Dynamic Branch & Bound Distribué

Imade Benelallam, Mustapha Belaissaoui, Redouane Ezzahir, El Houssine Bouyakhf

Dynamic Branch & Bound Distribué

‡I. Benelallam ⋆M. Belaissaoui ‡R. Ezzahir ‡E. H. Bouyakhf

‡LIMIARF, FS Rabat, Université Mohammed V-agdal, Maroc
⋆ENCG, Settat, Université Hassan I, Maroc

imade.benelallam@ieee.org, m.belaissaoui@encg-settat.ma, ezzahir@lirmm.fr, bouyakhf@fsr.ac.ma

Résumé

Les méthodes exploitant le principe d'explication [2] (**nogoods**) pour la résolution des réseaux de contraintes distribués semblent constituer la meilleure approche en termes de complexité théorique en temps. Néanmoins, rares sont les approches d'optimisation qui tirent le meilleur parti de la puissance des **nogoods valués** [16]. Dans cet article, nous proposons un nouvel algorithme de recherche, appliqué aux problèmes d'optimisation de contraintes distribués (DCOPs), appelé Dynamic Branch & Bound Distribué (DisDB&B). Les agents assignent leurs variables d'une manière séquentielle et transmettent en avant leurs nogoods valués. La combinaison des bornes inférieures induites à partir des nogoods valués contribue assurément à l'accélération de la recherche et à l'élimination des sous problèmes irréalisables. Dans cet article, nous montrons que notre algorithme est complet. L'analyse des résultats expérimentaux montrent l'intérêt de notre approche.

Abstract

Distributed constraint satisfaction problems based on **nogoods** reasoning [2] seem to be the best approach in terms of theoretical time complexity. However, only few optimization protocols get the most out of **valued nogoods**'s power [16]. In this paper, we present a new search algorithm for distributed constraint optimization problems (DCOPs), called Distributed Dynamic Branch & Bound (DisDB&B). Agents assigns their variables sequentially and forwards valued nogoods (lower bounds) synchronously to the next unassigned agent. Combining lower bounds on inferred valued nogoods can help to speed up the search, and prune infeasible sub-problems. We show that the algorithm proposed here (DisDB&B), is optimal and guaranteed to terminate, having polynomial space complexity. Detailed experimental results show that on benchmark problems, the proposed algorithm achieve over an order of magnitude reduction on terms of messages exchanged between agents.

1 Introduction

Le sujet des systèmes distribués, s'il n'est pas récent, est actuellement un axe de recherche très prometteur. Il semble être une approche rassurante pour la résolution ou l'optimisation des problèmes naturellement distribués [14]. Cette discipline s'intéresse aux comportements collectifs produits par les interactions de plusieurs entités flexibles et autonomes, appelées agents. L'évolution de ces derniers dans un environnement commun et partagé introduit généralement des contraintes entre leurs actions possibles. La satisfaction ou l'optimisation de ces contraintes nécessite la mise en place d'un protocole de communication assurant la convergence vers une solution correcte et optimale.

Le concept des problèmes de satisfaction de contraintes distribués (DisCSP), a été fondé à travers la proposition d'un ensemble d'algorithmes de résolution complets [19], *Asynchronous Backtracking* ABT et *Asynchronous weak-Commitment Search* WCS [18]. Ensuite, de nouvelles approches d'optimisation de contraintes distribués (DCOP) ont vu le jour, soit par une simple extension d'ABT [11], ou par la proposition des nouveaux algorithmes [12] d'optimisation à recherche incomplète. L'algorithme Synchronous Branch & Bound SynB&B [10], est le premier algorithme d'optimisation qui a vu le jour, mais avec une grande complexité théorique en temps. Pour remédier à ce problème, un algorithme asynchrone [20] a été proposé, mais sans aucune garantie de solution. *Asynchronous distributed constraint optimisation* Adopt [13] est l'un des algorithmes qui ont montré de très bonnes performances. Récemment, une nouvelle approche de recherche arborescente appelée *Asynchronous Forward-Bounding with Backjumping* AFB-BJ [7, 8] a été mise en place. Malgré

que cet algorithme présente encore certains points défaillants relativement remédier par *Asynchronous Breadth-First Search* AFBS [6], AFB-BJ a pu largement dépasser les performances aboutis par Adopt. A l'état actuel de nos connaissances, Adopt-ng [17] est le seul algorithme qui a tiré profit du concept des nogoods valués. Aucun autre algorithme n'a été orienté vers une utilisation optimale de ce concept. Les systèmes d'explications à base des nogoods valués semblent constituer l'approche la plus efficace pour la résolution des problèmes DCOPs. L'objectif du travail présent est de mettre en place une nouvelle méthode de résolution inspirée des approches centralisées introduites dans [5, 9].

