

# Assemblage de vins sous contraintes

Philippe Vismara, Remi Coletta, Gilles Trombettoni

▶ **To cite this version:**

Philippe Vismara, Remi Coletta, Gilles Trombettoni. Assemblage de vins sous contraintes. 9. Journées Francophones de Programmation par Contraintes (JFPC 2013), Jun 2013, Aix-en-Provence, France. 356 p. hal-02745194

**HAL Id: hal-02745194**
**https://hal.inrae.fr/hal-02745194**

Submitted on 3 Jun 2020

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

# Assemblage de vins sous contraintes

**Philippe Vismara**[1,2] **Remi Coletta**[1] **Gilles Trombettoni**[1]

[1] LIRMM, UMR5506 Université Montpellier II - CNRS, Montpellier, France
[2] MISTEA, UMR729 Montpellier SupAgro - INRA, Montpellier, France
{Philippe.Vismara,Remi.Coletta,Gilles.Trombettoni}@lirmm.fr

**Résumé**

En œnologie, l'assemblage consiste à mélanger plusieurs cuvées afin d'obtenir un vin cible. Des progrès dans l'analyse des arômes permettent aujourd'hui de mesurer un ensemble de composés chimiques influençant le goût d'un vin. Il devient ainsi possible de concevoir un outil d'aide à la décision pour le problème suivant : étant donné un ensemble de vins cibles à produire, quelles quantités doit-on prélever dans chaque cuvée afin de produire des vins qui respectent des contraintes de concentration en arômes, de volumes, de degré alcoolique et de prix. L'article décrit la modélisation de ce problème sous forme d'une optimisation Min-Max (pour minimiser les écarts obtenus avec les concentrations souhaitées pour chaque critère aromatique) sous contraintes numériques linéaires et quadratiques, et son traitement efficace avec le *branch and bound* à intervalles de `Ibex`.

**Abstract**

Assemblage consists in blending base wines in order to create a target wine. Recent advances in aroma analysis allow us to measure chemical compounds impacting the taste of wines. This chemical analysis makes it possible to design a decision tool for the following problem : given a set of target wines, determine which volumes must be extracted in each cuvée to produce wines that satisfy constraints on aroma concentration, volumes, alcohol levels and price. This paper describes the modeling of wine assemblage as a non linear constrained Min-Max problem (minimizing the gap to the desired concentrations for every aromatic criterion) efficiently handled by the `Ibex` interval branch and bound.

## 1 Introduction

L'assemblage est l'art de mélanger subtilement des vins de différentes parcelles et/ou de différents cépages, chacun apportant ses propres spécificités. Le choix des assemblages élaborés est généralement effectué par des œnologues. Si la qualité de leur travail est inégalable, une limite forte dans le nombre de dégustations quotidiennes tient à la saturation des goûts. C'est pourquoi la société Nyseos, qui nous a soumis ce problème, propose des outils d'analyse chimique permettant de soulager l'œnologue d'une partie du travail de dégustation. Ces outils analysent les arômes d'un vin en mesurant un ensemble de composés chimiques qui influencent le goût du vin [6]. Il devient ainsi possible de concevoir un outil d'aide à la décision pour le problème suivant : étant donné un ensemble de vins cibles à produire, quelles quantités doit-on prélever dans chaque cuvée pour produire des vins qui respectent des contraintes de concentration en arômes, de volumes, de degré alcoolique et de prix.

Moore et Griffin ont montré que la concentration en arômes d'un mélange de vins obéit à des équations linéaires [10]. Cependant, d'autres impératifs d'assemblage peuvent imposer des contraintes non linéaires. C'est notamment le cas pour la prédiction de la couleur d'un vin cible en fonction de la couleur des vins de base. Même si la modélisation de cette contrainte n'est pas encore aboutie, nous savons déjà qu'elle ne sera pas linéaire. Ce résultat est d'ailleurs confirmé par d'autres études [7]. Une autre difficulté provient du fait qu'il n'est pas possible de transvaser une trop faible quantité de vin entre deux cuves à cause du volume inévitablement perdu dans les tuyaux et du coût des manipulations. Comme nous le montrons dans l'article, cela conduit à définir une contrainte disjonctive qui peut être modélisée par des variables booléennes ou des contraintes non linéaires.

La principale étude sur l'aide à l'assemblage de vin a été publiée dans [7]. Les auteurs utilisent un réseau de neurones pour déterminer les quantités devant être extraites de chaque cuve de base afin de concevoir un vin respectant des critères aromatiques prédéfinis. Dans cette étude, les concentrations en arômes n'ont pas été mesurées chimiquement mais évaluées gusta-

tivement par un panel de personnes (étudiants). Le réseau de neurones est utilisé pour réaliser une optimisation multicritères afin d'ajuster chaque arôme. Il est ainsi difficile de comparer ce travail avec notre démarche dans la mesure où nous réalisons une optimisation mono-critère. Par ailleurs, [7] ne fournit aucune indication de performance (temps CPU).

Nous proposons dans cet article une modélisation mathématique du problème d'assemblage de vin. Le problème se modélise comme un programme non linéaire mixte (discret et continu) que nous transformons en CSP numérique non linéaire traité par des méthodes à intervalles. Pour minimiser, dans chaque vin cible, l'écart entre les concentrations aromatiques souhaitées et celles obtenues, nous modélisons ce problème d'optimisation à l'aide d'un Min-Max que nous parvenons à débarrasser de ses opérateurs valeur absolue et max. Le modèle a été particulièrement étudié pour être traité efficacement par des méthodes à intervalles. Nous montrons des premiers résultats très satisfaisants obtenus sur deux assemblages réels.

## 2 Le problème de l'assemblage de vins

Les éléments permettant de décrire un problème d'assemblage sont illustrés à la figure 1.

On dispose d'un ensemble de cuvées de base, appelées *bases*, numérotées de 1 à $\mathcal{B}$. On connaît le volume $\text{vol}_b$ de chaque base $b \in 1..\mathcal{B}$. Pour des raisons matérielles, on ne peut pas toujours vider entièrement un fond de cuve et l'on doit laisser un fond minimum noté $s_b^-$ (on a : $0 \leq s_b^- \leq \text{vol}_b$).

Toutes les bases sont analysées afin de mesurer leur teneur en composés aromatiques. Ces composés sont numérotés de 1 à $\mathcal{A}$ et on notera $c_{b,a}$ la concentration en arôme $a$ dans la base $b$.

