Comparaison de méthodes de résolution pour le problème de somme coloration

Clément Lecat, Chu Min Li, Corinne Lucet, Yu Li

Université de Picardie Jules Verne, Laboratoire MIS
Pôle Sciences, 33 rue St Leu, 80000 Amiens

{clement.lecat, chu-min.li, corinne.lucet, yu.li}@u-picardie.fr

Résumé

Le problème de somme coloration (*MSCP*) est un problème dérivé du problème de coloration (*GCP*). Pour ces deux problèmes NP-difficiles, il s'agit d'affecter une couleur c_i à chaque sommet v d'un graphe $G = (V, E)$, en respectant les contraintes de voisinage : deux sommets voisins ne peuvent avoir une couleur identique. Pour *MSCP*, nous considérons qu'un poids p_i est associé à chaque couleur c_i. Alors que *GCP* a pour but de *minimiser le nombre des couleurs utilisées*, *MSCP* a pour but de *minimiser la somme des poids des couleurs utilisées*.

À travers ce papier, nous présentons et comparons quatre méthodes exactes permettant de résoudre le problème *MSCP*. La première de ces méthodes est une méthode de séparation et évaluation, basée sur le calcul de bornes supérieures et inférieures. Les deux méthodes suivantes consistent à transformer le graphe en une instance *MaxSAT* et en une instance *MinSAT*, et de les résoudre avec un solveur approprié. La dernière méthode consiste à transformer le problème *MSCP* en un problème COP et à utiliser un solveur dédié à la résolution.

Abstract

The Minimum Sum Coloring Problem (*MSCP*) is an extension to the well known Graph Coloring Problem (*GCP*). Both NP-hard problems consist in assigning a color c_i to each vertex v of a graph $G = (V, E)$ while respecting the neighborhood constraints : two adjacent vertices cannot share a same color. For *MSCP*, a weight p_i is associated with each color c_i. While the objective of *GCP* is *to minimize the number of colors*, the objective of *MSCP* is *to minimize the sum of the color weights*.

In this paper, we propose and compare four methods to solve *MSCP*. The first one is a Branch-and-Bound method which uses upper and lower bounds to reduce search space. The second, the third and the fourth methods encode $MSCP$ instances into a $MaxSAT$, $MinSAT$ and COP instance respectively, and solve these instances with dedicated solvers.

1 Introduction

Le problème de coloration de graphe (Graph Coloring Problem *GCP*) est un problème majeur en optimisation combinatoire. De nombreuses études ont été menées autour de ce problème NP-difficile [7]. Les méthodes proposées pour résoudre ce problème sont multiples et se divisent en deux catégories, les méthodes exactes et les méthodes approchées. Les méthodes exactes visent à fournir une solution optimale au problème, elles regroupent les méthodes de Branch-and-Bound (*e.g.* [22]), de décomposition de graphe (*e.g.* [20]), mais aussi les formulations sous forme de problème SAT [26]. Les méthodes approchées fournissent une borne supérieure ou inférieure au problème et rassemblent les algorithmes gloutons comme le célèbre DSATUR [5] et des heuristiques et méta-heuristiques [24, 8, 19]. Dans la littérature ces méthodes sont généralement testées sur des benchmarks de référence comme DIMACS et COLOR [6, 10].

Le problème de somme coloration minimale d'un graphe (Minimal Sum Coloring Problem *MSCP*) est un problème dérivé de *GCP* qui fut introduit en 1989 par Kubicka et Schwenk [14]. Les principaux résultats publiés sur *MSCP* montrent des propriétés structurelles relatives aux familles de graphes pour lesquelles il existe des algorithmes efficaces et des bornes théoriques [11, 1, 21, 3]. Des études plus récentes proposent des heuristiques [18] et méta-heuristiques [25, 12, 4, 23] qui fournissent des bornes

pour les instances ci-dessus citées.

Les travaux que nous reportons dans cet article, portent principalement sur une approche exacte pour le problème *MSCP*, dédiées donc à produire une solution optimale pour ce problème. Nous proposons une méthode de Branch-and-Bound basée sur l'élaboration de bornes supérieures et inférieures pertinentes, et un encodage du problème *MSCP* en une instance $MaxSAT$, $MinSAT$ et COP. Nous analysons et comparons ensuite ces quatre méthodes exactes pour MSCP. Les résultats ont été obtenus sur des instances structurées des benchmarks de référence et sur des graphes aléatoires.

Ce papier est organisé de la façon suivante : Dans la section 2 nous donnons une définition formelle des problèmes *GCP* et *MSCP*. Dans la section 3, nous présentons le calcul de bornes inférieures (LB). Dans la section 4, nous présentons la transformation d'un problème *MSCP* en un problème *MaxSAT* puis en un problème *MinSAT* et pour terminer cette section, en un problème COP. Enfin dans la section 5, nous comparons le temps de réponse de ces différentes méthodes sur un ensemble d'instances.

2 Problème de somme coloration

2.1 Définitions préliminaires

Soit un graphe non orienté *G=(V,E)* où V est l'ensemble des sommets ($|V| = n$) et $E \subseteq V \times V$ l'ensemble des arêtes ($|E| = m$). Le voisinage $\mathcal{N}(v)$ d'un sommet $v \in V$ est défini comme suit : $\mathcal{N}(v) = \{u \in V \mid (v,u) \in E\}$. Le degré d'un sommet v est le nombre de sommets voisins de v, nous le notons $d(v)$. Le degré du graphe G est $\Delta(G) = max\{d(v); \forall v \in V\}$. Un *sous-graphe induit* $G' = (V', E')$ de *G=(V,E)* est un graphe tel que $V' \subseteq V$ et $\forall (u,v) \in E$ si $u \in V'$ et $v \in V'$ alors $(u,v) \in E'$. Une *clique* $C = (V_c, E_c)$ d'un graphe G est un sous-graphe induit complet de G, tel que si $u \in V_c$ et si $v \in V_c$ alors $(u,v) \in E$.

