Rapport de projet
Module : Modélisation et résolution pour l'optimisation

Julien Antigny, Aymene Mohammed Bouayed et Camelo Salhab Raissa

Novembre 2021

Table des matières

Sujet du projet **1**

1 Problèmes d'optimisation sous contraintes **1**
1.1 Introduction . . . 1
1.2 Modélisation du problème sous forme COP . . . 1
1.2.1 Variables & domaine de définition . . . 1
1.2.2 Contraintes . . . 1
1.2.3 Définition des critères à optimiser . . . 2
1.2.4 Résumé de la modélisation . . . 3
1.3 Résolution et comparaison des différents solveurs . . . 3
1.3.1 Configuration de l'ordinateur . . . 3
1.3.2 Génération des fichiers XML . . . 3
1.3.3 Résolution . . . 4
1.4 Conclusion . . . 6

2 Problèmes de satisfaction de contraintes valués **8**
2.1 Introduction . . . 8
2.2 Modélisation sous forme CSP valué du problème . . . 8
2.2.1 Variables & domaine de définition . . . 8
2.2.2 Contraintes . . . 8
2.2.3 Résumé de la modélisation . . . 9
2.3 Résolution avec le solveur Toulbar 2 . . . 9
2.3.1 Configuration de l'ordinateur . . . 9
2.3.2 Génération des fichiers CFN . . . 9
2.3.3 Résolution . . . 10
2.4 Conclusion . . . 11

Sujet du projet

Ce projet consiste à modéliser et résoudre à l'aide de solveurs le problème d'allocation de fréquences entre des stations. La situation initiale est la suivante : on dispose de n stations réparties dans k régions distinctes. Chaque station est composée d'un émetteur et d'un récepteur et ne peut exploiter que certaines fréquences. Pour pouvoir communiquer avec d'autres stations, chaque station doit disposer de deux fréquences : une pour son émetteur et une pour son récepteur.

Pour des raisons matérielles, l'écart entre les deux fréquences de la station i doit être égal à δ_i. Deux stations différentes peuvent donc avoir le même écart comme des écarts différents. Si deux stations sont proches l'une de l'autre, les fréquences utilisées par ces stations doivent être suffisamment espacées pour éviter les interférences. On notera $\Delta_{i,j}$ l'écart minimum à garantir entre les fréquences des stations i et j.

Enfin, pour chaque région, on souhaite limiter le nombre de fréquences différentes utilisées. On notera n_i le nombre maximal de fréquences différentes utilisées pour la région i.

Chapitre 1

Problèmes d'optimisation sous contraintes

1.1 Introduction

L'objectif de cette partie est d'utiliser le paradigme COP afin de résoudre le problème d'attribution de fréquences comme énoncé dans le sujet de ce projet tout en optimisant un des critères suivant :

- Minimiser le nombre de fréquences utilisées ;
- Utiliser les fréquences les plus basses possibles ;
- Minimiser la largeur de la bande de fréquences utilisées.

1.2 Modélisation du problème sous forme COP

Cette section aborde la définition des variables à utiliser pour résoudre le problème ainsi que la formalisation des contraintes posées dans l'énoncé dans un context de COP.

1.2.1 Variables & domaine de définition

Dans notre modélisation on utilise deux variables pour chaque station i :

- La première $e_i \in \mathbb{N}$ qui repésente la fréquence d'émission de la station i.
- La seconde $r_i \in \mathbb{N}$ qui repésente la fréquence de réception de la station i.

Formellement, nous définissons l'ensemble des variables X comme suit :

$$X = \{e_1, ..., e_n)\} \cup \{r_1, ..., r_n)\}$$

Et on a *apriori* les domaines de définition d_{e_i} et d_{r_i} de chacune des variables e_i et r_i qui sont égal à $\mathbb{N}$, cependant les domaines de définition exactes de chaque variable pour chaque instance sont recupérer depuis le fichier de données.

1.2.2 Contraintes

Dans cette partie, nous définissons l'ensemble des contraintes C du problème que nous traitons.

Contrainte I : Ecart entre les fréquences

Comme on doit respecter un écart δ_i entre la fréquence d'émission e_i et la fréquence de réception r_i d'une station i, on définie une contrainte $c_i^{(1)}$ pour chaque station $i \in \{1, \ldots, n\}$ comme suit :

$$c_i^{(1)} : |e_i - r_i| = \delta_i$$

Où n est le nombre de stations.