Un outil d'aide à l'assemblage doit permettre d'étudier globalement l'assemblage de plusieurs vins en puisant dans le même ensemble de bases. On dispose donc d'un ensemble de vins cibles numérotés de 1 à $\mathcal{W}$. Pour chaque vin $w$, on souhaite produire un volume de vin noté $\hat{\text{vol}}_w$. Ce volume peut cependant fluctuer jusqu'à une borne maximale (resp. minimale) notée $\text{vol}_w^+$ (resp. $\text{vol}_w^-$). Ces bornes permettent de ne pas produire un volume trop important pour être vendu ou au contraire de garantir un volume qui a été commandé. Au final, le volume $V_w$ du vin $w$ devra être le plus proche possible de $\hat{\text{vol}}_w$ tout en respectant :

$$\forall w \in 1..\mathcal{W}, \quad \text{vol}_w^- \leq V_w \leq \text{vol}_w^+ \tag{1}$$

Chaque vin cible $w$ étant obtenu en assemblant du vin issu de plusieurs bases, on notera $V_{w,b}$ le volume de vin $w$ provenant de la base $b$. Le volume total du

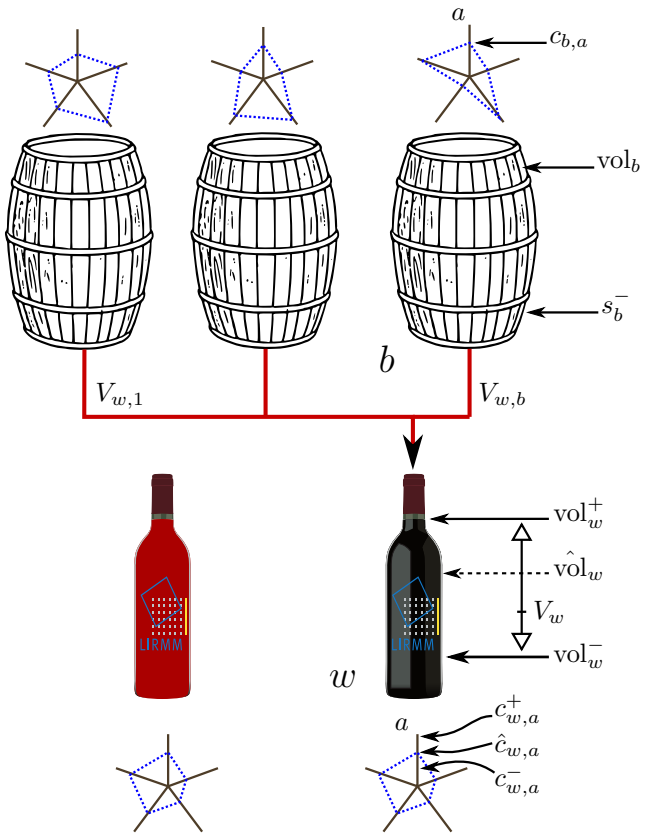



FIGURE 1 – L'assemblage de vins

vin $w$ est donc la somme de ces volumes partiels :

$$\forall w \in 1..\mathcal{W}, \quad V_w = \sum_{b=1}^{\mathcal{B}} V_{w,b} \tag{2}$$

On doit veiller à ce que la somme des volumes soutirés dans une même base $b$ respecte les contraintes de volume autorisé pour le surplus restant en fond de cuve. On pose la contrainte :

$$\forall b \in 1..\mathcal{B}, \quad s_b^- \leq \text{vol}_b - \sum_{w=1}^{\mathcal{W}} V_{w,b} \tag{3}$$

D'un point de vue pratique, il est impossible de soutirer un trop faible volume d'une cuve, notamment à cause du résidu perdu dans les tuyaux. Si $\delta_V$ désigne ce volume minimal, on pose la contrainte *disjonctive* suivante :

$$\forall w \in 1..\mathcal{W}, \forall b \in 1..\mathcal{B}, \ (V_{w,b} = 0) \ \vee \ (\delta_V \leq V_{w,b}) \tag{4}$$

En plus du volume souhaité, chaque vin cible est décrit par un profil aromatique recherché à l'intérieur de plages bien définies. Pour chaque vin cible $w$, on notera $\hat{c}_{w,a}$ la concentration souhaitée en arôme $a$. La

concentration minimale (resp. maximale) tolérée sera notée $c^-_{w,a}$ (resp. $c^+_{w,a}$).

On peut donc exprimer le fait que la concentration $C_{w,a}$ en arôme $a$ du vin $w$ doit être comprise entre ces deux bornes : pour tout $w \in 1..\mathcal{W}$ et pour tout $a \in 1..\mathcal{A}$, on a :

$$c^-_{w,a} \;\leq\; \frac{1}{V_w} \sum_{b=1}^{\mathcal{B}} (V_{w,b} \,.\, c_{b,a}) \;\leq\; c^+_{w,a} \qquad (5)$$

De manière analogue, on peut imposer des contraintes sur la concentration en alcool ou le prix au litre des vins cibles. Ces caractéristiques peuvent être considérées comme deux arômes supplémentaires.

## 3 Un modèle MINLP

Nous pouvons modéliser le problème de l'assemblage de vin comme un programme non linéaire mixte (MINLP). Nous verrons à la section 4 comment transformer ce MINLP en un CSP numérique (NCSP) traité par des méthodes à intervalles. De ce fait, nous allons directement traduire les contraintes de bornes par des domaines bornés, c'est-à-dire des intervalles.

### 3.1 Variables

Pour tenir compte des contraintes sur le volume minimum pouvant être soutiré (cf. (4)), nous créons pour chaque vin $w \in 1..\mathcal{W}$ et chaque base $b \in 1..\mathcal{B}$ :

- une variable $P_{w,b}$ de domaine 0/1 et
- une variable $V'_{w,b}$ de domaine $D(V'_{w,b}) = [\delta_V, \min(\mathrm{vol}^+_w, \mathrm{vol}_b)]$ représentant le volume transvasé de la base $b$ vers le vin cible $w$.
  (nous avons : $V_{w,b} \equiv P_{w,b} \,.\, V'_{w,b}$. L'introduction de la variable 0/1 évite la définition explicite de la contrainte disjonctive (4).)

Pour chaque vin $w \in 1..\mathcal{W}$, nous définissons aussi une variable $V_w$ de domaine $[\mathrm{vol}^-_w, \mathrm{vol}^+_w]$ représentant son volume (cf. (1)).

