Raffinement de la décomposition arborescente par fusion de clusters pour guider DGVNS

Mathieu Fontaine, Samir Loudni, Patrice Boizumault

Raffinement de décompositions arborescentes par fusion de clusters pour guider DGVNS

Mathieu Fontaine[1,2] **Samir Loudni**[1,2] **Patrice Boizumault**[1,2]

[1] Université de Caen Basse-Normandie, UMR 6072 GREYC, F-14032 Caen, France
[2] CNRS, UMR 6072 GREYC, F-14032 Caen, France
{prénom.nom}@unicaen.fr

Résumé

Dans nos précédents travaux, nous avons proposé la méthode DGVNS (Decomposition Guided VNS) permettant d'exploiter les clusters, issus de la décomposition arborescente du graphe de contraintes, pour guider l'exploration des voisinages dans les méthodes de type VNS. Dans cet article, nous proposons deux schémas de raffinement de la décomposition arborescente permettant de limiter la redondance entre clusters, afin de guider DGVNS vers des voisinages plus pertinents. A cet effet, nous proposons deux critères : le premier exploite la taille du plus grand séparateur, alors que le second prend en considération à la fois la taille des séparateurs et la taille des clusters. Les expérimentations menées sur des instances aléatoires (GRAPH) et des instances réelles (RLFAP, SPOT5 et tagSNP) montrent la pertinence et l'intérêt de notre approche.

Abstract

In this paper, we propose two schemes for refining tree decomposition in order to reduce the redundancy between clusters and to guide DGVNS (Guided Decomposition VNS) to more relevant neighborhoods structures. The main idea is to increase the proportion of proper variables in clusters by merging those having few proper variables in the tree decomposition. For this aim, we propose two criteria to merge clusters : the first one uses the size of the largest separator, while the second one takes into account both the size of the separators and the size of clusters. Experiments conducted on several difficult instances (RLFAP, SPOT5, GRAPHand tagSNP) show the relevance and the interest of strengthening the quality of neighborhoods in DGVNS.

1 Introduction

La notion de décomposition arborescente, proposée par Robertson et Seymour [24], vise à découper un problème en sous-problèmes (*clusters*) constituant un graphe acyclique. Chaque cluster correspond à un sous-ensemble de variables fortement connectées. Chaque sous-problème, étant plus petit que le problème original, devient plus facile à résoudre. L'intérêt d'exploiter les propriétés structurelles d'un problème a été attestée dans divers domaines : pour vérifier la satisfiabilité dans SAT [1, 14, 23] pour résoudre les CSP (CTE [7]), dans les réseaux bayésiens (AND/OR graph search [21]), en bases de données relationnelles [12, 13], pour les problèmes d'optimisation sous contraintes (BTD [27], Lc-BTD$^+$ [5], RDS-BTD [25], DB [19]). Toutes ces propositions exploitent la décomposition arborescente dans des méthodes de recherche complète.

Dans nos précédents travaux [9, 10], nous avons proposé la méthode DGVNS (Decomposition Guided VNS) permettant d'exploiter les clusters issus de la décomposition arborescente du graphe de contraintes, pour guider l'exploration des voisinages dans les méthodes de type VNS. Les expérimentations menées sur des instances aléatoires (GRAPH) et des instances réelles (CELAR, SPOT5 et tagSNP) montrent la pertinence de notre approche comparée à VNS/LDS+CP [20] ou à ID-Walk [22]. Cependant, pour la plupart de ces instances, les décompositions obtenues par MCS [1] comportent de nombreux clusters qui se chevauchent très fortement. De plus, ces clusters contiennent peu ou pas de variables propres. Cette forme de redondance entre clusters limite l'effort de diversification de DGVNS, en considérant plusieurs fois des voisinages très proches.

Dans cet article, nous proposons deux raffinements de la décomposition arborescente, permettant de limiter la redondance entre clusters, afin de guider

1. Maximum Cardinality Search, [26].

DGVNS vers des voisinages plus pertinents. L'idée sous-jacente est d'augmenter la proportion de variables propres dans les clusters, en fusionnant ceux qui sont globalement redondants dans la décomposition arborescente. A cet effet, nous proposons deux critères. Le premier consiste à fusionner les clusters partageant plus de variables qu'un seuil maximal fixé. Le second, basé sur la notion *d'absorption*, consiste à fusionner les clusters ayant un taux d'absorption (i.e. proportion de variables partagées) supérieur à un seuil maximal fixé. Les expérimentations menées sur des instances aléatoires (GRAPH) et des instances réelles (RLFAP, SPOT5 et tagSNP) montrent la pertinence et l'intérêt de fusionner les clusters globalement redondants pour renforcer la qualité des voisinages explorés par DGVNS.

La section 2 présente le contexte de nos travaux. La section 3 détaille nos deux schémas de raffinement de la décomposition arborescente. Les sections 4 et 5 présentent les jeux de test et les résultats expérimentaux. Enfin, nous concluons et présentons des perspectives.

2 Contexte et Définitions

2.1 Réseau de fonctions de coût

Un réseau de fonctions de coût (CFN) est un couple (X, W) où $X=\{x_1, \ldots, x_n\}$ est un ensemble de n variables et W est un ensemble de e fonctions de coût. Chaque variable $x_i \in X$ a un domaine fini D_i de valeurs qui peuvent lui être affectées. La taille du plus grand domaine est notée d. Une affectation de x_i à la valeur $a \in D_i$ est notée (x_i, a). Pour un sous-ensemble de variables $S \subseteq X$, on note D^S le produit cartésien des domaines des variables de S. Pour un n-uplet donné t, $t[S]$ représente la projection du n-uplet t sur l'ensemble de variables S. Une affectation *complète* $t=(a_1, ..., a_n)$ est une affectation de toutes les variables ; dans le cas contraire, on l'appelle affectation *partielle*. Une fonction de coût $w_S \in W$, de portée $S \subseteq X$, est une fonction $w_S : D^S \mapsto [0, k_\top]$ où $k_\top$ est un coût entier maximum (fini ou non) utilisé pour représenter les affectations interdites (exprimant des contraintes dures). Pour capturer fidèlement les contraintes dures, les coûts sont combinés par l'addition bornée $\oplus$, définie par $\alpha \oplus \beta = \min(k_\top, \alpha + \beta)$. Le problème consiste à trouver une affectation complète t de l'ensemble des variables minimisant la combinaison des fonctions de coût $\bigoplus_{w_S \in W} w_S(t[S])$.

2.2 Décomposition arborescente

Le graphe de contraintes d'un CFN est un graphe $G=(X,E)$ composé d'un sommet par variable et il existe une arête $\{u, v\} \in E$ si, et seulement si, $\exists\, w_S \in W, u, v \in S$.

Définition 1 *Une décomposition arborescente [24] de $G=(X, E)$ est un couple (C_T, T) où $T = (I, A)$ est un arbre avec pour ensemble de nœuds I et pour ensemble d'arêtes A, et $C_T = \{C_i \mid i \in I\}$ est une famille de sous-ensembles de X (appelés clusters) telle que :*
- $\cup_{i \in I} C_i = X$,
- $\forall (u, v) \in E$, $\exists C_i \in C_T$ *t.q.* $u, v \in C_i$,
- $\forall i, j, k \in I$, *si j est sur le chemin de i à k dans T, alors $C_i \cap C_k \subseteq C_j$.*

Nous notons $\texttt{parent}(C_i)$ (resp. $\texttt{fils}(C_i)$) le parent (resp. l'ensemble des fils) de C_i dans T.

Définition 2 *L'intersection entre deux clusters est appelée séparateur et notée $sep(C_i, C_j)$. Deux clusters C_i et C_j sont adjacents ssi $sep(C_i, C_j) \neq \emptyset$. Les variables appartenant à un et un seul cluster sont appelées variables propres.*

Définition 3 *Un graphe de clusters, pour une décomposition arborescente (C_T, T), est un graphe non-orienté $G_T = (C_T, E_T)$ dont les sommets sont les éléments de C_T et il existe une arête $(C_i, C_j) \in E_T$ entre les sommets C_i et C_j ssi $sep(C_i, C_j) \neq \emptyset$.*

La largeur d'une décomposition arborescente $T=(I, A)$ est définie par $w^-(T)=\max_{i \in I}(|C_i|-1)$. La largeur de décomposition $tw(G)$ d'un graphe G est la plus petite largeur de toutes les décompositions arborescentes possibles de G. Calculer la largeur de décomposition d'un graphe est un problème NP-complet [2]. Cependant, des heuristiques reposant sur la notion de *triangulation* de graphe [2] permettent le calcul de décompositions approchées. Elles fournissent un majorant de la largeur de décomposition.

