UQAC
Université du Québec
à Chicoutimi

OUTIL D'AIDE À LA DÉCISION POUR L'ÉLABORATION DES REPAS DANS LES GARDERIES EN MILIEU FAMILIAL

PAR GEOFFREY GLANGINE

MÉMOIRE PRÉSENTÉ À L'UNIVERSITÉ DU QUÉBEC À CHICOUTIMI

COMME EXIGENCE PARTIELLE EN VUE DE L'OBTENTION DU GRADE

DE MAÎTRE ÈS SCIENCES EN INFORMATIQUE

QUÉBEC, CANADA

RÉSUMÉ

Les enfants ont besoin d'avoir un régime alimentaire bien équilibré pour demeurer en bonne santé. C'est pour cela que les garderies ont pour obligation de fournir aux enfants des repas variés et sains chaque semaine. Il est difficile pour une petite garderie d'élaborer ces menus santé tout en gardant une bonne rentabilité. Pour résoudre ce problème, il est décrit dans ce mémoire, un nouvel outil d'aide à la décision pour proposer des menus équilibrés pour une semaine complète basés sur les prix courants dans les épiceries locales. Afin de correspondre aux besoins des garderies, la solution à réaliser doit être simple à utiliser, rapide et proposer des repas adaptés pour des enfants. Il est aussi important que les repas soit variés afin de permettre aux enfant de découvrir de nouvelles saveurs.

L'outil développé est capable de prendre en compte les ingrédients que l'utilisateur (gérant de garderie en milieu familial) possède déjà et les recettes qui ne lui plaisent pas. Il va automatiquement récupérer les prix des ingrédients, chaque semaine, dans les épiceries locales et trouve ses recettes sur un site de recette en ligne. Cet outil utilise des méthodes modernes d'extractions de données sur le web pour générer une base de donnée toujours à jour et il fournit une liste de repas pour garderie en milieu familial avec des prix raisonnables. La méthode est automatisée à 100% et est utilisable avec des paramètres par défaut sans action requise par l'utilisateur. L'outil est développé avec des technologies web et implémente un algorithme d'optimisation basé sur la programmation en nombres entiers, et particulièrement sur les méthodes de séparation et évaluation. Il a été testé par 9 garderies différentes et celles-ci ont noté un léger gain financier et une légère économie de temps grâce à l'utilisation du logiciel. Il est présenté dans ce mémoire l'architecture de l'outil, le modèle algorithmique, l'implémentation et les résultats obtenus par cette approche.

Tout d'abord, une mise en situation du contexte et une légère introduction à la recherche opérationnelle est proposée afin de donner les bases nécessaires à la compréhension du reste du mémoire au lecteur. Ensuite, une revue de littérature est effectuée pour rendre compte du travail de recherche déjà effectué sur la génération automatique de menus ainsi qu'en programmation en nombres entiers. Puis, le modèle mathématique et les méthodes d'optimisation utilisées pour répondre à la problématique sont présentées. Pour continuer, il est décrit comment l'architecture et le développement de l'application ont été conçus afin de répondre au besoins exprimés par le contexte dans lequel se fixe ce problème. Finalement les résultats résultats fournis par l'algorithme sont présentés ainsi que les résultats récoltés par l'étude qualitative auprès des garderies.

ABSTRACT

Children need to have a well-balanced diet to stay healthy. Therefore, daycare services have the obligation to provide varied and healthy menus each week. It is difficult for small daycare services to elaborate these healthy menus while having to balance their budget. To address this issue, we propose, in this paper, a new decision-making tool to create balanced menus for a week based on the prices of the different grocery products in the vicinity. The tool is available online via a simple and ergonomic web interface and all menus proposed are varied each week to let children discover new tastes.

This tool is also able to take into account the ingredients that the person already has. The tool crawl automatically, each week, the ingredient prices in the vicinity and it finds all the recipes on an online website. This tool uses modern web mining methods to generate an always up-to-date database and gives to daycare services a list of menus with realistic prices. The process is automated at 100% and default parameters are set in advance for computer-unfriendly users. The tool is developed as a web platform implementing an optimization algorithm based on integer programming and more precisely Branch-and-Bound. It has been preliminarily tested by nine daycare services which reported little money and time savings. In this master's thesis the architecture of the tool, the algorithmic model, the implementation and the preliminary results are presented.

At first, an introduction to operational research is proposed to let the reader get the theorical basis to understand this master's thesis. Then, the state of the art of automated meal planning generation and integer programming is made. Next, the mathematical formulation and method used for solving the problem are explained. After, the software architecture and development are presented. Finally, the results obtained by the algorithm and the preliminary study with daycare services are analyzed.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES ABRÉVIATIONS

API	Application Programming Interface
BIP	Binary Integer Program
CBC	COIN-OR Branch-and-Cut
CLP	COIN's native Linear Programming Solver
CMPL	COIN-OR Mathematical language
CoDIT	Conference on Control, Decision and Information Technologies
CPE	Centre de la Petite Enfance
CRUD	Create Read Update Delete
HTML	Hypertext Markup Language
IP	Integer Program
JSON	Javascript Object Notation
LIP	Linear Integer Program
LP	Linear Program
MIP	Mixed Integer Program
MVC	Model View Controller
NER	Named Entity Recognition
NP	Non-deterministic Polynomial-time
REST	Representational State Tranfert
RO	Recherche Opérationnelle
RPC	Remote Procedure Call
SOAP	Simple Object Access Protocol
URI	Uniform Resource Identifier
UX	User Experience
XML	eXtensible Markup Language

DÉDICACE

À ma très chère conjointe Morgane,

Ton encouragement et ton accompagnement tout au long de la rédaction de ce mémoire m'ont permis de trouver la motivation nécessaire les jours où je n'en avais pas. Merci d'être toujours à mes côtés depuis si longtemps. Je te le promets, un jour je terminerai mes études !

REMERCIEMENTS

Je tiens à remercier toutes les personnes qui m'ont aidé lors de la rédaction de ce mémoire.

Je voudrais dans un premier temps remercier particulièrement ma directrice de recherche Sara Séguin pour le temps précieux qu'elle a passé à relire et commenter chaque partie de ce mémoire tout au long de sa rédaction ainsi que pour son expertise en optimisation et les connaissances qu'elle m'a transmises tout au long de ma maîtrise.

Je remercie également tous mes co-directeurs de recherche Kévin Bouchard, Sébastien Gaboury et Bruno Bouchard pour leurs conseils et leur soutien tout au long de mes recherches.

Tous mes collègues du LIARA avec lesquels je vais certainement rester ami pour toutes les prochaines années et avec lesquels j'ai eu beaucoup de fun !

Mes parents pour leurs encouragements tout au long de mes études en France comme au Québec.

AVANT-PROPOS

Ce mémoire est le résultat d'une année complète de travaux de recherche effectuée au sein du laboratoire LIARA (Laboratoire d'Intelligence Ambiante pour la Reconnaissance d'Activité) à l'Université du Québec à Chicoutimi. Ayant suivi un cours de recherche opérationnelle à l'Université de Technologie de Belfort-Montbéliard en France qui m'a passionné, j'ai eu envie d'approfondir mes connaissances sur ce sujet. Lorsque Kévin Bouchard m'a proposé de travailler sur ce sujet de recherche intégrant de la recherche opérationnelle, j'ai tout de suite été motivé par le sujet. Cet algorithme d'aide à la décision pour l'élaboration des repas est très motivant, car il peut améliorer le quotidien de beaucoup de personnes et faire évoluer les habitudes alimentaires des personnes dès leur plus jeune âge. L'alimentation est l'une des plus grandes préoccupations de la société actuelle, et développer des algorithmes permettant à la population de découvrir de nouvelles recettes en réduisant les coûts associés à la nourriture est peut-être la clé pour pousser la population à prendre de bonnes habitudes alimentaires. C'est donc avec toutes ces idées et motivations en tête que j'ai démarré ces recherches.

CHAPITRE I

INTRODUCTION

Les habitudes alimentaires sont un élément très important dans le développement et la croissance d'un enfant. Au Québec, la fréquentation scolaire commence à l'âge de quatre ans (L'enfant doit avoir 4 ans avant le 30 septembre de l'année scolaire, sinon il commencera l'école l'année d'après). Entre zéro et quatre ans, 37% des enfants fréquentent une garderie en milieu familial composé de 6 à 9 enfants (Gouvernement du Québec, 2015). Il y a deux types de garderies : Les Centres de la Petite Enfance (CPE) et les garderies en milieu familial. Les CPE sont des centres de grande taille bénéficiant d'aides et de ressources de fournisseurs leur permettant d'obtenir des repas et des collations santé subventionnées par le gouvernement. Dans les garderies en milieu familial, c'est aux responsables des garderies de choisir des repas appropriés. Certaines garderies font appel à des nutritionnistes pour sélectionner des recettes et ainsi réaliser des repas équilibrés. D'autres font appel à des traiteurs qui leur fournissent des repas pour toute la semaine. Le reste des garderies font les repas et les planifient eux mêmes. La solution d'employer un nutritionniste ou un traiteur engendre des frais supplémentaires à la garderie et la préparation individuelle des repas consomme beaucoup de temps, tant au niveau de la planification que de la préparation. C'est dans ce contexte que ce mémoire présente un outil basé sur des algorithmes d'optimisation (Griva *et al.*, 2009) et d'intelligence artificielle (Russell *et al.*, 2010) permettant de trouver des recettes de repas automatiquement sur Internet et de trouver la meilleure combinaison de repas pour une semaine afin de réduire les coûts hebdomadaires des garderies en milieu familial.

L'outil va chercher automatiquement les promotions hebdomadaires sur les sites des épiceries locales. Le projet vise à réduire les coûts associés à la planification et à l'achat des repas pour les garderies en milieu familial. De plus, il serait facilement adaptable à des résidences de personnes âgées ou même des familles voulant avoir une alimentation variée sans avoir à perdre trop de temps à cuisiner ou planifier des repas. Ces travaux de recherche vont être présentés à la conférence CoDIT'19 (Conference on Control, Decision and Information Technologies). L'article a été accepté et sera publié après la conférence.

1.1 DÉFINITION DU PROBLÈME

Le problème abordé dans ce mémoire consiste à élaborer un menu hebdomadaire pour une garderie en milieu familial, en déterminant un nombre spécifique de recettes. Pour cette expérimentation, le nombre de recettes a été fixé à cinq, car il faut un repas du midi pour tous les jours ouvrables de la semaine. L'application programmée doit être modulable en vue de l'adapter facilement pour d'autres secteurs tels que les familles ou les maisons de retraite. En conséquence, ce nombre de recettes ne doit pas être fixe. En effet, si l'algorithme doit être utilisé dans un autre endroit qu'une garderie en milieu familial, le nombre de repas doit pouvoir changer sans avoir à écrire un nouveau modèle mathématique (par exemple il faudrait 14 repas pour une maison de retraite). Les repas ne doivent pas être trop similaires et être équilibrés. En effet, il n'est pas tolérable, par exemple, de proposer des repas à base de boeuf pour tous les jours de la semaine.

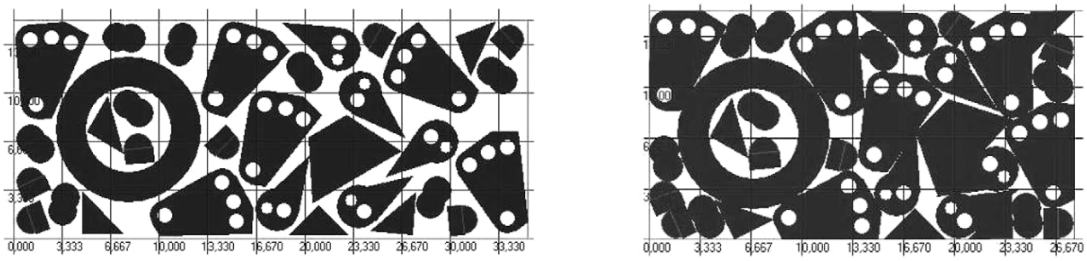
1.2 CONCEPTS DE BASE

Ce mémoire décrit la résolution d'un problème d'optimisation en nombres entiers (Wolsey, 1998). L'optimisation en nombres entiers étant un domaine de la recherche opérationnelle (Griva *et al.*, 2009), il est alors important de bien comprendre les concepts de base en optimisation avant de pouvoir travailler sur un problème d'optimisation en nombres entiers. En effet, ces problèmes sont souvent résolus avec des méthodes spécifiques utilisant certaines spécificités des algorithmes de résolution de problèmes linéaires classiques (Griva *et al.*, 2009). Pour que le lecteur comprenne correctement les algorithmes utilisés dans cette contribution, il est important de le familiariser avec le concept de recherche opérationnelle, et de formulation mathématique de problème d'optimisation. Ces concepts sont présentés dans cette section par le biais d'un exemple de problème.

1.2.1 DÉFINITION D'UN PROBLÈME DE RECHERCHE OPÉRATIONNELLE

La recherche opérationnelle (Griva *et al.*, 2009) est un domaine regroupant des méthodes et des algorithmes qui permettent de prendre des décisions visant à minimiser un coût ou maximiser un bénéfice. Elle utilise des approches provenant des mathématiques telles que la modélisation mathématique et la théorie polyédrale pour trouver la solution au problème de minimisation ou de maximisation. En recherche opérationnelle, le but est d'arriver à trouver une solution optimale ou le plus proche possible de l'optimalité. Elle peut s'appliquer dans beaucoup de domaines (ex : ingénierie, marché de la bourse, planification d'horaires, etc.) et permet de prendre des décisions dans des situations trop

complexes à résoudre pour un humain, ou alors trop longues à résoudre. Par exemple, pour une entreprise travaillant sur la découpe de formes complexes dans des plaques de métal (Dowsland et Dowsland, 1992), il est difficile de trouver un placement optimal des formes afin de gaspiller le moins possible de métal, et donc de maximiser les profits. Il est possible de trouver la solution en essayant toutes les combinaisons de formes à découper sur les plaques de métal, mais la solution trouvée par un humain sera coûteuse en temps et pas forcément optimale. Il est possible de développer un modèle de recherche opérationnelle pour résoudre le problème plus rapidement et de manière optimale. L'article de Chernov *et al.* (2010) présente les différentes techniques pour résoudre ce type de problèmes. La Figure 1.1 montre la différence d'espace occupé avec un placement optimal et un placement aléatoire des pièces de machinerie à découper. Un humain arrive aussi à produire un résultat meilleur que l'aléatoire, mais ne trouve pas à chaque fois la solution optimale et prend beaucoup plus de temps que l'algorithme d'optimisation.



(a) Placement aléatoire (b) Placement optimisé

Figure 1.1 : Placement aléatoire versus Algorithme d'optimisation

Source : Chernov *et al.* (2010)

Un problème de recherche opérationnelle (Griva *et al.*, 2009), ou d'optimisation,

est représenté sous forme mathématique. Il est composé de plusieurs éléments qui sont :

Fonction objectif : La fonction objectif est une équation mathématique qui vise à être maximisée ou minimisée (c'est la définition même du problème). Il faut savoir que l'on peut passer facilement une maximisation en minimisation (et vice versa) en multipliant la fonction objectif par -1.

Variables de décisions : C'est les valeurs de ces variables que l'on cherche à trouver en résolvant le problème. Elles permettent de minimiser ou de maximiser un problème (et donc la fonction objectif).

Paramètres : Les paramètres sont les éléments connus (les données) du problème.

Contraintes : Les contraintes sont aussi des équations définissant l'ensemble des solutions admissibles.

Bornes : Les bornes définissent les limites minimales et maximales de chacune des variables. Elles peuvent aussi être considérées comme des contraintes.

Le problème suivant est un problème de maximisation :

$$\max_{x} \quad c^T x \tag{1.1}$$

$$\text{Sujet à} \quad Ax \leq b \tag{1.2}$$

$$x \geq 0 \tag{1.3}$$

Où A est une matrice $n \times m$, c^T est un vecteur de dimension n , b est un vecteur de dimension m et x est un vecteur composé de n variables de décisions.

Si certaines variables doivent prendre des valeurs entières le problème sera un problème mixte en nombres entiers (Wolsey, 1998)

EXEMPLE D'UN PROBLÈME CONNU EN OPTIMISATION

Le problème du voyageur de commerce (Cook, 2015) est un problème très connu en optimisation. En effet, il est très simple à comprendre et à formuler, cependant il est difficile à résoudre. C'est pour cela que de nombreux algorithmes se servent de ce problème pour tester leurs performances. Il consiste à minimiser le trajet d'un voyageur devant faire une boucle (la plus courte) entre plusieurs villes en commençant par une ville de départ et en terminant dans cette même ville en ayant visité toutes les autres villes.

Dans ce cas, il y a autant de variables de décisions que de chemins possibles entre les villes. Ce sont des variables binaires données par :

$$x_{ij} = \begin{cases} 1 \text{ Si le chemin entre les villes } i \in \mathscr{V} \text{ et } j \in \mathscr{V} \text{ est emprunté,} \\ 0 \text{ sinon} \end{cases}$$

Avec $\mathscr{V}$ l'ensemble des villes.

Les paramètres sont les distances c_{ij} entre les villes.

Le problème consiste à minimiser le coût de la distance totale de la tournée :

$$\min_{x} \quad \sum_{i=1}^{n} \sum_{j \neq i, j=1}^{n} c_{ij} \times x_{ij} \tag{1.4}$$

$$\text{Sujet à} \quad \sum_{i=1}^{n} \sum_{j \neq i, j=1}^{n} = n \tag{1.5}$$

$$x \in \{0,1\}^n \tag{1.6}$$

avec c_{ij} le coût du chemin pour aller de la ville i à la ville j ; $i, j \in \mathscr{V}$ l'ensemble des villes du problème ; n le nombre de villes.

Les bornes, ou dans ce cas, le domaine de définition des variables du problème sont définis par l'équation (1.6). Le problème de base, ne contient qu'une seule contrainte (1.5) et elle impose que chaque ville soit visitée une seule fois. Ce problème est de complexité NP-difficile. C'est-à-dire qu'il est au moins aussi difficile à résoudre que le problème le plus difficile de la classe NP. La classe de complexité P est la classe pour laquelle les problèmes sont décidés en temps polynomial par une machine déterministe. La classe de complexité NP contient les problèmes décidés dans un temps polynomial par une machine non déterministe. Le problème du voyageur de commerce est sujet à l'explosion combinatoire (Papadimitriou et Steiglitz, 1998; Schuster, 2000), c'est-à-dire que la difficulté augmente très vite à cause du fait que le nombre de solutions possibles au problème augmente exponentiellement en changeant de manière minime une contrainte, une borne, ou les paramètres. Pour le voyageur de commerce, l'explosion combinatoire se fait au niveau de l'augmentation du nombre de villes.

1.3 OBJECTIFS DE RECHERCHE

Le problème de génération automatique de menus possède de nombreuses applications concrètes, mais il n'existe pas, dans la littérature scientifique, d'application à ce problème automatisée à 100%. En effet, la plupart des travaux effectués sur ce sujet ne vont pas chercher les recettes, les ingrédients et les prix automatiquement sur Internet. Le premier objectif de ce mémoire est de combiner des algorithmes d'optimisation (Griva *et al.*, 2009) avec des algorithmes d'intelligence artificielle (Russell *et al.*, 2010)

afin de générer un menu hebdomadaire pour une garderie en milieu familial. Le second objectif de recherche est de savoir si un tel système peut permettre de venir en aide aux gérants de garderies et s'il permet d'améliorer la variété, le temps de planification et de confection des repas ainsi que le prix des repas tout en fournissant un menu équilibré.