2.2 Coloration et somme coloration

Une coloration du graphe est une fonction $c : V \mapsto \{1, 2, \ldots, k\}$ qui associe à chaque sommet $v \in V$ une couleur $c(v)$ représentée ici par un entier naturel. Une coloration est dite valide, si $\forall (u,v) \in E,\ c(u) \neq c(v)$. Nous notons $X = X_1, X_2, \ldots, X_k$ une coloration de G, avec $X_i = \{v \in V \mid c(v) = i\}$, nommée classe couleur i.

GCP a pour objectif de trouver une coloration valide du graphe, en utilisant un nombre minimal de couleurs. Ce nombre minimal est appelé *nombre chromatique* du graphe et ce nombre est appelé $\chi(G)$. Nous notons qu'une clique de n_c sommets requiert n_c couleurs.

Pour *MSCP*, à chaque couleur i est associé un poids p_i. A une coloration donnée X du graphe, correspond donc un poids $P(X)$ égal à la somme des poids des couleurs utilisées par X.

$$P(X) = p_1 \times |X_1| + p_2 \times |X_2| + \ldots + p_k \times |X_k|$$

MSCP consiste à trouver une coloration valide dont la somme des poids associés est minimale. Nous notons $\Sigma(G)$ cette somme, appelée *somme chromatique* du graphe.

$$\Sigma(G) = min\{P(X) \mid X \textit{ est une coloration valide de } G\}$$

Kubicka et Schwenk [14] ont démontré que *MSCP* est un problème *NP-difficile*. Le plus petit nombre de couleurs utilisées pour colorier le graphe G dans une solution optimale pour *MSCP* est appelé *force* du graphe et noté $s(G)$.

Comme les classes de couleur X_i forment des ensembles de sommets deux à deux indépendants, il est possible d'échanger deux couleurs c_i et c_j, sans attenter à la validité de la coloration. Nous noterons $X' = Exch_{(i,j)}(X)$ la coloration obtenue par un tel échange. Nous parlons alors de colorations symétriques (X' est symétrique à X). Néanmoins, les colorations correspondantes n'auront pas forcément un même poids associé. Nous définirons donc une relation de dominance entre les colorations afin d'éviter l'énumération de colorations symétriques. Sans perte de généralité nous pouvons considérer que : $\forall i, j \in \{1, 2, \ldots, k\}$ si $i < j$ alors $p_i < p_j$.

Définition 1 *Une coloration X' est dite dominée par une coloration X si $\exists\ c_i, c_j$, telles que $X' = Exch_{(i,j)}(X)$ et $P(X) \leq P(X')$.*

Propriété 1 *La coloration $X^\theta = X_1, X_2, \ldots, X_k$ pour laquelle est vérifiée $|X_1| \geq |X_2| \geq \ldots \geq |X_k|$ domine toutes les colorations X' symétriques à X^θ. X^θ est la coloration dominante de sa classe.*

Dans la suite de cette étude, comme dans la majorité des travaux de la littérature nous considèrerons que : $\forall i,\ p_i = i$. A la couleur c_i correspondra donc le poids i. Il est immédiat de trouver la coloration dominante X^θ à partir d'une coloration quelconque du graphe : il suffit d'ordonner les classes de couleurs suivant l'ordre décroissant de leur cardinalité. Donc, quelque soit la méthode employée pour trouver une coloration valide, nous nous rapportons toujours à la coloration X^θ lors de l'évaluation d'une solution.

2.3 Propriétés

Nous pouvons trouver dans la littérature quelques résultats théoriques et structurels relatifs à $MSCP$ [11, 3, 1, 21]. Celui qui nous intéresse plus particulièrement, fournit une borne supérieure et une borne inférieure théoriques, nous permettant de diminuer l'espace de recherche dans la résolution de $MSCP$. Alavi et *al* [1] ont ainsi démontré que pour un graphe composé de m arêtes, la somme chromatique est encadrée comme suit : $\lceil\sqrt{8m}\rceil \leq \Sigma(G) \leq \lfloor\frac{3(m+1)}{2}\rfloor$. De plus Malafiejski [21] montre que la force du graphe est bornée par le degré du graphe plus un : $s(G) \leq \Delta(G)+1$. Pour la suite, nous utilisons ces bornes pour l'initialisation de la borne inférieure et pour l'initialisation du nombre de couleurs à considérer dans le Branch-and-Bound et dans la formulation SAT et COP de $MSCP$. Nous notons que même si le nombre chromatique d'un graphe est connu, nous ne pouvons nous limiter à ce nombre dans l'exploration de l'espace des solutions. La figure 1 illustre bien cette particularité. En effet, pour cet arbre de 8 sommets et 7 arêtes, dont le nombre chromatique est trivialement 2, il est nécessaire d'utiliser 3 couleurs pour obtenir la somme chromatique. De manière générale, $s(G)$ peut être très éloigné de $\chi(G)$, et l'espace des solutions de $MSCP$ peut être plus compliqué à explorer que celui de GCP [13].