### 3.2 Contraintes

Le système de contraintes de notre MINLP est ainsi défini :

- la contrainte de channeling (2) devient :

$$\forall w,\; V_w - \sum_{b=1}^{\mathcal{B}} (P_{w,b} \,.\, V'_{w,b}) = 0 \qquad (2.\mathrm{i})$$

- la contrainte (3) sur les surplus est similaire :

$$\forall b \in 1..\mathcal{B}, s^-_b \leq \mathrm{vol}_b - \sum_{w=1}^{\mathcal{W}} (P_{w,b} \,.\, V'_{w,b}) \qquad (3.\mathrm{i})$$

- pour améliorer les performances, nous avons introduit une contrainte redondante par rapport à (3.i). Cette contrainte garantit simplement que le volume total des vins cibles est inférieur au volume total des cuves de base :

$$\sum_{w=1}^{\mathcal{W}} V_w \leq \sum_{b=1}^{\mathcal{B}} vol_b \qquad (6)$$

La règle (5) concernant chaque concentration en arômes est décomposée en deux contraintes où chaque partie a été multipliée par le volume positif $V_w$. On a donc, $\forall w \in 1..\mathcal{W}$ et $\forall a \in 1..\mathcal{A}$ :

pour la borne inférieure,

$$0 \leq \sum_{b=1}^{\mathcal{B}} V_{w,b} \,.\, (c_{b,a} - c^-_{w,a})$$

d'où :

$$0 \leq \sum_{b=1}^{\mathcal{B}} P_{w,b} \,.\, V'_{w,b} \,.\, (c_{b,a} - c^-_{w,a}) \qquad (5.\mathrm{i}\text{ -})$$

et pour la borne supérieure :

$$0 \leq \sum_{b=1}^{\mathcal{B}} V_{w,b} \,.\, (c^+_{w,a} - c_{b,a})$$

d'où :

$$0 \leq \sum_{b=1}^{\mathcal{B}} P_{w,b} \,.\, V'_{w,b} \,.\, (c^+_{w,a} - c_{b,a}) \qquad (5.\mathrm{i}\text{ +})$$

### 3.3 Un Min-Max pour optimiser la qualité des vins

Dans l'assemblage de vins, le principal critère de qualité concerne les profils aromatiques. Cependant, il est tout à fait possible d'étendre les définitions qui vont suivre au degré alcoolique ou au prix des vins cibles.

Nous avons vu que chaque vin cible $w$ est caractérisé par un ensemble de concentrations souhaitées $\hat{c}_{w,a}$ pour chaque arôme $a$ mesuré. Pour optimiser la qualité d'un vin $w$, nous pouvons minimiser la somme des différences entre chaque concentration $\hat{c}_{w,a}$ souhaitée et la concentration $C_{w,a}$ obtenue (cf. (5)). De plus, nous voulons minimiser l'erreur maximale sur l'ensemble des vins cibles :

$$\max_{w \in 1..\mathcal{W}} \Omega_w (\lambda_{vol_w} \,.\, e_{vol_w} + \sum_{a \in 1..\mathcal{A}} (\lambda_{w,a} \,.\, e_{w,a})) \qquad (7)$$

où :

- $\Omega_w$ est un paramètre indiquant l'importance du vin $w$ (on suppose $\Omega_w \in [0,1]$). Il est ainsi possible d'accorder plus d'importance à certains vins cibles, notamment ceux de grande qualité.

– $e_{w,a}$ représente l'écart entre la concentration souhaitée $\hat{c}_{w,a}$ et celle obtenue $C_{w,a}$.
– $e_{vol_w}$ est l'écart entre le volume $\hat{\text{vol}}_w$ désiré pour le vin $w$ et le volume obtenu $V_w$.
– $\lambda_{w,a} \in [0,1]$ définit le poids de l'arôme $a$ pour le vin $w$ et $\lambda_{vol_w} \in [0,1]$ le poids accordé à son volume. Pour un vin $w$ donné, on suppose que $\lambda_{vol_w} + \sum_{a \in 1..\mathcal{A}} \lambda_{w,a} = 1$.

Tous les paramètres sont mesurés avec un certain taux d'erreur. $\varepsilon_a$ représente la marge d'erreur possible sur une mesure de l'arôme $a$. Nous cherchons donc à minimiser l'écart entre $\hat{c}_{w,a}$ et $C_{w,a}$ dans la limite de la marge d'erreur $\varepsilon_a$. En d'autres termes, si l'écart entre la concentration souhaitée et celle obtenue est inférieur au taux d'erreur, il sera considéré comme nul dans la fonction objectif. La variable $e_{w,a}$ décrit donc l'erreur de concentration normalisée pour l'arôme $a$ dans le vin $w$ :

$$\boldsymbol{e_{w,a}} = \max(\frac{|C_{w,a} - \hat{c}_{w,a}|}{\hat{c}_{w,a}} - \varepsilon_a, 0) \qquad (8)$$

Nous pouvons décrire par une expression similaire l'écart $e_{vol_w}$ entre le volume $V_w$ obtenu pour le vin cible $w$ et le volume souhaité $\hat{\text{vol}}_w$ :

$$\boldsymbol{e_{vol_w}} = \max(\frac{\hat{\text{vol}}_w - V_w}{\hat{\text{vol}}_w} - \varepsilon_{vol}, 0) \qquad (9)$$

Par rapport à la formule précédente, la suppression de la valeur absolue signifie simplement que l'erreur n'est pas prise en compte si le volume $V_w$ obtenu est compris entre le volume souhaité $\hat{\text{vol}}_w$ et le volume maximum autorisé $\text{vol}_w^+$. Cette règle est illustrée par la figure 2.

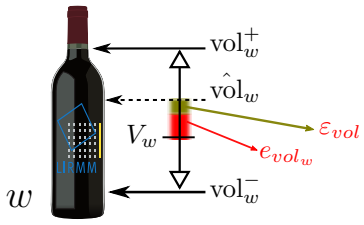



FIGURE 2 – Visualisation de l'écart $e_{vol_w}$ sur le volume

De façon classique pour un problème de Min-Max, nous définissons une variable $E \in [0, +\infty]$ à minimiser, en respectant les contraintes suivantes :

$$\forall w \in 1..\mathcal{W}, \quad \Omega_w(e_{vol_w}.\lambda_{vol_w} + \sum_{a \in 1..\mathcal{A}} (e_{w,a} \,.\, \lambda_{w,a})) \leq E \qquad (10)$$

### Suppression des opérateurs max et valeur absolue

Afin d'améliorer les performances, nous avons cherché à supprimer les opérateurs max et valeur absolue du modèle précédent. L'opérateur max peut être défini par :

$$e = max(x, y) \;\equiv\; e \geq x \wedge e \geq y \wedge (e = x \vee e = y).$$

De plus, si la quantité $e$ doit être minimisée pour une raison quelconque, la dernière conjonction peut être supprimée, ce qui simplifie la définition de l'opérateur max. Nous pouvons appliquer cette simplification à l'équation (8). Puisque chaque $\lambda_{w,a}$ est positif, minimiser $E$ revient à minimiser chaque variable $e_{w,a}$. On a donc, $\forall w \in 1..\mathcal{W}, \forall a \in 1..\mathcal{A}$ :

$$((e_{w,a} + \varepsilon_a)\,\hat{c}_{w,a} \geq |C_{w,a} - \hat{c}_{w,a}|) \;\wedge\; (e_{w,a} \geq 0) \qquad (11)$$

