Projet d'exploration: nombres de Ramsey

IFT-4001
Optimisation Combinatoire

Travail remis à
Claude-Guy Quimper

Alexis Roberge
Olivianne Vaillancourt

Automne 2025
Remis le 14 décembre 2025

# Introduction

La théorie de Ramsey est une branche de la combinatoire s'intéressant à l'apparence d'ordre dans une structure, d'habitude un graphe, lorsque celle-ci devient assez grande. Le théorème de Ramsey, un résultat fondamental du domaine, stipule que tout coloriage des arêtes d'un graphe complet suffisamment grand engendre des sous-graphes complets (cliques) monochromatiques. Dans le cadre de notre projet d'optimisation combinatoire, l'on s'intéresse en particulier aux nombres de Ramsey, qui décrivent la taille de graphe minimale nécessaire pour garantir leur existence. Leur calcul demande souvent l'appel à diverses méthodes computationnelles, puisque les résultats de la théorie de Ramsey tendent à décrire des objets de tailles extrêmes, et les démonstrations sont souvent non constructives. La recherche par force brute demeure parfois la seule ou la meilleure option, ce qui peut être formulé comme un problème de satisfaction de contraintes ou d'optimisation. L'on cherche à approcher le problème en adaptant ou en modifiant des méthodes décrites dans la littérature, puis l'on compare quantitativement celles les plus prometteuses afin d'évaluer leur pertinence générale à la résolution de problèmes de type Ramsey.

# Description du problème

Le problème se ramène à la recherche d'un coloriage de Ramsey $R(r, s; n)$ sur les arêtes d'un graphe complet à $n$ sommets, qu'on assigne à une de deux couleurs (dans notre modèle, *true* et *false*). L'on représente un coloriage par une matrice d'adjascence $C_{n\times n}$ où $C_{i,j}$ correspond à la couleur de l'arête liant les noeuds $i$ et $j$. De tels coloriages doivent n'avoir aucun sous-graphe complet (clique) de taille $r$ dont toutes les arêtes sont de la première couleur, et pareillement aucune clique de taille $s$ de la deuxième couleur. Si un tel coloriage peut être trouvé, on peut dire que $R(r, s) > n$. Sinon, on conclut que $R(r, s) \leq n$. On note qu'augmenter la borne inférieure est généralement plus facile que de rabaisser la borne supérieure, puisqu'un seul coloriage valide doit être trouvé pour le premier, alors que le second demande de parcourir l'arbre de recherche entier. Un article régulièrement mis à jour de l'*Electronic Journal of Combinatorics* recense les dernières avancées sur les nombres de Ramsey, dont les bornes inférieures et supérieures connues.[6]

L'on peut également généraliser le problème pour des coloriages de $k$ couleurs $R(c_1, c_2, ..., c_k; n)$, qui ne contient pas de clique monochromatique de taille $c_i$ et couleur $i$ pour $i = 1, ..., k$. Nous avons expérimenté avec des modèles à trois couleurs, et un modèle SAT de ce cas est fourni dans la remise. Notre analyse qui suit se concentre toutefois sur nos deux modèles les plus prometteurs, restreints au problème classique à deux couleurs.

# Approches proposées

## Première méthode: modèle SAT

On décrit d'abord un coloriage de Ramsey $R(r, s; n)$ valide comme un modèle SAT, qu'on résout avec OR Tools CP SAT 9.12.4544. Les tableaux $R$ et $S$ contenant l'ensemble des cliques possibles sont générés par le script *clique_creator.py*. Lorsque les cliques sont générées, la description du modèle s'avère simple, et ressemble à celle décrite pour le calcul exact de $R(4, 3, 3)$. [2]