Dans cet article, nous utilisons l'heuristique *Maximum Cardinality Search* (MCS) [26] qui constitue un bon compromis entre la largeur de la décomposition obtenue et le temps nécessaire à son calcul [17].

2.3 Decomposition Guided VNS

DGVNS (Decomposition Guided VNS) [9, 10] étend le principe de VNDS [15] (Variable Neighborhood Decomposition Search), en exploitant le graphe de clusters pour guider l'exploration de grands voisinages.

Les voisinages sont obtenus en désaffectant une sous-partie de la solution courante selon une heuristique de choix de variables. La reconstruction de la solution sur les variables désinstanciées est effectuée par une recherche arborescente partielle, LDS (*Limited discrepancy search*, [16]), aidée par la propagation des contraintes (CP) basée sur un calcul de minorants.

2. Un graphe est cordal ou triangulé si et seulement si tous ses cycles de taille supérieure à quatre ont une corde (une arête connectant deux sommets non-adjacents du cycle).

Algorithm 1: Pseudo-code de DGVNS

```
   function DGVNS(X, W, k_init, k_max, δ_max);
   begin
1      let G be the constraints graph of (X, W) ;
2      let (C_T, T) be a tree decomposition of G ;
       let C_T = {C_1, C_2, ..., C_p} ;
3      S ← genInitSol() ;
4      k ← k_init ;
5      i ← 1 ;
6      while (k < k_max) ∧ (notTimeOut) do
7          C_s ← CompleteCluster(C_i, k) ;
8          X_un ← Hneighborhood(C_s, N_{k,i}, S) ;
9          A ← S\{(x_i, a) | x_i ∈ X_un} ;
10         S' ← Rebuild(A, X_un, δ_max, f(S), S) ;
11         NeighbourhoodChangeDGVNS(S, S', k, i);
12     return S ;
   procedure NeighbourhoodChangeDGVNS (S, S', k, i);
   begin
13     if f(S') < f(S) then
14         S ← S' ;
15         k ← k_init, i ← succ(i) ;
16     else  k ← k + 1 ; i ← succ(i) ;
```

Définition 4 *Soit G_T=($\mathcal{C}_T$,E_T) le graphe de clusters associé à G. Soient $C_i \in \mathcal{C}_T$ un cluster de G_T et $k \in [1 \dots n]$ la dimension du voisinage. La structure de voisinage $N_{k,i}$ désigne l'ensemble des combinaisons de k variables parmi C_i.*

L'algorithme 1 présente le pseudo-code de DGVNS. Tout d'abord, il construit une décomposition arborescente de G (ligne 2), puis génère aléatoirement une solution initiale S (fonction genInitSol, ligne 3). Afin de favoriser les mouvements dans des régions fortement liées, DGVNS se base sur les structures de voisinages $N_{k,i}$ (cf. Définition 4). En effet, le concept de cluster permet d'exhiber ces régions, de part sa taille (plus petite que le problème initial), et de part la forte connexion entre les variables qu'il contient. Ainsi, l'ensemble de variables candidates C_s à désaffecter sont sélectionnées à partir du cluster C_i. Si ($k > |C_i|$), C_s est étendu aux variables des clusters C_j voisins de C_i afin de prendre en compte la topologie du graphe de clusters. Ce traitement est réalisé par la fonction CompleteCluster(C_i, k) (ligne 7). De plus, grâce à la forte connexion entre les variables de C_s, l'étape de reconstruction pourra bénéficier d'un plus fort filtrage et d'un meilleur calcul de minorants. Un sous ensemble X_{un} de k variables est sélectionné aléatoirement dans C_s parmi les variables en conflit par l'heuristique de voisinage Hneighborhood (line 8). Une affectation partielle $\mathcal{A}$ est générée à partir de la solution courante S en désaffectant les k variables de X_{un} (ligne 9). Ensuite, ces variables sont reconstruites (ligne 10) par une recherche arborescente partielle LDS [16], aidée par la propagation de contraintes (CP) (voir [20] pour plus de détails). La recherche s'arrête dès que la dimension maximale de voisinage k_{max} ou le *TimeOut* est atteint (ligne 6).

La procédure NeighborhoodChangeDGVNS contrôle les mécanismes d'intensification et de diversification de DGVNS (cf. Algorithme 1). Soit p le nombre total de clusters, *succ* une fonction de succession [3], et $N_{k,i}$ la structure de voisinage courante. Si LDS+CP ne trouve par de meilleure solution S' dans le voisinage de S, DGVNS cherche des améliorations dans $N_{(k+1),succ(i)}$ (la structure de voisinage où $(k+1)$ variables de C_s seront désaffectées) (ligne 16). Tout d'abord, la diversification réalisée par le déplacement du cluster C_i au cluster $C_{succ(i)}$ permet de favoriser l'exploration de nouvelles parties de l'espace de recherche et de rechercher de meilleures solutions dans celles-ci. Ensuite, quand un optimum local est atteint dans le voisinage courant, l'augmentation de la dimension du voisinage k permet aussi de diversifier la recherche en explorant de plus grandes régions.

Quand une solution de meilleure qualité S' est trouvée par LDS+CP dans le voisinage $N_{k,i}$, S' devient la solution courante (ligne 14), k est réinitialisé à k_{init} et le prochain cluster est considéré (ligne 15). En effet, rester dans le même cluster rend plus difficile la découverte de nouvelles solutions : se déplacer sur un nouveau cluster permet la diversification de la recherche autour de la nouvelle solution S'.

3 Raffinements par fusion de clusters

Comme nous l'avons indiqué dans la Section 1, la plupart des décompositions arborescentes, obtenues par MCS sur les différents problèmes étudiés, comportent de nombreux clusters contenant peu ou pas de variables propres, car se chevauchant très fortement. Cette forme de redondance entre clusters limite l'effort de diversification de DGVNS, en considérant, de manière répétée, des voisinages très proches.

Afin de limiter la redondance entre clusters, et donc de renforcer la qualité des structures de voisinage explorées par DGVNS, nous proposons de fusionner les clusters redondants dans la décomposition arborescente. L'idée sous-jacente est d'augmenter la proportion de variables propres dans les clusters. À cet effet, nous proposons deux critères : (i) la taille du plus grand séparateur (s_{max}), et (ii) le taux d'absorption maximal (Ab_{max}). Nous désignerons par DGVNS-*smax* (resp. DGVNS-*abs*) la méthode DGVNS exploitant la décomposition arborescente obtenue par le critère s_{max} (resp. le critère Ab_{max}). Notons que toutes les opérations de fusion sont effectuées à l'issue de la décomposition (cf. ligne 2, algorithme 1).

3. si $i < p$ alors $succ(i) = i + 1$ sinon $succ(p) = 1$.

