S_N)$.

Pour assurer une convergence rapide de la procédure ci-dessus, le calcul est limités. La procédure d'arrêt est proposé avec la méthode de descente. C'est-à-dire si après 100 itérations, la méthode ne peut pas améliorer la solution, l'algorithme s'arrête.

Une procédure constructive (complémentaire)

Une procédure constructive (PC) complémentaire fonctionne de la manière suivante (lorsqu'elle est appliquée au problème original) : soit $C = \{c_1, \ldots, c_n\}$ (resp. $S = \{S_1, \ldots, S_m\}$) être l'ensemble des $n = |C|$ clients (resp. $m = |S|$ services), E être l'ensemble des arêtes (liens existants) entre les deux ensembles C et S de $c \in C$ à $s \in S$ (chaque lien existant est noté (c, s)) et $nb_K = |K|$ être le nombre de clusters.

1. Soit $K = \{K_1, \ldots, K_{n_K}\}$ être le départ n_K des clusters vides(clusters dummy. Définir

   a) $S' = S$ et $C' = C$;

   b) $h = 1$ (l'indice h associé au premier cluster).

2. Définir $\delta(s_i),\ s_i \in S'$, par ordre décroissant, où $\delta(s_i)$ désigne le degré du sommet s_i-ième (caractérisant le nombre de clients-nœuds qui lui sont liés) ; Définir $\ell = \arg\max_{s_i \in S'} \Big\{\delta(s_\ell) = \delta(s_1) \mid \delta(s_\ell) \geq \delta(s_i)\Big\}$;

3. Définir $K_h = K_h \cup \{s_\ell\}$, $S' = S' \setminus \{s_\ell\}$, et mettre à jour l'ensemble C' chaque fois que s_ℓ est affecté, c'est-à-dire que de nouveaux ensembles S' et C' sont recalculés.

4. Pour le $n_K - 1$ les clusters restants, du cluster $h = n_K - 1$ jusqu'à $h = 2$ faire:

   a) Assigner un sommet $\ell' \in S'$ au h-ième cluster;

   b) Mettre à jour les deux ensembles S' et C'.

5. Définir $h = 1$;

6. Répéter les étapes (2) et (4) jusqu'à la fixation de tous les services dans les clusters.

En bref, PC, regroupe les services pour créer une solution initiale. Dans un premier temps, PC crée une liste triée des services par ordre décroissant avec leur nombre de connexions correspondant (nous appelons ici "degré du service indiqué" qui caractérise le nombre de clients qui lui sont liés). Il sélectionne ensuite un service qui a le plus de connexions avec les clients de l'étape 2 (c'est-à-dire la tête de la liste triée).

Ensuite, nous retirons le service sélectionné de la liste triée et nous mettons à jour la liste des services et des clients. Nous supposons ici que le cluster sélectionné est affecté au premier cluster fictif.

À l'étape 4, nous assignons les services de la liste triée aux clusters restants et ignorons leurs connexions. Le processus sera répété pour affecter tous les services aux clusters.

1.8 Combinaison de la recherche d'arrondi avec la stratégie de branchement local

Ces dernières années, il a été observé que le branchement local, lorsqu'elle est combinée à la fois avec le solveur de boîte noire *black-box* et les procédures *branch/fix and solve*, comme l'utilisation de la génération de colonnes, a permis de résoudre plusieurs variantes de problèmes d'optimisation combinatoire (Akeb et al., 2011, Cherfi and Hifi, 2009

et Fischetti and Lodi, 2003). Cependant, la difficulté que l'on peut rencontrer lors de l'application de ce type de résolution peut être liée au temps d'exécution qui peut augmenter de manière exponentielle, surtout lorsqu'on s'attaque à des instances de grande taille. Afin de surmonter leur lourdeur, une approche alternative efficace et puissante est proposée, qui, selon nous, est capable de fournir des solutions de haute qualité et en consommant un temps d'exécution "raisonnable".

Un branchement local (local branching, LB) standard peut être directement appliqué pour s'attaquer à $P_{k\text{CMBC}}$, qui peut être décrit par les points suivants :

1. Définir une borne supérieure de départ (pour un problème de minimisation) avec sa configuration ; cela est fourni par l'application de la PAB améliorée avec PD.
2. Initialiser le premier arbre local (à partir du premier nœud de la recherche en trois étapes) en utilisant la configuration de départ précédente.
3. Étape itérative :
   a) Résoudre complètement l'arbre local (ou corriger une limite d'exécution).
   b) Lorsque la recherche locale se termine,
      i. si une meilleure solution réalisable est atteinte, alors créer (à partir de cet arbre local) un nouvel arbre local en considérant la nouvelle solution obtenue comme étant la nouvelle solution de départ (solution de référence);
      ii. sinon, une amélioration est trouvée ; par conséquent, il faut abandonner le branchement local.
   c) Résoudre le reste de l'arbre de recherche
   d) Retourner la meilleure solution trouvée jusqu'à présent.

Le problème de l'achèvement minimum du clustering k a été résolu en utilisant une nouvelle version de l'algorithme basé sur la stratégie d'arrondi. Tout d'abord, une solution de départ a été construite en utilisant une procédure d'arrondi de base qui sélectionne étape par étape un élément fractionnaire du programme mixte. Deuxièmement, une procédure d'amélioration a été conçue, où plusieurs opérateurs locaux ont été considérés. Troisièmement, la stratégie d'abandon et de ré-optimisation a été combinée avec la stratégie de branchement local et, intégrée dans une recherche itérative pour mettre en œuvre une méthode efficace et puissante. La performance de la méthode proposée a été analysée expérimentalement sur un ensemble d'instances de problèmes disponibles dans la littérature, contenant des instances de petite, moyenne et grande taille. La partie expérimentale a montré que la méthode proposée reste compétitive et est capable de fournir de nouvelles bornes par rapport aux meilleures bornes disponibles dans la littérature.

Algorithm 2 − La Stratégie d'Arrondi combinée au Branchement Local (SABL)

Source. Une instance de $P_{k\text{CMBC}}$.

Output. Une meilleure solution $S^{\star}_{k\text{CMBC}}$.

1: Définir $S^{\star}_{k\text{CMBC}} = \emptyset$
2: Appliquer le PAB combiné au PD pour $\text{P}_{k\text{CMBC}}$ et définir $S'_{k\text{CMBC}}$ soit la solution fournie.
3: **repeat**
4: Définir $iter_{local} = 1$ et, définir β soit le paramètre de fixation
5: **repeat**
6: Fixer aléatoirement β variables de $S'_{k\text{CMBC}}$ et, appliquer la procédure complémentaire PC pour $P'_{k\text{CMBC}}$
7: Appliquer LB à la solution d'incubation $S'_{k\text{CMBC}}$
8: Améliorer la solution fournie S_{local} et l'améliorer avec la DP.
9: **if** $(z(S'_{k\text{CMBC}}) > z(S_{local}))$ **then**
10: Actualiser $S'_{k\text{CMBC}}$ avec S_{local}
11: Définir $iter = 1$
12: **else**
13: $iter = iter + 1$
14: **end if**
15: Actualiser $S^{\star}_{k\text{CMBC}}$ avec $S'_{k\text{CMBC}}$, si nécessaire
16: **until** $\Big($la condition locale d'arrêt est réalisée$\Big)$
17: **until** $\Big($la limite de temps est exécutée$\Big)$
18: **return** $S^{\star}_{k\text{CMBC}}$.

1.9 L'algorithme avancé basé sur la stratégie d'arrondi

L'algorithme 3 présente les principales étapes de l'algorithme RSBA(Rounding Strategy-Based Algorithm). L'entrée du RSBA est une instance de P'$_{KCMBC}$ et sa sortie est une solution quasi-optimale $\Lambda^{\star}_{KCMBC}$.

L'algorithme commence par générer une solution de départ (ligne 1) initialisée à un ensemble vide. RSBA est composé de deux boucles internes : (i) la première boucle interne `repeat` de la ligne 6 à la ligne 16 qui est appliquée pour générer une solution de départ courante, qui est améliorée en utilisant la phase d'intensification.

De plus, la condition d'arrêt local définie avec un nombre limité d'itérations basé sur la taille des instances. Alors que la seconde boucle `repeat` de la ligne 4 à la ligne RSBA:line17 sert à diversifier la recherche lorsque la première boucle interne stagne sur un optimum local. La condition globale d'arrêt est atteinte si la boucle globale ne peut pas générer des solutions de meilleure qualité. Les deux boucles sont intégrées dans une boucle globale `repeat` (de la ligne 2 à la ligne 20) servent à répéter l'amélioration et la diffusion sur une nouvelle solution générée par la PAB aléatoire combinée à la procédure constructive PC. La boucle globale est itérée jusqu'à ce que la limite d'exécution soit atteinte. Enfin (ligne 21), RSBA renvoie $\Lambda^{\star}_{KCMBC}$, la meilleure solution trouvée jusqu'à présent.

Le problème de complétion minimale de k-clustering a été résolu en utilisant un

Algorithm 3 The Rounding Strategy-Based Algorithm (RSBA)

Source. Une instance de P'_{KCMBC}.

Output. Une solution réalisable $\Lambda^{\star}_{KCMBC}$.

Étape d'initialisation.
Définir $\Lambda^{\star}_{KCMBC} = \emptyset$
repeat
  Appliquer le PAB combiné au PC pour P'$_{KCMBC}$ et définir Λ'_{KCMBC} soit la solution atteinte.
  repeat
    Définir $iter_{local} = 1$
    repeat
      Appliquer l'approche Hill-Climbing pour Λ'_{KCMBC}
      Définir Λ_{local} soit la nouvelle solution
      if $(z(\Lambda'_{KCMBC}) > z(\Lambda_{local}))$ **then**
        Remplacer Λ'_{KCMBC} avec Λ_{local}
        Définir $iter = 1$
      else
        $iter = iter + 1$
      end if
      Actualiser $\Lambda^{\star}_{KCMBC}$ avec Λ'_{KCMBC}, si nécessaire
    until $\Big($l'arrêt des conditions locales est effectué$\Big)$
    Appliquer la procédure de recherche par dispersion pour $\Lambda^{\star}_{KCMBC}$
    Définir Λ_{local} soit la solution locale atteinte.
  until $\Big($stopping global condition is performed$\Big)$
until $\Big($la limite de temps est exécutée$\Big)$
return $\Lambda^{\star}_{KCMBC}$.

algorithme basé sur une stratégie d'arrondi. Tout d'abord, une solution de départ a été construite en adaptant une procédure d'arrondi de base qui sélectionne étape par étape un élément fractionnaire du programme mixte en nombres entiers. Deuxièmement, une procédure améliorée a été développée, où quatre opérateurs locaux ont été considérés. Troisièmement, la stratégie d'arrondi et son amélioration ont été intégrées dans une recherche itérative pour mettre en œuvre une méthode efficace et puissante. La performance de toutes les procédures de solution proposées a été analysée expérimentalement sur un ensemble d'instances de problèmes disponibles dans la littérature, contenant des instances de taille moyenne et grande. La partie expérimentale a montré que la méthode proposée reste compétitive et est capable de fournir de nouvelles limites par rapport aux meilleures limites disponibles dans la littérature.

1.10 Conclusion

Cette thèse vise à traiter un complexité rencontrée pour optimiser un composite végétal et à assurer la durabilité des sites d'approvisionnement, de production et de commercialisation.

Le premier objectif est atteint en optimisant les caractéristiques du matériau composite à l'aide des méthodes d'intelligence artificielle dans la première partie. La seconde est proposée dans la deuxième partie de cette étude. Elle fusionne le ou les sites de la chaîne d'approvisionnement en utilisant la méthode du K-clustering. Différentes méthodes de solution d'optimisation sont proposées.

L'approche transversale appliquée permet le couplage des compétences présentes au sein de l'unité de recherche EPROAD. Les méthodes proposées sont issues du domaine de l'intelligence artificielle, de la modélisation discrète issue des mathématiques appliquées, de l'analyse de sensibilité, et du domaine du génie des procédés pour le développement de méthodes coopératives intelligentes.

Chapter 2

General introduction (*in english*)

Brief table of contents

2.1 Introduction

The economical and environmental effects of the constructions are the two important factors that must be considered in civil engineering. The use of vegetal resources in the composite materials is an alternative solution to the exploitation of fossil resources.

The best way to deal with the enviro-economical aspects is to upgrade the agricultural waste in construction. The reason is first: reuse the agricultural waste will decrease the magnitude of the disposal problem (Al-Mohamadawi, Benhabib, Dheilly, and Goullieux, 2016, Al-Mohamadawi et al., 2020), and second: the agricultural waste is available and low-cost (Khazma, Goullieux, Dheilly, and Quéneudec, 2012; Mahieu, Lenormand, Leblanc, and Vivet, 2015).

According to FAOSTAT, 2020[1], flax fiber production in France is 77% and 78% of the European flax fiber[2] production in 2017 and 2018 respectively. Flax shives[3] are low-cost, available, light, and have a honeycomb structure. These characteristics make them attractive lignocellulosic aggregates in the concrete field(Khazma, Goullieux, Dheilly, and Quéneudec, 2007; Khazma et al., 2012).

Several studies have been conducted to evaluate the use of vegetal materials in the cement matrix [4]including wood shaves (Bederina et al., 2009), fiber of coconut shell (Olorunnisola, 2009), and flax shives (Dupré, 2005). Although, most of the studies face undesirable characteristics such as high water absorption of the vegetal aggregate. High water absorption leads to high dimensional variations and low mechanical strengths of the concrete[5] (Khazma, Goullieux, Dheilly, Laidoudi, and Queneudec, 2011).

Therefore, a treatment seems necessary before the incorporation of the lignocellulosic aggregates into a cement matrix. The coating is used in most of the studies as an environmentally friendly treatment with low energy consumption. Coating [6] with mineral or organic substances such as cement and lime will isolate the vegetal aggregate from the cement matrix. It will reduce the water absorption, improve the mechanical strength, and reduce their shrinkage (Bederina et al., 2009; Khazma et al., 2011).

A study of the composite compressive strength with a case of a composite formulated with coated aggregates is presented in part I. Sensitive characteristics of the studied vegetal composite require a reliable and sustainable distribution and supply chain. In the second part (part II), we propose consolidating the supply facilities with the K-clustering problem.

This thesis consists of two parts. The first part presents a study of the composite compressive and flexural strengths using artificial intelligence methods. The second part deals with the optimization of the supply chain by consolidating the supply facilities with combinatorial optimization.

[1] Food and agriculture organization corporate statistical database
[2] Flax fiber is extracted from the bast or skin of the stem of flax plant
[3] Biological waste formed during the processing of flax straw
[4] Cement is the binder in concrete. The cement matrix is formed during the hydration process and binds the aggregates together (Petrovic, Vale, and Zari, 2017).
[5] Concrete is a composite material composed of fine and coarse aggregate bonded together with a fluid cement (cement paste) that hardens (cures) over time
[6] A thin layer or covering of substance

2.2 Study of the composite compressive strength: a case of a vegetal composite

The first part of this study concerns the optimization of a vegetal composite. It includes an introduction chapter 3 dealing with machine learning methods in 3.2 and the presentation of the experimental dataset of the vegetal composite 3.3.

Prediction with regression methods are presented in chapter4, methodology (4.3) and an example for the prediction problems (4.3), regression with two (4.4) and more (4.5) dimensions are also presented in this chapter.

Chapter 5 presents a methodology to predict the compressive strength of a vegetal composite using the gradient descent method. An introduction to the regression models (5.1) and the gradient descent method (5.2) are presented. The principal steps to apply the optimization method are explained in 5.3. The results of the computational analysis, a sensitivity analysis on the learning rate and the number of iterations is presented in 5.4 and the algorithm is evaluated in 5.5.

By considering the non-linear effect of the parameters, the complexity, and interchangeability of the data, we propose artificial neural networks to predict the composite compressive and flexural strengths in chapter 6. Artificial neural networks (6.1) and an approach to train the data is presented in 6.2. Several neural network models were designed and compared in 6.3.

Due to the importance of the compressive strength test, the last chapter 7 of part I, shows a different view of analyzing the vegetal composite by predicting the parameters of the test. The two gradient descent methods (the batch gradient descent and the stochastic gradient descent) are compared in 7.3. The results are reported in 7.4.

The experimental data

Despite the advantages of coating the vegetal aggregate with mineral or organic substances, there can be a nonlinear relation between the concrete components. Hence, it is more complicated to find a reliable and robust method that can predict the compressive strength of the composite based on the components of the concrete mixtures.

The compressive composite strength test (CCST) is a conventional test that takes 28 days to describe the essential features of the concrete mixtures (de Siqueira Tango, 1998). It is the most important test for quality control but by itself is time-consuming and therefore expensive for industries.

The experiments are conducted in the IMaP team of the EPROAD laboratory on the coating of flax straw before their incorporation into a cement matrix.

During the experiments two coating substances (CS) lime and cement are selected and a ratio of the quantity of water to the quantity of a CS and also a ratio of the quantity of straw to the quantity of a CS are varied. Characteristics of the shives after coating and characteristics of the resulted flax shive composites are measured.

Measured characteristics for the straws are bulk density, water absorption capacity, and decrease in water absorption capacity (comparison with untreated straws) and for the vegetal composites, the measured characteristics are their flexural strength, compressive strength, bulk density, dimensional variation on drying, and extreme dimensional variation of composites.

The objective of the experiment is to have high mechanical properties and low dimensional variation on the formulated concrete. The practical experiments are time-consuming and expensive as they need to be repeated. Thus, it is necessary to define mathematical formulations that predict the input parameters.

Generally, in spite of the reliability of the CST, considering that it will take 28 days, it is not a cost and time-efficient technic. Therefore, modeling technics and mathematical modeling seems to be a promising way (Baykasoğlu, Dereli, and Tanış, 2004; Tsivilis and Parissakis, 1995).

Optimization of the vegetal composite

The optimization of the vegetal composite is a multi-objective/multi-criteria problem, and it needs several experimental tests. Hence, numerical methods seem necessary. Furthermore, it is possible to predict the consequences of a change in the raw materials or manufacturing parameters on the characteristics of materials.

The development of the vegetal composite requires taking into account the time and space availability of bio-sourced raw materials, their interchangeability, and their consequences on the resulting functional characteristics.

The optimization of the vegetal composite faces with a variety of the available data, the complexity to establish cause and effect relationship, and the mandatory handling of the unexpected events (such as disaster, crises, break down, ...). Thus, a reliable and sustainable system is required in order to produce the vegetal composite with constant efficiency.

It seems that artificial intelligence methods can be used to establish models in order to constantly and dynamically understand the data and mastery in producing the targeted materials.

Artificial intelligence methods should respond to (i) the diversity of the available data, (ii) the difficulty of establishing cause-and-effect relationships, and (iii) the particular

need to ensure a reliable and sustainable production in order to keep the constant efficiency. These methods improve the understanding and control of the production related to the concerned materials.

Considering this uncertainty and changeability of the data, we applied machine learning and artificial intelligence to predict the parameters of the experiments. Artificial intelligence can predict the parameters dynamically. A model can adapt itself based on the training data. Predictions are dynamic and the results are data-oriented.

Studied models for the optimization of vegetal composite

The optimization of the vegetal composite is studied with three aspects: (i) to predict its compressive strength and (ii) to predict its flexural strength and (iii) a simulation model to predict the parameters of the composite compressive strength (CCS) test.

To overcome the mentioned problems, artificial intelligence and machine learning methods are suggested as a solution that learns from the old data in order to converge towards better local optima. The proposed scenarios are summarized in Table 2.1. These scenarios are created based on the experimental data.

Table 2.1: Summary of the scenarios

Scenario	Data	Model	Method	Target parameter	Objective
S_1	123	1	-GD -ANN	compressive strength of the vegetal composite	predict the compressive strength given the input parameters
S_2	123	1	-ANN	flexural strength of the vegetal composite	predict the flexural strength given the input parameters
S_3	60	4	-GD -SGD	the ratio of water per two coating substances lime (and cement) the ratio of shive per two coating substances lime (and cement)	simulate the cement compressive strength test

The first row of Table 2.1 summarizes the first scenario that aimed to predict the compressive strength of the vegetal composite. The first column presents the noted scenario S_1. In this scenario (mentioned in the second column), we used a total of 123 data of the experiments. This data includes 60 experiments on lime, 60 experiments on cement, and 3 experiments on composite with not coated shives.

The rest of the columns represent the number of the models created with scenario S_1, the methods used in the stated scenario, the target parameter, and the objective of this scenario. For scenario, S_1, the two methods: the gradient descent (GD) and the

artificial neural networks (ANNs) are applied to predict the compressive strength with the given input parameters.

The second scenario (S_2) is similar to the first scenario. It aims to predict the flexural strength of the vegetal composite. In scenario S_2, we used the same 123 experimental data used in scenario (S1). In this scenario, ANNs method is applied to predict the flexural strength given the input parameters.

Considering the importance of the CCS test, we decided to simulate the experiments by proposing mathematical formulations. The achieved equations from GD will simulate the parameters of the CCS test. For this purpose, scenario S_3 was applied on the total of 60 experiments on lime and the total of 60 experiments on cement individually.

A total of four individual models was developed in this scenario (S_3). In which, the two of four models is to predict the ratio of water and the ratio of shives per coating substance for lime and the two other models to predict the ratio of water and the ratio of shives per coating substance for cement. The two algorithms, the Stochastic Gradient Descent (SGD) and the GD, are applied to the data. This scenario was aimed to simulate the CCST. The proposed models are able to assist the experts performing the CCST to minimize the required number of experimental tests. The equations can assist the operator to select suitable ratios of water and shives for each coating substance individually.

2.3 Consolidating the supply facilities by combinatorial optimization: K-clustering

The second part of this thesis proposes consolidating the supply facilities to ensure continuous production and distribution of the studied vegetal coated composite. In this part, the k-clustering minimum completion problem is studied to consolidate the supply facilities.

Chapter 8 introduces the importance of a sustainable supply chain and reliable supply chain of the vegetal composites. Consolidating the supply facilities and the regrouping problem is discussed in 8.1. The operational research and combinatorial optimization is introduced in 8.2.

K-clustering minimum bi-clique completion problem is introduced in chapter 9 with an introduction and a review of the other studies (9.1), applications and introduction to the benchmark dataset (9.2). Mathematical formulations and the applied linearization method with an example is presented in 9.3.

Chapter 10, enhanced the rounding method by incorporating it with the local branching. The augmented version of the rounding search (10.2) and the main procedures

of combining the rounding search with local branching strategy (10.2) are presented in this chapter. The benchmark dataset was used to evaluate the model. The effect of the local branching and the behavior of this model is shown in 10.3.

Chapter 11 propose an advanced rounding strategy-based algorithm for this problem. This chapter includes an introduction to the main method (11.1), powerful enhancing solution methods (11.2). The proposed solution is evaluated through statistical analysis in section 11.4. Presented comparisons between the other methods (11.4) shows the superiority of the proposed model.

Sustainable development of supply chain facilities

The development of the vegetal composite requires taking into account the specific parameters of bio-sourced raw materials such as temporal availability, interchangeability, and the consequences on the resulting functional properties. The optimization of vegetal composites can be viewed as a complex problem related to different domains such as biology, physico-chemistry, and process engineering.

The sustainable optimization and production of vegetal materials also require the localization, centralization, and consolidation of the supply chain sites. The supply chain problems consist of localizing the production sites, routing, scheduling, storage of raw materials, final distribution and marketing.

Dealing with these constraints, it is important for the companies to build a reliable business model along with a competitive purchasing and marketing strategy. The modeling of the information flows related to the problems of the location and planning of a production unit on an urban and rural transportation network, generates very complex data. Herein, the combinatorial optimization methods can be efficient and reactive in the choice of implantation points for the vegetal composites production sites.

Localization and consolidation issues are in the area of operations research and decision-making. On the one hand, a localization problem leads to a single and multi-criteria modeling. On the other hand, extending the model to a robust variant can be attempted according to the scarcity of the materials needed to manufacture the vegetal composite and face with the criteria related to the decision-makers.

An element that can be considered to increase the robustness of an existing supply chain system is to consider the possibility of consolidating the supply system. Considering merging the supply centers is an element that can increase the sustainability and robustness of the system in case of losing a center due to an emergency shut down, disaster, or economic reasons.

Consolidating the supply facilities

The production of the vegetal composites is highly sensitive. The supply materials have to be available fully in a certain place (x) at a certain time (t). Since the vegetal materials are highly sensitive, it is highly important to consider a backup strategy to choose the supply chain centers.

Consolidating the supply, production, distribution, and retailing centers is an alternative solution to reduce the inventory and retail costs (Teo et al., 2001; van der Vlist and Broekmeulen, 2006).

Regrouping, consolidating, or clustering are referred to the act of merging two or more sits (Koch and Wäscher, 2016). It is the act of reducing the number of centers to smaller numbers (or clusters) by merging the existing centers. In order to keep the continuous efficiency in the supply and production chains, each site has to serve the whole services that used to be served by the replaced sites (Mansour et al., 2004).

Regrouping examples can be not only in supply centers but also in educational sites to regroup public schools into a smaller number of existing schools, warehouses, telecommunication systems, shops, distribution centers, etc.

Supply chain consists of three sections: (i) supply center(s), (ii) production center(s), (ii) and distribution center(s). The regrouping problem can be seen in any part of this chain. The k-clustering problem can be defined in any of the two parts of the supply chain that are in direct relations.

K-clustering minimum completion problem

Given a bipartite graph G(S, C, E), the k-Clustering Minimum Biclique Completion Problem (kCMBC) consists in finding k bipartite clusters (sub graphs) such that:

- $\forall i \in S$; S denotes the first set of vertices of G, it appears once in one cluster,
- $\forall j \in C$; C denotes the second set of vertices of G, it appears in each cluster in which at least one of its neighbors appears,
- the total number of edges to be added to each sub graph to become a complete bipartite sub graph is minimized

Such a problem is also described as a real-world application related to the telecommunication domain. Indeed, in order to bound some given channels in multicast transmissions, according to a set of clients' demands related to services, k multicast sessions should be built for including independent demands. On the one hand, a service is contained in a

single multicast session and, on the other hand, a client may appear in several sessions (Hifi and Sadeghsa, 2020a).

In the literature, deterministic and stochastic heuristics are often used to approximately solve the clustering problem. Establishing an adaptive model to the data with heuristic and meta-heuristic models in combinatorial optimizations will ensure the sustainability and reliability of the production of vegetal composites. Moreover, convergence results will lead to a reliable business model and a competitive purchasing and marketing strategy.

2.4 Conclustion

This thesis aims to deal with the complexity encountered to optimize vegetal composite and to ensure the sustainability of the supply, production, and marketing sits.

The former is achieved by optimizing the characteristics of the composite material using the artificial intelligence methods in the first part. The latter is proposed in the second part of this study. It merges the supply chain site(s) using the K-clustering method. Different optimization solution methods are proposed.

The applied transversal approach allows the coupling of skills presents within the research unit EPROAD. The proposed methods are from the field of artificial intelligence, discrete modeling resulting from applied mathematics, sensitivity analysis, and the field of process engineering for the development of intelligent cooperative methods.

Part I

Study of the composite compressive strength: case of a vegetal composite

In this part, we consider optimizing vegetal composite from two aspects using the machine learning methods. We first introduce the basic concepts of machine learning, its terminology, and some of its applications in chapter 3. We used data from an experimental study that is performed in the laboratory of *EPROAD* and is conducted by the *IMaP* team. The data are measured through a number of experiments with composite compressive strength test on composites elaborated with coated flax shives. We will discuss the data in 3.

In one hand, we propose a prediction method using regression on the dataset to predict the compressive strength of a vegetal composite in chapter 5. The prediction methodology is explained in this chapter. Later in chapter 6, we propose an algorithm based on artificial neural networks to predict the compressive and flexural strengths using the dataset.

In the other hand, we propose a method to simulate the compressive composite strength (CCS) test and to predict the ratios (shive/coating substance, water/coating substance) of the input coating parameters using the gradient descent method in chapter 7. This study is done to decrease the number of tests that is required to perform the CCS tests. Considering the fact that each CCS test is performed after a 28-day, this study can be used to minimize the number of experiments and costs of repeating the experimental CCS tests.

Chapter 3

Introduction

Brief table of contents

In this chapter, we present an introduction to machine learning. It includes the basics of machine learning, an introduction to different types of learning methods (3.2), their applications, definitions, and terminology of machine learning (3.2). The experimental data used in this study are described in 3.3.

3.1 Overview

The economical and environmental effects of the constructions are two important factors that must be considered in civil engineering. The best way to deal with both enviro-economical aspects is to upgrade the agricultural waste in construction. The reason is first: reuse the agricultural waste will decrease the magnitude of the disposal problem (Al-Mohamadawi et al., 2016, Al-Mohamadawi et al., 2020), and second: the agricultural waste is available and low-cost (Khazma et al., 2012; Mahieu et al., 2015).

According to FAOSTAT, 2020 [1], flax fiber production in France is 77% and 78% of the European flax fiber[2] production in 2017 and 2018 respectively. Flax shives[3] are low-cost, available, light, and have a honeycomb structure. These characteristics make them attractive lignocellulosic aggregates in the concrete (Khazma et al., 2007; Khazma et al., 2012). Several studies have been conducted to evaluate the use of vegetal materials in the cement matrix including wood shaves (Bederina et al., 2009), fiber of coconut shell (Olorunnisola, 2009), and flax shives (Dupré, 2005). Although, most of the studies face undesirable characteristics such as high water absorption of the vegetal. High water absorption leads to high dimensional variations and low mechanical strengths of the concrete[4] (Khazma et al., 2011).

Thus, a treatment seems necessary before the incorporation of the lignocellulosic aggregates into a cement matrix. Coating is used in most of the studies as an environmentally-friendly treatment with low energy consumption. Coating [5] with mineral or organic substances such as cement and lime will isolate the vegetal aggregate from the cement matrix. Thus, it will reduce the water absorption, improve the mechanical strength, and reduce their shrinkage (Bederina et al., 2009; Khazma et al., 2011).

Despite the advantages of coating the vegetal aggregate with mineral or organic substances, there can be a nonlinear relation between the concrete components. Hence, it is more complicated to find a reliable and robust method that can predict the compressive strength of the composite based on the components of the concrete mixtures.

Compressive cement strength (CCS) test is a conventional test that takes 28 days to describe the essential features of the concrete mixtures (de Siqueira Tango, 1998). It is the most important test for quality control but by itself is a time-consuming experiment and therefore expensive for industries.

Generally, there are two ways to perform a compressive strength test (CST), one is the experimental test and the second is predicting the variables using soft computing

[1] Food and agriculture organization corporate statistical database
[2] Flax fiber is extracted from the bast or skin of the stem of flax plant
[3] Biological waste formed during the processing of flax straw
[4] Concrete is a composite material composed of fine and coarse aggregate bonded together with a fluid cement (cement paste) that hardens (cures) over time
[5] A thin layer or covering of substance

methods.

In this study, the experiment is conducted in the IMaP team of the EPROAD laboratory on the coating of flax straw before their incorporation into a cement matrix[6]. Two coating substances (CS) lime and cement are selected. During the experiment a ratio of the quantity of water to the quantity of a CS and also a ratio of the quantity of straw to the quantity of a CS are varied. Characteristics of the shives after coating and characteristics of the resulted flax shive composites are measured.

Measured characteristics for the straws are bulk density, water absorption capacity, and decrease in water absorption capacity (comparison with untreated straws) and for the vegetal composites, the measured characteristics are their flexural strength, compressive strength, bulk density, dimensional variation on drying, and extreme dimensional variation of composites. More information about the dataset is discussed in 4.2.

The objective of the experiment is to have high mechanical properties and low dimensional variation on the formulated concrete.

During practical sessions, we noticed that this experiment is time-consuming and expensive as it needs to be repeated. Thus, we decided to define a mathematical formulation that predicts the input parameters of this study. We will discuss the two studied aspects in chapter 7.

Generally, in spite of the reliability of CST, considering that it will take 28 days, it is not a cost and time-efficient technique. Therefore, modeling techniques and mathematical modeling seem to be a promising way (Baykasoğlu et al., 2004; Tsivilis and Parissakis, 1995).

3.2 Introduction to machine learning

There are several definitions of machine learning in the preliminary studies. In general, machine learning is a computational science that uses past information to improve performance or to make accurate predictions. The process of using experiences from past information is called *"learning"*. Since the learning process feeds from the training data, the efficiency of a machine learning model highly depends on the input parameters. Since machine learning is related to data analysis science and statistics, it is also grouped in the data-driven methods.

In the twenties (20th) century, Arthur Samuel, a pioneer in machine learning, wrote a checkers playing program. His program was able to learn from the several board compositions, which board positions tend to win and which board positions tend to lose.

[6]The cement matrix, primarily determines the properties of a multi-phase material (here concrete) that confers resistance to stresses

As a result, a machine that has more passion than a human to repeat a process (in this case learning from a board match), could be trained to play chess even better than a human. He believed teaching computers to play games was very fruitful for developing tactics appropriate to general problems. He chose checkers as it is relatively simple though has a depth of strategy (To know more about Arthur Samuel please refer to McCarthy and Feigenbaum, 1990 or Wikipedia, 2021 that is written to his memoriam).

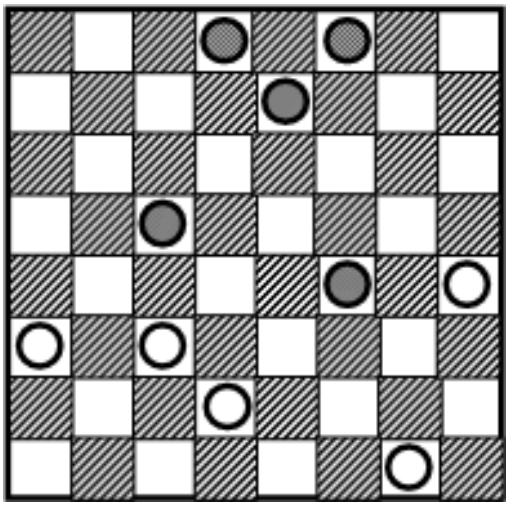


Figure 3.1: Checkers playing program: an example of machine learning

The term "machine learning" is coined by Arthur Samuel, 1959 as "*a field of study that gives computers the ability to learn without being explicitly programmed*".

Figure 3.1 represents the checkers playing program which is an early example of machine learning that Arthur Samuel initially programmed. Nowadays, more and more machine learning applications take a part in our daily lives. Now that I am typing my thesis dissertation, word suggestion system from the editor I am using is an application of machine learning. Image recognition in mobile phones, many advertisements on google, and spam recognition for emails are some of the daily applications of machine learning.

Mohri, Rostamizadeh, and Talwalkar, 2018 stated that machine learning can also be seen as *designing algorithms* to make accurate predictions. In other computer science methods, critical measures to compare the quality of algorithms are their computational time and space. However, machine learning algorithms additionally depend on "sample training complexity". It means their quality is also related to the complexity of the concept and the size of the training samples.

Which problems can be tackled with machine learning methods?

The variety of problems that can be tackled using machine learning algorithms are vast. Practical applications of machine learning are expanding to several areas. Here we mention some examples of the kinds of problems that can be studied with machine learning:

- Spam classification or document classification or any problem that lies in assigning a topic to a document or a text, image recognition, handwriting, or digit recognition
- Clustering or partition large set of data into homogeneous groups
- Regression or prediction of a real value such as prediction of the stuck value

The study presented herein is based on the comprehensive strength of a vegetal composite from a composite compressive strength test. The compressive strength of a vegetal composite is a positive value. Prediction using the studied data can be grouped within the regression or prediction problems. In this study, we will discuss the regression problem.

Machine learning algorithms

As its most basic, machine learning algorithms feed input data, analyze them to predict output values within a predefined range. It can be said that they learn from the input data to predict and analyze new data by developing "intelligence" over time.

Depending on the input data type, there are two main machine learning algorithms: supervised learning and unsupervised learning.

-*Supervised learning:* in supervised learning, the algorithm learns from the examples. The dataset includes both inputs and their corresponding output. The machine will try to find connections between the inputs and output by identifying patterns in the data. It will try to find a way to connect the input parameters to predict output while the operator knows the correct answer to the problem. The machine is taught by examples until it achieves to make predictions with a certain level of accuracy. For example, given a dataset of patients diagnosed having certain cancer or not, a machine can learn to classify either a patient has cancer or not.

-*Unsupervised learning:* unsupervised learning observes the data to identify their correlations and relationships automatically. The data are not labeled and there is no given output or answer keys to provide instructions. In unsupervised learning, the algorithm tried to organize or rearrange the data such as grouping the data into clusters.

For example, grouping a set of news with a similar story from a given list of news found in a journal.

In the history of machine learning, there are other scenarios such as semi-supervised learning, reinforcement learning, and online learning. The key factor that differs between the mentioned methods is the type of available training data.

Data used in this study are labeled and by giving the input features the output value (the value of CCS for each experiment) is known. Also, a penalty for inaccurate predictions can be calculated with the difference between the predicted and true values. This information assigns this study with the supervised learning methods.

Terminology of machine learning

We adopted a termionology that is used in machine learning studies. Here is a list of definitions that are used in machine learning algorithms:

- Example: Each instance of the data is an example. In our study, each example represents data of an experiment that is conducted in the laboratory.
- Training set: A percentage of the data that is fed to train the algorithm is called *"training set"*.
- Test set: A percentage of the data that is used to test the algorithm is called *"test set"*.
- Features or inputs: Set of attributes or factors taken from an example. Features are the parameters in the dataset that are in relation to the target value.
- Target value: The value that is assigned to an example.
- Cost or loss function: A function that measures the difference between the predicted values and true values.
- Hypothesis: A function that maps features of an example to the target value.
- Hyperparameters: Parameters that need to be tuned as the input of the learning algorithm such as learning rate and the number of iterations in the training process.

3.3 Introduction to the dataset of the vegetal composite

Cement is a preliminary and essential material that is used in civil engineering to construct buildings and houses. For civil engineers, one of the most important features of a concrete is its compressive strength.

What is compressive strength?

Compressive strength is the resistance of a material to breaking under compression.

Many studies have shown that the use of agricultural waste not only will elaborate the features of the vegetal composites but also will solve an environmental problem (Al-Mohamadawi et al., 2020).

The reuse of flax shives, an agricultural waste, in the elaboration of vegetal composites offers an interesting alternative to meet the challenge of their elimination and solve an environmental problem. Due to their honeycomb structure the flax shives can be used as lightweight aggregates inside cementitious composites and provide insulating properties.

However this vegetal waste is hygroscopic and can release water-soluble molecules responsible for setting retardation of the cement matrix, large dimensional variations and low mechanical strengths for the composite. These drawbacks can be reduced thanks to coating processes using cement, lime, linseed oil or paraffin as coating substances (as mentioned in Khazma, Goullieux, Dheilly, Rougier, and Quéneudec, 2014; Al-Mohamadawi et al., 2016, 2020; Monreal, Mboumba-Mamboundou, Dheilly, and Quéneudec, 2011). Compared to the raw shives composite, the composites elaborated with coated shives exhibited compressive strength 8 to 27-fold higher, drying shrinkage two to threefold lower and belong to the insulating class of the lightweight concrete.

The composites studied are developed by adding flax shives, coated or not coated, to Ordinary Portland Cement (OPC) and water. Previous works have allowed the determination of the water to OPC ratio (W/C) and shives to OPC (S/C) ratio in order to obtain a non-friable composite. The ratios were applied herein: $W/C = 0.5$ (mass ratio) and $S/C = 4$ (volume ratio).

In an experimental study conducted in *IMaP*, an agricultural waste has been coated with mineral substances. The compressive strength of vegetal composite is examined by varying the coating parameters, the type of mineral substance, the ratio of the shives per coating substance, and the ratio of water per coating substance.

A comparison between lime and cement: the two coating substances

To clarify the difference between the two studied coated substances, lime, and cement, we compared them by their hardness speed, brittleness, prone to cracking, mendability, workability, vapor barrier, and environmental aspects. A comparison between the mentioned features are summarized in Table 3.1. The comparison between characteristics of lime and cement can be also found in Li, Ni, and Yi, 2019; Pu, Zhu, Song, et al., 2020; Pu, Zhu, Zhao, et al., 2020 and Purton, 1970.

Table 3.1: Comparison of specific features between lime and cement as coating substances

Properties \ Coating substance	lime	cement
hardness speed	slower than cement	very quick
brittleness	brittle but less than cement	brittle
prone to cracking	less than cement	prone to cracking and may eventually require repair
mendability	cracked areas can absorb carbon dioxide and mend over time	no
workability	workable	too strong for some applications
vapor barrier	allowing vapors to pass through(can reduce moisture)	creates a waterproof barrier
environmental concerns	environmentally friendly	carbon dioxide is released during its production

The dataset of the vegetal composite

Dataset used in this study is given from a total of 123 experiments. Where total of 60 experiments are conducted on lime as coating substance and by varying ratio of shives per lime and ratio of water per lime.

Total of 60 experiments are conducted on cement as coating substance and by varying ratio of shives per cement and ratio of water per cement, and total of 3 experiments where composites are elaborated with non-coated shives.

The ratio of shives per coating substance is in range $\{1/3, 1/2, 2/3, 1, 2\}$ and for water per coating substance is in range $\{1/2, 3/4, 1, 2\}$.

In the experimental studies, the characteristics of the shives after coating and the characteristics of the flax shive composites are measured. Figure 3.2(a) visualizes a sample of a vegetal composite (Al-Mohamadawi et al., 2020) and figure 3.2 (b) shows a sample of flax shives (Barnat-Hunek et al., 2017).

For each experiment with or without a coating substance, the water absorbance capacity and the bulk density of shives are measured.

Certain characteristics of flax shive composites are measured for both non-coated and coated shives. These characteristics are: bending strength, compressive strength,

(a) Cement composite (b) Flax shives

Figure 3.2: A sample of (a) a vegetal composite (Al-Mohamadawi, Benhabib, Dheilly, and Goullieux, 2020) and (b) flax shives (Barnat-Hunek, Smarzewski, and Brzyski, 2017).

bulk density, drying shrinkage, and extreme dimensional variations.

Experiments with lime and cement as coating substanes are composed of 20 combinations of 5 groups of shives per coating substance and 4 groups of water per coating substance. Each of these experiments are repeated 3 times to the sum of 60 experiment for each coating substance.