Contrainte II : Liaison entre deux stations

Pour que deux stations i et j puissent communiquer, la fréquence d'émission de l'une doit être égale à la fréquence de réception de l'autre et vice-versa. On modélise ces contraintes pour chaque couple de stations i et j qui communiquent par la contrainte $c_{i,j}^{(2)}$ suivante :

$$c_{i,j}^{(2)} : r_i = e_j \wedge e_i = r_j \quad si \quad l_{i,j} = 1$$

Où $l_{i,j}$ est une variable indicatrice, obtenue depuis les données, et vaux 1 si la station i communique avec la station j, 0 sinon.

Contrainte III : Interférences entre les stations

Afin de gérer les interferences entre deux stations i et j et s'assurer que l'écart entre leurs fréquences est supérieur ou égale a un seuil minimal $\Delta_{i,j}$, on utilise pour tout couple de station i et j la contrainte $c_{i,j}^{(3)}$ suivante :

$$c_{i,j}^{(3)} : |e_i - e_j| \geq \Delta_{i,j} \ \wedge \ |r_i - r_j| \geq \Delta_{i,j} \quad si \quad I_{i,j} = 1$$

Où $I_{i,j}$ est une variable indicatrice, obtenue depuis les données, et vaux 1 lorsque les stations i et j intéfèrent, 0 sinon.

Contrainte IV : Nombre maximale de frequences par region

Cette contrainte permet d'imposer que le nombre de fréquences utilisées par les stations d'une région j ne dépase pas le seuil n_j maximal autorisé dans cette région. Formellement, on peut écrire une contrainte $c_j^{(4)}$ pour toute region $j \in \{1, \ldots, k\}$ comme suit :

$$c_j^{(4)} : NVALUES\left(\bigcup_{i=1}^{n}\{rg_{j,i} \cdot e_i\} \cup \bigcup_{i=1}^{n}\{rg_{j,i} \cdot r_i\}\right) \leq n_j$$

Où $rg_{j,i}$ est une variable indicatrice ,obtenue depuis les données, qui vaux 1 si la station i appartient à la région j et 0 sinon. Donc la multiplication $rg_{j,i} \cdot e_i$ indique si la fréquence e_i apartient à la region j ou pas (de même pour la fréquence r_i).

1.2.3 Définition des critères à optimiser

Modélisation du critère I

La premier critère qu'on pourrais optimiser est la minimisation du nombre de fréquences utilisées. On peut donc l'exprimer comme suit :

$$Minimiser\left(NVALUES\left(\bigcup_{i=1}^{n}\{e_i\} \cup \bigcup_{i=1}^{n}\{r_i\}\right)\right)$$

Modélisation du critère II

Le second critère qu'on pourrais optimiser est celui d'utiliser les fréquences les plus basses possibles. On minimise alors la fréquence la plus haute utilisée et on écrit :

$$Minimiser\left(Maximum\left(\bigcup_{i=1}^{n}\{e_i\} \cup \bigcup_{i=1}^{n}\{r_i\}\right)\right)$$

Modélisation du critère III

Le dernier critère qu'on pourrais optimiser est la minimisation de la largeur de la bande de fréquences utilisées c'est à dire l'écart entre la plus haute et la plus basse fréquence utilisée. On peut donc écrire :

$$Minimiser\left(Maximum\left(\bigcup_{i=1}^{n}\{e_i\}\cup\bigcup_{i=1}^{n}\{r_i\}\right)-Minimum\left(\bigcup_{i=1}^{n}\{e_i\}\cup\bigcup_{i=1}^{n}\{r_i\}\right)\right)$$

1.2.4 Résumé de la modélisation

Soit $\mathcal{P}' = (X, D, C, f)$ une instance du problème dans le context de COP. On a ainsi :

— $X = \{e_1, ..., e_n)\} \cup \{r_1, ..., r_n\}$

— $D = \{d_{e_i}, d_{r_i} | \forall i \in \{1, \ldots, n\}\}$ Où n est le nombre de stations et chacun des d_{e_i} et d_{r_i} est obtenu depuis le fichier des données pour chaque instance.