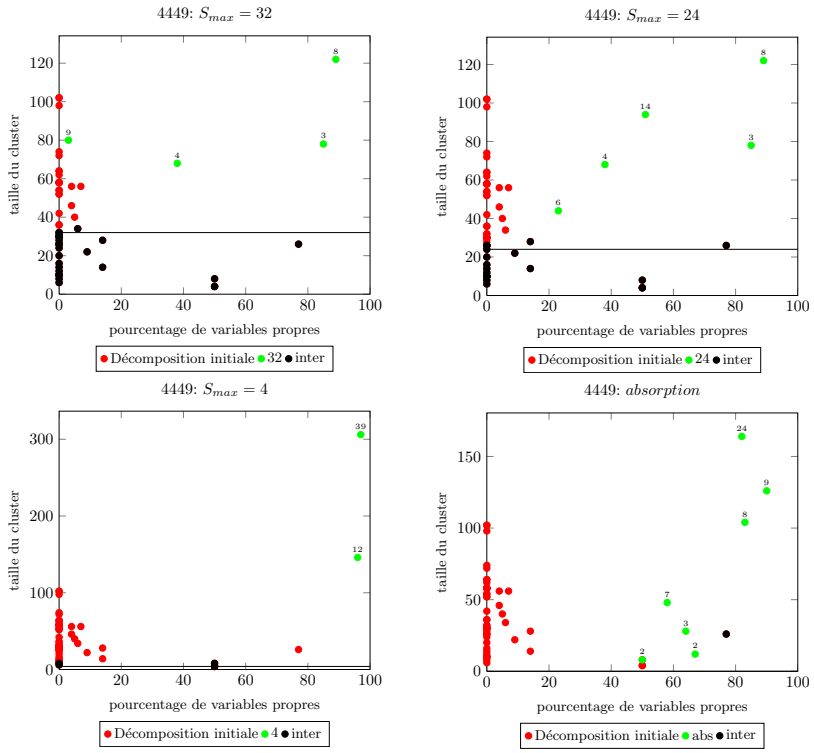

FIGURE 1 – Comparaison des décompositions obtenues par les deux méthodes de fusion sur une instance `tagSNP`.

3.1 Fusion de clusters basée sur s_{max}

La problématique de fusion de clusters est au coeur des méthodes complètes exploitant les décompositions arborescentes [6]. Dans [18], les auteurs proposent de fusionner les clusters en se basant sur le critère s_{max} afin d'augmenter le degré de liberté pour les heuristiques de choix de variables au sein des clusters. Dans [25], les auteurs utilisent le même principe pour réduire la complexité spatiale de la méthode utilisée.

Afin de générer de nouvelles décompositions à partir de celles obtenues par `MCS`, nous avons suivi le schéma proposé dans [18], en bornant la taille maximale des séparateurs s_{max}. Tout d'abord, nous calculons une décomposition arborescente en fixant s_{max} à $+\infty$ (cas où la taille des séparateurs n'est pas limitée). Ensuite, en partant des feuilles de la décomposition produite par `MCS`, nous fusionnons les clusters C_i qui partagent plus de s_{max} variables avec leur parent $\texttt{parent}(C_i)$. Nous obtenons ainsi une nouvelle décomposition, dans laquelle aucun séparateur ne contient plus de s_{max} variables.

Cette approche souffre cependant de sa rigidité. En effet, le critère de fusion ne permet pas de prendre en compte la structure de l'instance considérée : limiter s_{max} à 32 sur des instances de problèmes ayant des séparateurs de grande taille conduira à fusionner tous les clusters. De plus, avec ce critère, on pourra fusionner deux clusters de taille 100 partageant 32 variables mais pas ceux de taille 32 partageant 31 variables. Pour remédier à ces inconvénients nous proposons une fusion basée sur la notion *d'absorption*.

3.2 Fusion de clusters basée sur l'absorption

Afin de tenir compte des tailles respectives des clusters et de leurs séparateurs, dans le mécanisme de fusion, nous introduisons un second critère appelé *absorption*.

Définition 5 *Soit (C_T, T) une décomposition arborescente. Soit C_i et C_j deux clusters de C_T. L'absorption de C_i par C_j, notée $Ab(C_i, C_j)$, est définie par :*

$$Ab(C_i, C_j) = \max\left(\frac{|Sep(C_i, C_j)|}{|C_i|}, \frac{|Sep(C_i, C_j)|}{|C_j|}\right)$$

Soit Ab_{max} le taux d'absorption maximal autorisé, et min_{size} la taille minimale autorisée pour un cluster. La fusion s'effectue sur la décomposition obtenue par `MCS`. En partant des feuilles de cette décomposition, tous les clusters $< C_i, \texttt{parent}(C_i) >$ ayant un taux d'absorption supérieur à Ab_{max} ou tels que $|C_i| < min_{size}$ sont fusionnés.

Pour nos expérimentations (cf. Section 5), la taille minimale d'un cluster (min_{size}) a été fixée à la valeur 5 et Ab_{max} à 70%. Ainsi, les clusters partageant un peu plus de 2/3 de leurs variables seront fusionnés.

3.3 Comparaison des deux méthodes de fusion

Cette section compare, de manière qualitative, les décompositions obtenues par les deux méthodes de fusion sur une instance du problème `tagSNP` et sur une instance du problème `SPOT5` (cf. Section 4).

Les résultats de ces comparaisons sont présentés aux Figures 1 et 2. Pour chaque figure, les 3 premiers graphiques correspondent à la fusion s_{max} et le dernier correspond à la fusion par absorption. Chaque graphique indique, pour une valeur fixée de s_{max}, (i) la répartition des clusters de la décomposition initiale obtenue par `MCS`, et (ii) la répartition des clusters obtenus par fusion, en fonction de leur taille et de leur pourcentage de variables propres. Les points rouges représentent les clusters de la décomposition initiale, les points verts les clusters obtenus après fusion, et les points noirs les clusters communs. Chaque point vert indique aussi le nombre de clusters qui ont été fusionnés pour son obtention.

Considérons les résultats obtenus pour l'instance `tagSNP` #4449 (cf. Fig. 1). Pour (s_{max}=32), 24 clusters ont été fusionnés en 4 nouveaux clusters, dont un partageant plus de 90% de ses variables et un autre contenant plus de 120 variables. Mais, un grand nombre de clusters ayant 100% de leurs variables partagées n'ont pas été fusionnés. (s_{max}=24) permet de réduire quelque peu cette redondance. Par contre, la fusion par absorption permet de supprimer la totalité des clusters redondants et de générer une décomposition contenant des clusters ayant au moins 50% de variables propres. Par ailleurs, sur cette instance, $s_{max} = 14$. On peut dresser les mêmes constats sur la très grande majorité des autres instances `tagSNP`.

Considérons les résultats obtenus pour l'instance `SPOT5` #507 (cf. Fig. 2). Pour (s_{max}>12), les décompositions produites sont très similaires à la décomposition initiale. Celles-ci sont caractérisées par des clusters qui se recouvrent très fortement. Pour ($s_{max}\leq$12), les décompositions obtenues ont des largeurs arborescentes plus élevées, mais les clusters contiennent plus de variables propres. À l'inverse, la fusion par absorption produit une décomposition constituée de beaucoup plus de clusters qui se chevauchent moins fortement (i.e., $s_{max} = 10$). Par ailleurs, la valeur de la largeur de décomposition est deux fois moins élevée comparée à celle obtenue pour (s_{max}=4). On peut dresser les mêmes constats sur une grande majorité des autres instances `SPOT5`.

Les résultats obtenus montrent clairement l'apport de la fusion par absorption. En effet, pour des valeurs de s_{max} élevées, la fusion modifie peu la décomposition initiale. En revanche, pour des valeurs de s_{max} faibles, les décompositions produites ont peu de clusters et possèdent des largeurs arborescentes très élevées. De plus, la plupart des clusters partageant 100% de leurs variables sont conservés. À l'inverse, la fusion par absorption permet de produire des décompositions de bien meilleure qualité : des clusters de taille raisonnable ayant une forte proportion de variables propres. On observe toutefois une augmentation relative de la largeur de décomposition sur certaines instances.

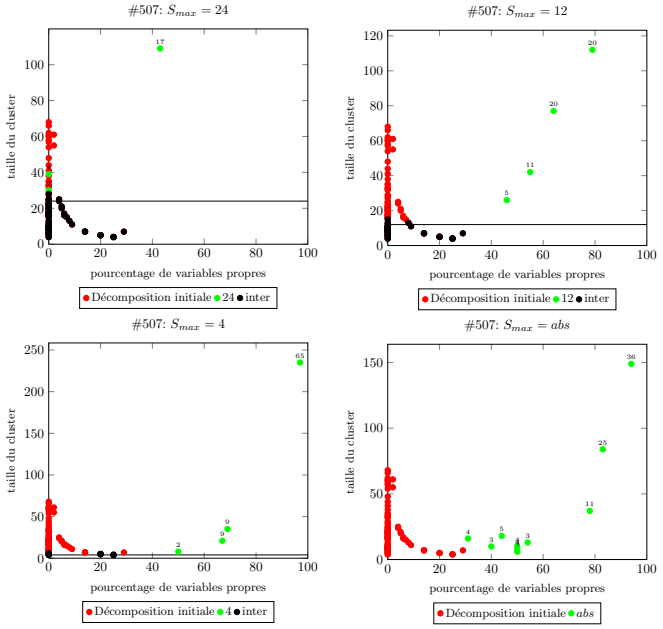

FIGURE 2 – Comparaison des décompositions obtenues par les deux méthodes de fusion sur une instance `SPOT5`.