### Ensembles et constantes

$$
\begin{aligned}
dom(n) &= \mathbb{N} \\
dom(r) &= \mathbb{N} \\
dom(s) &= \mathbb{N} \setminus \{1, ..., r-1\} \\
V &= \{1, ..., n\} && \text{Ensemble des noeuds} \\
n_r &= \binom{n}{r} = \frac{n!}{r!(n-r)!} && \text{Nombre de cliques de taille r} \\
n_w &= \binom{n}{s} = \frac{n!}{s!(n-s)!} && \text{Nombre de cliques de taille s} \\
dom(R_{i,j}) &= \mathbb{N} && \forall i \in \{1, ..., n_r\}, \forall j \in \{1, ..., r\} \\
dom(S_{i,j}) &= \mathbb{N} && \forall i \in \{1, ..., n_s\}, \forall j \in \{1, ..., s\}
\end{aligned}
$$

### Variables

$$
\text{Coloriage :} \qquad dom(C_{i,j}) = \mathbb{B} \qquad\qquad \forall i, j \in V
$$

### Contraintes

L'on pose que la matrice de coloriage ait des *false* sur la diagonale, et qu'elle soit symétrique.

$$(1) \qquad C_{i,i} = false \qquad \forall i \in V$$
$$(2) \qquad C_{i,j} = C_{j,i} \qquad \forall i, j \in V, i < j$$

On vérifie qu'il n'existe pas de cliques monochromatiques de taille r et couleur $false$, et de la même façon, on vérifie les cliques de taille $s$ pour ne pas y avoir de cliques de couleur $true$.

$$(3) \qquad \bigvee_{1 \leq j < k \leq r} C_{R_{i,j}, R_{i,k}} \qquad \forall i \in \{1, ..., n_r\}$$
$$(4) \qquad \bigvee_{1 \leq j < k \leq s} \neg C_{S_{i,j}, S_{i,k}} \qquad \forall i \in \{1, ..., n_s\}$$

### Bris de symétrie

En raison du grand nombre de symétries naturellement présentes dans les coloriages de graphes complets, plusieurs contraintes peuvent être ajoutées pour réduire la taille de l'arbre de recherche ainsi que le nombre de solutions isomorphes obtenues. Parmi celles testées, celle décrite par Codish et al.[5] s'est avérée de loin la plus efficace (contrainte 5). Elle décrit un ordre lexicographique amélioré $\preceq_{\{i,j\}}$ où l'on impose l'ordre de deux rangées de la matrice du coloriage après avoir enlevé les caractères aux index $i$ et $j$. L'on examine son impact sur le temps de résolution ainsi que le temps de première solution dans la discussion.

$$(5) \qquad \bigwedge_{1 \leq i < j \leq n} C_i \preceq_{\{i,j\}} C_j$$

## Deuxième méthode: optimisation locale

La conception de la deuxième méthode a été motivée par l'espoir de réduire le temps de recherche d'un coloriage pour de grandes instances. On propose un modèle d'optimisation inspiré d'un algorithme de recherche randomisé décrit par Exoo en 2023, mais qui était déjà bien connu dans la littérature.[3] Notre méthode exploite le mode de recherche locale de OR

Tools CP SAT 9.12.4544 (voir figure 1 dans l'annexe). Le mode *Large Neighborhood Search*[1] a également été testé pour ce modèle, et offre une performance située entre la recherche locale et la configuration par défaut.
On ajoute les tableaux $R'_{i,j}$ et $S'_{i,j}$ qui pointent vers des rangées des tableaux de cliques, et contiennent les cliques contenant une arête donnée.