1.4 ORGANISATION DU MÉMOIRE

Le chapitre 2 présente une revue de littérature des algorithmes utilisés pour résoudre des problèmes de programmation en nombres entiers et des travaux réalisés sur les différents types de problèmes de génération de menus. Il est aussi présenté un bref état de l'art des techniques de "text mining" pour la classification de chaînes de caractères. Le chapitre 3 décrit comment ce travail aborde le problème de génération automatique de menu en définissant le problème puis en décrivant le modèle mathématique proposé pour sa résolution. Le chapitre 4 présente les méthodes utilisées pour le développement de l'application pour les garderies en présentant l'architecture proposée pour le logiciel, les méthodes de prétraitement des données utilisées ainsi que la façon dont les ingrédients et les recettes sont récupérés automatiquement. Finalement, le chapitre 5 montre les résultats numériques obtenus par l'algorithme d'optimisation ainsi que les résultats fournis par une étude qualitative de notre logiciel auprès de garderies en milieu familial. Cette étude à reporté un léger gain de temps et d'argent de la part des garderies qui ont testé le produit pendant plusieurs semaines.

CHAPITRE II

REVUE DE LITTÉRATURE

Le second chapitre de ce mémoire a pour but d'explorer les principaux travaux scientifiques existants liés à la contribution présentée dans ce mémoire et d'expliquer les notions nécessaires à la bonne compréhension des objectifs de recherche de ce projet. Le premier concept présenté définit l'algorithme du simplexe (Dantzig *et al.*, 1955) pour la résolution d'un problème d'optimisation linéaire. Le second concept présente l'optimisation en nombres entiers (Wolsey, 1998) ainsi que les algorithmes utilisés dans ce mémoire pour résoudre ce type de problème. Par la suite, les méthodes d'optimisation pour la génération de menu dans la littérature scientifique sont présentées (Smith, 1959). Et finalement, une revue de concept de "text-mining" (Gupta et Lehal, 2009; Allahyari *et al.*, 2017) est effectuée.

2.1 ALGORITHME DU SIMPLEXE

L'algorithme du simplexe (Dantzig *et al.*, 1955) est un algorithme permettant de résoudre des problèmes d'optimisation linéaire. Dans ce mémoire, il est décrit la résolution d'un problème en nombres entiers, mais l'algorithme du simplexe reste une composante essentielle à la résolution de tels problèmes, car les algorithmes de "Branch-and-Bound" (Lawler et Wood, 1966) et de "Branch-and-Cut" (Padberg et Rinaldi, 1991) utilisent l'algorithme du simplexe pour résoudre les relaxations linéaires des problèmes et des sous-problèmes.

L'algorithme du simplexe consiste à minimiser ou maximiser une fonction linéaire composée de n variables réelles sur un ensemble de m contraintes admissibles. Tel que mentionné à la section 1.2.1, la fonction à optimiser est la fonction objectif et l'ensemble admissible est représenté par les bornes et les contraintes. L'ensemble admissible des solutions forme un polyèdre convexe si les contraintes sont linéaires. Le problème peut admettre une seule solution optimale ou une infinité de solutions, si une contrainte est égale à la fonction objectif. Si ce n'est pas le cas, c'est que le problème ne peut être résolu, advenant deux possibilités :

— Le problème ne possède pas de solutions, car il n'y a pas de domaine réalisable.

— Le problème est non borné (dans ce cas-là, une des arêtes du polyèdre a une longueur infinie).

Géométriquement, l'algorithme du simplexe consiste à se déplacer de point extrême en point extrême sur le polyèdre en longeant une des arêtes de façon à toujours réduire (si minimisation) la valeur de la fonction objectif.

Soit le problème suivant, à titre d'exemple :

$$\max_{x,y} \quad 3x+5y \tag{2.1}$$

$$\text{Sujet à :} \quad x+y \leq 50, \tag{2.2}$$

$$3x-y \leq 80, \tag{2.3}$$

$$x,y \geq 0. \tag{2.4}$$

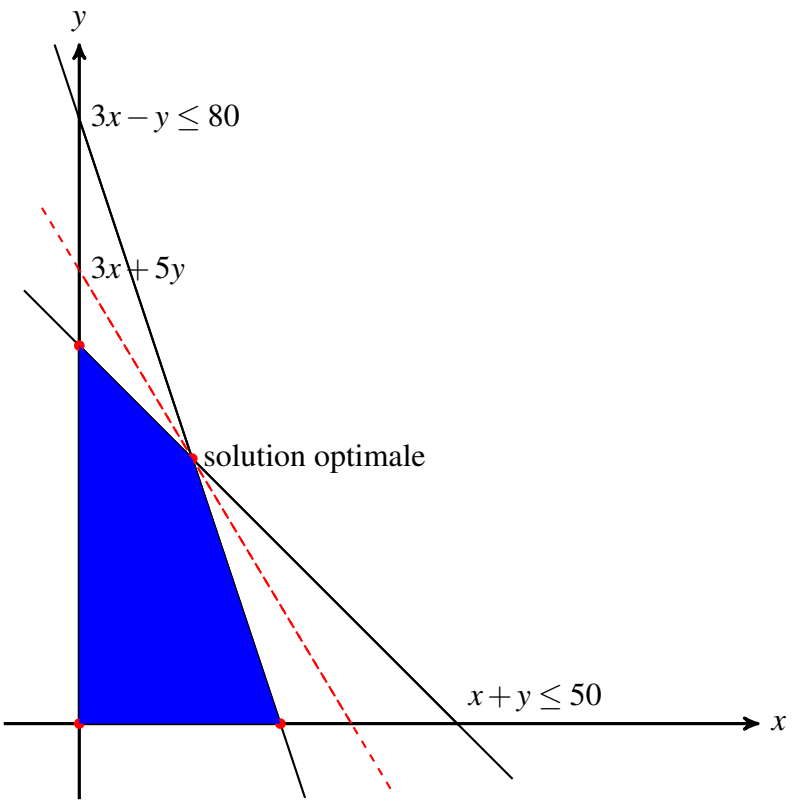


Figure 2.1 : Exemple géométrique du simplexe avec deux variables

La zone colorée en bleu de la Figure 2.1 est l'ensemble des solutions réalisables pour le problème décrit plus haut. Les points rouges sont les points extrêmes sur lesquels le simplexe itère pour trouver la solution optimale. La solution optimale est montrée sur la Figure 2.1. Pour la trouver géométriquement, il suffit de tracer la droite de la fonction objectif (ici en pointillés) et de positionner cette droite sur l'un des points extrêmes de façon à ce qu'aucun autre point extrême n'améliore cette fonction objectif (en déplaçant encore la droite de la fonction objectif).

Plus il y a de contraintes, plus le nombre de sommets augmente, c'est pour cela que l'algorithme du simplexe ne se contente pas d'énumérer tous les sommets du polyèdre et qu'il n'en calcule qu'une partie. Si sur l'un des sommets on obtient une preuve

d'optimalité, alors l'algorithme s'arrête et retourne la solution.

FONCTIONNEMENT DÉTAILLÉ

Dans un premier temps, un problème linéaire doit être écrit en forme standard pour être résolu par le simplexe. Cela signifie que toutes les variables du problème sont non négatives et que toutes les contraintes sont écrites sous forme d'égalité. Pour transformer une inégalité en égalité, il suffit d'ajouter ou de soustraire une variable d'écart. En effet, si la contrainte doit être inférieure à une valeur alors une variable d'écart est ajoutée ;

ainsi :

$$2x_1 + 3x_2 \leq 3 \tag{2.5}$$

devient :

$$2x_1 + 3x_2 + e_1 = 3; e_1 \geq 0 \tag{2.6}$$

avec e_1 la variable d'écart.

Si la contrainte doit être supérieure à une valeur, alors une variable de surplus est soustraite ;

ainsi :

$$2x_1 + 3x_2 \geq 3 \tag{2.7}$$

devient :

$$2x_1 + 3x_2 - s_1 = 3; s_1 \geq 0 \tag{2.8}$$

avec s_1 la variable d'écart.

Une fois le problème sous forme standard, la résolution par le simplexe peut

commencer. Dans un premier temps, il faut trouver une solution admissible au problème. Pour trouver cette solution il suffit, pour un problème avec n variables et m contraintes où $n \geq m$, de poser $n - m$ variables à 0. Ainsi le système d'équations possède le même nombre d'équations et de variables. Toutes les variables initialisées à zéro sont appelées variables hors-base. Les autres variables sont les variables en base. Il faut donc résoudre le système d'équations pour trouver la solution de base. Une fois la première solution trouvée, le simplexe vérifie si cette solution est optimale ; si elle n'est pas optimale, alors il effectue un pivot. Pour effectuer le pivot, le simplexe choisit une variable hors base pour la passer en base et choisi une variable en base pour la passer hors base. Il choisit la variable en base qui limite le plus l'amélioration de la fonction objective (nommée variable sortante) et choisit la variable hors base qui a un coût marginal permettant d'améliorer le plus possible l'objectif (appelée variable entrante). Le coût marginal doit être négatif si la fonction objective est une minimisation et positif si c'est une maximisation. Si aucun des coûts marginaux n'est négatif pour une minimisation ou positif pour une maximisation, alors la solution actuelle est optimale.

```
   Données : Un problème P sous forme standard
   Une solution S au problème
   Résultat : meilleure_solution
1  tant que S n'est pas optimale faire
2      Choix de la variable Entrante ;
3      Choix de la variable Sortante ;
4      La variable entrante entre en base, et la variable sortante sort de la base (Pivot)
5  fin
```

Algorithme 2.1 : Algorithme du simplexe en pseudo-code

2.2 PROGRAMMATION EN NOMBRES ENTIERS

Un problème de programmation de nombres entiers (Wolsey, 1998) est un programme d'optimisation mathématique où certaines variables doivent prendre des valeurs entières. Les problèmes d'optimisation en nombres entiers sont classés dans la catégorie des problèmes NP-complets, c'est-à-dire que la complexité de résolution de ces problèmes est exponentielle. Leur temps de résolution augmente donc très vite en fonction de leur taille si l'on cherche seulement à énumérer toutes les solutions. C'est pour cela qu'il existe des algorithmes tels que le "Branch-and-Bound" qui évitent d'énumérer toutes les solutions possibles afin de trouver une solution optimale dans un temps plus convenable.

2.2.1 FORMULATION DES PROBLÈMES DE PROGRAMMATION EN NOMBRES ENTIERS

Supposons que nous avons un problème linéaire :

$$\max_{x} \quad c^T x \tag{2.9}$$

$$\text{Sujet à :} \quad Ax \leq b, \tag{2.10}$$

$$x \geq 0. \tag{2.11}$$

où A est une matrice $n \times m$, c^T est un vecteur de dimension n , b est un vecteur de dimension m et x est un vecteur de dimension n composé de variables de décisions. Si certaines variables doivent prendre des valeurs entières, le problème sera un problème mixte en nombres entiers (Wolsey, 1998) :

$$\max_{x,y} \quad c^T x + h^T y \tag{2.12}$$

$$\text{Sujet à :} \quad Ax + Gy \leq b, \tag{2.13}$$

$$x \geq 0, \tag{2.14}$$

$$y \geq 0 \text{ et } y \in \mathbb{N}. \tag{2.15}$$

où A est une matrice $n \times m$, G est une matrice $n \times m$, h est un vecteur de taille p et y est un vecteur de taille p et contenant seulement des variables entières.

Si toutes les variables sont entières, alors c'est un problème en nombres entiers Wolsey (1998) :

$$\max_{x} \quad c^T x \tag{2.16}$$

$$\text{Sujet à :} \quad Ax \leq b, \tag{2.17}$$

$$x \geq 0 \text{ et } x \in \mathbb{N}. \tag{2.18}$$

Finalement, si toutes les variables ne peuvent prendre que les valeurs 0 et 1 alors c'est un problème en nombres entiers binaires (BIP) :

$$\max_{x} \quad c^T x \tag{2.19}$$

$$\text{Sujet à :} \quad Ax \leq b, \tag{2.20}$$

$$x \in \{0,1\}^n. \tag{2.21}$$

2.2.2 ALGORITHME DU BRANCH AND BOUND

L'algorithme du "Branch-and-Bound" (Lawler et Wood, 1966) est un algorithme de résolution de problèmes d'optimisation en nombres entiers. Il est basé sur le concept

de diviser pour mieux régner. L'algorithme divise le problème principal en plusieurs sous-problèmes plus simples à résoudre de manière à ce que la solution optimale d'un sous-problème devienne une borne supérieure ou inférieure au problème. L'algorithme fonctionne grâce à une énumération des solutions sous forme d'arbre.

Si l'algorithme se contente d'énumérer les solutions, la complexité de calcul est exponentielle. Par exemple, pour un BIP avec n variables de décision, il y a 2^n feuilles dans l'arbre d'énumération, car les valeurs que chaque variable peut prendre sont $0, 1$ (Figure 2.2). Dans la plupart des problèmes, il est impossible d'énumérer la totalité des solutions à cause du nombre de variables et des valeurs qu'elles peuvent prendre. C'est pour cela que "Branch-and-Bound" n'énumère pas toutes les solutions et résout les relaxations linéaires (permet à la variable de prendre des valeurs linéaires pour pouvoir résoudre le problème avec un algorithme d'optimisation classique) des variables entières en utilisant l'algorithme du simplexe et construit un arbre au fur et à mesure.

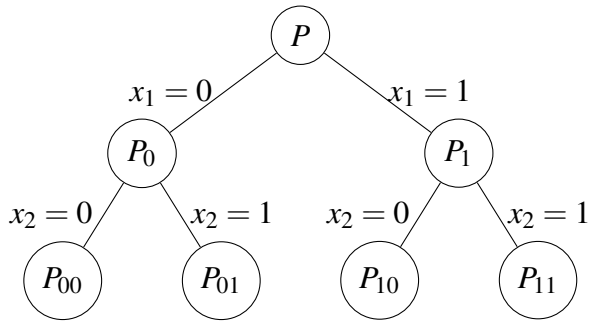


Figure 2.2 : Arbre d'énumération d'un problème binaire avec deux variables

ÉNUMÉRATION IMPLICITE

Comme mentionné dans la section précédente, la complexité d'une énumération est exponentielle. Il est donc impossible, pour un gros problème, de procéder à une énumération complète des solutions dans un temps raisonnable. Il faut donc éviter d'énumérer les solutions s'il est possible de trouver un moyen de savoir si un sous-problème contient des solutions moins bonnes que son voisin. Pour réaliser cela, il suffit d'utiliser les résultats de la relaxation linéaire (Griva *et al.*, 2009) du problème. En d'autres termes, le problème en nombres entiers est transformé en problème linéaire permettant aux variables de prendre des valeurs continues. Ainsi, l'algorithme du simplexe (Dantzig *et al.*, 1955) pourra être utilisé pour résoudre le problème à chaque nœud de l'arbre. En effet, si on utilise l'algorithme du simplexe sur le problème en nombres entiers, il est possible de trouver une solution entière. Si c'est le cas, alors la solution au problème est trouvée. Dans le cas échéant, il faut créer un nouveau sous-problème et ajouter des contraintes pour éviter de trouver à nouveau cette solution non entière.

Ensuite, dans le cas d'une maximisation, la valeur de la solution optimale de la relaxation linéaire du problème sera donc une borne supérieure sur la valeur de la solution optimale du problème principal. Si la solution optimale de la relaxation linéaire est une solution admissible du problème principal, alors la valeur de la solution de la relaxation linéaire devient une borne inférieure au problème. Si la solution optimale de la relaxation linéaire est une solution admissible du problème principal, alors elle est aussi la solution optimale du problème principal. Ainsi, il est possible d'éliminer de l'exploration les sous-problèmes avec des bornes inférieures ou supérieures moins

bonnes que celles déjà trouvées auparavant. Si la solution de la relaxation linéaire du problème n'est pas admissible, alors on va créer des sous-problèmes en ajoutant des contraintes pour éliminer les valeurs interdites à la variable dont la valeur n'est pas entière. Par exemple, si la solution de la relaxation linéaire donne une valeur ($x_1 = n$), alors l'algorithme va créer deux sous problèmes en ajoutant les contraintes suivantes (une dans chacun des deux sous problèmes) :

$$x_1 \leq \lfloor n \rfloor \tag{2.22}$$

$$x_1 \geq \lceil n \rceil \tag{2.23}$$

Dans l'algorithme 2.2 et dans la Figure 2.3 cela correspond aux sous-problèmes P1 et P2. La Figure 2.3 présente seulement un exemple numérique avec deux variables x_1 et x_2. Pour simplifier, la fonction objective n'est pas spécifiée.

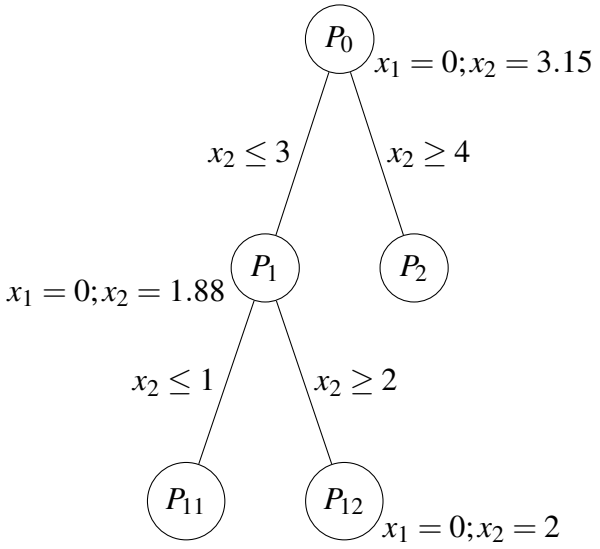


Figure 2.3 : Arbre "Branch-and-Bound" partiel

Le choix de l'ordre des nœuds à explorer est important, il influe beaucoup sur le temps nécessaire afin de trouver la solution optimale. Il existe beaucoup de stratégies

différentes afin de choisir quels nœuds parcourir en premier (Neveu *et al.*, 2016; Narendra et Fukunaga, 1977; Fukunaga et Narendra, 1975; Gupta et Ravindran, 1985). Il faut cependant privilégier l'exploration des fils d'un nœud afin d'utiliser correctement les propriétés du simplexe.

```
   Données : Un problème P
   borne_min := −∞
   n = nombre de variables
   Résultat : meilleure_solution
1  Fonction BnB (Problème P, borneMin)
2      solution := résoudre la relaxation linéaire de P ;
3      borne_max := valeur de l'objectif de la relaxation linéaire ;
4      si P n'admet pas de solution réalisable alors
5          On arrête l'exploration car le sous-problème n'admet pas de solution;
6          retourner null;
7      fin
8      si borne_max ≤ borne_min alors
9          On arrête l'exploration car les solutions de ce sous problèmes sont
             forcément moins bonne que celles trouvées ailleurs;
10         retourner null;
11     fin
12     si x_i ∈ ℕ, ∀i = 1,2,...,n alors
13         borne_min := borne_max;
14         On arrête l'exploration car une solution optimale a été trouvée;
15         retourner solution;
16     fin
17     P1 est P2 sont des sous problèmes de P ;
18     Ajouter la contrainte x_i ≥ ⌈x_i⌉ dans P1;
19     Ajouter la contrainte x_i ≥ ⌊x_i⌋ dans P2;
20     retourner maximum(BnB(P1, borneMin), BnB(P2, borneMin));
21 fin
```

Algorithme 2.2 : Algorithme en pseudo-code du "Branch-and-Bound" récursif dans le cas d'une maximisation

2.2.3 ALGORITHME DES PLANS COUPANTS

Dans certains cas, la description d'un problème d'optimisation peut être trop grande pour être représentée entièrement sur la mémoire d'un ordinateur ou d'un solveur d'optimisation. Pour éviter ce problème, il est possible de résoudre le problème petit à petit en résolvant des relaxations du problème d'optimisation principal. C'est-à-dire que le problème va être résolu avec certaines contraintes en moins. En procédant ainsi, un problème plus simple est résolu, et l'ensemble des solutions réalisables du sous-problème qui vient d'être résolu contient l'ensemble des solutions réalisables du problème principal.