4 Jeux de test

Les expérimentations ont été menées sur différentes instances de quatre problèmes différents.

- Instances `RLFAP` : Le `CELAR` (Centre d'électronique de l'Armement) a mis à disposition un ensemble d'instances du problème d'affectation de fréquences radio (`RLFAP`) [4]. L'objectif est d'assigner un nombre limité de fréquences à un ensemble de liens radios entre des couples de sites, afin de minimiser les interférences dues à la réutilisation des fréquences. Nous reportons les résultats sur les instances les plus difficiles : Scen06, Scen07 et Scen08.

- Instances `GRAPH` : Le générateur `GRAPH` (Generating Radio link frequency Assignment Problems Heuristically) a été développé par le projet CALMA [28] afin de proposer des instances aléatoires ayant une structure proche des instances `RLFAP`.

- Instances `SPOT5` : La planification quotidienne d'un satellite d'observation de la terre (`SPOT5`) consiste à sélectionner les prises de vue à effectuer dans la journée en prenant en compte les limites matérielles du

Inst.	s_{max}	Succ.	Temps	Moy.	$\|C_T\|$	Taille C_i moy.	max.
Scen06	4	50/50	368	3,389	45	4.1	20
$e = 1222$	$+\infty$	**50/50**	**112**	**3,389**	55	4.9	12
$S^* = 3,389$							
Scen07	4	5/50	1,908	344,215	76	4.1	52
$n = 200$	8	8/50	2,379	343,803	96	4.64	39
$e = 2,665$	12	7/50	2,400	343,601	100	4.85	35
$S^* = 343,592$	16	5/50	2,305	343,795	105	5.31	26
	24	5/50	2,272	348,640	109	5.72	23
	$+\infty$	**40/50**	**317**	**345,614**	110	5.72	23
Scen08	4	-	-	314	180	3.98	66
$n = 458$	8	-	-	314	228	4.5	65
$e = 5,286$	12	-	-	313	243	4.88	43
$S^* = 262$	16	-	-	312	251	5.17	31
	24	-	-	312	258	5.49	27
	$+\infty$	**3/50**	**1,811**	**275**	259	5.51	27
Graph06	4	50/50	287	4,123	9	24.22	193
$n = 200$	8	50/50	256	4,123	36	11.47	164
$e = 1,970$	12	50/50	262	4,123	41	11.32	160
$S^* = 4,123$	16	50/50	277	4,123	43	11.4	158
	24	50/50	263	4,123	49	12.43	151
	32	**50/50**	**230**	**4,123**	50	12.74	150
	$+\infty$	50/50	367	4,123	123	36.07	63
Graph11	4	-	-	31,286	8	45.5	331
$n = 340$	8	-	-	24,685	45	13.6	292
$e = 3417$	12	-	-	23,453	60	12.73	277
$S^* = 3,080$	16	-	-	23,353	63	12.75	274
	24	-	-	23,238	71	13.48	265
	32	-	-	23,144	73	13.9	263
	$+\infty$	**8/50**	**3,046**	**4,234**	191	65.04	118
Graph13	4	-	-	19,104	7	67.0	453
$n = 458$	8	3/50	2,428	12,010	62	13.32	397
$e = 4,915$	12	2/50	2,893	12,053	74	12.77	385
$S^* = 10,110$	16	2/50	3,129	12,192	75	12.8	384
	24	1/50	3,473	11,999	83	13.42	376
	32	**7/50**	**3,247**	**11,766**	85	13.76	374
	$+\infty$	-	-	22,489	262	83.98	155

TABLE 1 – Comparaison de DGVNS pour différentes valeurs de s_{max} sur les instances RLFAP et GRAPH.

Inst.	s_{max}	Succ.	Temps	Moy.	$\|C_T\|$	Taille C_i moy.	max.
#408	4	49/50	361	6,228	17	13.88	127
$n = 200$	**8**	**49/50**	**78**	**6,228**	34	10.03	89
$e = 2,232$	12	44/50	461	6,228	48	10.17	76
$S^* = 62,28$	16	44/50	845	6,228	56	10.79	67
	24	43/50	460	6,228	62	11.52	65
	32	42/50	298	6,228	71	14.0	45
	$+\infty$	49/50	117	6,228	72	14.46	43
#412	**4**	**48/50**	**136**	**32,381**	16	21.0	228
$n = 300$	8	48/50	241	32,381	34	13.18	132
$e = 4,348$	12	47/50	211	32,381	45	12.42	132
$S^* = 32,381$	16	43/50	365	32,381	51	12.65	101
	24	44/50	512	32,381	75	14.95	61
	32	42/50	619	32,381	89	17.02	58
	$+\infty$	36/50	84	32,381	96	19.04	45
#414	4	38/50	935	38,478	19	21.05	289
$n = 364$	**8**	**45/50**	**532**	**38,478**	38	13.68	192
$e = 2,242$	12	45/50	721	38,478	49	12.88	192
$S^* = 21,253$	16	36/50	654	38,478	55	13.04	161
	24	35/50	594	38,478	73	14.53	159
	32	41/50	826	38,478	77	15.31	158
	$+\infty$	38/50	554	38,478	99	26.68	111
#505	4	50/50	99	21,253	27	11.15	174
$n = 240$	8	50/50	212	21,253	50	9.0	71
$e = 2,242$	12	50/50	116	21,253	65	9.34	53
$S^* = 21,253$	16	49/50	218	21,253	82	10.38	42
	24	49/50	232	21,253	91	11.4	33
	$+\infty$	**50/50**	**63**	**21,253**	93	11.69	31
#507	4	32/50	565	27,390	20	17.55	235
$n = 311$	8	35/50	544	27,390	39	12.05	145
$e = 5,732$	**12**	**38/50**	**663**	**27,390**	49	11.71	112
$S^* = 27,390$	16	28/50	542	27,390	60	12.32	111
	24	29/50	1,039	27,390	81	14.28	109
	32	30/50	1,248	27,390	86	15.21	108
	$+\infty$	33/50	71	27,390	101	20.48	68
#509	4	43/50	1,036	36,446	19	20.21	273
$n = 348$	**8**	**47/50**	**438**	**36,446**	38	13.24	176
$e = 8,624$	12	45/50	814	36,446	50	12.52	145
$S^* = 36,446$	16	38/50	794	36,446	57	12.82	143
	24	29/50	1,039	27,390	73	14.23	143
	32	47/50	681	36,446	77	14.96	142
	$+\infty$	40/50	265	36,446	94	29	100

TABLE 2 – Comparaison de DGVNS pour différentes valeurs de s_{max} sur les instances SPOT5.

satellite, tout en maximisant l'importance des photographies sélectionnées [3]. Nous reportons les résultats sur six instances sans contraintes dures de capacité.

- Instances tagSNP : Un SNP (Single Nucleotide Polymorphism) -ou polymorphisme nucléotidique- est la variation d'une seule paire de nucléotides dans l'ADN de deux individus d'une même espèce ou dans une paire de chromosomes d'un même individu. Les SNP sont des marqueurs biologiques qui peuvent être utilisés pour la prédiction des risques de développement de certaines maladies [8, 11]. Le problème de sélection des tagSNP consiste à choisir un sous-ensemble de SNP, appelé tagSNP, qui permet de capturer le maximum d'information génétique. Ce problème est considéré comme très difficile du fait de sa proximité avec le problème de *set covering* (NP-difficile) [25]. Nous rapportons les résultats des expérimentations menées sur 10 instances issues des données du chromosome1 humain [4] avec r_0=0.5 (instances modélisées sous forme de CFN binaires [25] ayant jusqu'à $n = 1550$ variables, avec des domaines de taille d allant de 30 à 266 valeurs et jusqu'à $e = 250,000$ fonctions de coût). 8 instances sont de taille moyenne et 2 de grande taille.