— Pour les contraintes on a :

— Contrainte de l'écart entre les fréquences :

$$\forall i \in \{1, \ldots, n\},\ |e_i - r_i| = \delta_i$$

Où n est le nombre de stations.

— Contrainte de liaison :

$$\forall i \in \{1, \ldots, n\}, \forall j \in \{1, \ldots, n\},\ r_i = e_j \wedge e_i = r_j \quad si \quad l_{i,j} = 1$$

Où $l_{i,j}$ est une variable indicatrice qui vaux 1 lorsque la station i communique avec la station j et 0 sinon.

— Contrainte d'interférence :

$$\forall i \in \{1, \ldots, n\}, \forall j \in \{1, \ldots, n\},\ (|e_i - e_j| \geq \Delta_{i,j}) \wedge (|r_i - r_j| \geq \Delta_{i,j}) \quad si \quad I_{i,j} = 1$$

Où $I_{i,j}$ est une variable indicatrice qui vaux 1 lorsque les stations i et j intéfèrent et 0 sinon.

— Contrainte du nombre maximale de fréquences par région :

$$\forall j \in \{1, \ldots, k\},\ NVALUES\left(\bigcup_{i=1}^{n}\{rg_{j,i} \cdot e_i\} \cup \bigcup_{i=1}^{n}\{rg_{j,i} \cdot r_i\}\right) \leq n_j$$

Où $rg_{j,i}$ est une variable indicatrice qui vaux 1 si la station i appartient à la région j et 0 sinon.

— Fonction objectif f peut être une de ces fonctions :

— $Minimiser\,(NVALUES\,(\bigcup_{i=1}^{n}\{e_i\} \cup \bigcup_{i=1}^{n}\{r_i\}))$

— $Minimiser\,(Maximum\,(\bigcup_{i=1}^{n}\{e_i\} \cup \bigcup_{i=1}^{n}\{r_i\}))$

— $Minimiser\,(Maximum\,(\bigcup_{i=1}^{n}\{e_i\} \cup \bigcup_{i=1}^{n}\{r_i\}) - Minimum\,(\bigcup_{i=1}^{n}\{e_i\} \cup \bigcup_{i=1}^{n}\{r_i\}))$

1.3 Résolution et comparaison des différents solveurs

1.3.1 Configuration de l'ordinateur

L'ordinateur sur lequel nous avons fait nos expérimentations est composé d'un processeur Intel(R) Core(TM) i7-5557U CPU bi-coeurs avec une fréquence allant de 3,10 GHz à 3.40 GHz, 16Go de RAM et MAC OS Monterey comme système d'exploitation.

1.3.2 Génération des fichiers XML

Afin de générer les fichiers XML de chaque instance on a utilisé la bibliothèque `pyCSP3` (lien). Par le biais de cette bibliothèque nous avons écrit toutes les contraintes du problèmes qu'on traite ainsi que les différentes fonctions à optimiser comme des varients de modèles créé. Le code de notre implementation est disponible sur ce lien GitHub.

Le tableau 1.1 suivant résume le nombre de variables et contraintes générées pour chaque instance.

Instances	# Variables	# Contraintes
celar_50_7_10_5_0_800000_0	100	301
celar_50_7_10_5_0_800000_1	100	293
celar_50_7_10_5_0_800000_6	100	295
celar_50_7_10_5_0_800000_7	100	301
celar_50_7_10_5_0_800000_8	100	291
celar_150_13_15_5_0_800000_2	300	1.095
celar_150_13_15_5_0_800000_8	300	1.099
celar_150_13_15_5_0_800000_26	300	1.055
celar_150_13_15_5_0_800000_28	300	1.135
celar_150_13_15_5_0_800000_29	300	1.151
celar_250_25_15_5_0_820000_5	500	1.543
celar_250_25_15_5_0_820000_7	500	1.523
celar_250_25_15_5_0_820000_9	500	1.585
celar_250_25_15_5_0_820000_20	500	1.537
celar_250_25_15_5_0_820000_22	500	1.603
celar_500_30_20_5_0_870000_24	1.000	3.196
celar_500_30_20_5_0_870000_29	1.000	3.286
celar_500_30_20_5_0_870000_45	1.000	3.288
celar_500_30_20_5_0_870000_48	1.000	3.254

TABLE 1.1 – Nombre de variables et contraintes générées dans le fichier XML pour chaque instance.