Table 3.2: Introduction to the dataset

Coating substance	Nb.	Elements	Shive after coating		Flax shive composite				
			water absorbance %	bulk density kg/m^3	bending strength MP_a	compressive strength MP_a	bulk density kg/m^3	drying shrinkage mm/m	extrem dimensional variation mm/m
All data	123	AVG	90.32	203.59	0.9	2.6	747.6	5.23	4.24
		STD	28.25	59.12	0.65	2.1	129.17	1.47	1.76
		Min	46.81	96.5	0.14	0.31	556.09	0	1.31
		Max	222.2	340.75	3.28	9.56	1099.49	9.24	9.24
Lime	60	AVG	87.13	200.88	0.93	2.97	737.61	4.93	3.75
		STD	17.15	48.86	0.63	2.21	111.5	1.42	1.97
		Min	47.93	129.7	0.14	0.63	583.63	2.4	1.31
		Max	136.4	298.5	2.57	9.56	1047.93	8.17	8.65
Cement	60	AVG	87.29	211.48	0.92	2.33	766.01	5.36	4.5
		STD	23.44	64.98	0.67	1.93	142.03	1.32	1.1
		Min	46.81	130.6	0.16	0.31	556.09	0	3.14
		Max	132.54	340.75	3.28	7.41	1099.49	7.66	6.76
NTS^*	3	AVG	214.55	99.83	0.17	0.4	579.02	8.63	8.76
		STD	6.68	3.43	0.01	0.06	11.7	0.59	0.43
		Min	209.85	96.5	0.16	0.34	566.72	8.07	8.4
		Max	222.2	103.35	0.19	0.46	590	9.24	9.24

Table 3.2, shows the data with statistical elements including average(AVG), standard deviation(STD), the minimum(Min) and the maximum(Max) value of each feature. The data are grouped based on a material that flax shives are coated within 4 categories. The groups are: flax shives coated with lime, flax shives coated with cement, none treated shives(NTS), and a group with all the data. The experimental true data are reported in the appendices A 1.

3.4 Conclusion

In this chapter, we introduced the concept of machine learning, along with the two main machine learning types: supervised learning and unsupervised learning. The applications of machine learning, definitions, and terminology of machine learning algorithms are discussed.

We also mentioned the importance of cement in civil engineering and the advantages of reusing agricultural waste for the environment and the sustainability. The experimental data used in this study are described with their statistical characteristics.

Chapter 4

Prediction with regression models

Brief table of contents

Machine learning methods can be used in prediction models. In this chapter, we will describe a methodology to apply regression models on the experimental dataset. Linear models with two and three-dimensional regression methods are presented.

4.1 Overview

Machine learning methods can be grouped into two main types: supervised learning and unsupervised learning.

Supervised learning refers to the labeled data. For example, by labeling the data of a

population into two groups of patients contaminated with a virus and not contaminated people, we can predict the possibility of a person being contaminated or not. A way to recognize if we can treat a dataset with supervised learning methods is labeling them.

In other words, supervise learning methods can predict a parameter that we already have its real value. Thus, we can evaluate the quality of the predictions by the difference between the predicted and the true value.

Conversely, in unsupervised learning, the output value is not known in the dataset. The method infers the intrinsic patterns from a dataset without any reference label.

We recall that in this study, we use the data that we described in 3.3. In this dataset the target value for each example is available. Thus, we can calculate a penalty for the predictions using the true value by defining a supervised learning scenario on our labeled dataset.

Generally, there are two types of supervised learning. If it aims to predict continuous values, the algorithm is called *"regression"* problem and if it aims to predict a discrete value, it is called a *"classification"* problem.

4.2 Data description

Machine learning algorithms are data-oriented methods. As we described earlier, the data used in this study are composed of 123 experiments, in our model each experiment is an *example* of the model. We described each experiment with a vector of *features* and a *target value*. For each experiment, there are a total of 6 features that are defined as coating substance, the ratio of water per coating substance, the ratio of shives per coating substance, and characteristics of the composite including bulk density, drying shrinkage, and extreme dimensional variations. The target value for this prediction problem is the compressive strength in the produced flax shive composite. The methodology proposed in this study uses the dataset that is also described in 3.3.

4.3 Methodology

Depending on the complexity of the data, we can propose different scenarios for the regression problem. One way is to assume that there is a linear relationship between the data. In this case, a methodology based on the gradient descent method can be used to predict the data. In other cases, if we assume that the data are complex and there are interconnections between the parameters, we can apply more complex scenarios such as regression with artificial neural networks (we discuss this method in 6).

To ensure if there are any other machine learning algorithms that can predict better the compressive strength with a given data set, one way, is to fit this data using quadratic or any n-order polynomial function, where n can be from $\{1, 2, 3, ..., n\}$. A question is, which algorithm can give better results?

The answer depends on the complexity of the problem and the inter-relations between the parameters in the dataset and the result value. In fact, there is no unique order for the best machine learning method that fits every data type. The methods are different for each input data type and the size of the available data.

Linear regression model

In this study, we assume a simplified scenario with a linear relationship between the features and the target value. We recall that hypothesis is a function (or an equation) that by receiving the features, can generate an estimation for the target value. By considering all the features describes in 4.2, this regression problem has a total of 6 features and one target value.

By assuming that the hypothesis function is a linear equation, we assume that the output (Y) value has a linear relation with the input features (X: $\{x_1, x_2, ..., x_n, \}$). We recall that n represents the number of the features. Thus the model can be formulated as the hypothesis formulation in 4.1.

$$\upsilon(X) = \epsilon + x_1\beta_1 + x_2\beta_2 + ... + x_n\beta_n \tag{4.1}$$

Where $\upsilon(X)$ [1] is a linear funtion of X: $\{x_1, x_2, ..., x_n, \}$. $\upsilon(X)$ represents the predicted value for the compressive srength. In this example, $\beta_i, i = \{1, 2, ..., n\}$ represents the coefficient of the variables X: $\{x_1, x_2, ..., x_n, \}$.

We are going to apply an algoritgm to find the optimized values for $\beta_i, i = \{1, 2, ..., n\}$, coefficient of the parameters and ϵ as a fixed parameter. This formulation implies a straight line that can be fitted to the data.

An initial solution for this problem is to randomly assign some values to β_i and ϵ. To optimize this model, the algorithm needs to find the best values for β_i and ϵ that minimize the error between the predicted value and the true ones. We can define an optimization model as in 4.2.

[1] υ is the first letter of a greek word $\upsilon\pi o\theta\epsilon\sigma\eta$ meaning hypothesis

$$\begin{aligned} \min Z: \quad & Z = \frac{1}{2m} \sum_{i=1}^{m} \left(\upsilon(X)_i - Y_i\right)^2 \\ & \epsilon \quad , \beta_i \quad \forall i = \{1, 2, ..., n\} \end{aligned} \tag{4.2}$$

The objective function Z is also called the cost or loss function. It calculates the average squared of the prediction error. In this equation, m represents the number of the training examples, $\upsilon(X)_i$ is the predicted value, and Y_i is the ith example in the training set. β_i and ϵ are free variables that need to be calculated in the optimization process.

The presented methodology is to find the optimum solution for the quadratic problem. Herein, we proposed an optimization methodology using the gradient of the stated objective function. We can observe that the presented quadratic model 4.2, is convex. Thus, it is guaranteed that the proposed model is able to find optimum solutions.

A numerical example for the prediction problem

We use our dataset to predict the compressive strength of vegetal composite. The dataset is a list of examples of composites elaborated with different coating. It includes characteristics of shives after coating and characteristics of flax shive composites.

To simplify the example, we first consider only two features of the composites: compressive strength(MPa) and bulk density (kg/m^3). Figure 4.1, depicts these two features. The vertical axis is the compressive strength and the horizontal axis is the bulk density for the corresponding test. In this Figure, we depicted some examples with blue circles and the red dotted line represents their corresponding fitted straight line.

In this simplified example, we want to describe, how to predict the compressive strength of the vegetal composite with a given bulk density using machine learning methods.

In the first observation, we can conclude that the given dataset is compatible with supervised learning. In which, the correct value corresponding to the features of an example is given. This means for each given value of bulk density the corresponding actual value of the compressive strength is known.

In this example, the question is that how much will be the value of a compressive strength if the value of bulk density is ξ. Where ξ, can be any value within a defined range.

Since the output or answer value in this problem is the compressive strength of the vegetal composite, which is a positive and continuous variable. This problem is also

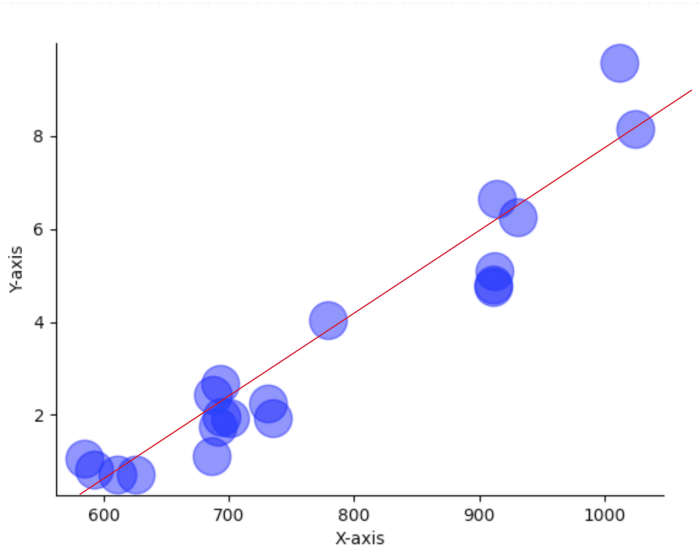


Figure 4.1: Univariate regression example: Visualization of the linear relation between compressive strength and bulk density. The red line shows a linear trend between the compressive strength on the axis Y and the bulk density on the axis X. The blue circles represent the data of the samples.

called a regression problem. The aim is to find an equation between the parameters to predict their corresponding answer, which here is the compressive strength.

4.4 Linear regression with two dimensions: application on the dataset

For this example, a learning algorithm can fit a straight line between the data and to predict a compressive strength value corresponding to the given bulk density. With this assumption, we can formulate a linear model that can fit the data. The linear equation can be defined as $Y = aX + b$. Given an input value X, by assigning a random value to a and b, we can have a line to predict Y. The proposed methodology in this study aims to find an optimum value for a and b.

As we can see in figure 4.1, maybe a straight line with a linear regression can actually, predict the result value with acceptable accuracy. We fit a regression line using Excel on 20 data of compressive strength and bulk density of vegetal composite.

The red dotted line in 4.1 represents the fitted regression. The observed linear regression equation with Excel is $Y = 0.0186X - 10.639$ where X corresponds to the bulk density and Y corresponds to the compressive strength. The R^2-value or coefficient of determination for this equation is 0.8042.

The complexity of the problem will increase by considering more features and the number of experiments. As stated before in 4.2, in this study we have 6 parameters from 123 experiences.

4.5 Regression on data with more dimensions

In the previous example described in 4.4, we only depicted one feature (bulk density) in a 2-dimensional figure to simplify the problem. However, there are other features, that can affect the compressive strength of a vegetal composite.

If we decide to predict the compressive strength of a vegetal composite using two features "bulk density" and "drying shrinkage", we can depict in Figure 4.2, how these two parameters affect the compressive strength value.

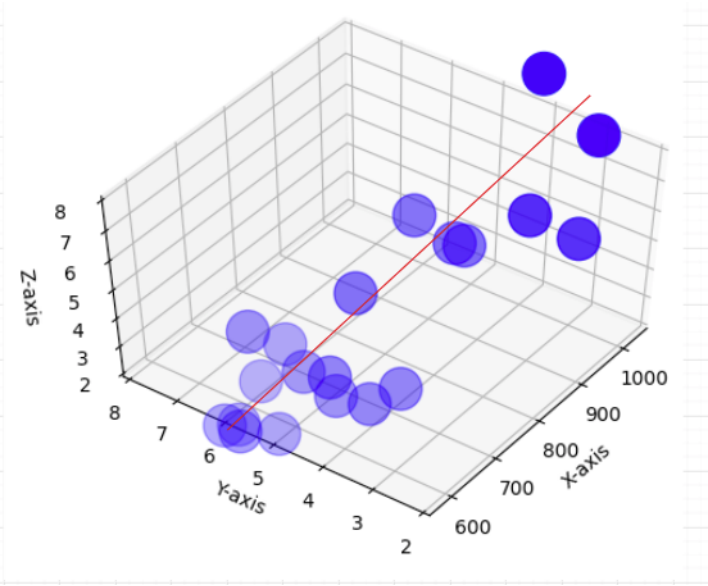


Figure 4.2: Example of the data for the 3-dimension regression model. Axis X: bulk density of a flax shive composite, Y: drying shrinkage, and Z: compressive strength of the samples. The red line shows a linear trend between the compressive strength, bulk density, and drying shrinkage. The blue circles represent the samples.

In our dataset, we have a total of 6 features to predict the compressive strength of a vegetal composite. Taking into account that we can visualize the parameters up to 3 dimensions, we are not able to depict the model with all the features. In this model, each feature corresponds to a dimension to the regression problem.

4.6 Conclusion

In this chapter, we presented a regression model and described a methodology to apply gradient descent on the data set. Linear models for two and three-dimensional regression are presented. Regression models are used to predict the compressive strength of composite from the experimental dataset.

Chapter 5

A methodology to predict the compressive strength of a vegetal composite using gradient descent method

Brief table of contents

In this chapter, we are going to propose a method to predict the compressive strength of the composite using the data. The used dataset is described earlier in 3.3. We present a regression model on the described data.

5.1 Introduction to the regression models

There are several studies in the literature to implement various optimization algorithms. Gradient descent is one of the most popular optimization methods that is used in studies on machine learning science(e.g. Bengio et al., 2013).

Gradient descent has three variants: (I) batch gradient descent, (II) stochastic gradient descent, and (III) mini-batch gradient descent. The difference between these methods lies in the amount of available data.

In this study collecting the data is time-consuming and expensive and the amount of available data is limited. Therefore, we propose to apply batch gradient descent on the data. Considering that the calculation time in the proposed algorithm is negligible, this algorithm can be an assist in the data collecting process.

Batch gradient descent calculates the gradient for the whole data set. Therefore, dealing with a large dataset, batch gradient descent converges very slowly. In this study, since the number of data is limited, we can safely use the batch gradient descent. The differences between the two types are also explained in Ruder, 2016.

5.2 Gradient descent methodology

One of the most applied methods in the literature to perform optimization and prediction is gradient descent due to its simplicity and applicability for several applications.

In this study, the gradient descent method is applied to find the optimum solution for the model proposed in 4.2. The objective function referred to as cost function can be formulated in Equation 5.1.

$$C(X) = \frac{1}{2m} \sum_{j=1}^{m} (\upsilon(X)_j - Y_j)^2 \tag{5.1}$$

Where $C(X)$ represents the cost function, m represents the number of the training examples and Y represents the actual value of compressive strength of the composite. For each example j, $\upsilon(X)$ represents the predicted value for the compressive strength of the composite. In fact, $\upsilon(X)$ is the hypothesis function that is defined in 4.1. It can be redefined in 5.2.

$$\upsilon(X) = \theta_0 + X\theta \tag{5.2}$$

Where X represents a vector of the features $X = \{X_1, X_2, ..., X_n\}$ and $\theta = \{\theta_1, \theta_2, ..., \theta_n\}$ represents their coefficients, and θ_0 represents the fixed value of the equation accordingly.

We can reformulate the equation 5.1 to the minimization model in 5.3.

$$\begin{aligned} \min Z: \quad & Z = \frac{1}{2m} \sum_{j=1}^{m} (v(X)_j - Y_j)^2 \\ & \theta_0 \quad , \theta_i \quad \forall i = \{1, 2, ..., n\} \end{aligned} \tag{5.3}$$

Where $j = \{1, 2, ..., m\}$ represents the examples and $i = \{1, 2, ..., n\}$ represents the features. To find the optimized solution we use the gradient of the cost funcion described in 5.3. We defined the following equation in 5.4:

$$\theta_{It+1} := \theta_{It} - \frac{\lambda}{m} \sum_{j=1}^{m} (P(\theta)_j - Y_i) X_j \tag{5.4}$$

The gradient descent method will minimize the objective function during It iterations by updating the coefficients of the parameters of the model. Considering $-\frac{1}{m} \sum_{j=1}^{m} (P(\theta)_j - Y_j) X_j$ is the gradient of the cost function described in Equation 5.1. At each iteration It of gradient descent, θ will be updated to the opposite direction of the gradient of the objective function. Parameter λ determines the learning rate. It determines step sizes to reach a local minimum. In other words, in each step, we move to the direction of the slope of the objective function towards the bottom of a slope, until we reach a local minimum. We recall that the model is convex[1], thus a local minimum is also the global minimum. Therefore the solutions are guaranteed to be optimum.

5.3 Principal steps to apply the optimization model

Data used in this study are analyzed using the following procedure:

- Shuffling data:

  As described in section 3.3 data are organized during the experimental test, thus we shuffle the data to avoid miscalculations and computational errors. Shuffling is performed in a way that only the order of the data will be permuted randomly not the parameters of the experiments.

- Separate dataset into a training set and a test set:

  Data is separated into two groups: a training set and a test set. Shuffling the data in the previous step is necessary so that there will be no difference in the quality

[1] not having any interior angles greater than 180∘

of the data in the training and test set. Statistical analysis will be performed in the training set and the quality of the method will be examined in the test set. Considering that in this study the data is limited to 123 experiments, we decided to assign 20 % of the data to the test set and the rest (80 %) is used to train the algorithm. To the best of our knowledge, there is no specific rule to decide the size of the training and test sets. The logic is to have enough data to feed to the algorithm so that there will be some examples of every type of input data and have some data to be able to test and validate the algorithm. In some other studies, this portion is 30% to 70% for the test and training sets.

- Mean normalizing the features (average and standard deviations):

- Scaling the features by mean normalization will guide the gradient descent algorithm to converge. Therefore, in order to perform gradient descent on the data, it is advised to mean normalize the data using the formulation 5.5:

$$\frac{X - \mu}{\sigma} \tag{5.5}$$

Where X is the value of the parameter to be normalized, μ and σ represents the average and standard deviations. Note that μ and σ are calculated once on the training set and then applied to normalize the test set.

- Randomly initialize theta:

Theta (θ) is the coefficient of the parameters described in section 5.2. Before applied gradient descent theta should be initialized. In this study, we initialized theta randomly between 0 and 1.

- Apply gradient descent algorithm to find an optimized theta:

By initializing the two parameters (number of iterations and learning rate) gradient descent algorithm can be applied to the data. One way to verify if the gradient descent algorithm is working correctly is to calculate the cost function and verify if the value of the average square of the prediction error as the cost function is decreasing in each iteration.

- Prediction equation:

After normalizing test set data using the average and standard deviations of the training set, the target value Y of each example can be predicted using Equation 5.6

$$Y = X \times \theta \quad \forall \theta \in \{\theta_0, \theta_1, \theta_2, ..., \theta_n\} \quad and \quad \forall X \in \{X_0, X_1, X_2, ..., X_n\} \tag{5.6}$$

Where X is the normalized data from the test set and n is the number of fatures. Recall that θ_0 is the coefficient of X_0 which represents the bias variable.

5.4 Computational analysis

The data are analyzed with *Python* version 3.7.7 on a Mac OS Catalina with 2.3 GHz Intel Core i5 processor. Parameters tuning of the gradient descent method are considered as follows:

- Learning rate α is set equal to 0.001, 0.01 and 0.1 and,
- The number of iterations is set equal to a maximum of 60 000 iterations.

Table 5.1 describes the parameters X_k with $k \in \{1, 2, 3, .., 6\}$ and the target value Y (compressive strength) along with their standard units.

Table 5.1: Description of the parameters

parameters	descriptions	units
X1	coating substance	lime or cement or none
X2	ratio shives per coating substance	none
X3	ratio water per coating substance	none
X4	bulk density	kg/m^3
X5	drying shrinkage	mm/m
X6	extrem dimensional variation	mm/m
Y	compressive strength	MPa

Learning rate can be defined as steps towards local optima. Bigger steps seem to lead the model to converge faster. Herein, we applied our model on three learning rates to compare and find the most efficient one.

A healthy algorithm must decrease the cost objective during some iterations. Once there will be no more decrease in the objective value, or if the decreasing gap is neglectable [2], we can stop the iterative algorithm. Of course, the number of iterations is in direct relation to computation time and learning rate.

Table 5.2 reports the coefficients of the features when using the optimized θ values. We recall that the equation related to each line of the table is represented as follows:

$$Y = \theta_0 + \sum_{k=1}^{m} \theta_k X_k \tag{5.7}$$

Where $m = 6$ represents the number of the features, θ_0 represents the coefficient of the fixed value (bias), and θ_k referes to the coefficient of the parameters.

[2]Depending on the sensitivity of the problem a gap can be neglectable with certain decimals(Habibzadeh and Habibzadeh, 2015). In this study, we considered a gap with less than 3 decimals as neglectable.

Table 5.2: Coefficients of the features

Model	Alpha	θ_0	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6
M1	0.1	1.10E-18	-0.25	0.14	0.05	-0.06	-0.08	0.80
M2	0.01	1.21E-07	0.76	-0.07	-0.23	0.05	0.12	-0.10
M3	0.001	-3.47E-19	-0.06	0.05	0.14	-0.08	-0.25	0.80

Table 5.2 reports the optimized coefficients of the parameters bias, X1, X2, X3, X4, X5, and X6 in the predicted equations regarding three models M1, M2, and M3 with learning rate (α) 0.1, 0.01, and 0.001 respectively. As we can see bias value for each model is very small. This value is used to balance the relations between the parameters and the target value. Bias is a constant value that helps the model in a way that it can fit best for the given data.

Table 5.3 reports the global analysis of the three tested models (M1, M2, and M3). Each model predicts the compressive strength of the vegetal composite by varying the learning rate α. We recall that in each model cost value decreases after each iteration. Column 3 represents the number of iterations that the cost value didn't decrease (With the magnitude of 3 decimals). The three tested models (M1, M2, and M3) are also compared by their corresponding MSE on the training set and the test set respectively.

From this table, we can conclude that:

- The minimum cost value founded is 0.119.
- The smallest learning rate takes more calculation time and more iterations to converge.
- There is no linear relation between learning rate and absolute errors.
- Contrary to the test set, the models with the learning rates 0.1 and 0.001 had fewer prediction errors than the model with a learning rate of 0.01 on the training set. $M2$ has less MSE on the test set than the two other methods($M1$ and $M3$).

Table 5.3: Global analysis

Model	Alpha	Iterations	Cost	Time	MSE	
					Training set	Test set
M1	0.1	126	0.119	0.0138	0.239	0.375
M2	0.01	918	0.119	0.0423	0.240	0.369
M3	0.001	9200	0.119	0.1520	0.239	0.375

Effect of the learning rate

We now evaluate the effect of the learning rate on the convergence of the gradient descent method. Generally, the more the learning rate is smaller the more the method converges slowler. It means with smaller learning rates, the algorithm takes more iterations to converge. As presented in Table 5.3, model $M3$ with the lowest learning rate takes 9200 iterations to converge. Whereas $M1$ with the largest learning rate0.1 takes 126 iterations to converge. That is the principle of the batch gradient descent method. Figure 5.1 confirms that phenomenon. Indeed, on the one hand, one can observe that the method provides a fast convergence with the highest value of $\alpha = 0.1$ while the smallest value of $\alpha = 0.001$ causes a very slow convergence. On the other hand, with $\alpha = 0.01$ it seems that the method converges smoothly towards the final solution.

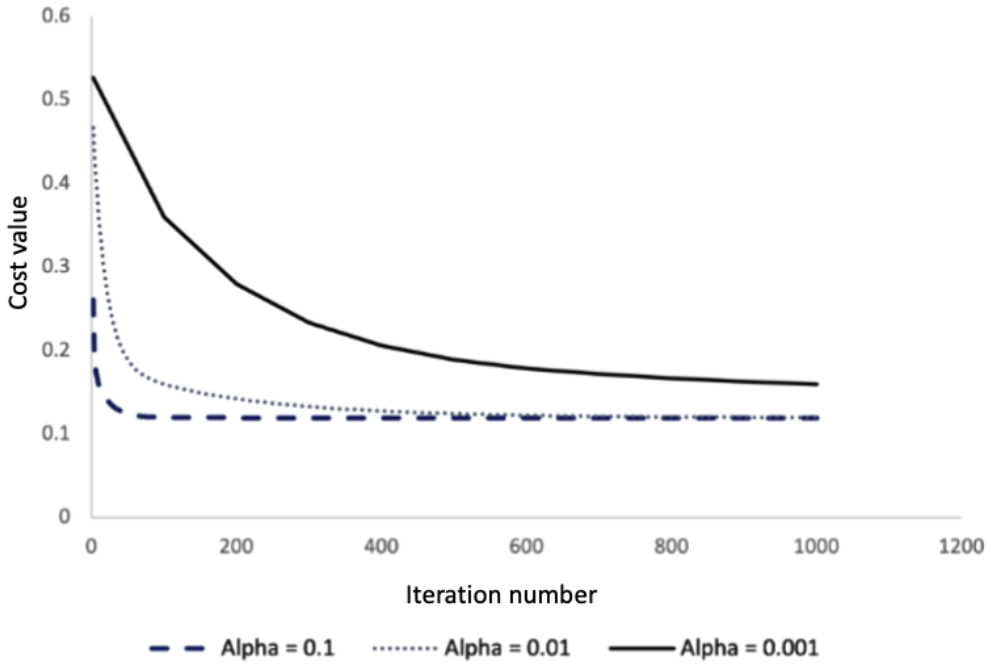


Figure 5.1: Trending the cost value by increasing iterations on three models with α equal to 0.1, 0.01, and 0.001

We can also obserbve from Figure 5.1, that there is more diversifications with the largest alpha (0.1). As we can see in early iterations (around iteration 126) the model has already found the minimum cost value (0.119).

Table 5.4: Iteration analysis

It.	Alpha=0.1		Alpha=0.01		Alpha=0.001	
	cost	time(min)	cost	time(min)	cost	time(min)
1	0.260	0.009	0.467	0.009	0.054	0.009
2	0.195	0.009	0.435	0.009	0.052	0.009
3	0.174	0.009	0.407	0.009	0.050	0.009
4	0.165	0.009	0.382	0.009	0.048	0.009
5	0.159	0.009	0.359	0.009	0.047	0.009
6	0.155	0.009	0.339	0.010	0.045	0.010
7	0.151	0.010	0.321	0.010	0.044	0.010
8	0.148	0.010	0.305	0.010	0.043	0.010
9	0.145	0.010	0.291	0.010	0.042	0.010
10	0.142	0.010	0.278	0.010	0.041	0.010
11	0.140	0.010	0.267	0.010	0.040	0.010
12	0.138	0.010	0.256	0.010	0.039	0.010
13	0.136	0.010	0.247	0.010	0.038	0.010
14	0.134	0.010	0.239	0.010	0.037	0.010
15	0.133	0.010	0.232	0.010	0.036	0.010

Effect of the number of iterations

Herein, we evaluate the effect of the number of iterations used on the convergence of the gradient descent method. Indeed, on the one hand, one can observe that the increasing the number of iterations, the more the cost value decreases, which eventually may converge to a certain value matching a local optimum. On the other hand, as we explained before, by increasing the learning rate, the cost function may also converge faster to a certain local optimum. Whenever increasing the number of iterations it can also induce a stagnation of the value related to the objective function or at least its decreases with a negligible value. Indeed, Figure 5.1 illustrates the variation of both the cost value and the number of iterations to predict the compressive strength with $\alpha = 0.001, 0.01$, and 0.1. One can observe that the method simultaneously converges to each local optimum, according to the rate α used, by using the closest number of iterations; that is less than the fixed maximum number of iterations. Of course, such a phenomenon can be explained by the fact that training examples are small.

We note that for the total of 123 available experiments, the average runtime is negligible (less than a minute), as underlined in Table 5.4.

5.5 Evaluating the algorithm

In order to evaluate the behavior of the gradient descent method with the parameter α, we consider the same variation as used in previous sections, i.e., α belongs to $\{0.001, 0.01, 0.1\}$.

Table 5.5 reports the prediction errors achieved by the gradient method on the test set.

The first six columns in Table 5.5, represents the values of the corresponding six parameters described in Table 5.1 (X1, X2, ..., and X6). Column 7 reports the target value in the test set. The rest of the columns (Column 8, Column 9, and Column 10) reports the prediction errors (predicted value - target value) by varying the parameter α in the interval $\{0.001, 0.01, 0.1\}$.

Table 5.5: Absolute errors on the test set

							Error (predicted value - Y)		
X1	X2	X3	X4	X5	X6	Y	α= 0.1	α= 0.01	α= 0.001
cement	0.33	1	974.30	3.94	3.14	2.88	0.84	0.83	0.84
lime	0.67	0.5	680.55	5.56	3.32	2.20	-0.29	-0.28	-0.29
cement	1	2	753.05	4.71	3.89	2.86	-0.49	-0.46	-0.49
lime	2	0.75	625.43	6.97	5.00	0.75	0.12	0.08	0.12
lime	2	0.5	616.06	7.61	7.91	1.08	0.07	-0.02	0.07
lime	1	2	782.11	5.48	3.10	2.57	0.21	0.19	0.21
cement	0.67	2	898.48	2.86	4.04	5.73	-1.04	-1.03	-1.04
lime	2	0.5	602.45	6.13	8.11	0.71	0.29	0.23	0.29
lime	0.33	2	705.04	3.24	1.72	0.63	0.90	0.97	0.90
lime	0.5	2	731.17	4.06	1.50	2.25	0.07	0.12	0.07
cement	0.33	0.75	909.26	5.95	3.50	2.00	0.83	0.80	0.83
lime	1	0.5	658.48	8.17	4.37	1.38	-0.06	-0.10	-0.06
cement	0.5	1	837.85	5.69	3.18	1.91	0.43	0.43	0.43
lime	0.33	0.75	1047.93	2.65	1.87	8.98	-1.71	-1.76	-1.71
cement	2	2	574.57	5.73	4.19	0.70	-0.52	-0.47	-0.52
lime	2	2	669.88	5.18	6.44	1.11	0.52	0.48	0.52
cement	0.33	1	901.37	4.92	3.23	2.55	0.53	0.52	0.53
cement	0.67	1	726.60	3.81	4.25	3.23	-0.82	-0.77	-0.82
lime	0.67	0.5	696.17	5.58	3.17	2.55	-0.40	-0.40	-0.40
cement	2	0.5	592.93	7.66	6.76	0.49	-0.31	-0.33	-0.31
lime	0.67	0.75	702.38	3.78	2.73	3.00	-0.53	-0.49	-0.53
lime	2	2	686.09	6.56	7.11	1.13	0.58	0.51	0.58
cement	0.5	0.75	792.50	4.58	3.68	1.80	0.29	0.31	0.29
lime	2	1	605.31	6.39	7.58	0.96	0.15	0.09	0.15
lime	0.5	0.5	639.69	5.30	3.28	1.58	-0.16	-0.13	-0.16

5.6 Conclusion

In this study, we studied the effects of the real-experimental case study related to the coating of flax straw before its incorporation into a cement matrix. A simple model was proposed for tackling the problem, where a batch descent gradient method was employed for predicting the compressive strength of a vegetal composite.

As the studied problem is related to a practical situation, there are plenty of possibil-

ities for further investigation involving efficient methods able to predict the parameters. First, neural networks can be applied to predict some parameters and so, in some cases they can be used for providing tight estimations. Second, the hybridization between neuronal networks and operational research technics can also be envisaged for achieving better predictions of the parameters, especially when multi-objective functions, like predicting parameters and minimizing the overall price of the raw material required, to design the target composite.

Chapter 6

Artificial neural networks to predict the composite compressive strength and flexural strength

Brief table of contents

This chapter focuses on predicting the compressive strength and flexural strength of flax shive composites. The cementitious composites are elaborated with coated shives. The coating process is conducted with two coating substances (lime and cement), different shive/coating substance, water/coating substance ratios, and tests without coating substances. Characteristics of flax shive composites (compressive and flexural strengths, bulk density, drying shrinkage, and extreme dimensional variation) are evaluated in the experimental study. The aim of this study is to design efficient artificial neural networks to predict the compressive and flexural strengths of a vegetal composite.

In what follows, we will introduce the artificial neural networks with the technical terms in 6.1. In section 6.2, the training approach and the application of the described method on a data set is explained in five steps. The experimental data used in this study are evaluated with several statistical analysis. The numerical results are reported in 6.3. The conclusions of this study can be found in 6.4.

6.1 Introduction to artificial neural networks

Artificial neural networks (ANNs) or simply neural networks (NNs) arose from both mathematical and biological insights. NNs is an inspired study of the human brain that mimics the process of solving solutions using neurons. ANN is known as a method that is able to model complex and nonlinear processes. It is widely used in many disciplines including civil engineering and modeling composite components (Asteris et al., 2016; Malagavelli and Manalel, 2014; Mansouri et al., 2016; Plevris and Asteris, 2014; Topcu and Sarıdemir, 2008).

Several simple-task elements, functioning in parallel are constructing NNs. Their major task is determined by their pattern of connectivity. These elements can process different types of input data using learning or adaptation and data processing. Learning or adaptation is the best feature of the NNs. NNs can learn from several examples of the input data. A trained NNs can predict the unknown variable(s) by finding a map through input parameters and the training examples.

Neurons and their relation weights are the main components of NNs. NNs is a multi-layer system with a minimum of three basic layers which are the input layer, hidden layer(s), and the output layer. This system is also called "multi layer perceptron". The perceptron algorithm is an elderly version of neural networks that were developed to classify, categorize, and separate data (Hastie, Tibshirani, and Friedman, 2009).

The first layer of a network is a layer with neurons that represents the parameters or features. We assume that there are nonlinear and hidden connections between the features or input parameters. These connections between the input features can lead the experimental studies to have a certain output value on the target parameter.

The middle layers of a network (all the layers between the input and output layer) are called "hidden layers". In an ANN, neurons in the hidden layer(s) will play the role of the inner connectivities between the input features. A network can have one or more hidden layers. Basic NNs have only one hidden layer. The number of hidden layers and the neurons in each hidden layer in a network is adjustable.

The main drawback of an ANN is its complexity. The number of neurons and hidden layers will increase the complexity of the model exponentially. Most of the statistical studies, analyze the data with no more than three hidden layers (Catto et al., 2003; Julián-Ortiz, Vicente, Pogliani, and Besalú, 2018; Yuen and Lam, 2006).

Each neuron in each hidden layer is connected with the former and later layers. We assume that there is no interconnection between the neurons in the same layer. The last layer address the output layer. In the presented prediction problem, the output layer has only one neuron which indicates the predicted value. However depending on the target value, the last layer can have several neurons. For example, in classification problems, there are more than one neuron in the last layer(Yang and Wang, 2020, Li, Delpha, Diallo, and Migan-Dubois, 2020, and Arunkumar et al., 2020).

The conceptual structure of a network is depicted in fig 6.1: In the input layer 6 features of a bio-based mixture composite are fed to the network.

Besides the features, a bias unit is also fed to the network as a constant parameter. Bias is an additional parameter that helps the model to learn the patterns, and it acts like an input node that always produces a constant value (with the value of 1).

Since bias is given as a constant value, they are not connected to any previous layer(s), it is just appended to the start/end of the input and each hidden layer, and is not affected by the values in the previous layer.

In another word, bias is an intercept of a linear equation to adjust the output along with the weighted sum of the inputs to the neuron.

The aim is to design networks to predict the compressive and flexural strengths of

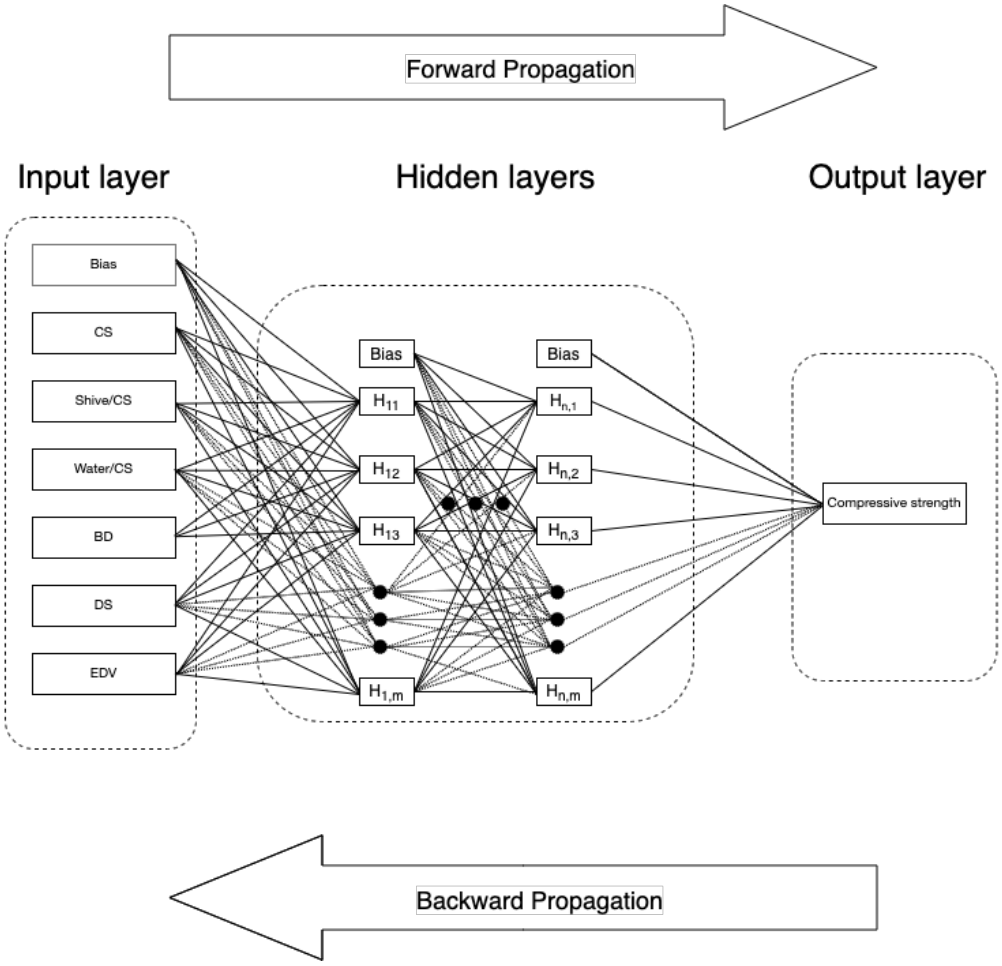


Figure 6.1: Global representation of NNs; The description of the abbreviations used in this figure are as follows; CS: coating substance, Shive/CS: ratio of the shive per coating substance, water/CS: ratio of the water per coating substance, BD: bending strength, DS: drying shrinkage, EDV: extreme dimensional variations, $H_{n,m}$: m^{th} element of n^{th} hidden layer

the composite. The experimental data of a 28-days CST are available in Goullieux et al., 2020a.

The data consists of 123 mixtures, presenting the characteristics of shives after coating and characteristics of flax shive composites for two different coating substances (lime, cement), and not treated flax shives. The data are also described in 3.3

Statistical analysis is performed to show the reliability of the predictions. The results show that proposed neural networks are able to predict and model the experiments. The proposed robust and reliable models are suggesting an efficient way to replace conventional time and cost consuming experiments.

6.2 Designing prediction/regression neural networks

The presented study proposes a heuristic approach to design reliable NNs that minimize the prediction errors. The steps to predict the compressive (or flexural) strength of the bio-based mixture composite are described as follows:

Step one: preprocessing the data

The first step is to preprocess the data. Preprocessing includes selecting the parameters and normalizing them.

There are some studies that discuss how to select the parameters in a generated dataset such as Shariati, Mafipour, Haido, et al., 2020. However, in this study since we are analyzing the experimental data, the considered parameters are selected by the experts and engineers that performed the experimental tests.

According to other related studies, normalization of the input data might increase the speed of the convergence. Herein we applied four normalization methods: 1) mean normalization, 2)min-max, 3)divide-max, and 4) none. According to Sola and Sevilla, 1997; Wu and Lo, 2010, the min-max method and mean normalization are the two most popular normalization methods in the literature.

Considering input feature x, μ the average value of the parameter x over the examples, and σ the standard deviation of the parameter x over the examples, the mean normalization formula is described as $\frac{x-\mu}{\sigma}$. With the same definition, if we define min (the minimum) and max (the maximum) value of the parameter over the examples, min-max normalization can be described as $\frac{x-min}{max-min}$. The divide-max method simply divides the parameter by the maximum value of the parameters over the examples. The last method (none) applies the algorithm without normalizing the data.

Step two: network configurations

The second step is to construct the network and define the number of hidden layers and number of the neurons in each hidden layer.

As we described earlier these two parameters are adjustable in the statistical analysis. Considering that by adding a hidden layer or a neuron to the network, the complexity of the model increases exponentially, we aim to find an efficient structure with the least complexity.

To have a visual understanding of the complexity that a neuron can add to the data, we drew the connections in Figure 6.1. The black connecting lines in the figure represent

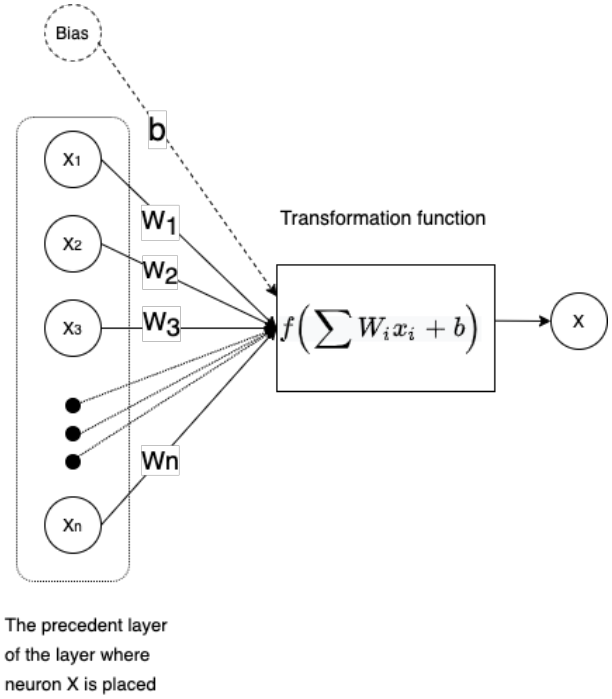


Figure 6.2: Data process in a neuron

the connections between the neurons.

Step three: initializing the network (forward propagation)

The third step is to assign weights and values to the network in forward propagation(FP). As the name "forward propagation" implies, the parameters of the network are fed to the network in a forward direction. To move from the initial layer to the successive layer, the weighted sum of the data is transformed through a transformation or activation function. Here an additional neuron called bias is added to the current layer to adjust the weights. Bias is an intercept added to a linear equation in order to adjust the output along with the weighted sum of the inputs to the neuron. It plays the role of a constant value in a linear equation.

Fig 6.2 describes the transformation process in a neuron.