Cet article est organisé comme suit. Dans la section 2, nous rappelons les notions de base propres aux problèmes d'optimisation des contraintes distribués (DCOPs) et au concept des nogoods valués. La section 3 présente notre algorithme DisDB&B alors que la section 4 discute les résultats expérimentaux. Enfin, la dernière section conclut ce travail tout en donnant des pistes pour des travaux futurs.

2 Rappels et contexte de l'étude

2.1 Problèmes d'optimisation de contraintes distribués

Formellement, un problème de satisfaction de contraintes (CSP) est défini par la donnée d'un triplet (X, D, C). X est un ensemble $\{X_1, ..., X_n\}$ de n variables. Chaque variable X_i prend ses valeurs dans un domaine fini donné dans $D = \{d_1, ..., d_n\}$. Les variables sont soumises à des contraintes définies dans $C = \{C_{ij}/1 \leqslant i, j \leqslant n\}$ (pour des raisons de simplicité nous nous limiterons ici aux contraintes binaires). Étant donnée une instance (X, D, C), le problème CSP consiste à déterminer l'existence d'une affectation globale qui satisfait toutes les contraintes.
Contrairement au CSP où les contraintes C_{ij} retourne un état logique (consistance ou inconsistance), dans les Problèmes d'Optimisation de Contraintes (COP), les contraintes C_{ij} sont modélisées par une fonction de coût retournant une valeur dans $\mathbb{R}^+$:

$$C_{ij} : d_i \times d_j \longrightarrow \mathbb{R}^+ \qquad (1)$$

Le problème COP consiste à trouver une affectation globale GA minimisant ou maximisant une fonction objectif φ défini comme suit :

$$\varphi(GA) = \sum_{\forall C_{ij} \in C} C_{ij}(GA(X_i), GA(X_j)) \qquad (2)$$

Un COP distribué (DCOP) est un COP dont les variables et les contraintes sont distribuées sur un ensemble des agents. Formellement, un réseau d'optimisation de contraintes distribué est un 6-uplet $(X, D, C, A, \varphi, \psi)$, où X, D, C et φ sont définis comme précédemment. $A = \{A_1, ..., A_k\}$ est un ensemble de k agents, et ψ une fonction de distribution faisant associe chaque variable X_i à chaque agent A_i, $\psi : X \longrightarrow A$. Dans cet article, et sans manque de généralité, nous considérerons uniquement le cas des contraintes binaires (i.e. contraintes impliquant deux variables/agents), et une fonction ψ bijective (i.e. une variables par agent). Ainsi, la structure du DCOP peut être représentée par le graphe (A, C), appelé graphe de contraintes (figure 1). Les sommets de ce graphe sont les agents de A et une arête connecte deux sommets si les variables/agents associés partagent une même contrainte.

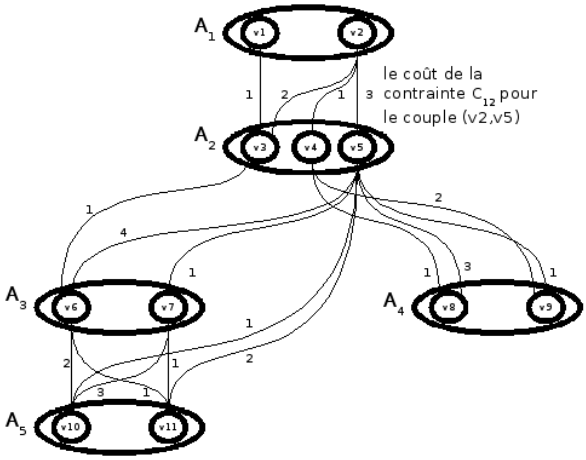

FIG. 1 – Graphe de contraintes dans le cadre de DCOP

2.2 les nogoods valués

La connaissance d'une affectation partielle responsable d'une incohérence globale (dépassement de la borne supérieure B) peut être réutilisée pour déduire une éventuelle incohérence globale de certaine extensions. Cette connaissance pertinente est appelée un **nogood** [15]. Dans notre problème nous nous intéressons au **nogood valué** [4] qui permet de mémoriser une information valuée déduite à partir d'une tentative infructueuse d'extension d'une affectation partielle.

