amU Faculté des sciences Aix Marseille Université

AIX-MARSEILLE UNIVERSITE
MASTER 2 INFORMATIQUE PARCOURS IAAA
UE MODÉLISATION ET RÉSOLUTION POUR L'OPTIMISATION

Résolution du problème d'allocation de fréquence, par COP et WCSP

9 décembre 2024

Nesrine GHANNAY

Kossi KOSSIVI

Encadré par : Cyril TERRIOUX

Table des matières

1 Introduction

Les problèmes de satisfaction de contraintes (CSP) sont des problèmes modélisés sous la forme d'un ensemble de contraintes à satisfaire qui permettent de résoudre des problèmes complexes et variés. Les CSP ont en général de nombreuses solutions candidates, mais pas toutes pertinentes, ainsi l'objectif de ce projet est d'explorer des méthodes plus avancées afin de trouver, pour une instance d'un problème, la meilleure solution possibles, si elle existe. Pour cela, nous allons adopter deux approches, les problèmes d'optimisation sous contraintes (COP) et les problèmes de satisfaction de contraintes valuées (WCSP). L'objectif sera de modéliser et de résoudre le problème d'allocation de fréquence entre des stations en utilisant ces formalismes, avec comme solveurs respectivement ACE [1] et CHOCO [2] pour la modélisation COP et Toolbar2 pour la modélisation WCSP [5].

Ainsi, on s'intéresse au problème d'allocation de fréquences entre des stations définit comme suit. On dispose de n stations réparties dans k régions distinctes. Chaque station est composée d'un émetteur et d'un récepteur et peut exploiter 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 Δ_{ij} 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 maximum de fréquences différentes utilisées pour la région i.

2 Problèmes d'optimisation sous contraintes

On souhaite résoudre le problème d'allocation de fréquences entre des stations en le modélisant dans un premier temps sous la forme d'un problème d'optimisation sous contraintes, Constraint Optimization Problem (COP) en anglais. Dans un second temps, on veut lancer la résolution et comparer l'efficacité des solveurs ACE et Choco. Chaque station est dotée de deux fréquences : une fréquence d'émission et une autre de réception.
On définit le problème COP à résoudre comme suit :

$$\mathcal{P} = (X, D, C, f)$$

avec,

- $X = \{e_1, e_2, \ldots, e_n\} \cup \{r_1, r_2, \ldots, r_n\}$ est l'ensemble des variables. Chaque station i possède deux variables :
  - e_i : la fréquence associée à l'émetteur de la station.
  - r_i : la fréquence associée au récepteur de la station.
- $D = \{d_{e_1}, d_{r_1}, \ldots, d_{e_n}, d_{r_n}\}$ est l'ensemble des domaines associés aux variables. Chaque domaine d_{x_i} contient les valeurs possibles pour la variable x_i, correspondant à un ensemble discret de fréquences disponibles.
- $C = \{w_1, w_2, \ldots, w_m\}$ est l'ensemble des contraintes, définies comme suit :
  1. **Écart matériel :** Pour des raisons matérielles, l'écart entre les fréquences de l'émetteur et

du récepteur d'une même station i doit être supérieur ou égal à δ_i.

$$|e_i - r_i| = \delta_i \tag{1}$$

2. **Interférences entre stations** Si deux stations i et j sont proches, leurs fréquences doivent être espacées d'au moins $\Delta_{i,j}$ pour éviter des interférences.

$$|e_i - e_j| \geq \Delta_{i,j} \tag{2}$$

$$|r_i - r_j| \geq \Delta_{i,j} \tag{3}$$

Nous supposons que les interférences entre stations ne peuvent survenir que pour des fréquences de même type. Bien que l'on aurait pu envisager de traiter également les interférences entre des fréquences de nature différente (e_i et r_j), nous avons privilégié cette première approche, que nous jugeons plus réaliste.

3. **Liaison directe** Il s'agit de la contrainte principale du problème : lier des stations entre elles. Si deux stations i et j sont en liaison directe, la fréquence émettrice de i doit correspondre à la fréquence réceptrice de j, et vice-versa.

$$e_i = r_j \tag{4}$$

$$r_i = e_j \tag{5}$$

4. **Nombre de fréquences par régions :** On souhaite limiter le nombre de fréquences utilisées par région. Pour cela, à chaque région k il est associé un nombre n_k de fréquences à ne pas dépasser. Grâce à la contrainte globale *NValues* [3], on arrive ainsi à formuler cette contrainte comme suit :

$$NValues(\{e_j \mid région(j) = k\}) + NValues(\{r_j \mid région(j) = k\}) \leq n_k \tag{6}$$

— f : une fonction objectif à définir selon le critère à optimiser

Il nous est proposé 3 critères d'optimisation. Chacune d'elle permet de définir une fonction objectif à optimiser selon les contraintes listées au-dessus.