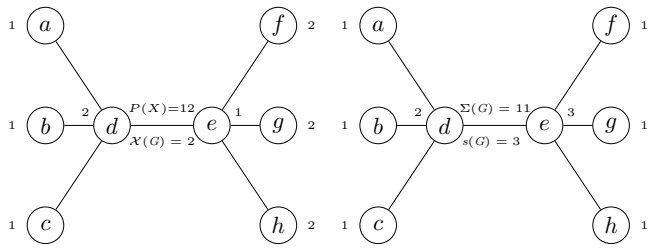

FIGURE 1 – Comparaison des solutions optimales pour GCP et $MSCP$

3 Branch-and-Bound pour MSCP

Nous présentons dans ce paragraphe une méthode de Branch-and-Bound (Séparation et Evaluation), pour la résolution exacte de $MSCP$. L'exploration complète de l'ensemble des solutions pourrait se représenter par un arbre de coloration comme le montre la figure 3 pour le cas d'une coloration du graphe de la figure 2. Chaque noeud de l'arbre correspond à un sommet du graphe et chaque arête pendante à une assignation de couleur pour le sommet correspondant. Le principe de la méthode développée, consiste à *séparer* l'ensemble des solutions en plusieurs sous-ensembles de solutions représentés par les différents noeuds, et à *évaluer* en chaque noeud la pertinence de continuer à explorer ce sous-ensemble de solutions. Les critères de séparation et évaluation employés sont décrits ci-dessous.

Séparation. Comme dans la majorité des méthodes énumératives, notre séparation de l'espace des solutions se fait par l'assignation d'une couleur à un sommet. Les couleurs sont comprises entre 1 et $\Delta(G) + 1$ comme mentionné dans le paragraphe précédent. Toute branche de longueur n, le nombre de sommets, correspond à une coloration X valide dont le poids associé est celui de la coloration dominante X^θ (cf. propriété 1). En chacun des noeuds, nous devons disposer de l'ensemble des couleurs *disponibles* pour le sommet correspondant, afin de séparer en autant de sous-ensembles. Pour plus de clarté dans la suite de l'exposé, nous convenons que la séparation relative au sommet v_i est effectuée au niveau i de notre arborescence (cf. figure 3).

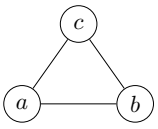

FIGURE 2 – Graphe

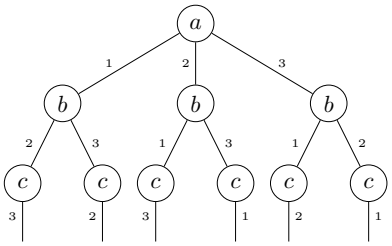

FIGURE 3 – Arbre de coloration pour $k = 3$ du graphe de la figure 2

Evaluation. Il nous est nécessaire pour cet évaluation de maintenir à jour une borne supérieure globale UB ainsi qu'une borne inférieure LB, relative au sous-ensemble de solutions représenté par le noeud en cours d'évaluation. Initialement ces bornes sont respectivement UB_{DSATUR} et $\lceil\sqrt{8m}\rceil$. $UB_{MDSATUR}$ est une borne supérieure calculée à partir de l'algorithme glouton $MDSATUR$ [18]. S'il s'avère lors de l'évaluation d'un noeud, que $LB \geq UB$, alors la branche est coupée, l'exploration de ce sous-ensemble de solutions est abandonnée, car toutes les solutions X contenues dans ce sous-ensemble verifient $P(X) \geq UB$.

Borne supérieure : Si une branche complète de l'arbre a pu être développée, elle correspond à une

coloration complète et valide X du graphe, et si $UB \geq P(X^\theta)$, alors la borne supérieure UB est mise à jour ($UB \leftarrow P(X^\theta)$).

Borne inférieure : La performance de cet algorithme repose également sur la qualité de la borne inférieure LB calculée en chaque noeud. Si nous considérons que nous nous trouvons au niveau i de l'arborescence, les sommets $v_1, v_2, \ldots, v_{i-1}$ ont une couleur qui leur a été préalablement assignée. Dans ce cas une borne inférieure triviale, LB^1, peut être déterminée en considérant par défaut que les $n-i+1$ sommets non coloriés, le sont avec leur plus petite couleur disponible, notée $ppcdispo(v_i)$. La valeur de cette borne est alors calculée comme suit :

$$LB^1 = \sum_{t=1}^{i-1} c(v_t) + \sum_{t=i}^{n} ppcdispo(v_t)$$

L'intérêt d'une telle borne est sa faible compléxité ($O(n)$), mais beaucoup trop de contraintes (les arêtes entre les sommets non coloriés) sont relâchées dans le calcul de LB^1 qui n'est donc pas très efficace dans la phase d'évaluation.

Nous proposons une borne inférieure préservant d'avantage les contraintes de voisinage du problème. Pour calculer cette borne, nous décomposons la partie du graphe non colorié noté $\widetilde{G} = (\widetilde{V}, \widetilde{E})$, en une partition de cliques $C_1, C_2, \ldots, C_l$ telle que $\forall i, j\ C_i \cap C_j = \emptyset$ et $\forall u \in \widetilde{G},\ \exists C_i\ tq.\ u \in C_i$. Pour obtenir une telle décomposition, nous utilisons un algorithme glouton $Clique_{Part}$ détaillé dans l'algorithme 3. Cet algorithme traite les sommets suivant un ordre fixe. Après avoir testé expérimentalement différents critères, nous avons choisi d'ordonnancer ces sommets suivant l'ordre croissant de leur degré. Chacune des cliques est alors coloriée en fonction des couleurs disponibles des sommets qui la composent. Nous notons $P_{lb}(C_i)$ la somme associée à la clique C_i. Considérons l'ensemble de toutes les couleurs disponibles d'une clique C_i : $Disp(C_i) = \bigcup_{v \in C_i} \{Dispo(v)\}$ et posons $|C_i| = \alpha_i$. Alors la somme des α_i plus petites couleurs disponibles ($P_{lb}(C_t)$) est une borne inférieure de la somme chromatique de la clique C_i. Par extension, la borne inférieure LB^2 est alors calculée comme suit :

$$LB^2 = \sum_{t=1}^{i-1} c(v_t) + \sum_{t=1}^{l} P_{lb}(C_t)$$

L'algorithme récursif $BBMSCP$ (cf. algorithme 1) synthétise l'ensemble de la méthode de séparation et évaluation présentée dans cet article. Dans cet algorithme, le paramètre G désigne le graphe non encore colorié, S la somme des couleurs assignées et UB la