1.3.3 Résolution

Afin de résoudre chaque instance, on a utilisé le fichier XML générés avec `pyCSP3` et un des deux solvers suivant :

- Le **Solver AbsCon** qui est inclue dans la bibliothèque `pyCSP3`. Afin d'utiliser ce solver nous avons juste utiliser le paramètre `-solve` dans la commande de génération des fichiers XML (Voir la première boucle du fichier `main.py` dans notre implementation sur GitHub).
- Le **Solver Choco** qui a été obtenu depuis le lien suivant : `(lien de téléchargement)`. Afin d'utiliser ce solver nous avons utiliser la commande `java -jar choco-parsers-4.10.7-jar-with-dependencies.jar fichier_xml.xml` (Voir le fichier `run-choco.py` dans notre implementation sur GitHub).

Résultats obtenus avec le Solver AbsCon

En utilisant le solver AbsCon ainsi que les fichiers XML généré pour résoudre les instances on obtient les résulats résumés dans le tableau 1.2, où on a :

- La colonne **Instance** qui représente l'instance traitée.
- La colonne **Variant M1** qui représente la résolution en optimisant le critère I.
- La colonne **Variant M2** qui représente la résolution en optimisant le critère II.
- La colonne **Variant M3** qui représente la résolution en optimisant le critère III.

On a aussi deux sous colonnes à savoir **Temps** et **Coût**. La sous colonne Temps représente le temps de résolution en secondes. Par contre, la sous colonne Coût représente la valeur minimale du critère à optimiser qu'on a pu atteindre.

Depuis le tableau 1.2 on remarque que pour tous les différents critères/variants on arrive à des coûts qui sont vraiments bas et ceci pour un nombre de stations très grand. Par exemple, en prenant le critère 1 (Variant M1) qui est le critère le plus intuitif, on voit bien qu'on arrive à utiliser au maximum 8 fréquences seulement et ceci pour 500 stations.

En ce qui concerne le temps d'execution, on remarque une grande différence entre les critères. Le critère le plus facile à optimiser est le critère 2 car on ne fait que calculer le maximum de la liste des fréquences utilisées. Pour le critère 3 on calcul le maximum et le minimum de cette liste ceci auguemente le temps de calcul par rapport au critère 2. Et finalement pour le critère 1 qui est le plus plus difficile à calculer parmi les autres et donc c'est lui qui prend le plus de temps à optimiser, ceci car on calcul le nombre de fréquences distinctes utilisées.

	Variant M1		Variant M2		Variant M3	
Instance	Temps	Coût	Temps	Coût	Temps	Coût
celar_50_7_10_5_0_800000_0	3.048	6	1.803	350	2.027	336
celar_50_7_10_5_0_800000_1	2.811	6	1.429	378	6.798	364
celar_50_7_10_5_0_800000_6	4.488	6	1.593	378	2.339	364
celar_50_7_10_5_0_800000_7	4.379	6	1.855	280	2.387	238
celar_50_7_10_5_0_800000_8	3.619	6	1.652	280	2.064	252
celar_150_13_15_5_0_800000_2	7.678	8	1.924	462	2.846	448
celar_150_13_15_5_0_800000_8	6.597	8	2.861	378	6.249	364
celar_150_13_15_5_0_800000_26	14.514	8	2.226	364	3.768	350
celar_150_13_15_5_0_800000_28	11.39	8	2.199	462	3.631	448
celar_150_13_15_5_0_800000_29	37.464	8	2.424	406	8.984	378
celar_250_25_15_5_0_820000_5	6.994	6	2.103	308	4.996	294
celar_250_25_15_5_0_820000_7	31.672	8	2.133	406	7.179	378
celar_250_25_15_5_0_820000_9	7.069	6	2.373	308	3.885	294
celar_250_25_15_5_0_820000_20	6.858	6	2.58	308	3.786	294
celar_250_25_15_5_0_820000_22	6.636	8	2.211	476	3.934	462
celar_500_30_20_5_0_870000_24	65.49	6	3.72	364	11.101	350
celar_500_30_20_5_0_870000_29	21.269	8	3.206	308	31.294	294
celar_500_30_20_5_0_870000_45	30.438	8	3.259	406	20.035	392
celar_500_30_20_5_0_870000_48	192.986	8	3.065	308	37.143	294

TABLE 1.2 – Résultats de résolution avec le solver AbsCon.