Remarque : Étant donnée qu'aucune contrainte n'est explicitement posé sur les valeurs prises par les fréquences des stations d'une même région (à part la gestion des interférences) on n'en a modélisé aucune. On s'autorise donc à avoir dans nos solutions des stations d'une même région qui ont des fréquences émettrices et réceptrices similaires tant qu'il n'y a aucun souci lié au interférence entre stations, tant que les stations entre lesquelles nous voulons principalement établir des liaisons communiquent et tant que le nombre maximum de stations par régions est respecté. *« Ce qui n'est pas interdit par défaut est autorisé. »*

2.1 Minimiser le nombre de fréquences utilisées

2.1.1 Formalisation

On veut minimiser le nombre total de fréquences utilisées. Pour cela, on se munit de la contrainte globale ***NValues*** qui garantit que le nombre de valeurs distinctes prises par les variables d'une liste de

variables spécifiée respecte une condition numérique. Par exemple, le nombre de valeurs distinctes doit être inférieur ou égal à une constante. Ainsi, on a :

$$f = min(NValues(e + r)) \tag{7}$$

Avec $e + r$ représentant la réunion des deux variables qui contiennent les fréquences.

2.1.2 Résultats

Lorsque l'on résout les différentes instances fournies, on observe que le solveur ACE[**unknown**] est bien meilleur que le solveur CHOCO[2]. Pour toutes les instances, ACE arrive à trouver une solution et prouver son optimalité bien avant le timeout tandis que CHOCO pour quelques instances (8/19) parvient à trouver une solution optimale, mais dans une durée beaucoup plus élevée. CHOCO ne trouve donc pas de solutions pour 11 des 19 instances, celles dont la valeur de la fonction objectif vaut **-1** dans le Tableau 1. Pour l'instance `celar_150_13_15_5_0.800000_2` par exemple, ACE trouve la solution optimale en 4.48 secondes tandis que CHOCO parvient à la même solution en 236.13 secondes.

	ACE		**CHOCO**	
Nom de l'instance	Valeur f.o.	Durée (s)	Valeur f.o.	Durée (s)
`celar_150_13_15_5_0.800000_2`	4	3.42	4	490.37
`celar_150_13_15_5_0.800000_26`	4	5.51	**-1**	600.50
`celar_150_13_15_5_0.800000_28`	4	5.41	**-1**	600.64
`celar_150_13_15_5_0.800000_29`	4	4.01	**-1**	600.66
`celar_150_13_15_5_0.800000_8`	4	4.96	**-1**	600.69
`celar_250_25_15_5_0.820000_20`	4	6.09	**-1**	600.75
`celar_250_25_15_5_0.820000_22`	4	7.84	**-1**	600.89
`celar_250_25_15_5_0.820000_5`	4	4.36	**-1**	600.85
`celar_250_25_15_5_0.820000_7`	4	4.46	**-1**	600.73
`celar_250_25_15_5_0.820000_9`	4	5.03	**-1**	600.86
`celar_500_30_20_5_0.870000_24`	4	14.87	**-1**	601.04
`celar_500_30_20_5_0.870000_29`	4	11.76	**-1**	601.36
`celar_500_30_20_5_0.870000_45`	4	13.25	**-1**	601.07
`celar_500_30_20_5_0.870000_48`	4	10.28	**-1**	600.85
`celar_50_7_10_5_0.800000_0`	4	2.15	4	102.66
`celar_50_7_10_5_0.800000_1`	4	2.65	4	167.69
`celar_50_7_10_5_0.800000_6`	4	**2.01**	4	190.33
`celar_50_7_10_5_0.800000_7`	4	3.47	4	**99.45**
`celar_50_7_10_5_0.800000_8`	4	2.25	4	133.74

Table 1 – Récapitulatif des valeurs de la fonction objectif (minimiser le nombre de fréquences utilisées) et du temps de résolution en secondes pour les différentes instances avec les solveurs ACE et CHOCO. Les **-1** des colonnes Valeur f.o. sont associés aux instances qui n'ont pas été résolus avant la limite de temps.