**Ensembles et constantes**

$$dom(n) = \mathbb{N}$$
$$dom(r) = \mathbb{N}$$
$$dom(s) = \mathbb{N} \setminus \{1, ..., r-1\}$$
$$e = \frac{n(n-1)}{2} \quad \text{Nombres d'arêtes du graphe}$$
$$E = \{1, ..., e\} \quad \text{Ensemble des arêtes}$$
$$V = \{1, ..., n\} \quad \text{Ensemble des noeuds}$$
$$n_r = \binom{n}{r} = \frac{n!}{r!(n-r)!} \quad \text{Nombre de cliques de taille } r$$
$$n_s = \binom{n}{s} = \frac{n!}{s!(n-s)!} \quad \text{Nombre de cliques de taille } s$$
$$dom(R_{i,j}) = \mathbb{N} \quad \forall i \in \{1, ..., n_r\}, \forall j \in \{1, ..., r\}$$
$$dom(S_{i,j}) = \mathbb{N} \quad \forall i \in \{1, ..., n_s\}, \forall j \in \{1, ..., s\}$$
$$n'_r = \binom{n-2}{r-2} = \frac{(n-2)!}{(r-2)!(n-r)!} \quad \text{Nombre de cliques locales de taille } r$$
$$n'_s = \binom{n-2}{s-2} = \frac{(n-2)!}{(s-2)!(n-s)!} \quad \text{Nombre de cliques locales de taille } s$$
$$dom(R'_{i,j}) = \mathbb{N} \quad \forall i \in E, \forall j \in \{1, ..., n'_r\}$$
$$dom(S'_{i,j}) = \mathbb{N} \quad \forall i \in E, \forall j \in \{1, ..., n'_s\}$$
$$\rho_r = s \quad \text{Poids pour } r \text{ dans la fonction objectif}$$
$$\rho_s = r \quad \text{Poids pour } s \text{ dans la fonction objectif}$$

### Variables

On ajoute plusieurs variables par rapport au modèle 1. On indique dans les tableaux $P_i$ et $Q_i$ si la clique $i$ des tableaux $R$ et $S$ respectivement est monochromatique. Ensuite, dans les tableaux $X$ et $Y$, on compte pour chaque arête $i$ le nombre de cliques monochromatiques de couleur $r$ et $s$ qui la contiennent, et on utilise finalement ces tableaux pour calculer la fonction objectif.

| | | |
|---|---|---|
| Coloriage : | $dom(C_{i,j}) = \mathbb{B}$ | $\forall i, j \in V$ |
| Cliques monochr. $r$ : | $dom(P_i) = \mathbb{B}$ | $\forall i \in \{1, ..., n_r\}$ |
| Cliques monochr. $s$ : | $dom(Q_i) = \mathbb{B}$ | $\forall i \in \{1, ..., n_s\}$ |
| Nbr. cliques monochr. $r$ locales : | $dom(X_i) = \{0, ..., n'_r\}$ | $\forall i \in E$ |
| Nbr. cliques monochr. $s$ locales : | $dom(Y_i) = \{0, ..., n'_s\}$ | $\forall i \in E$ |
| Total cliques monochr. $r$ : | $dom(x) = \mathbb{N}$ | |
| Total cliques monochr. $s$ : | $dom(y) = \mathbb{N}$ | |
| Fonction objectif : | $dom(f) = \mathbb{N}$ | |

### Contraintes

On utilise les mêmes contraintes que le modèle SAT pour dupliquer la matrice triangulaire et s'assurer que la diagonale est vide.

$$(1) \qquad C_{i,i} = false \qquad \forall i \in V$$

$$(2) \qquad C_{i,j} = C_{j,i} \qquad \forall i, j \in V, i < j$$

Pour savoir si une clique est monochromatique, on vérifie la couleur de toutes ses arêtes. On garde ces résultats dans les tableaux de booléens correspondants (selon $r$ ou $s$ pour la taille de la clique).

$$(3) \qquad P_i = \bigwedge_{1 \leq j < k \leq r} \neg C_{R_{i,j}, R_{i,k}} \qquad \forall i \in \{1, ..., n_r\}$$

$$(4) \qquad Q_i = \bigwedge_{1 \leq j < k \leq s} C_{S_{i,j}, S_{i,k}} \qquad \forall i \in \{1, ..., n_s\}$$

Pour faciliter le branchement du solveur vers des modifications locales on compte pour chaque arête le nombre de cliques monochromatiques qui la

contiennent.

$$X_i = |\{P_{R_{i,j}} \mid P_{R_{i,j}} = true, j \in \{1, ..., n'_r\}\}| \qquad \forall i \in E \tag{5}$$

$$Y_i = |\{Q_{S_{i,j}} \mid P_{S_{i,j}} = true, j \in \{1, ..., n'_s\}\}| \qquad \forall i \in E \tag{6}$$