Algorithme 1 : BBMSCP($G, \mathcal{S}, UB$)

Input : $G = (V, E)$
$\mathcal{S}$ somme des couleurs assignées
UB borne supérieure
Output : $\Sigma(G)$ la somme chromatique

```
1  begin
2      if | V |= ∅ then
3          return S;
4      Choisir v un sommet de G ;
5      Disp(v) := liste des couleurs disponibles de v
6      foreach c ∈ Disp(v) telle que S + c ⩽ UB
       do
7          foreach u ∈ N(v) do
8              Disp(u) := Disp(u) \ {c}
9          if LB²(G \ v) + S + c < UB then
10             UB :=BBMSCP(G \ v, S + c, UB);
11         foreach u ∈ N(v) do
12             Disp(u) := Disp(u) ∪ {c}
13     return UB;
```

Algorithme 2 : $LB^2(G)$

Input : $G = (V, E)$
Output : lb une borne inférieure de $\Sigma(G)$

```
1  begin
2      lb ← 0;
3      Décomposer G en une partition de cliques
       C1, C2, ..., Cl;
4      foreach Ci do
5          Disp(Ci) := ⋃_{v∈Ci} Dispo(v)
6          foreach x := 1 to |Ci| do
7              cmin := MIN{c | c ∈ Disp(Ci)}
8              lb := lb + cmin
9              Disp(Ci) := Disp(Ci) \ {cmin}
10     return lb;
```

Algorithme 3 : $Clique_{Part}(G)$

Input : $G = (V, E)$
Output : P une partion en cliques de G

```
1  begin
2      P = ∅;
3      while G n'est pas vide do
4          v ← le plus petit sommet de G;
5          G ← G\{v};
6          if Il existe une clique C dans P tel que v
           est connecté à tous les sommets de C then
7              insérer v dans C;
8          else
9              créer une nouvelle clique C;
10             insérer v dans C;
11             P := P ∪ {C};
12     return P;
```

meilleure borne supérieure rencontrée. Le premier appel sera donc $BBMSCP(G, 0, UB_{MDSATUR})$. L'algorithme LB^2 (cf. algorithme 2) donne les détails du calcul de la borne inférieure proposée dans ce paragraphe, basée sur le partitionnement en cliques de la partie du graphe non encore colorié.

4 Résolution par codage de MSCP en SAT et COP

4.1 Préliminaire

4.1.1 SAT

Le problème *SAT* est un problème issu de la logique propositionnelle qui consiste à tester si une formule $\mathcal{F}$ sous forme normale conjonctive (*CNF*) possède une interprétation $\mathcal{I}$ satisfaisant toutes les clauses $\mathcal{F}$. Cependant, le problème *SAT* n'est pas suffisant pour modéliser un problème d'optimisation tel que le problème $MSCP$. En effet, il est nécessaire de considérer le problème *weighted partial MaxSAT* ou le problème dual *weighted partial MinSAT*. Le problème *weighted partial MaxSAT* est une extension du problème *MaxSAT*, respectivement, le problème *weighted partial MinSAT* est une extension du problème *MinSAT*. Ces deux problèmes sont des formules *CNF* constituées de clauses *dures* et de clauses souples (voir exemple 1). Dans le cas d'un problème *weighted partial MaxSAT*, le but est de trouver une affectation qui satisfasse toutes les clauses dures en maximisant la somme des poids des clauses souples satisfaites. Pour le cas du problème *weighted partial MinSAT*, l'objectif est de déterminer une affectation qui satisfasse toutes les clauses dures tout en minimisant la somme des poids des clauses souples satisfaites.

Exemple 1 *Soit une formule $\mathcal{F}$ bien formée pour le problème weighted partial MaxSAT composée des clauses $\{(C_1, \infty), (C_2, \infty), (C_3, 3), (C_4, 5)\}$, $C_1 = x \vee y$, $C_2 = \neg x \vee y$, $C_3 = x$ et $C_4 = \neg x$ où C_1 et C_2 sont des clauses dures, et C_3 et C_4 sont clauses souples. L'interprétation $\{x = faux, y = vrai\}$ est une solution optimale du problème weighted partial MaxSAT contrairement à l'interprétation $\{x = vrai, y = vrai\}$ qui est une solution optimale pour le problème weighted partial MinSAT.*

4.1.2 COP

Le problème de satisfaction de contraintes (*CSP*) a pour objectif de déterminer s'il existe une instanciation des variables telle que toutes les contraintes du problème soient satisfaites. Le formalisme simple de *CSP* permet de faciliter la modélisation de nombreux problèmes. Un *CSP* est défini par un triplet (X,D,C), où X est un ensemble fini de variables $\{x_1, x_2, ..., x_n\}$, D est l'ensemble des domaines associés à chaque variable du problème $\{dom(x_1), dom(x_2), ..., dom(x_n)\}$ et C est un ensemble fini de contraintes $\{c_1, c_2, ..., c_m\}$. Une contrainte $c_i \in C$ est une relation portant sur un ensemble de variables appartenant à X. Le problème d'optimisation sous contraintes (COP) est une extension du problème de satisfaction de contraintes (CSP) auquel est associé une fonction objectif $F : D_{x_1} \times D_{x_2} \times ... \times D_{x_n} \rightarrow \mathbb{N}$. Une solution optimale du CSP est un solution qui minimise ou maximise la fonction objectif tout en respectant l'ensemble de contraintes C. Un COP est donc défini par un quadruplet (X,D,C,F).