Stations	**0 → 17**	**17 → 0**	**6 → 38**	**38 → 6**	**7 → 11**	**11 → 7**	**13 → 22**
Émetteur	0 : 280	17 : 14	6 : 280	38 : 14	7 : 280	11 : 14	13 : 280
Récepteur	17 : 280	0 : 14	38 : 280	6 : 14	11 : 280	7 : 14	22 : 280
Stations	**22 → 13**	**14 → 31**	**31 → 14**	**32 → 36**	**36 → 32**	**37 → 42**	**42 → 37**
Émetteur	22 : 14	14 : 14	31 : 280	32 : 84	36 : 350	37 : 280	42 : 14
Récepteur	13 : 14	31 : 14	14 : 280	36 : 84	32 : 350	42 : 280	37 : 14

Table 2 – Attribution des fréquences émettrice et récepteurs aux différentes stations entre lesquelles on souhaite établir les liaisons après résolution de l'instance celar_50_7_10_5_0.800000_1. Les flèches indiquent la direction des émissions.

2.1.3 Analyse d'une solution

Pour l'instance celar_50_7_10_5_0.800000_1, la solution optimale obtenue correspond aux fréquences uniques 280, 350, 84 et 14. En utilisant uniquement ces quatre fréquences, il est possible d'attribuer les fréquences des émetteurs et récepteurs de manière à assurer la réalisation des liaisons souhaitées.

2.2 Utiliser les fréquences les plus basses possibles

2.2.1 Modélisation

Ou souhaite maintenant utiliser les fréquences les plus basses possibles. On définit alors f comme suit :

$$f = min(Maximun(e + r)) \tag{8}$$

Idéalement, nous aurions souhaité minimiser la somme des valeurs prises par les variables e_i et r_i, mais le filtrage d'une telle contrainte s'est avéré trop coûteux en termes de calcul. Le solveur n'était pas en mesure de trouver une solution dans un temps réaliste. Nous avons donc opté pour une seconde approche consistant à minimiser la valeur maximale des fréquences utilisées.

2.2.2 Résultats

Avec ce nouveau critère d'optimisation, on constate que le solveur ACE résout parfaitement toutes les instances, tandis que le solveur CHOCO échoue à résoudre 9 instances sur 19. Lorsque CHOCO réussit à résoudre certaines instances, il met considérablement plus de temps que ACE pour trouver la solution et prouver son optimalité. Cela est illustré dans le Tableau 3, où CHOCO met 401.92 secondes pour trouver une solution optimale, contre seulement 2.95 secondes pour ACE. ACE s'avère être un meilleur solveur que CHOCO sur les instances et la modélisation étudiées.

2.3 Minimiser la largeur de la bande de fréquences utilisées

2.3.1 Modélisation

On veut ici minimiser l'écart entre la fréquence la plus haute utilisé et la plus basse.

$$f = min(Maximun(e + r) - Minimum(e + r)) \tag{9}$$

	ACE		CHOCO	
Nom de l'instance	Valeur f.o.	Durée (s)	Valeur f.o.	Durée (s)
celar_150_13_15_5_0.800000_2	392	2.95	392	92.92
celar_150_13_15_5_0.800000_26	294	2.61	294	461.91
celar_150_13_15_5_0.800000_28	392	2.68	392	269.15
celar_150_13_15_5_0.800000_29	322	2.95	322	401.92
celar_150_13_15_5_0.800000_8	308	3.38	308	210.07
celar_250_25_15_5_0.820000_20	252	4.37	**-1**	600.62
celar_250_25_15_5_0.820000_22	392	2.81	**-1**	600.61
celar_250_25_15_5_0.820000_5	252	2.88	**-1**	600.68
celar_250_25_15_5_0.820000_7	322	3.04	**-1**	600.72
celar_250_25_15_5_0.820000_9	252	3.29	**-1**	600.61
celar_500_30_20_5_0.870000_24	308	4.31	**-1**	600.89
celar_500_30_20_5_0.870000_29	252	4.33	**-1**	600.87
celar_500_30_20_5_0.870000_45	322	4.40	**-1**	600.92
celar_500_30_20_5_0.870000_48	252	4.44	**-1**	600.76
celar_50_7_10_5_0.800000_0	294	1.41	294	50.32
celar_50_7_10_5_0.800000_1	322	**1.25**	322	19.22
celar_50_7_10_5_0.800000_6	322	1.43	322	17.55
celar_50_7_10_5_0.800000_7	210	1.62	210	**14.81**
celar_50_7_10_5_0.800000_8	224	1.51	224	32.07