Définition 1 (nogood valué) *Un nogood valué est un triplet (AP, ϑ, C), avec $AP = \{X_1^{v_1}, ..., X_k^{v_k}\}$[1] une affectation partielle, ϑ une valuation telle que toute extension AP' de AP ne peut jamais avoir une valuation strictement inférieure à ϑ. C est l'ensemble de contraintes violées par AP, appelée justification de ϑ.*

[1] $X_k^{v_k}$ est une affectation : $X_k = v_k$

Exemple 1 : dans la figure 1, le nogood valué $(X_1^{v_2}, 1, C_{12})$ est valide car toute affectation qui possède $X_1^{v_2}$ a une valuation supérieure ou égale à 1.

Dans ce qui suit nous allons essayer de rappeler les différentes propriétés de manipulation des nogoods valués, introduites dans [4] et récemment reprise dans Adopt-ng [17]. Les deux propriétés suivantes donnent les principales méthodes de production d'un nogood valué.

Propriété 1 (Construction) *[4] Soit AP une affectation partielle, et C un ensemble de contraintes violées par AP, $(AP, \varphi(AP), C)$ est un nogood valué, tel que $\varphi(AP)$ est la valuation de AP, $\varphi(AP) = \sum_{\forall c \in C}(c)$.*

Exemple 2 : dans la figure 1, le nogood valué $(\{X_1^{v_2}, X_2^{v_5}\}, 3, C_{12})$ est construit à partir de la propriété de construction.

En réalité la construction des nogoods valués par cette méthode ne présente aucun profit pour le mécanisme d'apprentissage, car il serait vraiment non judicieux de mémoriser une information déjà présente dans un contexte courant. Les seules nogoods valués qui portent une information future sont les nogoods issus d'une opération d'union (i.e. min-résolution Propriété 2). La production de ces nogoods est effectuée généralement lors d'un backtrack, interdisant provisoirement la valeur courante d'une variable/agent instanciée. Pour cela nous présentons un nogood valué comme étant une implication :

$$(\{X_1^{v_1}, ..., X_{k-1}^{v_{k-1}}\}, \vartheta, C) \longrightarrow X_k^{\neq v_k} / \vartheta \geqslant B \quad (3)$$

Ce qui est logiquement équivalent à l'expression :

$$(\{X_1^{v_1}, ..., X_k^{v_k}\}, \vartheta, C) / \vartheta \geqslant B$$

X_k est la variable conclusion. Son choix est souvent déterminer en se basant sur la méthode de Ginsberg [9].

Au cours des explorations, les extensions échouées d'un contexte courant nous permettent de déduire de nouveaux nogoods à travers la propriété suivante :

Propriété 2 (Min-résolution) *[4] Soit AP une affectation partielle et X_j une variable/agent non encore instanciée. Soit AP_1,...,AP_n, l'ensemble des extensions possibles de AP avec toutes les valeurs possibles v_1,...,v_n de X_j. Si (AP_1, ϑ_1, C_1),..., (AP_n, ϑ_n, C_n) sont des nogoods valués, alors, on produit le nogood suivant $(AP, \min_{i \in D_j}(\vartheta_i), \bigcup_{i \in D_j} C_i)$.*

Exemple 3 : d'après la propriété de construction nous pouvons produire les deux nogoods valués suivants : $(\{X_1^{v_1}, X_2^{v_3}\}, 2, C_{12})$ et $(\{X_1^{v_1}, X_2^{v_4}\}, 1, C_{12})$, une union de ces derniers donne $(X_1^{v_2}, 1, C_{12})$.