Finalement, on calcule la fonction objectif à minimiser, qui correspond à la somme pondérée (contrainte 9) des sommes des tableaux de cliques monochromatiques locales de taille $r$ et $s$ (contraintes 7 et 8).

$$x = \sum_{k=1}^{e} X_k \tag{7}$$

$$y = \sum_{k=1}^{e} Y_k \tag{8}$$

$$f = \rho_r * x + \rho_s * y \tag{9}$$

**Heuristiques**

Afin de converger plus rapidement vers un coloriage de Ramsey valide, l'on assigne un état initial $\hat{C}$ à la matrice de coloriage $C$ (dans MiniZinc, cela se fait avec *warm-start*), tel que:

$$\begin{aligned} \hat{C}_{i,i} &= false && \forall i \in V \\ \hat{C}_{i,j} &= \hat{C}_{j,i} && \forall i, j \in V \\ \mathbb{P}(\hat{C}_{i,j} = false) &= \frac{r}{r+s} && \text{pour } 1 \leq i < j \leq n \end{aligned}$$

Ce choix de probabilité est motivé par une estimation de la fréquence relative de chaque couleur dans des coloriages de Ramsey extrémaux (de taille $R(s,t) - 1$) tirés de jeux de donnés connus.[4] Exoo avait posé une probabilité de 0.47 pour *false* dans la recherche de $R(5, 6; 58)$[3] alors que nous aurions $\frac{5}{5+6} \approx 0.456$. Plusieurs heuristiques de recherche ont été testées sans que l'on puisse noter d'amélioration substantielle, donc aucune annotation supplémentaire n'est fournie au solveur.

# Protocole d'expérimentation

Tous les fichiers de données utilisés dans notre projet sont générés à l'aide du programme Python *clique_creator.py* fourni. Pour un nombre de noeuds

$n$, et les tailles de cliques $r$, $s$, $t$, le programme trouve tous les sous-graphes complets possibles pour chaque taille différente et crée le fichier .dzn correspondant. L'ensemble des cliques est représenté par un tableau 2D de taille $C(n, k)$ par $k$, où $k$ est la taille du sous-graphe (soit $r$, $s$ ou $t$). Par exemple, une rangée du tableau qui vaut $[1, 5, 6, 9, 10]$ correspond à une clique composée des noeuds 1, 5, 6, 9 et 10. Lorsque le mode local est activé, des tableaux de cliques locales sont également générés (utilisés dans le deuxième modèle). Pour chaque test effectué, l'on doit créer de nouveaux fichiers de données.

Nous évaluons le temps requis pour trouver un premier coloriage de Ramsey pour le modèle SAT sans et avec bris de symétrie, et pour le modèle d'optimisation locale. On évalue également pour le modèle SAT, le temps pour prouver qu'il n'en existe aucun. On se limite à une période d'une heure pour l'exécution.

# Résultats

## Modèle SAT sans bris de symétrie

Dans les tableaux de données ci-dessous, les points d'interrogations signifient que le solveur n'a pas pu terminer en moins d'une heure, soit pour trouver un coloriage ou pour prouver qu'il n'en existe pas. Les cases en bleu correspondent aux valeurs exactes connues du nombre de Ramsey correspondant. Nous utilisons le solveur OR Tools avec 12 threads et l'option "free search" pour tous nos résultats obtenus.