Where $W_1, W_2, W_3, ...W_n$ represents the weights from the parameters of the previous layer $(X_1, X_2, X_3, ...X_n)$ to the stated neuron. The weighted average $(\sum W_i X_i)$ of the neurons in the previous layer are added to the bias unit b. They form a linear formulation as $\sum W_i X_i + b$. They will pass through the transformation function and will result in the stated neuron $(n = f(\sum W_i X_i + b))$. The achieved value is the predicted output value if the stated neuron is the neuron of the last layer.

Why and how to initialize the weights of the network?

To initialize the weights of the networks in FP, there are three possible scenarios as we described in the following.

1. To not initialize the weights or initializing the weights with zero: In this case, every neuron in every layer will act the same and the effect of the parameters will be ignored!

2. Or initializing the weights with a unique value: The network will be symmetric and all the neurons of all the layers will act the same and they will return the same output!

3. Or randomly initialize the weights: The network will not be symmetric and neurons will not perform the same computations.

Of course, as is explained earlier, the last scenario (scenario 3) is the only acceptable one. By randomly initializing the weights of the network to different values very close to zero, neurons can perform independently. Thus we can achieve accurate networks.

Why we use activation functions?

An activation function (also called transfer function) in a neural network defines how the weighted sum of the input is transformed into an output from a node or nodes in a layer of the network.

Activation functions are also typically differentiable, meaning the first-order derivative can be calculated for a given input value. This is required given that neural networks are typically trained using the back propagation of error algorithm that requires the derivative of prediction error in order to update the weights of the model.

The most commonly used activation functions are rectified linear unit (Relu), hyperbolic tangent (Tanh), and sigmoid or logistic (Chollet et al., 2018; Goodfellow, Bengio, Courville, and Bengio, 2016).

Sigmoid or logistic activation function is typically used in the logistic regression classification algorithm. The function takes any real value as input and outputs values in the range 0 to 1.

Relu is a function that always provides positive values (Nair and Hinton, 2010). It means that if the input value (x) is negative, then a value 0 is returned, otherwise, the value is returned ($f(x) = max(0, x)$).

Hyperbolic tangent (Tanh) takes any real value as input and outputs values in the range -1 to 1. The larger the input (more positive), the closer the output value will be to 1, whereas the smaller the input (more negative), the closer the output will be to -1.

Since activation functions demolish the values of the neurons, it is important to use activation function corresponding to the used data. The two activation functions used in this study are Relu and Tanh. Both of these activation functions provide continuous and positive values that are in line with the value of the parameters of this study and the value of the composite compressive and fexural strengths as the target value.

Step four: optimization the network (back propagation)

After constructing and initializing the weights of the network, it is time to optimize the weights. The fourth step is the optimization of ANNs with back propagation.

Herein, the training process is based on the gradient descent algorithm by updating the internal weights. The gradient descent algorithm aims to gradually update the network weights in a way that the predicted output be approximately equal to the given output value on the training set. Later the network with the optimized weights will be able to predict the output value using different input data. Considering that the data of the CST is limited, we decided to assign 20 % of the data to the test set, and the rest (80 %) is used to train the algorithm.

To the best of our knowledge, there is no specific rule to decide the size of the training and test sets. The logic is to have enough data to feed to the algorithm so that there will be some examples of every type of input data and have some data to be able to test and validate the algorithm.

The final reached value with the forward propagation is the predicted value. In order to decrease the distance between the expected output value and the predicted value (prediction error), the network should be trained and the weights of the network should be optimized. Back propagation (BP) is a method to optimize the network in the training phase.

BP as its name implies will start from the last layer and will move backward to the first layer. Gradually, it will tune the weights in the network. One way to recognize the impact of the back propagation is to calculate the prediction error after some iterations. BP should converge the predicted value to the expected value.

The main drawback of the BP is that BP needs a nonlinear activation or transformation function since it uses a gradient of the activation function.

In some studies, optimization methods or more precisely evolutionary algorithms such as genetic algorithm (GA) and particle swarm optimization (PSO) are used to

replace BP (Shariati, Mafipour, Mehrabi, Ahmadi, et al., 2020; Shariati et al., 2019). This can lead to find a better design for the NNs. However, they might not be feasible in real-world applications, because it will increase the complexity of the computations in terms of time and CPU and for implementation they need more parameters to tune.

Step five: evaluate the model

The last step is to compare the structures. A network can be evaluated using mean squared error (MSE) and R^2-score (coefficient of determination).

$$MSE = \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{n} \tag{6.1}$$

MSE (as defined in 6.1) represents a positive value. It implies that the better the predictions, the least the square errors.

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \tag{6.2}$$

R^2-score is defined in 6.2, where y_i,$\hat{y}_i$ and $\bar{y}$ represents the real value, predicted value and the average value respectively. It can be negative if the model is malfunctioning. The value of R^2-score for the best NNs will converge to one. If the model always predicts exactly the expected output value (overfitting problem), the R^2-score will be equal to zero.

6.3 Computational results

A total of 123 mixtures were taken from the 28 days of CST. The data represents the composite of shive and water with both lime and cement as coating substances, and the characteristics of flax shive composite.

In total 6 parameters are selected to predict the compressive strength of the bio-based composite. The parameters representing the characteristics of the flax shive are: 1) an integer parameter in range {1, 2, 3} that represents respectively lime, cement as coating substance, and tests without coated flax shives. 2) a portion of shive and 3) water with the coating substance. The composites are characterized by 4) their bulk density, 5) the drying shrinkage, 6) their extreme dimensional variation.

Parameter tuning

The analysis is conducted on MacOS operating system with 2.3 GHz, intel, Corei5 processor, and 16 GB memory. The proposed algorithm is implemented in Python 3.8 and by adapting the sklearn module Pedregosa et al., 2011.

In this study, we used the same experimental data described in 3.3. From the total of 123 mixtures, 80 percent (by means of 98 examples) of the data are randomly selected to train each network and 20 percent of the data (by means of 25 remaining examples out of the total 123 examples) are used to test the reliability of the network by applying MSE and R^2-score. Table 6.1 summarizes the parameters used to train each networks.

Table 6.1: Training parameters bounds and descriptions

Parameter tuning	Description or range
Training algorithm	gradient descent
Number of hidden layers	1, 2, and 3
Number of input features	6
Output variable	compressive strength
Number of neurons Per hidden layer	from 1 to 20
Normalization methods	min_max, mean normalization, devide by max, and none
Training coefficient	0.1
Maximum iterations	1000, 5000, 10000
Cost function	squared error
Activation function	relu(R),tanh (T)
Initial weights	(random between 0 and 1) * 0.001

We designed networks with one, two, and three hidden layers. As we explained earlier in Figure 6.1, increasing the number of hidden layer will increase the complexity of the models, thus we decided to train the models with not more than three layers. In fact, a network is able to recognize internal patterns using hidden layers. On the contrary, excessive hidden layers will force the network to memorize the training examples (Hiemstra, 1996).

Each model has 1 to 20 neurons per hidden layer. The models are represented in a numeric format. For example, a model with format 11-16-6 represents a model with 3 hidden layers and 11, 16, and 6 neurons in the hidden layers respectively.

The number of iterations is selected depending on the complexity of the models. More complex models require more iterations to converge in the optimization model. We fixed the number of iterations to 1000, 5000, and 10000 for models with one, two, and three hidden layers respectively.

In global 67360 different NNs with one, two, and three layers were designed for each prediction target; where each layer could have from 1 to 20 neurons. The designed networks are tested using the four normalization methods, and the two activation functions that were discussed in step four in section 6.2. The designed NNs architectures

were evaluated with MSE and R^2-score.

Comparison between the NNs models

The last step is to compare the structures. A network can be evaluated using mean squared error (MSE) and R^2-score (coefficient of determination). As we discussed earlier, MSE (as defined in 6.1) represents a positive value. It implies that the better the predictions, the least the square errors.

On the other hand, R^2-score (as defined in 6.2), can be negative if the model is malfunctioning. The value of R^2-score for the best NNs will converge to one. If the model always predicts exactly the expected output value (overfitting problem), the R^2-score will be equal to zero.

For the rest of the analysis, based on the characteristics of these two evaluation methods, we decided to sort our data based on the value of the R^2-score on the test set as for our purpose with R^2-score we can have more information and more analysis.

In this study, we also evaluated the effect of the normalization methods and activation functions for each N-layer ($N = \{1, 2, 3\}$) network design. Experiences are sorted based on their R^2 score. We selected the top 50 experiences for each N-layer network design.

Evaluation of the activation functions and normalization methods

In this study, we also designed NNs models to predict the flexural strength of a vegetal composite using the described parameters. To predict the flexural strength, training the NNs remains the same as described in 6.2, we also used the same parameters to tune the algorithm described in Table 6.1.

Table 4 and Table 4 in the appendix B represents the total of 150 experiences to predict the compressive and flexural strengths respectively.

Table 6.2 represents the number of times that a normalization method or an activation function were used in the top 150 (50 models for the one-layer networks, 50 models for the two-layer networks, and 50 models for the three-layer networks) different architectures for each target parameter (compressive strength and flexural strength).

The analysis of the designed models to predict the compressive strength from this table 6.2 shows that for one-layer models, divide-max normalization method (14 best configurations from 50 configurations) and relu activation function (40 best configurations from 50 configurations) were applied the most.

For two-layer models, min-max normalization method with 17 and relu activation

function with 41 best configurations from 50 configurations and for three-layer models min-max normalization method with 21 and relu activation function with 46 best configurations from 50 configurations, were applied the most in the 150 models ranked based on R^2-score on the test set.

The same analysis can be seen from the analysis of the designed models to predict the flexural strength on the N-layer models.

For one-layer models, the min-max normalization method (14 best configurations from 50 configurations) and relu activation function (41 best configurations from 50 configurations) were applied the most.

For two-layer and three-layer models, the divide-max normalization method with 15 and 16 configurations from 50 configurations and relu activation function with 47 and 45 best configurations from 50 configurations, were applied the most in the 150 models ranked based on R^2-score on the test set.

In total, min-max and divide-max for normalization methods and relu for activation function had appeared the most in the top 150 models of both predicted parameters.

Table 6.2: Evaluation of the activation and normalization methods on the best 50 experiences for each N-layer network ($N = \{1, 2, 3\}$), analysis with compressive strength and flexural strength

Target parameter	Model type	Normalization method				Activation function	
		mean	min-max	devide-max	none	tanh	relu
Compressive strength	One-layer	13	12	14	11	10	40
	Two-layer	7	17	14	12	9	41
	Three-layer	5	21	12	12	4	46
Flexural strength	One-layer	13	14	12	11	9	41
	Two-layer	8	8	15	19	3	47
	Three-layer	7	11	16	16	5	45
sum		53	83	83	81	44	260
most frequent			✓	✓			✓

Evaluation of the NNs models

The results shown herein were derived from the best designed networks. For both of the prediction targets (compressive strength and flexural strength), the best model ranked with R^2-score on the test set has three layers.

In the best designed NNs model to predict compressive strength, the parameters were normalized using the mean normalization method, tanh for the activation function. This architecture has $1.03E-5$ and 0.09 MSE and 1.00 and 0.98 R^2-score on training and test sets respectively.

Whereas, the best designed NNs model to predict flexural strength is normalized

with divide-max method and relu as activation function. This model has 19 neurons in the first hidden layer, 2 neurons in the second hidden layer, and 20 neurons in the last hidden layer. The MSE values for the training and test sets are 0.017 and 0.024 respectively. The R^2 scores are 0.96 and 0.93 respectively.

The detailed information from the best configurations of each L-layer model is summarized in Table 6.3. The columns represent the normalization method, activation function, layer format (it represents the number of the neurons in each hidden layer), the value of MSE and R^2-score on the training set, and MSE and R^2-score on the test set.

In general, Table 6.3 represents the best designs for each model with one, two, and three hidden layers.

The best founded configuration for 2-layer models to predict compressive strength is normalized with the mean normalization method and its activation function is relu. Its R^2-score on the test set is reported 0.97. This information is also reported in the third row of the table 6.3).

The best 1-layer model configuration to predict compressive strength is reported in the second row of the table 6.3. This model has 11 neurons in its hidden layer. The reported MSE on the test set is 0.16.

Similar information regarding the configurations of each N-layer mode to predict flexural strength can be found in the rest of the table 6.3.

For the rest of the models the R^2-score, MSE values, and other details are reported in Table 4 in the appendix B.

Table 6.3: Best configuration network design for the models with one, two, and three layers

Analysis	**Target parameter**					
	Compression strength			Flexural strength		
	1-layer	2-layer	3-layer	1-layer	2-layer	3-layer
Feature scaling	devide-max	mean	mean	min-max	min-max	devide-max
Activation	relu	relu	tanh	tanh	relu	relu
Layer format	11	9-10	11-16-6	10	10-10	19-2-20
MSE (training set)	0.05	0.02	1.03E-05	0.003	0.022	0.02
R2 (training set)	0.99	1.00	1.00	0.99	0.95	0.96
MSE (test set)	0.16	0.14	0.09	0.06	0.03	0.02
R2 (test set)	0.97	0.97	0.98	0.86	0.89	0.93

Comparison between the prediction and true values

We evaluated the designed models with the prediction error for each experimental data. The predicted error is defined as:

Prediction error= Experimental value - Predicted value

In total, there are six set of prediction models for each target parameter. We evaluated each of them in two groups of training prediction errors and test prediction errors.

Prediction error analysis on the compressive strength

Figure 6.3 shows the negative (in green) and the positive (in yellow) prediction errors on each data of the test set on the compressive strength and Figure 6.5 shows the prediction errors on the training set accordingly.

Error analysis on the test set

Figure 6.3a,6.3b, and 6.3c shows the errors of the prediction for the compressive strength on the test set for the 1-layer,2-layer, and 3-layer designed models respectively.

From total of 25 test set data, number of the positive prediction errors are 11,10 and 13 and number of the negative errors are 14,12, and 15 for the N-layer models. From the figure 6.3, we can conclude the followings:

- For the 1-layer model: The maximum negative error is less than the maximum positive errors. The maximum negative error and the maximum positive errors are 0.65 and 0.94 respectively.

  However, the minimum positive error is less than the minimum negative error. The minimum negative error is 0.05 and the minimum positive error is 0.002.

- For the 2-layer model: The maximum negative error is 0.67 and the maximum positive error is 0.81. The minimum negative error is 0.03 and the minimum positive error is 0.05.

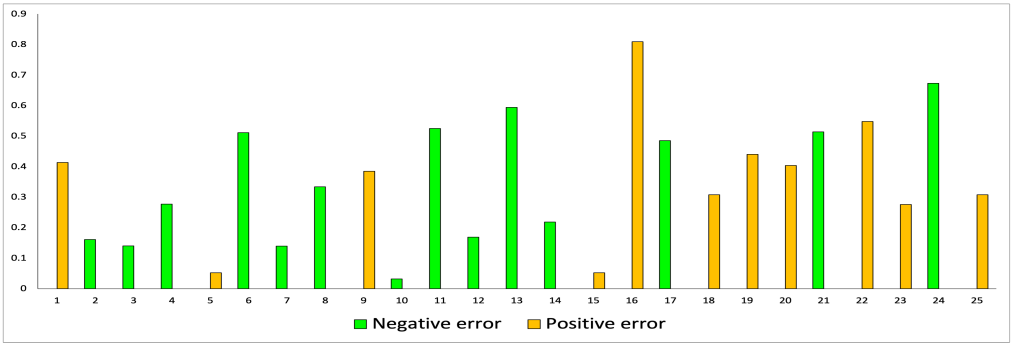
- For the 3-layer model: The maximum negative error is 0.65 and the maximum positive error is 0.94. The minimum negative error is 0.05 and the minimum positive error is 0.002.

- There are more negative errors than positive errors.

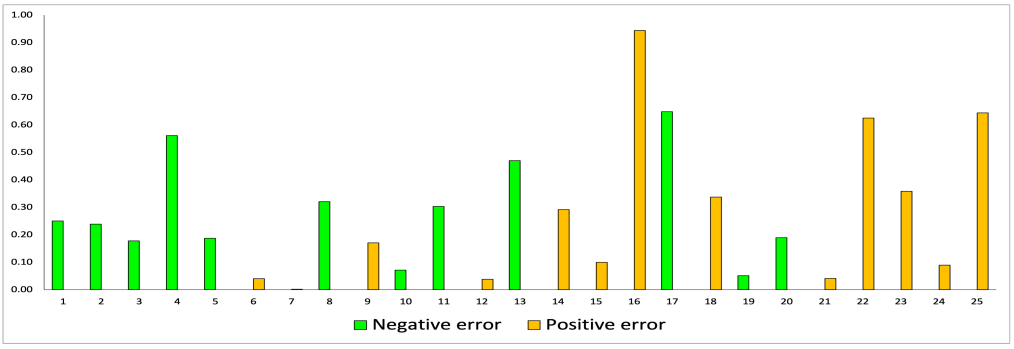
- In total, the maximum of the minimum error is 0.05.

Error analysis on the training set

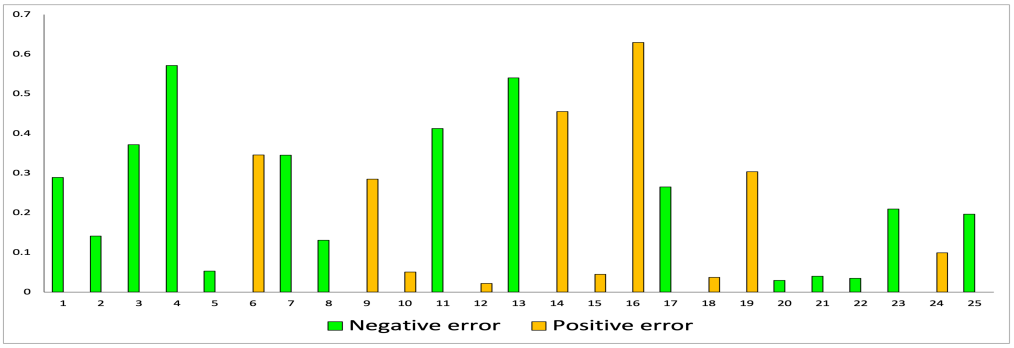
Figure 6.4a, shows the errors of the prediction for the compressive strength on the training set for the first-layer designed model.



(a) One-layer model



(b) Two-layer model



(c) Three-layer model

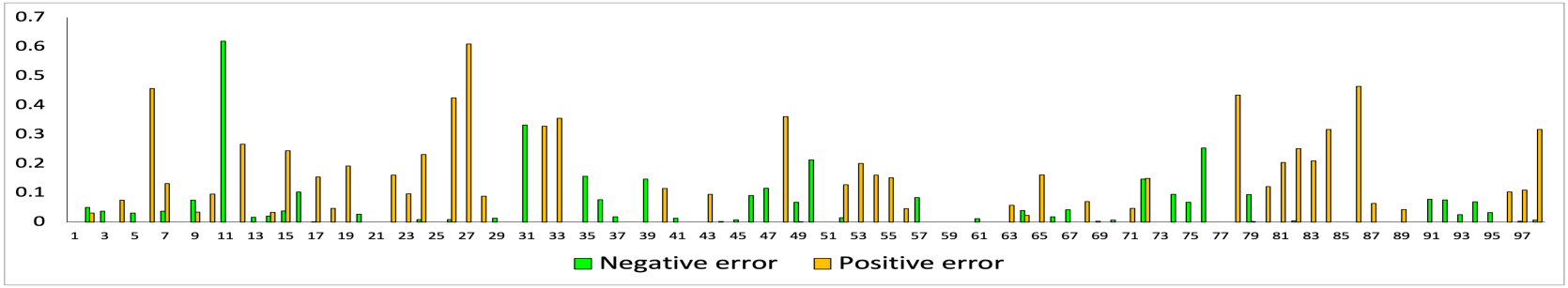
Figure 6.3: Compressive strength prediction error. Analysis on the test set for one, two and three layer models

The maximum negative error is 0.567 and the maximum positive error is 0.609. The minimum negative error is 0.00107 and the minimum positive error is 0.00115.

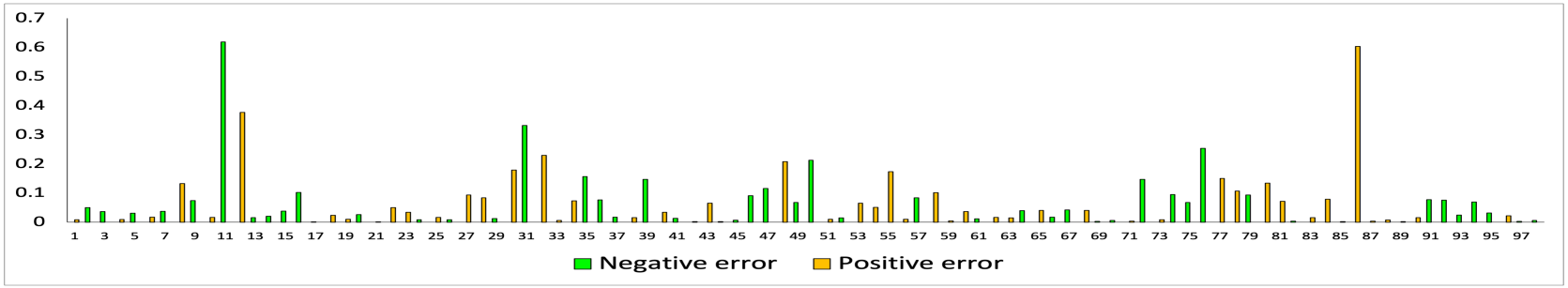
Figure 6.4b, shows the errors of the prediction for the compressive strength on the training set for the two-layer designed model. From total of 98 training data, total of 50 prediction errors are negative and total of 48 prediction errors are positive. The maximum negative error is 0.62 and the maximum positive error is 0.61. The minimum negative error is 0.00107 and the minimum positive error is 0.00173. Figure 6.4c, shows

the errors of the prediction for the compressive strength on the training set for the three-layer designed model.

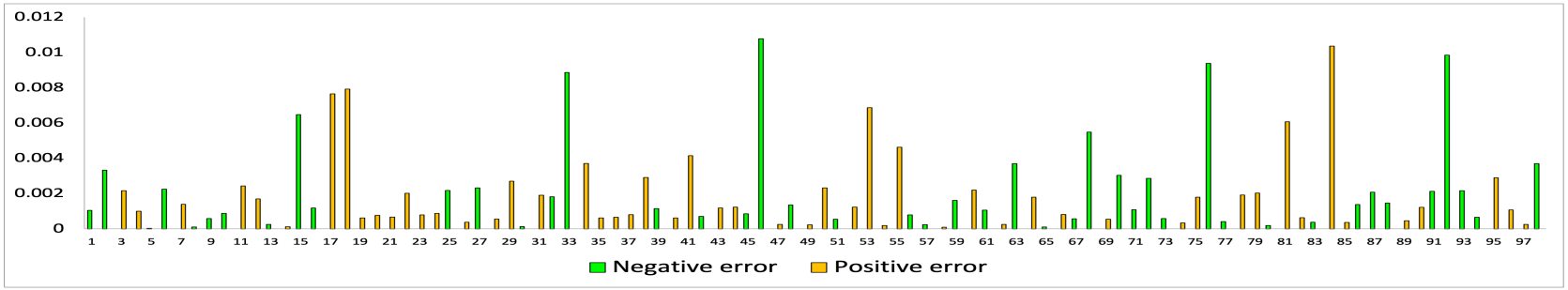
For the three layer models (also in Figure 6.4c), from total of 98 training data, total of 53 prediction errors are negative and total of 45 prediction errors are positive. The maximum negative error is 0.011 and the maximum positive error is 0.0103. The minimum negative error is 4.00642E-05 and the minimum positive error is 9.48172E-05.



(a) One-layer model



(b) Two-layer model



(c) Three-layer model

Figure 6.4: Compressive strength prediction error. Analysis on the training set for one, two and three layer models

Prediction error analysis on the flexural strength

Figure6.5 presents scatter plots of the predicted values versus respective experimental values for the given one, two and three layer models.

The statistical performance of the predicted models are given in Table 6.4. Table 6.4 shows: (i) the number and the percentage of the positive and negative errors; (ii) the value of the maximum positive and negative errors (ii) the value of the minimum positive and negative errors.

The analysis are performed on the prediction of the flexural strength for the N-layer $N = \{1, 2, 3\}$ models. The total of 6 models to predict the flexural strength are divided into test set and training set. Based on the analysis Table 6.4, we can conclude the followings:

- Among all the N-layer models, the minimum positive errors on the test and training sets are 0.0072 and 0.0003 on the 2-layer and 1-layer models respectively.
- Among all the N-layer models, the minimum negative errors on the test and training sets are 0.0029 and 0.0007, both on the 1-layer models respectively.
- The minimum positive and negative error is 0.0003 on the training and 0.0029 on the test set. This indicates that the minimum error is on the training set.
- The minimum of the maximum (positive and negative) errors are 0.2635 on the test set and 0.1830 on the training set.

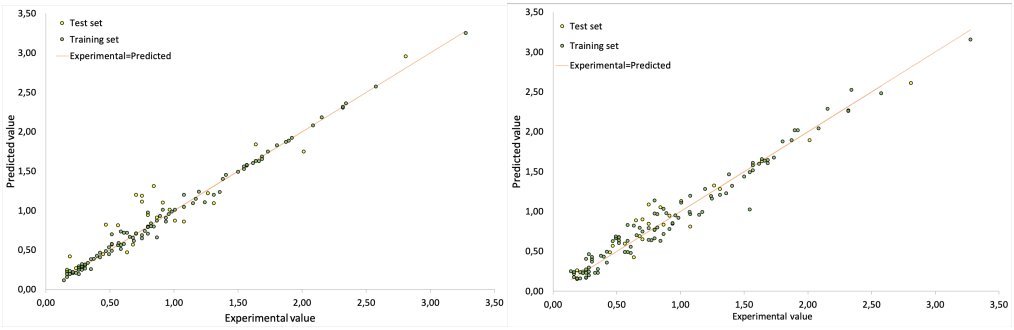
Table 6.4: Error analysis on the flexural strenght

Error= experimental - predicted	Test set			Training set		
	1-layer	2-layer	3-layer	1-layer	2-layer	3-layer
Positive errors (Percentage)	7(28%)	9(36%)	9(36%)	48(49%)	52(53%)	43(44%)
Maximum positive	0.2635	0.2670	0.3090	0.2048	0.2670	0.3389
Minimum positive	0.0450	0.0072	0.0184	0.0003	0.0076	0.0064
Negative errors(Percentage)	18(72%)	16(64%)	16(64%)	50(51%)	46(47%)	55(56%)
Maximum negative	0.5000	0.3387	0.2846	0.1830	0.3418	0.4700
Minimum negative	0.0029	0.0100	0.0090	0.0007	0.0121	0.0013

6.4 Conclusion

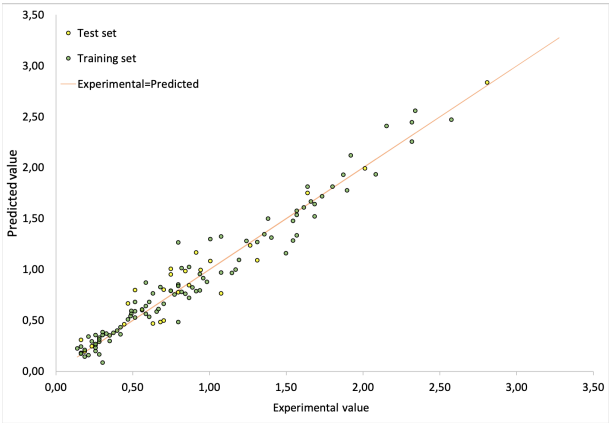
This study presented an application of neural networks on predicting the compressive strength of the flax shive composites.

The predicted values are compared to the experimental results evaluated from a 28-day of CST in the laboratory of EPROAD. Total of 67360 different N-layer networks ($N = \{1, 2, 3\}$) with from 1 to 20 neurons in the hidden layers for each prediction target.



(a) One-layer model

(b) Two-layer model



(c) Three-layer model

Figure 6.5: Flexural strength prediction versus experimental value on the training and test set for one, two and three layer models

The designed networks are tested using four normalization methods and two activation functions. The predicted results are validated with MSE and R^2-score.

The best design model ranked with R^2-score to predict compressive strength on the test set has three layers. The parameters were normalized using the mean normalization method and tanh activation function. This architecture has $1.03E-5$ and 0.09 MSE and 1.00 and 0.98 R^2-score on training and test sets respectively.

Whereas, the best designed NNs model to predict flexural strength is normalized with divide-max method and relu as activation function. This model has 19 neurons in the first hidden layer, 2 neurons in the second hidden layer, and 20 neurons in the last hidden layer. The MSE values for the training and test sets are 0.017 and 0.024 respectively. The R^2 scores are 0.96 and 0.93 respectively.

All the predictive models had excellent predicting abilities with high R^2 values and low MSE, which implies that these models were well-developed.

Chapter 7

Mathematical model and its optimization to predict the parameters of the compressive strength test

Brief table of contents

In this chapter, we propose an optimization descent method for predicting the composite compressive strength estimated parameters for lime and cement as coating substance for flax shives before their incorporation in the cement matrix. We first describe a simple model to express the main parameters used in this study. We then describe all equations related to both the amount of water to the amount of coating substance and the ratio of quantity of straw to quantity of coating substance. We then solve the provided model by applying an optimization method that is best upon a variant of the batch-gradient descent method. From a real-experimental study conducted on the coating of flax straw, its goal is to provide an average estimated error closest to zero for both used materials. Finally, the experimental results shows that the proposed model with its solution procedure is able to predict well the parameters related to all properties.

7.1 Introduction

We often resort to the use of means of transport resistant to temperature variations, to live in more or less well protected and insulated houses or to work in well soundproofed places, in addition to the characteristics that may be required for a depot. Protecting these moving and immobile assets from the ravages of time and the environment, users have often oriented their choices towards safe structures, such as choosing materials that are inherently strong and often cheaper. The most important mechanical property of a material, such as concrete or an equivalent product, is often related to its compressive strength. The compressive strength test of a material is a commonly used experiment to measure the different characteristics and the properties of that material.

In this chapter, we focus on the effects of the real-experimental case study related to the coating of flax straw before its incorporation into a special material (like cement matrix). During the actual experimental study, the relationship between the amount of water and the amount of coating substance (denoted CS) as well as the relationship between the amount of straw and the amount of CS may vary. Usually, on the one hand, measured characteristics are related to the bulk density, the water absorption capacity and the decrease in water absorption capacity (comparison with untreated straws) of straws. On the other hand, several other properties, which can be linked, can be taken

into account, such as flexural strength, compressive strength, bulk density, dimensional variation on drying as well as that linked to the extreme dimensional variation of composites. Therefore, the aim of the experiment is to have high mechanical properties and low dimensional variation on the formulated product.

7.2 Background

The mechanical property of a material is often measured by its compressive strength, where in some cases its effect is mainly employed to measure several characteristics related to properties of that material.

As discussed in de Siqueira Tango, 1998, the composite compressive strength Test (namely CCST) was first introduced for analyzing the behavior of the target material, which conventionally takes 28 days for a real-experiment. Of course, two directions may be considered for performing CCST-based experiment:

1. *AST method:* the first way consists in providing a direct real-experimental study, where *Accelerated Strength Testing* approach (namely AST) is applied. In some case, such an approach may be decomposed into several phases (as considered in C684-99, 1999): (a) using the warm water method; (b) using a boiling water approach; and (c) other approaches.

2. *Soft Computing-Based approach*: the second way consist on predicting target material with its estimated parameters by using optimization and/or learning based methods (de Siqueira Tango, 1998 and, Snell et al., 1989).

Several methods have been designed for predicting the strength of cement, mortar, or concrete. Indeed, de Siqueira Tango (in de Siqueira Tango, 1998) proposed an exploration-based method for estimating the compressive strength of a special material; that is the hydraulic cement product like a Brazilian case study. In the aforementioned study, a function related to resistance according to the time was used to strain the extrapolation of the predicted resistance of the material, where three manners were investigated: (i) taking into account a late age, (ii) taking into account an average age which is composed on two basic resistances and, (iii) low age. The experimental part showed that the proposed method had a good behavior, where it made it possible to obtain satisfactory estimates of the compressive strength of the material studied.

Kabir, Hasan, and Miah, 2013 tackled the prediction of concrete strength by developing a simplified mathematical model. Their model was tailored for the different ages while a single day concrete strength test result is needed. By using some special parameters, the experimental part showed the dependence between parameters' tuning

and the quality of the predictions, especially for regulating the concrete strength gaining characteristics regarding the age.

Other popular approaches, like regression methods, have been adapted for estimating (and predicting) several parameters either for mono-criteria or multi-criteria problems (de Siqueira Tango, 1998 and, Snell et al., 1989). The behavior of materials by itself may depend on both chemical and physical parameters that affect each other in nature. Therefore, detecting and optimizing these materials is not the most efficient method and so, both statistical and analytical models with the use of soft computing-based approaches, including regression analysis seems a promising way to the composite compressive strength prediction problem.

The optimization of concrete (and generally composites) concerns selecting values for the constituent parameters. Regarding this direction of research, Baykasoğlu, Öztaş, and Özbay, 2009 proposed a two-step approach for solving the multi-objective optimization of high-strength concretes. In their study, they proposed the use of the regression analysis, the neural networks and the gen expression programming to predict high-strength concretes parameters in the first step. This step established equations related to the concrete characteristic in terms of its components. According to these equations, a multi-objective optimization model has been designed, where it has been solved by using a genetic algorithm; that is the second step of the approach. In Baykasoğlu et al., 2004 another approach has been proposed for tackling the same problem, where the gene expression programming, the neural networks and the stepwise regression analysis were used as soft computing techniques.

Herein, the problem studied by itself is time-consuming and therefore expensive for industries. Because the importance of such a problem, we first propose to simulate the composite compressive strength prediction problem by a series of equations. The equations will return the parameters that needs to be varied in a real-experiment. Results of the equations provided in this study can be replaced with achieved experiment.

7.3 The composite compressive strength: parameters' prediction

Practical experiments are conducted in the Laboratory EPROAD, where 120 experiments were conducted on the vegetal composites including: (i) 60 experiments on *Lime* as coating substance, (ii) 60 experiments on *Cement* as coating substance. Because both lime and cement are used, we then considered two different experiments:

1. *Lime's study.* The objective of the first study is to predict the amount of quantity of water to the amount of the quantity of CS and, the ratio of quantity of straw to quantity of CS for lime by having the bulk density and water absorption capacity

Table 7.1: Representation of the used variables

Symbol	Parameter
X_1	Shive water absorbance (%)
X_2	Shive bulk density (kg/m^3)
X_3	Composite flexural strength (MPa)
X_4	Composite compressive strength (MPa)
X_5	Composite bulk density (kg/m^3)
X_6	Composite DS (mm/m)
X_7	Composite EDV (mm/m)
Y_1	Shive/Coating substance (S/CS)
Y_2	Water/Coating Substance (W/CS)

of straws and flexural strength, compressive strength, bulk density, dimensional variation on drying (drying shrikage or DS) and extreme dimensional variation (EDV) of composites. This experiment will induce two equations where each of them is used to predict a ratio.

2. *Cement's study.* In addition to the first study, the second experiment will calculate the two ratios related to the cement.

Table 7.1 illustrates the parameters used in this study, where all parameters are considered in both experiments, i.e., for the *Lime* and the *Cement*.

Therefore, by using all symbols introduced in Table 7.1,

- On the one hand, for the first experiment the parameters are represented by the set $\{X_1, X_2, X_3, X_4, X_5, X_6, X_7\}$ such that Y_1 denotes the variable to predict for *Lime*.
- On the other hand, for the second equation the parameters are characterized with the set $\{X_1, X_2, X_3, X_4, X_5, X_6, X_7\}$ such that Y_2 is the predicted variable.

We note that for the second experiment, the similar parameters are selected using data of *Cement*.

Regression analysis

Regression analysis is a powerful statistical approach that can be applied for removing a relationship between both dependent and independent variables. It generally may help in modeling some relationship between the used variables, where several types of regressions may be considered, like linear regression, non-linear regression and multiple linear regression. As the literature mentions the preference for the use of linear or multi-linear regressions, we decided to apply the linear regression in this study.

In addition, the use of a linear regression requires the use of one of the gradient descent methods. The gradient descent is an optimization algorithm used for minimizing (or maximizing) the cost (profit) function in various machine learning algorithms. It is basically used for updating the parameters of the considered learning model. As discussed in the literature (Bengio et al., 2013 and, Ruder, 2016), there are three types of gradient descent-based methods:

- Batch gradient descent method: At each iteration of the search process, all the training samples are processed. However, this version remains interesting when the data size remains small. Therefore, the larger the size of the training examples (data) more the convergence of the method is slower.
- Stochastic gradient descent method: At each iteration of the search process, only one training sample is processed and all parameters are updated. In this case, one can observe that this version becomes interesting whenever the data size (training samples) remains small. It induces an acceleration of that version when compared to the previous version, even if it remains heavy when the number of training samples increases.
- Mini-batch gradient descent method: Often this version is faster than the first two versions, even if the manipulation of this approach remains subtle for a good parallelization.

Herein, as mentioned above, data is expensive and collecting the data is time consuming. Therefore, the amount of available data is limited and so, the batch gradient descent calculates the gradient for the whole data set. Because the number of data are limited in this study, we can safely use that version of the descent method.

Adaptation of the gradient descent

In this section, the linear equations are established for predicting the composite compressive strength test (CCST) parameters. Such a model can be described as follows:

$$P(\theta) = \theta_0 + X\theta, \tag{7.1}$$

where θ_0 refers to a constant value, X denotes the parameters of the model and θ is the coefficient related to all parameters.

Batch gradient descent method

For this version of the gradient descent, the objective function is defined as the average summation of the squared errors of the prediction for the linear regression. Differently

stated, that objective function can be written as the following minimisation cost function:

$$C(\theta) = \frac{1}{2m} \sum_{i=1}^{m} \big(P(\theta)_i - Y_i\big)^2, \tag{7.2}$$

where C(θ) represents the cost function for each θ, m denotes the number of the training examples while Y is the actual value of the variable; that is predicted using the cost function that should be minimized.

The method tries to minimize the objective function using an iterative procedure, where the coefficients of the parameters of the model are updated as follows:

$$\theta_j := \theta_j - \frac{\alpha}{m} \sum_{i=1}^{m} (P(\theta)_i - Y_i) X_i \tag{7.3}$$

At each iteration j of the gradient descent, θ will be updated to the opposite direction of the gradient of the objective function. The parameter α determines the learning rate. It determines step sizes to reach a local minimum. In other words, at each step we move to the direction of the slope of the objective function towards the bottom of a slope, until a local minimum is reached.

Stochastic gradient descent

The difference with the above version of the gradient method lies in the form of the objective function used and the optimization process. As underlined above, it is preferable to use that method whenever the size of data becomes large. Herein, the training process is performed on each training example while all parameters are updated at each iteration. The used cost function is represented as follows:

$$C(\theta, (x^i, y^i)) = \frac{1}{2} \big(P(\theta, x^i) - Y^i\big)^2 \tag{7.4}$$

and

$$C_{train} = \frac{1}{m} \sum_{i=1}^{m} (C(\theta, (x^i, y^i))), \tag{7.5}$$

where C(θ) denotes the cost function for each θ, m represents the number of the training examples while Y is the actual value of the variable. Such a value is predicted by using the cost function that should be minimized. Finally, (x^i, y^i) represents features and true value of a training example.

Moreover, the stochastic gradient descent repeatedly applies the following equation:

$$\theta_j := \theta_j - \alpha (P(\theta, x^i) - y^i) x_j^i, \tag{7.6}$$

where $(P(\theta, x^i) - y^i) x_j^i$ is the derivative of the cost functions such that Equation (7.6) will be for each $i = 1, 2, \ldots, m$ and $j = 0, 1, \ldots, n$, where m is the number of examples n the number of features.

Optimization and prediction processes

This section is composed of two parts. First, the *data collection*, which reflects the results induced from the real-experimentation, is presented (where the detailed experiments for both lime and cement are reported in the Appendix A). Second and last, the set of data is decomposed into subsets in order to establish correlations between the real experimental values and those to estimate (predict).

Data collection

Practical experiments are conducted in the EPROAD Laboratory of the University of Picardie Jules Verne, where 120 experiments were conducted on the vegetal composites including: (i) 60 experiments on *Lime*, (ii) 60 experiments on *Cement*. In the experiment, for each CS, the ratio of quantity of water to quantity of CS is varied in the interval {0.33, 0.5, 0.75 , 1} while the ratio of the quantity of straw to quantity of CS is varied in interval {0.5 , 0.75 1, 2}.

As illustrated in Table 7.1, for each CS the measured parameters are shive water absorbance (%) , shive bulk density (kg/m^3) , composite flexural strength (MPa), composite compressive strength (MPa), composite bulk density (kg/m^3), composite DS (mm/m), composite EDV (mm/m).

We recall that the goal of the experiment is to preserve high mechanical properties and low dimensional variation on the formulated vegetal composites (The experimental data used in this study are summarized in both Tables 1 and 2 of the Appendix B: the first table for the lime coating and the second one for the cement coating).

The search process

Data described and explained in section 7.3 are analyzed by applying the following process (for a more detailed comprehensive, the reader may refer to Ng, 2018):

Shuffling data. As described in Section 7.3, data are organized during the experimental test. Thus, we shuffled the data to avoid miscalculations and computational errors. The shuffling is performed in a way that only the order of the data will be permuted randomly not the parameters of the experiments.

Creating both training and test sets. Data is separated into two groups: the *training set* and the *test set*. Shuffling the data in the previous step is necessary so that there will be no difference in the quality of the data in training and test sets.

Statistical analysis will be performed in the training set and the quality of the method will be examined in the test set. Note that the size of the test set is setting equal to 20% of the data and the rest (i.e, 80% of data) represents the size of the training set. Of course, limited computational results showed that the used values remains acceptable because of the small number of training samples.

Average and standard deviations. Measured data from CCST as represented in Table 3.2 has various ranges. Table 3.2 represents average, standard deviations, minimumm and maximum value for each parameter. Statistical data of lime and cement is presented in the second and third row respectively. More information about all the used dataset in this study can be found in section 3.3.

Since the range of the values are diversified, one parameter can be 100 times greater than the other one. In this case, scaling the features by mean normalization will drive the gradient descent-based algorithm to converge. Therefore, in order to perform gradient descent on the data, it is advised to mean normalize the data as follows:

$$\frac{X - \mu}{\sigma}, \tag{7.7}$$

where X denotes the value of the parameter to be normalized, μ and σ represent the average and the standard deviations, respectively. Note that μ and σ are computed once on the training set and then they are applied to normalize the test set.