La même simplification peut s'appliquer à (9), ce qui donne, $\forall w \in 1..\mathcal{W}$ :

$$((e_{vol_w} + \varepsilon_{vol})\,\hat{\text{vol}}_w \geq (\hat{\text{vol}}_w - V_w)) \wedge\; (e_{vol_w} \geq 0) \qquad (12)$$

Nous pouvons aussi supprimer l'opérateur *valeur absolue* qui s'exprime sous la forme d'un *max* :

$$\begin{aligned} e = |x| \quad &\equiv \quad e = max(x, -x) \\ &\equiv \quad e \geq x \;\wedge\; e \geq -x \;\wedge\; (e = x \vee e = -x) \end{aligned}$$

Comme précédemment, si la quantité $e$ doit être minimale pour une raison quelconque, nous pouvons remplacer l'opérateur *valeur absolue* par deux inégalités. Cette simplification peut s'appliquer à l'équation (11). Rappelons en effet que chaque variable $e_{w,a}$ doit être minimisée et notons que $\hat{c}_{w,a}$ est toujours positif. Il s'ensuit, $\forall w \in 1..\mathcal{W}, \forall a \in 1..\mathcal{A}$, :

$$(e_{w,a} + \varepsilon_a)\,\hat{c}_{w,a} \geq \frac{1}{V_w}.\sum_{b=1}^{\mathcal{B}}(P_{w,b}.V'_{w,b}.c_{b,a}) - \hat{c}_{w,a}$$

$$(e_{w,a} + \varepsilon_a)\,\hat{c}_{w,a} \geq -\frac{1}{V_w}.\sum_{b=1}^{\mathcal{B}}(P_{w,b}.V'_{w,b}.c_{b,a}) + \hat{c}_{w,a}$$

En multipliant chaque côté de ces inégalités par le volume $V_w$ qui est positif, on obtient finalement trois catégories de contraintes : $\forall w \in 1..\mathcal{W}, \forall a \in 1..\mathcal{A}$,

$$e_{w,a} \geq 0 \qquad (13)$$

$$V_w.(e_{w,a} + \varepsilon_a + 1)\,.\,\hat{c}_{w,a} - \sum_{b=1}^{\mathcal{B}}(P_{w,b}.V'_{w,b}.c_{b,a}) \geq 0 \qquad (14)$$

$$V_w.(e_{w,a} + \varepsilon_a - 1)\,.\,\hat{c}_{w,a} + \sum_{b=1}^{\mathcal{B}}(P_{w,b}.V'_{w,b}.c_{b,a}) \geq 0 \qquad (15)$$

En conséquence, nous avons réussi à supprimer de notre modèle initial tous les opérateurs max et valeur absolue. Bien que le résolveur `Ibex` que nous allons utiliser sache gérer ces opérateurs, les performances seront améliorées et ce modèle simplifié peut être mis en œuvre dans d'autres résolveurs.

### 3.4 Synthèse : le modèle MINLP final

En plus des variables $P_{w,b}$, $V'_{w,b}$ et $V_w$ définies à la section 3.1, nous avons introduit de nouvelles variables pour le Min-Max : une variable $E \in [0, +\infty]$, $\mathcal{W}.\mathcal{A}$ variables $e_{w,a} \in [0, 1]$ (qui englobent la contrainte unaire (13)) et $\mathcal{W}$ variables $e_{vol_w} \in [0, 1]$ qui englobent la contrainte unaire (12).

En plus des contraintes (2.i), (3.i), (6), (5.i-), (5.i+) définies à la section 3.2, nous ajoutons de nouvelles contraintes pour le Min-Max : (10), (12), (14), (15). La fonction objectif consiste simplement à minimiser la valeur de la variable $E$.

## 4 Résolution du MINLP par un B&B à intervalles

Le MINLP détaillé précédemment peut être résolu par n'importe quel résolveur de MINLP [12, 2]. Cependant, aucun d'entre eux n'est *rigoureux*. Cela signifie qu'il peuvent ne pas trouver la meilleure solution à cause d'erreurs d'arrondi de l'arithmétique en virgule flottante à certaines étapes de la résolution. À notre connaissance, un seul *branch and bound* (B & B) rigoureux à intervalles, nommé `IBBA` [11], est doté d'un mécanisme simple de manipulation des variables intégrales. Ainsi, comme :

- notre modèle d'assemblage de vins ne contient qu'un seul type de variables 0/1,
- le résolveur à intervalles `Ibex` [5, 4] fournit un B & B très efficace nommé `IbexOpt` [13], et
- les auteurs ont une bonne connaissance de `Ibex`,

nous avons décidé de coder le MINLP sous la forme d'un NCSP, c'est-à-dire un système standard de contraintes continues non linéaires sur des nombres réels. Pour cela, les variables 0/1 sont codées par des variables réelles $P'_{w,b}$ de domaine $[0, 1]$. Afin de garantir que ces variables ne pourront prendre que la valeur 0 ou 1, nous ajoutons simplement la contrainte quadratique suivante :

$$\forall w \in 1..\mathcal{W} \text{ et } \forall b \in 1..\mathcal{B},\ 4(P_{w,b} - \frac{1}{2})^2 = 1 \qquad (16)$$

Cela signifie que les contraintes disjonctives initiales, qui sont à l'origine des contraintes mixtes dans le MINLP, sont gérées par des contraintes continues quadratiques dans le NCSP.

### 4.1 Optimisation globale sous contraintes par un B&B sur intervalles

Un problème d'optimisation globale sous contraintes est défini de la manière suivante.

**Definition 1 (Optimisation )**
*Considérons un vecteur de variables* $x = (x_1, ..., x_i, ...x_n)$ *de domaine* $[x] = [x_1] \times \cdots \times [x_n]$, *une fonction continue* $f : \mathbb{R}^n \to \mathbb{R}$, *des fonctions* $g : \mathbb{R}^n \to \mathbb{R}^m$ *et* $h : \mathbb{R}^n \to \mathbb{R}^p$. *(Nous notons* $g = (g_1, ..., g_m)$ *et* $h = (h_1, ..., h_j, ..., h_p)$.*)*

*Soit le système* $S = (f, g, h, [x])$, *le problème d'optimisation globale sous contraintes consiste à trouver*

$$\min_{x \in [x]} f(x) \quad \textit{soumis à} \quad g(x) \leq 0 \wedge h(x) = 0.$$

*$f$ est appelée fonction objectif; $g$ et $h$ sont des contraintes d'inégalité et d'égalité respectivement.*

Notre optimiseur global (sous contraintes) `IbexOpt` [13] calcule un vecteur de nombres à virgule flottante $x$ $\epsilon$-minimisant [1] :

$$f(x) \ s.t. \ g(x) \leq 0 \wedge (-\epsilon_{eq} \leq h(x) \leq +\epsilon_{eq}).$$

Notons que les égalités $h_j(x) = 0$ sont relâchées par des équations "épaisses" $h_j(x) \in [-\epsilon_{eq}, +\epsilon_{eq}]$, soit deux inégalités : $-\epsilon_{eq} \leq h_j(x) \leq +\epsilon_{eq}$. `IbexOpt` garantit l'optimum global du système relâché, quoique $\epsilon_{eq}$ puisse être souvent choisi arbitrairement petit. (La plupart des optimiseurs globaux comme `Baron` [12] ou `Couenne` [2] ne peuvent fournir aucune garantie.)