Cette opération n'est effectuée que si toutes les extensions de l'affectation partielle s'avèrent incohérentes. La mise en place de ce formalisme pour les nogoods valués, répond largement au besoin de mémorisation d'informations construites durant l'exploration d'un réseau de contraintes. Les nogoods valués ainsi construits doivent choisir entre une petite justification, une valuation élevée et un nombre minimal de variables. L'efficacité de leur utilisation nécessite cependant quelques raffinements supplémentaires. C'est dans cette optique que nous rappelons les propriétés suivantes :

Propriété 3 (Projection par contraintes) *[4] Soit (AP, ϑ, C) un nogood valué et C' un sous-ensemble des contraintes de C que les extensions de AP ne peuvent pas violer. $(AP, \vartheta, C \backslash_{C'})$ est un nogood valué.*

Exemple 4 : $(\{X_1^{v_1}, X_2^{v_3}, X_3^{v_6}, X_4^{v_8}\}, 2, C_{12}, C_{23}, C_{24})$ est un nogood valué dont C_{24} vérifie la propriété précédente, d'où la projection suivante : $(\{X_1^{v_1}, X_2^{v_3}, X_3^{v_6}, X_4^{v_8}\}, 2, C_{12}, C_{23})$.

Cette opération est très utile pour enlever les justifications inutiles. En principe, dans un nogood valué nous distinguons entre deux types de justifications, celle des contraintes totalement instanciées (i.e., C_{12} dans l'exemple 2), que nous appelons justification passées (généralement inutile à mémorisée) et celle des contraintes partiellement ou non instanciées que nous appelons justifications futures (i.e. C_{12} dans l'exemple 1), et qui est potentiellement pertinente.

Propriété 4 (Projection par affectation) *[4] Soit (AP, ϑ, C) un nogood valué, et AP' l'ensemble des affectations de AP absentes dans le n-uplets interdits par les contraintes de C .Donc $(AP \backslash_{AP'}, \vartheta, C)$ est un nogood.*

Cette propriété est une conséquence directe de la propriété de projection par contrainte.

Exemple 5 : Le nogood valué de l'example précédant $(\{X_1^{v_1}, X_2^{v_3}, X_3^{v_6}, X_4^{v_8}\}, 2, C_{12}, C_{23})$ présente une affectation $X_4^{v_8}$ non interdite par la justification. L'utilisation de la projection par affectation permet de nous donner le nogood valué suivant : $(\{X_1^{v_1}, X_2^{v_3}, X_3^{v_6}\}, 2, C_{12}, C_{23})$.

Ces deux opérations vont nous permettre d'enlever les informations superflues et de rendre les nogoods

de plus en plus pertinents et légers dans la phase de traitement. Vu que certains coûts des contraintes sont inutiles à sauvegarder, un recours à une opération au prix d'une réduction de la valuation du nogood est inévitable.

Propriété 5 (Réduction) [2] *[4] Soit le nogood valué* $N = (AP, \vartheta, C)$*, avec* $c \in C$ *tel que* $C' = C \backslash_c$*, et* $\vartheta' = \vartheta \ominus \vartheta_c$*. Alors* $(AP \downarrow_{C'}, \vartheta', C')$[3]*s est un nogood valué noté* $\Downarrow_c (N)$*.*

On note $\Downarrow_C (N)$ l'ensemble des réductions successives de N par les contraintes appartenant à C.

> ***Exemple 6 :*** Le nogood valué suivant $(\{X_1^{v_2}, X_2^{v_5}\}, 4, C_{12}, C_{23})$ peut être réduit de la manière suivante $(\{X_1^{v_2}, X_2^{v_5}\}, 1, C_{23})$, puis raffiné par projection $(\{X_2^{v_5}\}, 1, C_{23})$

La réduction d'un nogood valué permet d'enlever les informations calculables sachant que nous pouvons bien les restituer à partir du contexte courant. D'où l'opération d'augmentation des nogoods valués.

Propriété 6 (Augmentation) *[4] Soient le nogood valué suivant* $N = (AP, \vartheta, C)$ *et c une contrainte violée par AP, tel que* $c \nsubseteq C$*. Alors* $(AP, \vartheta \oplus \vartheta c, C \cup c)$ *est une augmentation de N par c, noté* $\Uparrow^c (N)$*.*

On note $\Uparrow^{AP} (N)$ l'ensemble des augmentations successives de N par les contraintes violées par AP et non figurant dans la justification de N.