Randomly initializing θ. The parameter θ is the coefficient of the parameters that is described in section 7.3. Before applying the gradient descent method, θ should be initialized. Herein, it is randomly initialized in the interval $[0, 1]$.

Optimizing the parameter θ. By defining the two parameters (number of iterations and learning rate), the gradient descent algorithm can be applied on the data. One way to verify if the gradient descent algorithm works correctly, is to calculate the cost function and to verify if the value related to the average square of the prediction error as the cost function is decreasing at each iteration.

Using the prediction process. After normalizing test set data, using the average and standard deviations of the training set, predicting the variables may be computed as follows:

$$Y = X\theta, \tag{7.8}$$

where X denotes the normalized data from the test set while Y is related to the predicted value.

7.4 Computational results

The data are analyzed using GNU Octave, version 5.2.0 on a Mac OS Catalina with 2.3 GHz Intel Core i5 processor. Parameters tuning of the gradient descent method are considered as follows:

- Learning rate α is set equal to 0.001, 0.01 and 0.1 and,
- The number of iterations is set equal to a maximum of 60 000 iterations.

As discussed above, there are 60 data for each experiments; thus, the average runtime is negligible (less than a minute), as underlined in Section 7.3.

We note that because two versions of the gradient descent method are considered, we first provide an experimental part with the batch gradient descent method which seems more appropriate for small sized-problems. We then provide a second part in which the stochastic version of the method is applied with the best tuning achieved for the batch version, i.e., the number of iterations used and the best learning rate for α.

Table 7.2: Resulted coefficient of the parameters for the linear prediction model

Coating Substance	Target	α	θ_0	X_1	X_2	X_3	X_4	X_5	X_6	X_7
Lime	Y_1	0.001	0.89	0.12	-0.27	-0.01	-0.19	0.26	-0.02	0.31
Lime	Y_1	0.01	0.89	-0.02	-0.19	0.12	-0.02	0.31	-0.28	0.28
Lime	Y_1	0.1	0.89	-0.19	-0.28	-0.02	0.31	0.28	0.12	-0.02
Lime	Y_2	0.001	1.06	-0.20	-0.96	-0.23	1.00	-0.05	-0.01	0.11
Lime	Y_2	0.01	1.06	-0.02	-0.06	0.10	-0.23	-1.00	-0.21	1.05
Lime	Y_2	0.1	1.06	-0.06	0.10	-0.23	-0.02	-1.00	1.05	-0.21
Cement	Y_1	0.001	0.85	0.07	0.21	-0.13	0.05	-0.10	-0.12	-0.26
Cement	Y_1	0.01	0.85	0.20	-0.12	0.05	-0.10	0.07	-0.13	-0.26
Cement	Y_1	0.1	0.85	0.07	-0.13	-0.26	0.20	-0.12	-0.10	0.05
Cement	Y_2	0.001	1.14	-0.09	0.51	-0.02	-0.35	0.04	-0.30	0.07
Cement	Y_2	0.01	1.14	-0.09	0.51	0.07	-0.02	-0.35	-0.30	0.03
Cement	Y_2	0.1	1.14	-0.09	0.51	-0.02	-0.30	0.03	0.07	-0.35

Table 7.2 reports the coefficients of the features when using the optimized θ values. We recall that the equation related to each line of the table is represented as follows:

$$Y_t = \theta_0 + \sum_{k=1}^{m} \theta_k X_k$$

where $m = 7$ and $t = 1$ (resp $t = 2$) for expressing the lime (resp. cement).

Figure 7.1 (resp. Figure 7.2 and Figure 7.3) illustrates the variation of both real-experimental values related to both Y_1 and Y_2 when considering the prediction for lime −the first graph− (resp. cement −the second graph−) while varying the learning rate $\alpha = 0.001$ (resp. $\alpha = 0.01$ and $\alpha = 0.1$).

First, predicted values related to Y_1 for lime seem closest to those extracted from the real-experimentation and those associated to Y_2 have also the same behavior, except

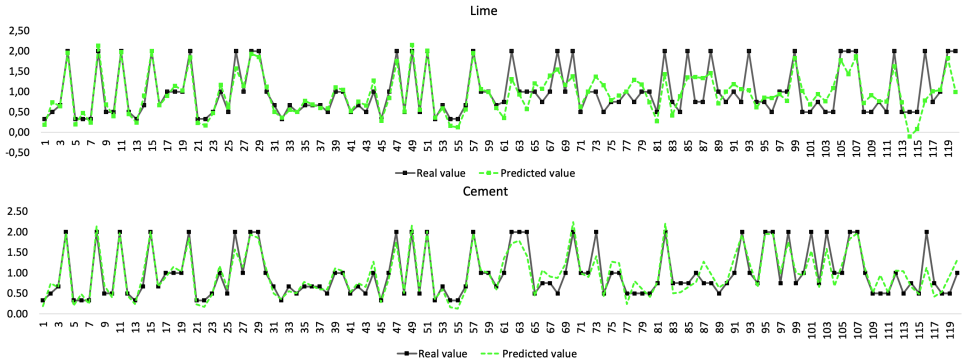


Figure 7.1: Real-values versus predicted-values for predicting Y_1 and Y_2 with $\alpha = 0.001$.

for one observation/prediction for which the achieved predicted value is equal to -0.11 while its real-value is equal to 0.50. In this case, this defect can be observed on the three figures while varying α in the interval $\{0.001; 0.01; 0.1\}$.

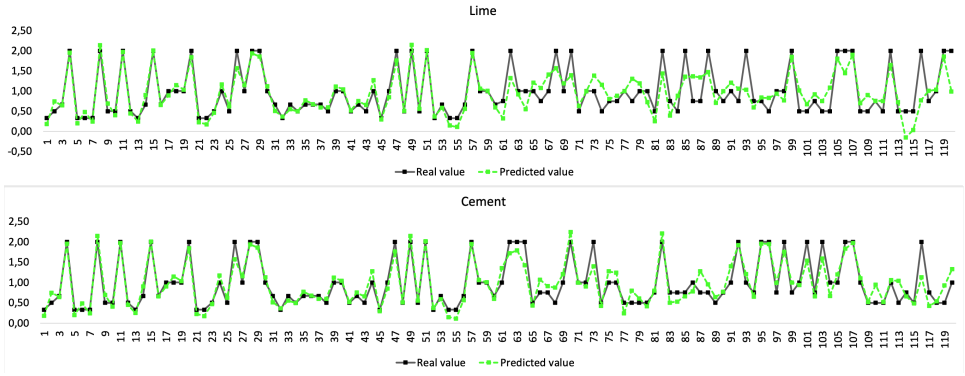


Figure 7.2: Real-values versus predicted-values for predicting Y_1 and Y_2 with $\alpha = 0.01$.

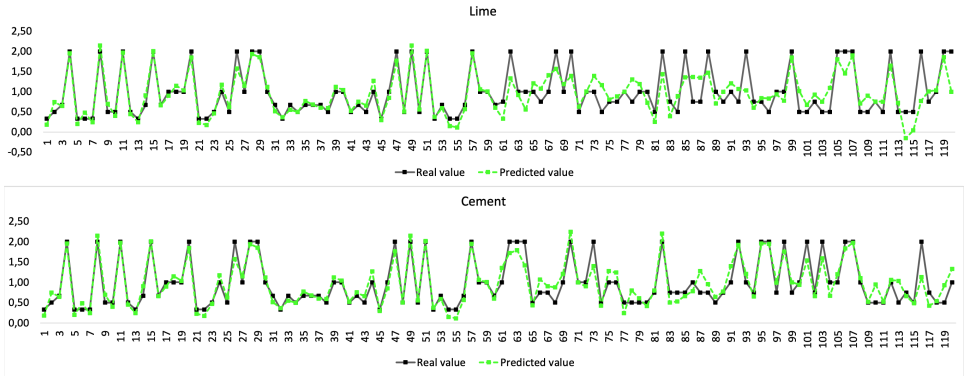


Figure 7.3: Real-values versus predicted-values for predicting Y_1 and Y_2 with $\alpha = 0.1$.

Second, predicted values related to Y_1 when considering the cement are more interesting. Indeed, in this case, no-defect is observed on the three figures corresponding to the variation of the learning rate α.

Effect of the number of iterations

Herein, we evaluate the effect of the number of iterations used on the convergence of the gradient descent method. Indeed, on the one hand, one can observe that increasing the number of iterations, more the cost value decreases, which eventually may converge to a certain value matching a local optimum. On the other hand, as we explained before, by increasing the learning rate, the cost function may also converge faster to a certain local optimum. Whenever increasing the number of iterations it can also induce a stagnation of the value related to the objective function or at least it decreases with a negligible value. Indeed, Figure 7.4 illustrates the variation of both the cost value and the number of iterations to predict Y_1 for lime with $\alpha = 0.001, 0.01$ and 0.1. One can observe that the method simultaneously converges to each local optimum, according to the rate α used, by using a closest number of iterations; that is less than the maximum number of iterations fixed. Of course, such a phenomenon can be explained by the fact that training examples is small.

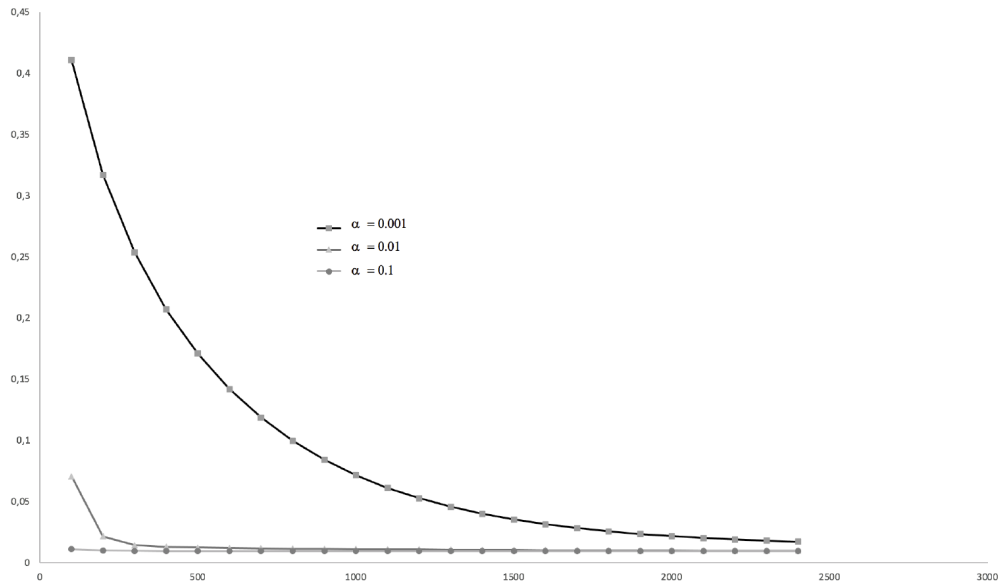


Figure 7.4: Behavior of the cost function for predicting Y_1 vs the number of iterations when varying the parameter α.

One can observe that the method stagnates with $\alpha = 0.01$, it reduces slowly the cost value with $\alpha = 0.001$ while it seems to converge smoothly towards the final solution with $\alpha = 0.1$.

Effect of the learning rate

We now evaluate the effect of the learning rate on the convergence of the gradient descent method. Generally, more the rate is important more the method converges slowly; that is the principle of the batch gradient descent method. Indeed, Figure 7.5 confirms this phenomenon. Indeed, on the one hand, one can observe that the method provides a fast convergence with the highest value of $\alpha = 0.1$ while the smallest value of $\alpha = 0.001$ causes a very slow convergence. On the other hand, with $\alpha = 0.01$ it seems that the method converges towards the final solution as observed for $\alpha = 0.1$.

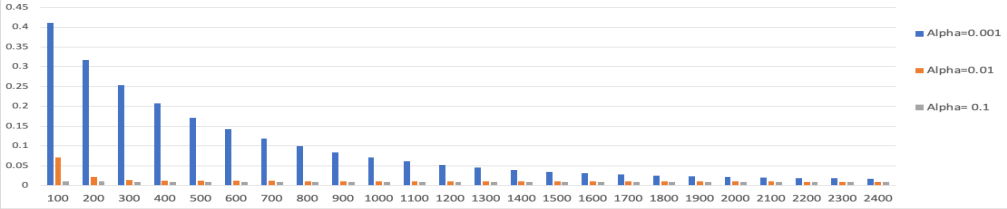


Figure 7.5: Variation of the cost function for a maximum of 2400 iteration with varying α to Y_1 (lime)

Statistical analysis

In order to evaluate the behavior of the gradient descent method with the parameter α, we consider the same variation as used in previous sections, i.e., α belongs to $\{0.001, 0.01, 0.1\}$. Table 7.3 reports the absolute errors achieved by the gradient descent method when varying the parameter α in the interval $\{0.001, 0.01, 0.1\}$. Columns 1 and 2 of the table report the instance information: the material and the prediction variable Y_1 and Y_2. Column 3 tallies the real-experimental value of the material and column 4 (resp. column 5 and column 6) displays the absolute errors with $\alpha = 0.001$ (resp. with $\alpha = 0.01$ and with $\alpha = 0.1$).

Moreover, before fixing the final value of α, we introduce a statistical analysis on the average relative errors achieved by the three versions of the method following the variation of the value assigned to α. In the comparative study, both the *sign test* and the *Wilcoxon signed-rank test* statistics are considered. We then use the following hypothesis:

$$H_0\text{: } \mathrm{M}_{\alpha_1} \text{ - } \mathrm{M}_{\alpha_2} = \mu, \text{ where } \alpha_1 \neq \alpha_2,$$

to express that method M_{α_1} performs better than method M_{α_2} and, the hypothesis $\overline{H}_0$: method M_{α_2} is better than method M_{α_1} to express the rejection of the hypothesis H_0. Herein, the smallest the absolute error and the greater the number of better predictions, the better the corresponding version of the gradient descent method.

Table 7.3: Absolute errors vs the variation of the learning rate α

CSs	Predict	Real-Value	Absolute Errors $\alpha = 0.001$	$\alpha = 0.01$	$\alpha = 0.1$
Lime	Y_1	2.00	0.15	0.15	0.15
Lime	Y_1	0.50	0.06	0.06	0.06
Lime	Y_1	2.00	0.01	0.01	0.01
Lime	Y_1	0.33	0.03	0.04	0.04
Lime	Y_1	0.67	0.08	0.08	0.08
Lime	Y_1	0.33	0.17	0.18	0.18
Lime	Y_1	0.33	0.21	0.22	0.22
Lime	Y_1	0.67	0.10	0.11	0.11
Lime	Y_1	2.00	0.05	0.05	0.05
Lime	Y_1	1.00	0.06	0.06	0.06
Lime	Y_1	1.00	0.00	0.01	0.01
Lime	Y_1	0.67	0.07	0.07	0.07
lime	Y_2	0.50	0.41	0.40	0.40
lime	Y_2	0.75	0.02	0.01	0.01
lime	Y_2	0.50	0.26	0.25	0.25
lime	Y_2	2.00	0.38	0.35	0.35
lime	Y_2	0.50	0.24	0.23	0.23
lime	Y_2	0.50	0.61	0.65	0.65
lime	Y_2	0.50	0.42	0.46	0.46
lime	Y_2	2.00	1.21	1.23	1.23
lime	Y_2	0.75	0.26	0.26	0.26
lime	Y_2	1.00	0.04	0.04	0.04
lime	Y_2	2.00	0.17	0.14	0.14
lime	Y_2	2.00	1.00	1.01	1.01
Cement	Y_1	2.00	0.15	0.15	0.15
Cement	Y_1	0.50	0.06	0.06	0.06
Cement	Y_1	2.00	0.01	0.01	0.01
Cement	Y_1	0.33	0.03	0.04	0.04
Cement	Y_1	0.67	0.08	0.08	0.08
Cement	Y_1	0.33	0.17	0.18	0.18
Cement	Y_1	0.33	0.21	0.22	0.22
Cement	Y_1	0.67	0.10	0.11	0.11
Cement	Y_1	2.00	0.05	0.05	0.05
Cement	Y_1	1.00	0.06	0.06	0.06
Cement	Y_1	1.00	0.00	0.01	0.01
Cement	Y_1	0.67	0.07	0.07	0.07
Cement	Y_2	0.50	0.02	0.02	0.02
Cement	Y_2	0.50	0.44	0.44	0.44
Cement	Y_2	0.50	0.02	0.02	0.02
Cement	Y_2	1.00	0.06	0.06	0.06
Cement	Y_2	0.50	0.54	0.54	0.54
Cement	Y_2	0.75	0.09	0.09	0.09
Cement	Y_2	0.50	0.02	0.02	0.02
Cement	Y_2	2.00	0.87	0.87	0.87
Cement	Y_2	0.75	0.33	0.33	0.33
Cement	Y_2	0.50	0.03	0.03	0.03
Cement	Y_2	0.50	0.43	0.43	0.43
Cement	Y_2	1.00	0.33	0.33	0.33

Table 7.4: p-values for both *Sign test* and *Wilcoxon rank-test* on all samples and predicted values by varying α: the case of cement.

	$\mu^{(1)}$	$\mu^{(2)}$	$\mu^{(3)}$
p-value (Sign test)	0.536	0.536	1
N^+	60	60	56
N^-	60	60	52
p-value (Wilcoxon)	0.463	0.463	1

Table 7.4 reports, by varying α_1 and α_2 ($\alpha_1 \neq \alpha_2$), the statistical study on the cement material by using the *sign rank test* (the detailed results containing the absolute errors are those illustrated in the second part of Table 7.3). The different values related to μ represent what follows: $\mu^{(1)} = \mathrm{V}_{0.001}$ vs $\mathrm{V}_{0.01}, \mu^{(2)} = \mathrm{V}_{0.001}$ vs $\mathrm{V}_{0.1}$ and $\mu^{(3)} = \mathrm{V}_{0.01}$ vs

$V_{0.1}$. Columns from 2 to 4 display the statistical results of the method when varying α: line 1 displays the p-value corresponding to the sign test, line 2 tallies the number of times that the first version of the method dominates the second version (resp. line 3 reports the number of times that the first version is dominated by the second one) and line 4 tallies the p-value related to the rank sign test(wilcoxon). From Table 7.4, we observe what follows:

- For $\mu^{(1)}$ the p-value related to the sign test (resp. Wilcoxon rank-test) is greater to the significance level $\theta = 0.05$, indicating that $V_{0.01}$ performs better than $V_{0.001}$ (accepting the hypothesis H_0). Also, column 3 indicates that the version $V_{0.1}$ dominates $V_{0.001}$.

- Regarding the comparative study between the three versions, $V_{0.1}$ performs better than the first two versions according to Wilcoxon's p-value related to $\mu^{(3)}$ (equals to 1).

- In terms of the number of absolute errors matched by the three versions, the first version $V_{0.001}$ provides 50% of best absolute errors, as both other versions, while $V_{0.1}$ achieves 46.33% which is smaller to that provided by $V_{0.01}$.

Therefore, because we seek to find solutions with smallest absolute errors and with a greatest number of dominance, we then fix $\alpha = 0.1$ for the rest of the experimental part.

Behavior of the second version of the descent method

Herein, the behavior of the stochastic gradient descent (noted V2) is analyzed for the target variable Y_1 related to the cement material. The aim of the study is to confirm (or not) the effectiveness of that version while applied to a instance with small-sized. Hence, the method is called by fixing the parameter $\alpha = 0.1$, where the same maximum number of iterations is used. We also note the batch gradient descent method as V1, in what follows.

Table 7.5 reports the real values corresponding to the values of the real-experiment and those provided by both V1 and V2. Column 3 reports the predicted values achieved by V1 while column 4 tallies those reached by V2. Figure 7.6 illustrates the variation of the cost (the results detailed in Table 7.5) reached by both V1 and V2 when compared to the real values. From the figure, on the one hand, we can observe that the curve of V1 is quite close to the curve reflecting the results of the real experiment while that of V2 remains less close. On the other hand, the average squared error (that is the difference between the real value and the predicted value squared) is equal to 0.067 for V1 while that of V2 is equal to 1.6 times larger.

Table 7.5: Batch method versus stochastic one for predicting the value related to the target variable Y_1: the case of the cement.

Target	Real	Version 1 Prediction	Version 2 Prediction
Y_1	0.67	0.75	1.06
Y_1	0.33	0.41	0.21
Y_1	0.33	0.15	0.18
Y_1	2.00	1.80	1.94
Y_1	0.67	0.76	0.65
Y_1	0.67	1.25	1.27
Y_1	0.50	0.56	0.81
Y_1	0.50	0.80	1.18
Y_1	0.50	0.81	0.55
Y_1	0.67	0.72	0.38
Y_1	2.00	1.73	1.87
Y_1	0.33	0.22	0.48
Y_1	1.00	0.73	0.71
Y_1	0.33	0.26	0.55
Y_1	2.00	1.70	1.77
Y_1	2.00	1.74	1.79
Y_1	0.67	0.68	0.81
Y_1	0.67	0.95	0.99
Y_1	1.00	1.27	1.45
Y_1	0.33	0.24	0.33
Y_1	1.00	1.23	1.11
Y_1	0.67	0.57	0.09
Y_1	1.00	1.03	1.11
Y_1	2.00	1.55	1.60
Y_1	1.00	1.09	1.33
Y_1	0.33	0.05	-0.02
Y_1	0.67	0.93	0.95
Y_1	2.00	1.75	1.57
Y_1	1.00	1.22	1.09
Y_1	0.67	0.53	0.38
Y_1	0.67	0.98	1.38
Y_1	0.50	0.46	0.22
Y_1	0.50	0.57	0.38
Y_1	2.00	1.53	1.46
Y_1	0.33	0.40	0.51
Y_1	0.67	0.67	0.69
Y_1	0.50	0.63	0.83
Y_1	1.00	1.36	1.26
Y_1	0.50	0.67	0.77
Y_1	0.33	0.14	0.04
Y_1	1.00	1.39	1.20
Y_1	0.33	0.21	0.17
Y_1	2.00	1.75	1.97
Y_1	1.00	0.95	1.30
Y_1	0.67	0.81	0.76
Y_1	0.50	0.11	0.51
Y_1	0.50	0.34	-0.26
Y_1	0.50	0.56	0.50
Y_1	2.00	1.56	1.40
Y_1	0.50	0.72	0.86
Y_1	0.33	0.40	0.31
Y_1	1.00	1.19	1.19
Y_1	0.50	0.78	0.80
Y_1	0.33	0.29	0.32
Y_1	1.00	1.08	1.08
Y_1	2.00	1.51	1.68
Y_1	0.33	-0.01	0.25
Y_1	2.00	1.44	1.36
Y_1	2.00	1.57	1.38
Y_1	1.00	1.40	1.39

Figure 7.7 shows the variation of squared errors related to both V1 and V2, whereas expressed by both curves, we can remark that V1's curve remains better than that of V2.

As we mentioned, the batch version performs generally better than the stochastic version whenever a few number of samples is considered. In order to evaluate the behavior of both versions (a comparative study), applied to the same sample of 60 real data characterizing the cement as coating substance, we run these versions with the same

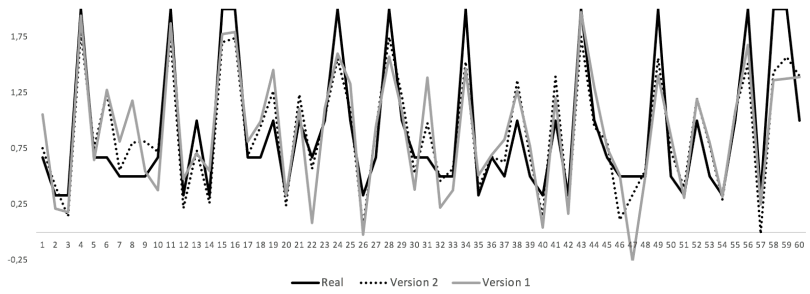


Figure 7.6: Cost fluctuation of both V1 and V2 on the training samples regarding the cement as coated substance: the case of the target variable Y_1.

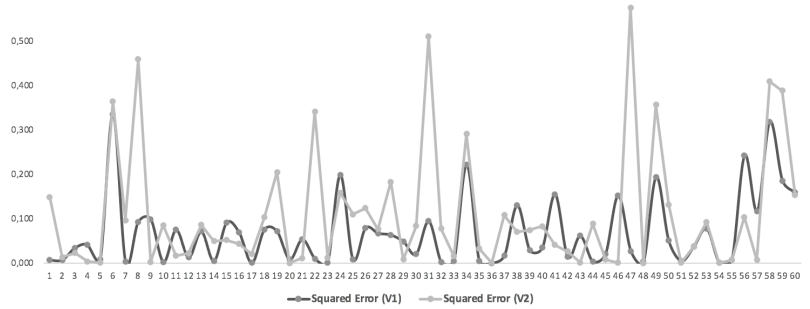
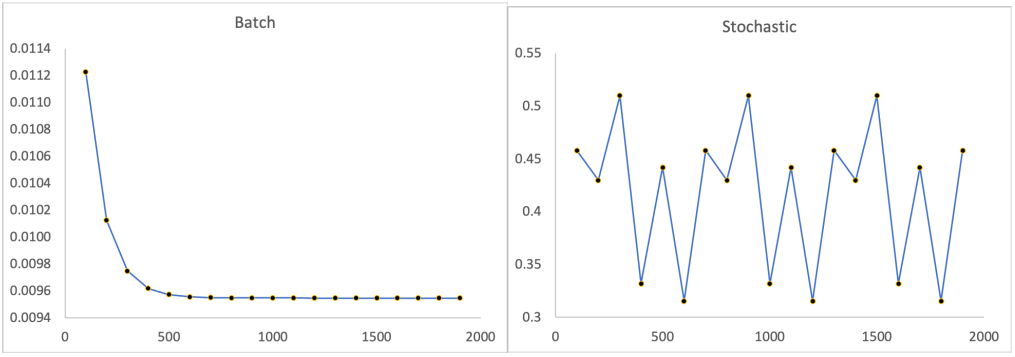


Figure 7.7: Variation of the squared error of both costs (compared to the real value) provided by both V1 and V2: the case of the target variable Y_1 (cement).



(a) Version 1: The batch gradient descent model (b) Version 2: The stochastic gradient descent model

Figure 7.8: Variation of the cost when using (a) V1 and (b) V2: the case of the target variable Y_1 (training set) for cement.

parameters. Figure 7.8 illustrates the fluctuations related to both versions V1 (on the left-hand) and V2 (on the right-hand), where V1 seems to converge smoothly towards the final solution while V2 causes a very aggressive oscillation before tending towards a plausible solution. We also note that the average runtime consumed by the batch version is equal to 0.91 seconds while the stochastic version needs 63.54 seconds.

7.5 Conclusion

In this chapter, we studied the effects of a real-experimental case of coating the flax straw before its incorporation into a cement matrix. A simple model was proposed for tackling the problem, where a batch gradient descent method was employed for predicting the parameters related to the composite compressive strength. Such a problem needs 28 days for achieving a final material while the proposed method converge to good approximate solutions by consuming less than one minute.

As the studied problem is related to a practical situation, there are plenty possibilities for further investigation involving efficient methods able to predict the parameters. First, neural networks can be applied to predict some parameters and so, in some cases they can be used for providing tight estimations. Second, the hybridization between neuronal networks and operational research technics can also be envisaged for achieving better predictions of the parameters, especially when multi-objective functions, like predicting parameters and minimizing the overall price of the raw material required, to design the target material.

Part II

Consolidating the supply facilities by combinatorial optimization: K-clustering

In this part, the k-clustering minimum completion problem is studied to consolidate the supply facilities.

Consolidating the supply facilities can be an approach to ensure continuous production and distribution of the studied vegetal coated material from the first part of this study.

Chapter 8 introduces the importance of a sustainable supply chain and reliable supply chain of the vegetal materials. Consolidating the supply facilities and the regrouping problem discussed in 8.1. The operational research and combinatorial optimization is introduced in 8.2.

K-clustering minimum bi-clique completion problem is introduced in chapter 9 with an introduction and a review of the other studies 9.1, applications and introduction to the benchmark dataset 9.2. Mathematical formulations and the applied linearization method with an example is presented in 9.3.

Chapter 10, introduce the rounding method. An enhanced algorithm is presented by incorporating the rounding search with the local branching method. The augmented version of the rounding search 10.2 and the main procedures of combining the rounding search with local branching strategy 10.2 are presented in this chapter. The benchmark dataset was used to evaluate the model. The effect of the local branching and the behavior of this model is shown in 10.3.

Chapter 11 propose a rounding strategy-based algorithm for this problem. This chapter includes an introduction to the main method 11.1, powerful enhancing solution methods 11.2. The proposed solution is evaluated through statistical analysis in section 11.4. Presented comparisons between the other methods 11.4 shows the superiority of the proposed model.

Chapter 8

Introduction

Brief table of contents

In this chapter, we will discuss the importance of sustainable and reliable supply chain of the vegetal materials. Consolidating the supply facilities using K-clustering problem is proposed. We will also introduce the operational research and combinatorial optimization.

8.1 Overview

The development of the vegetal composite requires taking into account the specific parameters of bio-sourced raw materials such as temporal availability, interchangeability, and the consequences on the resulting functional properties.

The optimization of a vegetal-based composite is confronted with a variety of characteristics of raw materials and operating parameters, the difficulty of establishing the

relationships existing between raw materials, processes and finished product, and the need to manage unexpected events (such as disasters, crises, breakdowns, ...). Thus, a reliable and sustainable system is requested in order to produce agro-composites with continuous efficiency (Baykasoğlu et al., 2004).

The sustainable optimization and production of vegetal materials also require the localization, centralization, and consolidation of the supply chain sites. Supply chain is defined as "the sequence of processes involved in the production and distribution of a commodity" (Simpson and Weiner, 1989).

Sustainability in supply chain

To achieve continuous production and online marketing it is necessary to ensure reliability in all the supply chain parts. Figure 8.1 represents an example of a supply chain. In general a supply chain system consists of three main parts: supply, production, and distribution facilities.

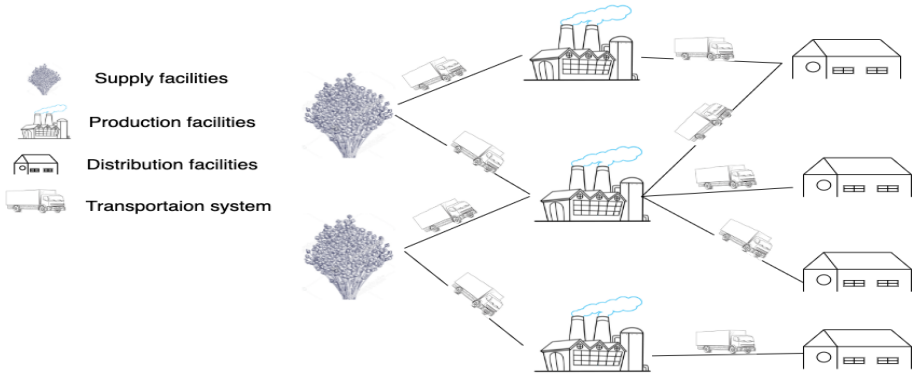


Figure 8.1: An example of a supply chain system

The supply chain decisions consist of the strategic (long term) decisions to create a system, midterm and short term decisions to maintain an existing system. The main supply chain problems are localizing the production sites, routing, and scheduling, storage of raw materials, inventory management, optimization of the transportation systems, distribution and marketing.

Dealing with these problems, it is important for companies to build a reliable business model and a competitive purchasing and marketing strategy. The modeling of the information flows related to the problems of location and planning of a production unit on an urban and rural transport network, generates very complex system (Daskin et al., 2005).

The localization and consolidation issues are in the area of operations research and

decision-making. On the one hand, a localization problem leads to a single and multi-criteria modeling as well as the proposition of efficient resolution methods. On the other hand, extending the model to a robust variant can be attempted according to the scarcity of the materials needed for the manufacture of vegetal composite and the criteria related to the decision-makers.

The production of the vegetal composites is highly sensitive. The supply materials have to be available fully in a certain place (x) at a certain time (t). Due to the super sensitivity of the vegetal materials, it is highly important to consider a backup strategy to choose the supply chain centers.

An element that can be considered to increase the robustness of an existing supply chain is to consider the possibility of consolidating the supply system. Considering merging the supply centers is an element that can increase the sustainability and robustness of the system in case of losing a center due to an emergency shut down, disaster, or economic reasons.

Consolidating the supply facilities

Consolidating the facilities in any part of the supply chain whether production, distribution, and retailing centers is an alternative solution to reduce the inventory and retail costs (Teo et al., 2001; van der Vlist and Broekmeulen, 2006).

Regrouping, consolidating, or clustering are referred to the act of merging two or more sits (Koch and Wäscher, 2016). It is the act of replacing a center (or centers) with a smaller number of existing centers. In order to keep the continuous efficiency in the supply and production chains, each site has to serve the whole services that used to be served by the replaced sites (Mansour et al., 2004).

Regrouping examples can be not only in supply centers but also in educational sites to regroup public schools into a smaller number of existing schools, warehouses, telecommunication systems, shops, distribution centers, etc.

The regrouping problem can be seen in any part of the supply chain. Consolidating the supply facilities can be aimed to increase the sustainability and to ensure a continuous production and also continuous or online sell (marketing).

Herein, we study the k-clustering minimum bi-clique completion problem. The k-clustering problem is defined in a bipartite graph. Therefore, it can be defined in any of the two parts of the supply chain that are in direct relations. Later, we will discuss how combinatorial optimizations can be used to find solutions for the k-clustering problem.

8.2 Optimization and operational research

An optimization problem can be defined as the search for a minimum or maximum value of a given objective (mathematical function) while respecting a certain number of constraints (Collette and Siarry, 2002).

In operational research, there are two types of solution methods: (i) exact solution methods and (ii) the approximate solutions. The quality of a solution method is usually evaluated based on the value of their objective function and their calculation time.

The exact solution methods ensure to find the optimum solution (Hifi, 2001). A most popular way to achieve an exact solution is to find the optimum with its mathematical modeling. Therefore, an exact solver such as Cplex as a black box can be used. From the most popular methods that can reach to an exact solution method, we can name the branch and bound(Cung, Hifi, and Le Cun, 2000; Hifi, 1998), branch and cut/price algorithm (Barnhart, Johnson, Nemhauser, Savelsbergh, and Vance, 1998), column/row generation (Lübbecke and Desrosiers, 2005), Lagrangian relaxations and state of the art studies (Desaulniers, Desrosiers, and Spoorendonk, 2010). However, if the computation time increases exponentially, for large size instances, the exact solution methods are not appropriate.

The approximate solution methods, can provide near or optimum solutions. They can not ensure to provide the optimum solutions, but they can find relatively good solutions in reasonable time. Heuristics and meta heuristic methods are sub filed of approximate solution methods.

In one hand, heuristics methods depends on the problem. They are created with the particularities of the problem at hand. Since heuristic solutions try to find better solutions in a greedy manner, they usually fall in a local optima. However, diversification technics expand their searching area and improve the quality of the solutions.

Large Neighborhood Search (LNS) is a heuristic solution method. It is applied in several combinatorial optimization problems such as vehicle routing problem (Shaw, 1998). The LNS is based on the concepts of building and exploring a neighborhood; that is, a neighborhood defined implicitly by a destroy and a repair procedure.

Similarly, local search methods are applied to intensify a search in a certain area. Local search can be enhanced with diversifying technics (Goullieux, Hifi, and Sadeghsa, 2020b; Hifi, Michrafy, and Sbihi, 2006).

Meta-heuristics, on the other hand, are independent of the problem. As such, they do not take advantage of any specificity of the problem and, therefore, can be used as black boxes. In general, they are not greedy. In fact, they may even accept a temporary deterioration of the solution, which allows them to explore more thoroughly the solution

space and thus to get a hopefully better solution (that will sometimes coincide with the global optimum).

Greedy randomized adaptive search procedure (GRASP) is a meta heuristic approach for the combinatorial optimization problems. The GRASP algorithm initially construct solutions. At each iteration one element is selected randomly from a restricted candidate list (RCL) and added to the current solution. The RCL is a set of high quality ordered elements with respect to a greedy function, which estimates the benefit of element selection based on the current solution. The RCL is then updated by adding a new element. The construction process continuous until a feasible solution is found. Later local search is applied to explore the neighborhoods of the current solution (Feo and Resende, 1995).

The simulated-annealing method(Hifi, Paschos, Zissimopoulos, et al., 1994), genetic algorithms (Hifi, 1997), ant colonies (El Baz, Hifi, Wu, and Shi, 2016), particle swarm optimization (Dahmani, Ferroum, Hifi, and Sadeghsa, 2020) are the examples of meta heuristic methods.

Please note that although a meta-heuristic is a problem-independent technique, it is nonetheless necessary to do some fine-tuning of its intrinsic parameters in order to adapt the technique to the problem at hand.

The approximate solution methods can be mono-solution or population based. If the algorithm creates and improves one solution at hand, it is mono-solution based. Otherwise, an algorithm can find the best solution by manipulating several solutions simultaneously.

8.3 Combinatorial optimization

Combinatorial optimization is a branch of operational research and algorithm theory. The combinatorial optimization problems are problems with discrete domains that can be modeled with integer programming. Such problems are defined in a graph structure. The aim of the combinatorial optimization problems is to find the best solution from a set of discrete objects.

Typically, these problems are complex and exhaustive search can not reach the optimum solution in a real time. Thus, heuristics and meta heuristic methods are used.

Knapsack (Al-Douri, 2018), traveling salesman problem (Reinelt, 1991), sphere placement problem(Yousef, 2017), and K-clustering (Moussa, 2015) are the examples of discrete models that can be considered within the combinatorial optimization problems. Since these methods are usually complex, exhaustive search methods can not be used.

8.4 Conclusion

In this chapter we discussed the importance of a reliable supply chain in optimization and production of the vegetal materials. We proposed consolidating the supply facilities as an alternative solution that ensures a reliable system with continuous production. An introduction to operational research and combinatorial optimization is presented.

Chapter 9

k-Clustering Minimum Biclique Completion Problem

Brief table of contents

In this chapter, we will introduce the k-Clustering Minimum Biclique Completion Problem. The study of the literature is presented in 9.2. The proposed methods are evaluated with a benchmark dataset presented in 9.2. The mathematical formulations and its linearization method is presented with a numerical example in 9.3.

9.1 Introduction

The *k-Clustering Minimum Bi-clique Completion* problem (noted P_{KCMBC}) splits an undirected bipartite graph into k clusters, with the aim of minimizing the sum of the edges that need to be added to each cluster to make a complete bipartite sub graph.

$G(S, C, E)$ denotes a bipartite graph whose partition has the parts S and C, with E denoting the edges of the graph. In a complete bipartite graph E connects every vertex in S with all vertices in C. It follows that a complete bipartite graph has $|S| = |C|$ edges.

$KCMBC$ divides the initial graph into k complete bipartite sub graphs such that each vertex i in S will appear only in one of the clusters. $\bar{E}$ denotes the set of edges that complete the current graph. i.e $|\bar{E}| \cup |E| = |S| \times |E|$.

To summarize, the $KCMBC$ finds k complete bipartite clusters (sub graphs) such that:

1. $\forall i \in S$ (S denotes the first set of vertices of G), it appears once in one cluster,
2. $\forall j \in C$ (C denotes the second set of vertices of G), it appears in each cluster in which at least one of its neighbors appears,
3. the total number of edges to be added to each subgraph to become a complete bipartite subgraph is minimized.

9.2 Overview on the studied solution methods in the literature

The hardness of the problem was discussed in Faure et al., 2007. The authors proved that the $KCMBC$ is NP-hard which makes its exact resolution intractable in a reasonable runtime.

The complexity of P_{KCMBC} is also investigated in Duginov, 2015. This study explicitly stated that P_{KCMBC} is Np-hard. It implies that exact methods are not able to find the optimum solution for the instances in a reasonable time.

Thus, approximate solution methods can be preferred for tackling difficult and large-scale instances. Deterministic and stochastic heuristics are often used to approximately solve P_{KCMBC} which can achieve relatively good solutions for small and medium instances while provide relatively moderate quality for large-scale ones.

There are several studies in the literature that address a comparable solution method for P_{KCMBC}.

The $KCMBC$ problem has been initially studied in Faure et al., 2007, where the authors proved its NP-hardness and designed a mathematical model and, tailored exact column generation-based method. They adapted their proposed method for approximately solving large-scale problem instances. In their paper, the authors also underlined the non-approximability of the problem.

Gualandi, 2009 designed a hybrid algorithm that combined both constraint programming and semidefinite programming (the used mathematical model was already discussed in Faure et al., 2007. Gualandi et al., 2013 proposed a branch-and-price algorithm, where they showed that the method was able to reach optimal bounds, especially for small-sized instances. The authors tried to enhance the upper bounds by incorporating the Cplex solver.

In another related study, Hifi, Moussa, Saadi, and Saleh, 2015 designed an adaptive neighborhood search algorithm, where both intensification and diversification strategies where combined in order to either diversify the solution process or to explore the search space. Both strategies are mainly based upon a greedy random walk procedure where two phases were implemented.

There are other studies that provide solution methods for the $KCMBC$, such as the study of Al-Iedani *et al.* (Al-Iedani, Hifi, and Saadi, 2016) where they proposed a variable neighborhood search heuristic. Their proposed method starts with an initial solution. Next, an iterative search was applied in order to improve a series of current solution at hand. In the experimental part, the authors demonstrated experimentally the superiority of such an approach when comparing its provided results to those achieved by all existing methods of the literature.

Salah and Hifi (Saleh and Hifi, 2017) developed a quick alternative iterative procedure for $KCMBC$ problem, where a series of sub-solution spaces were explored using a simple tabu search, that is combined with degrading and rebuilding strategies. The performance of the method was evaluated on the benchmark instances extracted from the literature, whereby the provided results are compared to those reached by the Cplex solver and available results of the literature, especially on small and medium instances.

More recently, $KCMBC$ has been tackled with an iterative rounding strategy-based algorithm (Hifi and Sadeghsa, 2020a), in which it has been demonstrated the efficiency of such a method in particular for instances of reasonable size.

The authors also proposed a solution based on the local branching on the rounding method(Hifi and Sadeghsa, 2020b).

Hifi and Sadeghsa (Hifi and Sadeghsa, 2020a) proposed an adaptation of the rounding strategy-based algorithm for approximately solving $KCMBC$ problem. Their method is based upon the drop-and-fix strategy to diversify the search process, where that

method included (i) a rounding method that is based on rounding a series of fractional variables (as used in Hifi Hifi, 2013), (ii) a greedy-random initialization procedure used for providing a quick initial solution for the problem and, (iii) an iterative search embedding neighbor operators, like intensification and diversification, for highlighting the quality of the solutions. The preliminary experimental part showed that the rounding algorithm remains competitive and is able to match the best solution values available in the literature.