**Temps de résolution pour $R(3, s; n)$**

| s \ n | 14 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 5 | 7 m | | | | | | | | | | | | |
| 6 | // | 3 s | ? | | | | | | | | | | |
| 7 | // | 3 s | 5 s | 9 s | 12 s | 23 s | 35 s | ? | | | | | |
| 8 | // | 6 s | 9 s | 15 s | 26 s | 42 s | 1 m | 1.5 m | 3.5 m | 5 m | 7 m | 11 m | ? |
| 9 | // | 6 s | 13 s | 25 s | 44 s | 1.5 m | 2 m | 4 m | 7.5 m | 11 m | 19 m | ? | ? |

**Temps de résolution pour $R(4, s; n)$**

| s \ n | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 |
|---|---|---|---|---|---|---|---|---|---|
| 4 | $< 1$ s | 1 h | | | | | | | |
| 5 | 1s | 1.5 s | 3 s | 3 s | 3.5 s | 4.5 s | 5 s | 6.5 s | ? |

**Temps de résolution pour $R(3, 3, t; n)$**

| t \ n | 16 | 17 | 25 | 26 | 27 | 28 | 29 | 30 |
|---|---|---|---|---|---|---|---|---|
| 3 | $< 1$ s | 7 s | | | | | | |
| 4 | // | // | 2.5 s | 3 m | 21 m | 58 m | ? | ? |

## Modèle avec bris de symétrie

Ici, on utilise le même banc d'essai, mais l'on rajoute une contrainte lexicographique qui brise la symétrie de nos coloriages dans le modèle SAT. À noter que ce modèle peut aussi prouver que $R(4, 4) = 18$ en 5 secondes, mais ne peut toujours pas montrer que $R(4, 5) = 25$ en moins d'une heure.

**Temps de résolution pour $R(3, s; n)$**

| s \ n | 14 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 5 | 1 s | | | | | | | | | | | | |
| 6 | // | 3 s | 5 s | | | | | | | | | | |
| 7 | // | 4 s | 6 s | 18 s | 28 s | 48 s | 1.5 m | 2.5 m | | | | | |
| 8 | // | 6 s | 10 s | 17 s | 1.5 m | 1 m | 2 m | 3 m | 4.5 m | 6.5 m | 11.5 m | 16 m | ? |
| 9 | // | 8 s | 14 s | 27 s | 1 m | 2 m | 2.5 m | 6 m | 11 m | 15.5 m | 26 m | ? | ? |

## Modèle d'optimisation locale

Voici les données des temps d'exécution pour trouver un coloriage de Ramsey avec le modèle *Ramsey_optimisation_local.mzn*. Ce modèle est conçu pour trouver heuristiquement un coloriage, et non parcourir l'ensemble des coloriages possibles.

**Temps de résolution pour $R(3, s; n)$**

| s \ n | 14 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 5 | N/A | | | | | | | | | | | | |
| 6 | // | 8 s | N/A | | | | | | | | | | |
| 7 | // | 16 s | 27 s | 42 s | 1 m | 1.5 m | 2.5 m | N/A | | | | | |
| 8 | // | 29 s | 51 s | 1.5 m | 2.5 m | 4 m | 7 m | 10.5 m | 16.5 m | 24.5 m | ? | ? | N/A |
| 9 | // | 36 s | 1 m | 2.5 m | 7.5 m | 24.5 m | 41.5 m | ? | ? | ? | ? | ? | ? |

# Discussion

## Temps d'exécution

Notre modèle de base *Ramsey_SAT.mzn* est généralement le plus rapide pour trouver un premier coloriage de Ramsey. Puisque ce modèle est très peu contraignant, nous n'enlevons aucune solution possible, ce qui semble accélèrer la recherche d'un premier coloriage valide quelconque. Cependant, on constate qu'il est considérablement plus lent lorsqu'il doit parcourir tout l'arbre de recherche pour calculer exactement un nombre de Ramsey. Le modèle avec bris de symétrie s'avère plus lent pour trouver un premier coloriage valide, mais son arbre de recherche est beaucoup plus petit, ce qui nous permet de calculer des nombres de Ramsey exacts jusqu'à $R(3,7) = 23$.