4.2 Codages de MSCP

4.2.1 vers SAT

À travers cette section, nous présentons le passage d'un problème *MSCP* vers un problème *weighted partial MaxSAT* et *weighted partial MinSAT*. Les variables booléennes x_{ai} du codage *SAT* sont définies comme suit :

$$x_{ai} = \begin{cases} 1 \textit{ si nous colorions le sommet } a \\ \textit{avec la couleur } i \\ \\ 0 \textit{ sinon} \end{cases}$$

Les clauses dures du problème *weighted partial MaxSAT* sont identiques à celles du problème *weighted partial MinSAT*. L'ensemble des clauses dures correspond

à l'ensemble des contraintes du problème *GCP*. Ces dernières sont les suivantes :

— Attribuer à un sommet quelconque une couleur $1, 2..., k$ (1) :

$$x_{a1} \vee x_{a2} \vee ... \vee x_{ak}$$

— Un sommet quelconque ne peut posséder qu'une seule couleur (2) :

$$\neg x_{ai} \vee \neg x_{aj} \textit{ pour tout } i \neq j$$

— Deux sommets voisins a et b ne peuvent pas être coloriés de la même façon (3) :

$$\neg x_{ai} \vee \neg x_{bi} \textit{ pour tout } i$$

Le nombre de clauses dures ainsi générées est :

$$\underbrace{n}_{(1)} + \underbrace{(n \times \binom{k}{2})}_{(2)} + \underbrace{(m \times k)}_{(3)}$$

L'ensemble des clauses souples est un ensemble de clauses unitaires correspondant aux sommets et aux couleurs. Chaque clause souple est pondérée selon un poids. Dans le cas du problème *weighted partial MaxSAT*, nous souhaitons maximiser la somme des poids des clauses souples satisfaites, contrairement au problème *weighted partial MinSAT* qui a pour objectif de minimiser la somme des poids des clauses souples satisfaites. Dans les deux cas, le nombre de clauses souples générées est de :

$$(n \times k)$$

La réécriture d'une instance $MSCP$ génère une CNF de complexité spatiale en $\mathcal{O}(n \times k^2 + m \times k)$.

Exemple 2 *Si nous prenons le graphe de la figure 1, pour $k = 3$, l'instance générée est la suivante :*

clauses dures :
$x_{a1} \vee x_{a2} \vee x_{a3}$
$x_{b1} \vee x_{b2} \vee x_{b3}$
$x_{c1} \vee x_{c2} \vee x_{c3}$
$\neg x_{a1} \vee \neg x_{a2}$
$\neg x_{a1} \vee \neg x_{a3}$
$\neg x_{a2} \vee \neg x_{a3}$
$\neg x_{b1} \vee \neg x_{b2}$
$\neg x_{b1} \vee \neg x_{b3}$
$\neg x_{b2} \vee \neg x_{b3}$
$\neg x_{c1} \vee \neg x_{c2}$
$\neg x_{c1} \vee \neg x_{c3}$
$\neg x_{c2} \vee \neg x_{c3}$
$\neg x_{a1} \vee \neg x_{b1}$
$\neg x_{a2} \vee \neg x_{b2}$
$\neg x_{a3} \vee \neg x_{b3}$
$\neg x_{a1} \vee \neg x_{c1}$
$\neg x_{a2} \vee \neg x_{c2}$
$\neg x_{a3} \vee \neg x_{c3}$
$\neg x_{b1} \vee \neg x_{c1}$
$\neg x_{b2} \vee \neg x_{c2}$
$\neg x_{b3} \vee \neg x_{c3}$

weighted partial MaxSAT (encodage 1)

clauses souples :
x_{a1} 3
x_{a2} 2
x_{a3} 1
x_{b1} 3
x_{b2} 2
x_{b3} 1
x_{c1} 3
x_{c2} 2
x_{c3} 1

weighted partial MinSAT (encodage 1)

clause souples :
$\neg x_{a1}$ 3
$\neg x_{a2}$ 2
$\neg x_{a3}$ 1
$\neg x_{b1}$ 3
$\neg x_{b2}$ 2
$\neg x_{b3}$ 1
$\neg x_{c1}$ 3
$\neg x_{c2}$ 2
$\neg x_{c3}$ 1

Dans l'instance MaxSAT et l'instance MinSAT, un poids plus grand est associé à une clause souple correspondant à une couleur plus petite, pour que le solveur MaxSAT (MinSAT) la satisfasse (falsifie) en priorité. En effet, la satisfaction (falsification) d'une clause souple dans MaxSAT (MinSAT) signifie l'affectation de la couleur correspondante au sommet correspondant. Par exemple, la satisfaction de la clause

x_{11} dans MaxSAT signifie l'affectation de la couleur 1 au sommet 1, de même que la la falsification de la clause $\neg x_{11}$ dans MinSAT. À noter que la seule différence entre le codage MinSAT et le codage MaxSAT est le signe des clauses souples. Comme nous allons voir, cette petite différence a un très grand impact dans la résolution de MSCP. En effet, la résolution MinSAT repose sur le calcul d'une borne supérieure détectant les clauses souples ne pouvant être falsifiées en même temps. Avec cet encodage, nous observons par exemple que les clauses souples $\neg x_{a1}$ et $\neg x_{a2}$ ne peuvent être falsifiées au même moment car la clause dure $\neg x_{a1} \vee \neg x_{a2}$ serait violée. À travers cet exemple nous pouvons donc observer que le codage MinSAT permet une meilleure prise en considération des spécificités de l'instance $MSCP$. Un autre encodage valide $MinSAT$ et $MaxSAT$ est d'utiliser le même ensemble de clauses dures et de coder l'ensemble des clauses souples de la façon suivante :