The applications of the clustering problem

$KCMBC$ arises in a variety of the real-world applications such as flexible manufacturing systems, commercial and telecommunication fields.

The different variants of bi-clique problems were addressed in the theoretical studies such as the bi-clique vertex coloring problem and the star vertex coloring problem (the reader can refer to (Dantas et al., 2007).

Regrouping, consolidating or clustering are referred to the act of merging two or more sits (Koch and Wäscher, 2016). It is the act of replacing a center (or centers) with a smaller number of existing centers. In order to keep the continuous efficiency in the supply and production chains, each site has to serve the whole services that used to be served by the replaced sites (Mansour et al., 2004).

Consolidating the supply, production, distribution, and retailing centers is an alternative solution to reduce the inventory and retail costs (Teo et al., 2001; van der Vlist and Broekmeulen, 2006).

Regrouping examples cannot be only in supply centers but also in educational sites to regroup public schools into a smaller number of existing schools, warehouses, telecommunication systems, shops, distribution centers, etc.

In logistics, completion problems can also be seen in batching a group of distribution warehouses. In the study of Koch and Wäscher, 2011, the authors considered grouping the order of different customers into a picking patch such that they collect all the requested demands. This study aims to propose solution methods that are applied to the completion problem in a bipartite graph using operational research techniques.

$KCMBC$ was studied in Faure et al., 2007, where authors described an application related to the telecommunication domain. Indeed, in order to bound some given channels in multicast transmissions, according to a set of clients' demand related to services, k multicast sessions should be build to iclude independent demands. On the one hand, a service is contained in a single multicast session and, on the other hand, a client may appear in several sessions.

Finally, a bipartite graph G', $G' \subseteq G$, is defined as a multicast session, which links c times an unsolicited service, where c denotes to the number of edges that must be added to G' to reach a bi-clique. In this case, the cost c denotes a measure of the "waste of bandwidth" of the corresponding multicast session to be minimized.

The benchmark data set

This thesis used the benchmark dataset that was introduced in Gualandi et al., 2013. Later other studies in the literature applied the same dataset on the different proposed solution methods (Hifi et al., 2015,Al-Iedani et al., 2016).

The dataset is divided in three families depending on the number of the centers.

(i) the first set contains 9 small-sized instances, extracted from (Gualandi et al., 2013), where their optimal values are known,

(ii) the second set includes 18 medium and large-scale instances

(ii) the third set includes 36 huge instances.

Note that for the first set, the number of the services (resp. clients) related to S (resp. C) varies in the discrete interval$\{15, 18, 20\}$, $|S| = |C|$, the number of the clusters n_k varies in the discrete interval $\{3, 4, 5\}$ and, the density of the bipartite graph $d$was generated using the following formula: $d = |E| \div |S| \times |C|$; that varies in the interval $\{0.3, 0.5, 0.7\}$.

The second and third set extracted from (Hifi et al., 2015) were generated from the instances of Gualandi et al., 2013.

For the second set, the number of services (resp. clients) of S (resp. C) varies in the discrete interval $\{50, 80, 100\}$ while all other parameters are the same. The proposed method was coded in C++ and run on the Intel Pentium Core i7 with 2.1 GHz.

In the instances of the third set the number of services (resp. clients) of S (resp. C) varies in the discrete interval $\{120, 150, 200, 300\}$ where the number of the clusters n_k varies in the discrete interval $\{5, 10, 15\}$.

The aim of P_{KCMBC} is to minimize the number of the added edges to the sessions in order to transform them into a complete sub-bipartite graph. Note that in each cluster, a client from C can be served by at least one service from S. Thus a service s can appear only in one session, however, a customer c can benefit from several sessions.

9.3 Mathematical formulations

The described $KCMBC$ problem is presented with mathematical formulations. The initial mathematical formulation for $KCMBC$ is proposed in Gualandi et al., 2013. Since, the model is nonlinear a linear model is proposed.

The quadratic model

First, let the decision variable x_{ik} equal to 1 if node i in partition k, $k \in S$, 0 otherwise. Second, set y_{ik} equal to 1 if node j, $j \in C$, is assigned to the cluster k, 0 otherwise. Thus, the formal description of $KCMBC$ (noted $\mathrm{Q}_{k\mathrm{CMBC}}$) is described as follows (Faure et al., 2007):

$$\sum_{k \in K} \sum_{(i,j) \in \bar{E}} x_{ik} y_{ik} \tag{9.1}$$

$$\text{s.t.} \quad \sum_{k \in K} x_{ik} = 1, \forall i \in S \tag{9.2}$$

$$x_{ik} \leq y_{jk}, \forall (i,j) \in E, \forall k \in K, \tag{9.3}$$

$$x_{ik} \in \{0,1\},\ y_{jk} \in \{0,1\}, \forall i \in S, \forall j \in C, \forall k \in K, \tag{9.4}$$

Where the objective function (9.1) minimizes the number of edges added to the bipartite subgraph that inducing a complete bipartite subgraph. Each constraint (9.2) ensures the assignment of a node i belonging to S to only one cluster and, the each second constraint (9.3) indicates that each node j belonging to C is assigned to the same cluster as its corresponding node in S if there is any edge of E that connects the couple of nodes (i, j).

Integer linear programming

Although the quadratic formulation (*1a - 1d*) can be solved with quadratic programming solvers, the model can be converted to linear formulation by defining $z_{ijk} = x_{ik} y_{ik}$.

The nonlinear expression of the quadratic model (equation 9.1) occurs in the objective function at which the two binary variables $x_{ik} y_{jk}$ are multiplied. A classical linearization method for such a nonlinear model is to replace both decision variables x_{ik} and y_{jk} with the new binary variable z_{ijk} ($z_{ijk} = x_{ik} y_{ik}$). Herein, we might need to add the following linear constraints:

$$z_{ijk} \leq x_{ik} \quad \text{and} \quad z_{ijk} \leq y_{jk} \quad \forall (i,j) \in \bar{E}, \forall k \in K \tag{9.5}$$

$$x_{ik} + y_{jk} \leq 1 + z_{ijk}, \quad \forall (i,j) \in \bar{E}, \forall k \in K \tag{9.6}$$

However, considering the existing constraints in the model 9.1 the equation 9.5 is always satisfied. The formal reformulation, denoted P'_{KCMBC}, is described as follows:

$$\omega(x) = \min \sum_{k\in K} \sum_{(i,j)\in \bar{E}} z_{ijk} \tag{9.7}$$

$$x_{ik} + y_{jk} \leq 1 + z_{ijk}, \quad \forall (i,j) \in \bar{E}, \forall k \in K \tag{9.8}$$

$$\text{s.t.} \sum_{k\in K} x_{ik} = 1, \quad \forall i \in S$$

$$x_{ik} \leq y_{jk}, \quad \forall (i,j) \in E, \forall k \in K \tag{9.9}$$

$$x_{ik}, y_{jk} \in \{0,1\}, \quad \forall i \in S, \forall j \in C, \forall k \in K, \tag{9.10}$$

$$z_{ijk} \in \{0,1\}, \quad \forall (i,j) \in \bar{E}, \forall k \in K,$$

Note that $\overline{E}$ denotes the set of complementary links between services of S and clients of C. It means that it is composed with the edges that are not in the set E. One can observe that P'_{KCMBC} remains difficult to solve optimally. In this case, the exact solvers may converge very slow for small and medium-sized instances. Thus, It is not possible to solve larger instances with reasonable runtime).

Consequently, instead of solving either the quadratic or linear versions of the model, one can start by solving their respective linear relaxations for building a near-optimal solution regarding the structure provided by any used black-box solver or tailored heuristic. Conventionally, to relax a binary model, all decision variables must be in the continuous interval $[0, 1]$. However, herein relaxing the variables related to the x and y remains sufficient for the proposed method, which induces the following relaxed model (noted MIP_{kCMBC}):

$$\overline{z}(x) = \quad \sum_{k\in K} \sum_{(i,j)\in \bar{E}} z_{ijk} \tag{9.11}$$

$$\text{s.t.} \quad x_{ik} + y_{jk} \leq 1 + z_{ijk}, \quad \forall (i,j) \in \bar{E}, \forall k \in K \tag{9.12}$$

$$\sum_{k\in K} x_{ik} = 1, \quad \forall i \in S \tag{9.13}$$

$$x_{ik} \leq y_{jk}, \quad \forall (i,j) \in E, \forall k \in K \tag{9.14}$$

$$0 \leq x_{ik} \leq 1, \quad 0 \leq y_{jk} \leq 1, \quad \forall i \in S, \forall j \in C, \forall k \in K \tag{9.15}$$

$$z_{ijk} \in \{0,1\}, \quad \forall (i,j) \in \bar{E}, \forall k \in K. \tag{9.16}$$

An example of the K-clustering problem

We consider an example of K-clustering problem. In this example, we consider a bipartite graph with 4 vertices in services (S) and 5 vertices in clients (C). It is required to merge the service centers so that there will be 2 clusters ($K = 2$). Each new cluster has to serve the clients without loosing anyone. By this mean new edges must be added so that the new clusters be a complete bipartite graph.

Figure 9.1 pictures an initial graph with the two equal solutions. In Figure 9.1(i), an example of a bipartite graph $G(S, C, K)$ is demonstrated where $S = \{a, b, c, d\}$, $C = \{1, 2, 3, 4, 5\}$, and $K = 2$.

Figure 9.1(ii)and(iii) are the two different solutions with equal objective values. The boxes around each node of S, represents the clusters. As in the Figure 9.1, to obtain the objective value for each solution it is sufficient to assign each node $i \in S$ in a cluster k. The number near a vertex in S represents the number of edges leaving the vertex.

In solution (ii) the number (2+1) near node b in S means 2 edge that leaves node b from the initial graph plus one additional edge to make a complete sub-graph.

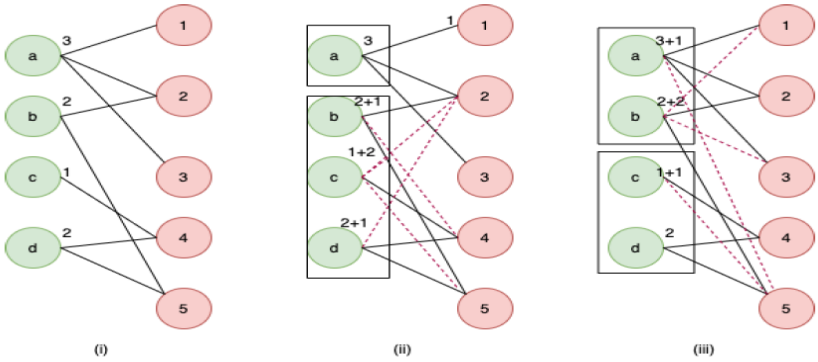


Figure 9.1: Example of $KCMBC$: (i) is an initial network with 4 vertices in S, 5 vertices in C, (ii) and (iii) are the equal solutions with the objective value equal to 4

9.4 Conclusions

In this chapter, we presented the K-clustering Minimum Biclique Completion problem. The solution methods in the literature is studied. The benchmark dataset used in the studied methods is presented. Mathematical formulations and the applied linearization model of the model is presented with a numerical example.

Chapter 10

Effect of local branching on the iterative rounding-based method: The case of k-clustering minimum completion problems

Brief table of contents

In this chapter, we study the effect of the local branching strategy on the iterative rounding solution procedure, especially when tackling the k-clustering minimum completion problem. The designed method is based upon three features: (i) proposing an enhancing basic rounding procedure for providing a starting / current solution, (ii) dropping a part of the structure of the current solution and completing it with an extended local search combined with local branching strategy and, (iii) embedding the aforementioned steps into an iterative search. The performance of the proposed method is evaluated on benchmark instances of the literature, where its achieved results are compared to those reached by the state-of-the-art Cplex solver and the best methods available in the literature. Encouraging results have been obtained.

10.1 Introduction

In the literature, deterministic and stochastic heuristics are often used to approximately solve the stated problem, which generally achieved solutions with relatively good qualities for small and medium instances while provide relatively moderate quality for large-scale ones. More recently, $KCMBC$ has been tackled with an iterative rounding strategy-based algorithm (Hifi and Sadeghsa, 2020a), in which it has been demonstrated the efficiency of such a method in particular for instances of reasonable size.

Hifi and Sadeghsa, 2020a proposed an adaptation of the rounding strategy-based algorithm for approximately solving $KCMBC$ problem. Their method is based upon the drop-and-fix strategy to diversify the search process, where that method included (i) a rounding method that is based on rounding a series of fractional variables (as used in Hifi, 2013), (ii) a greedy-random initialization procedure used for providing a quick initial solution for the problem and, (iii) an iterative search embedding neighbor operators, like intensification and diversification, for highlighting the quality of the solutions. The preliminary experimental part showed that the rounding algorithm remains competitive and is able to match the best solution values available in the literature.

Herein, a new version of the rounding strategy-based algorithm is proposed to approximately solve $KCMBC$. The algorithm can be viewed as a cooperative method, where the rounding strategy (Hifi, 2013), which incorporates *local branching strategy*, is tailored for $KCMBC$.

In recent years, it has been observed that local branching when combined with both *black-box solver* and *branch and solve* procedures have been successful in solving several variants of combinatorial optimization problems (Aider, Gacem, and Hifi, 2020, Cherfi and Hifi, 2009, Fischetti and Lodi, 2003, and Hill and Voß, 2018.)

However, the difficulty that can be encountered for this type of resolution may be related to the runtime that can exponentially grow, especially when tackling large-sized instances. In order to overcome their heaviness, an efficient and powerful alternative approach is proposed that is able of provide solutions with hight quality and consuming a "reasonable" runtime.

In this chapter, we propose an augmented version of the rounding strategy-based algorithm, where three key features are considered:

1. A *fixing strategy* is used for positioning integral and fractional values related to some items of the relaxed problem,
2. A complementary solution procedure is called for completing the current partial solution.
3. A neighboring strategy is used for improving the quality of the solutions at hand; that is a local branching strategy applied for highlighting the search process.

These strategies are embedded into an iterated process till satisfying a stopping criterion. The proposed algorithm is then analyzed computationally on benchmark instances of the literature, where its provided results are compared to the best bounds available in the literature.

We note that in order to make the chapter self-containing, some parts of the paper published by Hifi and Sadeghsa, 2020a may be summarized in what follows.

10.2 An augmented version of the rounding solution procedure

The Model

First, let the decision variable x_{ik} equal to 1 if node i in partition k, $k \in S$, 0 otherwise. Second, set y_{ik} equal to 1 if node j, $j \in C$, is assigned to the cluster k, 0 otherwise.

Thus, the formal description of $KCMBC$ (noted $\mathrm{Q}_{k\mathrm{CMBC}}$) is described as follows (Faure et al., 2007):

$$\sum_{k\in K}\sum_{(i,j)\in\bar{E}} x_{ik}y_{ik} \tag{10.1}$$

$$\text{s.t.} \quad \sum_{k\in K} x_{ik} = 1, \forall i \in S \tag{10.2}$$

$$x_{ik} \leq y_{jk}, \forall (i,j) \in E, \forall k \in K, \tag{10.3}$$

$$x_{ik} \in \{0,1\},\ y_{jk} \in \{0,1\}, \forall i \in S, \forall j \in C, \forall k \in K, \tag{10.4}$$

where the objective function (10.1) minimizes the number of edges added to the bipartite subgraph that inducing a complete bipartite subgraph. Each constraint (10.2) ensures the assignment of a node i belonging to S to only one cluster and, the each second constraint (10.3) indicates that each node j belonging to C is assigned to the same cluster as its corresponding node in S if there is any edge of E that connects the couple of nodes (i,j).

A restricted linear relaxation for $KCMBC$ problem

An initial configuration for $KCMBC$ can be provided by solving a linear relaxation of the original problem $P_{k\mathrm{CMBC}}$ derived from $\mathrm{Q}_{k\mathrm{CMBC}}$. On the one hand, the quadratic problem related to the objective function can be transformed to the following integer linear programming (by setting $z_{ijk} = x_{ik}y_{ik}$):

$$\sum_{k\in K}\sum_{(i,j)\in\bar{E}} z_{ijk} \tag{10.5}$$

$$\text{s.t.} \quad x_{ik} + y_{jk} \leq 1 + z_{ijk}, \quad \forall (i,j) \in \bar{E}, \forall k \in K \tag{10.6}$$

$$\sum_{k\in K} x_{ik} = 1, \quad \forall i \in S \tag{10.7}$$

$$x_{ik} \leq y_{jk}, \quad \forall (i,j) \in E, \forall k \in K \tag{10.8}$$

$$x_{ik}, y_{jk} \in \{0,1\}, \quad \forall i \in S, \forall j \in C, \forall k \in K \tag{10.9}$$

$$z_{ijk} \in \{0,1\}, \quad \forall (i,j) \in \bar{E}, \forall k \in K. \tag{10.10}$$

On the other hand, the resulting integer programming P_{kCMBC} (above) remains very complex to solve, especially when tackling large-scale problem instances. Indeed, using an exact method for solving that problem may converge very slowly; consequently, instead of solving either the quadratic or linear versions of the model, one can start by solving their respective linear relaxations for building a near-optimal solution regarding the structure provided by any used black-box solver or tailored heuristic. Note that relaxing the variables related to the x and y is sufficient for the proposed method. The proposed

relaxed model (noted MIP_{kCMBC}) is as follows:

$$\overline{z}(x) = \sum_{k\in K}\sum_{(i,j)\in\bar{E}} z_{ijk} \tag{10.11}$$

$$\text{s.t.} \quad x_{ik} + y_{jk} \leq 1 + z_{ijk}, \quad \forall (i,j) \in \bar{E}, \forall k \in K \tag{10.12}$$

$$\sum_{k\in K} x_{ik} = 1, \quad \forall i \in S \tag{10.13}$$

$$x_{ik} \leq y_{jk}, \quad \forall (i,j) \in E, \forall k \in K \tag{10.14}$$

$$0 \leq x_{ik} \leq 1, \quad 0 \leq y_{jk} \leq 1, \quad \forall i \in S, \forall j \in C, \forall k \in K \tag{10.15}$$

$$z_{ijk} \in \{0,1\}, \quad \forall (i,j) \in \bar{E}, \forall k \in K. \tag{10.16}$$

The main principle of the starting procedure

A *Basic Rounding Procedure* (BRP) is applied to the mixed linear relaxation of the original problem, namely $\text{MIP}_{k\text{CMBC}}$, described above. We recall that the relaxation is related only to both vectors x and y while the variables of z are maintained to their original domains. Then, the main steps of BRP are described in Algorithm 4.

Algorithm 4 − A Basic Rounding Procedure (BRP)

1. Set $\text{P}_{reduce} = \text{MIP}_{KCMBC}$.
2. Solve P_{reduce} using the Cplex solver, and let $(\underline{x}, \underline{y}, \underline{z})$ be the achieved solution.
3. Fix the value of each $\underline{x}_i$, if $\underline{x}_i = 1$ in $\underline{x}$.
4. Select among the non-integer variables, the variable $\underline{x}_i$ having the largest fractional value, and round it to one.
5. Remove all of the fixed variables from the current problem and let P_{reduce} be the new reduced problem.
6. Repeat steps from (2) to (6) until fixing all the elements of the original problem.
7. Retrieve all the items satisfying constraints from (10.12) to (10.14), and fix them to 1, 0 otherwise.

Finally, BRP provides a feasible integer solution for $\text{P}_{k\text{CMBC}}$. Because of the greedy nature of BRP and of the complexity of $\text{P}_{k\text{CMBC}}$'s precedence constraints (inequalities of type (10.14)), BRP may remain interesting for instances with moderate size while it may fail to achieve solutions with high quality. Indeed, on the one hand, fixing y_{jk} (service) doesn't induce a direct fixation of its containing clients (items); on the other hand, fixing a client (x_{ik}) to one imposes the fixation of its services. Hence, BRP may either fix or remove items in an aggressive way.

In order to overcome to such fixation, when considering items, the following four neighborhoods are considered. The first two operators introduce moves with a slight degree of freedom, i.e., two clusters are considered, while the last two operators are used for exploring the search by using more clusters:

1. Operator M_1: Let C_i and C_k, $i \neq k$, be two clusters, where $|C_i| \geq 2$ and, $y_i^{(1)}, y_i^{(2)} \in C_i$. Move both $y_i^{(1)}$ and $y_i^{(2)}$ from C_i to C_k and, evaluate the resulting solution.

2. Operator M_2: Let C_i and C_k, $i \neq k$, be two clusters, where $|C_i| \geq 1,\ C_k| \geq 1$ and, $y_i^{(1)} \in C_i$ and $y_k^{(1)} \in C_k$. Make an exchange between both $y_i^{(1)}$ and $y_k^{(1)}$, i.e., move $y_i^{(1)}$ from C_i to C_k and, move $y_k^{(1)}$ from C_k to C_i and, evaluate the new achieved solution.

3. Operator M_3: Let $C_\ell,\ \ell \in \{1, \ldots, n_K\}$ and (i, j, k) the indices characterizing three different clusters, such that at least two clusters contain at least two services, for instance C_i and C_j. Suppose that $y_i^{(1)} \in C_i$ and $y_j^{(1)} \in C_j$; move both $y_i^{(1)}$ (from C_i) and $y_j^{(1)}$ (from C_j) to C_k and, evaluate the new achieved solution.

4. Operator M_4: Let $C_\ell,\ \ell \in \{1, \ldots, n_K\}$ and (i, j, k) the indices characterizing three different clusters, such that at least one cluster contains at least three services, for instance C_i. Suppose that $y_i^{(1)} \in C_i$ and $y_i^{(2)} \in C_i$; either (i) move $y_i^{(1)}$ from C_i to C_j and $y_i^{(2)}$ from C_i to C_k or, (ii) move $y_i^{(2)}$ from C_i to C_j and $y_i^{(1)}$ from C_i to C_k. Evaluate the new provided solution.

Herein, both clusters and services are randomly selected, and the described operators are applied with a descent method (as described in Algorithm 5).

Algorithm 5 − A Descent Procedure (DP)

1. Let S be the solution at hand (initially reached by BRP).
   Let S_N (initially $z(S_N) = +\infty$) be the best solution related to the global neighborhood (containing all operators).
2. Repeat
   a) Randomly generate ρ from the discrete interval $\{1, \ldots, 4\}$.
   b) Apply the operator M_ρ to the current solution S and let S' be the new provided solution.
   c) If $z(S_N) > z(S')$, set $S_N := S'$, then update S with S'.
   d) Restart the search process with the new solution.
3. Repeat until the stopping criteria is performed.
4. Exit with S_N with value $z(S_N)$.

We note that, in order to provide a quick convergence of the above procedure, the method stops if the new solution hasn't improved after 100 iterations.

A rounding strategy combined with local branching

Local branching

Local Branching (LB) is a specialized optimization technique that can be viewed as an alternative to exact methods for tackling several hard combinatorial optimization problems. To the best of our knowledge, LB has been first designed by Fischetti and Lodi, 2003, where its aim is to mimics an optimal resolution of MIPs. Fischetti, Polo, and Scantamburlo, 2004, extended LB, where a two-level heuristic is considered for tackling more hardness combinatorial optimization problems. The standard LB is based upon branching conditions that are expressed through linear inequalities. Given the following Mixed Integer Programming (MIP):

$$\max \quad c^T x \tag{10.17}$$

$$Ax \leq b \tag{10.18}$$

$$x_j \geq 0 \qquad \forall j \in G,\ x_j \text{ integer} \tag{10.19}$$

$$x_k \geq 0 \qquad \forall k \in C \tag{10.20}$$

$$x_i \in \{0,\ 1\} \quad \forall i \in B \neq 0, \tag{10.21}$$

Where $N = \{1, ..., n\}$ is partitioned into the following triplet sets (B, G, C) such that B denotes the set of binary variables, and G and C representing the sets of integer and continuous variables, respectively. A starting solution $\overline{x}$, considered as the reference solution of MIP, and $k \in N^\star$, the k_{Opt} neighborhood related to $\overline{x}$ corresponds to the set of feasible solutions of MIP which satisfies the following additional local (branching) constraint:

$$\Delta(x, \bar{x}) := \sum_{j \in S}(1 - x_j) + \sum_{j \in B \setminus S} \leq k, \tag{10.22}$$

Where $S = \{j \in B \mid \bar{x} = 1\}$ and the term of the left-side of inequality (10.22) is the number of binary variables switching their values, according to $\overline{x}$, either from 1 to 0 or from 0 to 1. Assume that the cardinality of the binary set B is fixed. Than the additional constraint can be rewritten as follows:

$$\Delta(x, \bar{x}) := \sum_{j \in S}(1 - x_j) \leq k', \tag{10.23}$$

Where $k' = \frac{k}{2}$ (halved). Adding the above constraint represents a branching criterion within an enumerative scheme for a MIP. In this case, according to both configurations, the space of feasible solutions related to the current branching node can be divided according to the following two complementary constraints:

$$\Delta(x, \bar{x}) \leq k \qquad \text{or} \qquad \Delta(x, \bar{x}) \geq k + 1, \tag{10.24}$$

Where k is often provided experimentally. According to the value assigned to k, the search process may be iterated by altering between normal and local branches. A local

branch is related to a complete resolution prior to branching for solving the current problem with a normal branch. Whenever a new enhancement is reached (in the local branch), local branching can iterate the resolution with the new achieved solution, where a new constraint is added to the remaining normal branch. Following the same branching principle, the last node is divided again in two branches by adding two new disjunctive constraints, i.e.,

$$\Delta(x, \bar{x}) > k, \quad \Delta(x, \bar{x}') \leq k \quad \text{(local branch)}, \tag{10.25}$$

and

$$\Delta(x, \bar{x}) > k, \quad \Delta(x, \bar{x}') \geq k + 1 \quad \text{(normal branch)}, \tag{10.26}$$

Where $\bar{x}'$ denotes the new solution reached in the local branch.

The above search process is iterated until reaching a final solution which can be considered either as an optimal solution if all local branches are exactly solved or as an approximate solution if some stopping criteria are used for curtailing the search process.

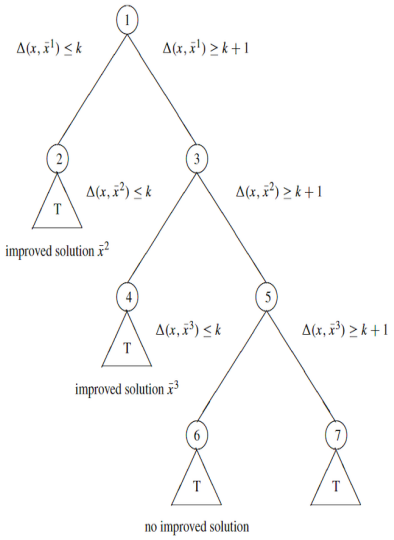


Figure 10.1: Local branching-based criterion

Fig 10.1 illustrates LB, where the triangles marked by "T" (Tactical) correspond to the branching subtrees to be explored through a standard "tactical" branching criterion such as, e.g., branching on fractional variables. The starting solution $\bar{x}^1$ denotes an incumbent solution assigned to the root node. The left-branch (node 2) corresponds to the optimization within the k_{Opt} neighborhood $\nu(\bar{x}^1, k)$, which is performed through a tactical branching scheme converging (hopefully in short computing time) to an optimal solution in the neighborhood, say $\bar{x}^2$. The last solution then becomes the new incumbent solution. The scheme is then repeated to the right-branch (node 3), where the exploration of $\nu(\bar{x}^2, k) \backslash \nu(\bar{x}^1, k)$ on the right-branch (added to node 4) provides a new incumbent solution $\bar{x}^3$. The same process is iterated till solving all subtrees hoping to reach a

final solution that can be considered either as an optimal solution or as an approximate solution one (as described above).

The dropping and re-optimizing procedure

As shown above, BRP (Algo. 4) tries to fix some decision variables step by step and, the local operators from M_1 to M_2 (using the descent procedure DP − Algo. 5) try to enhance the solution achieved by BRP. Herein, two operators are combined: (i) degrading and (ii) re-optimization strategies. These strategies are applied to escape from local optima and to diversify the search process. Indeed, the degrading strategy fixes a partial solution, according to the solution at hand, by dropping some elements while the re-optimization strategy selects new elements and adds them into the perturbed solution for reaching a new solution.

For a given solution S, the procedure starts by reducing the original problem, where a subset of variables of S are randomly fixed and, the rest of the problem with free variables is solved using a local-branching strategy. It can be summarized as follows:

Step 1: From S, fix β% of the nodes belonging to each cluster.

Step 2: Build the reduced problem, namely P'_{KCMBC} and solve it using a complementary procedure described below (Algorithm. 6).

A complementary constructive procedure works as follows (when applied to the original problem): let $C = \{c_1, \ldots, c_n\}$ (resp. $S = \{S_1, \ldots, S_m\}$) be the set of $n = |C|$ clients (resp. $m = |S|$ services), E be the set of edges (existing links) between both sets C and S from $c \in C$ to $s \in S$ (each existing link is noted (c, s)) and $nb_K = |K|$ be the number of clusters.

We note that the smaller the value of β, the larger the computational time. In this case, the search process can perturb the solutions, which may induce a slow convergence of the method. Limited computational results showed that a better value for β should be in the interval $[50,\ 90]$ which is also related to the size of the instance considered.

Combining the rounding search with local branching strategy

In recent years, it has been observed that local branching when combined with both *black-box solver* and *branch/fix and solve* procedures, like using column generation, have been successful in solving several variants of combinatorial optimization problems (Akeb et al., 2011, Cherfi and Hifi, 2009) and,Fischetti and Lodi, 2003). However, the difficulty that can be encountered when applying such type of resolution may be related to the runtime that can exponentially grow, especially when tackling large-sized instances.

Algorithm 6 − Completing a partial Solution (CS)

1. Let $K = \{K_1, \ldots, K_{n_K}\}$ be the starting n_K dummy clusters (empty clusters). Set
   a) $S' = S$ and $C' = C$;
   b) $h = 1$ (the index h associated to the first cluster).
2. Set $\delta(s_i),\ s_i \in S'$, in decreasing order, where $\delta(s_i)$ denotes the degree of the s_i-th vertex (characterizing the number of clients-nodes linked to it); Set $\ell = \arg\max_{s_i \in S'} \Big\{\delta(s_\ell) = \delta(s_1) \mid \delta(s_\ell) \geq \delta(s_i)\Big\}$;
   Set $K_h = K_h \cup \{s_\ell\}$, $S' = S' \setminus \{s_\ell\}$, and update the set C' whenever s_ℓ is affected, i.e., new sets S' and C' are recomputed.
3. For the $n_K - 1$ remaining clusters, from the cluster $h = n_K - 1$ down to $h = 2$ do:
   a) Assign a vertex $\ell' \in S'$ to the h-th cluster;
   b) Update both sets S' and C'.
4. Set $h = 1$;
   Repeat steps (2) and (4) till fixing all services in the clusters.

Exit with the best solution (S', C').

In order to overcome their heaviness, an efficient and powerful alternative approach is proposed, which we believe it is able to provide solutions with high quality and consuming a "reasonable" runtime.

Adaptation of the local branching for $KCMBC$

A standard local branching (LB) can be directly applied for tackling $\mathrm{P}_{k\mathrm{CMBC}}$, which can be described by the following points:

1. Let a starting upper bound (for a minimization problem) with its configuration; that is provided by applying BRP improved with DP.
2. Initialize the first local tree (from the first node of the three search) using the previously starting configuration.
3. Iterative Step:
   a) Solve completely the local tree (or fix a runtime limit).
   b) When the local search terminates,
      i. if a better feasible solution is reached, then create (from this local tree) a new local tree by considering the new solution achieved as the new starting solution (reference solution);
      ii. otherwise, enhancement is found; therefore, abort the local branching.
   c) Solve the rest of the search tree.
   d) Return the best solution found up to now.

Algorithm 7 – The Rounding Strategy combined with Local Branching (RSLB)

Input. An instance of $P_{k\text{CMBC}}$.

Output. A best solution $S^{\star}_{k\text{CMBC}}$.

1: Set $S^{\star}_{k\text{CMBC}} = \emptyset$
2: Apply BRP combined with DP to $P_{k\text{CMBC}}$ and let $S'_{k\text{CMBC}}$ be the provided solution.
3: **repeat**
4: Set $iter_{local} = 1$ and, let β be the fixation parameter
5: **repeat**
6: Fix randomly β variables from $S'_{k\text{CMBC}}$ and, apply the complementary procedure CP to $P'_{k\text{CMBC}}$
7: Apply local branching LB (Section 10.2) to the incubent solution $S'_{k\text{CMBC}}$
8: Improve the provided solution S_{local} and improve it with DP.
9: **if** $(z(S'_{k\text{CMBC}}) > z(S_{local}))$ **then**
10: Update $S'_{k\text{CMBC}}$ with S_{local}
11: Set $iter = 1$
12: **else**
13: $iter = iter + 1$
14: **end if**
15: Update $S^{\star}_{k\text{CMBC}}$ with $S'_{k\text{CMBC}}$, if necessary
16: **until** $\Big($stopping local condition is performed$\Big)$
17: **until** $\Big($the time limit is performed$\Big)$
18: **return** $S^{\star}_{k\text{CMBC}}$.

Injecting the local branching strategy

As underlined in Section 10.2 (Fischetti and Lodi, 2003), LB can be viewed as a specialized approach capable to reach optimal solutions for several problems. Nevertheless, using an exact solver for each branch induces a heavy algorithm, where in some cases achieving solutions with good quality becomes compromised. We then propose to combine the rounding strategy based algorithm with the local branching strategy, where the black-box solver is called for solving the complementary subproblem, as described in Section 10.2.

Algorithm 7 describes the main principle of the rounding procedure combined with local branching, where it is composed of two main loops. The internal loop (from line 6 to line 16) represents the iterative rounding procedure, where a starting solution is reached at line 3 by calling BRP combined with the descent method. For each current solution, the dropping phase coupled with the re-optimization phase is called in line 7. Then the local branching takes place at line 17 for providing a local solution which should be improved by using the descent method (line 18). The internal loop (the loop repeat from line 6 to line 16) is iterated according to the (new) solution built while the global loop (lines from 4 to 20) is iterated till matching the stopping condition.

10.3 Computational results

To evaluate the effectiveness of the proposed method on benchmark instances, three sets of instances are considered: (i) the first set contains 9 small-sized instances, extracted from Gualandi et al., 2013, where their optimal values are known, (ii) the second set includes 18 medium and large-scale instances, extracted from Hifi et al., 2015 and, (iii) the third set contains 36 large-scale instances extracted from Moussa, 2015. For the first set, the number of services (resp. clients) related to S (resp. C) varies in the discrete interval {15,18,20}, with $|S| = |C|$, the number of the clusters n_K varies in the discrete interval {3,4,5} and, the density of the bipartite graph d was generated using the following formula: $d = \frac{|E|}{|S|.|C|}$; that varies in the interval {0.3, 0.5, 0.7}. The instances of the second set were generated following Gualandi *et al.*'s generator, where the number of services (resp. clients) of S (resp. C) varies in the discrete interval {50,80,100} while all other parameters are the same. Finally, each instance of the third set has its number of services (resp. clients) related to S (resp. C) varies in the discrete interval {120,150,200,300} with the same densities as for the first two sets. All method was coded in C++ and run on the Intel Pentium Core i5 with 2.3 GHz.

Table 10.1: Solution quality of BRP-DPs' starting upper bounds on all instances

#Inst.	Best	U_{BRP-DP}	A_{BRP-DP}
15-15-5-0.3	51	51	0
15-15-5-0.5	50	50	0
15-15-5-0.7	39	39	0
18-18-5-0.3	92	92	0
18-18-5-0.5	86	86	0
18-18-5-0.7	63	63	0
20-20-5-0.3	119	119	0
20-20-5-0.5	123	123	0
20-20-5-0.7	82	82	0
Av.	78.33	78.33	0.00
50-50-5-0.3	1319	1418	7.506
50-50-5-0.5	1072	1090	1.679
50-50-5-0.7	671	683	1.788
50-50-10-0.3	938	1057	12.687
50-50-10-0.5	875	937	7.086
50-50-10-0.7	575	596	3.652
80-80-5-0.3	3815	4075	6.815
80-80-5-0.5	2862	2916	1.887
80-80-5-0.7	1768	1792	1.357
80-80-10-0.3	3202	3566	11.368
80-80-10-0.5	2571	2751	7.001
80-80-10-0.7	1618	1656	2.349
100-100-5-0.3	6248	6547	4.786
100-100-5-0.5	4650	4737	1.871
100-100-5-0.7	2842	2863	0.739
100-100-10-0.3	4298	4328	0.698
100-100-10-0.5	5427	5487	1.106
100-100-10-0.7	2644	2720	2.874
Av.	2633.06	2734.39	4.292

Behavior of BRP-DP on the first two sets

In the preliminary study, the behavior of BRP augmented with DP (noted BRP-DP) is evaluated on the instances of the first two sets. Table 11.1 reports the results achieved by BRP-DP, where column 1 shows the instance information, the best upper bound (column 2) extracted from the literature, the bound achieved by the starting procedure (displayed in column 3), and its relative percentage experimental approximation ratio $\mathrm{A}_{\mathrm{BRP-DP}} = (\frac{\mathrm{U}_{\mathrm{BRP-DP}}}{Best} - 1) \times 100$, reported in column 4 ($0 \leq \mathrm{A}_{\mathrm{BRP-DP}} \leq 1$ and the best bound (the smallest one) $\mathrm{A}_{\mathrm{BRP-DP}}$). As mentioned above, BRP solves a series of MIP and so, repeating that resolution may easily increase its runtime; herein, for the tested instances, the runtime varies from one second to five minutes. Of course, one can observe that accelerating the rounding procedure can be realized if at each step of BRP a subset of fractional variables is rounded up, but the degradation of the solution's quality can be expected too. In what follows, we comment on the results of Table 11.1:

- BRP-DP matches all best solutions when compared to those achieved by the best methods of the literature, where $|S|$ varies between 15 and 20.
- Despite the simplicity of BRP, the experimental approximation ratio varies from 0.698 (#Inst 100-100-10-0.3) and 12.687 (#Inst 50-50-10-0.3). We notice that in some cases, the approximation value is better for large instances than that for the smaller ones. Hence, it can be encouraging to predict a combination of such a procedure with other intensification and diversification strategies.
- Finally, BRP-DP's average global bound (2734.39: column 2, last line) remains interesting; in this case, BRP-DP achieves an average experimental ratio of 4.292.

Effect of the parameter β on the iterative BRP-DP

Herein, we evaluate the effect of the parameter β used when applying the dropping and re-optimizing strategies, on the second set of instances considered in Section 10.3 (In fact, because BRP-DP is able to match all better solutions for the small instances, we then considered only the instances for which the bounds may be improved when intensifying/diversifying the search process). The iterative BRP-DP (noted IBRP-DP) can be stated as follows:

1. Apply BRP-DP to $\mathrm{P}_{k\mathrm{CMBC}}$ and let $S'_{k\mathrm{CMBC}}$ be the provided solution.
   Set $S^{best}_{k\mathrm{CMBC}} = S'_{k\mathrm{CMBC}}$ and let β be the dropping parameter.
   Repeat
   a) Drop randomly β variables from $S'_{k\mathrm{CMBC}}$.

Table 10.2: Illustration of the best bounds achieved by the IBRP-DP (on instances of the second set), when varying the parameter β.

	Iterative BRP-DP: variation of β					
#Inst	50%	60%	70%	80%	90%	Best(Min)
50-50-5-0.3	1323	1319	1315	1332	1342	1315
50-50-5-0.5	1075	1088	1068	1072	1090	1068
50-50-5-0.7	672	671	671	671	683	671
50-50-10-0.3	938	950	938	957	957	938
50-50-10-0.5	886	871	875	886	875	871
50-50-10-0.7	575	585	583	575	577	575
80-80-5-0.3	3805	3815	3805	3815	3903	3805
80-80-5-0.5	2864	2864	2862	2862	2864	2862
80-80-5-0.7	1771	1768	1768	1770	1768	1768
80-80-10-0.3	3270	3219	3219	3188	3236	3188
80-80-10-0.5	2751	2570	2620	2659	2596	2570
80-80-10-0.7	1617	1617	1617	1651	1656	1617
100-100-5-0.3	6247	6315	6247	6427	6427	6247
100-100-5-0.5	4672	4650	4650	4659	4684	4650
100-100-5-0.7	2842	2836	2832	2832	2863	2832
100-100-10-0.3	4318	4318	4298	4298	4318	4298
100-100-10-0.5	5487	5426	5427	5426	5427	5426
100-100-10-0.7	2657	2644	2644	2650	2720	2644
Min	575	585	583	575	577	
Av	2653.89	*2640.33*	**2635.50**	2651.67	2665.89	
Max	6247	6315	6247	6427	6427	
St Dev	1715.17	1714.62	1706.38	1726.07	1732.02	
Nb_Best	5	*8*	**13**	7	1	

b) Apply the constructive procedure CP to $P'_{k\text{CMBC}}$ (the reduced problem) and let S_{Comp} be its achieved solution.

c) Let S_{local} be the local solution reached (containing the first partial solution and the complementary one related to the reduced problem).

d) Apply the descent procedure DP to S_{local}.

   i. Update $S'_{k\text{CMBC}}$ with S_{local}, if $z(S'_{k\text{CMBC}}) > z(S_{local})$.

   ii. Update $S^{best}_{k\text{CMBC}}$, if necessary.

Until the *stopping condition is performed*.

2. Return $S^{best}_{k\text{CMBC}}$.

The IBRP-DP (repeating IBRP with both dropping and re-optimizing phases) needs to tune (i) the stopping criterion used by the internal loop (the number of times that each current solution S is dropped and re-optimized), and (ii) the number of items to drop (or to fix, if using the dual problem). Because the proposed method applies local branching, we first considered the stopping criterium as a maximum runtime limit fixed to one minute (of course, augmenting that limit may induce a better value at the expense of the runtime). Second, the value related to the dropping strategy was fixed from 50% to 90%, by a step of 10%.

Table 10.2 reports the bounds achieved by the IBRP-DP when varying β. Column 1 refers to the instance's information (because no new solutions have been provided for the first set, we only tested the instances of the second set), column 2 represents the best objective value and columns from 3 to 7, under the value assigned to β, display the bounds reached by IRBP-DP. The last two lines of the table (Average and NB_Best) tally the global average value and the number of times that corresponding method provides a better bound, over all tested instances. From Table 10.2, we observe what follows:

- The best global average value is achieved for $\beta = 70$ (column 4, both lines Average and Nb_Best).
- The value $\beta = 60$ seems to be the value that provides closest bounds to those reached by IBRP-DP with $\beta = 70$. Indeed, in this case, IBRP-DP provides a global average bound equals to 2640.33 (the second block of Table 10.2, line Average, column 3). In term of gap, IBRP-DP's global average bound (with $\beta = 70$) achieves an average gap of 4.83.
- For small values of β as well as for large values of β, one can observe that IBRP-ID's global average bound degrades, when compared to that achieved with $\beta = 70$ (even for $\beta = 60$).
- IBRP-DP with $\beta = 70$ provides 13 better bounds (column 4, line NB_Best) while the maximum number of better bounds achieved by the best value of β, over the rest of the variations, is equal to 8 (column 3, line Av).