Malheureusement, la complexité du modèle d'optimisation le rend beaucoup plus lent, surtout pour de grandes instances, et incapable de trouver un coloriage pour certaines instances que les modèles SAT pouvait résoudre. En moyenne, le modèle de base est 45% plus rapide que le modèle avec bris de symétrie et presque cinq fois plus rapide que le modèle d'optimisation locale. Si notre but est de trouver un coloriage quelconque, nous devrions utiliser le modèle SAT de base. Cependant, si l'on cherche à prouver qu'il n'existe pas de coloriages valides, le modèle avec bris de symétrie serait meilleur. Cela semble confirmer les données recueillies par Codish et al.[5]. Il est difficile de trouver un modèle équilibré, puisque réduire le nombre de coloriages accélère la recherche complète de l'arbre, mais ralentit aussi la recherche d'une solution valide.

## Autres résultats

Dans la majorité des bancs d'essais, seulement environ 30% du temps total d'exécution est alloué à la résolution en soi plutôt que la conversion en FlatZinc. Même en augmentant le nombre de threads utilisés, la solution n'est pas trouvée beaucoup plus rapidement. De plus, la taille du fichier Flatzinc devient aussi une limitation importante. Dans plus grande instance solutionnée, soit $R(3,9;26)$, le temps de recherche pour trouver un coloriage avec le modèle SAT sans bris de symétrie demeure bien en-dessous de la limite imposée d'une heure, laissant croire qu'un coloriage $R(3,9;27)$ puisse être trouvé. Cependant, la quantité de mémoire RAM nécessaire dépassait 20 Go,

empêchant d'emblée la résolution du modèle. Cette limitation nous était assez inattendue, puisque le cours met davantage l'accent sur la complexité en termes de temps que celle en termes de mémoire. Nous avons également été étonnés des temps de compilation considérables observés dans nos modèles MiniZinc.

# Conclusion

Pour conclure, la nature exponentielle du problème rend le temps d'exécution nécessaire difficile à atteindre pour les bornes les plus hautes des instances de nombres de Ramsey connues. Néanmoins, le modèle SAT a dépassé nos attentes lorsqu'il s'agit de trouver un coloriage de Ramsey rapidement ainsi que de calculer des nombres de Ramsey exacts avec le bris de symétrie. MiniZinc étant peu optimisé pour la compilation de modèles, il pourrait être pertinent d'essayer un compilateur comme BEE ou Picat.

Comparativement au modèle SAT, notre modèle d'optimisation locale a moins bien performé qu'espéré. Nous souhaitions que ce modèle soit plus efficace dans de plus grandes instances, mais la taille importante du modèle, particulièrement le nombre de variables, fait en sorte que le temps de compilation FlatZinc soit beaucoup trop long. Si nous devions continuer ce projet, il pourrait être intéressant d'implémenter notre propre contrainte dans le solveur OR Tools (ou trouver un autre solveur l'ayant déjà) pour traiter spécifiquement le comptage local des cliques monochromatiques, sans devoir faire appel à de grands tableaux de variables pour le faire. Il nous est difficile de prévoir l'efficacité d'un tel modèle d'optimisation locale par rapport à l'algorithme randomisé courant, mais cela constituerait une piste de recherche intéressante.

# Annexe

## Contraintes de bris de symétrie supplémentaires

En plus de celles abordées dans le rapport, plusieurs autres contraintes ont été testées avec le modèle SAT. L'une d'entre elles devait compter le nombre d'arêtes de couleur *true* dans chaque rangée de la matrice, et mettre le résultat dans un tableau de variables. Or, même sans utiliser ces variables

pour contraindre le modèle, l'on a constaté que leur présence seule semblait parfois aider le solveur à faire de meilleurs branchements. Il est difficile de conclure si cela était dû à la chance ou non, nous incluons les temps d'exécution dans le tableau ci-dessous. Autre cas intéressant: $R(4,4) = 18$ peut être démontré en 33 minutes, soit environ deux fois moins de temps que le modèle SAT de base.