Exemple 3

weighted partial MaxSAT (encodage 2)

$$\begin{array}{rl} \textit{clause souples}: & \neg x_{a1}\ 1 \\ & \neg x_{a2}\ 2 \\ & \neg x_{a3}\ 3 \\ & \neg x_{b1}\ 1 \\ & \neg x_{b2}\ 2 \\ & \neg x_{b3}\ 3 \\ & \neg x_{c1}\ 1 \\ & \neg x_{c2}\ 2 \\ & \neg x_{c3}\ 3 \end{array}$$

weighted partial MinSAT (encodage 2)

$$\begin{array}{rl} \textit{clause souples}: & x_{a1}\ 1 \\ & x_{a2}\ 2 \\ & x_{a3}\ 3 \\ & x_{b1}\ 1 \\ & x_{b2}\ 2 \\ & x_{b3}\ 3 \\ & x_{c1}\ 1 \\ & x_{c2}\ 2 \\ & x_{c3}\ 3 \end{array}$$

Il apparaît clairement que ce codage n'est pas du tout adapté au problème MinSAT. En effet, contrairement au codage précédent il est seulement possible d'en déduire que seules les clauses souples x_{a1}, x_{a2} et x_{a3} ne peuvent être falsifiées simultanément. Le calcul de la borne supérieure s'en trouve dégradé. Il est à noter que pour le problème MaxSAT, le choix de l'encodage 2 est plus efficace. Une raison est le fait de la proximité de la lower bound et l'upper bound durant la résolution de MaxSAT [15]. Dans la suite de ce papier, l'encodage 1 est utilisé pour la transformation de $MSCP$ en un problème $MinSAT$ et l'encodage 2 est utilisé pour la transformation de $MSCP$ en un problème $MaxSAT$.

4.2.2 vers COP

Pour transformer un problème de somme coloration en COP il est dans un premier temps nécessaire de déterminer l'ensemble des variables de X, puis l'ensemble des domaines de chaque variable, et enfin déterminer les différentes contraintes liées au problème $MSCP$. Nous définissons donc le COP de la façon suivante :

- X=$\{v_1, v_2, ..., v_n\}$ chaque variable v_i correspond à un sommet du graphe ;
- $dom(v_1)$=$dom(v_2)$=...=$dom(v_n)$=$\{1,2,...,k\}$ correspond au différentes couleurs possibles ;
- Si v_i et v_j sont deux sommets du graphe tel que $(v_i, v_j) \in E$ alors $c_{ij} \in C$ et nous définissons c_{ij} : $v_i \neq v_j$;
- La fonction objectif à minimiser : $F = \sum_{i=1}^{n} v_i$.

La réécriture d'une instance $MSCP$ génère un CSP de complexité spatiale en $\mathcal{O}(n+m)$.

Exemple 4 *Si nous prenons le graphe de la figure 1, pour un k valant 3, le COP (X,D,C) générée est la suivante :*

- *X=*$\{a,b,c\}$ *;*
- *dom(a)=dom(b)=dom(c)=*$\{1,2,3\}$ *;*
- *C=*$\{c_{ab}, c_{ac}, c_{bc}\}$ *;*
- *minimiser* $F = a + b + c$.

5 Resultat

Nous avons mené nos expérimentations sur un processeur Intel Westmere Xeon E7-8837 de 2.66GHz. Comme il s'agit du tout début de nos travaux, les seules instances que nous avons utilisées sont des graphes aléatoires et quelques instances issues de COLOR [10] et DIMACS [6].

Les graphes aléatoires sont des instances de taille et de densité variables. Un graphe aléatoire est construit de la façon suivante : la densité d d'un graphe aléatoire est comprise entre 0.1 et 0.9. Pour chaque couple u, v de sommets, l'arête (u, v) est ajoutée au graphe avec la probabilité d. Les graphes structurés ont été utilisés comme Benchmark lors du "computational symposium" [10] et lors du deuxième challenge DIMACS [6]. Ils correspondent à des applications modélisés par des graphes.

5.1 Résultats expérimentaux

Nous comparons les méthodes suivantes :

- BBMSCP : l'algorithme de Branch-and-Bound que nous proposons dans ce papier ;
- MaxSatz [16] : nous avons utilisé la version de ce solveur de *weighted partial MaxSAT* qui a participé à l'évaluation de MaxSAT de 2009 ;
- ISAC [2] : nous avons utilisé la version de ce solveur de *weighted partial MaxSAT* qui a participé à l'évaluation de MaxSAT de 2013 ;
- MinSatz [17] : nous avons utilisé la version *weighted partial MinSAT* de 2013 ;
- Choco3 [9] : nous avons utilisé la version 3.3.0.