5 Expérimentations

5.1 Protocole expérimental

Chaque méthode a été appliquée sur chaque instance, avec une *discrepancy* de 3 pour LDS, ce qui correspond à la meilleure valeur trouvée sur les instances RLFAP [20]. Les valeurs de k_{min} et k_{max} ont été respectivement fixées à 4 et n (nombre total de variables). Le $TimeOut$ à été fixé à 3 600 secondes pour les instances RLFAP et SPOT5. Pour les instances tagSNP de taille moyenne (resp. de grande taille), le $TimeOut$ a été fixé à 2 heures (resp. 4 heures). Nous avons considéré différentes valeurs de s_{max} : $4, 8, 12, 16, 24$ et 32. Un ensemble de 50 essais par instance a été réalisé sur un AMD opteron 2.1 GHz et 256 Go de RAM. Toutes les méthodes ont été implantées en C++ en utilisant la librairie toulbar2 [5].

Pour chaque instance et chaque méthode, nous reportons (i) le nombre de succès (l'optimum est atteint), (ii) le temps de calcul moyen pour atteindre l'optimum, (iii) le coût moyen des solutions trouvées sur les 50 essais, (iv) le nombre de clusters de la décomposition obtenue, ainsi que (v) la valeur moyenne et la valeur maximale de la taille des clusters.

Dans cette section, nous évaluons successivement les apports de la fusion basée sur la taille maximale des séparateurs (cf. Section 5.2) et les apports de la fusion basée sur le taux d'absorption (cf. Section 5.3). Puis, nous dressons une étude comparative des deux méthodes de fusion de clusters (cf. Section 5.4).

4. http://www.costfunction.org/benchmark

5. http://carlit.toulouse.inra.fr/cgi-bin/awki.cgi/SoftCSP

Inst.	s_{max}	Succ.	Temps	Moy.	$\|C_T\|$	Taille C_i moy.	max.
#3792	4	50/50	2,381	6,359,805	11	49.36	283
$n = 528$	8	50/50	1,498	6,359,805	20	29.8	255
$d = 59$	12	50/50	1,391	6,359,805	29	23.68	157
$e = 12,084$	16	50/50	959	6,359,805	37	21.64	139
$S^* = 5,094,256$	24	50/50	1,028	6,359,805	42	21.42	133
	32	**50/50**	**832**	**6,359,805**	**49**	**22.18**	**121**
	$+\infty$	50/50	954	6,359,805	70	30.37	95
#4449	4	50/50	2,399	5,094,256	7	67.85	305
$n = 464$	8	50/50	1,396	5,094,256	16	33	195
$d = 64$	12	50/50	1,216	5,094,256	17	31.70	195
$e = 12,540$	16	50/50	996	5,094,256	19	29.73	159
$S^* = 5,094,256$	24	50/50	663	5,094,256	26	27.07	121
	32	**50/50**	**587**	**5,094,256**	**49**	**22.18**	**121**
	$+\infty$	50/50	665	5,094,256	56	36.71	101
#9319	4	50/50	1,027	6,477,229	14	41.85	197
$n = 562$	8	50/50	853	6,477,229	22	28.72	173
$d = 58$	12	50/50	897	6,477,229	30	23.93	171
$e = 14,811$	16	50/50	698	6,477,229	35	22.48	115
$S^* = 6,477,229$	24	50/50	656	6,477,229	40	22	115
	32	**50/50**	**555**	**6,477,229**	**44**	**22.5**	**115**
	$+\infty$	50/50	788	6,477,229	62	34.41	89
#16421	4	50/50	2,170	3,436,849	5	82.2	207
$n = 404$	8	50/50	2,273	3,436,849	7	60.14	201
$d = 75$	12	50/50	2,209	3,436,849	11	42.27	201
$e = 12,138$	16	50/50	2,369	3,436,849	14	36.28	133
$S^* = 3,436,849$	24	50/50	1,253	3,436,849	17	33.47	133
	32	**50/50**	**1,038**	**3,436,849**	**20**	**32.8**	**133**
	$+\infty$	50/50	2,673	3,436,849	35	42.54	113
#16706	4	50/50	131	2,632,310	11	41.36	129
$n = 438$	8	50/50	132	2,632,310	19	26.68	125
$d = 30$	12	50/50	105	2,632,310	33	19.66	55
$e = 6,321$	**16**	**50/50**	**99**	**2,632,310**	**37**	**19**	**51**
$S^* = 2,632,310$	24	50/50	124	2,632,310	46	18.91	49
	32	50/50	137	2,632,310	49	19.45	49
	$+\infty$	49/50	153	2,632,310	49	19.45	49

TABLE 3 – Comparaison de DGVNS pour différentes valeurs de s_{max} sur les instances tagSNP.

Inst.	s_{max}	Succ.	Temps	Moy.	$\|C_T\|$	Taille C_i moy.	max.
#6835	4	46/50	10,781	4,571,140	3	167	469
$n = 496$	8	46/50	9,925	4,571,122	3	167	469
$d = 90$	12	50/50	8,075	4,571,108	7	76.71	443
$e = 18,003$	16	50/50	7,966	4,571,108	7	76.71	443
$S^* = 4571108$	24	50/50	3,775	4,571,108	16	44.62	235
	32	50/50	3,321	4,571,108	23	39.26	229
	$+\infty$	**50/50**	**2,409**	**4,571,108**	**56**	**53.5**	**109**
#8956	4	42/50	9,037	6,660,310	9	56.11	215
$n = 486$	8	46/50	7,109	6,660,309	14	38.42	215
$d = 106$	12	44/50	7,592	6,660,310	15	36.6	215
$e = 20,832$	16	42/50	7,580	6,660,310	16	35.12	215
$S^* = 6,660,308$	24	41/50	7,015	6,660,311	26	29.53	215
	32	46/50	7,192	6,660,309	30	29.33	215
	$+\infty$	**50/50**	**4,911**	**6,660,308**	**54**	**52.03**	**185**
#14226	4	26/50	11,973	26,131,481	20	54.7	237
$n = 1,058$	8	13/50	11,737	26,387,778	31	37.19	185
$d = 95$	12	28/50	11,657	25,758,768	38	32.21	185
$e = 36,801$	16	35/50	11,333	25,828,540	42	30.52	179
$S^* = 25,665,437$	24	46/50	10,930	25,712,052	58	27.48	179
	32	22/50	10,434	26,108,198	66	27.54	179
	$+\infty$	**46/50**	**7,606**	**25,688,751**	**94**	**38.19**	**159**
#15757	4	50/50	195	2,278,611	13	27.61	215
$n = 342$	8	50/50	82	2,278,611	19	21	101
$d = 47$	**12**	**50/50**	**54**	**2,278,611**	**25**	**18.44**	**101**
$e = 5,091$	16	50/50	64	2,278,611	29	17.82	73
$S^* = 2,278,611$	24	50/50	72	2,278,611	41	18.31	73
	32	50/50	100	2,278,611	43	18.62	71
	$+\infty$	50/50	60	2,278,611	45	20.24	67
#17034	4	5/50	13,096	39,635,091	21	56.23	445
$n = 1,142$	8	2/50	11,979	39,665,758	30	41.26	445
$d = 123$	12	5/50	12,419	39,573,906	42	32.42	233
$e = 47,967$	16	14/50	12,149	39,359,507	55	27.98	231
$S^* = 38,318,224$	24	24/50	11,407	39,114,487	63	27.19	231
	32	32/50	11,058	38,838,844	79	27.12	229
	$+\infty$	**41/50**	**8,900**	**38,563,232**	**139**	**52.95**	**189**

TABLE 4 – Comparaison de DGVNS pour différentes valeurs de s_{max} sur les instances tagSNP.

5.2 Apports de la fusion basée sur s_{max}

Dans cette section, nous considérons différentes valeurs de s_{max} et nous comparons les performances respectives des DGVNS-s_{max} sur les instances RLFAP, SPOT5, GRAPH et tagSNP que nous avons retenues. s_{max}=$+\infty$ désigne le cas où aucune fusion n'est effectuée sur la décomposition initiale obtenue par MCS.