Résultats obtenus avec le Solver Choco

En utilisant le solver Choco ainsi que les fichiers XML générés pour résoudre les instances on obtient les résulats résumés dans le tableau 1.3, où on a :

— La colonne **Instance** qui représente l'instance traitée.
— La colonne **Variant M1** qui représente la résolution en optimisant le critère I.
— La colonne **Variant M2** qui représente la résolution en optimisant le critère II.
— La colonne **Variant M3** qui représente la résolution en optimisant le critère III.

On a aussi deux sous colonnes à savoir **S/O** et **Temps**. La sous colonne S/O représente si le solveur à trouvé la solution optimale (O) ou bien il n'a trouvé qu'une solution (S). Par contre, la sous colonne Temps représente le temps pris par le solver pour trouver l'optimum.

On remarque qu'on a pris des caractéristiques différentes par rapport au solver AbsCon ceci car le solver Choco prend beaucoup plus de temps pour converger vers la solution qui minimise le critère à optimiser. Donc on a limité le temps de calcul pour chaque instance à 1 minute et on note si le solver trouve l'optimum ou pas. Le choix d'une minute pour le temps de calcul par instance est motivé par le fait que AbsCon converge dans la majorité des cas en moins de 40sec et aussi par le fait qu'on a beaucoup d'instances et de variants.

Depuis le tableau 1.3 on remarque que pour le critère 1 et 3 le solver Choco n'atteint quasiement jamais l'optimum dans 1 minute. Par contre il atteint quasiment toujours l'optimum pour le critère 2. Ceci est

Instances	Variant M1		Variant M2		Variant M3	
	S/O	Temps	S/O	Temps	S/O	Temps
celar_50_7_10_5_0_800000_0	S	-	O	9.52	S	-
celar_50_7_10_5_0_800000_1	S	-	O	7.68	S	-
celar_50_7_10_5_0_800000_6	S	-	O	5.72	S	-
celar_50_7_10_5_0_800000_7	S	-	O	5.55	S	-
celar_50_7_10_5_0_800000_8	S	-	O	4.62	O	49.72
celar_150_13_15_5_0_800000_2	S	-	O	18.20	S	-
celar_150_13_15_5_0_800000_8	S	-	O	48.60	S	-
celar_150_13_15_5_0_800000_26	S	-	O	9.41	S	-
celar_150_13_15_5_0_800000_28	S	-	O	18.91	S	-
celar_150_13_15_5_0_800000_29	S	-	O	21.82	S	-
celar_250_25_15_5_0_820000_5	S	-	O	19.20	S	-
celar_250_25_15_5_0_820000_7	S	-	O	22.96	S	-
celar_250_25_15_5_0_820000_9	S	-	O	26.58	S	-
celar_250_25_15_5_0_820000_20	S	-	O	23.29	S	-
celar_250_25_15_5_0_820000_22	S	-	O	12.49	S	-
celar_500_30_20_5_0_870000_24	S	-	O	35.30	S	-
celar_500_30_20_5_0_870000_29	S	-	S	-	S	-
celar_500_30_20_5_0_870000_45	S	-	O	16.93	S	-
celar_500_30_20_5_0_870000_48	S	-	S	-	S	-

TABLE 1.3 – Résultats de résolution avec le solver Choco

probablement dû aux algorithmes de résolution de ce solver mais aussi ça dépend de la complexité du critère à optimiser. Où on a le critère 2 qui est juste le calcul du maximum donc il est beaucoup plus simple à calculer que les autres critères qui consiste à calculer le maximum et le minimum ou bien calculer le nombre de valeurs distinctes dans une liste.

Comparaison

Depuis les expérimentations que nous avons fait et dont les résultats sont illustrés dans les tableaux 1.2 et 1.3, on peut clairement voir que pour le problème qu'on traite le solver AbsCon est meilleur. Ceci car il atteint toujours le minimum de du critère qu'on optimise et il le fait en des temps très rapides peut importe la complexité du critère chose que le solver Choco n'arrive pas à faire dans la majorité des cas.

En prenant le cas du critère 2 (Variant M2) car c'est le seul variant où le solver Choco atteint l'optimum, on remarque bien que les temps de résolution du solver Choco sont à au moins un ordre de magniture plus grand que ceux de AbsCon (Voir figure 1.1).