L'algorithme des plans coupants consiste à résoudre un problème relaxé et ajouter itérativement des contraintes violées par la solution courante, jusqu'à ce qu'il n'y en ait plus à ajouter. Ces contraintes peuvent être des contraintes de valeurs entières (comme pour le "Branch-and-Bound") ou alors des contraintes définies dans la définition du problème. Une contrainte est dite violée si la valeur d'au moins une des variables du problème ne respecte pas cette contrainte. Ces contraintes sont appelées des plans coupants. Cet algorithme ne nécessite pas une liste complète des contraintes à ajouter, mais seulement d'une méthode permettant de générer efficacement des inégalités valides. Les coupes ne sont pas spécifiques au problème résolu et peuvent être utilisées sur tous les problèmes. Il y a plusieurs types de coupes connues comme des coupes de Gomory (Gomory, 1958) ou les méthodes du "lift and project" (Grötschel *et al.*, 1984). Ces coupes sont souvent peu efficaces (elle ne permettent pas d'éliminer assez de solutions tout en gardant les ensembles intéressants) pendant la résolution de problèmes difficiles

et c'est pour cela qu'il est quand même important d'étudier les coupes plus spécifiques aux problèmes à résoudre.

2.2.4 ALGORITHME DU "BRANCH-AND-CUT"

L'algorithme du "Branch-and-Cut" (Padberg et Rinaldi, 1991) est très similaire au "Branch-and-Bound" (Lawler et Wood, 1966). En effet le "Branch-and-Cut" est une modification du "Branch-and-Bound". Au lieu de résoudre systématiquement une relaxation linéaire par le simplexe, l'algorithme va utiliser la méthode des plans coupants (Wolsey, 1998) pour trouver de nouvelles contraintes au problème afin de réduire l'espace de définition du problème en conservant la meilleure solution entière, mais en faisant en sorte que la solution linéaire trouvée se retrouve interdite. "Branch-and-Cut" a été utilisé pour la première fois dans le problème de "linear ordering" de Balas *et al.* (1996) mais le nom de "Branch-and-Cut" a été utilisé pour la première fois par Padberg et Rinaldi (1987). Quelques années plus tard, Padberg et Rinaldi (1991) font un véritable état de l'art des techniques de "Branch-and-Cut" en exposant de nouvelles stratégies de séparation et des nouvelles procédures telles que la génération de colonnes.

2.2.5 UTILISATION DANS LA LITTÉRATURE

Les algorithmes de "Branch-and-Bound" et de "Branch-and-Cut" sont réputés pour être très efficaces pour trouver des solutions à des problèmes de programmation en nombres entiers. C'est pour cela que la littérature regorge d'articles démontrant l'utilité de ces algorithmes pour résoudre tout type de problèmes avec des variables entières. Par exemple, le problème de tournée de facteur présenté par Ávila *et al.* (2016) obtient de

très bons résultats avec l'algorithme de "Branch-and-Cut".

Cependant, parmi toutes les recherches qui ont été effectuées sur les algorithmes de "Branch-and-Bound" et de "Branch-and-Cut", il semble que les problèmes de planification de menus n'aient pas été suffisamment abordés. Ces algorithmes sont souvent utilisés à des fins de planification d'horaires de travail. Le problème de planification de menus qui est décrit dans ce mémoire peut s'apparenter à un problème de planification d'horaire. C'est pour cela que le choix d'un algorithme de "Branch-and-Cut" est pertinent pour un tel problème.

2.3 GÉNÉRATION AUTOMATIQUE DE MENUS

Dans le domaine de la génération automatique de menus (Smith, 1959), beaucoup d'approches ont été développées. Il est possible d'observer dans la littérature scientifique deux grands groupes de modélisation du problème de génération automatique des menus. En effet, le problème est souvent abordé de manière multiobjective (Deb et Deb, 2014). Ce qui permet une meilleure souplesse vis-à-vis de la modularité des résultats et des préférences variables des utilisateurs finaux. Cependant, il nécessite que l'utilisateur règle des paramètres.

D'autres ont vu le problème comme un problème d'optimisation avec un seul objectif. L'un des premiers à voir le problème de cette façon est Balintfy (1964). Il avait, à l'époque, permis de choisir entre plusieurs fonctions objectives. Une permettant de minimiser le coût de la planification des repas, et l'autre permettant de maximiser une des composantes préférées par les utilisateurs (ex : nutrition) prises en compte lors du

calcul. L'algorithme peut aussi être transformé en problème multi-objectifs pour prendre plusieurs fonctions objectives en compte. Le problème, c'est que l'algorithme prend en argument une liste de recettes de base pour générer un menu. Chaque recette est de type différent (entrée, plat principal, dessert...) et l'algorithme ne fait que choisir parmi cette liste pour générer des menus complets. En plus, la liste doit être générée à la main. L'approche qui est proposée dans ce mémoire permet de trouver automatiquement une liste de recettes sur Internet et une liste d'ingrédients sur des sites d'épicerie en ligne pour générer des menus et obtenir des prix en temps réels.

Un travail intéressant de Chavez-Bosquez *et al.* (2014) présente une architecture basée sur une approche d'optimisation hybride pour le problème de planification de menus. Leur système est composé de deux modules différents. Le premier est un modèle de programmation mathématique et le second est basé sur le modèle de "belief merging", qui est un modèle peu connu permettant de combiner les conseils nutritionnels de professionnels de la santé et les préférences des utilisateurs. Leur approche est très complète, mais elle se concentre beaucoup trop sur les apports nutritionnels et ne prend pas en compte le temps de préparation ainsi que le prix des menus générés.

Seljak (2009) propose une approche basée sur un algorithme évolutionnaire pour trouver une panification de repas. Elle modélise le problème comme un problème de sac à dos à dimensions multiples (Chu et Beasley, 1998). En effet, le problème de sac à dos à dimensions multiples consiste à trouver la meilleure combinaison possible de nourriture dans un sac avec des volumes fixés à certaines valeurs. Les valeurs sont définies pour correspondre au problème de planification de menus (qualité des aliments,

coût et quelques autres paramètres). Cependant, utiliser un algorithme évolutionnaire ne permet pas de trouver la solution optimale, mais une valeur approchée de cette solution. De plus, cette recherche n'est pas basée sur des recettes, mais des ingrédients individuels pour parvenir à trouver un menu équilibré. Il n'y a donc pas de cohérence entre les ingrédients choisis pour un menu. Par exemple dans les résultats présentés, il est possible d'observer que le premier repas est composé de : pain, de poivre grillé, d'huile d'olive, de bleuets et de sucre.

La recherche effectuée par Hsiao et Chang (2010) a pour but de créer une application mobile permettant de générer des menus équilibrés remplissant tous les besoins nutritionnels d'un individu. Le problème est basé sur une approche multiobjective. Les objectifs qu'ils ont choisi de maximiser sont :

— améliorer les valeurs nutritionnelles

— réduire le prix des aliments

— privilégier les préférences des utilisateurs vis-à-vis du goût des aliments

Cette approche est similaire à celle présentée dans ce mémoire, car elle prends en compte le prix, les aliments et les préférences des utilisateurs. De plus, elle se présente sous forme d'application mobile. Cependant, la recherche décrite dans ce mémoire est basée sur un modèle d'optimisation mono-objectif permettant à l'utilisateur final de recevoir sa planification hebdomadaire de repas sans avoir d'action à effectuer, qu'il soit sur tablette, sur ordinateur ou sur téléphone intelligent. De plus dans ce travail, les repas proposés sont choisi dans les restaurants locaux, le prix des repas est donc supérieur au prix d'une planification de repas générée à partir de menus à faire sois-même et il n'est

pas envisageable d'emmener des enfants de garderie au restaurant tous les jours de la semaine.

Les travaux de Hooker et Barnes (2015) montrent une métaheuristique multiobjective permettant de générer une planification de menus pour une semaine en fonction de plusieurs critères, tels que la valeur nutritive, le temps pour faire la recette, le coût de la recette, la note de la recette. La liste des recettes qu'ils utilisent se trouve sur le site "USDA what's cooking ?" (United States Departement of Agriculture, 2015). Grâce à leur modélisation multiobjective, une fois que l'algorithme a terminé de calculer les recettes, une interface utilisateur est proposée et les utilisateurs peuvent choisir avec des pourcentages quels critères ils préfèrent maximiser en se déplaçant sur le front de Pareto (Kim et de Weck, 2005). L'algorithme leur fournit un résultat en fonction de leurs préférences. Cependant, dans leur approche, il y a certains critères de sélection qui semblent peu pertinents, ou difficiles à prendre en compte correctement. Par exemple sur "USDA whats cooking ?" les prix ne sont pas numériques, ils sont représentés sur une échelle de 1 à 5. C'est pareil pour les notes données par les utilisateurs. Par contre le fait que "USDA whats cooking ?" fournisse une liste exacte de toutes les valeurs nutritionnelles pour chaque recette apporte une véritable valeur ajoutée à leur recherche. En effet, grâce à ces données, leur algorithme prend en compte le facteur nutrition de manière très stricte. L'approche proposée dans ce mémoire ne prend pas en compte les valeurs nutritionnelles, mais elle propose un modèle d'estimation des prix en allant chercher les prix de chacun des ingrédients sur les sites de plusieurs épiceries en ligne. Cependant, elle propose un module permettant à l'utilisateur de toujours avoir des recettes variées et de types différents sur plusieurs semaines.

L'approche proposée dans ce mémoire est basée sur un modèle d'optimisation avec un seul objectif ce qui permet d'avoir la meilleure solution en tout temps sur le critère du prix total de la semaine et de ne pas avoir de décisions à prendre en étant face à un front de Pareto (Kim et de Weck, 2005). Cependant, nous effectuons un prétraitement des données et une modification des contraintes afin de personnaliser les résultats en fonction des préférences de chaque utilisateur individuellement. Comme dans les travaux de Hsiao et Chang (2010) notre application propose une interface utilisateur permettant de choisir selon ses préférences et elle trouve les prix des aliments de manière très fiable directement sur les sites web des épiceries locales.

2.4 CLASSIFICATION D'INGRÉDIENTS

L'un des buts principaux de cette recherche est de trouver automatiquement des recettes via Internet et de les lier avec des ingrédients trouvés aussi via des sites d'épiceries locales afin d'avoir les prix en temps réel et de pouvoir optimiser les menus en fonction des réductions hebdomadaires. Pour parvenir à ce résultat, il est indispensable de trouver un moyen de lier les ingrédients trouvés sur les sites des épiceries aux ingrédients trouvés sur les sites de recettes. Il faut définir les types d'ingrédients et ainsi créer des groupes de classes communes aux recettes et aux produits trouvés sur les sites d'épiceries. Ces méthodes relèvent du domaine du "text-mining" (Gupta et Lehal, 2009; Allahyari *et al.*, 2017) et plus précisément de la "named entity recognition" (David Nadeau et Satoshi Sekine, 2007; Marrero *et al.*, 2013). Le "text-mining" est un domaine s'intéressant à la récupération d'informations d'un texte de manière automatisée. Contrairement aux domaines habituels de "machine learning" (Kelleher *et al.*, 2015) et

de "data mining" (Tan *et al.*, 2019), le "text-mining" permet de traiter des données non structurées. Le "text-mining" est le processus permettant de récupérer des informations de bonne qualité au sein d'un texte. Une information de bonne qualité est souvent extraite grâce à des schémas de mots ou un apprentissage statistique de ces schémas. Comme décrit sur la Figure 2.4 le "text-mining" transforme une entrée de texte en une information structurée enregistrée dans une base de données ou un fichier structuré tel que JSON ou XML afin de passer à la dernière phase permettant d'évaluer l'information en utilisant des algorithmes spécifiques pour trouver des corrélations, catégoriser des parties de texte ou encore résumer des documents. Une de ces méthodes est nommée "Named entity recognition". Elle permet plus précisément de localiser ou de classer des entités dans un texte non structuré en catégories définies par avance. Elle peut catégoriser des mots ou des groupes de mots en classes telle que des dates, des prix, des quantités, des noms de lieux ou d'autres catégories définies à l'avance. Elle procède par étiquetage des objets classables en ajoutant des balises ou en liant les éléments avec des classes en base de données. Les systèmes d'étiquetage sont souvent alimentés par des bases de données (dans ce mémoire, des listes d'ingrédients ont été utilisées). Mais dans les cas où les classifications doivent être par catégories plus vastes, les algorithmes de classification par "Named Entity Recognition" utilisent des règles de grammaire associées à des modèles statistiques. Les méthodes de "Named Entity Recognition" nécessitent très souvent de paramétrer et créer certaines bases de données manuellement. C'est dans cette optique que l'outil présenté dans la section 4.2.1 et en Figure 4.7 a été développé.

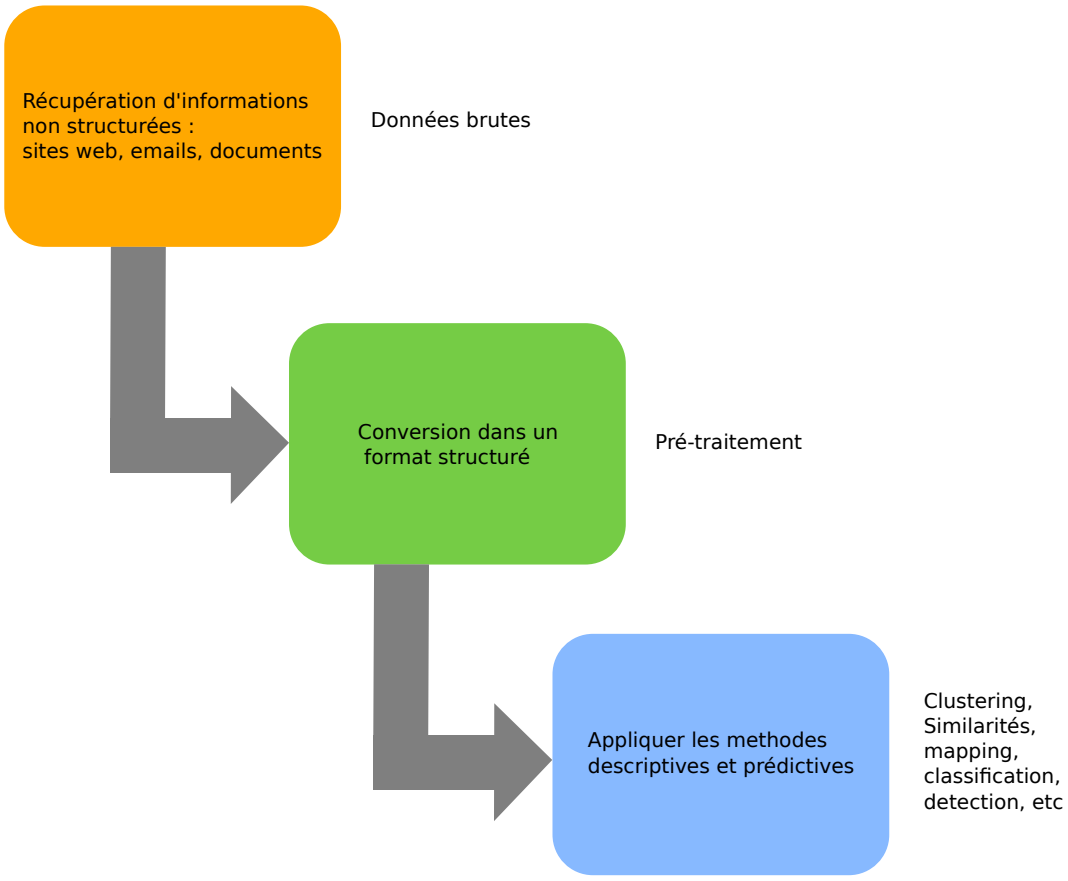


Figure 2.4 : Processus global du "text-mining"

CHAPITRE III

OPTIMISATION

Les concepts de programmation en nombres entiers définis dans le chapitre précédent sont utiles afin de définir mathématiquement le problème de génération automatique de menus. La première partie de ce chapitre est dédiée à l'étude des contraintes pour répondre aux besoins du problème. La seconde partie de ce chapitre présente le modèle mathématique et les méthodes de résolution utilisées.

3.1 ÉQUILIBRE ET VARIÉTÉ DES RECETTES

Pour avoir des repas équilibrés, un prétraitement manuel des recettes est effectué après les avoir récupérées automatiquement sur Internet afin de supprimer les recettes n'ayant pas un bon équilibre alimentaire. Les recettes sont aussi classées en fonction de 5 types pour permettre une diversité des recettes au fil de la semaine. Les cinq types de recettes sont :

— Boeuf et viande rouge

— Volaille

— Poisson et fruits de mer

— Végétarien et recettes à base d'oeufs

— Porc

Par défaut, le module d'optimisation fournit une recette de chaque type pour la semaine, mais il est possible pour l'utilisateur de choisir combien de recettes il veut pour

chaque type. L'utilisateur peut aussi choisir de ne pas avoir de préférence de variété de recettes et donc d'obtenir les cinq recettes permettant d'avoir le prix le plus bas de la semaine. L'objectif à minimiser est donc le prix sans contrainte de variété de type de recette.

3.2 MODÈLE PROPOSÉ

Cette section présente l'entièreté du modèle permettant de générer la sélection de recettes. En premier lieu, les ensembles utilisés par le modèle seront définis. Ensuite, les paramètres nécessaires à la résolution du problème et les variables de décisions sont présentés. Finalement, la fonction objective et les contraintes sont définies.

3.2.1 ENSEMBLES

$\mathscr{I} = \{1, 2, ..., 2000\}$: L'ensemble des ingrédients

$\mathscr{R} = \{1, 2, ..., 200\}$: L'ensemble des recettes

$\mathscr{T} = \{\text{boeuf}, \text{porc}, \text{volaille}, \text{poisson}, \text{végétarien}\}$: L'ensemble des types de recettes

3.2.2 PARAMÈTRES

Les paramètres connus du problème sont les types (boeuf, volaille...) de chaque recette ainsi que les ingrédients contenus dans les recettes.

— $P_r^i = \begin{cases} 1 \text{ si l'ingrédient } i \in \mathscr{I} \text{ est dans la recette } r \in \mathscr{R} \\ 0 \text{ sinon} \end{cases}$

— $cost(i)$: le prix de l'ingrédient $i \in \mathscr{I}$

— $Z_r^t = \begin{cases} 1 \text{ si la recette } r \in \mathscr{R} \text{ contient le type } t \in \mathscr{T} \\ 0 \text{ sinon} \end{cases}$

3.2.3 VARIABLES

Le modèle proposé utilise des variables de décision binaires permettant de déterminer l'affectation des recettes au menu de la semaine.

— $x_r = \begin{cases} 1 \text{ si la recette r } \in \mathscr{R} \text{ est choisie} \\ 0 \text{ sinon} \end{cases}$

— $Z_r^t = \begin{cases} 1 \text{ si la recette r } \in \mathscr{R} \text{ contient le type } t \in \mathscr{T} \\ 0 \text{ sinon} \end{cases}$

3.2.4 FONCTION OBJECTIF

La fonction objectif minimise la somme des coûts des ingrédients qui sont utilisés dans les recettes choisies.

$$\min_x \sum_{i \in \mathscr{I}} \sum_{r \in \mathscr{R}} P_r^i \times cost(i) \times x_r \tag{3.1}$$

3.2.5 CONTRAINTES

La somme des recettes choisies est égale au nombre de repas à préparer. Avec n le nombre de recettes à choisir pour la semaine.

$$\sum_{r \in \mathscr{R}} x_r = n, \tag{3.2}$$

La liste des recette contient au moins une recette de chaque type.

$$\sum_{t \in \mathscr{T}} Z_r^t \geq 1, \ \forall r \in \mathscr{R}, \tag{3.3}$$

Et l'ensemble de définition des variables.

$$x_r \in \{0,1\}, \ \forall r \in \mathscr{R} \tag{3.4}$$

Le modèle mathématique du problème de génération de planification de menus est donc mis en place avec des ensembles, des paramètres, des variables de décision, une fonction objectif ainsi que des contraintes. Il est maintenant possible de présenter sa résolution.