Effect of the local branching strategy

In this section, the performance of the proposed method that combines IBRP-DP and LB (noted IBRP-DP$_{\text{LB}}$) is evaluated on the last two sets containing medium and large-scale instances, as used in Section 10.3 (`Set II`) and in Moussa, 2015 (`Set III`). Because the proposed method uses a series of local-branching constraints, we then report its behavior by varying the parameter k. In this case, the goal of this analysis is to fix the average best value related to the parameter k whenever the constraints are injected into the reduced problem.

Table 10.3 reports the upper bounds provided by the direct adaptation of the *Local Branching*, as a method, for solving P$_{k\text{CMBC}}$ (noted LB) when varying the value of the parameter k in the discrete interval $\{5, 10, 15, 20, 25\}$ with a runtime limit of five minutes. Of course, we believe that more the runtime limit is augmented, more the quality of the solutions reached can be improved. However, as we use LB in a search procedure, we only limit its use to a few iterations, explaining then the suggested choice. Column 1 of the table refers to the instance's information (instances belonging to the third set (Set III), column 2 rallies the best upper bound provided by LB over the five values of k, columns

Table 10.3: Illustration of the best bounds achieved by LB when varying the parameter k.

		Standard Local Branching LB: variation of k				
#Inst	LB_{UB}	5	10	15	20	25
50-50-5-0.3	1452	1462	1452	1488	1452	1460
50-50-5-0.5	1092	1103	1128	1092	1092	1120
50-50-10-0.3	1073	1225	1225	1331	1088	1073
50-50-10-0.5	957	984	964	957	957	988
80-80-5-0.3	4108	4210	4248	4193	4108	4108
80-80-5-0.5	2951	2951	2961	2969	2961	2953
80-80-10-0.3	3903	3958	3903	3958	3958	3937
80-80-10-0.5	2758	2758	2799	2809	2763	2803
100-100-5-0.3	6638	6721	6735	6638	6638	6702
100-100-5-0.5	4740	4763	4740	4763	4763	4774
100-100-10-0.3	6390	6392	6412	6390	6390	6405
100-100-10-0.5	9097	9128	9140	9190	9097	9114
Av	3763.25	3804.58	3808.92	3814.83	**3772.25**	*3786.42*
Nb_Best		2	3	4	**7**	2

from 3 to 7, under the value assigned to k, display the upper bound (feasible solution) achieved by LB. Finally, the last two lines summarize the global average bounds achieved by the method according to the value of k and, the number of times the corresponding version (according to k) matches the best upper bound.

From Table 10.3, one can observe that,

- The best global average value (Line Av), over all tested instances, is achieved for $k = 20$ (3772.25: in bold-space).
- For $k = 25$, the method achieves very close value (i.e., 3786.42) to that reached by LB with $k = 25$; however, the number of matched best upper bounds is smaller. Indeed, for $k = 20$, this version of LB matches a greatest number of better values (representing a percentage of 58.33% of solutions) while for the version with $k = 25$ only two instances have been matched (representing a percentage of 16.67%).
- For the other values assigned to k, the overall average values are of lower quality.

Because we seek solutions with high quality and with more best bounds, we then fix $k = 20\%$ for the rest of the experimental part.

Behavior of IBRSP-DP with local branching strategy

In this section, the behavior of IBRP-DP with local branching (noted IBRP-DP_{LB}) is analyzed on the last two sets of instances (Set II and Set III) containing medium and large-sized instances. Its achieved results are compared to those reported in Hifi et al., 2015: an Adaptive Neighborhood Search (noted ANS), the Neighborhood Search

Heuristic (Moussa, 2015) (noted NSH) and the Cplex solver, that solver was tested using several tunings (each version was fixed to one hour): (i) automatic search method, (ii) dynamic search, and (iii) traditional branch-and-cut search (for each of these versions, the RINS heuristic was fixed to 100); hence, the best objective value achieved by these versions are retuned as the best solution value of Cplex. We note that IBRP-DP$_{\mathrm{LB}}$'s runtime was fixed to thirty minutes for instances of Set II and to sixty minutes for more largest instances of Set III.

Table 10.4: Solution quality of BRP-DP$_{\mathrm{LB}}$s' versus other methods of the literature (Set II and Set III)

#Inst.	Best$_{\mathrm{Cplex}}$	Best$_{\mathrm{ANS}}$	Best$_{\mathrm{NSH}}$	Best$_{\mathrm{New}}$
50-50-5-0.3	1423	1319	1319	1315
50-50-5-0.5	1091	1072	1070	1068
50-50-5-0.7	676	671	671	671
50-50-10-0.3	1135	938	938	938
50-50-10-0.5	930	875	875	871
50-50-10-0.7	578	575	575	575
80-80-5-0.3	4166	3815	3815	3805
80-80-5-0.5	2931	2862	2862	2862
80-80-5-0.7	1775	1768	1768	1768
80-80-10-0.3	3735	3202	3202	3188
80-80-10-0.5	2675	2571	2571	2571
80-80-10-0.7	1634	1618	1618	1617
100-100-5-0.3	6645	6248	6248	6247
100-100-5-0.5	4716	4650	4650	4650
100-100-5-0.7	2854	2842	2842	2836
100-100-10-0.3	6119	4298	4297	4297
100-100-10-0.5	9111	5427	5427	5426
100-100-10-0.7	2704	2644	2644	2644
120-120-5-0.3	8420		8415	8357
120-120-5-0.5	6106		6019	6019
120-120-5-0.7	4344		4264	4264
120-120-10-0.3	8205		8195	8036
120-120-10-0.5	5841		5821	5821
120-120-10-0.7	4141		4063	4063
120-120-15-0.3	8859		7808	7675
120-120-15-0.5	5908		5728	5562
120-120-15-0.7	4388		3913	3913
150-150-5-0.3	14547		14541	14434
150-150-5-0.5	2136		2005	1996
150-150-5-0.7	8594		8480	8480
150-150-10-0.3	14821		14095	14095
150-150-10-0.5	1687		1528	1528
150-150-10-0.7	8810		8127	8127
150-150-15-0.3	16102		15312	13646
150-150-15-0.5	11583		11082	10937
150-150-15-0.7	9596		8971	7988
200-200-5-0.3	29353		29318	28770
200-200-5-0.5	19518		19456	19456
200-200-5-0.7	19829		19457	19457
200-200-10-0.3	29701		28374	28024
200-200-10-0.5	19931		18989	18985
200-200-10-0.7	19932		19002	18944
200-200-15-0.3	45385		41861	27385
200-200-15-0.5	22443		21547	18492
200-200-15-0.7	22377		20526	18502
300-300-5-0.3	72059		68219	68219
300-300-5-0.5	56694		56219	55906
300-300-5-0.7	56938		56463	56463
300-300-10-0.3	73806		72008	72008
300-300-10-0.5	56694		55764	54929
300-300-10-0.7	56938		55008	55008
300-300-15-0.3	75321		72861	71668
300-300-15-0.5	58026		55218	55218
300-300-15-0.7	56713		55508	55508
Av1		2633.06	2632.89	**2630.50**
Av2	18664.55	2710.35	17929.02	**17432.40**
↑	0	0	0	**28**
=	0	0	*26*	*26*
↓	54	18	28	0

Table 10.4 reports the solution values achieved by the Cplex solver, ANS, NSH and the proposed algorithm IBRP-DP$_{\mathrm{LB}}$ on all instances of Set II and Set III. Column 1 of the table shows the instance's label, column 2 reports the Cplex solver's integer bound (noted Best$_{\mathrm{Cplex}}$), column 3 tallies the best upper bound reached by ANS (noted Best$_{\mathrm{ANS}}$) extracted from Hifi et al., 2015, column 4 displays the best bound achieved by Moussa, 2015's algorithm (noted Best$_{\mathrm{NSH}}$) and column 5 shows the best bounds achieved by the proposed method IBRP-DP$_{\mathrm{LB}}$) (noted Best$_{New}$).

The second block of the table is composed of two lines (Av1. denoting the average value over the instances tested with $|S|$ varies from 50 to 100, and Av2. denoting the global average value over all tested instances). Finally, the block of the table summarizes the average solution values achieved by each method (the symbol "$\uparrow$" means that the best achieved value improves the best solution value available in the literature, the symbol "$\downarrow$" indicates the superiority of the other methods while the symbol "$=$" is used when at least one method matches the same lower bound).

In what follows, we comment on the results reported in Table 10.4, where we compare the number of results obtained by the proposed method (the solution values) with the best solution values provided by the other methods.

1. Cplex versus other methods: both ANS and RSBA dominate the Cplex solver despite several tunings applied to the solver settings.

2. BRP-DP$_{\mathrm{LB}}$ versus ANS on set II: ten new solution values (bounds, in bold-space) are provided by BRP-DP$_{\mathrm{LB}}$, and it matches all the rest of solution values achieved by ANS. The average results achieved by BRP-DP$_{\mathrm{LB}}$ is smallest than that achieved by ANS. Indeed, BRP-DP$_{\mathrm{LB}}$ reaches an average best solution value of 2630.50 (column 5, line Av1.) while ANS reaches a greatest average best solution value of 2632.89 (column 3, line Av1.).

3. BRP-DP$_{\mathrm{LB}}$ versus NSH:

   a) On the one hand, the average results achieved by BRP-DP$_{\mathrm{LB}}$ is smallest than that achieved by NSH.

   b) On the other hand, one can observe that BRP-DP$_{\mathrm{LB}}$ outperforms NSH over instances of both sets. Indeed, the proposed algorithm is able to achieve better solution values than those reached by NSH; BRP-DP$_{\mathrm{LB}}$ provides 28 (new) lowest solution values than those achieved by NSH and matches 26 (the rest of) solution values.

10.4 Conclusion

In this paper, the k-clustering minimum completion problem was solved by using a new version of the rounding strategy-based algorithm. First, a starting solution was built by using a basic rounding procedure that selects step by step a fractional item from the mixed-integer program. Second, an improvement procedure was designed, where several local operators were considered. Third, the dropping and re-optimization strategy were combined with the local branching strategy and, embedded into an iterative search for implementing an efficient and powerful method. The performance of the proposed method was experimentally analyzed on a set of problem instances available in the literature, containing small, medium, and large-sized instances. The experimental part showed that the proposed method remains competitive and was able to provide new bounds when compared to the best bounds available in the literature.

Chapter 11

A rounding strategy-based algorithm for the k-Clustering minimum biclique completion problem

Brief table of contents

The k-clustering minimum completion is an NP-hard combinatorial optimization problem. In this chapter, we propose to approximately solve the problem by using a

rounding strategy-based algorithm. Such an algorithm can be viewed as a hybrid method where both rounding strategy and augmented local search cooperate for highlighting the quality of the solutions achieved. The performance of the proposed method is evaluated on a set of benchmark instances taken from the literature, where its provided results (bounds) are compared to those achieved by the best method available in the literature. New bounds have been obtained.

11.1 Introduction

The rounding method is applied in several contexts with binary or mixed-integer problems (MIP) (from the recent studies we can address Tanksale and Jha, 2020 and, Reihaneh and Ghoniem, 2018).

A similar method to the rounding strategy is seen in Wang, 2020 for quadratic integer programming problems with linear constraints. The proposed algorithm reduces the domain by integration into a branch-and-bound algorithm.

In what follows, we propose an alternative solution approach that mimics the first branch-and-bound level of the exact method.

A *Basic Rounding Procedure* (BRP) is applied to the mixed linear relaxation of the original problem, namely MIP_{KCMBC}, described as follows:

$$
\begin{aligned}
\text{MIP}_{KCMBC} : \overline{z}(x) = \min \quad & \sum_{k \in K} \sum_{(i,j) \in \bar{E}} z_{ijk} \\
\text{s.t.} \quad & x_{ik} + y_{jk} \leq 1 + z_{ijk}, \quad \forall (i,j) \in \bar{E}, \forall k \in K && (11.1) \\
& \sum_{k \in K} x_{ik} = 1, \quad \forall i \in S && (11.2) \\
& x_{ik} \leq y_{jk}, \quad \forall (i,j) \in E, \forall k \in K && (11.3) \\
& 0 \leq x_{ik} \leq 1, \; 0 \leq y_{jk} \leq 1, \quad \forall i \in S, \forall j \in C, \forall k \in K, \\
& z_{ijk} \in \{0,1\}, \quad \forall (i,j) \in \bar{E}, \forall k \in K,
\end{aligned}
$$

Where all the variables related to both clients and services are relaxed. One can observe that calling BRP induces an approximate solution for P'_{KCMBC}, if all the variables $\{x_{ik}$ and $y_{jk}\}$ were binary (note that x_{ik} and y_{jk} are not binary in BRP). However, the approximate solution is reached by rounding the variables having a fractional value in the optimal solution (achieved from MIP_{KCMBC}) and by respecting the rest of the constraints, i.e., inequalities from (11.1) to (11.3). We recall that the relaxation is related only to both vectors x and y while the variables of z are maintained to their original domains.

The used BRP can be described as follows:

1. Set $\mathrm{P}_{reduce} = \mathrm{MIP}_{KCMBC}$.
2. Solve P_{reduce} using the Cplex solver, and let $(\underline{x}, \underline{y}, \underline{z})$ be the achieved solution.
3. Fix the value of each $\underline{x}_i$, if $\underline{x}_i = 1$ in $\underline{x}$.
4. Select among the non-integer variables, the variable $\underline{x}_i$ having the largest fractional value, and round it to one.
5. Make sure that the fixed values, respects the constraints from (11.1) to (11.3) if not go to step 4 and select the next largest fractional value
6. Remove the fixed variables from the current problem and let P_{reduce} be the new reduced problem.
7. Repeat steps from (2) to (6) until fixing all the elements of the original problem.
8. Retrieve all the items satisfying constraints from (11.1) to (11.3), and fix them to 1, 0 otherwise.

Finally, BRP provides a feasible integer solution for P'_{KCMBC}. Because of the greedy nature of BRP and of the complexity of P'_{KCMBC}'s precedence constraints (Inequalities of type (11.3)), BRP may provides moderate-quality solutions. In fact, fixing a decision variable y_{jk} (service j belonging to the partitions k) doesn't induce a direct fixation of its containing items (clients) while fixing x_{ik} to one (client i belonging to the partitions k), imposes the fixation of its services.

Thus, BRP may eliminate some items by fixing their corresponding decision variables to one. Hence, BRP is hardly used as a stand-alone procedure, but rather as part of an elaborate computational approach.

Akeb et al., 2011 proposed a "generic" solution method based on the rounding strategy combined with a restricted exact solution. They mentioned that an exact solver such as Cplex is not a suitable way to solve large-sized instances. As an observation, fixing some variables and completing the reduced problem using local search may improve the local search although the improvement highly depends on the local search procedures and imposed runtime limit. Moreover, the process needs an accelerating method. Herein we applied both drop and fix strategies to scatter the solutions. This is applied to discover new (sub)search spaces.

11.2 Enhancing the current solution

To refine the quality of a solution, four neighborhoods are considered. The first two operators introduce moves with a slight degree of freedom while the last two operators try to extend the search neighborhood to several clusters. Move operators are described as follows:

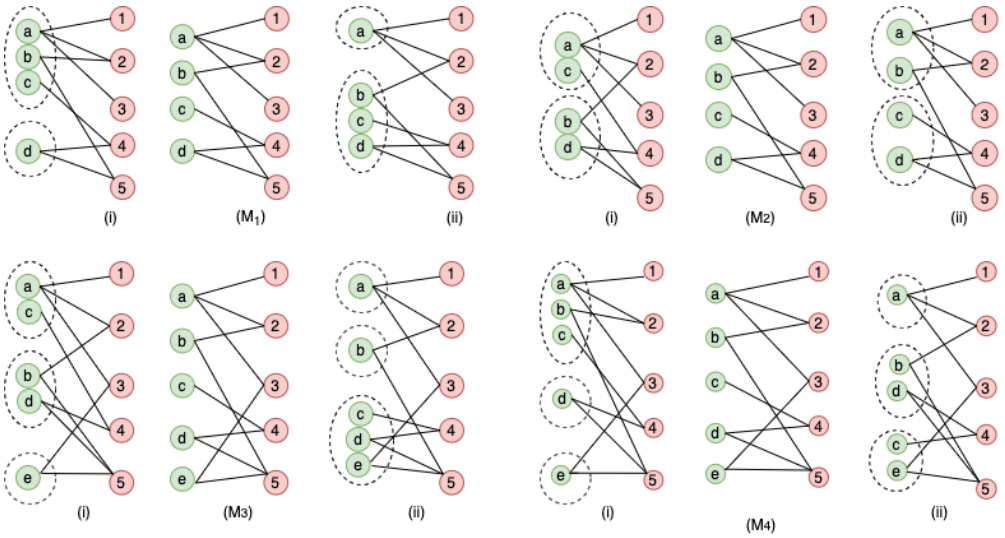


Figure 11.1: Illustration of the four local operators (M_1, M_2, M_3 and M_4) on two instances of P_{KCMBC}

1. Operator M_1: Let C_i and C_k, $i \neq k$, be two clusters, where $|C_i| \geq 2$ and, $y_i^{(1)}, y_i^{(2)} \in C_i$. Move both $y_i^{(1)}$ and $y_i^{(2)}$ from C_i to C_k and, evaluate the new achieved solution.

2. Operator M_2: Let C_i and C_k, $i \neq k$, be two clusters, where $|C_i| \geq 1,\ C_k| \geq 1$ and, $y_i^{(1)} \in C_i$ and $y_k^{(1)} \in C_k$. Make an exchange between both $y_i^{(1)}$ and $y_k^{(1)}$, i.e., move $y_i^{(1)}$ from C_i to C_k and, move $y_k^{(1)}$ from C_k to C_i and, evaluate the new achieved solution.

3. Operator M_3: Let $C_\ell,\ \ell \in \{1, \ldots, n_K\}$ and (i, j, k) the indices characterizing three different clusters, such that at least two clusters contain at least two services, for instance C_i and C_j. Suppose that $y_i^{(1)} \in C_i$ and $y_j^{(1)} \in C_j$; move both $y_i^{(1)}$ (from C_i) and $y_j^{(1)}$ (from C_j) to C_k and, evaluate the new achieved solution.

4. Operator M_4: Let $C_\ell,\ \ell \in \{1, \ldots, n_K\}$ and (i, j, k) the indices characterizing three different clusters, such that at least one cluster contains at least three services, for instance C_i. Suppose that $y_i^{(1)} \in C_i$ and $y_i^{(2)} \in C_i$; either (i) move $y_i^{(1)}$ from C_i to C_j and $y_i^{(2)}$ from C_i to C_k or, (ii) move $y_i^{(2)}$ from C_i to C_j and $y_i^{(1)}$ from C_i to C_k. Evaluate the new provided solution.

Figure 11.1 illustrates the transformations induced by the four local operators on two instances of P_{KCMBC}.

1. Using both M_1 and M_2 operators on the first instance − (top-side of Figure 11.1):

   - First (top-left side), the instance is composed of two sets {a, b, c, d } and {1, 2, 3, 4, 5} denoting services and clients, respectively. The current solution

(top-left side (i)) is composed of 2 clusters C_1 = {a, b, c} and C_2 = {d}: the other members related to each cluster are the customers that are connected to the services with their corresponding edges. Note that we neglect representing the customers in the clusters to have better visualizations in the graphs. The cost of this solution is the summation of the costs for each cluster; then, C_1 is evaluated to 9 and C_2 to 0. Thus the objective value of the current solution is equal to 9. Using the first operator M_1 induces a new solution (top-left side (ii)) with an objective value equal to 4.

- Second (top-right side), the current solution (top-right side (i)) is composed of two clusters $C_1\{a, c\}$ and $C_2 = \{b, d\}$, where its objective value is equal to 6. By applying the local operator M_2 on the aforementioned solution, it induces the solution displayed on the top-right side (ii) of the figure; that is, a solution with an objective value equal to 4 (since for the first cluster C_1 its cost is equal to 3 whiles for C_2 it is equal to 1).

2. Using both M_3 and M_4 operators on the second instance − (bottom-side of Figure 11.1):

   - The bottom-left side: both sets of the second instance (services and clients) are composed of five nodes: {a, b, c, d, e } and {1, 2, 3, 4, 5}, respectively. The current solution (bottom-left side (i)) is composed of three clusters $C_1 = \{a, c\}$, $C_2 = \{b, d\}$ and $C_3 = \{e\}$, with an objective value equal to 6 (sine the the cost of C_1 is equal to 4, that of C_2 is equal to 2 and C_3's cost is equal to 0). By applying the local operator M_3 to the current solution, a new solution is provided, as shown in Figure 11.1 (bottom-left side (ii)); that is a solution achieving an objective value of 4.
   - The bottom-right side (i): the current solution is now composed of three clusters $C_1 = \{a, b, c\}$, $C_2 = \{d\}$ and $C_3 = \{e\}$, with an objective value equal to 9. Using the last operator M_4 built a new solution as displayed in Figure 11.1 (bottom-right side (ii)); that is a solution achieving an objective value of 5.

Applying this strategy may penalize the complexity and the average runtime. Hence, we propose a *hill-climbing strategy* instead of considering the entire neighborhood. This procedure is derived from a descent method, where both clusters and services are randomly selected. The used Hill-Climbing procedure is described as follows:

1. Define the current solution Λ as the initial state.
   Let S_N (initially $z(S_N) = +\infty$) be the best solution related to the global neighborhood (containing all operators).

2. Repeat

a) Apply one operator, among the four operators from M_1 to M_4, to the current solution Λ and let solution Λ' be the new solution.
Set $S_N := S'$ (if $z(S_N) > z(S')$).

b) Update S with S' if $z(S) > z(S')$; in this case, restarting the search process with the new solution.

3. Until the stopping criteria is performed.

A (complementary) constructive procedure

A complementary Constructive Procedure (CP) works as follows (when applied to the original problem): let $C = \{c_1, \ldots, c_n\}$ (resp. $S = \{S_1, \ldots, S_m\}$) be the set of $n = |C|$ clients (resp. $m = |S|$ services), E be the set of edges (existing links) between both sets C and S from $c \in C$ to $s \in S$ (each existing link is noted (c, s)) and $nb_K = |K|$ be the number of clusters.

1. Let $K = \{K_1, \ldots, K_{n_K}\}$ be the starting n_K dummy clusters (empty clusters). Set

   a) $S' = S$ and $C' = C$;

   b) $h = 1$ (the index h associated to the first cluster).

2. Set $\delta(s_i),\ s_i \in S'$, in decreasing order, where $\delta(s_i)$ denotes the degree of the s_i-th vertex (characterizing the number of clients-nodes linked to it); Set $\ell = \arg\max_{s_i \in S'} \Big\{\delta(s_\ell) = \delta(s_1) \mid \delta(s_\ell) \geq \delta(s_i)\Big\}$;

3. Set $K_h = K_h \cup \{s_\ell\}$, $S' = S' \setminus \{s_\ell\}$, and update the set C' whenever s_ℓ is affected, i.e., new sets S' and C' are recomputed.

4. For the $n_K - 1$ remaining clusters, from the cluster $h = n_K - 1$ down to $h = 2$ do:

   a) Assign a vertex $\ell' \in S'$ to the h-th cluster;

   b) Update both sets S' and C'.

5. Set $h = 1$;

6. Repeat steps (2) and (4) till fixing all services in the clusters.

In other words, CP groups the services to create an initial solution. Initially, CP creates a sorted list of the services in descending order with their corresponding number of connections (here we called "degree of the stated service" that characterize the number of clients linked to it). It then selects a service that has the most connection with the clients in step 2 (i.e., head of the sorted list).

Then we remove the selected service from the sorted list and we update the list of the services and clients. Here we assume that the selected cluster is assigned to the first dummy cluster.

In step 4, we assign services from the sorted list to the remaining clusters and ignore their connections. The process will be repeated to assign all the services to the clusters.

Scatter operators

In addition to the enhancing procedure (Section 11.2), several neighborhood procedures-based strategies are applied. As used in Hifi and Michrafy, 2013 (for more sophisticated strategies, see also Glover, Laguna, and Marti, 2003 and, Anaya-Arenas, Prodhon, Renaud, and Ruiz, 2019), two complementaries strategies can be combined: (i) degrading and (ii) re-optimization strategies. The mentioned strategies are applied to escape from local optima and to diversify the search process. Indeed, the degrading strategy fixes a partial solution, according to the solution at hand, by dropping some elements while the re-optimization strategy selects new elements and adds them into the perturbed solution for reaching a new scattered solution.

The *soft-scatter* and *agressive-scatter* strategies may be summarized as follows:

- *Soft-scatter:* for a solution Λ, it selects a cluster with the highest cost, namely C_k. Next, it will remove $\beta\%, \beta \geq 0$, of the nodes from the aforementioned cluster and will dispatch them into the other clusters. The aim of such an operator is to defeat the most expensive cluster. Since only one cluster is affected by the degradation, then neighbors remain with small diversification.
- *Aggressive-scatter:* for solution Λ, it tries to reduce the problem by randomly fixing a subset of variables of Λ as follows:

  Step 1: From Λ, fix $\beta\%$ of the nodes belonging to each cluster.

  Step 2: Build the reduced problem, namely P''_{KCMBC} and solve it using a complementary CP.

The smaller the value of β, the larger the computational time. In this case, the search process can perturb the solutions, which may induce a slow convergence of the method.

We note that there are two special cases for β: (i) $\beta = 0\%$ for which the reduced problem is empty and so, any exact method will be forced to solve a problem equivalent to the original one (hyperbole) and, (ii) $\beta = 100\%$ means all of the number of nodes belonging to the cluster and so, the original solution remains unchanged (belittlement). Limited computational results showed that a better value for β should be in the interval $[50,\ 90]$ which is also related to the size of the instance problem.

Algorithm 8 The Rounding Strategy-Based Algorithm (RSBA)

Input. An instance of P'_{KCMBC}.

Output. A feasible solution $\Lambda^{\star}_{KCMBC}$.

1: **Initialization Step.**
   Set $\Lambda^{\star}_{KCMBC} = \emptyset$
2: **repeat**
3: Apply BRP combined with CP to P'$_{KCMBC}$ and let Λ'_{KCMBC} be the solution reached.
4: **repeat**
5: Set $iter_{local} = 1$
6: **repeat**
7: Apply the intensification phase (Section 11.2) using Hill-Climbing to Λ'_{KCMBC}
8: Let Λ_{local} be the new solution
9: **if** $(z(\Lambda'_{KCMBC}) > z(\Lambda_{local}))$ **then**
10: Replace Λ'_{KCMBC} with Λ_{local}
11: Set $iter = 1$
12: **else**
13: $iter = iter + 1$
14: **end if**
15: Update $\Lambda^{\star}_{KCMBC}$ with Λ'_{KCMBC}, if necessary
16: **until** $\Big($stopping local condition is performed$\Big)$
17: Apply the scatter search procedure (Section 11.2) to $\Lambda^{\star}_{KCMBC}$
18: Let Λ_{local} be the local solution reached.
19: **until** $\Big($stopping global condition is performed$\Big)$
20: **until** $\Big($the time limit is performed$\Big)$
21: **return** $\Lambda^{\star}_{KCMBC}$.

11.3 An overview of the proposed method

Algorithm 8 illustrates the main steps of the Rounding Strategy-Based Algorithm (RSBA). The input of RSBA is an instance of P'$_{KCMBC}$ and its output is a near-optimal solution $\Lambda^{\star}_{KCMBC}$.

The algorithm begins by generating a staring solution (line 1) initialized to an empty set. RSBA is composed of two internal loops: (i) the first internal loop `repeat` from line 6 to line 16 that is applied for generating a current starting solution, which is enhanced using the intensification phase. Where stopping local condition defined with a limited number of iterations based on the size of the instances. While the second loop `repeat` from line 4 to line 19 serves to diversify the search whenever the first internal loop stagnates on a local optimum. The stopping global condition is reached if the global loop can not generate solutions with better qualities. Both loops are embedded into a global loop `repeat` (from line 2 to line 20) serve to repeat the enhancement and the scattering on a new solution generated by the randomized BRP combined with the constructive procedure CP. The global loop is iterated till the runtime limit is performed. Finally (line 21), RSBA returns $\Lambda^{\star}_{KCMBC}$, the best solution found so far.

11.4 Computational results

To evaluate the effectiveness of RSBA on benchmark instances, two sets of instances are considered: (i) the first set contains 9 small-sized instances, extracted from Gualandi et al., 2013, where their optimal values are known and, (ii) the second set includes 18 medium and large-scale instances, extracted from Hifi et al., 2015.

Note that for the first set, the number of the services (resp. clients) related to S (resp. C) varies in the discrete interval {15,18,20}, $|S| = |C|$, the number of the clusters n_K varies in the discrete interval {3,4,5} and, the density of the bipartite graph d was generated using the following formula: $d = \frac{|E|}{|S|.|C|}$; that varies in the interval {0.3, 0.5, 0.7}. The instances of the second set were generated following Gualandi *et al.*'s generator, where the number of services (resp. clients) of S (resp. C) varies in the discrete interval {50,80,100} while all other parameters are the same. The proposed method was coded in C++ and run on the Intel Pentium Core i7 with 2.1 GHz.

Behavior of the basic rounding procedure

Table 11.1 reports the results achieved by BRP on all tested instances: column 1 shows the instance information, the best upper bound (column 2) extracted from the literature, the bound achieved by the starting procedure (displayed in column 3) and its relative percentage experimental approximation ration $\mathrm{A_{BRP}} = (\frac{\mathrm{U_{BRP}}}{Best} - 1) \times 100$, reported in column 4 ($0 \leq \mathrm{A_{BRP}} \leq 1$ and the smallest value of $\mathrm{A_{BRP}}$, better the provided solution). As mentioned above, BRP solves a series of MIP and so, repeating that resolution may easily increase its runtime; herein, for the tested instances, the runtime varies from one second to five minutes. Of course, one can observe that accelerating the rounding procedure can be realized if at each step of BRP a subset of fractional variables is rounded up, but the degradation of the solution's quality can be expected. In what follows, we comment on the results of Table 11.1:

- BRP matches all best solutions when compared to those achieved by the best methods of the literature, where $|S|$ varies between 15 and 20.
- Despite the simplicity of BRP, the experimental approximation ratio varies from 0.698 (Int. 100-100-10-0.3) and 12.687 (Int. 50-50-10-0.3). In some cases, the approximation value is better for large instances.
- Finally, BRP's average global bound (2734.39: column 2, last line) remains interesting; in this case, BRP achieves an average experimental ratio of 4.292.

Table 11.1: Solution quality of BRPs' starting upper bounds on all instances

#Inst.	Best	U_{BRP}	A_{BRP}
15-15-5-0.3	51	51	0
15-15-5-0.5	50	50	0
15-15-5-0.7	39	39	0
18-18-5-0.3	92	92	0
18-18-5-0.5	86	86	0
18-18-5-0.7	63	63	0
20-20-5-0.3	119	119	0
20-20-5-0.5	123	123	0
20-20-5-0.7	82	82	0
Av.	78.33	78.33	0.00
50-50-5-0.3	1319	1418	7.506
50-50-5-0.5	1072	1090	1.679
50-50-5-0.7	671	683	1.788
50-50-10-0.3	938	1057	12.687
50-50-10-0.5	875	937	7.086
50-50-10-0.7	575	596	3.652
80-80-5-0.3	3815	4075	6.815
80-80-5-0.5	2862	2916	1.887
80-80-5-0.7	1768	1792	1.357
80-80-10-0.3	3202	3566	11.368
80-80-10-0.5	2571	2751	7.001
80-80-10-0.7	1618	1656	2.349
100-100-5-0.3	6248	6547	4.786
100-100-5-0.5	4650	4737	1.871
100-100-5-0.7	2842	2863	0.739
100-100-10-0.3	4298	4328	0.698
100-100-10-0.5	5427	5487	1.106
100-100-10-0.7	2644	2720	2.874
Av.	2633.06	2734.39	4.292

Effect of the first phase

Herein, we evaluate the effect of the Enhanced BRP (EBRP) on the same instances considered in Section 11.4.

Table 11.2 displays EBRPs' results: column 1 shows the instance information, the best upper bound displayed in column 2 (also used in Gualandi et al., 2013; Hifi et al., 2015; Al-Iedani et al., 2016), the objective value reached by EBRP is displayed in column 3 while the last column 4 tallies its Gap; that is the difference between U_{RBR} and U_{EBRP}, i.e., Gap = U_{RBR} - U_{EBRP}. From Table 11.2, we can observe what follows:

1. The first set of instances (top-side) shows that EBRP matches all the best solution values available in the literature.

2. The second set of instances shows that EBRP is able to improve all bounds achieved by BRP. It is able to provide two new bounds (in bold-space) when compared to those achieved by the best method of the literature.

3. The decrease of EBRP's global average objective value is significant. Indeed, it is able to reach solutions with an average bound equals to 1781.52 (Column 4, the

Table 11.2: Solution quality of EBRPs' starting upper bounds on all instances

			The first phase (EBRP)	
#Inst	Best	U_{RBR}	U_{EBRP}	Gap
15-15-5-0.3	51	51	51	0
15-15-5-0.5	50	50	50	0
15-15-5-0.7	39	39	39	0
18-18-5-0.3	92	92	92	0
18-18-5-0.5	86	86	86	0
18-18-5-0.7	63	63	63	0
20-20-5-0.3	119	119	119	0
20-20-5-0.5	123	123	123	0
20-20-5-0.7	82	82	82	0
50-50-5-0.3	1319	1418	1319	**99**
50-50-5-0.5	1072	1090	**1069**	**21**
50-50-5-0.7	671	683	672	**11**
50-50-10-0.3	938	1057	943	**114**
50-50-10-0.5	875	937	875	**62**
50-50-10-0.7	575	596	585	**11**
80-80-5-0.3	3815	4075	3815	**260**
80-80-5-0.5	2862	2916	2862	**54**
80-80-5-0.7	1768	1792	1769	**23**
80-80-10-0.3	3202	3566	**3195**	**371**
80-80-10-0.5	2571	2751	2571	**180**
80-80-10-0.7	1618	1656	1618	**38**
100-100-5-0.3	6248	6547	6248	**299**
100-100-5-0.5	4650	4737	4650	**87**
100-100-5-0.7	2842	2863	**2836**	**27**
100-100-10-0.3	4298	4328	4298	**30**
100-100-10-0.5	5427	5487	5427	**60**
100-100-10-0.7	2644	2720	2644	**76**
Av.	**1781.48**	1849.04	*1781.52*	**67.52**

last line Av.) whereas BRP's average global value is equal to 1849.04: column 3, line Av.).

4. EBRP is capable of providing less average value (1781.48, column 2, line Av.) compared to RBR (0.04 GAP for EBRP and 67.56 GAP for RBR). EBRP is also able to achieve three new bounds (in bold) compared to the best values reported in the literature.

5. Finally, EBRP is able to enhance BRPs' bounds since it achieves a Gap that varies from 11 (Ints. 50-50-5-0.7 and 50-50-10-0.7) to 29 (instance 100-100-5-0.3).

Because of the random aspect of EBRP, ten trials (noted from Tr1 to Tr10) are considered for each tested instance in Table 11.3. Note that EBRP needs to fix the stopping criteria and the restarting procedure whenever one of the local operators achieves a new solution. After several tunings, stopping criteria were fixed to the number of edges that contain the instance solved while the local parameter related to a plateau, was fixed to 1000; that means a higher chance was given to each service for appearing into a series of solutions.

Table 11.3: Solution quality of EBRPs' starting upper bounds on the second set of instances

	Using the first phase (EBRP): ten trials											
#Inst.	Tr1	Tr2	Tr3	Tr4	Tr5	Tr6	Tr7	Tr8	Tr9	Tr10	$\text{Best}_{\text{EBRP}}$	Av_{EBRP}
15-15-5-0.3	51	51	51	51	51	51	51	51	51	51	51	51.00
15-15-5-0.5	50	50	51	50	51	50	50	50	50	50	50	50.20
15-15-5-0.7	39	39	39	39	39	39	39	39	39	39	39	39.00
18-18-5-0.3	92	92	92	92	92	92	92	92	92	92	92	92.00
18-18-5-0.5	86	86	86	86	86	86	86	86	86	86	86	86.00
18-18-5-0.7	63	63	63	63	63	63	63	63	63	63	63	63.00
20-20-5-0.3	119	119	119	119	119	119	119	119	119	119	119	119.00
20-20-5-0.5	123	123	123	123	123	123	123	123	123	123	123	123.00
20-20-5-0.7	82	82	82	82	82	82	82	82	82	82	82	82.00
Average	78.33	78.33	78.44	78.33	78.44	78.33	78.33	78.33	78.33	78.33	78.33	
50-50-5-0.3	1319	1321	1319	1319	1319	1321	1319	1321	1319	1319	1319	1319.55
50-50-5-0.5	1072	1069	1087	1075	1069	1075	1069	1069	1069	1072	1069	1072.55
50-50-5-0.7	672	672	683	683	673	672	672	672	672	681	672	674.82
50-50-10-0.3	943	950	943	947	943	947	943	943	950	947	943	944.91
50-50-10-0.5	888	875	883	875	875	875	883	875	883	875	875	878.36
50-50-10-0.7	585	586	586	585	586	587	586	587	585	585	585	584.82
80-80-5-0.3	3841	3815	3918	3899	3815	3815	3918	3815	3815	3815	3815	3843.73
80-80-5-0.5	2916	2862	2862	2916	2862	2862	2862	2862	2862	2862	2862	2871.82
80-80-5-0.7	1769	1772	1769	1775	1775	1775	1775	1769	1775	1769	1769	1771.91
80-80-10-0.3	3224	3195	3195	3211	3202	3195	3195	3195	3202	3211	3195	3202.45
80-80-10-0.5	2653	2571	2571	2590	2571	2571	2596	2571	2575	2590	2571	2584.55
80-80-10-0.7	1652	1618	1618	1618	1652	1618	1618	1652	1618	1618	1618	1627.27
100-100-5-0.3	6427	6281	6248	6334	6315	6281	6268	6248	6248	6334	6248	6293.82
100-100-5-0.5	4650	4684	4650	4650	4672	4684	4671	4671	4659	4650	4650	4662.82
100-100-5-0.7	2836	2842	2836	2836	2842	2842	2842	2836	2836	2836	2836	2838.73
100-100-10-0.3	4318	4318	4298	4298	4298	4298	4328	4298	4318	4298	4298	4306.18
100-100-10-0.5	5427	5428	5428	5427	5427	5427	5427	5427	5427	5428	5427	5427.27
100-100-10-0.7	2720	2657	2644	2644	2720	2657	2644	2657	2650	2720	2644	2668.82
Average	2661.78	2639.78	2641.00	2649.00	2645.33	2639.00	2645.33	2637.11	2636.83	2645.00	2633.11	
Av.Global	1800.63	1785.96	1786.81	1792.11	1789.70	1785.44	1789.67	1784.19	1784.00	1789.44	1781.52	1788,13

Effect of the dropping parameter β

RSBA needs to tune (i) the stopping criterion used in the global loop, and (ii) the number of items to drop. First, we considered the stopping criterium as a maximum runtime limit fixed to thirty minutes (this value was retuned after several tunings: we also augmented that runtime limit without success). Second, we fixed β to 50%, 60%, 70%, 80% and 90%.

Table 11.4 reports the bounds achieved by RSBA when varying β. Column 1 refers to the instance's information (because no new solutions have been provided for the first set, we only tested the instances of the second set), column 2 represents the best objective value and columns from 3 to 7, under the value assigned to β, display the bounds reached by RSBA. The last two lines of the table (Average and NB_Best) tally the global average value and the number of times that the corresponding method provides a better bound, overall tested instances. From Table 11.4, we observe what follows:

Table 11.4: Illustration of the best bounds achieved by RSBA on large-scale problem instances when varying the parameter β.

	RSBA: variation of β					
#Inst	50%	60%	70%	80%	90%	Best(Min)
50-50-5-0.3	1323	1319	1315	1332	1342	1315
50-50-5-0.5	1075	1088	1068	1072	1090	1068
50-50-5-0.7	672	671	671	671	683	671
50-50-10-0.3	938	950	938	957	957	938
50-50-10-0.5	886	871	875	886	875	871
50-50-10-0.7	575	585	583	575	577	575
80-80-5-0.3	3805	3815	3805	3815	3903	3805
80-80-5-0.5	2864	2864	2862	2862	2864	2862
80-80-5-0.7	1771	1768	1768	1770	1768	1768
80-80-10-0.3	3270	3219	3219	3188	3236	3188
80-80-10-0.5	2751	2570	2620	2659	2596	2570
80-80-10-0.7	1617	1617	1617	1651	1656	1617
100-100-5-0.3	6247	6315	6247	6427	6427	6247
100-100-5-0.5	4672	4650	4650	4659	4684	4650
100-100-5-0.7	2842	2836	2832	2832	2863	2832
100-100-10-0.3	4318	4318	4298	4298	4318	4298
100-100-10-0.5	5487	5426	5427	5426	5427	5426
100-100-10-0.7	2657	2644	2644	2650	2720	2644
Min	575	585	583	575	577	
Av	2653.89	*2640.33*	**2635.50**	2651.67	2665.89	
Max	6247	6315	6247	6427	6427	
St Dev	1715.17	1714.62	1706.38	1726.07	1732.02	
Nb_Best	5	*8*	**13**	7	1	

- The best global average value is achieved for $\beta = 70$ (column 4, both lines Average and Nb_Best).
- The value $\beta = 60$ seems to be the value that provides closest bounds to those reached by RSBA with $\beta = 70$. Indeed, RSBA provides a global average bound

equals to 2640.33 (the second block of Table 11.4, line Average, column 3). In term of the gap, RSBA's global average bound achieves an average gap of 4.83.

- For small values of β as well as for large values of β, one can observe that RSBA's global average bound degrades when compared to that achieved with $\beta = 70$ (even for $\beta = 60$).
- RSBA with $\beta = 70$ provides 13 better bounds (column 4, line NB_Best) while the maximum number of better bounds achieved by the best value of β, over the rest of the variations, is equal to 8.

For a comparative study, we set the hypothesis H_0: $RSBA\alpha_1 - RSBA\alpha_2 = V$, where $\alpha_1 \leq \alpha_2$, to express that algorithm $RSBA\alpha_1$ performs better than $RSBA\alpha_2$ otherwise, H_a: algorithm $RSBA\alpha_2$ is better than $RSBA\alpha_1$.