Dans notre problème de mélange de vins, nous prenons une constante $\epsilon^1_{eq}$ égale à `1e-4` dans les contraintes d'égalité (16). Un autre $\epsilon^2_{eq}$ est fixé à `1e-1` dans les contraintes d'égalité (2.i). Cette incertitude correspond à un décilitre, soit moins que 0.1% des volumes des vins cibles (au moins 500 litres).

Cela signifie que les volumes sont calculés avec une approximation significativement meilleure que les erreurs inéluctables faites pendant le mélange effectif, c'est-à-dire les erreurs induites par les mesures et par la perte de matière résiduelle pendant le transvasement d'un vin d'une cuve de base à la cible [2].

### 4.2 Opérateurs algorithmiques d'Ibex

`IbexOpt` est implanté en `Ibex` (Interval Based EXplorer) et enrichit cette bibliothèque en `C++` dédiée à la "résolution" par intervalles [5].

`IbexOpt` suit un schéma de type *Branch & Contract & Bound* sur les intervalles [13] . Le processus débute avec une boîte initiale $[x]$ subdivisée récursivement par l'opération de branchement. L'arbre est parcouru en meilleur d'abord, en sélectionnant à chaque itération

1. $\epsilon$-minimiser $f(x)$ signifie minimiser $f(x)$ avec une précision $\epsilon$, c'est-à-dire trouver $x$ tel que pour tout $y$, nous avons $f(y) \geq f(x) - \epsilon$.

2. Notons qu'une prochaine version de `Ibex` permettra de rigoureusement $\epsilon$-optimiser une fonction objectif sous des contraintes d'égalités vraies (non relaxées).

la boîte de coût inférieur le plus petit. Les opérations suivantes s'enchaînent sur chaque boîte (nœud) traité :
**Brancher :** une variable $x_i$ est choisie et son intervalle $[x_i]$ est découpé en deux sous-intervalles. Les deux sous-boîtes produites subissent les deux opérations suivantes.
**Contracter :** un processus de filtrage contracte la boîte étudiée (c'est-à-dire améliore les bornes de ses intervalles) sans perte de solutions.
**Borner :** l'amélioration de la borne inférieure est similiaire à une contraction (en considérant une variable correspondant au coût de l'objectif) : la borne inférieure garantit qu'aucune solution ne se trouve plus bas. L'amélioration de la borne supérieure revient à trouver un point réalisable de coût le plus bas possible, de manière à couper des branches de l'arbre de recherche de coût supérieur.

Le processus débute avec une boîte initiale $[x]$ et s'arrête quand la différence entre la borne supérieure et la borne inférieure atteint une précision donnée ($\epsilon$-optimisation) ou encore quand tous les nœuds explorés atteignent une taille inférieure à une précision donnée en entrée.

A chaque nœud du B&B, `IbexOpt` appelle des opérateurs efficaces pour réduire l'espace de recherche et améliorer les bornes inférieure et supérieure de la fonction objectif :

- L'algorithme de propagation de contraintes de référence (pour le continu) `HC4` [3, 9] est d'abord utilisé pour contracter la boîte traitée.
- Ensuite, l'opérateur algorithmique `X-Newton` utilise une forme de Taylor sur intervalles spécifique pour convexifier l'espace de recherche. Il contracte la boîte et améliore la borne inférieure [1].
- Deux algorithmes cherchent enfin à améliorer la borne supérieure en extrayant d'abord de manière heuristique une région *intérieure* (ou entièrement réalisable) ne contenant que des points - solutions - satisfaisant les contraintes. D'où l'intérêt de relaxer légèrement les équations. En simplifiant quelque peu, l'algorithme `InHC4` est un algorithme dual de `HC4` ; `InnerPolytope` est un algorithme dual de `X-Newton`.

La stratégie d'optimisation par défaut de `Ibex` utilise comme heuristique de branchement (bissection) la variante `SmearSumRel` de l'heuristique de Kearfott basée sur la fonction *Smear* [8]. Les heuristiques de branchement `SmearSumRel` et `SmearMaxRel` sont décrites dans [13].

# 5 Expérimentations

Nous avons modélisé et résolu plusieurs instances d'assemblage de vins. Nous verrons à la section 5.6 que notre démarche a également été validée par une séance de dégustations réelle.

Pour réaliser l'optimisation, nous avons imposé une précision $\epsilon =$`1e-4` sur le coût (objectif). Cette précision est très inférieure au taux d'erreur $\epsilon_a$ des outils chimiques de mesure des concentrations en arômes. La même précision est imposée sur la taille des solutions (boîte) : au dessous de cette taille, une boîte n'est pas étudiée (découpée) par le B & B à intervalles.

## 5.1 Les instances testées

Nous avons modélisé trois instances du problème d'assemblage des vins. La première (`WineBlending0`) est une petite instance artificielle qui nous a permis de régler le modèle MINLP/NCSP vu plus haut jusqu'à l'obtention d'un temps de réponse suffisamment rapide. L'instance `WineBlending0` contient 21 variables et est résolue en 0.18 secondes avec seulement 6 branchements par la stratégie par défaut, indépendamment de la précision demandée (`1e-4` ou `1e-8`).

### Instance réelle 1

La seconde instance (`WineBlending1`) est une instance réelle fournie par la société Nyseos. Il s'agit de produire $\mathcal{W} = 2$ vins cibles à partir de $\mathcal{B} = 7$ vins de base, en prenant en compte $\mathcal{A} = 11$ arômes.

Le problème Min-Max (section 3.4), contient 55 variables et 116 contraintes :

- 2 contraintes de channeling sur les volumes,
- 7 contraintes de surplus pour les bases,
- 44 contraintes de concentration en arômes,
- 49 contraintes liées au Min-Max,
- 14 contraintes quadratiques modélisant les contraintes disjonctives sur les volumes réalistes.

Cette instance forme un système relativement important pour un optimiseur global sous contraintes déterministe (exact).