3.3 RÉSOLUTION DU MODÈLE

Pour ce projet de recherche, le solveur CBC (COIN-OR Branch-and-Cut)(Forrest *et al.*, 2018) de COIN-OR (Computational Infrastructure for Operations Research) est utilisé. Le modèle est résolu via un algorithme de "branch-and-cut".

3.3.1 OUTILS UTILISÉS

COIN-OR (Wächter et Biegler, 2006) est une infrastructure contenant une suite de logiciels mathématiques "open source". La vocation du projet COIN-OR est de fournir un grand nombre de logiciels mathématiques, et plus particulièrement pour la recherche opérationnelle, de qualité et de manière totalement ouverte. Cette contribution utilise la librairie de "branch-and-cut" (CBC) de COIN-OR qui est une implémentation de l'algorithme de "branch-and-cut" standard. CBC étant une librairie de "branch-and-cut",

elle nécessite un solveur de programmation linéaire pour fonctionner. Il est recommandé d'utiliser CLP (COIN's native Linear Programming Solver). Un logiciel, nommé Coliop, permet d'écrire un modèle mathématique dans un langage spécifique (CMPL "COIN-OR Mathematical Language") via une interface graphique et ensuite d'interpréter ce langage pour l'envoyer au solveur et résoudre le problème. Coliop a donc beaucoup été utilisé pour réaliser les tests préliminaires avec le modèle mathématique.

Cependant, il n'est pas possible de passer par une interface graphique pour résoudre le problème d'optimisation, car il faut que ce processus soit transparent pour l'utilisateur final. C'est pour cela que l'infrastructure présentée dans cette contribution utilise le serveur d'optimisation de CMPLserver (Steglich, 2016) de COIN-OR. CMPL-Server fonctionne aussi avec CMPL comme Coliop mais il n'a pas besoin d'interface graphique pour fonctionner. Il reçoit un document XML par un socket réseau et grâce aux informations contenues dans le XML reçu, il régénère le fichier CMPL de base, le résout et renvoie les résultats au format XML via un socket réseau. CMPLServer dispose d'une interface en Java (JCMPL) et d'une interface en Python (pyCMPL). Ces interfaces permettent de définir les variables, les contraintes et les ensembles du modèle mathématique, de générer le fichier XML à envoyer et d'interpréter le contenu de la réponse du serveur. Grâce à JCMPL ou pyCMPL et un CMPLserver, il est donc possible de résoudre le problème en java ou en python via les algorithmes de branch-and-cut de CBC.

3.3.2 PARAMÉTRAGE DE CBC

CBC (Forrest *et al.*, 2018) est paramétrable sur certains points tels que les méthodes d'exploration et de comparaison des solutions. En effet, l'ordre dans lequel les nœuds sont explorés peut grandement influencer la performance d'un algorithme de "branch-and-cut". CBC donne un contrôle complet de l'ordre des nœuds à explorer tout au long du fonctionnement de l'algorithme. Il est même possible de changer dynamiquement la stratégie de sélection en fonction de l'avancement de la recherche. Puisque le logiciel est "open source", il est même possible de développer une nouvelle stratégie d'exploration si les stratégies implémentées de base ne conviennent pas pour la résolution d'un problème spécifique. Le modèle défini dans ce mémoire est un modèle plutôt simple à résoudre pour un solveur de ce type, il n'a donc pas été nécessaire de développer une nouvelle stratégie d'exploration. Les stratégies d'exploration disponibles de base dans CBC sont :

Recherche en profondeur : La recherche en profondeur explore toujours en premier le nœud le plus profond dans l'arbre. Cela permet de garder une taille d'arbre relativement petite, mais elle peut prendre beaucoup de temps pour trouver une solution.

Recherche en largeur : La recherche en largeur explore tous les nœuds d'une profondeur avant de passer au prochain niveau de profondeur.

Comparaison d'objectif : Cette stratégie choisit toujours le nœud ayant la meilleure valeur de fonction objectif possible. Avec cette méthode, la première solution est trouvée rapidement et il est facile d'élaguer beaucoup de branches à l'arbre. Cette

méthode s'apparente à la recherche gloutonne (Curtis, 2003).

Comparaison d'estimation : Cette stratégie utilise des heuristiques pour prédire la valeur de la fonction objectif dans les nœuds plus en profondeur. Elle choisit donc le nœud avec la prédiction la meilleure. Une heuristique souvent utilisée est la recherche en coût uniforme (Felner, 2011).

La stratégie de recherche utilisée pour ce projet est implémentée de manière à faire une recherche en profondeur jusqu'à trouver une première solution. Ensuite, elle cherche avec une estimation sur les autres nœuds s'il est possible de trouver une meilleure solution. Si un grand nombre de nœuds sont explorés et que plusieurs des solutions ont été trouvées, alors une recherche en largeur sera utilisée. Si avec la recherche en largeur, la taille de l'arbre devient trop importante, l'algorithme retourne sur une recherche en profondeur. Cette stratégie est implémentée de base dans le code source de CBC et est recommandée dans la documentation officielle.

Grâce à la modélisation mathématique définie dans ce chapitre et à l'utilisation du solveur "open source" CBC il est possible de résoudre le problème de génération de planification de menus avec des données réelles. Ces données sont les ingrédients et les recettes qui sont récupérées directement sur plusieurs sites Internet. La procédure de récupération des données et l'architecture logicielle globale du projet sont présentées dans le chapitre suivant.

CHAPITRE IV

DÉVELOPPEMENT DE L'APPLICATION

Ce chapitre présente la méthodologie et l'architecture proposées pour développer l'application web permettant aux garderies d'avoir une interface personnalisée pour la gestion de leurs recettes hebdomadaires. La première section décrit l'architecture globale du projet composée de différents modules ayant chacun une fonction spécifique. La seconde section présente comment les données de recettes et des ingrédients sont récupérés automatiquement sur Internet. Ensuite il est expliqué comment les données récupérées sur Internet peuvent être mises en forme pour correspondre au modèle d'optimisation défini dans le chapitre 3 et fonctionner avec CMPL. Finalement le développement de l'application web côté "front end" et côté "back end" sont présentés.

4.1 ARCHITECTURE DU LOGICIEL

De plus en plus d'utilisateurs consultent le web sur des appareils mobiles. Chaque année, la part des ordinateurs par rapport aux périphériques mobiles ne fait que diminuer (Murphy et Meeker, 2011). Cela est dû à l'amélioration des performances des téléphones cellulaires ainsi que des améliorations constantes des réseaux sans fil. Ainsi selon l'étude réalisée par We are social et Hootsuite (2018), 72 % des habitants du Canada possèdent un téléphone intelligent et 82% des utilisateurs ont un accès à internet. C'est pour cela que l'application présentée dans ce mémoire doit être disponible sur les périphériques mobiles. Le choix de développer une application mobile native peut être intéressant s'il est possible d'identifier exactement le public cible et de connaître le type d'appareils

utilisé (Android ou iOS). Pour ce projet, le public cible étant des garderies en milieu familial, il n'est pas possible d'avoir une plateforme uniforme, car cela dépend des préférences des utilisateurs. Comme démontré sur la Figure 4.1, au Canada, il y a environ 54% d'utilisateurs utilisant iOS contre environ 45% d'utilisateurs utilisant android. Ainsi, il est nécessaire de développer deux applications différentes pour qu'une application fonctionne avec les deux systèmes d'exploitation.

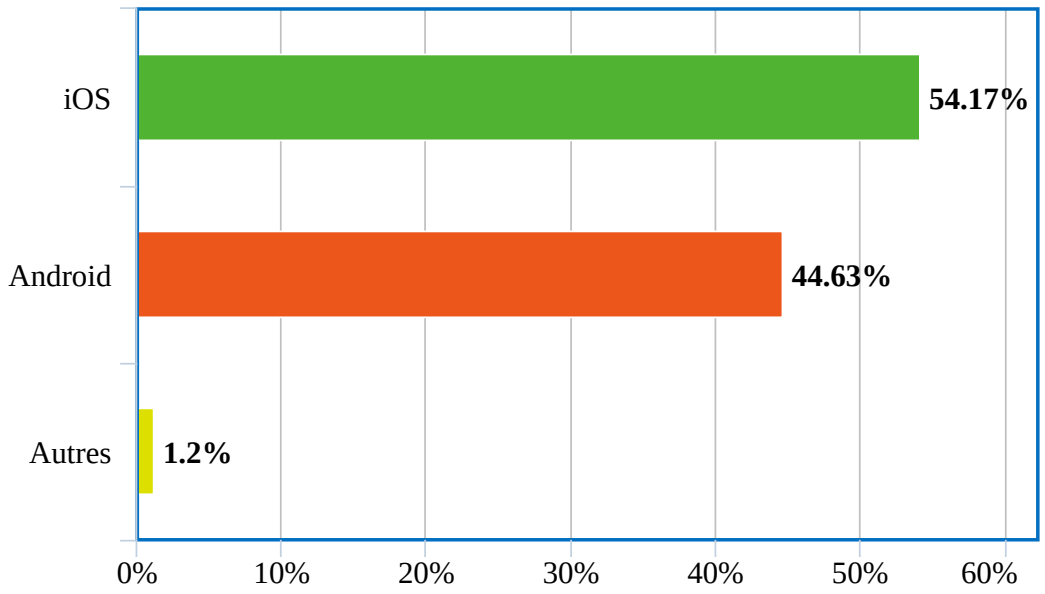


Figure 4.1 : Parts de marché des systèmes d'exploitation mobiles au Canada en 2018

Source : statcounter (2018)

Il est toutefois possible de développer une application mobile hybride avec des technologies comme Cordova (Wargo, 2015) ou Phonegap (Wargo, 2012) qui permettent d'utiliser des technologies web afin de créer une application mobile. Cette application mobile utilise une partie de code natif permettant d'ouvrir un navigateur web allégé et d'exécuter le code JavaScript et HTML développé. Cette technologie est avantageuse

puisqu'elle permet de développer une application mobile sans dupliquer le code afin de créer une application compatible avec iOS et Android. Cependant, elle nécessite d'utiliser des langages web pour exécuter les algorithmes. Pour l'optimisation cela pose problème, car les logiciels ne sont pas conçus pour fonctionner en web et les performances des appareils mobiles ne permettent pas de réaliser des calculs complexes.

Malgré le fait que les parts de marché des périphériques mobiles grandissent alors que les parts de marché des ordinateurs diminuent, la part de marché des ordinateurs est quand même élevée (comme le présente la Figure 4.2, le marché est représenté par environ 42% de postes Windows) et si une application vise à toucher le plus d'utilisateurs possible, il faut qu'elle soit compatible avec les systèmes d'exploitation des ordinateurs.

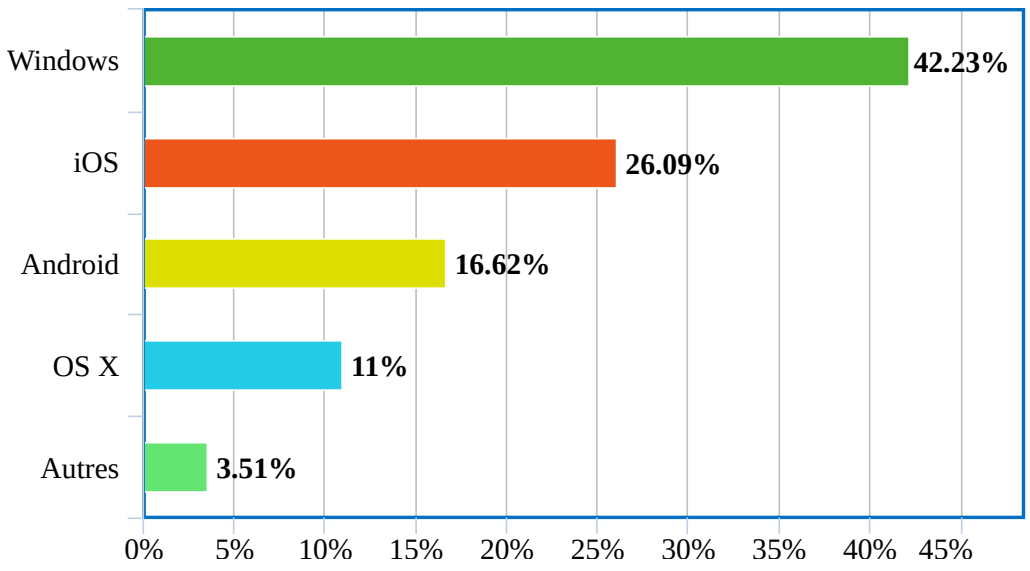


Figure 4.2 : Parts de marché des systèmes d'exploitation au Canada en 2018

Source : statcounter (2018)

Pour qu'une application soit compatible avec tous les périphériques mobiles et

les ordinateurs, il est très intéressant d'utiliser les technologies web. En effet, tous ces périphériques disposent d'un navigateur web et permettent de visiter un site web. Il est aussi possible de changer l'interface en fonction de la taille de l'écran pour la rendre plus ergonomique et utilisable facilement sur un écran de petite taille (responsive design). Il est aussi possible d'utiliser n'importe quel langage de programmation pour implémenter la partie serveur étant donné que le but de cette approche est de développer une API (Application programming interface) et ensuite de développer l'interface graphique dans un langage pour client web. La plupart des langages disposent de librairies pour faire des API, par exemple Spring en Java, Ruby on Rails, ou Django pour Python. Avec ces technologies, il est très simple de faire du MVC (Modèle Vue Contrôleur) et ainsi de totalement séparer l'interface graphique du code de contrôle.

L'application est développée en Java (côté serveur) et fournit une API REST (Representational State Tranfert) pour que le client accède aux données nécessaires. La Figure 4.3 présente la vue globale de l'architecture de l'application. Un serveur CMPL est disponible sur le réseau et l'application Java communique avec ce serveur via le protocole XML RPC (Remote Procedure Call). XML RPC (Merrick *et al.*, 2006) est un protocole de web service permettant à deux applications, dans deux environnements différents, de communiquer et de s'échanger des données. C'est une procédure distante utilisant le protocole HTTP pour le transport des données et XML pour l'encodage des données. XML RPC était défini comme la façon la plus simple de faire de la programmation distribuée à l'époque où il est sorti (Merrick *et al.*, 2006). Ensuite il a évolué et est devenu SOAP (Simple Object Access Protocol) qui est une API basée sur XML. Aujourd'hui, la tendance a encore évolué et c'est REST qui est le protocole

dominant pour les échanges de messages entre applications. XML RPC étant le seul protocole de communication accepté par le serveur CMPL, il est obligatoire de l'utiliser. C'est pour cela que REST est utilisé seulement pour communiquer avec le client web seulement et non pour communiquer avec le serveur CMPL.

Quant à elle, l'application côté client est développée en Vuejs qui est un framework de développement web "front end". Elle fait des appels à l'API REST fournie par le serveur pour récupérer et envoyer des données et les affiche à l'utilisateur. L'application serveur est composée de 4 modules. Deux modules sont dédiés à la récupération des ingrédients et des recettes, le troisième module est destiné à générer le modèle d'optimisation et le dernier module permet de contrôler l'API REST et de faire le lien entre les trois autres modules et l'application côté client. Elle est présentée plus en détail dans la section suivante.

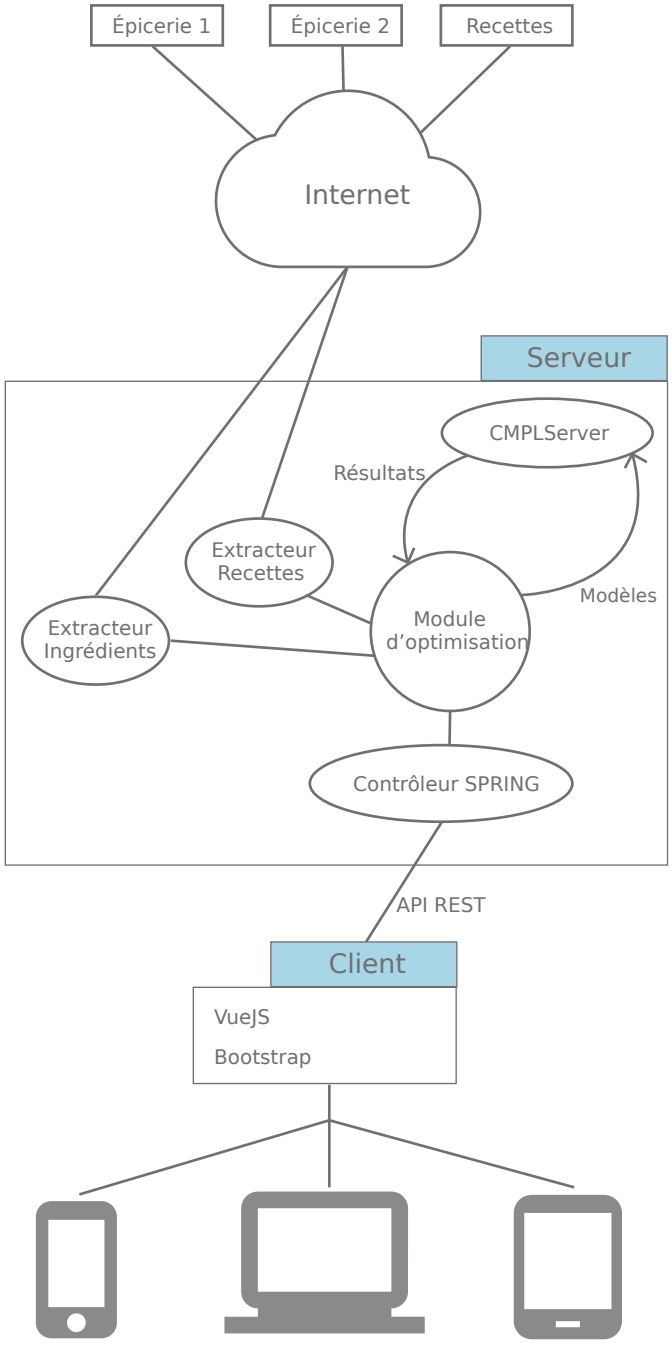


Figure 4.3 : Architecture globale de l'application

4.2 "BACK END"

En programmation logicielle, les termes "front end" et "back end" réfèrent à la séparation des rôles entre les couches de présentation ("front end") et les couches d'accès aux données ("back end"). Dans une application client-serveur, le client est considéré comme étant le "front end" et le serveur comme étant le "back end". Dans l'architecture logicielle proposée dans ce travail, ici le "back end" se compose de trois blocs distincts. Le premier est le module de récupération des données sur internet, le second est le module dédié à l'optimisation et le troisième est module permettant de gérer une API mettant à disposition toutes les données et fonctionnalités des deux modules précédents. La partie "Back end" est présentée dans le bloc serveur de la Figure 4.3

4.2.1 RÉCUPÉRATION DES DONNÉES

Le premier module permet de récupérer les données sur les sites de recettes et d'épiceries locales. Il permet de récupérer les prix des aliments une fois par semaine (le jeudi, car les circulaires de réductions hebdomadaires sont effectives à partir du jeudi). Pour les recettes, il suffit de les récupérer une seule fois, car elles ne changent pas souvent. De plus, un prétraitement des données initiales a été réalisé dans la base de données des recettes pour éliminer toutes les recettes qui n'étaient pas adaptées pour une garderie en milieu familial. Ce prétraitement a été complété manuellement, car il n'est pas possible pour un algorithme de définir ce qu'est une bonne recette en ne connaissant que les ingrédients et le temps de préparation. Ce procédé est donc le seul module qui n'est pas automatisé dans le processus. Cependant, cette étape est nécessaire une seule

fois au moment de la création de la base de données ou lorsque que des nouvelles recettes doivent être ajoutées. Dans la définition du problème et dans les tests préliminaires d'optimisation, le nombre de recettes était fixé à 200, mais dans la version finale de la base de données, celle-ci en contient cent, dont environ vingt recettes de chaque type (boeuf, volaille, poisson et fruits de mer, porc, végétarien et oeufs) car certaines recettes peu adaptées ont été retirées. Une fois les recettes récupérées et triées, elles sont classées automatiquement dans un des cinq types. Par exemple, si une recette contient au moins un ingrédient de type boeuf, la recette sera classée automatiquement par un algorithme dans la catégorie boeuf. Si elle ne contient pas de viande ni de poisson alors elle sera classée en recette végétarienne. Après exécution de l'algorithme, il ne contenait que quelques faux positifs sur les recettes qui étaient composées de plusieurs types de viandes. Si une recette contient plusieurs viandes différentes, elle est catégorisée dans la catégorie de sa viande principale (celle présente en plus grande quantité). Finalement, en base de données une recette est représentée par :

— Un identifiant unique

— Un nom

— Un type

— Une liste d'ingrédients (comprenant un type d'ingrédient et une quantité).