Herein, we consider that the smallest the average bound and the greater the number of better bounds, the better the corresponding version of RSBA. Table 11.5 reports, by varying α_1 and α_2 ($\alpha_1 \neq \alpha_2$), the statical study on all instances by using both the *sign test* and the *sign rank test* (the detailed results containing the average bounds for all versions of RSBA are reported in Table 11.7). Columns from 1 to 4 display the statistical results of RSBA when varying α: line 1 (resp. line 4) tallies the p-value corresponding to the sign test (resp. rank sign test) and line 2 reports the number of times that the first algorithm dominates the second one (resp. line 3). Moreover, in order to reduce the comparative study between all versions, we started by testing the first two versions, then only the better version is retained to compare it to the next version, and so on.

Table 11.5: p-values for sign test and Wilcoxon rank test on all tested instances with the significance level $\alpha = 0.05$ (D_1=RSBA$_{50\%}$ vs RSBA$_{60\%}$, D_2=RSBA$_{60\%}$ vs RSBA$_{70\%}$, D_3=RSBA$_{70\%}$ vs RSBA$_{80\%}$ and, D_4=RSBA$_{70\%}$ vs RSBA$_{90\%}$

	D1	D2	D3	D4
p-value (sign test)	0.941	0.981	0.029	0.004
N^+	10	9	3	2
N=	3	6	4	5
N^-	5	3	11	11
p-value (Wilcoxon test)	0.866	0.958	0.013	0.002

From Table 11.5 one can observe what follows:

- For D_1 the p-value related to the sign test (resp. Wilcoxon test) is greatest to the significance level $\alpha = 0.05$, indicating that RSBA$_{60\%}$ performs better than RSBA$_{50\%}$ (rejecting the hypothesis H_0). In this case, RSBA$_{60\%}$ is the candidate version to compare to RSBA$_{70\%}$. D_2's p-value related to the sign test (resp. Wilcoxon test) is also greatest to the significance level $\alpha = 0.05$; it means that RSBA$_{70\%}$ performs better than RSBA$_{60\%}$ and RSBA$_{70\%}$ is returned for the nest comparison.

- For D_3, the p-value related to the sign test (resp. Wilcoxon test) is smallest than the significance level $\alpha = 0.05$, indicating that $\text{RSBA}_{70\%}$ has a better behavior. Also, increasing the value of α doesn't improve the quality of EFCS's average bounds: both p-values related to D_4 are equal to 0.004 and 0.002, which means that $\text{EFCS}_{70\%}$ evolves better than the other versions.

Therefore, because we seek solutions with high quality and with more best bounds, we then fix $\beta = 70$ for the rest of the experimental study.

Table 11.6: Performance of RSBA versus two other algorithms of the literature on both sets of instances. On the one hand, the value in the "**bold-space**" means that the corresponding method provides a better (average) bound. On the other hand, the number of better provided (matched) bounds are reported on the last part of the table.

#Inst.	$\text{Best}_{\text{Cplex}}$	Best_{ANS}	$\text{Best}_{\text{NSBH}}$	$\text{Best}_{\text{RSBA}}$
15-15-5-0.3	*51*	*51*	52	*51*
15-15-5-0.5	*50*	*50*	51	*50*
15-15-5-0.7	*39*	*39*	41	*39*
18-18-5-0.3	*92*	*92*	94	*92*
18-18-5-0.5	*86*	*86*	87	*86*
18-18-5-0.7	*63*	*63*	*63*	*63*
20-20-5-0.3	*119*	*119*	120	*119*
20-20-5-0.5	*123*	*123*	124	*123*
20-20-5-0.7	*82*	*82*	90	*82*
50-50-5-0.3	1423	1319	1485	**1315**
50-50-5-0.5	1091	1072	1090	**1068**
50-50-5-0.7	676	*671*	681	*671*
50-50-10-0.3	1135	*938*	1151	*938*
50-50-10-0.5	930	875	928	**871**
50-50-10-0.7	578	*575*	588	*575*
80-80-5-0.3	4166	3815	-	**3805**
80-80-5-0.5	2931	*2862*	-	*2862*
80-80-5-0.7	1775	*1768*	-	*1768*
80-80-10-0.3	3735	3202	-	**3188**
80-80-10-0.5	2675	2571	-	**2570**
80-80-10-0.7	1634	1618	-	**1617**
100-100-5-0.3	6645	6248	-	**6247**
100-100-5-0.5	4716	*4650*	-	*4650*
100-100-5-0.7	2854	2842	-	**2832**
100-100-10-0.3	6119	*4298*	-	*4298*
100-100-10-0.5	9111	5427	-	**5426**
100-100-10-0.7	2704	*2644*	-	*2644*
Min	39	39	41	39
Av1.	435.87	410.33	443.00	**409.53**
Av2.	2059.37	1781.48		**1779.63**
Max	9111	6248	1485	6247
St Dev	2376.92	1846.96	500.75	1845.90
Nb_Best	9	18	1	**27 (10)**

Behavior of RSBA versus other available methods

RSBA is compared to those reported in Hifi et al., 2015: an Adaptive Neighborhood Search (ANS), the Neighborhood Search-Based Heuristic of Al-Iedani et al., 2016 (NSBH) and the Cplex solver, that solver was tested using several tunings (each version was fixed to one hour): (i) automatic search method, (ii) dynamic search, and (iii) traditional

branch-and-cut search (for each of these versions, the RINS heuristic was fixed to 100). Hence, the best objective value achieved by these versions is retuned as the best solution value of Cplex. We note that RSBA's runtime was fixed to five minutes for small and medium instances and to thirty minutes for larger instances.

Table 11.6 reports the solution values achieved by the Cplex solver, ANS, NSBH, and the proposed algorithm RSBA on all instances. Column 1 of the table shows the instance's label, column 2 reports the Cplex solver's integer bound ($\text{Best}_{\text{Cplex}}$), column 3 tallies the best upper bound reached by ANS (Best_{ANS}) extracted from Hifi et al., 2015, column 4 displays the best bound achieved by Al-Iedani et al., 2016's algorithm ($\text{Best}_{\text{NSBH}}$) and column 5 shows the best bounds achieved by the proposed method RSBA (Best_{RSBA}). The second block of the table is composed of two lines (Av1. denoting the mean value over the instances with $|S|$ varies from 15 to 50, and Av2. denoting the global average value over all tested instances).

1. Cplex versus other methods: both ANS and RSBA dominate the Cplex solver despite several tunings applied to the solver settings.
2. RSBA versus NSBH: one can observe that RSBA outperforms NSBH. For the instance tested by NSBH, RSBA is able to achieve better solution values than those reached by NSBH; RSBA provides three (new) lowest solution values than those achieved by NSBH and matches 12 (the rest of) solution values.
3. RSBA versus ANS: ten new solution values (bounds, in bold-space) are provided by RSBA, and it matches all the best solution values achieved by ANS. The average results achieved by RSBA (1779.63 (column 5, line Av2.)) is less than that achieved by ANS(1781.48 (column 3, line Av2.)).
4. In the comparative study, we set the hypothesis H_0: Algo vs RSBA=D, where Algo denotes either Cplex solver, ANS, or NSBH to express that algorithm Algo performs better than RSBA and H_a: RSBA has a better behavior than Algo to reject H_0.

In Table 11.7, columns from 1 to 4 display the statistical results of RSBA versus the three other algorithms: line 1 (resp. line 4) tallies the p-value corresponding to the sign test (resp. Wilcoxon rank test), line 2 reports the number of times that the first algorithm dominates RSBA while line 4 reports the number of times that the first algorithm is dominated by RSBA. Line 3 reports the number of times that the compared algorithms reached the same solution value.

From Table 11.7 we obserce that:

- P-values for D_1, D_2 and D_3 is greater than the significance level $\beta = 0.05$ for both sign and Wilcoxon test. It indicates that RSBA performs better than Cplex solver,

Table 11.7: p-values for sign test and Wilcoxon rank test on all tested instances with the significance level $\alpha = 0.05$ (D_1=Cplex vs RSBA, D_2=ANS vs RSBA and, D_3=NSBH vs RSBA

	D1	D2	D3
p-value (sign test)	1	1	1
N^+	18	10	14
N=	9	17	1
N^-	0	0	0
p-value (Wilcoxon test)	0.866	0.958	0.013

ANS, and NSBH, by rejecting the hypothesis H_0 in all the experiments.

- The numbers of occasions that RSBA dominates the best bounds of the other methods are equal to 18, 10, and 14 (line 2, columns from 2 to 4) respectively. It also matches the same solution values in 9, 17 and 1 occasions, respectively, when compared Cplex, ANS and NSBH (line 3, columns from 2 to 4).

Hence, the statistical analysis confirms the competitiveness of RSBA on the available benchmark instances.

Herein, we programmed the algorithms proposed in Hifi et al., 2015; Hifi and Sadeghsa, 2020a; Al-Iedani et al., 2016. We tested the state-of-the-art algorithms with the same environment and the same machine as in this study. We also recorded the run time for each test. The resulted comparative table that compares our method with the state-of-the-art algorithms is reported in Table11.8.

Columns of the table represent the objective values of the instances tested with RSBA (presented in this study), ARSBA from Hifi and Sadeghsa, 2020a, ANS in Hifi et al., 2015, and NSBH from Al-Iedani et al., 2016. The calulation CPU time(in minute) of each test is reported respectively.

To clarify the comparison in Table11.8, we divided the table into three boxes. First section reports the comparison results for the instances with 80 and 100 services(clients) with $k \in \{5, 10\}$. The second box reports the results of the tests on the larger instances composed of 120, 150, 200, and 300 with $k \in \{5, 10\}$. Due to the huge complexity of the instances with $k = 15$ clusters, we separated the instances in the last box.

The average line represents the superiority of RSBA in terms of quality of the solutions and CPU to the other studies in every tested scenario.

From Table 11.8 and the tested instance, we can conclude that in total the average of the objective values and the CPU time with RSBA were lower than every other tested method.

Table 11.8: Comparison between RSBA and the state-of-art algorithms

#Inst.	RSBA	CPU	ARSBA	CPU	ANS	CPU	NSBH	CPU
80-80-5-0.3	3805	0.31	3873	0.30	3938	0.91	3999	3.01
80-80-5-0.5	2862	0.30	2879	0.31	2930	4.88	2948	2.67
80-80-5-0.7	1769	0.30	1777	0.62	1809	4.05	1839	4.81
80-80-10-0.3	3188	0.31	3237	0.31	3333	0.29	3364	3.20
80-80-10-0.5	2596	0.30	2609	0.61	2661	6.14	2679	3.61
80-80-10-0.7	1617	0.30	1643	0.62	1869	4.45	1747	4.68
100-100-5-0.3	6247	0.60	6332	0.63	6392	3.69	6680	4.86
100-100-5-0.5	4650	0.61	4675	0.65	4769	5.00	4770	4.95
100-100-5-0.7	2832	0.61	2857	1.21	3003	2.43	3003	3.89
100-100-10-0.3	4298	0.61	5578	0.63	5621	4.88	5635	4.94
100-100-10-0.5	5426	0.61	5832	0.66	6314	3.87	9311	5.26
100-100-10-0.7	2644	0.61	2708	0.61	2739	3.84	2812	4.96
Av	3494.50	0.46	3666.67	0.60	3781.50	3.70	4065.58	4.24
120-120-5-0.3	8190	0.01	8357	0.62	8436	5.10	8719	5.25
120-120-5-0.5	6051	0.04	6079	0.88	6210	5.02	6345	5.32
120-120-5-0.7	4323	0.03	4394	1.38	4489	2.61	4489	1.94
120-120-10-0.3	8036	0.61	7660	1.25	7859	5.01	7979	5.18
120-120-10-0.5	5823	0.62	5876	0.62	5964	5.13	5966	3.34
120-120-10-0.7	4114	0.62	4155	0.74	4281	5.23	4325	2.48
150-150-5-0.3	14365	0.04	14405	1.67	15097	5.94	15097	6.64
150-150-5-0.5	1996	0.81	2158	1.61	2225	2.67	10062	1.62
150-150-5-0.7	8569	0.07	8590	1.72	8856	5.66	8856	2.87
150-150-10-0.3	13389	0.82	13443	0.86	13752	5.96	14000	6.63
150-150-10-0.5	1569	0.81	1584	1.61	2016	2.66	2046	1.61
150-150-10-0.7	8234	0.89	8272	0.84	8563	6.14	8678	7.11
200-200-5-0.3	28417	1.21	28770	1.13	29266	7.09	29453	10.63
200-200-5-0.5	19420	1.29	19448	1.33	19932	8.61	19932	11.38
200-200-5-0.7	19408	1.21	19442	1.29	19932	8.17	19932	10.45
200-200-10-0.3	28024	1.06	26949	2.36	28040	7.85	28059	8.92
200-200-10-0.5	18906	1.10	18985	1.16	19790	7.77	19882	14.97
200-200-10-0.7	18814	1.25	18876	1.20	19588	8.40	19796	14.91
300-300-5-0.3	72386	1.73	70040	1.59	73341	3.86	73718	1.68
300-300-5-0.5	55572	6.12	55657	2.76	56694	5.90	56694	1.29
300-300-5-0.7	56938	2.24	53725	3.28	56938	6.02	56938	1.80
300-300-10-0.3	71800	1.22	66572	1.94	70992	1.12	71800	1.22
300-300-10-0.5	54220	5.02	54357	3.17	56490	2.73	56671	1.70
300-300-10-0.7	56254	9.25	52513	2.97	55880	2.74	56850	1.69
Av	8486.34	1.19	8551.92	1.24	8821.15	4.74	9240.92	5.69
120-120-15-0.3	7057	8.69	7099	5.29	7274	6.54	7368	8.21
120-120-15-0.5	5437	8.94	5500	5.46	5688	6.11	5673	6.12
120-120-15-0.7	4050	6.89	3956	5.27	4157	6.51	4166	6.16
150-150-15-0.3	12545	6.07	12574	10.17	13057	12.63	12970	26.30
150-150-15-0.5	1672	4.18	1267	27.09	1778	8.13	1874	8.15
150-150-15-0.7	7920	7.42	7967	6.46	8323	9.29	8362	8.49
200-200-15-0.3	25495	13.20	25577	9.40	26429	17.45	26217	26.47
200-200-15-0.5	18304	8.58	18381	11.20	19320	11.44	19220	18.91
200-200-15-0.7	18282	14.10	18334	9.53	19361	14.35	19309	19.17
300-300-15-0.3	65438	10.27	67284	12.37	71668	22.52	67529	28.22
300-300-15-0.5	55214	12.00	55627	11.32	56577	24.67	56000	23.42
300-300-15-0.7	54630	6.32	56263	11.67	56819	15.98	56055	22.57
Av	23003.67	8.89	23319.08	10.44	24204.25	12.97	23728.58	16.85

11.5 Conclusion

In this paper, the k-clustering minimum completion problem was solved by using a rounding strategy-based algorithm. First, a starting solution was built by adapting a basic rounding procedure that selects step by step a fractional item from the mixed integer program. Second, an enhanced procedure was developed, where four local operators were considered. Third, the rounding strategy and its enhancement were embedded into an iterative search for implementing an efficient and powerful method. The performance of all proposed solution procedures was experimentally analyzed on a set of problem instances available in the literature, containing medium and large-sized instances. The experimental part showed that the proposed method remains competitive and was able

to provide new bounds when compared to the best bounds available in the literature.

General conclusion and perspectives

Conclusions

The applied transversal approaches allow the coupling of skills present within the research unit EPROAD. The proposed methods are from the field of artificial intelligence, discrete modeling resulting from applied mathematics, sensitivity analysis, and the field of process engineering for the development of intelligent cooperative methods.

The studied vegetal composite is a complex material. Changing one parameter will affect the quality of the produced material. Optimization of a vegetal-composite and consolidating the supply facilities by artificial intelligence methods and combinatorial optimization consist of two parts.

The first part presents a study of the composite compressive strength and the composite flexural strength. Considering the uncertainty and changeability of the data, we applied machine learning and artificial intelligence to predict the parameters of the experiments. Artificial intelligence can predict the parameters dynamically. A model can adapt itself based on the training data. Predictions are dynamic and the results are data-oriented.

An introduction to machine learning methods is presented in chapter 3. This chapter also presents the experimental dataset of the vegetal composite (3.3).

We proposed a methodology of prediction with regression methods in chapter 4. Where an example for the prediction problems (4.3) , regression with two dimensions (4.4) and regression with more dimensions (4.5) are presented in this chapter.

In chapter 5 we extended the regression method (presented in chapter 4). This chapter (chapter 5) presents a methodology to predict the compressive strength of a vegetal composite using gradient descent method. The principal steps to apply the optimization method are explained in 5.3. A sensitivity analysis on the learning rate and the number of iterations is performed in the computational analysis 5.4 to find the best hyper-parameters that fit the algorithm.

We assumed the complexity and interchangeability of the data, by considering that

the effect of the parameters can be non-linear. Therefore, we proposed an approach based on the artificial neural networks to predict the compressive and flexural strengths of the composite in chapter 6. An introduction to the artificial neural networks is presented in 6.1. The proposed approach to train the data is presented in 6.2. Several neural network models were designed and compared, parameters to design the networks, and error analysis is presented in 6.3.

Which one can provide the best prediction model?

From the reported results in the experimental studies of chapter 5 and chapter 6, we can compare the two prediction methods gradient descent (GD) and artificial neural networks(ANNs) on the compressive strength of the vegetal composite. The compared criteria are as follows:

- Complexity of the computations: In general, GD is easier to apply since the calculations are less complex.

- Number of the parameters needed to be tuned: There are fewer parameters to tune the algorithm in GD than ANNs. Parameters to tune in GD are the number of iterations, and learning rate. Whereas, the parameters to tune in ANNs are the number of iterations, learning rate, number of hidden layers, number of neurons in the hidden layers, activation function, cost function, back propagation and optimization method.

- Future developments: The resulted model provided by GD is linear and provides linear equations that make it possible to apply bi-objective models on the target value. In the other hand, ANNs can be developed by improving any of the hyper-parameters. In this study we applied sensitivity analysis on the hyper-parameters to tune the algorithm.

- Real world applications: The achieved linear equation from GD can be programmed in simple calculators. Contrarily, ANNs required advanced calculation tools to program and optimize the non-linear equations.

- The reliability and precision of the predictions: By comparing the best prediction models, the overall prediction errors with ANNs are lower than the one with GD. As reported in table 5.3 in chapter 5 and table 6.3 in chapter 6, the value of the mean squared errors (MES) on the best prediction results with GD and ANNs on the training set are 0.239 and 1.03E-05, on the test set 0.369, 0.09 respectively.

The last chapter of the first part, studies the vegetal composite with a distinict perspective. Due to the importance of the composite compressive strength test, the last chapter 7, presents a different view of analyzing the coated shive composite by predicting the parameters of the test.

The two batch and stochastic gradient descent methods are compared in 7.3. As also reported in 7.4, the batch version performs generally better than the stochastic version whenever a few number of samples is considered. We analyzed the fluctuations of the both versions. The batch gradient descent seems to converge smoothly towards the final solution while the stochastic version causes aggressive oscillations before tending towards a plausible solution. We also note that the average runtime consumed by the batch version is equal to 0.91 seconds while the stochastic version needs 63.54 seconds to converge.

Deterministic and stochastic heuristics are often used to approximately solve the clustering problem. Establishing an adaptive model to the data with heuristic and meta-heuristic models in combinatorial optimizations will ensure the sustainability and reliability of the production of vegetal composites. Moreover, convergence results will lead to a reliable business model and a competitive purchasing and marketing strategy.

The second part proposed K-clustering method to consolidate the supply facilities in order to ensure continuous production and distribution of the studied vegetal material. In this part, the k-clustering minimum completion problem is studied with several combinatorial optimization models. Different models are proposed and the results of applying benchmark dataset are compared with the literature.

Chapter 8 emphasized the importance of a sustainable supply chain and introduced combinatorial optimizations. An introduction to the k-clustering minimum bi-clique completion problem is presented in chapter 9. A review of the other studies and applications of the studied K-clustering minimum biclique completion problem is presented in 9.2. The benchmark dataset used for this part is introduced in 9.2. Mathematical formulations, the applied linearization method, and an example of the problem is presented in 9.3.

We proposed powerful solution methods for the k-clustering minimum bi-clique completion problem. Chapter 10 introduced the rounding method. An enhanced method is presented by incorporating it with the local branching. The augmented version of the rounding search and the main procedures of combining the rounding search with local branching strategy is presented in 10.2. The benchmark dataset was used to evaluate the model. The effect of the local branching and the behavior of this model is shown in 10.3.

Chapter 11 propose a rounding strategy-based algorithm for this problem. This chapter presents the main method 11.1. Enhancing solution methods 11.2 were proposed using hill-climbing strategy and a decomposition method. The proposed solution is evaluated through statistical analysis in section 11.4. Presented comparisons between

the other methods show the superiority of the proposed model.

Future studies

The optimization of composites formulated with vegetal materials and the reuse of agricultural waste is an alternative solution in the context of global changes. The optimization of the vegetal composites is a complex subject, and it requires several multi-discipline aspects.

On the one hand, considering the complexity and diversity of this subject, this thesis can be extended with several aspects. For example, a study of a sustainable supply chain of the vegetal-based composite, could facilitate the development of eco-materials industries and contribute to the circular economy.

In addition, the study presented herein could be adapted to other disciplines. The main strategy of the proposed solution methods can be applied in several combinatorial optimization problems.

On the other hand, presenting bi-objective models in a transversal approach, incorporating other advanced optimization models to optimize the neural networks is suggested.

Bibliography

Aider, M., Gacem, O., & Hifi, M. (2020). Branch and solve strategies-based algorithm for the quadratic multiple knapsack problem. *Journal of the Operational Research Society*, 1–18.

Akeb, H., Hifi, M., & Mounir, M. (2011). Local branching-based algorithms for the disjunctively constrained knapsack problem. *Computers & Operations Research*, *60*(4), 811–820.

Anaya-Arenas, A., Prodhon, C., Renaud, J., & Ruiz, A. (2019). An iterated local search for the biomedical sample transportation problem with multiple and interdependent pickups. *Journal of the Operational Research Society*, 1–16. doi:doi.org/10.1080/01605682.2019.1657369

Arunkumar, N., Mohammed, M. A., Mostafa, S. A., Ibrahim, D. A., Rodrigues, J. J., & de Albuquerque, V. H. C. (2020). Fully automatic model-based segmentation and classification approach for mri brain tumor using artificial neural networks. *Concurrency and Computation: Practice and Experience*, *32*(1), e4962.

Asteris, P., Kolovos, K., Douvika, M., & Roinos, K. (2016). Prediction of self-compacting concrete strength using artificial neural networks. *European Journal of Environmental and Civil Engineering*, *20*(sup1), s102–s122.

Barnat-Hunek, D., Smarzewski, P., & Brzyski, P. (2017). Properties of hemp–flax composites for use in the building industry. *Journal of Natural Fibers*, *14*(3), 410–425. doi:10.1080/15440478.2016.1212764. eprint: https://doi.org/10.1080/15440478.2016.1212764

Barnhart, C., Johnson, E. L., Nemhauser, G. L., Savelsbergh, M. W., & Vance, P. H. (1998). Branch-and-price: Column generation for solving huge integer programs. *Operations research*, *46*(3), 316–329.

Baykasoğlu, A., Dereli, T., & Tanış, S. (2004). Prediction of cement strength using soft computing techniques. *Cement and concrete research*, *34*(11), 2083–2090.

Baykasoğlu, A., Öztaş, A., & Özbay, E. (2009). Prediction and multi-objective optimization of high-strength concrete parameters via soft computing approaches. *Expert Systems with Applications*, *36*(3), 6145–6155.

Bederina, M., Laidoudi, B., Goullieux, A., Khenfer, M., Bali, A., & Quéneudec, M. (2009). Effect of the treatment of wood shavings on the physico-mechanical characteristics of wood sand concretes. *Construction and Building Materials*, *23*(3), 1311–1315.

Bengio, Y., Boulanger-Lewandowski, N., & Pascanu, R. (2013). Advances in optimizing recurrent networks. In *2013 ieee international conference on acoustics, speech and signal processing* (pp. 8624–8628). IEEE.

C684-99, A. (1999). Standard test method for making, accelerated curing, and testing concrete compression test specimens. ASTM International West Conshohocken.

Catto, J. W., Linkens, D. A., Abbod, M. F., Chen, M., Burton, J. L., Feeley, K. M., & Hamdy, F. C. (2003). Artificial intelligence in predicting bladder cancer outcome: A comparison of neuro-fuzzy modeling and artificial neural networks. *Clinical Cancer Research*, *9*(11), 4172–4177.

Cherfi, N., & Hifi, M. (2009). Hybrid algorithms for the multiple-choice multi-dimensional knapsack problem. *International Journal of Operational Research*, *5*(1), 89–109.

Chollet, F. et al. (2018). *Deep learning with python*. Manning New York.

Collette, Y., & Siarry, P. (2002). *Optimisation multiobjectif*. Editions Eyrolles.

Cung, V.-D., Hifi, M., & Le Cun, B. (2000). Constrained two-dimensional cutting stock problems a best-first branch-and-bound algorithm. *International Transactions in Operational Research*, *7*(3), 185–210.

Dahmani, I., Ferroum, M., Hifi, M., & Sadeghsa, S. (2020). A hybrid swarm optimization-based algorithm for the set-union knapsack problem. In *2020 7th international conference on control, decision and information technologies (codit)* (Vol. 1, pp. 1162–1167). IEEE.

Dantas, S., Groshaus, M., Guedes, A., Machado, R., Ries, B., & Sasaki, D. (2007). On star and biclique edge-colorings. *International Transactions in Operational Research*, *24*(1-2), 339–346.

Daskin, M. S., Snyder, L. V., & Berger, R. T. (2005). Facility location in supply chain design. In *Logistics systems: Design and optimization* (pp. 39–65). Springer.

de Siqueira Tango, C. (1998). An extrapolation method for compressive strength prediction of hydraulic cement products. *Cement and Concrete Research*, *28*(7), 969–983. doi:https://doi.org/10.1016/S0008-8846(98)00074-X

Desaulniers, G., Desrosiers, J., & Spoorendonk, S. (2010). The vehicle routing problem with time windows: State-of-the-art exact solution methods. *Wiley encyclopedia of operations research and management science*.

Al-Douri, T. (2018). *Méthodes heuristique pour les problèmes de type knapsack* (Doctoral dissertation, Université de Picardie Jules Verne, Amiens).

Duginov, O. (2015). On the complexity of the clustering minimum biclique completion problem.

Dupré, B. (2005). *Contribution à la valorisation des coproduits du lin: Impact du vécu et de la variabilité génétique sur les propriétés des composites élaborés* (Doctoral dissertation, Amiens).

El Baz, D., Hifi, M., Wu, L., & Shi, X. (2016). A parallel ant colony optimization for the maximum-weight clique problem. In *2016 ieee international parallel and distributed processing symposium workshops (ipdpsw)* (pp. 796–800). IEEE.

FAOSTAT. (2020). Fao statistical database. Retrieved from http://faostat.fao.org/site/567/default. ((accessed: 04.11.2020))

Faure, N., Chrétienne, P., Gourdin, E., & Sourd, F. (2007). Biclique completion problems for multicast network design. *Discrete Optimization*, *4*(3-4), 360–377.

Feo, T. A., & Resende, M. G. (1995). Greedy randomized adaptive search procedures. *Journal of global optimization*, *6*(2), 109–133.

Fischetti, M., & Lodi, A. (2003). Local branching. *Mathematical programming*, *98*(1), 23–47.

Fischetti, M., Polo, C., & Scantamburlo, M. (2004). A local branching heuristic for mixed-integer programs with 2-level variables, with an application to a telecommunication network design problem. *Networks: An International Journal*, *44*(2), 61–72.

Glover, F., Laguna, M., & Marti, R. (2003). Scatter search. In A. Ghosh & S. Tsutsui (Eds.), *Advances in evolutionary computing. natural computing series* (pp. 519–537). doi:https://doi.org/10.1007/978-3-642-18965-4_20

Goodfellow, I., Bengio, Y., Courville, A., & Bengio, Y. (2016). *Deep learning*. MIT press Cambridge.

Goullieux, A., Hifi, M., & Sadeghsa, S. (2020a). An iterative descent method for predicting the compressive cement strength estimated parameters. In *Communication papers of the 2020 federated conference on computer science and information systems* (p. 7).

Goullieux, A., Hifi, M., & Sadeghsa, S. (2020b). Self learning strategy for the team orienteering problem (sls-top). In *Icores* (pp. 336–343).

Gualandi, S. (2009). K-clustering minimum biclique completion via a hybrid cp and sdp approach. In *Van hoeve, w-j., hooker, j.n. (eds.): Integration of ai and or techniques in constraint programming for combinatorial optimization problems* (Vol. 5547, pp. 87–101). LNCS, Springer Berlin Heidelberg.

Gualandi, S., Maffioli, F., & Magni, C. (2013). Biclique completion problems for multicast network design. *International transactions in operational research*, 101–117.

Habibzadeh, F., & Habibzadeh, P. (2015). How much precision in reporting statistics is enough? *Croatian medical journal*, *56*(5), 490.

Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The elements of statistical learning: Data mining, inference, and prediction*. Springer Science & Business Media.

Hiemstra, Y. (1996). Linear regression versus back propagation networks to predict quarterly stock market excess returns. *Computational Economics*, *9*(1), 67–76.

Hifi, M. (2013). An iterative rounding search-based algorithm for the disjunctively constrained knapsack problem. *Engineering Optimization*, *46*(8), 1109–1122.

Hifi, M., & Michrafy, M. (2013). An iterative rounding search-based algorithm for the disjunctively constrained knapsack problem. *Engineering Optimization*, *46*(8), 1109–1122.

Hifi, M., Moussa, I., Saadi, T., & Saleh, S. (2015). An adaptive neighborhood search for k-clustering minimum bi-clique completion problems. In *Proceedings of modellling,*

computation and optimization in information systems and management sciences (pp. 15–25). Springer, Cham.

Hifi, M., Paschos, V., Zissimopoulos, V., et al. (1994). *Circular cutting problem: A simulated annealing approach*. Université Panthéon-Sorbonne (Paris 1).

Hifi, M., & Sadeghsa, S. (2020a). Adaptation of the rounding search-based algorithm for the k-clustering minimum completion problem. In *2020 7th international conference on control, decision and information technologies (codit)* (Vol. 1, pp. 1150–1155). IEEE.

Hifi, M. (1997). A genetic algorithm-based heuristic for solving the weighted maximum independent set and some equivalent problems. *Journal of the Operational Research Society*, *48*(6), 612–622.

Hifi, M. (1998). Exact algorithms for the guillotine strip cutting/packing problem. *Computers & Operations Research*, *25*(11), 925–940.

Hifi, M. (2001). Exact algorithms for large-scale unconstrained two and three staged cutting problems. *Computational Optimization and Applications*, *18*(1), 63–88.

Hifi, M., Michrafy, M., & Sbihi, A. (2006). A reactive local search-based algorithm for the multiple-choice multi-dimensional knapsack problem. *Computational Optimization and Applications*, *33*(2-3), 271–285.

Hifi, & Sadeghsa. (2020b). Effect of local branching on the iterative rounding-based method: The case of k-clustering minimum completion problems. *Cybernetics and Systems*.

Hill, A., & Voß, S. (2018). Generalized local branching heuristics and the capacitated ring tree problem. *Discrete Applied Mathematics*, *242*, 34–52.

Al-Iedani, N., Hifi, M., & Saadi, T. (2016). Neighborhood search-based heuristic for the k-clustering minimum biclique completion problem. In *Proceedings of the international conference on control, decision and information technologies (codit)* (pp. 639–643). IEEE.

Julián-Ortiz, D., Vicente, J., Pogliani, L., & Besalú, E. (2018). Modeling properties with artificial neural networks and multilinear least-squares regression: Advantages and drawbacks of the two methods. *Applied Sciences*, *8*(7), 1094.

Kabir, A., Hasan, M., & Miah, M. K. (2013). Strength prediction model for concrete. *International Journal on Civil and Environmental Engineering*, *2*(1), 14.

Khazma, M., Goullieux, A., Dheilly, R., & Quéneudec, M. (2007). Lignocellulosic light weight concrete: Comparison between different coating processes. *Advances in Eco-materials*, 143–151. doi:978-1602316

Khazma, M., Goullieux, A., Dheilly, R. M., Laidoudi, B., & Queneudec, M. (2011). Impact of aggregate coating with a pec elastomer on properties of lightweight flax shive concrete. *Industrial Crops and Products*, *33*(1), 49–56.

Khazma, M., Goullieux, A., Dheilly, R.-M., & Quéneudec, M. (2012). Coating of a lignocellulosic aggregate with pectin/polyethylenimin mixtures: Effects on flax shive and cement-shive composite properties. *Cement and Concrete Composites*, *34*(2), 223–230. doi:https://doi.org/10.1016/j.cemconcomp.2011.07.008

Khazma, M., Goullieux, A., Dheilly, R.-M., Rougier, A., & Quéneudec, M. (2014). Optimization of flax shive-cementitious composites: Impact of different aggregate treatments using linseed oil. *Industrial Crops and Products*, *61*, 442–452.

Koch, S., & Wäscher, G. (2011). A grouping genetic algorithm for the order batching problem in distribution warehouses. *Working Paper Series*.

Koch, S., & Wäscher, G. (2016). A grouping genetic algorithm for the order batching problem in distribution warehouses. *Journal of Business Economics*, *86*(1-2), 131–153.

Li, B., Delpha, C., Diallo, D., & Migan-Dubois, A. (2020). Application of artificial neural networks to photovoltaic fault detection and diagnosis: A review. *Renewable and Sustainable Energy Reviews*, 110512.

Li, W., Ni, P., & Yi, Y. (2019). Comparison of reactive magnesia, quick lime, and ordinary portland cement for stabilization/solidification of heavy metal-contaminated soils. *Science of The Total Environment*, *671*, 741–753.

Lübbecke, M. E., & Desrosiers, J. (2005). Selected topics in column generation. *Operations research*, *53*(6), 1007–1023.

Mahieu, A., Lenormand, H., Leblanc, N., & Vivet, A. (2015). 100% biobased particleboards based on new agricultural wastes. *Academic Journal of Civil Engineering*, *33*(2), 705–710.

Malagavelli, V., & Manalel, P. (2014). Modeling of compressive strength of admixture-based self compacting concrete using fuzzy logic and artificial neural networks. *Asian Journal of Applied Sciences*, *7*(7), 536–551.

Mansour, N., Tabbara, H., & Dana, T. (2004). A genetic algorithm approach for regrouping service sites. *Computers & Operations Research*, *31*(8), 1317–1333.

Mansouri, I., Ozbakkaloglu, T., Kisi, O., & Xie, T. (2016). Predicting behavior of frp-confined concrete using neuro fuzzy, neural network, multivariate adaptive regression splines and m5 model tree techniques. *Materials and Structures*, *49*(10), 4319–4334.

McCarthy, J., & Feigenbaum, E. A. (1990). In memoriam: Arthur samuel: Pioneer in machine learning. *AI Magazine*, *11*(3), 10–10.

Al-Mohamadawi, A., Benhabib, K., Dheilly, R.-M., & Goullieux, A. (2016). Influence of lignocellulosic aggregate coating with paraffin wax on flax shive and cement-shive composite properties. *Construction and Building Materials*, *102*, 94–104. doi:https://doi.org/10.1016/j.conbuildmat.2015.10.190

Al-Mohamadawi, A., Benhabib, K., Dheilly, R.-M., & Goullieux, A. (2020). Hygric behavior of cement composites elaborated with flax shives, a byproduct of the linen industry. *Waste and Biomass Valorization*, *11*(9), 5053–5066. doi:https://doi.org/10.1007/s12649-019-00798-4

Mohri, M., Rostamizadeh, A., & Talwalkar, A. (2018). *Foundations of machine learning*. MIT press.

Monreal, P., Mboumba-Mamboundou, L., Dheilly, R., & Quéneudec, M. (2011). Effects of aggregate coating on the hygral properties of lignocellulosic composites. *Cement and Concrete Composites*, *33*(2), 301–308.

Moussa, I. (2015). *Moèèles de résolution approchée et efficace pour les problèmes des réseaux de transport et de télécommunication* (Doctoral dissertation, Université de Picardie Jules Verne).

Nair, V., & Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In *Icml*.

Ng, A. (2018). Machine learning. stanford university. Retrieved from https://www.%20coursera.%20org/learn/machine-learning. ((accessed: 14.04.2017))

Olorunnisola, A. O. (2009). Effects of husk particle size and calcium chloride on strength and sorption properties of coconut husk–cement composites. *Industrial Crops and Products*, *29*(2-3), 495–501.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... Dubourg, V., et al. (2011). Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, *12*, 2825–2830.

Petrovic, E. K., Vale, B., & Zari, M. P. (2017). *Materials for a healthy, ecological and sustainable built environment: Principles for evaluation*. Woodhead Publishing.

Plevris, V., & Asteris, P. (2014). Modeling of masonry compressive failure using neural networks. *Proceedings of the OPT-i*, 2843–2861.

Pu, S., Zhu, Z., Song, W., Wan, Y., Wang, H., & Xu, X. (2020). Comparative study of compressibility and deformation properties of silt stabilized with lime, lime, and cement, and seu-2 binder. *Arabian Journal for Science and Engineering*, *45*(5), 4125–4139.

Pu, S., Zhu, Z., Zhao, L., Song, W., Wan, Y., Huo, W., ... Hu, L. (2020). Microstructural properties and compressive strength of lime or/and cement solidified silt: A multi-scale study. *Bulletin of Engineering Geology and the Environment*, *79*(10), 5141–5159.

Purton, M. (1970). Comparison of the binding properties of hydrated lime and cement in brick manufacture. *Journal of Applied Chemistry*, *20*(9), 293–299.

Reihaneh, M., & Ghoniem, A. (2018). A branch-cut-and-price algorithm for the generalized vehicle routing problem. *Journal of the Operational Research Society*, *69*(2), 307–318.

Reinelt, G. (1991). Tsplib—a traveling salesman problem library. *ORSA journal on computing*, *3*(4), 376–384.

Ruder, S. (2016). An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*.

Saleh, & Hifi. (2017). A fast method for optimizing the k-clustering bi-clique completion problem in telecommunication. *Journal of Duhok University*, 175–183.

Samuel, A. L. (1959). Some studies in machine learning using the game of checkers. *IBM Journal of research and development*, *3*(3), 210–229.

Shariati, M., Mafipour, M. S., Haido, J. H., Yousif, S. T., Toghroli, A., Trung, N. T., & Shariati, A. (2020). Identification of the most influencing parameters on the properties of corroded concrete beams using an adaptive neuro-fuzzy inference system (anfis). *Steel and Composite Structures*, *34*(1), 155–170.

Shariati, M., Mafipour, M. S., Mehrabi, P., Ahmadi, M., Wakil, K., Trung, N. T., & Toghroli, A. (2020). Prediction of concrete strength in presence of furnace slag and fly ash using hybrid ann-ga (artificial neural network-genetic algorithm). *Smart Structures and Systems*, *25*(2), 183–195.

Shariati, M., Mafipour, M. S., Mehrabi, P., Bahadori, A., Zandi, Y., Salih, M. N., ... Poi-Ngian, S. (2019). Application of a hybrid artificial neural network-particle swarm optimization (ann-pso) model in behavior prediction of channel shear connectors embedded in normal and high-strength concrete. *Applied Sciences*, *9*(24), 5534.

Shaw, P. (1998). Using constraint programming and local search methods to solve vehicle routing problems. In *International conference on principles and practice of constraint programming* (pp. 417–431). Springer.

Simpson, J., & Weiner, E. (1989). *The oxford english dictionary (20 volume set)*. Oxford University Press, USA.

Snell, L. M., Van Roekel, J., & Wallace, N. D. (1989). Predicting early concrete strength. *Concrete International*, *11*(12), 43–47.

Sola, J., & Sevilla, J. (1997). Importance of input data normalization for the application of neural networks to complex industrial problems. *IEEE Transactions on nuclear science*, *44*(3), 1464–1468.

Tanksale, A., & Jha, J. (2020). Benders decomposition approach with heuristic improvements for the robust foodgrain supply network design problem. *Journal of the Operational Research Society*, *71*(1), 16–36.

Teo, C. P., Ou, J., & Goh, M. (2001). Impact on inventory costs with consolidation of distribution centers. *Iie Transactions*, *33*(2), 99–110.

Topcu, I. B., & Sarıdemir, M. (2008). Prediction of compressive strength of concrete containing fly ash using artificial neural networks and fuzzy logic. *Computational Materials Science*, *41*(3), 305–311.

Tsivilis, S., & Parissakis, G. (1995). A mathematical model for the prediction of cement strength. *Cement and concrete research*, *25*(1), 9–14.

van der Vlist, P., & Broekmeulen, R. A. (2006). Retail consolidation in synchronized supply chains. *Zeitschrift für Betriebswirtschaft*, *76*(2), 165–176.

Wang, F. (2020). A successive domain-reduction scheme for linearly constrained quadratic integer programming problems. *Journal of the Operational Research Society*, 1–14.

Wikipedia. (2021). Arthur samuel. [Online; accessed 22-February-2021]. Retrieved from https://en.wikipedia.org/wiki/Arthur_Samuel

Wu, G.-D., & Lo, S.-L. (2010). Effects of data normalization and inherent-factor on decision of optimal coagulant dosage in water treatment by artificial neural network. *Expert Systems with Applications*, *37*(7), 4974–4983.

Yang, G. R., & Wang, X.-J. (2020). Artificial neural networks for neuroscientists: A primer. *arXiv preprint arXiv:2006.01001*.

Yousef, L. (2017). *Contribution á la résolution des problèmes de placement en trois dimensions* (Doctoral dissertation, Université de Picardie Jules Verne, Amiens).