1.4 Conclusion

Dans cette partie on a modélisé la problématique du sujet traité dans ce projet sous le format d'un COP. Par suite, on a implementé et testé notre modélisation en utilisant la bibliothèque `pyCSP3` et nous avons essayé de résoudre plusieurs instances en utilisant 3 différents critères et 2 différents solvers notemment le solver AbsCon et le solver Choco. Nous avons trouvé que le solver AbsCon est le plus performant car pour chaque instance il atteint l'optimum du critère à optimiser en un temps qui ce mesure en secondes et qui est au pire des cas un ordre de magnitude plus petit que celui du solver Choco.

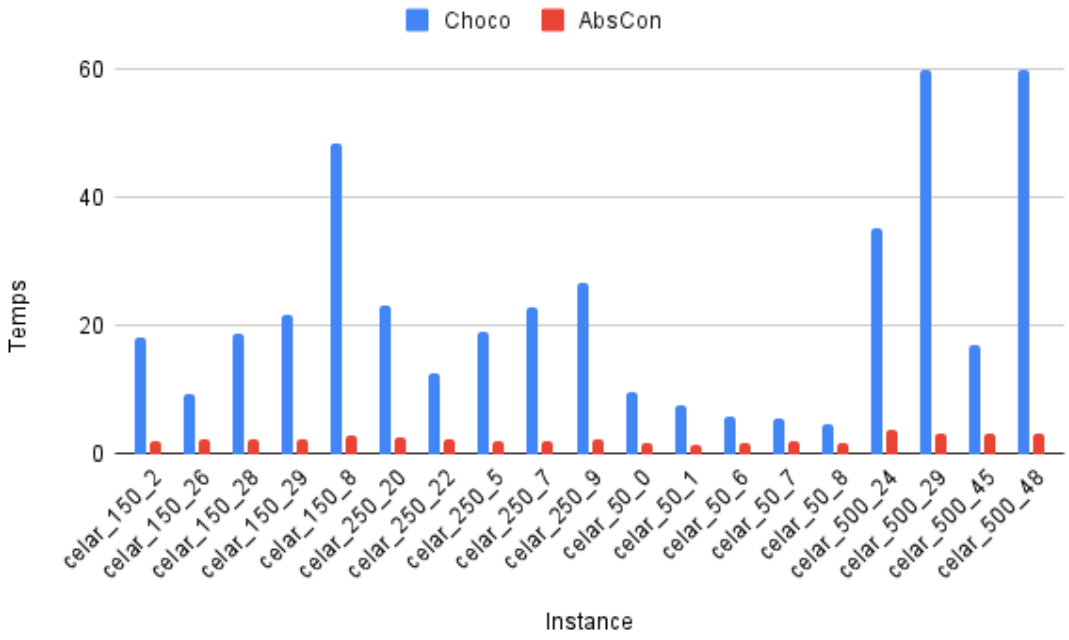

FIGURE 1.1 – Différence en temps entre le solver AbsCon et Choco sur toutes les instances en utilisant le critère 2 (Variant M2).

Chapitre 2

Problèmes de satisfaction de contraintes valués

2.1 Introduction

Certains CSP ne peuvent avoir de solution. Néanmoins, il est possible de trouver des solutions satisfaisant autant que possible les contraintes en modélisant les problèmes sous forme de CSP valués. Dans cette partie nous nous intéréssons à l'utilisation de ce paradigme de résolution pour des instances qui ne sont pas satifiables dans un cadre de COP.

2.2 Modélisation sous forme CSP valué du problème

Cette section aborde la définition des variables à utiliser pour résoudre le problème ainsi que la formalisation des contraintes posées dans l'énoncé dans un context de CSP valué.

2.2.1 Variables & domaine de définition

Pour la définition et modélisation des variables on utilise la même définition de variables que celle utilisée dans le context des COPs. On aura alors :

$$X = \{e_1, ..., e_n)\} \cup \{r_1, ..., r_n)\}$$

Où chaque e_i et r_i represente la fréquence d'émission et de récéption de la station i. De même, les domaines de définition d_{e_i} et d_{r_i} exactes de chaque variable e_i et r_i pour chaque instance sont recupérer depuis le fichier de données.