### Instance réelle 2

La troisième instance (`WineBlending2`) décrit l'assemblage de $\mathcal{W} = 3$ vins cibles à partir de $\mathcal{B} = 6$ bases, en tenant compte de $\mathcal{A} = 7$ arômes.

Le problème Min-Max contient 64 variables et 118 contraintes :

- 3 contraintes de channeling sur les volumes,
- 6 contraintes de surplus pour les bases,
- 42 contraintes de concentration en arômes,
- 49 contraintes liées au Min-Max,
- 18 contraintes quadratiques modélisant les contraintes disjonctives sur les volumes réalistes..

### 5.2 Résultats obtenus avec l'optimiseur par défaut

Tous les résultats présentés dans cet article ont été obtenus sur un ordinateur portable MacBook de 2010 (Intel Core 2 Duo à 2.4 GHz).
Nous avons tout d'abord utilisé l'optimisateur par défaut d'Ibex avec une erreur de `1e-4` et un temps-limite de 5 minutes.

L'optimiseur atteint le temps-limite pour `WineBlending1` et `WineBlending2` tout en retournant des solutions relativement bonnes :

- pour `WineBlending1`, après 5 mn de calcul et 7894 nœuds, une erreur de 0.002 est obtenue sur l'objectif (c.-à-d., l'erreur maximum $E$).
- pour `WineBlending2`, en 5 mn et 8520 nœuds, une erreur de 0.0054 est obtenue.

### 5.3 Analyse algorithmique

Les résultats précédents ont été obtenus en utilisant la stratégie par défaut disponible dans `Ibex` et qui est brièvement décrite à la section 4.2.

Nous avons ensuite déterminé quelles options par défaut pouvaient avoir une influence sur la performance. Cette analyse a été très révélatrice.

Concernant l'aspect contraction (filtrage), l'opérateur de linéarisation sur intervalles `X-Newton` est étonnamment contre-productif. À l'inverse, toutes les contractions sont réalisées par l'opérateur `HC4` de programmation par contraintes. Dans la mesure où le modèle d'assemblage de vins est majoritairement linéaire, ce résultat n'est pas intuitif. Nous n'avons pas encore suffisamment d'éléments pour en comprendre la cause.

A l'opposé, nous avons fait une observation plus intuitive sur les régions internes extraites dans la phase d'amélioration de la borne supérieure. L'opérateur `InHC4`, issu des principes de la programmation par contraintes, ne semble pas utile. À l'inverse, l'algorithme `InnerPolytope` (dual de `X-Newton`) s'avère crucial pour extraire un polytope intérieur de l'espace réalisable. Sans cette fonctionnalité, nous ne pourrions pas obtenir de réponse de l'optimiseur.

La stratégie de branchement (bissection) est le second critère ayant une influence majeure sur la performance. Nous avons observé que les heuristiques élémentaires de bissection fournies par `Ibex` sont inefficaces ou peu performantes. L'heuristique `largestFirst`, qui sélectionne une variable de plus grande largeur, empêche le B&B de trouver un point réalisable pendant la recherche arborescente. L'heuristique `roundRobin` offre également de mauvaises performances, sauf si nous modifions l'ordre statique des variables en nous inspirant de la première analyse présentée à la section 5.4.

Seules les stratégies de la famille Smear fonctionnent bien : les approches historiques `SmearMax` et `SmearSum` [8] ; la variante `SmearSumRel` appelée par l'optimiseur par défaut [13], et enfin la variante `SmearMaxRel` qui joue un rôle fondamental dans notre problème d'assemblage de vins.

Partant de cette analyse, nous avons élaboré une nouvelle stratégie. L'opérateur `X-Newton` a tout d'abord été désactivé, ce qui a permis d'améliorer les performances d'un facteur 4 sur les deux instances réelles. Nous avons ensuite comparé les quatre variantes de l'heuristique Smear. Le tableau 1 présente les résultats obtenus.

| | WineBlending | 1 | 2 |
|---|---|---|---|
| SmearSum | temps cpu (s) | $> 300$ | 57 |
| | précision | 0.00014 | 1e-10 |
| | #nœuds | 21880 | 4146 |
| SmearSumRel | temps cpu (s) | $> 300$ | $> 300$ |
| | précision | 0.0018 | 0.00017 |
| | #nœuds | 25864 | 24270 |
| SmearMax | temps cpu (s) | $> 300$ | 111 |
| | précision | 0.0013 | 1e-10 |
| | #nœuds | 25864 | 8851 |
| SmearMaxRel | temps cpu (s) | 0.35 | 27.4 |
| | précision | 1e-10 | 1e-10 |
| | #nœuds | 23 | 2048 |

TABLE 1 – Comparaison entre les différentes variantes de l'heuristique de branchement Smear. Chaque case contient 3 informations sur les résultats obtenus pour une variante donnée sur une des deux instances réelles. La première valeur indique le temps CPU (avec un temps-limite de 5 mn) ; la seconde valeur correspond à l'erreur finale (`1e-10` signifie que le denier appel à l'algorithme du simplexe réalisé par `InnerPolytope` trouve une solution sans erreur (arrondie à `1e-10`)) ; la troisième valeur indique le nombre de nœuds de branchement.

Nous pouvons remarquer que `SmearSum`, et surtout `SmearMaxRel`, donnent de bons résultats. C'est pourquoi nous avons choisi `SmearMaxRel` pour notre problème d'assemblage de vins.

### 5.4 Vers une stratégie de branchement dédiée

Nous avons également étudié expérimentalement quelles variables étaient sélectionnées dans différents processus d'optimisation. Nous avons ainsi déterminé expérimentalement que les variables $P_{w,b}$ et $V'_{w,b}$ doivent être souvent sélectionnées alors que les variables liées à l'optimisation du Min-Max doivent l'être moins souvent. Nous avons donc élaboré la stratégie