Table 3 – Récapitulatif des valeurs de la fonction objectif (utiliser les fréquences les plus basses possibles donc la colonne contient la fréquence de valeur maximale) et du temps de résolution en secondes pour les différentes instances avec les solveurs ACE et CHOCO. Les **-1** des colonnes Valeur f.o. sont associés aux instances qui n'ont pas été résolus avant la limite de temps.

2.3.2 Résultats

Comme pour les deux premiers problèmes, ACE affiche de meilleures performances que CHOCO en termes de nombre d'instances résolues (100% pour ACE contre 8/19 pour CHOCO) et en temps de résolutions moyens. Les solutions obtenues sont bien des configurations dans lesquelles l différence entre la fréquence maximale et la fréquence minimale est la plus basse.

2.4 Bilan

En résumé, notre modélisation du problème d'attribution de fréquences, permet de trouver une solution optimale dans un temps réaliste avec le solveur ACE pour 100% des instances (données) qui nous ont été fournies. Les configurations obtenues pour chaque critère d'optimisation assurent la bonne liaison entre les différentes stations. Le solveur CHOCO semble être beaucoup moins performant que le solveur ACE en termes de temps d'exécution et d'efficacité. On remarque qu'il n'arrive pas à trouver de solutions précisément pour les instances possédant un nombre de variables important, donc plus de contraintes (voir Table 5). Cela pourrait être lié au filtrage des contraintes globales utilisées dans notre

	ACE		CHOCO	
Nom de l'instance	Valeur f.o.	Durée (s)	Valeur f.o.	Durée (s)
celar_150_13_15_5_0.800000_2	378	4.48	378	236.13
celar_150_13_15_5_0.800000_26	280	3.26	**-1**	600.53
celar_150_13_15_5_0.800000_28	378	3.07	378	442.83
celar_150_13_15_5_0.800000_29	308	3.22	308	306.70
celar_150_13_15_5_0.800000_8	294	4.03	**-1**	600.56
celar_250_25_15_5_0.820000_20	238	4.62	**-1**	600.92
celar_250_25_15_5_0.820000_22	378	3.56	**-1**	600.57
celar_250_25_15_5_0.820000_5	238	5.13	**-1**	600.85
celar_250_25_15_5_0.820000_7	308	5.07	**-1**	600.69
celar_250_25_15_5_0.820000_9	238	4.34	**-1**	600.60
celar_500_30_20_5_0.870000_24	294	9.94	**-1**	600.96
celar_500_30_20_5_0.870000_29	238	14.15	**-1**	600.89
celar_500_30_20_5_0.870000_45	308	14.02	**-1**	600.92
celar_500_30_20_5_0.870000_48	238	8.60	**-1**	600.79
celar_50_7_10_5_0.800000_0	280	2.31	280	19.62
celar_50_7_10_5_0.800000_1	308	1.96	308	**17.28**
celar_50_7_10_5_0.800000_6	308	1.99	308	37.20
celar_50_7_10_5_0.800000_7	196	**1.68**	196	66.38
celar_50_7_10_5_0.800000_8	210	2.16	210	41.27

Table 4 – Récapitulatif des valeurs de la fonction objectif (minimiser la largeur de la bande de fréquences utilisées donc la colonne contient la différence entre la valeur maximale et la valeur minimale) et du temps de résolution en secondes pour les différentes instances avec les solveurs ACE et CHOCO. Les **-1** des colonnes Valeur f.o. sont associés aux instances qui n'ont pas été résolu avant la limite de temps.

modélisation, notamment *NValues* mais cette hypothèse reste à vérifier, ce qui n'est pas le sujet de ce projet. Toutes les résolutions ont été effectuées sur une machine équipée de 16 Go de RAM et d'un processeur comportant 8 cœurs CPU cadencés à environ 3 GHz.