**Temps de résolution pour** $R(3, s; n)$

| s \ n | 14 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 5 | 1 s | | | | | | | | | | | | |
| 6 | // | 2 s | ? | | | | | | | | | | |
| 7 | // | 5 s | 7 s | 11 s | 17 s | 27 s | 39 s | ? | | | | | |
| 8 | // | 7 s | 12 s | 23 s | 37 s | 59 s | 1.5 m | 3 m | 4.5 m | 6 m | 8 m | 12.5 m | ? |
| 9 | // | 8 s | 17 s | 34 s | 1 m | 1.5 m | 3 m | 5 m | 7.5 m | 12.5 m | 21 m | ? | ? |

Nous avons aussi essayé une contrainte lexicographique simple, qui impose que des rangées consécutives de la matrice du coloriage soient lexicographiquement ordonnées, créant $O(n)$ contraintes plutôt que $O(n^2)$. Cependant, toutes ces contraintes n'amélioraient pas beaucoup les temps de résolution par rapport au modèle de base SAT, et ne pouvaient pas résoudre des instances plus grandes qu'avec la contrainte utilisée (contrainte 5).

## Paramètre de recherche locale

Pour activer le mode de recherche locale avec OR Tools, l'on ouvre l'éditeur de configuration du solveur, puis on ajoute le paramètre illustré ci-dessous dans la section "Extra configuration parameters" (voir figure 1). Le paramètre *use_lns_only* permet quant à lui d'activer le mode *Large Neighborhood Search*.

## Autres modèles

Nous avions aussi conçu d'autres modèles dont ce rapport n'a pas été question. Le modèle *Ramsey_optimisation.mzn* était notre première tentative d'améliorer le modèle SAT, et avant de concevoir le modèle d'optimisation locale. Celui-ci minimise le nombre de cliques monochromatiques du graphe, et permet donc de trouver le coloriage le plus près possible d'un coloriage de Ramsey valide lorsqu'aucun n'existe. Il est toutefois moins efficace que le

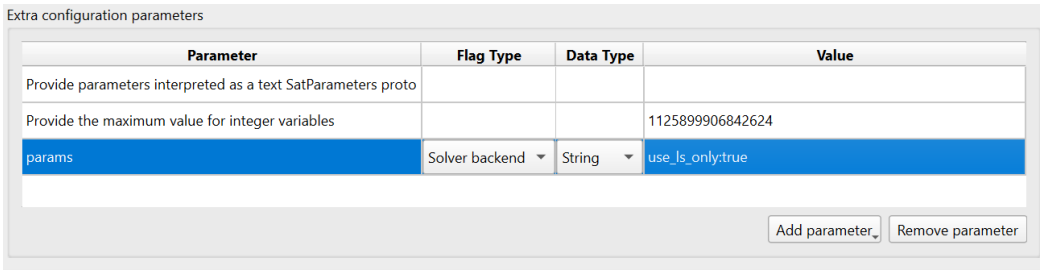


Figure 1: Paramètre de recherche locale

modèle d'optimisation locale. Il y a aussi le modèle *Ramsey_3_colors.mzn* qui suit la même logique que notre modèle SAT de base, mais traite les instances à trois couleurs au lieu de deux.

# Bibliography


[1] *7.7. Large neighborhood search (LNS): the Job-Shop Problem*. URL: https://acrogenesis.com/or-tools/documentation/user_manual/manual/metaheuristics/jobshop_lns.html.

[2] M. Codish et al. *Computing the Ramsey Number R(4,3,3) using Abstraction and Symmetry breaking*. 2016. URL: https://www.dcs.gla.ac.uk/~alice/papers/ramseyConstraints2016.pdf.

[3] Geoffrey Exoo. *A Lower Bound for R(5, 6)*. 2023. URL: https://arxiv.org/pdf/2310.17099.

[4] Brendan McKay. *Ramsey Graphs*. URL: https://users.cecs.anu.edu.au/~bdm/data/ramsey.html.

[5] Codish Michael et al. *Constraints for Symmetry breaking in Graph Representation*. 2018. URL: https://www.dcs.gla.ac.uk/~alice/papers/MillerCONSTRAINTS_POST_ACCEPTANCe.pdf.

[6] S. P. Radziszowski. *Small Ramsey Numbers*. 1994. URL: https://www.combinatorics.org/ojs/index.php/eljc/article/view/DS1/pdf.