Le tableau 1 synthétise l'ensemble des résultats expérimentaux obtenus pour un ensemble d'instances aléatoires, tandis que le tableau 2 présente les résultats pour les instances structurées. Sont représentés dans les différentes colonnes, N le nombre de sommets, M le nombre d'arêtes, $\Sigma(G)$ la somme chromatique et pour *BBMSCP*, *MaxSatz*, *ISAC*, *MinSatz* et *Choco3* les temps d'exécution de chaque solveur en secondes. Si après 1 heure d'exécution la methode n'est pas terminée, le processus est stoppé (N/A).

Ces résultats préliminaires montrent l'efficacité des solveurs *weighted partial MinSAT* et *weighted partial MaxSAT* dans la résolution du problème $MSCP$, notamment le solveur ISAC. Une première explication est que pour couper les branches, ces solveurs utilise une borne calculée en exploitant des raisonnements propositionnels comme les règles de référence et la propagation unitaire en prenant en compte des relations entre les clauses souples. Pour permettre une exploitation optimale de la borne supérieure de $MinSAT$ et la borne inférieure de $MaxSAT$, il est nécessaire d'adapter le codage de l'instance $MinSAT$ et $MaxSAT$. Cette remarque est confirmée par le tableau 3. Celui-ci contient les résultats expérimentaux obtenus de ces deux encodages (cf exemple 2 et 3) pour un ensemble d'instances aléatoires en utilisant les solveurs $MinSatz$, $MaxSatz$ et $ISAC$.

Nous pouvons observer que les résultats confirment l'importance majeure du choix de l'encodage pour la résolution de $MaxSAT$ et $MinSAT$. Ces résultats nous montrent que l'efficacité de la résolution d'un problème ne dépend pas uniquement du solveur, mais également du choix de l'encodage du problème. Nous notons que le solveur MinSatz n'est pas adapté pour la résolution de problèmes de grande taille, car le nombre maximum de clauses souple est limité à 10000. Ceci est dû à l'utilisation d'une matrice adjacente implémentée dans le solveur MinSatz. Nous allons remédier à ce problème dans la prochaine amélioration de MinSatz.

Malgré le peu de différence entre le codage d'une instance $MaxSAT$ et d'une instance $MinSAT$, nous observons que le problème $MinSAT$ peut être pertinent dans la résolution de certains problèmes. Nous notons que la complexité spatiale du codage de $MSCP$ en CNF est quadratic en nombre de couleurs, et que cette complexité est linéaire pour le codage de $MCSP$ en COP. Cependant, les tableaux 1 et 2 montrent des résultats peu encourageant pour la résolution de $MSPC$ par COP.

Nous pouvons également voir que le solveur BBMSCP, première ébauche de nos travaux, reste compétitif. En effet, le nombre d'instances résolues par le solveur BBMSCP est supérieur à celui des instances résolues par Choco3, et il possède un temps de résolution comparable aux solveurs MaxSatz, ISAC et MinSatz.

6 Conclusion

À travers cet article, nous avons présenté *MSCP* et proposé différentes méthodes exactes de résolution. La première méthode se base sur un algorithme de branch-and-bound ($BBMSCP$), pour lequel les premiers résultats montrés sont prometteurs. $BBMSCP$ est à notre connaissance le premier solveur exact dédié à la résolution de $MSCP$. Les résultats obtenus nous encouragent à développer de nouvelles bornes et de nouveaux types de branchement pour BBMSCP, afin d'en augmenter l'efficacité sur les instances de plus grande taille. Nous avons également proposé un codage sous forme de problème *weighted partial MinSAT* et *weighted partial MaxSAT*, et remarqué qu'ils sont particulièrement adapté pour la résolution de *MSCP*. Une piste très prometteuse serait de combiner l'exploitation des structures des graphes et des bornes de $BBMSCP$ avec la puissance de raisonnement logique de ces solveurs dans la résolution de $MSCP$.

Remerciements : Nous remercions sincèrement les reviewers pour leurs remarques constructives qui nous ont permis d'améliorer la présentation de nos travaux.

				BBMCP	MaxSatz	ISAC	MinSatz	Choco3
Instances	N	M	$\Sigma(G)$	Temps	Temps	Temps	Temps	Temps
S10A23	10	23	20	0	0	0	0	30
S10A40	10	40	34	0	0	0	0	48
S10A5	10	5	14	0	0	0	0	3
S20A100	20	100	55	10	126	0	172	N/A
S20A19	20	19	27	0	0	0	0	2
S20A50	20	50	39	0	0	0	0	74
S20A75	20	75	52	8	144	0	18	N/A
S20A95	20	95	50	3	29	0	32	307
S25A100	25	100	63	96	1051	0	150	N/A
S25A30	25	30	40	0	0	0	0	25
S30A100	30	100	65	105	242	0	22	N/A
S30A44	30	44	48	0	0	0	0	1551
S35A60	35	60	60	54	2	0	0	N/A

TABLE 1 – Comparaison du temps d'exécution en secondes nécessaire à la détermination de la somme chromatique d'un pool d'instances aléatoires.

				BBMCP	MaxSatz	ISAC	MinSatz	Choco3
Instances	N	M	$\Sigma(G)$	Temps	Temps	Temps	Temps	Temps
1FI3	30	100	54	2	9	0	2	707
miles250.col	128	387	334	N/A	2	0	0	N/A
miles500.col	128	1170	715	N/A	20	0	0	N/A
myciel3.col	11	20	21	0	0	0	0	3
myciel4.col	23	71	45	0	9	0	0	1944
queen5_5.col	25	160	75	3036	N/A	N/A	N/A	N/A

TABLE 2 – Comparaison du temps d'exécution en secondes nécessaire à la détermination de la somme chromatique d'un pool d'instances structurées.