Type d'instance	`celar...800000_X`	`celar...820000_X`	`celar...860000_X`
Nombre de stations	50, 150	250	500
ACE	**10/10**	**5/5**	**4/4**
CHOCO	≈ 7.6/10	0/5	0/4

Table 5 – Nombre d'instances résolues en moyenne par chaque solveur en fonction du type d'instances considéré, type définis par le nombre de stations

3 Problèmes de satisfaction de contraintes valuées

Dans cette seconde partie du projet, on s'intéresse à la modélisation du problème d'allocation de fréquence sous forme de problème de Satisfaction de Contrainte Valuées (Valued Constraint Satisfaction Problem, VCSP). Cette approche est particulièrement utile dans les cas où toutes les contraintes ne peuvent pas être satisfaites simultanément, ou lorsque certaines solutions sont préférables à d'autres en fonction d'un critère quantifiable.

3.1 Modélisations

Le problème WCSP est défini comme un quadruplet :

$$P = (X, D, W, SV)$$

où :

- $X = \{e_1, e_2, \ldots, e_n\} \cup \{r_1, r_2, \ldots, r_n\}$ est l'ensemble des variables. Chaque station i possède deux variables :
  - e_i : la fréquence associée à l'émetteur de la station.
  - r_i : la fréquence associée au récepteur de la station.
- $D = \{D(e_1), D(r_1), \ldots, D(e_n), D(r_n)\}$ est l'ensemble des domaines associés aux variables. Chaque domaine $D(x_i)$ contient les valeurs possibles pour la variable x_i, correspondant à un ensemble discret de fréquences disponibles.
- $W = \{w_1, w_2, \ldots, w_m\}$ est l'ensemble des contraintes pondérées, définies comme suit :

1. **Contrainte 1 : écart matériel** Pour des raisons matérielles, l'écart entre les fréquences de l'émetteur et du récepteur d'une même station i doit être supérieur ou égal à δ_i. Formulation :

$$w_1(e_i, r_i) = \begin{cases} 0, & \text{si } |e_i - r_i| \geq \delta_i, \\ \textit{coût maximal} & \text{sinon (contrainte dure).} \end{cases}$$

   Étant donné que la raison de cette contrainte est due à des nécessités du point de vue matériel, nous sommes obligés de la satisfaire.

2. **Contrainte 2 : interférences entre stations** Si deux stations i et j sont proches, leurs fréquences doivent être espacées d'au moins $\Delta_{i,j}$ pour éviter des interférences. Formulation pour les émetteurs :

$$w_2(e_i, e_j) = \begin{cases} 0, & \text{si } |e_i - e_j| \geq \Delta_{i,j}, \\ 0.1 * \textit{coût maximal} * (|\Delta_{i,j} - |e_i - e_j||), & \text{sinon.} \end{cases}$$

   Une contrainte similaire s'applique aux récepteurs r_i et r_j.
   Nous avons décidé que cette contrainte soit une contrainte souple, car nous considérons que des interférences peuvent avoir lieu. Cependant, nous pénalisons tout de même ces interférences dans l'objectif de les minimiser. La pénalisation est proportionnelle à l'écart manquant pour atteindre la distance minimale $\Delta_{i,j}$, ce qui permet d'intégrer un compromis entre la satisfaction des contraintes strictes et la recherche d'une solution réalisable.

3. **Contrainte 3 : liaison directe** Si deux stations i et j sont en liaison directe, la fréquence émettrice de i doit correspondre à la fréquence réceptrice de j, et vice-versa. Formulation

pour e_i et r_j :

$$w_3(e_i, r_j) = \begin{cases} 0, & \text{si } e_i = r_j, \\ 0.2 * \textit{coût maximal}, & \text{sinon.} \end{cases}$$

Une contrainte symétrique s'applique pour e_j et r_i.

Cette contrainte est également une contrainte souple, car nous considérons que des écarts entre e_j et r_i peuvent être tolérés dans certaines situations où il est difficile de satisfaire toutes les contraintes strictement. Cependant, nous pénalisons toute différence entre e_j et r_i afin d'encourager des solutions respectant les correspondances requises pour les liaisons directes.

— $SV = (\mathcal{N} \cup \{+\infty\}, \leq, +, 0, +\infty)$

L'objectif est de trouver une affectation $A : X \to D$ qui minimise le coût total :

$$\text{Minimiser} \sum_{j=1}^{m} w_j(A[S_j])$$

où $A[S_j]$ est l'affectation des variables dans le scope S_j de la contrainte w_j.

Explication des valuations choisies

Nous avons essayé de choisir des valuations cohérentes pour chaque contrainte, et ce, de manière proportionnel notamment au nombre de stations du problème à traiter.