2.2.2 Contraintes

Dans le context des CSP valué on utilise les mêmes contraintes que celles déjà utilisées dans le context des COPs. Cependant, on doit distinguer entre les contraintes dûres et les contraintes molle. Ceci en affectant des poids à ces contraintes et en imposant un poids de $+\infty$ aux contraintes dûres si elles sont voilées et un poids $< +\infty$ pour les contraintes molles si elles sont voilées. En suivant ces règles on redefinies les contraintes précédentes comme suit :

Contrainte I : Ecart entre les fréquences

La contrainte de l'écart entre la fréquence d'envoie et de récéption d'une station étant essentiel au bon fonctionnement de cette station, on définie alors cette contrainte comme étant une contrainte dûre et on a :

$$c_i^{(1)} : |e_i - r_i| = \delta_i \quad \rightarrow (0,\ +\infty)$$

Contrainte II : Liaison entre deux stations

La contrainte de l'égalité entre la fréquence de récéption et d'envoie des fréquences des stations qui communiquent est primordiale pour le fonctionnement du réseau. Alors cette contrainte est une contrainte dûre et dans le format de CSP valué on peut l'écrire comme suit :

$$c_{i,j}^{(2)} : r_i = e_j \wedge e_i = r_j \quad si \quad l_{i,j} = 1 \quad \rightarrow (0,\ +\infty)$$

Où $l_{i,j}$ est une variable indicatrice obtenue depuis les données qui vaux 1 lorsque la station i communique avec la station j et 0 sinon.

Contrainte III : Interférences entre les stations

Afin de gérer les interferences dans le context les CSP valué on a opté pour mettre cette contrainte comme une contrainte mollé et on la modélise pour chaque couple de stations i et j qui interfèrent comme suit :

$$c_{i,j}^{(3)} : |e_i - e_j| \geq \Delta_{i,j} \wedge |r_i - r_j| \geq \Delta_{i,j} \quad si \quad I_{i,j} = 1 \quad \rightarrow (0,\ 5)$$

Où $I_{i,j}$ est une variable indicatrice obtenue depuis les données qui vaux 1 lorsque les stations i et j intéfèrent et 0 sinon.

2.2.3 Résumé de la modélisation

Soit $\mathcal{P}' = (X, D, C, SV, \phi)$ une instance du problème dans le context de CSP valué. On a ainsi :

- $X = \{e_1, ..., e_n)\} \cup \{r_1, ..., r_n)\}$
- $D = \{d_{e_i}, d_{r_i} | \forall i \in \{1, \ldots, n\}\}$ Où n est le nombre de stations et chaque d_{e_i} et d_{r_i} est obtenu depuis le fichier des données.
- Pour les contraintes ainsi que leur valuation on a :
  - Contrainte de l'écart entre les fréquences :

$$\forall i \in \{1, \ldots, n\}, |e_i - r_i| = \delta_i \quad \rightarrow (0,\ +\infty)$$

  - Contrainte de liaison :

$$\forall i \in \{1, \ldots, n\}, \forall j \in \{1, \ldots, n\},\ r_i = e_j \wedge e_i = r_j \quad si \quad l_{i,j} = 1 \quad \rightarrow (0,\ +\infty)$$

  - Contrainte d'interférence :

$$\forall i \in \{1, \ldots, n\}, \forall j \in \{1, \ldots, n\},\ (|e_i - e_j| \geq \Delta_{i,j}) \ \wedge \ (|r_i - r_j| \geq \Delta_{i,j}) \quad si \quad I_{i,j} = 1 \quad \rightarrow (0,\ 5)$$

- $SV = (\mathbb{N} \cup \{+\infty\}, \leq, +, 0, +\infty)$

2.3 Résolution avec le solveur Toulbar 2

2.3.1 Configuration de l'ordinateur

L'ordinateur sur lequel nous avons fait nos expérimentations est composé d'un processeur Intel(R) Core(TM) i7-5557U CPU bi-coeurs avec une fréquence allant de 3,10 GHz à 3.40 GHz, 16Go de RAM et MAC OS Mentrey comme système d'exploitation.

2.3.2 Génération des fichiers CFN

Afin de générer les fichiers CFN qui résume l'instance du problème et qu'on donne au solver ToulBar2 pour la résolution, nous avons utilisé la bibliothèque `pyToulBar2`. Ceci en construisant un objet CFN dont on lui donne :

- La borne supérieure pour la résolution.
- Le domaine de définition de chacune des variables.