Références

[1] Y. Alav, P.J. Malde, and A.J. Schwenk. Tight bounds on the chromatic sum of a connected graph. *Journal of Graph Theory*, 13 :353–357, 1989.

[2] C. Ansótegui, Y. Malitsky, and M. Sellmann. Maxsat by improved instance-specific algorithm configuration. *Association for the Advancement of Artificial Intelligence*.

[3] A. Bar-Noy and G. Kortsarz. Minimum color sum of bipartite graphs. *J. Algolrithms*, 28(2) :339–365, 1998.

[4] U. Benlic and J.K. Hao. A study of breakout local search for the minimum sum coloring problem. *SEAL 2012, Lecture Notes in Computer Science*, 7673 :128–137, 2012.

[5] D. Brélaz. New methods to color vertices of a graph. *Commun. ACM*, 22(4) :251–256, 1979.

[6] ftp ://dimacs.rutgers.edu/pub/challenge/graph/benchmarks/color/.

[7] R. Garey and S. Johnson. *Computers and Intractability ; A Guide to the Theory of NP-Completeness*. 1990.

[8] A. Hertz, M. Plumettaz, and N. Zufferey. Variable space search for graph coloring. *Discrete Applied Mathematics*, 156(13) :2551 – 2560, 2008.

[9] http ://choco solver.org.

[10] http ://mat.gsia.cmu.edu/COLOR/instances.html.

[11] K. Jansen. The optimum cost chromatic partition problem. In *CIAC*, pages 25–36, 1997.

[12] Y. Jin, J.K. Hao, and J.P. Hamiez. A memetic algorithm for the minimum sum coloring problem. *Computers & OR*, 43 :318–327, 2014.

[13] L.G. Kroon, A. Sen, H. Deng, and A. Roy. The optimal cost chromatic partition problem for trees and interval graphs. In *WG'96*, pages 279–292, 1997.

				MinSatz		MaxSatz		ISAC	
				Encodage 1	Encodage 2	Encodage 1	Encodage 2	Encodage 1	Encodage 2
Instances	N	M	$\Sigma(G)$	Temps	Temps	Temps	Temps	Temps	Temps
S10A23	10	23	20	0	0	0	0	0	0
S10A40	10	40	34	0	15	0	0	0	0
S10A5	10	5	14	0	0	0	0	0	0
S20A100	20	100	55	172	N/A	N/A	126	N/A	0
S20A19	20	19	27	0	2213	0	0	0	0
S20A50	20	50	39	0	N/A	8	0	0	0
S20A75	20	75	52	18	N/A	1372	144	N/A	0
S20A95	20	95	50	32	N/A	N/A	29	N/A	0
S25A100	25	100	63	150	N/A	N/A	1051	N/A	0
S25A30	25	30	40	0	N/A	0	0	0	0
S30A100	30	100	65	22	N/A	N/A	242	N/A	0
S30A44	30	44	48	0	N/A	3098	0	N/A	0
S35A60	35	60	60	0	N/A	N/A	2	N/A	0

TABLE 3 – Comparaison du temps d'exécution en secondes nécessaire à la détermination de la somme chromatique d'un pool d'instances aléatoires entre l'encodage du problème MinSAT 1 et 2.

[14] E. Kubicka and A.J. Schwenk. An introduction to chromatic sums. In *CSC'89 : Proceedings of the 17th conference on ACM Annual Computer Science Conference*, pages 39–45, New York, NY, USA, 1989. ACM.

[15] C.M. LI, F. Manya, and J. Planes. Exploiting unit propagation to compute lower bounds in branch and bound maxsat solvers. In *CP 05 : Proceedings of the 11st international conference on Principles and Practice of Constraint Programming*, pages 403–414, Sitges, SPAIN, 2005.

[16] C.M. Li, F. Manyà, and J. Planes. New inference rules for max-sat. *Journal Of Arteficial Intetelligence Research*, 30 :321–359, 2007.

[17] C.M. Li, Z. Zhu, F. Manyà, and L. Simon. Minimum satisfiability and its applications. *IJCAI*, 2011.

[18] Y. Li, C. Lucet, A. Moukrim, and K. Sghiouer. Greedy algorithms for the minimum sum coloring problem. In *International Workshop : Logistics and transport*, 2009.

[19] Z. Lü and J.K. Hao. A memetic algorithm for graph coloring. *European Journal of Operational Research*, 203(1) :241 – 250, 2010.

[20] C. Lucet, F. Mendes, and A. Moukrim. An exact method for graph coloring. *Computers & Operations Research*, 33(8) :2189–2207, 2006.

[21] M. Malafiejski. *Sum coloring of graphs*, chapter 4, pages 55–66. Graph Coloring. New-York (USA), 2004.

[22] I. Méndez-Díaz and P. Zabala. A branch-and-cut algorithm for graph coloring. *Discrete Applied Mathematics*, 154(5) :826–847, 2006.

[23] A. Moukrim, K. Sghiouer, C. Lucet, and Y. Li. Lower bounds for the minimum sum coloring problem. *Electronic Notes in Discrete Mathematics*, 36 :663–670, 2010.

[24] D.C. Porumbel, J.K. Hao, and P. Kuntz. A search space "cartography" for guiding graph coloring heuristics. *Comput. Oper. Res.*, 37 :769–778, April 2010.

[25] K. Sghiouer, Y. Li, C. Lucet, and A. Moukrim. A memetic algorithm for the minimum sum coloring problem. In *Conférence Internationale en Recherche Opérationnelle*, 2010.

[26] Z. Zhou, C.M. Li, C.H. Huang, and R. Xu. An exact algorithm with learning for the graph coloring problem. *Computers & OR*, 51 :282–301, 2014.