— Concernant le choix du *coût maximal*, nous avons décidé d'opter pour une valeur permettant de pondérer les pénalités de manière proportionnelle à la taille du problème. Il est défini comme 10 fois le nombre de stations, garantissant ainsi que les valuations des contraintes sont adaptées à l'échelle du problème.

— Pour la *Contrainte 1 : écart matériel*, comme expliqué précédemment, il est essentiel de la considérer comme une contrainte dure avec un coût maximal en cas de violation. Cette décision se justifie par le fait que les contraintes matérielles sont jugées obligatoires et ne peuvent être ignorées.

— Pour la *Contrainte 2 : interférences entre stations*, nous avons opté pour une contrainte souple avec une pénalisation proportionnelle au degré de violation, c'est-à-dire à quel point la distance entre les fréquences des deux stations i et j est éloignée de la distance attendue $\Delta_{i,j}$. De plus, on pondère la pénalisation par $0.1 * \textit{coût maximal}$ afin de minimiser les interférences en offrant une pénalisation graduelle selon l'intensité de la violation, et ce, de manière proportionnelle au problème.

— Enfin, pour la *Contrainte 3 : liaisons entre les stations*, nous avons choisi comme précédemment une contrainte souple avec une pénalisation en cas de non-correspondance des fréquences, pondérée par 0.2 * coût maximal. Ce choix s'explique par le fait que qu'il est important de maintenir des liaisons directes entre les stations requises. Et il est préférable d'avoir des communications perturbées plutôt que de ne pas avoir de communication entre deux stations demandées.

3.2 Implémentation

Afin de mettre en œuvre le formalisme WCSP présenté dans la sous-section précédente et le rendre compatible avec le solveur Toulbar2, nous générons un fichier `.wcsp` structuré de la manière suivante :

— La première ligne du fichier contient les informations générales du problème : `freq_alloc`

`<nombre_de_variables> <taille_max_des_domaines> <nombre_de_contraintes> <coût_maximal>`.

— Ensuite, nous énumérons la taille des domaines de chaque variable, c'est-à-dire le nombre de fréquences que peut prendre une variable émettrice ou réceptrice d'une station.

— Pour chaque contrainte, nous définissons : son arité, les variables impliquées, le coût par défaut de la contrainte et le nombre de tuples concernés par la contrainte.

— Enfin, nous énumérons pour chaque variable les tuples candidats satisfaisant la contrainte avec leur coût associé (0), ainsi que les tuples ne satisfaisant pas la contrainte avec la valuation définie pour cette violation.

Ce fichier `.wcsp` peut ensuite être utilisé avec `pytoulbar2` pour résoudre le problème de satisfaction de contraintes valuées.

Méthodologie de Résolution : `pytoulbar2`

Les instances WCSP sont résolues à l'aide du solveur Toulbar2 utilisée via son interface Python `pytoulbar2`. Les calculs ont été réalisés sur Google Colab en utilisant exclusivement des ressources CPU (RAM du système 1.4 / 12.7 GB ; Disque 32.7 / 107.7 GB). De plus, les résolutions sont effectuées via un processus distinct pour chaque instance afin de mieux gérer les limites de temps et minimiser les interférences entre les tâches (gérer par la classe `MultiCFN` de `pytoulbar2`, permettant combiner linéairement plusieurs CFN) [4] [6].

3.3 Résultats

Dans cette section, nous présentons les résultats obtenus en utilisant le solveur `pytoulbar2` pour résoudre les instances du problème WCSP.

Les résultats présents dans le tableau 6 montrent que le solveur `pytoulbar2` est capable de résoudre de manière efficace les instances du problème WCSP d'allocation de fréquence. Il parvient à faire une résolution avec des temps variant selon la complexité des instances traitées. En effet, pour des petites instances telles que `celar_50_7_10_5_0.800000_X` (instance avec peu de station), les solutions sont trouvées rapidement avec un coût minimal. Certaines grosses instances telles que `celar_500_30_20_5_0.870000_X` n'ont pas pu être résolues dans le temps imparti, probablement en raison de leur complexité (ces instances comprennent un nombre élevé de stations, de variables et de contraintes). Par ailleurs, l'utilisation d'un processus indépendant pour chaque résolution assure une gestion efficace du délai `max_time`.