Instances RLFAP**.** Sur ces trois instances (cf. Table 1), la fusion dégrade considérablement les performances de DGVNS en termes de taux de succès et de temps de calcul, comparé à $s_{max} = +\infty$. Pour l'instance Scen06, toutes les décompositions donnent 100% de succès mais c'est $s_{max} = +\infty$ qui obtient les meilleurs temps de calcul. Notons que, sur cette instance, la taille du plus grand séparateur de la décomposition initiale est de 8. Par conséquent, toutes les décompositions obtenues avec $s_{max} \geq 8$ sont identiques. Seuls les résultats de DGVNS-4 sont reportés. Pour l'instance Scen07, DGVNS-8 obtient cinq fois moins de succès et le temps de calcul moyen est multipliée par 7. Enfin, pour Scen08, DGVNSs_{max} est incapable de trouver l'optimum. La meilleure solution trouvée est de coût 309.

Instances GRAPH**.** Les résultats varient selon les instances (cf. Table 1). Pour Graph11, la fusion dégrade considérablement les performances de DGVNS. En revanche, pour les deux autres instances, la fusion procure des gains substantiels. En effet, pour Graph06, DGVNS-4 améliore le temps de calcul de DGVNS d'environ 21% comparé à s_{max}=$+\infty$. Cette amélioration atteint 37% avec s_{max}=32 (meilleur valeur de s_{max}). Pour Graph13, DGVNS-s_{max} atteint l'optimum pour la première fois (excepté pour s_{max}=4). Les meilleurs résultats sont obtenus avec s_{max}=32 : 7/50 essais avec succès et une déviation moyenne de l'optimum d'environ 16%.

Instances SPOT5**.** Sur les instances SPOT5 (cf. Table 2), la fusion permet d'améliorer très sensiblement les performances de DGVNS à la fois en termes de taux de succès et en temps de calcul. Pour la plupart de ces instances (exceptée #507), s_{max}=8 donne les meilleurs résultats. En effet, cette valeur présente un meilleur compromis entre le nombre de clusters, la taille moyenne des clusters et la taille du plus grand cluster (cf. les colonnes 6-8, Table 2). Pour les instances #412 (resp. #509), DGVNS-8 améliore le taux de succès de DGVNS de 24% (resp. 14%).

Instances tagSNP**.** Sur les cinq instances de tagSNP décrites à la Table 4, la fusion dégrade considérablement les performances de DGVNS aussi bien en termes de taux de succès qu'en temps de calcul. Cette tendance s'amplifie pour les petites valeurs de s_{max} (≤ 12). Pour l'instance #8956, te taux de succès diminue de 8% (resp. 18%) pour s_{max}=32 (resp. s_{max}=24), et le temps de calcul augmente de 42% (resp. 46%) pour s_{max}=24 (resp. s_{max}=32). Pour $s_{max} \in \{12, 16\}$, le temps de calcul est multiplié par 2. On retrouve des comportements similaires sur les instances #14226 et

Inst.	Méthode	Succ.	Temps	Moy.	$\|C_T\|$	Taille C_i moy.	max.
Scen06	DGVNS	50/50	112	3,389	55	4.9	12
S^* =3,389	DGVNS-4	50/50	368	3,389	45	4.1	20
	DGVNS-abs	**50/50**	**103**	**3,389**	**11**	**11.82**	**26**
Scen07 2	DGVNS	40/50	317	345,614	110	5.72	23
S^* =343,59	DGVNS-8	8/50	2,379	343,803	106	4.64	39
	DGVNS-abs	**50/50**	**514**	**343,592**	**17**	**14.82**	**49**
Scen08	DGVNS	3/50	1,811	275	276	5.51	22
S^* =262	DGVNS-16	-	-	312	251	5.17	31
	DGVNS-abs	**5/50**	**726**	**274**	**47**	**12.02**	**63**
Graph06	DGVNS	50/50	367	4,123	119	36.07	61
S^* =4,123	DGVNS-32	**50/50**	**230**	**4,123**	**50**	**12.74**	**150**
	DGVNS-abs	50/50	260	4,123	4	69.5	134
Graph11	DGVNS	8/50	3,046	4,234	194	65.04	121
S^* =3,080	DGVNS-32	-	-	23,144	73	13.9	263
	DGVNS-abs	**25/50**	**2,967**	**3,789**	**3**	**122.0**	**320**
Graph13	DGVNS	-	-	22,489	265	83.98	147
S^* =10,110	DGVNS-**32**	**7/50**	**3,247**	**11,766**	85	13.76	374
	DGVNS-abs	**1/50**	**3,520**	**16,756**	**5**	**134.2**	**292**

TABLE 5 – Comparaison entre DGVNS, DGVNS-abs et DGVNS-s_{max} sur les instances RLFAP et GRAPH.

Inst.	Méthode	Succ.	Temps	Moy.	$\|C_T\|$	Taille C_i moy.	max.
#408	DGVNS	49/50	117	6,228	72	14.46	43
S^* =6,228	DGVNS-8	49/50	78	6,228	34	10.03	89
	DGVNS-abs	**50/50**	**71**	**6,228**	**9**	**25.56**	**84**
#412	DGVNS	36/50	84	32,381	96	19.04	45
S^* =32,381	DGVNS-4	48/50	136	32,381	16	21.0	228
	DGVNS-abs	**50/50**	**84**	**32,381**	**8**	**41.25**	**137**
#414	DGVNS	38/50	554	38,478	99	26.68	111
S^* =38,478	DGVNS-8	45/50	532	38,478	38	13.68	192
	DGVNS-abs	**48/50**	**635**	**38,478**	**8**	**49.25**	**200**
#505	DGVNS	**50/50**	**63**	**21,253**	**93**	**11.69**	**31**
S^* =21,253	DGVNS-4	50/50	99	21,253	27	11.15	174
	DGVNS-abs	50/50	76	21,253	15	21.33	71
#507	DGVNS	33/50	71	27,390	101	20.48	68
S^* =27,390	DGVNS-12	38/50	663	27,390	49	11.71	112
	DGVNS-abs	**49/50**	**665**	**27,390**	**10**	**35.1**	**149**
#509	DGVNS	40/50	265	36,446	94	29	100
S^* =36,446	DGVNS-8	47/50	438	36,446	38	13.24	176
	DGVNS-abs	**50/50**	**437**	**36,446**	**9**	**42.67**	**184**

TABLE 6 – Comparaison entre DGVNS, DGVNS-abs et DGVNS-s_{max} sur les instances SPOT5.

#17034, pour lesquelles DGVNS-1 obtient les plus mauvais résultats, plus particulièrement pour des petites valeurs de s_{max}.

Les contre-performances de DGVNS-1 sur ces instances (cf. Table 4) peuvent s'expliquer par le fait que la fusion basée sur s_{max} tend à augmenter les largeurs (w^-) des décompositions obtenues, en particulier pour des petites valeurs de s_{max}. Ainsi, l'impact de la stratégie de diversification de DGVNS s'en trouve fortement affaiblie : les voisinages deviennent trop grands et leur nombre trop petit. Pour l'instance #6835, avec s_{max}=4, w^- passe de 108 à 468 et le nombre de clusters diminue de 56 à 3.

A l'inverse, pour les cinq instances de la Table 3, la fusion permet d'améliorer les performances de DGVNS. Pour la plupart de ces instances (excepté l'instance #16706), s_{max}=32 donne les meilleurs résultats. Pour l'instance #16421, avec s_{max}=32, le temps de calcul moyen est divisé par 2 comparé à s_{max}=$+\infty$, tandis que pour l'instance #9315, le gain en temps de calcul est d'environ 29%. Pour les autres instances, ce gain est d'environ 12%.

Sur ces instance, la fusion permet de supprimer les clusters qui se recouvrent fortement. Les voisinages obtenus sont plus pertinents (clusters faiblement connectés et de tailles raisonnables), car la plupart des clusters fusionnés possèdent peu de variables propres. De plus, pour ces instances, la largeur de décomposition ainsi que le nombre de clusters varient peu (cf. colonnes 6-8 de la Table 3).