— La porté ainsi que les coûts de la satisfaction ou de violation de chacune des contraintes.

Par suite, il suffit juste d'appeler la méthode `.Dump` avec le nom du fichier et le fichier CFN sera créé et sauvgarder (Voir le fichier `create_cfn.py` sur le GitHub du projet).

Afin d'utiliser ce fichier CFN, on crée une variable de type CFN, on spécifie la même borne supérieure pour la résolution que celle du fichier puis on fait un appel à la méthode `.Read` avec le chemin vers le fichier. Enfin, on appel la méthode `.Solve` pour lancer le processus de résolution.

2.3.3 Résolution

Afin de résoudre les différentes instances WCSP on a utilisé le solver `ToulBar2` (lien) plus précisément `pyToulBar2` (lien). Les resultats qu'on a obtenu sont résumé dans le tableau 2.1 où on a :

— La colonne **"Instance"** qui represente le nom de l'instance/fichier WCSP traité.
— La colonne **"Solution ?"** qui represente si on a trouvé une solution à ce problème ou pas. Dans le cas où la réponse est "non", c'est parce qu'on a une violation d'une contrainte dûre. Ceci vient du fait qu'on avait mis la borne superieur du solveur au TOP pour que toutes les contraintes dûre soit satisfaites.
— La colonne **"Coût"** qui represente le coût de la solution trouvée. Ce coût est différent du TOP si on peut trouver une solution sans violé une contrainte dûre.
— La colonne **"Nb violations"** qui represente le nombre de contraintes molles violées. Les résultats de cette colonne sont obtenus en calculant $Cot\%5$ où 5 est le cout de violation d'une contrainte molle.
— La colonne **"Temps (sec)"** qui représente le temps pris par le solver pour retourner une réponse.

Instances	Solution ?	Coût	Nb violations	Temps (sec)
celar_50_7_10_5_0_800000_0	Non	TOP	-	0,012
celar_50_7_10_5_0_800000_1	Oui	10	2	0,031
celar_50_7_10_5_0_800000_4	Oui	10	2	0,040
celar_50_7_10_5_0_800000_9	Non	TOP	-	0,021
celar_50_8_10_5_0_800000_8	Oui	10	2	0,021
celar_150_13_15_5_0_800000_18	Oui	10	2	0,222
celar_150_13_15_5_0_800000_2	Oui	0	0	0,177
celar_150_13_15_5_0_800000_20	Non	TOP	-	0,090
celar_150_13_15_5_0_800000_25	Oui	10	2	0,154
celar_150_13_15_5_0_800000_9	Oui	30	6	0,342
celar_250_25_15_5_0_820000_0	Oui	30	6	0,273
celar_250_25_15_5_0_820000_17	Oui	30	6	0,249
celar_250_25_15_5_0_820000_29	Oui	20	4	0,241
celar_250_25_15_5_0_820000_6	Oui	10	2	0,307
celar_250_25_15_5_0_820000_8	Oui	30	6	0,390
celar_500_30_20_5_0_870000_0	Oui	50	10	0,846
celar_500_30_20_5_0_870000_25	Oui	20	4	0,487
celar_500_30_20_5_0_870000_30	Oui	20	4	0,464
celar_500_30_20_5_0_870000_49	Oui	80	16	0,511
celar_500_30_20_5_0_870000_8	Oui	10	2	0,421

TABLE 2.1 – Resultats obtenus avec le solver ToulBar2 sur les instances WCSP.

Depuis le tableau 2.1 on remarque qu'on a pu resoudre 85% des instances fournis avec un nombre de violations qui est négligable par rapport à la taille des données en un temps qu'au maximum est égal à 50 sec.

2.4 Conclusion

Dans cette partie nous avons modélisé la problématique du sujet traité dans ce projet sous le format d'un CSP valué. Par suite, on a implementé et testé notre modélisation en utilisant la bibliothèque `pyToulBar2` et nous avons pu résoudre 85% des instances avec un nombre de violation minime.

Code

Le code que nous avons écrit pour ce projet (Partie 1 et Partie 2) peut être retrouvé sur le lien GitHub suivant : https://github.com/aymene98/Gestion-frequences