Ainsi, les résultats mettent en lumière l'efficacité de `pytoulbar2` pour résoudre des problèmes complexes, en particulier pour les instances de taille modérées. Cependant, pour de très grandes instances, les performances sont limitées par une taille de l'espace de recherche qui croît rapidement. Néanmoins, `pytoulbar2` reste un outil puissant pour résoudre des problèmes WCSP. Nous pouvons même envisager de rajouter davantage de contrainte au problème, telles que la contrainte de régions présentée dans la partie 2.

Nom de l'instance	Coût maximal	Résultat	Temps d'exécution
celar_150_13_15_5_0.800000_2	1500	0.0	0.84
celar_150_13_15_5_0.800000_9	1500	300.0	3.11
celar_150_13_15_5_0.800000_18	1500	300.0	1.67
celar_150_13_15_5_0.800000_20	1500	600.0	3.87
celar_150_13_15_5_0.800000_25	1500	0.0	0.94
celar_250_25_15_5_0.820000_0	2500	500.0	6.28
celar_250_25_15_5_0.820000_6	2500	500.0	3.84
celar_250_25_15_5_0.820000_8	2500	500.0	4.40
celar_250_25_15_5_0.820000_17	2500	500.0	1.34
celar_250_25_15_5_0.820000_29	2500	Aucune solution	-
celar_500_30_20_5_0.870000_0	5000	Aucune solution	-
celar_500_30_20_5_0.870000_8	5000	0.0	1.96
celar_500_30_20_5_0.870000_25	5000	Aucune solution	-
celar_500_30_20_5_0.870000_30	5000	Aucune solution	-
celar_500_30_20_5_0.870000_49	5000	Aucune solution	-
celar_50_7_10_5_0.800000_0	500	300.0	0.34
celar_50_7_10_5_0.800000_1	500	100.0	0.48
celar_50_7_10_5_0.800000_4	500	100.0	0.22
celar_50_7_10_5_0.800000_9	500	100.0	0.22
celar_50_8_10_5_0.800000_8	500	100.0	0.25

Table 6 – Performance de résolution d'instances du problème WCSP par le solveur Toulbar2

4 Conclusions

Nous avons vu que le problème d'allocation de fréquences entre stations pouvait être modélisé comme un problème d'optimisation sous contraintes de manière à obtenir des solutions optimales en un temps raisonnable à condition d'utiliser le solveur adapté. Ainsi, on a pu traiter l'ensemble des données qui nous ont été fournies. Notre modélisation et la nature des instances à résoudre (nombre de variables et de contraintes) ont fait du solveur ACE un meilleur solveur que CHOCO (dans ce contexte spécifique). Dans un second temps, on s'est intéressé à la modélisation du problème sous la forme d'un problème de satisfaction de contraintes valuées, que nous avons ensuite résolu à l'aide du solveur Toulbar2. Le principal défi résidait dans la définition d'une fonction de coût adéquate. Néanmoins, notre modélisation s'est avérée efficace, car elle nous a permis de trouver des solutions pour la majorité des instances étudiées.

La modélisation et les tentatives de résolutions du problème d'allocations de fréquences entre stations, à l'aide de ces différents solveurs, nous aura permis de mettre en pratique les différentes connaissances acquises lors des cours de Modélisation et Résolution pour l'Optimisation. Ce projet nous aura également permis de développer de nouvelles compétences en modélisation de problèmes et nous familiariser avec l'utilisation des différents solveurs.

Références

[1] Christophe Lecoutre. *ACE, a generic constraint solver*. Jan. 2023. DOI : 10.48550/arXiv.2302.05405.

[2] Charles Prud'homme et Jean-Guillaume Fages. « Choco-solver : A Java library for constraint programming ». In : *Journal of Open Source Software* 7.78 (2022), p. 4708. DOI : 10.21105/joss.04708. URL : https://doi.org/10.21105/joss.04708.

[3] PyCSP. *NValues Constraints*. Accessed : 2024-12-09. 2024. URL : https://pycsp.org/documentation/constraints/NValues/.

[4] *pytoulbar2.MultiCFN Documentation*. Accessed : 2024-12-09. URL : https://toulbar2.github.io/toulbar2/ref/ref_python.html#pytoulbar2.MultiCFN.

[5] Francesca Rossi, Peter van Beek et Toby Walsh. *Handbook of Constraint Programming*. Elsevier, 2006.

[6] *toulbar2 GitHub Repository*. Accessed : 2024-12-09. URL : https://github.com/toulbar2/toulbar2.