Pour les ingrédients, la procédure est un peu plus compliquée que pour les recettes, car il faut absolument que tout soit automatisé pour pouvoir exécuter l'algorithme automatiquement une fois par semaine et ne pas corrompre la base de données. Avant toute chose, la base de données est copiée dans un répertoire de sauvegarde. Comme cela, en cas de corruption des données à cause d'une erreur dans l'un des algorithmes,

il est facile de revenir en arrière pour l'administrateur du système. Dans un premier temps, les ingrédients sont récupérés via un web crawler sur les sites de deux épiceries locales différentes. Le web crawler a été développé spécialement pour cette application. Il est basé sur une conversion du code HTML en un arbre XML (Plus simple à parcourir en Java) à l'aide de la librairie Jsoup. Seules les catégories contenant de la nourriture sont parcourues pour éviter de polluer la base de données avec des produits inutiles. Ensuite à l'aide d'expressions régulières, seuls les éléments importants sont récupérés. Un ingrédient est représenté en base de données par :

— Un identifiant unique

— Un nom

— Une marque

— Un prix

— Un type.

La base de données compte environ 7000 ingrédients par épicerie. Le prix calculé est le prix médian de tous les ingrédients d'un même type. Cette stratégie permet d'avoir seulement environ 600 ingrédients strictement différents pour le module d'optimisation. Le type des ingrédients est automatiquement fixé grâce à un algorithme de "text-mining"(Gupta et Lehal, 2009) présenté dans la partie suivante.

CLASSIFICATION DES INGRÉDIENTS

Les ingrédients sont classés via un algorithme de classification (David Nadeau et Satoshi Sekine, 2007) basé sur une distance de Levenshtein (Yujian et Bo, 2007).

En "text-mining" la distance de Levenshtein est une métrique utilisée pour comparer des chaînes de caractères. La distance de Levenshtein entre deux mots est le nombre minimum de modifications simples (insertion, suppression, substitution) sur un seul caractère à faire pour qu'une séquence se transforme en une autre séquence. Par exemple : la distance de Levenshtein entre parure et toiture est de 4 :

1. toiture $\rightarrow$ poiture (Substitution)
2. poiture $\rightarrow$ paiture (Substitution)
3. paiture $\rightarrow$ parture (Substitution)
4. parture $\rightarrow$ parure (Suppression)

Une distance de Levenshtein de 0 est la meilleure distance possible. Cela signifie que les deux chaînes de caractères sont identiques.

L'algorithme de classification compare donc les ingrédients récupérés sur les sites d'épiceries avec une liste de mots comprenant la plupart des légumes et des viandes communes sur le marché. Cette liste de mots est représentée sous forme de "Trie" (Aref, 1998).

Un "Trie" est une structure de données permettant de stocker des données ayant un préfixe commun (souvent des chaînes de caractères) sous forme d'arbre et de les parcourir de manière plus optimisée que dans une liste. Le premier nœud correspond à la chaîne de caractères vide. La Figure 4.4 présente un exemple de "Trie". La complexité algorithmique de recherche dans un tel arbre est de $O(n)$ pour tous les cas avec n la taille de la donnée à récupérer (pour une chaîne de caractères, c'est le nombre de caractères). Comparativement, une liste possède une complexité de $O(n)$ pour le pire des cas avec n

la taille de la liste.

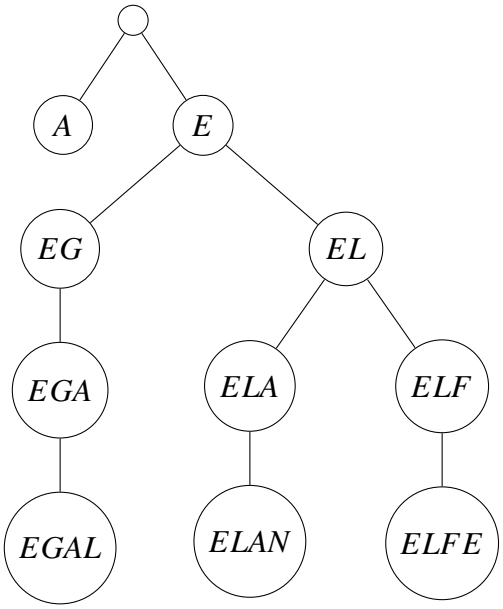


Figure 4.4 : Exemple d'un "Trie" avec les mots : a, egal, elan, elfe

Comme présenté sur la Figure 4.5, si les données à stocker dans l'arbre sont volumineuses il est facile de compresser l'arbre en mettant dans chaque nœud seulement la nouveauté par rapport au préfixe. Dans ce cas, il faudra reconstruire l'information tout au long du parcours de l'arbre.

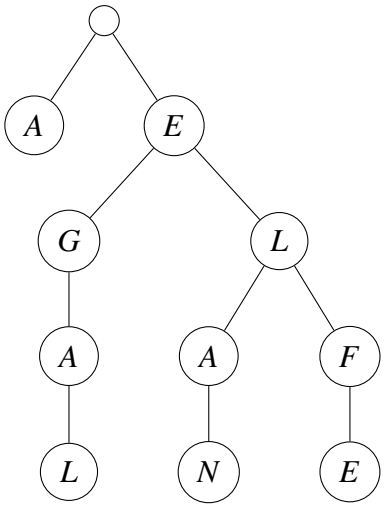


Figure 4.5 : Premier niveau de compression du "Trie"

Cette méthode de compression peut être utile dans le cas de stockage de données lourdes dans les nœuds, car elle évite de surcharger la mémoire. Cependant elle ne permet pas de réduire le temps de calcul, car la taille de l'arbre reste identique. Pour stocker des chaînes de caractères, qui ne sont pas des informations très lourdes, cette modification ne possède qu'un impact mineur. Il est aussi possible de compresser l'arbre en contractant les nœuds intermédiaires. C'est-à-dire que chaque nœud ne possédant qu'un seul fils sera fusionné avec. Cette méthode peut réduire grandement la complexité de parcours de l'arbre, car les nœuds n'ayant qu'un seul fils sont supprimés. Cependant le temps d'ajout de données dans la structure est augmenté, car dans le cas ou une donnée ayant le même préfixe qu'une autre est ajoutée, il faut recréer un nœud pour le préfixe, un autre pour l'ancienne donnée, et un dernier pour la nouvelle donnée. Cette modification est répertoriée dans la littérature sous le nom de "radix Trie" (Leis *et al.*, 2013). La Figure 4.6 représente les mêmes données que sur la Figure 4.4 mais sous forme de "radix Trie". Cette structure de donnée permet d'obtenir une complexité algorithmique de $O(n)$ dans le pire des cas, mais en moyenne cette complexité est réduite.

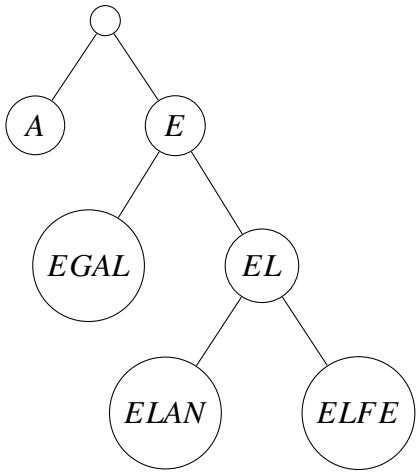


Figure 4.6 : Exemple d'un "radix Trie" avec les mots : a, egal, elan, elfe

Étant donné que dans l'utilisation courante de l'algorithme il n'y a aucune donnée à ajouter à l'arbre, cette solution est la meilleure pour stocker les types de produits. La liste des produits a été générée dans un premier temps par une liste des légumes, des types de viandes et des poissons différents. Mais cette liste était très incomplète et il a donc été nécessaire de la mettre à jour. Une interface graphique a donc été créée pour afficher tous les ingrédients non classés et pouvoir entrer un nouveau type manuellement (cette interface graphique est présentée sur la Figure 4.7). Une fois tous les types entrés dans le "Trie", l'algorithme de classification classe de manière autonome tous les ingrédients.

Certains ingrédients sont indésirables pour notre base de données. En effet, pour la cuisine, il ne faut que des ingrédients non modifiés et sur les sites d'épiceries il est commun de retrouver des aliments tels que vinaigrette aux pommes qui était classée comme de la pomme, par exemple. Il a donc été ajouté un nouveau "Trie" d'ingrédients interdits. Si un ingrédient peut être classé dans l'un de ces types, alors il ne faut pas le mettre dans la base de données. Ensuite l'algorithme a encore été amélioré pour

reconnaître de manière plus fine les viandes et les poissons. La première version de l'algorithme ne reconnaît que le type de viande ou de poisson (Boeuf, Porc, Saumon, Truite, etc.). Une version améliorée a donc été implémentée. Cette version contient un troisième "Trie" contenant une liste des morceaux de poisson ou de viande (Filet, jambon, magret, cuisse, etc.). Ainsi avec cette version, les types d'ingrédients classés correspondent exactement au genre d'ingrédients retrouvés sur les sites de recettes.

Une capture d'écran de l'interface développée pour aider à saisir tous les types dans les "Tries" est présentée en Figure 4.7. Il est possible d'y voir la liste des ingrédients disponibles dans les recettes du site de recettes sur la gauche. Sur la droite le premier champ de texte représente le type dans lequel un ingrédient a été classé par l'algorithme et les champs d'après permettent de le classer dans un autre type et de rentrer ce type s'il n'existe pas.

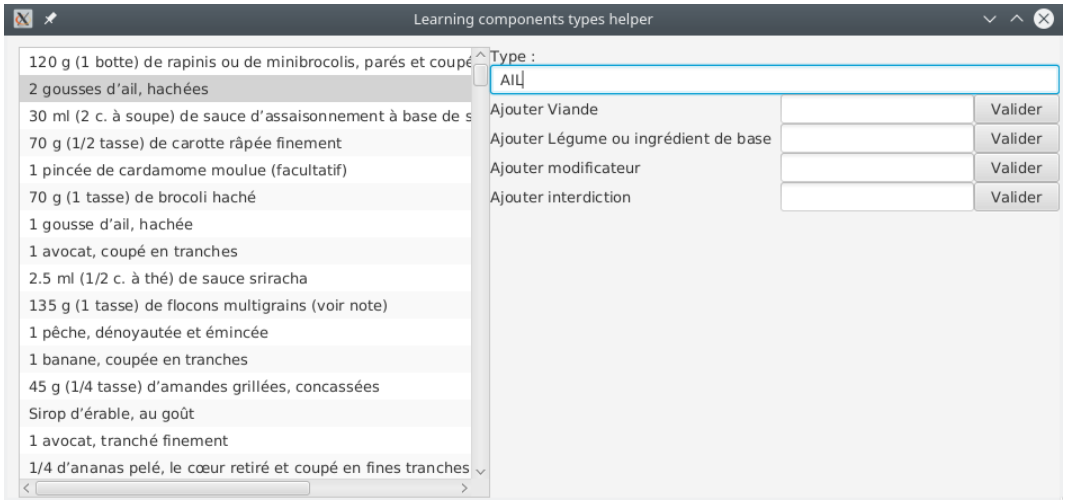


Figure 4.7 : Interface graphique développée pour vérifier et corriger les décisions de l'algorithme de classification

Après avoir ajouté les types d'ingrédients manquant dans les "Tries" grâce à

l'interface développée, l'algorithme fonctionne de manière autonome chaque semaine pour classer à nouveau les ingrédients récupérés sur les sites des épiceries. Maintenant que la classification des ingrédients a été définie, il est présenté dans la section suivante comment les données sont utilisées par le module d'optimisation.

4.2.2 MODULE D'OPTIMISATION

Le chapitre 3 présente le modèle mathématique défini pour résoudre le problème ainsi que le solveur "CBC de coin-or" (Forrest *et al.*, 2018) utilisé pour résoudre le problème. Pour automatiser la résolution du problème, il est donc nécessaire de développer une interface en Java pour convertir les paramètres et les données réelles du problème (ingrédients, recettes et préférences utilisateur) dans un format admissible par le solveur d'optimisation. Avec le serveur d'optimisation CMPLserver (Steglich, 2016) utilisé pour ce projet, en Java, il existe deux solutions. Soit utiliser l'interface jCMPL (Java CMPL) qui est une interface développée pour Java permettant de créer des modèles mathématiques directement en Java et d'y insérer les paramètres. La seconde solution, plus laborieuse, envoie un modèle mathématique écrit avec CMPL et un fichier de paramètres via l'interface XML RPC du serveur. La première méthode est un raccourci de la seconde méthode, car dans l'interface jCMPL les méthodes Java permettent de créer le fichier du modèle mathématique et de l'envoyer au serveur via XML RPC de manière transparente pour le développeur. Malheureusement l'interface jCMPL ne fonctionne pas correctement dans le cas de ce projet, car les fichiers de paramètres sont trop gros et cela déclenche des exceptions dans le code de jCMPL

le rendant inutilisable. Sous les conseils de son créateur, le Prof. Dr. Mike Steglich [1] pour des fichiers de cette taille, il vaut mieux utiliser directement un fichier CMPL à envoyer au serveur via XML RPC, car la version actuelle de jCMPL possède un bug. Il a donc été nécessaire de créer un module permettant de générer le fichier de paramètres et de l'envoyer au serveur avec XML RPC. Ce fichier contient la liste de tous les ingrédients ainsi que leur prix au kilogramme, la liste de toutes les recettes, une matrice indiquant la quantité de chaque ingrédient (en kilogramme) dans les recettes ainsi que les préférences des utilisateurs (ingrédients déjà possédés et nombre de recettes de chaque type). Ce fichier de paramètres est ensuite envoyé au serveur accompagné du modèle mathématique. Après avoir envoyé le fichier, le module d'optimisation attend la réponse du serveur. Cette réponse peut être lue avec la librairie jCMPL. Elle contient beaucoup d'information, mais les seules qui sont utilisées et récupérées dans cette approche sont :

— La valeur de la fonction objective

— La solution optimale

— Le temps de résolution

Il y a aussi un affichage des bornes minimales et maximales de chacune des variables et des contraintes satisfaites ou insatisfaites. Une fois la réponse reçue, toutes les informations sont stockées dans des instances qui seront renvoyées au client via l'API REST (Fielding, 2000) présentée dans la section suivante.

1. Technical University of applied science Wildau, Germany

4.2.3 CONTRÔLEUR MVC

Tel que défini dans la section 4.2, le dernier module de la partie serveur présenté est le module permettant de mettre à disposition toutes les informations de la base de données et les fonctionnalités du module d'optimisation. Ces fonctions sont rendues accessibles depuis l'extérieur du programme serveur via le style d'architecture REST (Representational State Transfer). Roy Fielding (2000) a défini REST dans sa thèse de doctorat "Architectural Styles and the Design of Network-based Software Architectures" à l'université de Californie à Irvine. Il y a développé le style d'architecture REST en parallèle du protocole HTTP 1.1 de 1996 à 1999, basé sur le modèle existant de HTTP 1.0 de 1996. REST est un style architectural pour services web permettant d'allier performances, simplicité, portabilité et flexibilité (Bouhaï et Saleh, 2017). Ces services web permettent à des périphériques de dialoguer entre eux et ce peu importe les langages qu'ils utilisent. Les web services utilisent des protocoles de communication communs comme HTTP, XML ou JSON (JavaScript Object Notation). Un service web est disponible sur internet et dispose d'une interface programmée de manière interprétable automatiquement par des applications clientes. L'utilisation de langages et de protocoles indépendants des plateformes renforce l'interopérabilité entre services (Bouhaï et Saleh, 2017). REST, plus précisément est basé sur plusieurs concepts :

— L'API doit être sans états. C'est-à-dire que lors d'un transfert de données, aucune des deux parties ne connaît l'état de l'autre et lors de l'envoi d'une donnée il n'y a pas d'accusé de réception.

— Il doit être possible d'accéder en lecture à toutes les ressources disponibles sur le serveur en utilisant seulement l'URI (Uniform Resource Identifier).

— Il ne doit pas y avoir de chiffrement intégré à l'API.

— Il ne doit pas y avoir de session.

— Le seul protocole de communication utilisé doit être HTTP.

— Pour faire des opérations de type CRUD (Create Read Update Delete) il faut utiliser respectivement les méthodes HTTP post, get , put and delete.

— Les données doivent être transférées uniquement par le biais de formats de données légers (JSON, XML...).

Si un service web utilise tous les principes énumérés ci-dessus, ce service est RESTful. Le module développé dispose donc de plusieurs méthodes pouvant créer, lire, modifier et supprimer les données de la base de données, mais il peut aussi démarrer une optimisation avec des paramètres ou sans paramètres.

La stratégie utilisée pour faire les requêtes de type CRUD est triviale et respecte strictement le cahier des charges de REST. Il suffit, par exemple, de faire une requête get vers l'URI "/ingrédients" pour récupérer la liste des ingrédients en base de données. De la même manière, s'il est nécessaire de récupérer un ingrédient spécifique, il faut faire une requête get sur l'URI "/ingrédient/id" pour récupérer l'ingrédient portant l'identifiant "id". Il est aussi possible de récupérer les ingrédients d'un même type avec l'URI "/ingrédient/type". Les mêmes fonctionnalités sont implémentées pour les recettes et le fonctionnement est très similaire pour les autres opérations (création, suppression et modification). Par contre, les requêtes utilisées pour lancer l'optimisation ont un fonctionnement un peu différent. En effet, elles ne sont pas destinées comme les autres à seulement faire de la lecture et de l'écriture de données, car elles exécutent des

calculs en fonction des données reçues. Le cas général est le cas où une méthode get est utilisée sur l'URI "/optimiser". Dans ce cas, aucune donnée n'est fournie et le contrôleur va demander au module d'optimisation de lancer une optimisation sans paramètres, c'est-à-dire que le modèle de base sera exécuté et que cinq menus de type différents seront trouvés au coût le plus faible pour la semaine. Néanmoins il est aussi possible d'envoyer une requête avec une méthode post sur l'URI "/optimiser". Le contenu de cette requête est un fichier JSON contenant des préférences de l'utilisateur telles que le nombre de recettes de chaque type pour la semaine, les recettes qu'il ne souhaite plus jamais obtenir en résultat et les ingrédients qu'il possède. Lorsque le contrôleur reçoit ce type de requête, il formate et envoie à nouveau toutes les informations au module d'optimisation qui génère un fichier de paramètres personnalisé et l'envoyer au serveur d'optimisation. Dans tous les cas, la réponse du contrôleur au client est un fichier JSON contenant une semaine pour chacune des épiceries avec les recettes pour chaque jour, ainsi que les prix totaux de chacune des semaines. La Figure 4.8 présente le fichier JSON présent dans la requête du client et la Figure 4.9 montre le fichier de réponse du contrôleur au client.

```
{
  "recettesInterdites": [
    {
      "id": 14,
      "name": "Gnocchis au chou-fleur caramélisé",
      "url": "https://www.ricardocuisine.com/recettes/7724-gnocchis-au-chou-fleur
        -caramelise",
      "type": "vegan"
    },
    {…}
  ],
  "IngredientsPossédés": [
    "200g SAUMON"
  ],
  "IngredientsInterdits": [
    "Arachide"
  ],
  "équilibre": [
    "1",
    "0",
    "1",
    "1",
    "2"
  ]
}
```


Figure 4.8 : Format de la requête en JSON du client


```
{
  "prix1": 89.1036,
  "Epicerie1": {
    "Lundi": {
      "id": 60,
      "nom": "Quesadillas à la viande fumée et au cheddar",
      "url": "https://www.ricardocuisine.com/recettes/6488-quesadillas-a-la-viande
        -fumee-et-au-cheddar",
      "type": "boeuf"
    },
    "mardi": {…},
    "mercredi": {…},
    "jeudi": {…},
    "vendredi": {…}
  },
  "Epicerie2": {…},
  "prix2": 102.304
}
```


Figure 4.9 : Format de la réponse en JSON du contrôleur

4.3 "FRONT END"

En génie logiciel, le "front end" est la partie de programmation qui se consacre à transformer les données brutes en une interface graphique permettant de les interpréter ou de les générer. Pour cette application, il a été choisi de la développer sous forme

d'application à page unique (Mikowski et Powell, 2013). Une application à page unique est une application web qui interagit avec l'utilisateur en modifiant dynamiquement le contenu de la page sans avoir à recharger entièrement de nouvelles pages. Cette approche permet d'éviter des interruptions de l'expérience utilisateur (Hassenzahl et Tractinsky, 2006). Le développement d'une application à page unique implique une communication dynamique entre le client et le serveur. Cette communication s'effectue grâce à l'API REST définie dans la section 4.2.