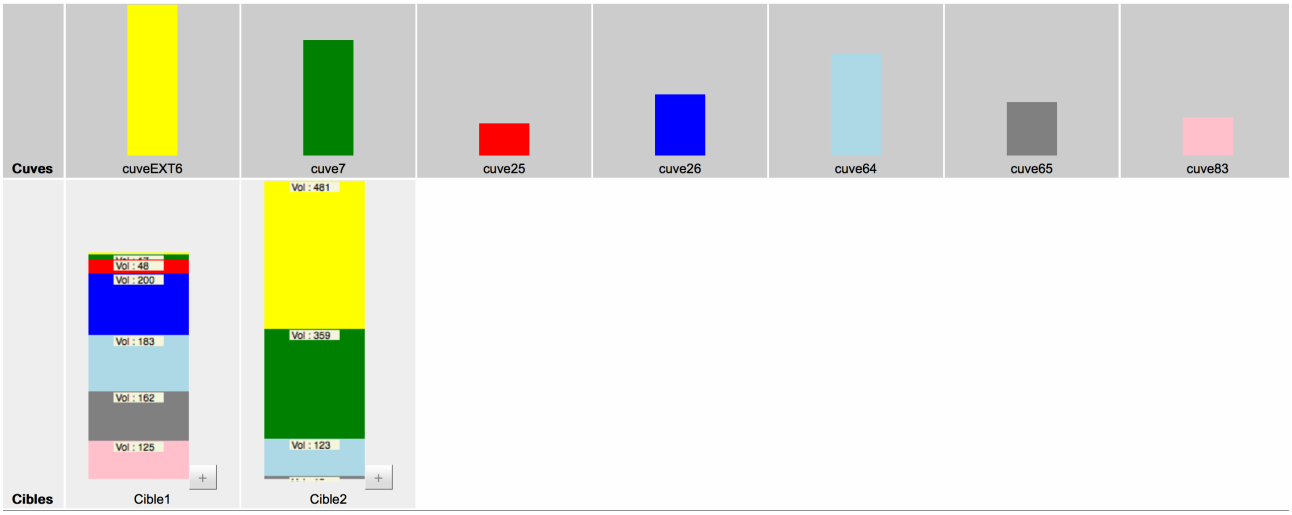



FIGURE 3 – Résultats obtenus pour `WineBlending1` : les volumes des vins cibles (en bas) sont le résultat de l'assemblage des vins de base (en haut).

dédiée suivante :

- seules les variables mentionnées ci-dessus peuvent être sélectionnées pour la bissection ;
- l'heuristique `SmearMaxRel` est appliquée au sous-ensemble de variables sélectionnées.

Cette heuristique dédiée offre des performances à peine meilleures que `SmearMaxRel` (23 secondes contre 27 secondes sur `WineBlending2`). Nous avons bien sûr besoin de plus d'instances pour mieux apprendre à partir des données.

### 5.5 Les vins résultats

La figure 3 présente la solution calculée pour `WineBlending1`. Les concentrations en arômes ($C_{w,a}$) dans les vins cibles sont détaillées par $\mathcal{W} = 2$ graphes radar à la figure 4. Le fait que les lignes bleues ($\hat{c}_{w,a}$) soient comprises entre les lignes vertes ($\hat{c}_{w,a} - \varepsilon_a$ et $\hat{c}_{w,a} + \varepsilon_a$) montre visuellement que la solution optimale est obtenue sous le seuil de tolérance $\varepsilon_a$.

Les concentrations en arômes pour les vins cibles de l'instance `WineBlending2` sont illustrées par $\mathcal{W} = 3$ graphes radar à la figure 5.

### 5.6 Séance de dégustation

Une première validation qualitative de notre travail a été réalisée en collaboration avec un œnologue. La société Nyseos lui a demandé d'élaborer un vin cible en assemblant différents vins de base. Elle a ensuite analysé le vin élaboré afin d'identifier les concentrations souhaitées en arômes ainsi que les concentrations dans les vins de base. Nyseos a alors soumis ce problème d'assemblage à notre outil qui a proposé des volumes à assembler différents de ceux choisis par l'œnologue. Une dégustation en aveugle a enfin été réalisée sur les deux vins produits.

Bien que les deux assemblages soient significativement différents, l'œnologue n'a pas été en mesure de différencier les deux vins.

Cette unique expérience n'est bien entendu pas suffisante pour valider la démarche mais c'est un résultat encourageant pour la pertinence de notre outil.

### 5.7 Un outil de configuration pour l'assemblage de vins

Les résultats expérimentaux sont encore préliminaires et n'ont été obtenus que sur deux cas réels. Ils suggèrent cependant que la stratégie actuelle, éventuellement dotée d'une heuristique de branchement dédiée, est efficace pour traiter la plupart des cas utiles en pratique. Par conséquent, pour mieux répondre aux souhaits des clients, nous pouvons imaginer une utilisation interactive de notre algorithme d'optimisation, à l'intérieur d'un outil de configuration. L'utilisateur pourrait interagir avec le système via des graphes radar correspondant aux différents vins cible, comme ceux des figures 4 et 5.

Une façon de modifier un assemblage est d'augmenter (ou diminuer) le poids relatif $\Omega_w$ d'un vin cible $w$. Un simple curseur sous chaque graphe radar permettrait par exemple de modifier cette valeur, ce qui entraînerait le recalcul d'une solution par l'optimiseur.

En suivant la même idée, l'utilisateur pourrait modifier le poids $\lambda_{w,a}$ d'un arôme $a$ pour un vin $w$ (par exemple, via un menu contextuel apparaissant lorsque le curseur de la souris se trouve sur l'axe correspondant

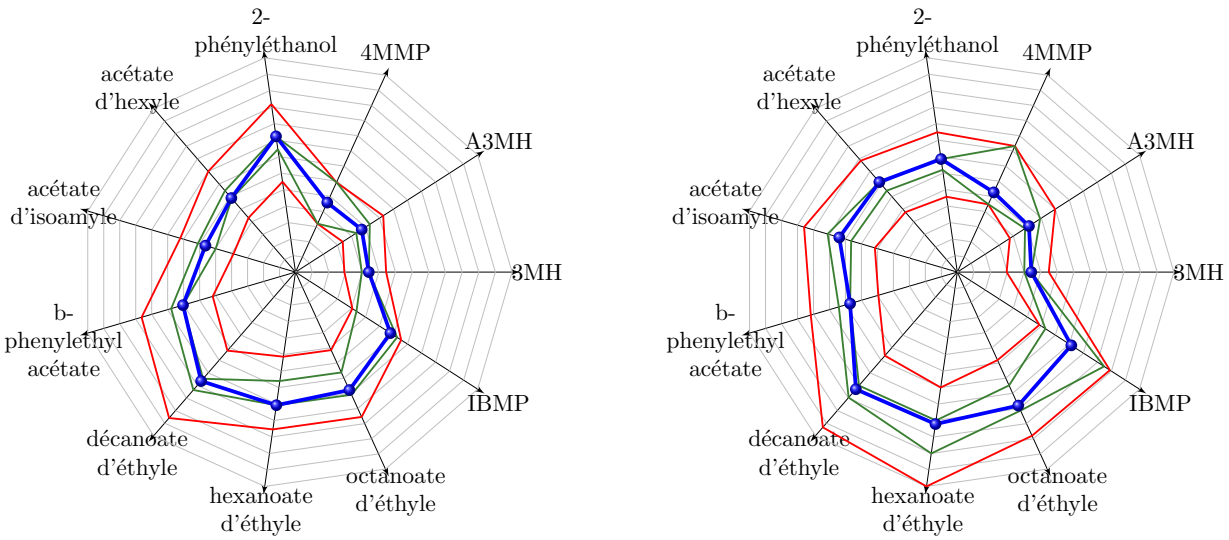



FIGURE 4 – Solution obtenue par notre modèle Min-Max sur les deux vins cibles de l'instance `WineBlending1`. Chaque axe du graphe radar correspond à un composé aromatique. La concentration obtenue $C_{w,a}$ (en bleu) doit rester comprise entre les limites imposées par les bornes $c^-_{w,a}$ et $c^+_{w,a}$ (en rouge) tout en restant le plus proche possible de la concentration souhaitée $\hat{c}_{w,a}$. Cette dernière est représentée par les deux courbes en vert $\hat{c}_{w,a} - \varepsilon_a$ et $\hat{c}_{w,a} + \varepsilon_a$ qui traduisent l'erreur de mesure tolérée pour l'arôme $a$ correspondant.