Cette section a montré que fusionner, selon le critère s_{max}, les clusters redondants (contenant peu de variables propres) permet de renforcer la pertinence des décompositions obtenues et d'améliorer les performance de DGVNS sur les instances SPOT5 et sur certaines instances tagSNP. Au contraire, sur les autres instances, l'amélioration n'est pas systématique.

5.3 Apports de la fusion basée sur l'absorption

Dans cette section, nous comparons les performances de DGVNS et DGVNS-*abs* sur les instances RLFAP, SPOT5, GRAPH et tagSNP. Pour toutes nos expérimentations, Ab_{max} a été fixé à 70%. Ainsi, les clusters partageant un peu plus de 2/3 de leurs variables seront fusionnés.

Instances RLFAP. Sur ces instances, DGVNS-*abs* surclasse DGVNS (cf. Table 5). Pour Scen06, les deux méthodes obtiennent 100% de succès ; toutefois, DGVNS-*abs* est plus rapide. Pour Scen07, DGVNS-*abs* améliore le taux de succès de DGVNS de 20%. Pour Scen08, DGVNS-*abs* obtient deux succès de plus que DGVNS et le temps de calcul est divisé par 2.

Instances GRAPH. Sur ces instances, l'apport de la fusion par absorption est très important (cf. Table 5), particulièrement sur les instances Graph11 et Graph13 réputées difficiles, pour lesquelles on observe de fortes améliorations par rapport à DGVNS. Pour Graph11, DGVNS-*abs* améliore le taux de succès de DGVNS de 34% (de 16% à 50%) et réduit la déviation moyenne de l'optimum de (37.4% à 23%). Pour Graph13, DGVNS-*abs* obtient un succès (DGVNS n'atteint jamais l'optimum) et réduit la déviation moyenne de l'optimum de moitié (de 122% à 65%). Pour Graph06, les deux méthodes obtiennent 100% de succès, toutefois, DGVNS-*abs* est plus rapide (gain en temps de calcul de 29% par rapport à DGVNS).

Instances SPOT5. DGVNS-*abs* se montre aussi très efficace sur les instances SPOT5 (cf. Table 6), où il obtient 100% de succès sur quatre des six instances. Sur les instances de grande taille (#412, #414, #507 et #509), DGVNS-*abs* améliore le taux moyen de succès de DGVNS de 25%. Sur les autres instances, les deux méthodes obtiennent les mêmes taux de succès, mais DGVNS-*abs* est plus rapide.

Instances tagSNP. Une fois de plus, les résultats sont en faveur de DGVNS-*abs* (cf. Table 7). Pour les instances de taille moyenne (excepté #6835 et #8956), DGVNS-*abs* obtient de meilleurs temps de calcul comparé à DGVNS. Pour les instances #16421 et #16706, DGVNS-*abs* est 2 à 3 fois plus rapide. Pour les instances de grande taille, DGVNS-*abs* améliore le taux moyen de succès de 10%. Pour les instances de grande taille, c'est encore DGVNS-*abs* qui obtient les meilleures performances.

5.4 Comparaison des deux méthodes de fusion

Cette section compare DGVNS-*abs* et DGVNS-s_{max} pour la valeur de s_{max} ayant obtenu les meilleurs résultats.

Instances RLFAP. Les résultats sont en faveur de DGVNS-*abs* (cf. Table 5). Pour Scen06, DGVNS-*abs* est trois fois plus rapide. Pour Scen07, DGVNS-*abs* obtient 100% de succès contre 22% pour DGVNS-8 et divise le temps de calcul par 4. Pour Scen08, DGVNS-abs obtient 5 succès et une déviation moyenne à l'optimum de 4% contre 19% pour DGVNS-16.

Les bonnes performances de DGVNS-*abs* sur ces instances peuvent s'expliquer par la qualité des décompositions produites par la fusion basée sur l'absorption. En effet, ces dernières présentent des clusters ayant une forte proportion de variables propres et contenant en moyenne 13 variables. Ceci renforce la pertinence du mécanisme de diversification de DGVNS car les clusters sont beaucoup moins redondants. Ce n'est pas le cas pour les décompositions fournies par la fusion basée sur s_{max}, où beaucoup de clusters partageant 100% de leur variables sont encore conservés (cf. Section 3.3). Par ailleurs, en raison du nombre peu élevé de clusters obtenus, différents régions de l'espace de recherche sont couvertes beaucoup plus rapidement, permettant de converger vers la région contenant l'optimum global.

Instances SPOT5. La tendance se confirme sur les instances SPOT5 (cf. Table 6), pour lesquelles DGVNS-*abs* obtient de légères améliorations aussi bien en taux de succès qu'en temps de calcul. Notons, toutefois, que pour l'instance #507, l'amélioration est beaucoup plus significative, où DGVNS-*abs* obtient 98% de succès contre 76% pour DGVNS-12 (gain de 22%). Sur ces instances, les mêmes constats, que ceux dressés sur les instances du RLFAP, peuvent être effectués concernant la pertinence des décompositions produites par la fusion par absorption. De plus, les largeurs des décompositions (w^-) obtenues restent comparables, voire nettement inférieures à celles des décompositions produites par la fusion basée sur s_{max}. Par exemple, pour l'instance #412, w^- passe de 227 (avec $s_{max} = 4$) à 136 (avec l'absorption).

Inst.	Méthode	Succ.	Temps	Moy.	$\|C_T\|$	Taille C_i moy.	max.
#3792	DGVNS	50/50	954	6,359,805	70	30.37	95
S^* =6,359,805	DGVNS-32	50/50	832	6,359,805	49	22.18	121
	DGVNS-abs	**50/50**	**602**	**6,359,805**	**12**	**50.17**	**140**
#4449	DGVNS	50/50	665	5,094,256	56	36.71	101
S^* =5,094,256	DGVNS-32	50/50	587	5,094,256	49	22.18	**121**
	DGVNS-abs	**50/50**	**578**	**5,094,256**	**8**	**64.5**	**164**
#6835	DGVNS	**50/50**	**2,409**	**4,571,108**	**56**	**53.5**	**109**
S^* =4,571,108	DGVNS-32	50/50	3,321	4,571,108	23	39.26	229
	DGVNS-abs	50/50	2,947	4,571,108	4	140.0	262
#8956	DGVNS	**50/50**	**4,911**	**6,660,308**	**54**	**52.03**	**185**
S^* =6,660,308	DGVNS-32	46/50	7,192	6,660,309	30	29.33	215
	DGVNS-abs	50/50	6,555	6,660,308	6	88.17	216
#9319	DGVNS	50/50	788	6,477,229	62	34.41	89
S^* =6,477,229	DGVNS-**32**	**50/50**	**555**	**6,477,229**	**44**	**22.5**	**115**
	DGVNS-abs	50/50	738	6,477,229	10	59.9	181
#15757	DGVNS	50/50	60	2,278,611	45	20.24	67
S^* =2,278,611	DGVNS-**12**	**50/50**	**54**	**2,278,611**	**25**	**18.44**	**101**
	DGVNS-abs	**50/50**	**58**	**2,278,611**	**10**	**40.4**	**92**
#16421	DGVNS	50/50	2,673	3,436,849	35	42.54	113
S^* =3,436,849	DGVNS-32	**50/50**	**1,038**	**3,436,849**	**20**	**32.8**	**133**
	DGVNS-abs	**50/50**	**1,114**	**3,436,849**	**6**	**74**	**134**
#16706	DGVNS	49/50	153	2,632,310	49	19.45	49
S^* =2,632,310	DGVNS-16	50/50	99	2,632,310	37	19	51
	DGVNS-abs	**50/50**	**53**	**2,632,310**	**12**	**41.33**	**94**
#17034	DGVNS	41/50	8,900	38,563,232	139	52.95	189
S^* =38,318,224	DGVNS-32	32/50	11,058	38,838,844	79	27.12	229
	DGVNS-abs	**47/50**	**13,108**	**38,318,226**	**15**	**80.73**	**237**
#14226	DGVNS	46/50	7,606	25,688,751	94	38.19	159
S^* =25,665,437	DGVNS-24	46/50	10,930	25,712,052	58	27.48	179
	DGVNS-abs	**50/50**	**10,417**	**25,665,437**	**15**	**76.27**	**184**

TABLE 7 – Comparaison entre DGVNS, DGVNS-abs et DGVNS-s_{max} sur les instances tagSNP.