4.3.1 CHOIX ERGONOMIQUES

Étant donné que l'application est dédiée à des personnes dont le métier ne tourne pas autour de l'outil informatique, il est important que l'application soit facile à utiliser et qu'elle ne demande pas de formation pour être utilisée. C'est pour cela que dans un cas standard d'utilisation, il n'y a aucune action à faire autre que se rendre sur le site pour recevoir la liste des recettes et la liste d'épicerie. La liste des recettes s'affiche directement au centre de l'écran, car en conception d'interfaces le centre de l'écran est réservé pour les informations les plus importantes. La Figure 4.10 présente l'affichage de la liste des recettes. Une estimation du prix pour la semaine est affichée juste en dessous à droite des recettes.

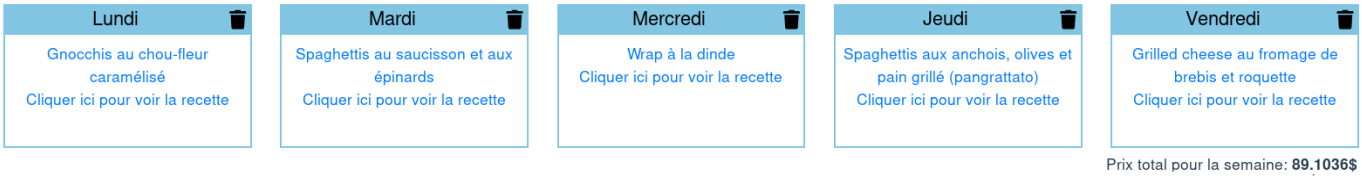


Figure 4.10 : Affichage de la liste des recettes

La seule action requise par l'utilisateur est le choix de l'épicerie via une liste déroulante, si l'épicerie par défaut n'est pas celle où il a l'habitude de se rendre. Chaque semaine, à partir du samedi, l'application propose une nouvelle liste de repas automatiquement en fonction des promotions hebdomadaires en conservant les préférences que les utilisateurs ont entrées les semaines précédentes. Les prix sont récupérés sur les sites des épiceries tous les jeudis (c'est le jour où les promotions hebdomadaires prennent effet).

Si les utilisateurs souhaitent utiliser l'application de manière plus avancée, il existe plusieurs options permettant d'améliorer la liste de recettes. L'option la plus simple à utiliser est de supprimer une recette si elle ne convient pas aux goûts de l'utilisateur. Pour cela, il suffit de cliquer sur l'icône de poubelle présente sur chaque recette et ensuite de cliquer sur "Optimiser la liste des repas". Cela va redémarrer le processus d'optimisation sur le serveur et fournir une nouvelle liste sans la recette indésirable. Il est possible pour l'utilisateur d'indiquer les aliments qu'il possède dans son garde-manger afin que l'algorithme fournisse (si possible) des recettes contenant ces ingrédients. Une fenêtre de paramètres est aussi disponible permettant à l'utilisateur de choisir le nombre de repas de chaque type qu'il souhaite chaque semaine. Cette fenêtre est présentée en Figure 4.11.

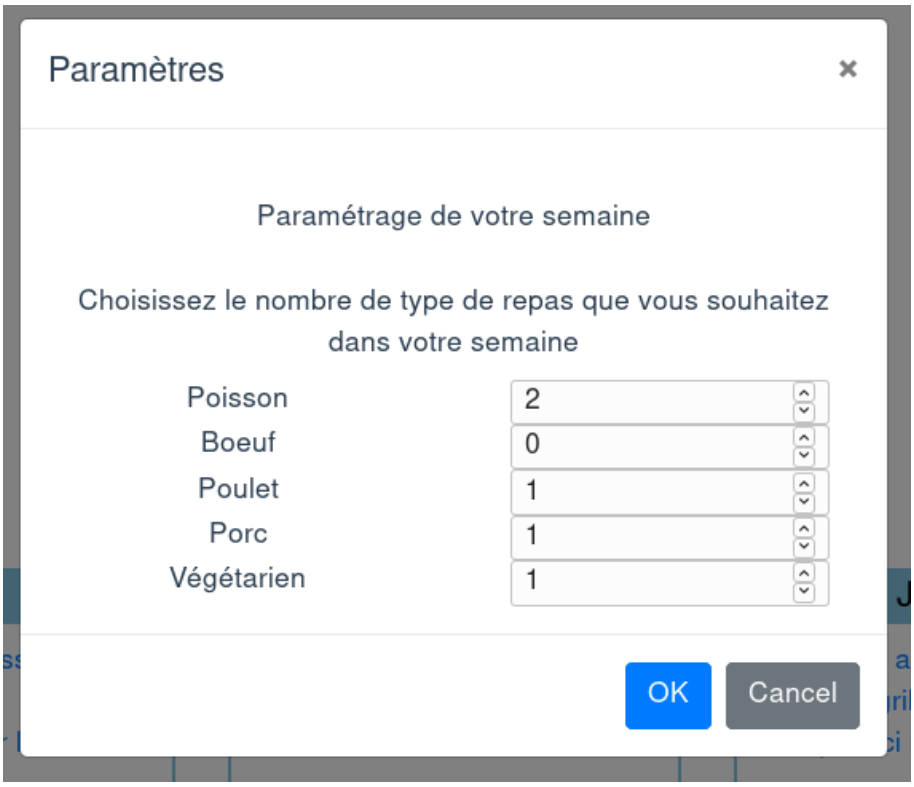


Figure 4.11 : Fenêtre de paramétrage de l'application

C'est une fenêtre modale qui vient recouvrir le reste de l'application jusqu'à ce qu'elle soit fermée. Lorsqu'elle est active, le reste de l'application est désactivé. En design d'expériences utilisateur, les fenêtres modales sont utilisées dans un contexte où il est nécessaire pour l'utilisateur d'entrer des informations sans perdre l'environnement contextuel dans lequel il se trouve. Finalement, un petit aide-mémoire est affiché dans le coin supérieur gauche pour permettre à un utilisateur non habitué de découvrir les fonctionnalités avancées de l'application. Lors de la réalisation de l'expérimentation dans les garderies en milieu familial, un guide d'utilisation détaillé de l'application a été fourni à tous les utilisateurs. Ce guide est disponible en Annexe A de ce mémoire. La vue globale de l'interface graphique est disponible en Annexe B de ce mémoire. Chaque préférence de l'utilisateur telle que les recettes qui ont été supprimées et les paramètres

de types de recettes sont enregistrés pour les semaines suivantes afin d'éviter que le paramétrage de l'application soit répétitif pour l'utilisateur.

CHAPITRE V

RÉSULTATS

Ce chapitre présente le protocole expérimental et les résultats obtenus par l'algorithme d'optimisation sur le problème de génération automatique de planification de menus. La première section décrit comment l'algorithme a été testé numériquement ainsi que les résultats obtenus. Ensuite, une étude qualitative a été mise en place pour que des gestionnaires de garderies puissent tester le logiciel et nous retourner leur appréciation du logiciel. Pour l'expérimentation, nous disposions d'une certification du comité d'éthique de l'UQAC pour procéder aux expérimentations du logiciel en contexte réel ainsi que pour recueillir les données et appréciations.

5.1 RÉSULTATS NUMÉRIQUES

Les résultats numériques sont les résultats obtenus par l'utilisation de l'algorithme. Dans un premier temps, l'algorithme a été testé pour voir si les résultats qu'il retourne sont cohérents et si la vitesse d'exécution est satisfaisante. Ensuite, toutes les données numériques collectées pendant l'expérimentation ont été analysées pour vérifier si des incohérences n'arrivent pas après de nombreuses utilisations de l'algorithme.

5.1.1 VALIDATION DE L'ALGORITHME

Pour cette section, le nombre de repas par semaine est fixé à 5, car c'est le nombre de jours ouvrables dans une semaine de garderie. Il n'y a pas d'ingrédients enregistrés

dans les "ingrédients possédés". L'algorithme fonctionne avec deux épiceries différentes et retourne une planification pour chacune d'entre elles. Les deux planifications peuvent être différentes, car la sélection des recettes se fait au niveau des prix dans les épiceries. Dans un premier temps, le programme a été testé sur 4 semaines (exécution de l'algorithme 4 fois en changeant manuellement la date) pour vérifier si toutes les contraintes sont bien respectées. Le Tableau 5.1 représente les 4 planifications pour la première épicerie et le Tableau 5.2 représente les 4 planifications pour la seconde épicerie. Les prix ont été récupérés sur les circulaires des épiceries en vigueur la semaine du 17 au 23 janvier 2019.

L'algorithme d'optimisation trouve une solution en 0.02 secondes pour les deux épiceries, cependant, il y a quelques délais supplémentaires dus au temps de communication entre l'interface, l'API et le serveur CMPL. La durée totale pour obtenir un menu de 5 repas pour une semaine et deux épiceries est d'environ 3 secondes pour 200 recettes dans la base de données de recettes et environ 7000 ingrédients dans chaque épicerie. Le problème se compose de 200 variables binaires (une pour chaque recette dans la base de données) et de 6 contraintes.

En ce qui concerne le web crawler, son temps d'exécution est assez long. Pour extraire les prix des ingrédients dans les deux épiceries, le temps n'est pas toujours le même. Au fil des semaines, des temps d'exécution entre 20 et 45 minutes sont observés. Cela est dû au fait que les requêtes HTTP sur les sites des épiceries sont lentes. De plus, pour éviter au web crawler d'être détecté par les politiques anti-robots des sites web parcourus, un temps aléatoire entre 1 et 3 secondes d'attente est ajouté entre chaque

requête. Cet ajout de temps n'est pas handicapant pour le reste de la contribution, car il est effectué une seule fois par semaine alors que le reste de l'infrastructure continue de fonctionner. Une fois la nouvelle base de données générée et vérifiée, alors l'ancienne base de données est échangée avec la nouvelle et l'algorithme peut fonctionner avec les nouvelles données hebdomadaires.

Tableau 5.1 : Planification sur 4 semaines calculée avec les prix de première épicerie

	Lundi	Mardi	Mercredi	Jeudi	Vendredi
Semaine 1 (89.10$)	Gnocchis au chou-fleur caramélisé	Spaghettis au saucisson et aux épinards	Wrap à la dinde	Spaghettis aux anchois, olives et pain grillé	Grilled cheese au fromage et roquette
Semaine 2 (112.26$)	Poisson blanc, sauce vierge aux olives	Carbonara au brocoli	Bavette express et salade de tomates et d'avocat à la sauce soya	Pizza merguez et hummus	Pâtes au poulet citronnées au persil
Semaine 3 (126.30$)	Saumon aux canneberges	Salade de kale au poulet et aux figues	Pâtes à la sauce puttanesca	Sandwich aux oeufs et au jambon	Boeuf au brocoli et au tofu
Semaine 4 (131.83$)	Omelettes aux blancs d'œufs et salade de kale	Poisson poêlé et salade d'orange au fenouil	Salade de haricots au melon et au canard fumé	Côtelettes de porc en croûte d'épices sur purée de chou-fleur	Bouraks à la viande

Tableau 5.2 : Planification sur 4 semaines calculée avec les prix de la seconde épicerie

	Lundi	Mardi	Mercredi	Jeudi	Vendredi
Semaine 1 (102.40$)	Saumon aux canneberges	Quesadillas à la viande fumée et au cheddar	Salade de spaghettis au chou	Sandwich aux oeufs et au jambon	Wrap à la dinde
Semaine 2 (131.56$)	Gnocchis au choufleur caramélisé	Spaghettis express à la tomate et au thon	Gnocchis au fromage crémeux, aux champignons et au bacon	Pâtes au poulet citronnées au persil	Grilled cheese au fromage de brebis et roquette
Semaine 3 (144.06$)	Poisson blanc, sauce vierge aux olives	Panini à la mortadelle et au provolone	Grilled cheese revisité	Boeuf au brocoli et au tofu	Salade de haricots au melon et au canard fumé
Semaine 4 (153.21$)	Omelettes aux blancs d'œufs et salade de kale	Wrap au poulet croustillant	Bavette à la grenade et à la feta	Spaghettis aux anchois, olives et pain grillé	Tartiflette express

Toutes les recettes présentes dans ces exemples de 4 semaines montrent que l'algorithme respecte bien les contraintes qui sont définies. Dans les 5 recettes proposées pour chaque semaine, il est possible de trouver les 5 types d'aliments différents (boeuf, poisson, volaille, porc et végétarien [1]). Dans les 4 semaines proposées par l'application, aucune des recettes n'est proposée plusieurs fois, ce qui permet de valider la contrainte de diversité des recettes pour un mois complet d'utilisation. Après un mois d'utilisation, les recettes déjà proposées peuvent revenir à nouveau. Cependant, les recettes que l'utilisateur a supprimées manuellement ne reviendront jamais.

Il est aussi possible de remarquer une légère différence de menus entre les deux épiceries, qui est due aux différences de prix sur certains produits qui rendent certaines recettes moins chères dans l'une des deux épiceries.

Par contre, une augmentation du prix total de la semaine est visible sur les 4 semaines. Au fur et à mesure que l'algorithme propose des planifications de repas, ces menus sont de plus en plus chers. C'est en effet dû au fait que l'algorithme trouve la combinaison de repas la moins chère possible pour un ensemble de prix (circulaire en vigueur modifiée chaque semaine) et un ensemble de recettes qui, lui, ne change pas chaque semaine. Le fait que les repas déjà proposés les dernières semaines ne soient plus disponibles élimine cet ensemble de 5 repas qui sont la solution optimale du problème d'optimisation est interdit donc le prix augmente forcément.

Ces tests ont été réalisés en une seule fois avec toujours les mêmes prix dans

1. Le type végétarien est composé des recettes sans viande et sans poisson, mais les recettes peuvent contenir des matières animales telles que les oeufs ou du lait.

l'épicerie ce qui fait que l'augmentation des prix est très flagrante. Dans un contexte réel avec un changement des prix chaque semaine, cette augmentation est moins importante. Cet exemple qui a été généré avec 4 semaines fictives, a été réalisé dans le but de vérifier que l'algorithme peut fournir une liste de repas variés pendant 4 semaines consécutives.

5.1.2 RÉSULTATS NUMÉRIQUES PROVENANT DE L'ÉTUDE

Pour la réalisation de l'expérience en milieu réel par les garderies, il a été décidé d'enregistrer toutes les informations demandées au serveur d'optimisation et renvoyées par celui-ci. Sur 8 semaines de tests (deux expériences de 4 semaines) il y a eu 144 utilisations différentes de l'application. Sur ces 144 utilisations, à chaque fois l'algorithme a retourné le bon nombre de recettes et le bon nombre de chaque type demandé. À aucun moment l'algorithme a été incapable de trouver de solution. La moyenne des prix calculés pour la première épicerie est de 106.78$ et la médiane de 104$ pour un menu hebdomadaire. Pour la seconde épicerie, la moyenne est de 102.93$ tandis que la médiane est de 102$. Le prix maximum quant à lui est de 242$, cependant ce nombre n'est pas vraiment significatif pour les données, car il a été trouvé avec une instance où l'utilisateur a demandé seulement des recettes contenant du boeuf. Grâce à ces données issues des expériences en milieu réel, il est possible de confirmer la supposition qui est faite dans le paragraphe précédent qui démontre que le prix augmente beaucoup chaque semaine à cause du fait que les prix à l'épicerie ne sont pas changés pour les 4 instances. En effet en condition réelle, les prix n'ont pas une telle augmentation au bout d'un mois d'utilisation.

5.2 RÉSULTATS QUALITATIFS

Finalement, une étude qualitative a été réalisée auprès de 9 garderies pour tester le logiciel en milieu réel. Le but de cette étude est de savoir si l'application est facilement utilisable par un service de garderie et si elle permet bien de réduire les coûts associés au repas. Les garderies participantes ne sont pas obligées de se contenter de prendre la liste de recettes telle quelle et ont le droit d'utiliser toutes les fonctionnalités de l'application pour modifier cette liste. Les garderies participantes ont utilisé l'application pendant 4 semaines et ont répondu chaque semaine à un formulaire composé de 6 questions. Ces questions sont :

Q1 : Combien avez vous réalisé de menus ? (sur une échelle de 1 à 5)

Q2 : Pourquoi n'avez-vous pas réalisé tous les menus ?

Q3 : Sur une échelle de 1 à 5 (1 étant pas du tout) quel est votre satisfaction au niveau des repas proposés ?

Q4 : Si vous avez utilisé l'application, croyez vous avoir diminué le temps de planification de vos repas (incluant les achat à l'épicerie) ?

Q5 : À combien de dollars estimez vous les économies réalisées à l'aide de l'application ?

Q6 : Avez-vous constaté une différence de prix entre vos achats et l'estimation du coût par l'application ?

Ces 6 questions permettent de répondre de vérifier si l'application a été utilisée, si les utilisateurs sont satisfaits, s'ils économisent du temps ou de l'argent et si l'estimation des prix que l'algorithme prend en compte est juste.

5.2.1 PREMIÈRE EXPÉRIMENTATION

Une première étude avec 5 garderies a été réalisée. Sur cette première série de tests, une des garderies a utilisé son droit de retrait, car elle jugeait que l'application n'était pas adaptée pour son travail et que les recettes étaient trop compliquées à réaliser. Pour les 4 autres garderies qui ont participé à l'expérimentation, les commentaires sont très similaires. En effet beaucoup de recettes sont soit trop compliquées à réaliser ou alors ne sont pas vraiment adaptées à des enfants. Cependant la question sur la différence de prix estimé par l'application et de prix réel a toujours été répondue sur "peu ou pas de différence". La base de données de recettes a donc été retravaillée pour enlever toutes les recettes indésirables et ne laisser plus que 100 recettes pertinentes (plutôt que plus de 200 recettes avec une grande partie de ces recettes non adaptées pour des enfants ou trop compliquées à réaliser). En vue de ces résultats et ces commentaires, une seconde expérimentation a été réalisée avec 5 garderies. Une des garderies de l'étude précédente a accepté de participer une seconde fois et tester à nouveau l'application avec la base de données modifiée.