du graphe radar), et l'outil déclencherait une nouvelle optimisation. Enfin, une dernière possibilité plus intrusive serait de permettre le renforcement des concentrations maximales $c^+_{w,a}$ (ou minimales $c^-_{w,a}$) admissibles pour un arôme $a$ dans un vin $w$. L'utilisateur pourrait simplement sélectionner une borne et notre système exécuterait deux optimisations afin de déterminer deux informations (en supposant que $c^+_{w,a}$ a été sélectionnée) :

- la plus petite valeur de $c^+_{w,a}$ possible (en respectant les contraintes),
- la plus petite valeur possible de $c^+_{w,a}$ qui n'augmente pas l'erreur globale $E$.

## 6 Conclusion

Nous avons présenté dans cet article la première approche en programmation par contraintes pour résoudre le problème de l'assemblage de vins. Ce problème peut être modélisé par un MINLP ou un CSP numérique supportant des contraintes disjonctives qui s'avèrent cruciales en pratique. Ces contraintes sont notamment liées à des impératifs physiques sur le transfert d'un volume de vin entre deux cuves. Nous nous sommes attachés à définir un modèle Min-Max sans opérateurs max ou valeur absolue. Nous avons conçu une stratégie d'optimisation de l'assemblage de vins qui permet d'obtenir, en quelques secondes, une solution (sans erreurs sur les volumes ou les concentrations) pour deux instances réelles fournies par la société Nyseos. Une séance de dégustations réalisée avec un expert œnologue a permis de valider l'intérêt de cette approche. Ces résultats encourageants permettent d'envisager l'utilisation de notre modèle dans un outil interactif de configuration dédié à l'assemblage de vins.

## Références

[1] I. Araya, G. Trombettoni, and B. Neveu. A Contractor Based on Convex Interval Taylor. In *Proc. CPAIOR*, pages 1–16. LNCS 7298, 2012.

[2] P. Belotti. Couenne, a user's manual, 2013. `www.coin-or.org/Couenne/couenne-user-manual.pdf`.

[3] F. Benhamou, F. Goualard, L. Granvilliers, and J.-F. Puget. Revising Hull and Box Consistency. In *Proc. ICLP*, pages 230–244, 1999.

[4] G. Chabert. Interval-Based EXplorer, 2013. `www.ibex-lib.org`.

[5] G. Chabert and L. Jaulin. Contractor Programming. *Artificial Intelligence*, 173 :1079–1100, 2009.

[6] L. Dagan. *Potentiel aromatique des raisins de Vitis vinifera L. cv. Petit Manseng et Gros Manseng. Contribution à l'arôme des vins de pays*

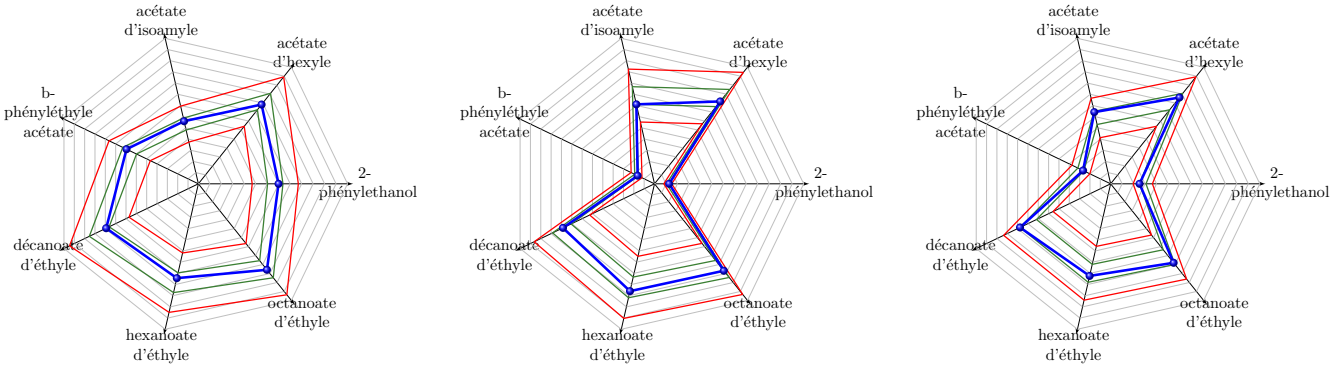



FIGURE 5 – Graphes radar obtenus pour l'instance `WineBlending2`.

*Côtes de Gascogne*. PhD thesis, École nationale supérieure agronomique (Montpellier), 2006.

[7] J. Ferrier and D. Block. Neural-network-assisted optimization of wine blending based on sensory analysis. *American Journal of Enology and Viticulture*, 52(4) :386–395, 2001.

[8] R.B. Kearfott and M. Novoa III. INTBIS, a portable interval Newton/Bisection package. *ACM Trans. on Mathematical Software*, 16(2) :152–157, 1990.

[9] F. Messine. *Méthodes d'optimisation globale basées sur l'analyse d'intervalle pour la résolution des problèmes avec contraintes*. PhD thesis, LIMA-IRIT-ENSEEIHT-INPT, Toulouse, 1997.

[10] D. B. Moore and T. G. Griffin. Computer blending technology. *American Journal of Enology and Viticulture*, 29(1) :50–53, 1978.

[11] J. Ninin, F. Messine, and P. Hansen. A Reliable Affine Relaxation Method for Global Optimization. Technical Report RT-APO-10-05, IRIT, 2010.

[12] M. Tawarmalani and N. V. Sahinidis. A Polyhedral Branch-and-Cut Approach to Global Optimization. *Mathematical Programming*, 103(2) :225–249, 2005.

[13] G. Trombettoni, I. Araya, B. Neveu, and G. Chabert. Inner Regions and Interval Linearizations for Global Optimization. In *AAAI*, pages 99–104, 2011.