Instances GRAPH. Sur ces instances (cf. Table 5), aucune des deux méthodes ne domine clairement l'autre. En effet, DGVNS-*abs* surclasse DGVNS-32 sur Graph11 (50% de succès), alors que sur Graph13, c'est DGVNS-32 qui obtient les meilleurs résultats en termes de taux de succès (gain de 12%) et en qualité des solutions obtenues (réduction de la déviation moyenne de l'optimum de 65.73% à 16.37%). Notons, que sur cette instance difficile, la fusion permet à DGVNS d'atteindre l'optimum pour la première fois, avec un taux de succès de 14%. La contre-performance de DGVNS-*abs* sur Graph13 peut probablement s'expliquer par la taille moyenne des clusters qui est trop importante (134 variables par cluster) et par un fort chevauchement entre clusters (s_{max}=153). Pour comparaison, les clusters issus de la fusion basée sur s_{max} contiennent en moyenne 14 variables et s_{max}=29.

Instances tagSNP. Pour les instances de taille moyenne (cf. Table 7), DGVNS-*abs* obtient de meilleurs résultats sur 5 instances parmi 8 (meilleur taux de succès sur l'instance #8956 et meilleurs temps de calcul sur quatre instances). Pour les trois autres instances (exceptée pour #15757 où les résultats sont très similaires), c'est DGVNS-32 qui obtient les meilleurs temps de calcul. Pour les instances de grande taille (#14226 et #17034), DGVNS-*abs* surclasse nettement DGVNS-32, avec un gain en taux de succès moyen de 12.5%.

Bilan expérimental. Sur l'ensemble des instances, la fusion de clusters procure des gains substantiels comparés aux résultats obtenus par DGVNS sur la décomposition initiale de MCS. Ces résultats confirment l'intérêt de réduire la redondance entre clusters pour identifier des structures de voisinages plus pertinentes et ainsi renforcer l'effort de diversification de DGVNS.

Par ailleurs, la fusion par absorption permet de produire des décompositions avec des clusters de taille moyenne et ayant une forte proportion de variables propres. En effet, pour la plupart des instances (cf. les colonnes 6-8 des différentes figures), la taille des clusters est en moyenne 2 fois plus grande comparée à celle de la décomposition initiale. De plus, le nombre de clusters reste relativement petit. Sur l'instance #17034, le nombre de clusters diminue de 139 à 15, la taille moyenne augmente de 52.95 à 80.73 et w^- passe de 189 à 237.

6 Conclusions et perspectives

Nous avons proposé deux schémas de raffinement de la décomposition arborescente permettant de limiter la redondance entre clusters, dans le but de guider DGVNS vers des structures de voisinages plus pertinentes. Les expérimentations menées sur plusieurs instances difficiles montrent la pertinence et l'intérêt de la fusion par absorption pour renforcer la qualité des voisinages. Comme travail futur, il serait intéressant d'envisager des critères de fusion dynamiques lors de la recherche, selon les valeurs des variables instanciées.

Acknowledgements. Ce travail à été soutenu par l'Agence Nationale de la Recherche, référence projet FiCOLOFO ANR-10-BLA-0214.

Références

[1] E. Amir and S. Mcllraith. Solving satisfiability using decomposition and the most constrained subproblem. *Electronic Notes in Discrete Mathematics*, 9 :329–343, 2001.

[2] S. Arnborg, D. G. Corneil, and A. Proskurowski. Complexity of finding embeddings in a k-tree. *SIAM. J. on Algebraic and Discrete Methods*, 8 :277–284, 1987.

[3] E. Bensana, M. Lemaître, and G. Verfaillie. Earth observation satellite management. *Constraints*, 4(3) :293–299, 1999.

[4] B. Cabon, S. de Givry, L. Lobjois, T. Schiex, and J.P. Warners. Radio link frequency assignment. *Constraints*, 4(1) :79–89, 1999.

[5] S. de Givry, T. Schiex, and G. Verfaillie. Exploiting tree decomposition and soft local consistency in weighted CSP. In *AAAI*, pages 22–27. AAAI Press, 2006.

[6] R. Dechter and Y. El Fattah. Topological parameters for time-space tradeoff. *Artif. Intell.*, 125(1-2) :93–118, 2001.

[7] R. Dechter and J. Pearl. Tree clustering for constraint networks. *Artif. Intell.*, 38(3) :353–366, 1989.

[8] C.S. Carlson et al. Selecting a maximally informative set of single-nucleotide polymorphisms for association analyses using linkage disequilibrium. *Am. J. of Hum. Genetics*, 74(1) :106–120, 2004.

[9] M. Fontaine, S. Loudni, and P. Boizumault. Guiding VNS with tree decomposition. In *ICTAI*, pages 505–512, 2011.

[10] M. Fontaine, S. Loudni, and P. Boizumault. Exploiting tree decomposition for guiding neighborhoods exploration for VNS. *RAIRO Operations Research*, 47(2) :91–123, 2013.

[11] S. Gopalakrishnan and Z. S. Qin. Tagsnp selection based on pairwise ld criteria and power analysis in association studies. In *Pacific Symposium on Biocomputing*, pages 511–522, 2006.

[12] G. Gottlob, S. T. Lee, and G. Valiant. Size and treewidth bounds for conjunctive queries. In Jan Paredaens and Jianwen Su, editors, *PODS*, pages 45–54. ACM, 2009.

[13] G. Gottlob, R. Pichler, and F. Wei. Tractable database design through bounded treewidth. In Stijn Vansummeren, editor, *PODS*, pages 124–133. ACM, 2006.

[14] D. Habet, L. Paris, and C. Terrioux. A tree decomposition based approach to solve structured SAT instances. In *ICTAI*, pages 115–122, 2009.

[15] P. Hansen, N. Mladenovic, and D. Perez-Brito. Variable neighborhood decomposition search. *Journal of Heuristics*, 7(4) :335–350, 2001.

[16] W. D. Harvey and M. L. Ginsberg. Limited discrepancy search. In *IJCAI (1)*, pages 607–615, 1995.

[17] P. Jégou, S. Ndiaye, and C. Terrioux. Computing and exploiting tree-decompositions for solving constraint networks. In *CP*, pages 777–781, 2005.

[18] Ph. Jégou, S. Ndiaye, and C. Terrioux. Strategies and Heuristics for Exploiting Tree-decompositions of Constraint Networks. In *WIGSK'06*, pages 13–18, 2006.

[19] M. Kitching and F. Bacchus. Exploiting decomposition on constraint problems with high tree-width. In *IJCAI*, pages 525–531, 2009.

[20] S. Loudni and P. Boizumault. Combining VNS with constraint programming for solving anytime optimization problems. *EJOR*, 191(3) :705–735, 2008.

[21] R. Marinescu and R. Dechter. AND/OR branch-and-bound search for combinatorial optimization in graphical models. *Artif. Intell.*, 173(16-17) :1457–1491, 2009.

[22] B. Neveu, G. Trombettoni, and F. Glover. ID Walk : a candidate list strategy with a simple diversification device. In *CP*, pages 423–437, 2004.

[23] I. Rish and R. Dechter. Resolution versus search : Two strategies for SAT. *J. Autom. Reasoning*, 24(1/2) :225–275, 2000.

[24] N. Robertson and P.D. Seymour. Graph minors. ii. algorithmic aspects of tree-width. *J. Algorithms*, 7(3) :309–322, 1986.

[25] M. Sánchez, D. Allouche, S. de Givry, and T. Schiex. Russian doll search with tree decomposition. In *IJCAI*, pages 603–608, 2009.

[26] R. E. Tarjan and M. Yannakakis. Simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs. *SIAM J. Comput.*, 13(3) :566–579, 1984.

[27] C. Terrioux and P. Jégou. Hybrid backtracking bounded by tree-decomposition of constraint networks. *Artificial Intelligence*, 146(1) :43–75, 2003.

[28] H. van Benthem. GRAPH : Generating radiolink frequency assignment problems heuristically, 1995.