Yuen, K.-V., & Lam, H.-F. (2006). On the complexity of artificial neural networks for smart structures monitoring. *Engineering Structures*, *28*(7), 977–984. doi:https://doi.org/10.1016/j.engstruct.2005.11.002

Appendices

Appendix A

Herein, the experimental data for each experiment related to both lime and cement are represented in Tables 1 and 2, respectively

Table 1: Experimental data for each experiment with lime as coating substance

Coating Substance	Y_1	Y_2	X_1	X_2	X_3	X_4	X_5	X_6	X_7
Lime	0.33	0.50	71.031	240	1.357	6.65	812.227	5.454	2.756
Lime	0.33	0.50	70.181	266.1	1.076	5.994	810.469	5.302	2.69
Lime	0.33	0.50	74.639	264.15	0.749	6.256	836.094	2.958	2.656
Lime	0.33	0.75	84.603	260.95	2.083	8.138	1025.339	2.916	2.554
Lime	0.33	0.75	86.271	256.8	1.638	8.981	1047.93	2.645	1.87
Lime	0.33	0.75	81.717	265.55	1.872	9.556	1011.992	3.894	2.295
Lime	0.33	1.00	68.983	291.95	2.012	4.738	911.354	4.606	1.838
Lime	0.33	1.00	70.052	296.15	1.732	4.8	910.859	4.41	1.903
Lime	0.33	1.00	67.112	297.3	1.638	5.1	911.875	5.42	2.095
Lime	0.33	2.00	71.285	281.3	0.281	0.631	705.039	3.238	1.722
Lime	0.33	2.00	71.633	292.6	0.257	0.694	725.938	3.77	1.31
Lime	0.33	2.00	73.438	298.5	0.304	0.675	691.328	6.492	1.431
Lime	0.50	0.50	77.848	221.55	0.491	1.788	643.555	5.454	2.798
Lime	0.50	0.50	81.236	215.4	0.515	1.575	639.688	5.302	3.28
Lime	0.50	0.50	69.654	215.35	0.655	1.95	651.172	2.958	2.898
Lime	0.50	0.75	73.469	216.75	0.819	2.813	685.195	3.545	2.848
Lime	0.50	0.75	68.654	217.65	0.796	2.769	671.016	4.944	3.227
Lime	0.50	0.75	78.013	227.65	0.866	2.781	705.352	3.683	2.972
Lime	0.50	1.00	73.763	211.05	1.544	6.644	914.18	3.211	2.075
Lime	0.50	1.00	72.059	226.55	1.381	6.269	897.266	4.122	3.429
Lime	0.50	1.00	75.692	218.45	1.661	6.244	930.742	2.403	3.047
Lime	0.50	2.00	71.701	216.1	0.819	1.938	734.883	6.48	3.355
Lime	0.50	2.00	70.274	220.3	0.796	2.25	731.172	4.063	1.495
Lime	0.50	2.00	76.993	219.65	0.889	1.963	730.586	7.784	2.686
Lime	0.67	0.50	86.806	215.35	0.959	2.55	696.172	5.581	3.17
Lime	0.67	0.50	92.871	207.3	1.076	1.925	700.234	5.774	3.565
Lime	0.67	0.50	70.475	213.3	0.983	2.2	680.547	5.56	3.323
Lime	0.67	0.75	94.567	192.05	0.866	3	702.383	3.784	2.734
Lime	0.67	0.75	90.641	195.3	1.17	2.669	693.047	6.846	2.706
Lime	0.67	0.75	98.472	194.7	1.006	3.45	716.25	3.327	2.85
Lime	0.67	1.00	86.515	207.5	1.685	4.044	779.18	5.405	2.814
Lime	0.67	1.00	81.803	213.2	1.685	3.631	760.898	3.29	2.63
Lime	0.67	1.00	47.933	205	1.802	3.25	795.273	2.976	2.697
Lime	0.67	2.00	83.439	203.1	2.574	5	836.198	3.542	2.132
Lime	0.67	2.00	82.937	197.15	2.317	4.75	792.344	2.695	2.405
Lime	0.67	2.00	80.99	190.65	2.34	5.25	839.922	3.263	2.311
Lime	1.00	0.50	92.349	183.3	0.608	1.581	651.263	4.094	4.788
Lime	1.00	0.50	100.362	177.05	0.562	1.419	661.016	6.6	4.876
Lime	1.00	0.50	93.248	191.85	0.585	1.381	658.477	8.173	4.371
Lime	1.00	0.75	88.156	152.65	0.468	1.556	676.641	5.421	3.872
Lime	1.00	0.75	87.234	162.25	0.562	1.756	690.977	5.019	4.056
Lime	1.00	0.75	90.253	166.85	0.608	1.963	672.031	5.425	4.35
Lime	1.00	1.00	90.37	174.6	0.842	2.438	686.953	5.101	3.318
Lime	1.00	1.00	82.245	169.6	0.772	1.956	693.633	4.355	3.862
Lime	1.00	1.00	99.006	166.95	0.749	2.588	694.297	5.636	3.761
Lime	1.00	2.00	88.778	166.85	0.749	2.675	779.206	4.045	2.589
Lime	1.00	2.00	81.107	176.6	0.913	2.563	742.422	4.192	2.724
Lime	1.00	2.00	98.734	177.25	0.936	2.569	782.109	5.479	3.101
Lime	2.00	0.50	113.138	143.15	0.187	0.706	602.448	6.125	8.108
Lime	2.00	0.50	136.398	137	0.164	1.075	616.055	7.61	7.911
Lime	2.00	0.50	93.137	139.85	0.211	0.713	610.234	6.318	8.648
Lime	2.00	0.75	112.381	130.75	0.187	0.75	625.43	6.968	4.995
Lime	2.00	0.75	93.471	137	0.257	0.719	625.508	5.617	8.477
Lime	2.00	0.75	118.891	136.85	0.14	1.088	629.805	6.367	7.221
Lime	2.00	1.00	109.414	131.75	0.164	0.838	591.771	6.437	7.278
Lime	2.00	1.00	118.127	129.7	0.281	1.069	583.633	6.021	7.732
Lime	2.00	1.00	101.876	131.35	0.211	0.956	605.313	6.385	7.577
Lime	2.00	2.00	114.51	130.55	0.281	0.938	693.854	5.802	5.108
Lime	2.00	2.00	129.682	135.25	0.304	1.131	686.094	6.557	7.108
Lime	2.00	2.00	117.383	131.5	0.257	1.106	669.883	5.179	6.436

Table 2: Experimental data for each experiment with cement as coating substance

Coating Substance	Y_1	Y_2	X_1	X_2	X_3	X_4	X_5	X_6	X_7
Cement	0.33	0.50	59.946	320.15	0.468	1.25	811.406	5.261	3.708
Cement	0.33	0.50	69.067	304.45	0.679	1.313	829.258	6.503	3.827
Cement	0.33	0.50	54.545	330.2	0.491	1.313	791.328	6.503	3.731
Cement	0.33	0.75	55.089	320.35	1.076	2.1	905.586	3.898	3.983
Cement	0.33	0.75	54.961	311.05	0.702	2	909.258	5.947	3.503
Cement	0.33	0.75	80.284	335.9	0.936	1.813	906.719	5.17	3.152
Cement	0.33	1.00	60.361	340.75	1.006	3.519	924.961	6.171	3.401
Cement	0.33	1.00	70.052	320.55	0.796	2.875	974.297	3.939	3.144
Cement	0.33	1.00	54.042	323.15	1.498	2.55	901.367	4.916	3.234
Cement	0.33	2.00	79.858	292.25	1.568	6.563	995.234	3.751	3.184
Cement	0.33	2.00	50.47	294.75	1.615	6.556	974.961	4.435	3.452
Cement	0.33	2.00	85.714	315.25	1.568	5.913	965.195	5.555	3.326
Cement	0.50	0.50	73.815	243.9	0.702	1.619	760.586	5.309	3.753
Cement	0.50	0.50	82.759	224.45	0.515	1.6	769.922	4.186	4.227
Cement	0.50	0.50	72.098	232.1	0.796	1.538	754.219	5.63	3.598
Cement	0.50	0.75	66.928	235.5	0.679	1.669	837.852	4.462	3.49
Cement	0.50	0.75	67.366	233.85	0.585	1.744	838.359	5.126	3.612
Cement	0.50	0.75	63.943	251.35	0.749	1.8	792.5	4.584	3.679
Cement	0.50	1.00	66.03	234.35	1.404	2.7	844.414	5.328	3.634
Cement	0.50	1.00	61.719	224.3	1.31	2.006	838.359	4.595	3.487
Cement	0.50	1.00	91.884	244.4	1.264	1.913	837.852	5.693	3.176
Cement	0.50	2.00	87.66	263.35	2.808	6.713	1055.547	4.4	3.588
Cement	0.50	2.00	89.893	249.4	1.895	6.506	1099.492	3.765	3.739
Cement	0.50	2.00	67.757	254.4	2.317	6.194	1073.008	0	3.208
Cement	0.67	0.50	81.06	211.95	0.632	1.75	725.82	6.845	6.166
Cement	0.67	0.50	83.977	202.2	0.702	1.35	748.945	5.143	6.419
Cement	0.67	0.50	91.859	190.05	0.866	1.413	737.305	5.1	5.023
Cement	0.67	0.75	65.435	206.2	1.544	2.606	760	5.334	5.468
Cement	0.67	0.75	79.749	191.35	0.842	3.463	743.047	5.862	5.065
Cement	0.67	0.75	128.798	195.4	0.913	2.175	743.828	5.068	5.042
Cement	0.67	1.00	67.597	198.7	1.24	3.225	726.602	3.808	4.254
Cement	0.67	1.00	86.014	207.9	1.568	2.875	733.672	4.879	4.55
Cement	0.67	1.00	87.159	203.4	0.796	2.813	736.172	3.811	3.946
Cement	0.67	2.00	80	184.95	3.276	6.444	912.773	3.347	3.924
Cement	0.67	2.00	82.085	211	1.919	7.406	937.266	5.407	4.406
Cement	0.67	2.00	85.475	200.95	2.153	5.725	898.477	2.858	4.041
Cement	1.00	0.50	102.193	171.65	0.421	0.85	661.25	7.05	5.788
Cement	1.00	0.50	108.021	159.6	0.374	0.788	633.281	5.581	6.014
Cement	1.00	0.50	81.448	161.55	0.304	0.9	612.227	7.086	6.36
Cement	1.00	0.75	87.454	167.1	0.632	1.369	603.789	6.253	6.566
Cement	1.00	0.75	87.234	163.4	0.445	0.969	634.336	5.371	5.708
Cement	1.00	0.75	90.253	167	0.398	1.25	606.172	7.599	5.73
Cement	1.00	1.00	106.072	163.15	0.585	1.438	684.57	5.877	4.161
Cement	1.00	1.00	85.666	160.2	0.515	1.038	686.211	6.021	4.535
Cement	1.00	1.00	82.213	168	0.515	1.519	686.445	4.515	5.808
Cement	1.00	2.00	112.333	152.3	1.31	2.863	753.047	4.71	3.886
Cement	1.00	2.00	113.636	154.3	1.147	2.206	724.414	5.568	4.174
Cement	1.00	2.00	46.812	168.8	1.193	2.575	769.063	4.969	3.871
Cement	2.00	0.50	112.762	140.65	0.257	0.388	596.758	7.128	6.46
Cement	2.00	0.50	110.828	138.9	0.164	0.313	627.344	7.558	5.191
Cement	2.00	0.50	126.966	141.25	0.187	0.488	592.93	7.655	6.761
Cement	2.00	0.75	123.774	135.75	0.351	0.575	606.836	6.769	6.381
Cement	2.00	0.75	127.657	134.6	0.257	0.475	575.313	6.545	6.763
Cement	2.00	0.75	132.538	138.45	0.164	0.463	607.109	7.525	5.473
Cement	2.00	1.00	119.208	133.05	0.281	0.563	612.227	5.988	4.385
Cement	2.00	1.00	119.938	130.6	0.234	0.506	583.945	6.35	4.583
Cement	2.00	1.00	131.653	134.25	0.234	0.5	578.242	5.94	5.347
Cement	2.00	2.00	124.673	130.6	0.351	0.7	574.57	5.734	4.185
Cement	2.00	2.00	120.671	135.3	0.421	1.006	599.063	5.164	3.695
Cement	2.00	2.00	97.852	133.95	0.328	0.794	556.094	6.061	4.909

Table 3: Experimental data for each experiment with none treated shives

Coating Substance	Y_1	Y_2	X_1	X_2	X_3	X_4	X_5	X_6	X_7
none	0	0	209.854	96.500	0.164	0.406	566.719	8.581	8.400
none	0	0	222.204	103.350	0.187	0.463	580.352	8.065	8.657
none	0	0	211.602	99.650	0.164	0.338	590.000	9.238	9.235

Appendix B

Herein, the proposed N-layer models with $N = \{1, 2, 3\}$ to predict the compressive strength of a vegetal composite are reported. The models are sorted individually based on the highest R^2 value in the test set. From the 3 proposed models, a set of 50 best configurations of each model (a total of 150 best configurations) are presented. As in results, the model with 3 layers, normalized with "mean normalization method" and "tanh" as activation function has the most R^2 value in the test set.

Table 4: 150 best configurations with artificial neural networks to predict the compressive strength

rank	type	normalisation	activation	layer-format	MSE(training set)	R2(training set)	MSE(test set)	R2(test set)
1	3Layer	meanNorm	tanh	11-16-6	1.03E-05	1.000	0.092	0.978
2	3Layer	meanNorm	relu	5-12-16	0.021	0.995	0.092	0.978
3	3Layer	meanNorm	relu	10-3-9	0.041	0.991	0.097	0.976
4	3Layer	min-max	relu	14-17-14	0.007	0.998	0.099	0.976
5	3Layer	devide_max	tanh	11-6-19	0.000	1.000	0.102	0.975
6	3Layer	non	relu	10-20-13	0.050	0.989	0.103	0.975
7	3Layer	min-max	relu	13-11-16	0.029	0.993	0.106	0.974
8	3Layer	meanNorm	relu	14-14-12	0.045	0.990	0.112	0.973
9	3Layer	non	relu	11-9-20	0.013	0.997	0.112	0.973
10	3Layer	min-max	relu	14-20-8	0.020	0.995	0.112	0.973
11	3Layer	min-max	relu	17-13-18	0.096	0.978	0.112	0.973
12	3Layer	min-max	relu	13-14-20	0.036	0.992	0.114	0.972
13	3Layer	devide_max	relu	7-16-16	0.020	0.995	0.114	0.972
14	3Layer	non	relu	7-7-17	0.009	0.998	0.115	0.972
15	3Layer	min-max	tanh	6-1-19	0.025	0.994	0.115	0.972
16	3Layer	devide_max	relu	18-12-9	0.032	0.993	0.116	0.972
17	3Layer	devide_max	relu	14-15-6	0.023	0.995	0.116	0.972
18	3Layer	meanNorm	relu	13-15-9	0.089	0.980	0.116	0.972
19	3Layer	non	relu	10-18-19	0.019	0.996	0.117	0.972
20	3Layer	min-max	relu	5-11-16	0.075	0.983	0.117	0.972
21	3Layer	non	relu	9-9-8	0.040	0.991	0.118	0.971

Continued on next page

Table 4 (Continue): 150 best configurations with artificial neural networks

rank	type	normalisation	activation	layer-format	MSE(training set)	R2(training set)	MSE(test set)	R2(test set)
22	3Layer	min-max	relu	6-9-6	0.140	0.968	0.119	0.971
23	3Layer	devide_max	relu	7-16-17	0.019	0.996	0.120	0.971
24	3Layer	non	relu	12-4-14	0.049	0.989	0.121	0.971
25	3Layer	non	relu	10-8-20	0.031	0.993	0.121	0.971
26	3Layer	min-max	relu	10-15-19	0.056	0.987	0.121	0.971
27	3Layer	min-max	relu	15-19-16	0.078	0.982	0.122	0.970
28	3Layer	min-max	relu	5-19-2	0.046	0.990	0.122	0.970
29	3Layer	min-max	relu	19-11-6	0.022	0.995	0.122	0.970
30	3Layer	non	relu	16-12-10	0.031	0.993	0.122	0.970
31	3Layer	min-max	relu	18-8-8	0.091	0.979	0.123	0.970
32	3Layer	non	relu	15-12-16	0.004	0.999	0.123	0.970
33	3Layer	devide_max	relu	5-8-12	0.028	0.994	0.123	0.970
34	3Layer	non	relu	7-9-3	0.040	0.991	0.123	0.970
35	3Layer	non	relu	19-14-19	0.050	0.989	0.124	0.970
36	3Layer	min-max	relu	7-13-18	0.056	0.987	0.124	0.970
37	3Layer	min-max	tanh	8-7-5	0.001	1.000	0.124	0.970
38	3Layer	devide_max	relu	8-14-11	0.061	0.986	0.125	0.970
39	3Layer	min-max	relu	19-10-17	0.037	0.992	0.125	0.970
40	3Layer	devide_max	relu	19-17-8	0.030	0.993	0.125	0.970
41	3Layer	devide_max	relu	18-12-5	0.036	0.992	0.126	0.970
42	3Layer	min-max	relu	10-6-7	0.063	0.986	0.126	0.969

Continued on next page

Table 4 (Continue): 150 best configurations with artificial neural networks

rank	type	normalisation	activation	layer-format	MSE(training set)	R2(training set)	MSE(test set)	R2(test set)
43	3Layer	non	relu	4-15-16	0.045	0.990	0.126	0.969
44	3Layer	min-max	relu	19-8-6	0.085	0.981	0.126	0.969
45	3Layer	devide_max	relu	9-13-3	0.088	0.980	0.126	0.969
46	2Layer	meanNorm	relu	9-10	0.016	0.996	0.137	0.969
47	3Layer	devide_max	relu	17-5-6	0.045	0.990	0.127	0.969
48	3Layer	devide_max	relu	9-19-7	0.044	0.990	0.127	0.969
49	3Layer	min-max	relu	14-16-5	0.091	0.979	0.127	0.969
50	3Layer	min-max	relu	10-7-19	0.092	0.979	0.128	0.969
51	3Layer	min-max	relu	6-20-6	0.039	0.991	0.128	0.969
52	2Layer	min-max	relu	17-5	0.113	0.974	0.143	0.965
53	2Layer	non	relu	9-19	0.067	0.985	0.143	0.965
54	2Layer	non	relu	14-16	0.038	0.991	0.144	0.965
55	2Layer	non	relu	10-7	0.067	0.985	0.146	0.965
56	2Layer	min-max	relu	6-20	0.059	0.987	0.148	0.964
57	2Layer	meanNorm	relu	15-6	0.117	0.973	0.149	0.964
58	2Layer	non	relu	9-11	0.058	0.987	0.150	0.963
59	2Layer	min-max	relu	10-13	0.025	0.994	0.154	0.962
60	2Layer	devide_max	relu	15-10	0.192	0.956	0.155	0.962
61	2Layer	non	relu	19-7	0.011	0.998	0.156	0.962
62	2Layer	non	relu	12-17	0.038	0.991	0.157	0.962
63	2Layer	meanNorm	relu	7-19	0.209	0.953	0.157	0.962

Continued on next page

Table 4 (Continue): 150 best configurations with artificial neural networks

rank	type	normalisation	activation	layer-format	MSE(training set)	R2(training set)	MSE(test set)	R2(test set)
64	2Layer	non	relu	3-15	0.051	0.988	0.157	0.962
65	2Layer	min-max	relu	13-7	0.125	0.971	0.159	0.961
66	2Layer	meanNorm	tanh	19-1	0.000	1.000	0.159	0.961
67	2Layer	non	relu	16-9	0.031	0.993	0.160	0.961
68	2Layer	devide_max	relu	17-10	0.124	0.972	0.160	0.961
69	2Layer	meanNorm	relu	10-14	0.143	0.967	0.162	0.961
70	2Layer	min-max	relu	16-8	0.031	0.993	0.163	0.960
71	2Layer	devide_max	relu	19-18	0.008	0.998	0.163	0.960
72	2Layer	non	relu	17-18	0.027	0.994	0.165	0.960
73	1Layer	devide_max	relu	11	0.048	0.989	0.156	0.970
74	2Layer	devide_max	relu	17-13	0.017	0.996	0.166	0.960
75	2Layer	min-max	relu	12-12	0.038	0.991	0.167	0.959
76	2Layer	min-max	relu	12-14	0.043	0.990	0.167	0.959
77	2Layer	min-max	relu	17-15	0.134	0.970	0.167	0.959
78	2Layer	min-max	tanh	15-20	0.153	0.965	0.168	0.959
79	2Layer	non	relu	13-11	0.050	0.989	0.169	0.959
80	2Layer	devide_max	tanh	15-18	0.163	0.963	0.169	0.959
81	2Layer	devide_max	relu	17-20	0.147	0.967	0.170	0.959
82	2Layer	meanNorm	relu	11-16	0.109	0.975	0.170	0.959
83	2Layer	devide_max	relu	14-17	0.035	0.992	0.171	0.958
84	2Layer	devide_max	tanh	6-16	0.206	0.953	0.172	0.958

Continued on next page

Table 4 (Continue): 150 best configurations with artificial neural networks

rank	type	normalisation	activation	layer-format	MSE(training set)	R2(training set)	MSE(test set)	R2(test set)
85	2Layer	min-max	relu	13-18	0.043	0.990	0.172	0.958
86	2Layer	devide_max	relu	18-6	0.020	0.995	0.173	0.958
87	2Layer	devide_max	relu	4-11	0.051	0.988	0.173	0.958
88	2Layer	devide_max	relu	8-4	0.071	0.984	0.174	0.958
89	2Layer	min-max	tanh	20-20	0.165	0.962	0.174	0.958
90	2Layer	min-max	relu	10-5	0.094	0.979	0.175	0.958
91	2Layer	meanNorm	relu	7-13	0.181	0.959	0.175	0.957
92	2Layer	min-max	tanh	10-16	0.190	0.957	0.177	0.957
93	2Layer	min-max	relu	5-10	0.102	0.977	0.177	0.957
94	2Layer	non	tanh	10-20	0.183	0.958	0.177	0.957
95	2Layer	non	relu	16-6	0.009	0.998	0.177	0.957
96	2Layer	devide_max	relu	4-13	0.092	0.979	0.178	0.957
97	2Layer	min-max	tanh	19-14	0.163	0.963	0.179	0.957
98	2Layer	min-max	tanh	14-16	0.133	0.970	0.180	0.956
99	2Layer	min-max	relu	19-20	0.124	0.972	0.180	0.956
100	2Layer	devide_max	relu	12-4	0.095	0.978	0.180	0.956
101	2Layer	devide_max	relu	6-14	0.048	0.989	0.180	0.956
102	1Layer	devide_max	relu	11	0.064	0.986	0.185	0.955
103	1Layer	devide_max	relu	16	0.063	0.986	0.186	0.955
104	1Layer	meanNorm	relu	11	0.020	0.996	0.200	0.951
105	1Layer	devide_max	relu	7	0.176	0.960	0.211	0.949

Continued on next page

Table 4 (Continue): 150 best configurations with artificial neural networks

rank	type	normalisation	activation	layer-format	MSE(training set)	R2(training set)	MSE(test set)	R2(test set)
106	1Layer	devide_max	relu	9	0.057	0.987	0.219	0.947
107	1Layer	min-max	relu	9	0.097	0.978	0.224	0.946
108	1Layer	min-max	relu	15	0.022	0.995	0.240	0.942
109	1Layer	devide_max	relu	13	0.059	0.987	0.245	0.940
110	1Layer	min-max	relu	16	0.140	0.968	0.252	0.939
111	1Layer	non	relu	12	0.057	0.987	0.259	0.937
112	1Layer	devide_max	relu	10	0.083	0.981	0.267	0.935
113	1Layer	non	relu	15	0.060	0.986	0.268	0.935
114	1Layer	min-max	relu	8	0.170	0.961	0.268	0.935
115	1Layer	devide_max	relu	18	0.053	0.988	0.269	0.935
116	1Layer	min-max	relu	20	0.049	0.989	0.270	0.934
117	1Layer	non	relu	11	0.170	0.961	0.278	0.932
118	1Layer	min-max	relu	18	0.044	0.990	0.279	0.932
119	1Layer	non	relu	20	0.043	0.990	0.282	0.931
120	1Layer	meanNorm	tanh	19	0.191	0.957	0.297	0.928
121	1Layer	meanNorm	tanh	13	0.260	0.941	0.297	0.928
122	1Layer	non	relu	18	0.026	0.994	0.302	0.927
123	1Layer	meanNorm	relu	18	0.229	0.948	0.304	0.926
124	1Layer	min-max	relu	11	0.187	0.957	0.304	0.926
125	1Layer	meanNorm	relu	12	0.029	0.993	0.305	0.926
126	1Layer	meanNorm	relu	17	0.232	0.947	0.306	0.926

Continued on next page

Table 4 (Continue): 150 best configurations with artificial neural networks

rank	type	normalisation	activation	layer-format	MSE(training set)	R2(training set)	MSE(test set)	R2(test set)
127	1Layer	non	tanh	8	0.011	0.998	0.314	0.924
128	1Layer	meanNorm	relu	18	0.002	1.000	0.314	0.924
129	1Layer	devide_max	relu	20	0.041	0.991	0.314	0.924
130	1Layer	devide_max	relu	12	0.053	0.988	0.323	0.921
131	1Layer	min-max	relu	10	0.062	0.986	0.327	0.921
132	1Layer	non	relu	19	0.037	0.992	0.330	0.920
133	1Layer	min-max	tanh	4	0.100	0.977	0.337	0.918
134	1Layer	min-max	relu	17	0.029	0.993	0.338	0.918
135	1Layer	devide_max	relu	17	0.027	0.994	0.341	0.917
136	1Layer	devide_max	tanh	15	0.366	0.917	0.343	0.917
137	1Layer	non	relu	17	0.049	0.989	0.344	0.917
138	1Layer	non	relu	10	0.230	0.948	0.346	0.916
139	1Layer	meanNorm	relu	3	0.397	0.910	0.348	0.916
140	1Layer	non	tanh	16	0.360	0.918	0.351	0.915
141	1Layer	meanNorm	relu	16	0.002	1.000	0.351	0.915
142	1Layer	devide_max	tanh	6	0.048	0.989	0.352	0.915
143	1Layer	min-max	tanh	12	0.387	0.912	0.359	0.913
144	1Layer	meanNorm	tanh	5	0.104	0.976	0.360	0.913
145	1Layer	meanNorm	relu	15	0.304	0.931	0.361	0.912
146	1Layer	devide_max	relu	6	0.264	0.940	0.362	0.912
147	1Layer	non	relu	9	0.074	0.983	0.364	0.912

Continued on next page

Table 4 (Continue): 150 best configurations with artificial neural networks

rank	type	normalisation	activation	layer-format	MSE(training set)	R2(training set)	MSE(test set)	R2(test set)
148	1Layer	meanNorm	relu	20	0.294	0.933	0.365	0.911
149	1Layer	meanNorm	relu	16	0.386	0.912	0.365	0.911
150	1Layer	min-max	tanh	15	0.407	0.907	0.366	0.911

Appendix B

Herein, the proposed N-layer models with $N = \{1, 2, 3\}$ to predict the flexural strength of a vegetal composite are reported. The models are sorted individually based on the highest R^2 value in the test set. From the 3 proposed models, a set of 50 best configurations of each model (a total of 150 best configurations) are presented. As in results, the model with 3 layers, normalized with "devide-max" and "relu" as activation function has the most R^2 value in the test set.

Table 5: 150 best configurations with artificial neural networks to predict the flexural strength

rank	type	normalisation	activation	layer-format	MSE(training set)	R2(training set)	MSE(test set)	R2(test set)
1	3layer	devide_max	relu	19-2-20	0.017	0.961	0.025	0.928
2	3layer	non	relu	20-15-4	0.012	0.972	0.030	0.912
3	3layer	meanNorm	relu	11-19-11	0.000	1.000	0.030	0.912
4	3layer	meanNorm	relu	5-10-15	0.031	0.929	0.030	0.911
5	3layer	non	relu	20-3-5	0.018	0.958	0.031	0.910
6	3layer	non	tanh	4-3-5	0.003	0.993	0.032	0.906
7	3layer	Min_Max	relu	12-5-13	0.008	0.983	0.032	0.905
8	3layer	non	relu	9-5-2	0.028	0.936	0.032	0.905
9	3layer	non	relu	10-3-3	0.027	0.938	0.034	0.901
10	3layer	meanNorm	relu	10-14-12	0.000	0.999	0.034	0.901
11	3layer	Min_Max	tanh	10-16-4	0.000	0.999	0.034	0.901
12	3layer	non	relu	17-9-5	0.033	0.924	0.034	0.901
13	3layer	devide_max	relu	18-11-12	0.019	0.956	0.034	0.901
14	3layer	devide_max	relu	17-17-18	0.041	0.904	0.034	0.900
15	3layer	Min_Max	tanh	10-16-3	0.000	0.999	0.034	0.900
16	3layer	devide_max	relu	16-11-20	0.057	0.869	0.034	0.899
17	3layer	devide_max	relu	20-8-4	0.005	0.989	0.034	0.899
18	3layer	Min_Max	relu	15-11-4	0.094	0.783	0.034	0.899
19	3layer	devide_max	relu	14-4-5	0.021	0.952	0.034	0.899
20	3layer	devide_max	relu	11-15-19	0.055	0.874	0.034	0.899
21	3layer	non	relu	2-4-9	0.079	0.818	0.035	0.898

Continued on next page

Table 4 (Continue): 150 best configurations with artificial neural networks

rank	type	normalisation	activation	layer-format	MSE(training set)	R2(training set)	MSE(test set)	R2(test set)
22	3layer	devide_max	relu	5-15-1	0.028	0.936	0.035	0.898
23	3layer	meanNorm	relu	9-6-17	0.073	0.833	0.035	0.898
24	3layer	devide_max	relu	5-19-10	0.015	0.966	0.035	0.898
25	3layer	Min_Max	relu	7-20-3	0.026	0.939	0.035	0.898
26	3layer	devide_max	relu	4-18-14	0.087	0.798	0.035	0.896
27	3layer	non	relu	20-6-20	0.005	0.989	0.036	0.896
28	3layer	non	tanh	7-11-14	0.000	1.000	0.036	0.896
29	3layer	non	relu	8-19-20	0.125	0.712	0.036	0.895
30	3layer	meanNorm	relu	15-8-12	0.000	0.999	0.036	0.895
31	3layer	devide_max	tanh	13-4-13	0.000	0.999	0.036	0.895
32	3layer	meanNorm	relu	10-12-2	0.048	0.889	0.036	0.895
33	3layer	Min_Max	relu	4-15-14	0.021	0.952	0.036	0.895
34	3layer	non	relu	19-6-2	0.027	0.938	0.036	0.894
35	3layer	non	relu	7-3-3	0.032	0.927	0.036	0.894
36	3layer	Min_Max	relu	9-10-13	0.047	0.892	0.036	0.893
37	3layer	devide_max	relu	19-15-18	0.000	0.999	0.037	0.893
38	3layer	devide_max	relu	13-13-18	0.022	0.948	0.037	0.892
39	3layer	non	relu	6-7-15	0.022	0.949	0.037	0.892
40	3layer	Min_Max	relu	13-2-3	0.026	0.940	0.037	0.892
41	3layer	devide_max	relu	20-3-5	0.015	0.965	0.037	0.892
42	3layer	Min_Max	relu	18-17-4	0.041	0.906	0.037	0.891

Continued on next page

Table 4 (Continue): 150 best configurations with artificial neural networks

rank	type	normalisation	activation	layer-format	MSE(training set)	R2(training set)	MSE(test set)	R2(test set)
43	3layer	non	relu	9-19-10	0.067	0.846	0.037	0.891
44	3layer	non	relu	7-2-3	0.076	0.825	0.037	0.891
45	2Layer	Min_Max	relu	10-10	0.022	0.949	0.037	0.891
46	3layer	devide_max	relu	6-6-20	0.025	0.943	0.037	0.890
47	3layer	Min_Max	relu	12-18-17	0.018	0.958	0.037	0.890
48	3layer	non	relu	6-10-2	0.031	0.929	0.037	0.890
49	3layer	devide_max	relu	17-19-18	0.041	0.905	0.037	0.890
50	3layer	Min_Max	relu	16-15-14	0.062	0.857	0.037	0.890
51	3layer	meanNorm	relu	9-4-11	0.017	0.962	0.037	0.890
52	2Layer	Min_Max	relu	16-12	0.030	0.932	0.042	0.876
53	2Layer	meanNorm	tanh	12-3	0.000	1.000	0.043	0.875
54	2Layer	devide_max	relu	8-17	0.036	0.918	0.043	0.875
55	2Layer	Min_Max	relu	11-3	0.041	0.907	0.043	0.874
56	2Layer	Min_Max	relu	9-3	0.031	0.928	0.043	0.874
57	2Layer	devide_max	relu	6-8	0.039	0.909	0.046	0.865
58	2Layer	non	relu	19-16	0.101	0.767	0.046	0.865
59	2Layer	non	relu	19-20	0.069	0.840	0.046	0.864
60	2Layer	devide_max	relu	19-3	0.027	0.937	0.047	0.863
61	2Layer	Min_Max	relu	17-20	0.109	0.748	0.048	0.858
62	2Layer	meanNorm	relu	3-4	0.043	0.902	0.048	0.858
63	2Layer	Min_Max	relu	12-11	0.014	0.968	0.049	0.857

Continued on next page

Table 4 (Continue): 150 best configurations with artificial neural networks

rank	type	normalisation	activation	layer-format	MSE(training set)	R2(training set)	MSE(test set)	R2(test set)
64	2Layer	Min_Max	relu	4-19	0.034	0.921	0.049	0.857
65	2Layer	Min_Max	relu	16-3	0.077	0.824	0.049	0.855
66	2Layer	devide_max	relu	12-18	0.103	0.762	0.050	0.854
67	2Layer	Min_Max	relu	7-6	0.031	0.929	0.050	0.853
68	2Layer	non	relu	16-14	0.055	0.873	0.051	0.851
69	2Layer	meanNorm	relu	10-20	0.040	0.908	0.051	0.851
70	2Layer	devide_max	relu	4-10	0.043	0.902	0.051	0.850
71	2Layer	devide_max	relu	9-18	0.046	0.895	0.051	0.850
72	2Layer	meanNorm	tanh	6-13	0.000	0.999	0.052	0.848
73	2Layer	non	relu	18-15	0.070	0.839	0.052	0.847
74	2Layer	non	relu	8-14	0.018	0.957	0.053	0.845
75	2Layer	devide_max	relu	12-20	0.110	0.746	0.053	0.845
76	2Layer	meanNorm	relu	11-17	0.068	0.844	0.053	0.845
77	2Layer	devide_max	relu	20-6	0.014	0.967	0.053	0.845
78	2Layer	non	relu	11-3	0.124	0.714	0.053	0.843
79	2Layer	devide_max	relu	8-8	0.028	0.935	0.054	0.842
80	2Layer	non	relu	20-16	0.063	0.854	0.054	0.841
81	2Layer	non	tanh	5-19	0.001	0.998	0.054	0.841
82	2Layer	devide_max	relu	10-9	0.116	0.732	0.054	0.841
83	2Layer	devide_max	relu	12-19	0.115	0.736	0.054	0.841
84	2Layer	devide_max	relu	9-12	0.036	0.916	0.055	0.839

Continued on next page

Table 4 (Continue): 150 best configurations with artificial neural networks

rank	type	normalisation	activation	layer-format	MSE(training set)	R2(training set)	MSE(test set)	R2(test set)
85	2Layer	non	relu	16-13	0.061	0.860	0.055	0.839
86	2Layer	non	relu	11-5	0.032	0.925	0.055	0.839
87	2Layer	devide_max	relu	9-15	0.065	0.851	0.055	0.838
88	2Layer	meanNorm	relu	2-12	0.046	0.894	0.055	0.837
89	2Layer	non	relu	16-16	0.049	0.887	0.056	0.837
90	2Layer	non	relu	7-14	0.135	0.688	0.056	0.836
91	2Layer	Min_Max	relu	7-15	0.028	0.935	0.056	0.836
92	2Layer	non	relu	6-6	0.069	0.840	0.056	0.835
93	2Layer	meanNorm	relu	7-17	0.001	0.998	0.056	0.835
94	2Layer	meanNorm	relu	11-11	0.070	0.840	0.056	0.835
95	2Layer	Min_Max	relu	9-9	0.131	0.699	0.056	0.834
96	2Layer	devide_max	relu	12-4	0.027	0.938	0.057	0.834
97	2Layer	devide_max	relu	4-4	0.032	0.927	0.057	0.833
98	2Layer	non	relu	19-17	0.055	0.874	0.057	0.832
99	2Layer	non	relu	17-10	0.122	0.720	0.057	0.832
100	2Layer	Min_Max	relu	12-10	0.112	0.743	0.057	0.832
101	1Layer	Min_Max	tanh	10	0.003	0.992	0.058	0.859
102	1Layer	devide_max	relu	20	0.132	0.696	0.058	0.829
103	1Layer	devide_max	relu	9	0.021	0.952	0.063	0.814
104	1Layer	non	relu	7	0.035	0.919	0.064	0.813
105	1Layer	meanNorm	relu	15	0.129	0.702	0.065	0.809

Continued on next page

Table 4 (Continue): 150 best configurations with artificial neural networks

rank	type	normalisation	activation	layer-format	MSE(training set)	R2(training set)	MSE(test set)	R2(test set)
106	1Layer	devide_max	relu	15	0.029	0.933	0.066	0.807
107	1Layer	Min_Max	relu	13	0.092	0.789	0.067	0.804
108	1Layer	devide_max	relu	12	0.028	0.936	0.067	0.804
109	1Layer	non	relu	13	0.033	0.924	0.070	0.795
110	1Layer	devide_max	relu	11	0.140	0.678	0.070	0.794
111	1Layer	devide_max	relu	14	0.029	0.934	0.071	0.792
112	1Layer	devide_max	relu	16	0.132	0.696	0.073	0.785
113	1Layer	devide_max	tanh	6	0.015	0.966	0.074	0.783
114	1Layer	devide_max	relu	9	0.111	0.744	0.074	0.782
115	1Layer	devide_max	relu	17	0.029	0.934	0.077	0.775
116	1Layer	devide_max	relu	13	0.018	0.959	0.078	0.771
117	1Layer	Min_Max	relu	20	0.015	0.966	0.079	0.768
118	1Layer	Min_Max	relu	18	0.023	0.946	0.079	0.767
119	1Layer	meanNorm	relu	14	0.089	0.796	0.080	0.764
120	1Layer	meanNorm	tanh	18	0.103	0.762	0.082	0.761
121	1Layer	Min_Max	tanh	4	0.023	0.948	0.082	0.758
122	1Layer	non	relu	19	0.010	0.976	0.083	0.755
123	1Layer	Min_Max	relu	14	0.032	0.925	0.084	0.754
124	1Layer	Min_Max	relu	6	0.043	0.901	0.084	0.753
125	1Layer	devide_max	tanh	2	0.045	0.896	0.085	0.750
126	1Layer	meanNorm	relu	10	0.145	0.665	0.086	0.746

Continued on next page

Table 4 (Continue): 150 best configurations with artificial neural networks

rank	type	normalisation	activation	layer-format	MSE(training set)	R2(training set)	MSE(test set)	R2(test set)
127	1Layer	Min_Max	tanh	20	0.001	0.998	0.087	0.744
128	1Layer	Min_Max	tanh	18	0.000	0.999	0.087	0.743
129	1Layer	meanNorm	relu	5	0.027	0.937	0.088	0.743
130	1Layer	meanNorm	relu	4	0.143	0.671	0.089	0.739
131	1Layer	devide_max	tanh	5	0.020	0.953	0.089	0.738
132	1Layer	Min_Max	relu	16	0.155	0.644	0.089	0.737
133	1Layer	devide_max	relu	10	0.031	0.929	0.089	0.737
134	1Layer	Min_Max	relu	19	0.015	0.965	0.090	0.736
135	1Layer	non	relu	10	0.029	0.934	0.091	0.734
136	1Layer	non	relu	16	0.025	0.943	0.091	0.732
137	1Layer	Min_Max	relu	15	0.030	0.931	0.091	0.732
138	1Layer	devide_max	relu	20	0.022	0.950	0.091	0.731
139	1Layer	non	relu	6	0.162	0.627	0.091	0.731
140	1Layer	non	relu	20	0.170	0.609	0.092	0.731
141	1Layer	meanNorm	relu	18	0.106	0.755	0.092	0.730
142	1Layer	non	relu	14	0.100	0.769	0.092	0.729
143	1Layer	non	tanh	8	0.004	0.990	0.092	0.729
144	1Layer	meanNorm	tanh	16	0.141	0.675	0.092	0.729
145	1Layer	meanNorm	tanh	11	0.122	0.719	0.093	0.726
146	1Layer	Min_Max	relu	15	0.183	0.577	0.093	0.726
147	1Layer	meanNorm	relu	13	0.002	0.996	0.094	0.725

Continued on next page

Table 4 (Continue): 150 best configurations with artificial neural networks

rank	type	normalisation	activation	layer-format	MSE(training set)	R2(training set)	MSE(test set)	R2(test set)
148	1Layer	Min_Max	relu	19	0.158	0.637	0.094	0.724
149	1Layer	non	relu	12	0.149	0.658	0.094	0.723
150	1Layer	devide_max	relu	6	0.062	0.858	0.094	0.723

Prédiction, réseau de neurones et optimisation: Applications aux domaines des agro-matériaux et de la télécommunication

Résumé *(en français)*

Cette thèse a pour but de traiter la complexité rencontrée pour optimiser le composite végétal et assurer la durabilité des centres d'approvisionnement, de production et de commercialisation. Le premier objectif est atteint en optimisant les caractéristiques du matériau composite à l'aide de l'intelligence artificielle et est détaillé en première partie. Le second objectif est exposé dans la deuxième partie de cette étude. Il s'agit de la fusion des sites de la chaîne d'approvisionnement en utilisant la méthode du K-clustering. Différentes méthodes de solution d'optimisation sont proposées. L'approche transversale appliquée permettant la mise en œuvre de plusieurs compétences est présente dans l'unité de recherche EPROAD. Les méthodes proposées sont issues du domaine de l'intelligence artificielle, de la modélisation discrète issue des mathématiques appliquées, de l'analyse de sensibilité, et du domaine du génie des procédés pour le développement de méthodes coopératives intelligentes.

Mots-clés Méthodes numériques, Matériaux agrosourcés, Prédiction, Intelligence artificielle, Optimisation combinatoire.

Prediction, neural network and optimization: Applications in the fields of agro-materials and telecommunication

Abstract *(In english)*

This thesis aims to deal with the complexity encountered to optimize the vegetal composite and to ensure the sustainability of the supply, production, and marketing sites. The former is achieved, in the first part, by optimizing the characteristics of the composite material using artificial intelligence methods. The latter is presented in the second part of this study. The proposed method merges the supply chain site(s) using the K-clustering problem. Different optimization solution methods are proposed. The applied transversal approach allowing the coupling of skills is presented within the research unit EPROAD. The proposed methods are from the field of artificial intelligence, combinatorial optimization, discrete modeling resulting from applied mathematics, sensitivity analysis, and the field of process engineering for the development of intelligent cooperative methods.

Keywords Numerical studies, Vegetal-composite, Prediction, Artificial intelligence, Combinatorial optimization.