5.2.2 SECONDE EXPÉRIMENTATION

Pour cette seconde série de tests, les 5 gérants de garderies ont participé à l'étude jusqu'à la fin. D'après les résultats obtenus, tous les participants ont noté que les prix estimés par l'algorithme n'ont pas ou ont peu de différences avec les prix réels dans les épiceries sélectionnées. Une seule personne n'a pas répondu à cette question, car elle n'a pas l'habitude de faire ses courses dans les deux épiceries proposées par l'application

et elle n'a pas voulu le faire pour l'expérimentation. Elle n'a donc pas répondu à cette question. Les utilisateurs ont réalisé une moyenne de 4 repas sur 5 proposés par l'application par semaine. Les raisons pour lesquelles ils n'ont pas réalisé les 5 repas sont souvent qu'un des repas ne leur plaisait pas ou le fait qu'ils souhaitaient réaliser un repas pour faire plaisir aux enfants. Le niveau moyen d'appréciation de l'application est de 4/5. Les Figures 5.1 et 5.2 montrent respectivement le nombre de repas réalisés par les utilisateurs et la note sur 5 d'appréciation qu'ils ont donné à l'application.

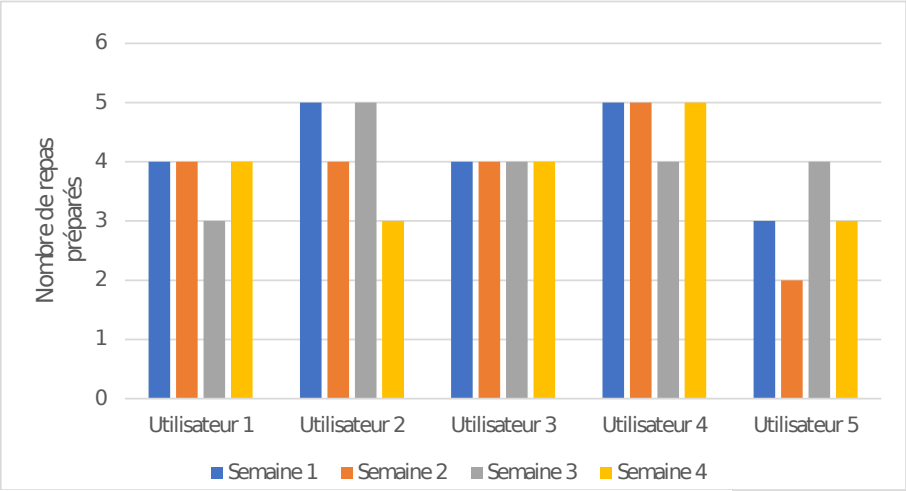


Figure 5.1 : Nombre de repas préparés par semaine et par utilisateurs

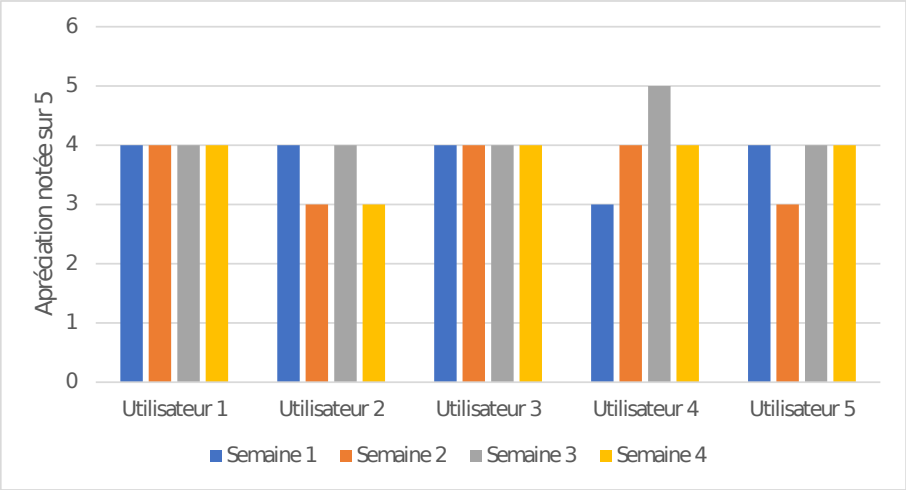


Figure 5.2 : Note de satisfaction des utilisateur pour chaque semaine d'utilisation de l'application

En moyenne l'application a diminué le temps de préparation et de planification de 22 minutes (avec une médiane à 15 minutes) par semaine aux utilisateurs. Ce temps inclut le temps de planification, le temps passé à l'épicerie ainsi que le temps de préparation des repas. La Figure 5.3 montre les économies de temps incluant la planification, la préparation des repas et les achats à l'épicerie pour chaque semaine et par utilisateur. Il est possible de remarquer sur la figure que pour 3 des utilisateurs, les économies de temps sont comprises entre 10 et 20 minutes, alors que les deux autres utilisateurs ont observé une économie de temps beaucoup plus importante.

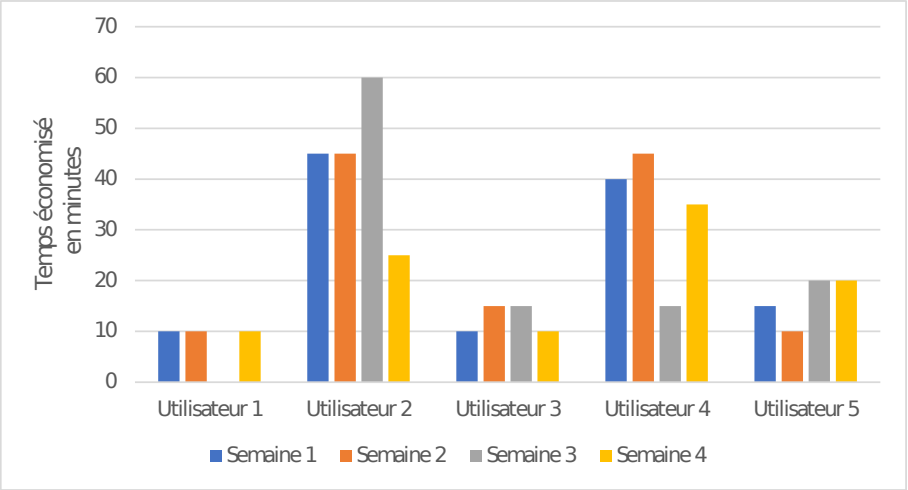


Figure 5.3 : Diminution du temps de préparation par utilisateur pour chaque semaine d'utilisation de l'application

Pour le gain financier, une des personnes dit n'avoir réalisé aucune économie. Elle dit même avoir payé plus cher pour l'une des semaines. Cependant les autres utilisateurs ont réalisé en moyenne 18$ d'économie par semaine (avec une médiane à 20$). La Figure 5.4 présente les économies financières réalisées par les utilisateurs chaque semaine d'utilisation de l'application.

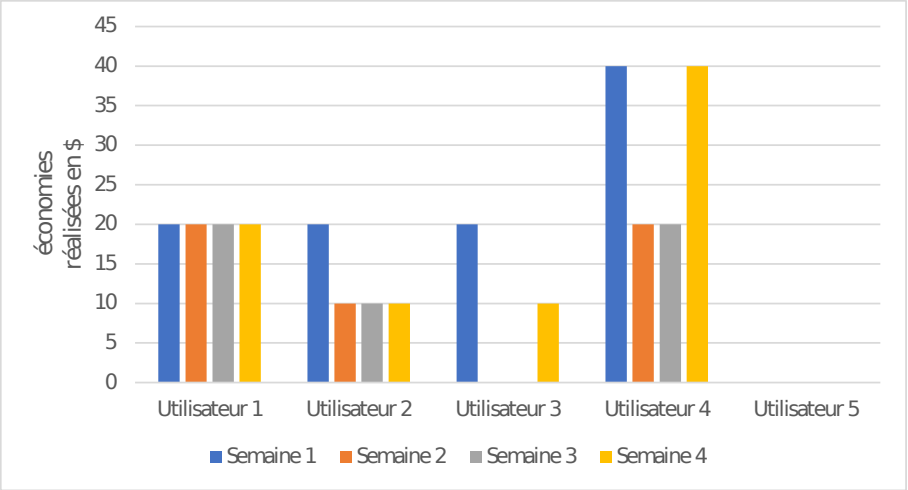


Figure 5.4 : Argent économisé par utilisateur pour chaque semaine d'utilisation de l'application

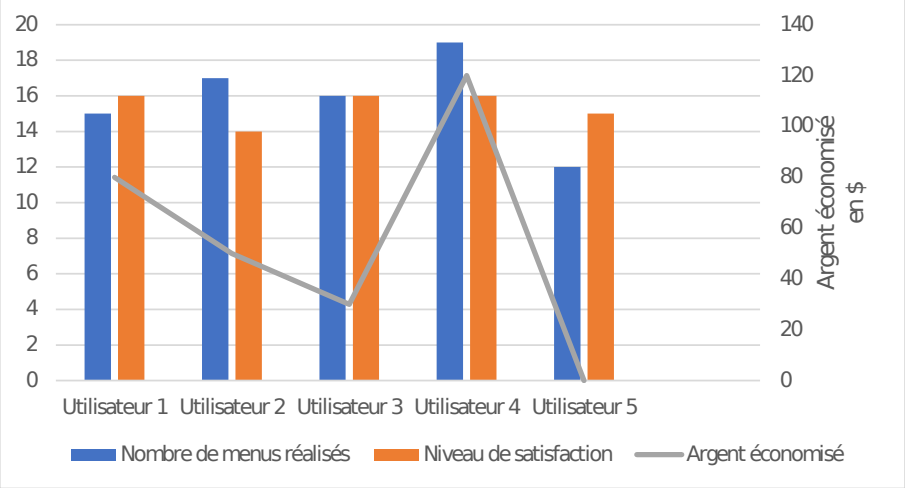


Figure 5.5 : Niveau de satisfaction, nombre de repas et argent économisé sur toute la durée de l'expérimentation

La Figure 5.5 montre l'argent économisé vis-à-vis des évaluations et du nombre de repas préparés il est possible de remarquer que la personne qui a effectué le plus d'économie est aussi la personne qui a préparé le plus de repas et que la personne qui a fait le moins d'économies est aussi la personne qui a réalisé le moins de repas. Malheureusement, l'échantillon est trop faible pour effectuer un test de corrélation afin d'observer un réel lien. La Figure 5.6 montre quant-à elle le temps économisé vis-à-vis des évaluations et du nombre de repas préparés. Il est seulement possible de remarquer que le temps économisé ne dépend pas vraiment du nombre de repas préparé, et qu'il n'influence pas la note donnée à l'application.

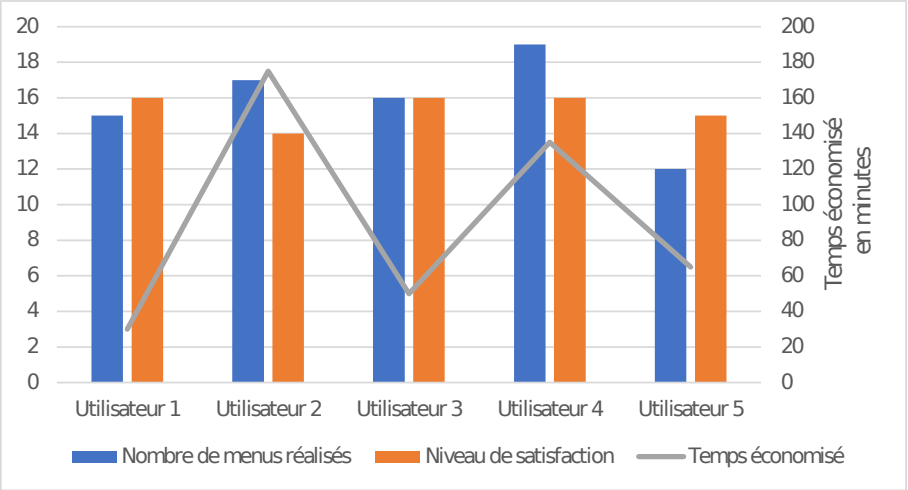


Figure 5.6 : Temps économisé par utilisateur pour chaque semaine d'utilisation de l'application

Les utilisateurs semblent avoir apprécié l'application et la plupart ont fait des économies en temps et en argent en utilisant l'application. La critique retenue est que l'application ne trouve pas les prix des épiceries à rabais et c'est pourtant dans ces magasins-là que les gérants de garderies se fournissent en majorité. Il a été également rapporté que parfois les repas semblent compliqués à préparer et qu'ils sont inhabituels. Cependant, il a été commenté que certains repas sont semblables aux repas que les gérants de garderies sont habitués de faire et qu'ils leur ont donné de bonnes idées de repas sans forcément suivre la recette proposée à la lettre.

CONCLUSIONS

Ce chapitre conclut les travaux ayant mené à ce mémoire. Un travail permettant aux gérants de garderies en milieu familial de trouver des repas variés à des prix abordables chaque semaine a été présenté. Ce chapitre présente les contributions effectuées ainsi que les possibles améliorations à effectuer dans les travaux futurs pour améliorer ce travail. Le chapitre 1 de ce mémoire a introduit les concepts d'optimisation nécessaires à la compréhension des algorithmes utilisés dans ce travail. Le chapitre 2 a décrit l'état actuel de la littérature scientifique dans ce domaine de recherche. Il a été remarqué que dans beaucoup de travaux ont été réalisé dans le domaine de la génération automatique de menus, mais qu'aucune approche n'était automatisée à 100% avec une gestion des prix et des réductions dans les épiceries locales. Le modèle présenté dans ce mémoire est basé sur une approche utilisant des recettes plutôt que des ingrédients comme les approches présentées par Seljak (2009) et Elsweiler et Harvey (2015).

Le chapitre 3 a défini le modèle mathématique et les solutions d'optimisation utilisés pour parvenir à résoudre ce problème. Ces méthodes sont basées sur la programmation en nombres entiers (Wolsey, 1998) car les variables principales pour ce problème de génération de planification de menus sont des variables binaires. Le chapitre 4 a montré comment l'application a été développée avec un modèle client-serveur basé sur REST (Fielding, 2000) avec une application web ergonomique pour que les utilisateurs puissent utiliser l'application depuis n'importe quel périphérique. Le chapitre 5 présente les résultats obtenus par notre algorithme. L'algorithme trouve une solution optimale au problème de génération automatique de menus en moins de 3 secondes via l'application

web sur Internet. Une expérimentation en milieu réel a été réalisée dans des garderies pour vérifier l'appréciation d'un tel logiciel par des gérants de garderies en milieu familial. Cette expérimentation montre que l'algorithme permet aux gérants de garderies d'économiser du temps et de l'argent pour leur préparation de repas et que le logiciel est plutôt apprécié.

5.3 REVUE DES CONTRIBUTIONS

Les contributions apportées par ce travail de recherche sont multiples. La première est la modélisation mathématique du problème de génération automatique de menus comme un problème de génération d'emplois du temps avec des variables binaires. Ce problème est solvable par un algorithme de "branch-and-cut" en un temps très rapide (0.02 seconde). La seconde contribution est l'implémentation d'un logiciel web en 3 modules permettant d'utiliser le modèle d'optimisation pour générer les menus en fonction des préférences des utilisateurs. Le premier module permet d'extraire les ingrédients sur les sites des épiceries, le second module permet d'extraire les recettes et le troisième module est le module d'optimisation. Une interface graphique permet à ces trois modules d'être utilisés par un utilisateur non expérimenté.

Ces travaux de recherche vont être présentés à la conférence CoDIT'19 (Conference on Control, Decision and Information Technologies). L'article a été accepté et sera publié après la conférence.

5.4 TRAVAUX FUTURS

Bien que la solution proposée tout au long de ce mémoire apporte une réponse aux problèmes posés, la présente solution peut encore être améliorée et surtout plus testée. Ainsi l'application pourrait disposer d'encore plus d'épiceries dans sa liste pour extraire les prix. Il était difficile de pouvoir en avoir d'autre étant donnée le temps nécessaire pour développer un web crawler et le fait que deux épiceries différentes est une bonne base pour une preuve de concept. D'autant plus que les épiceries à rabais proposées par les gérants de garderies ne disposent pas de site web permettant de trouver les prix directement par Internet. Même s'il n'est pas possible de trouver les prix par internet, il serait peut-être possible de négocier une entente avec ces enseignes d'épiceries pour pouvoir disposer de leur base de données de produits sans passer par leur site web. De plus cette solution possède en base de données seulement cent recettes ce qui parait très faible et certaines recettes peuvent revenir souvent dans les planifications de menus. À force d'utilisation, il est possible de s'attendre à avoir des planifications de repas très similaires chaque mois. Pour résoudre ce problème, il faudrait pouvoir améliorer la base de données de recettes afin d'en avoir plus. Enfin, pour améliorer l'algorithme sur le plan nutritionnel, il serait important de collaborer avec une équipe de nutritionnistes afin de remplir la base de données avec seulement des recettes qui respectent les critères de nutrition pour un enfant. Ainsi avec un grand nombre de recettes de très bonne qualité et une exploration des prix sur les épiceries à rabais utilisées habituellement par les gérants de garderies en milieu familial, ce logiciel d'aide à la décision pourrait être utilisé dans beaucoup de garderies au Québec et améliorer les habitudes alimentaires de beaucoup d'enfants dès leur plus jeune âge.

BIBLIOGRAPHIE

Allahyari, M., Pouriyeh, S., Assefi, M., Safaei, S., Trippe, E. D., Gutierrez, J. B. et Kochut, K. (2017). A Brief Survey of Text Mining : Classification, Clustering and Extraction Techniques. *arXiv :1707.02919 [cs]*. de `http://arxiv.org/abs/1707.02919`

Aref, Walid (Plainsboro, N. B. D. P. N. (1998). de `http://www.freepatentsonline.com/5768423.html`

Ávila, T., Corberán, Á., Plana, I. et Sanchis, J. M. (2016). A branch-and-cut algorithm for the profitable windy rural postman problem. *European Journal of Operational Research*, *249*(3), 1092--1101.

Balas, E., Ceria, S. et Cornuéjols, G. (1996). Mixed 0-1 Programming by Lift-and-Project in a Branch-and-Cut Framework. *Management Science*, *42*(9), 1229--1246. de `http://pubsonline.informs.org/doi/abs/10.1287/mnsc.42.9.1229`

Balintfy, J. L. (1964). Menu planning by computer. *Communications of the ACM*, *7*(4), 255--259. de `http://portal.acm.org/citation.cfm?doid=364005.364087`

Bouhaï, N. et Saleh, I. (2017). *Internet des objets : évolutions et innovations*. Londres : Iste éditions. OCLC : 1014202255.

Chavez-Bosquez, O., Marchi, J. et Pozos-Parra, P. (2014). Nutritional Menu Planning : A Hybrid Approach and Preliminary Tests. In *Computing Science, 2014*, volume 82 93--104.

Chernov, N., Stoyan, Y. et Romanova, T. (2010). Mathematical model and efficient algorithms for object packing problem. *Computational Geometry*, *43*(5), 535--553. de `http://linkinghub.elsevier.com/retrieve/pii/S0925772109001576`

Chu, P. et Beasley, J. (1998). A Genetic Algorithm for the Multidimensional Knapsack Problem. *Journal of Heuristics*, *4*(1), 63--86. de `https://doi.org/10.1023/A:1009642405419`

Cook, W. J. (2015). *In Pursuit of the Traveling Salesman : Mathematics at the Limits of Computation*. de `https://doi.org/10.1515/9781400839599`

Curtis, S. (2003). The classification of greedy algorithms. *Science of Computer Programming*, *49*(1-3), 125--157. de http://linkinghub.elsevier.com/retrieve/pii/S0167642303000340

Dantzig, G. B., Orden, A. et Wolfe, P. (1955). The generalized simplex method for minimizing a linear form under linear inequality restraints. *Pacific J. Math.*, *5*(2), 183--195.

David Nadeau et Satoshi Sekine (2007). A survey of named entity recognition and classification. *Lingvisticae Investigationes*, *30*(1), 3--26. de https://benjamins.com/catalog/li.30.1.03nad

Deb, K. et Deb, K. (2014). Multi-objective Optimization. In E. K. Burke et G. Kendall (dir.), *Search Methodologies* 403--449. Boston, MA : Springer US

Dowsland, K. A. et Dowsland, W. B. (1992). Packing problems. *European Journal of Operational Research*, *56*(1), 2--14. de http://linkinghub.elsevier.com/retrieve/pii/037722179290288K

Elsweiler, D. et Harvey, M. (2015). Towards Automatic Meal Plan Recommendations for Balanced Nutrition. Vienna, Austria.

Felner, A. (2011). Position Paper : Dijkstra's Algorithm versus Uniform Cost Search or a Case Against Dijkstra's Algorithm. p. 5.

Fielding, R. T. (2000). REST architectural styles and the design of network-based software architectures. *Doctoral dissertation, University of California*.

Forrest, J., Ralphs, T., Vigerske, S., LouHafer, Kristjansson, B., Jpfasano, EdwinStraver, Lubin, M., Haroldo Gambini Santos, Rlougee et Saltzman, M. (2018). Coin-Or_cbc. de https://zenodo.org/record/1317565

Fukunaga, K. et Narendra, P. (1975). A Branch and Bound Algorithm for Computing k-Nearest Neighbors. *IEEE Transactions on Computers*, *C-24*(7), 750--753. de http://ieeexplore.ieee.org/document/1672890/

Gomory, R. E. (1958). Outline of an algorithm for integer solutions to linear programs. *Bulletin of the American Mathematical Society*, *64*(5), 275--279. de http://www.ams.org/journal-getitem?pii=S0002-9904-1958-10224-4

Gouvernement du Québec (2015). *Situation des centres de la petite enfance, des garderies et de la garde en milieu familial au Québec*. Rapport technique.

Griva, I., Nash, S. et Sofer, A. (2009). *Linear and nonlinear optimization* (2nd ed éd.). Philadelphia : Society for Industrial and Applied Mathematics. OCLC : 236082842.

Grötschel, M., Jünger, M. et Reinelt, G. (1984). A Cutting Plane Algorithm for the Linear Ordering Problem. *Operations Research*, *32*(6), 1195--1220. de `http://pubsonline.informs.org/doi/abs/10.1287/opre.32.6.1195`

Gupta, O. K. et Ravindran, A. (1985). Branch and Bound Experiments in Convex Nonlinear Integer Programming. *Management Science*, *31*(12), 1533--1546. de `http://pubsonline.informs.org/doi/abs/10.1287/mnsc.31.12.1533`

Gupta, V. et Lehal, G. S. (2009). A Survey of Text Mining Techniques and Applications. *Journal of Emerging Technologies in Web Intelligence*, *1*(1). de `http://ojs.academypublisher.com/index.php/jetwi/article/view/11`

Hassenzahl, M. et Tractinsky, N. (2006). User experience - a research agenda. *Behaviour & Information Technology*, *25*(2), 91--97. de `https://doi.org/10.1080/01449290500330331`

Hooker, J. et Barnes, K. (2015). Computer-Based Meal Planning to Satisfy Preferences for Taste, Cost, Nutrition, and Time Spent

Hsiao, J.-H. et Chang, H. (2010). SmartDiet : A personal diet consultant for healthy meal planning. Dans *2010 IEEE 23rd International Symposium on Computer-Based Medical Systems (CBMS)*, 421--425., Bentley, Australia. IEEE. de `http://ieeexplore.ieee.org/document/6042681/`

Kelleher, J. D., Mac Namee, B. et D'Arcy, A. (2015). *Fundamentals of machine learning for predictive data analytics : algorithms, worked examples, and case studies*. Cambridge, Massachusetts : The MIT Press.

Kim, I. et de Weck, O. (2005). Adaptive weighted-sum method for bi-objective optimization : Pareto front generation. *Structural and Multidisciplinary Optimization*, *29*(2), 149--158. de `http://link.springer.com/10.1007/s00158-004-0465-1`

Lawler, E. L. et Wood, D. E. (1966). Branch-and-Bound Methods : A Survey. *Operations Research*, *14*(4), 699--719. de http://pubsonline.informs.org/doi/abs/10.1287/opre.14.4.699

Leis, V., Kemper, A. et Neumann, T. (2013). The adaptive radix tree : ARTful indexing for main-memory databases. Dans *2013 IEEE 29th International Conference on Data Engineering (ICDE)*, 38--49., Brisbane, QLD. IEEE. de http://ieeexplore.ieee.org/document/6544812/

Marrero, M., Urbano, J., Sánchez-Cuadrado, S., Morato, J. et Gómez-Berbís, J. M. (2013). Named Entity Recognition : Fallacies, challenges and opportunities. *Computer Standards & Interfaces*, *35*(5), 482--489. de http://linkinghub.elsevier.com/retrieve/pii/S0920548912001080

Merrick, P., Allen, S. et Lapp, J. (2006). *XML remote procedure call (XML-RPC)*. Google Patents.

Mikowski, M. et Powell, J. (2013). *Single page web applications : JavaScript end-to-end*. Manning Publications Co.

Murphy, M. et Meeker, M. (2011). Top mobile internet trends. *KPCB Relationship Capital*.

Narendra et Fukunaga (1977). A Branch and Bound Algorithm for Feature Subset Selection. *IEEE Transactions on Computers*, *C-26*(9), 917--922. de http://ieeexplore.ieee.org/document/1674939/

Neveu, B., Trombettoni, G. et Araya, I. (2016). Node selection strategies in interval Branch and Bound algorithms. *Journal of Global Optimization*, *64*(2), 289--304. de http://link.springer.com/10.1007/s10898-015-0375-3

Padberg, M. et Rinaldi, G. (1987). Optimization of a 532-city symmetric traveling salesman problem by branch and cut. *Operations Research Letters*, *6*(1), 1--7. de http://linkinghub.elsevier.com/retrieve/pii/0167637787900022

Padberg, M. et Rinaldi, G. (1991). A Branch-and-Cut Algorithm for the Resolution of Large-Scale Symmetric Traveling Salesman Problems. *SIAM Review*, *33*(1), 60--100. de http://epubs.siam.org/doi/10.1137/1033004

Papadimitriou, C. H. et Steiglitz, K. (1998). *Combinatorial optimization : algorithms and complexity*. Mineola, N.Y : Dover Publications.

Russell, S. J., Norvig, P. et Davis, E. (2010). *Artificial intelligence : a modern approach* (3rd ed éd.). Prentice Hall series in artificial intelligence. Upper Saddle River : Prentice Hall.

Schuster, P. (2000). Taming combinatorial explosion. *Proceedings of the National Academy of Sciences*, *97*(14), 7678--7680. de http://www.pnas.org/cgi/doi/10.1073/pnas.150237097

Seljak, B. K. (2009). Computer-based dietary menu planning. *Journal of Food Composition and Analysis*, *22*(5), 414--420. de http://linkinghub.elsevier.com/retrieve/pii/S0889157509000829

Smith, V. E. (1959). Linear Programming Models for the Determination of Palatable Human Diets. *Journal of Farm Economics*, *41*(2), 272. de https://academic.oup.com/ajae/article-lookup/doi/10.2307/1235154

statcounter (2018). de http://gs.statcounter.com/os-market-share

Steglich, M. (2016). CMPLServer - An open source approach for distributed and grid optimisation. *AKWI - Anwendungen und Konzepte der Wirtschaftsinformatik*, *2016*, 10--22.

Tan, P.-N., Steinbach, M., Karpatne, A. et Kumar, V. (2019). *Introduction to data mining* (second edition éd.). NY NY : Pearson.

United States Departement of Agriculture, U. (2015). de https://whatscooking.fns.usda.gov/

Wächter, A. et Biegler, L. T. (2006). On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, *106*(1), 25--57. de https://doi.org/10.1007/s10107-004-0559-y

Wargo, J. M. (2012). *PhoneGap essentials : Building cross-platform mobile apps*. Addison-Wesley.

Wargo, J. M. (2015). *Apache cordova API cookbook*. Pearson Education.

We are social et Hootsuite (2018). de `https://digitalreport.wearesocial.com/`

Wolsey, L. A. (1998). *Integer programming*. Wiley-Interscience series in discrete mathematics and optimization. New York : Wiley.

Yujian, L. et Bo, L. (2007). A Normalized Levenshtein Distance Metric. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *29*(6), 1091--1095. de `http://ieeexplore.ieee.org/document/4160958/`

ANNEXE A

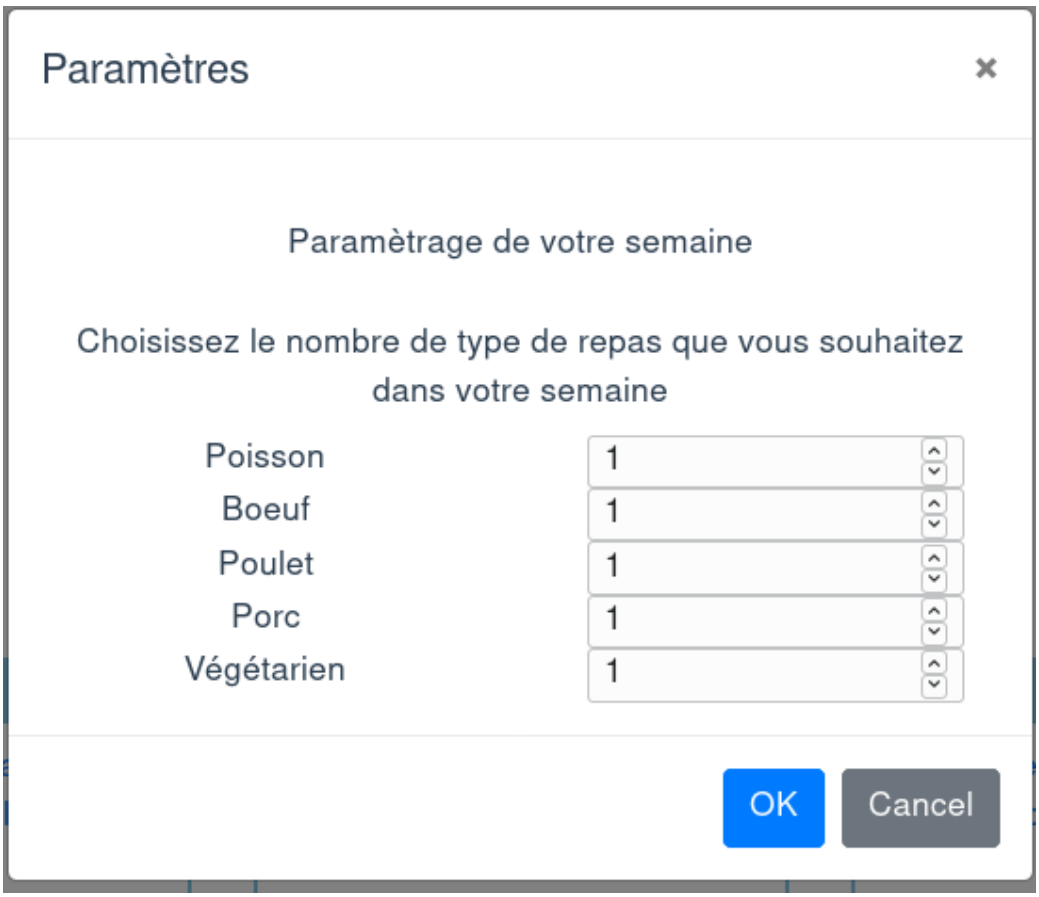
GUIDE UTILISATEUR

Guide d'utilisation site de génération de recettes **Glangine Geoffrey**
Laboratoire LIARA
Université du Québec à Chicoutimi 2018

Paramètres

Il est possible de paramètrer votre semaine en fonction de vos préférences. Pour cela, il suffit de cliquer sur le bouton paramètres en haut à droite de l'écran :

Vous arriverez donc sur ce menu qui vous demande de choisir le nombre de chaque type de repas que vous désirez :



Notez que vous pourrez avoir plus de repas d'un type si la somme de tous les types que vous indiquez est inférieure à 5 car l'application fournira toujours 5 repas. Par exemple, si vous indiquez :

— Poisson : 1

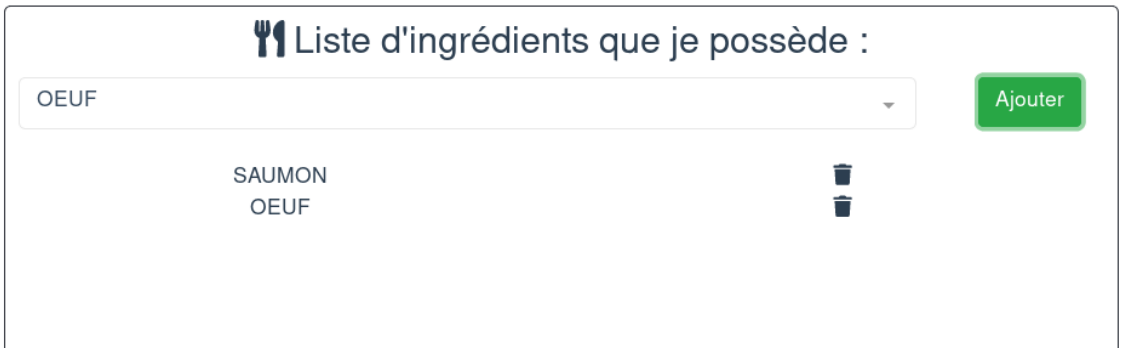
— Boeuf : 1
— Poulet : 0
— Porc : 0
— Végétarien : 0

Alors l'application devra trouver 3 repas en plus de ce que vous avez demandé et donc, vous pourrez avoir des repas de type Poulet, Porc ou Végétarien.

Si vous ne voulez absolument pas d'un type de repas, alors mettez le à zéro et mettez un autre type à deux pour toujours demander spécifiquement 5 repas.

Liste d'ingrédients possédées

Si vous possédez des ingrédients, vous pouvez les ajouter dans l'application avant de générer la liste des repas afin que l'application essaie de trouver des repas avec les ingrédients que vous possédez. Pour faire ceci, vous devrez vous rendre dans la petite section en bas à gauche :

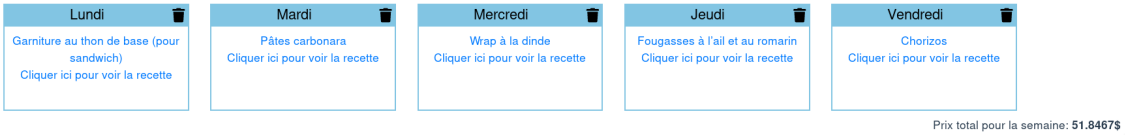


Dans cette section, vous pouvez ajouter vos ingrédients en cliquant sur la liste et tapant du texte pour rechercher. Une fois l'ingrédient sélectionné, vous pourrez cliquer sur ajouter. l'ingrédient va s'afficher dans la liste un peu plus bas. Si vous voulez le supprimer de la liste, il suffite de cliquer sur l'icône de poubelle.

Générer la liste de repas

Une fois les paramètres (facultatif) modifiés et les ingrédients entrés (facultatif) vous pourrez cliquer sur le bouton "Optimiser la liste des repas" au centre de l'écran afin de générer une liste de 5 repas pour la semaine.

Après avoir cliqué, il faudra attendre quelques secondes et les repas vont s'afficher au centre de l'écran.



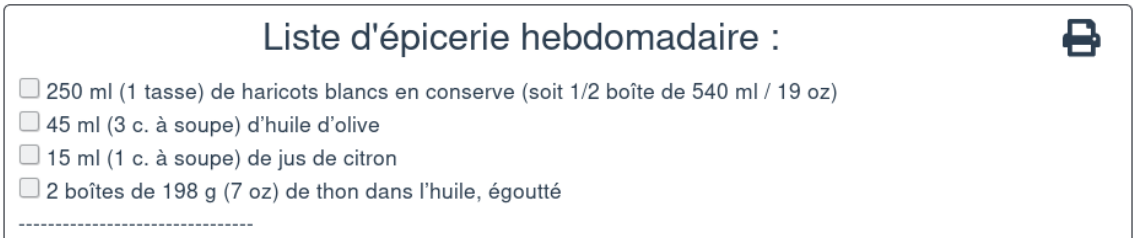
Le prix de la semaine pour l'épicerie sélectionnée est affiché en bas à droite des repas. A tout moment vous pouvez changer l'épicerie via la liste déroulante :

En changeant d'épicerie les repas vont peut être changer car l'application va toujours fournir les 5 repas les moins cher en fonction de l'épicerie choisie.

Finalement si les repas ne vous conviennent pas, vous pourrez cliquer sur l'icône de poubelle à côté des jours, pour supprimer la recette. Ensuite vous devrez cliquer à nouveau sur "optimiser la liste des repas" en l'application va vous fournir une nouvelle liste sans les repas que vous avez supprimé.

Liste d'épicerie

Une fois les repas générés et l'épicerie choisie, l'application génère une liste d'épicerie avec tous les ingrédients à acheter. Vous pouvez cliquer sur l'icône d'imprimante pour l'imprimer.



ANNEXE B

FRONT-END DE L'APPLICATION

Figure B.1 : Front-end de l'application

ANNEXE C

APPROBATION D'ÉTHIQUE

UQAC

Comité d'éthique de la recherche
Université du Québec à Chicoutimi

APPROBATION ÉTHIQUE

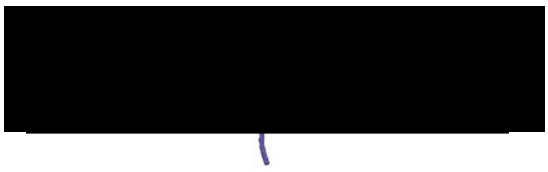
Dans le cadre de l'*Énoncé de politique des trois conseils : éthique de la recherche avec des êtres humains 2* (2014) et conformément au mandat qui lui a été confié par la résolution CAD-7163 du Conseil d'administration de l'Université du Québec à Chicoutimi, approuvant la *Politique d'éthique de la recherche avec des êtres humains* de l'UQAC, le Comité d'éthique de la recherche avec des êtres humains de l'Université du Québec à Chicoutimi, à l'unanimité, délivre la présente approbation éthique puisque le projet de recherche mentionné ci-dessous rencontre les exigences en matière éthique et remplit les conditions d'approbation dudit Comité.

Les membres jugent que ce projet rencontre les critères d'une recherche à risque minimal.

Responsable(s) du projet de recherche :	*Monsieur Geoffrey Glangine, Étudiant* *Maîtrise en informatique, UQAC*
Direction de recherche : *(telle qu'indiquée dans la demande d'approbation éthique)*	*Madame Sara Séguin, Professeure* *Département d'informatique et de mathématique*
Codirection de recherche : *(telle qu'indiquée dans la demande d'approbation éthique)*	*Monsieur Kévin Bouchard, Professeur* *Département d'informatique et de mathématique* *Monsieur Bruno Bouchard, Professeur* *Département d'informatique et de mathématique*
Cochercheur(s) :	*Monsieur Sébastien Gaboury, Professeur* *Département d'informatique et de mathématique*
Projet de recherche intitulé :	*Système expert d'aide à la décision pour l'élaboration des repas dans les garderies en milieu familial*
No référence du certificat :	*602.624.01*
Financement : *(tel qu'indiqué dans la demande d'approbation éthique)*	*N/A*

La présente est valide jusqu'au 30 juillet 2019.
Rapport de statut attendu pour le 31 mai 2019 (rapport final).
N.B. le rapport de statut est disponible à partir du lien suivant : *http://recherche.uqac.ca/rapport-de-statut/*

Date d'émission initiale de l'approbation : *20 septembre 2018*
Date(s) de renouvellement de l'approbation :



Tommy Chevrette,
Professeur et président du Comité d'éthique de la recherche avec des êtres humains de l'UQAC