> ***Exemple 7 :*** On peut à tout moment revenir au nogood d'origine par l'augmentation suivante $\Uparrow^{C_{12}} (\{X_2^{v_5}\}, 1, C_{23}) = (\{X_1^{v_2}, X_2^{v_5}\}, 4, C_{12}, C_{23})$.

Il apparaît tout d'abord que grâce à cette augmentation, un nogood valué réduit donnera toujours une estimation des valuations globales au moins aussi intéressantes que celles d'origines. Ce qui va nous permettre de prévoir un retour arrière aussi tôt que possible. Les nogoods enregistrés de pars et d'autres par les agents, présentent des expériences différentes, et leurs agrégation donnera un avantage majeur pour l'augmentation de la borne inférieure, d'où la propriété de cumul :

Propriété 7 (Cumul) *[4] Soient* $N_1 = (AP_1, \vartheta_1, C_1)$ *et* $N_2 = (AP_2, \vartheta_2, C_2)$ *deux nogoods valués compatibles ente eux avec* $C_1 \cap C_2 = \emptyset$*, alors* $(AP_1 \cup AP_2, \vartheta_1 \oplus \vartheta_2, C_1 \cup C_2)$ *est un nogood valué.*

[2] $\ominus$ (resp. $\oplus$) est un opérateur idempotent de soustraction (resp. d'agrégation) défini dans [4]

[3] On note $AP \downarrow_C$ la projection de l'affectation partielle AP sur l'ensemble des variables figurant dans C.

Deux affectations partielles AP_1 et AP_2 sont ditent compatibles ssi les variables communes sont affectées aux même valeurs.

> ***Exemple 8 :*** Soient les deux nogoods suivants $(\{X_1^{v_2}\}, 1, C_{12})$ et $(\{X_2^{v_5}\}, 2, C_{23}, C_{24})$, l'application de cette propriété de cumul donnera le nogood valué suivant $(\{X_1^{v_2}, X_2^{v_5}\}, 3, C_{12}, C_{23}, C_{24})$.

Dans ce qui suit et pour des raisons de simplicité, toutes les opérations de projection seront effectuées implicitements.

3 Dynamic Branch & Bound Distribué

Dans cette section nous présentons un nouvel algorithme pour la résolution des problèmes DCOPs, appelé Dynamic Branch & Bound Distribué DisDB&B. Cet algorithme combine deux avantages, premièrement, l'affectation d'une valeur à une varialble/agent n'est effectuée que ssi elle est compatible avec toutes les variables passées, deuxiement, tout agent arrivant à trouver une solution locale (une affectation pour sa variable) transmet en avant le nogood valué correspondant. Le cumul de ces nogood valués transmisent d'un agent vers un autre, permet de construire une borne inférieure de valuation très élevée, permetant ainsi de déclencher plus tôt une opération de retour arrière.

Dans le cadres des systèmes Multi-Agents plusieurs approches de résolutions peuvent être envisagées. Cependant, comme nous cherchons à construire un algorithme tout à fait distribué pouvant s'adapter à l'exécution et comme il n'est pas toujours possible d'avoir une vision globale du système au niveau de chaque agent, nous allons nous focaliser sur une approche décentralisée, qui consiste à mettre à jour localement la politique de chaque agent, et ceci, en fonction des informations reçus sur l'état actuel des agents passés, que nous appelons le contexte courant. Nous montrons que la méthode la plus efficace pour assurer la complétude est d'adapter une gestion de stockage séparée au niveau de chaque agent. Pour se faire nous définissons les structures de données suivantes :

- $N_{X_i}^{v_i}$: Le nogood valué associé à l'affectation $X_i^{v_i}$.
- N_{X_i} : L'ensemble des nogoods valués relatifs à toutes les valeurs de la variable X_i, il constitue la base de connaissance de chaque agent A_i. Cette liste est mise à jour dynamiquement en fonction du contexte courant du système.
- $CCTX$: Le contexte courant (the Current ConTeXt), il contient trois structures de données. Une liste des affectations $PA = \{X_1^{v_1}, ..., X_k^{v_k}\}$

des variables/agents passées, une liste des nogoods valués $\{N_{X_1}^{v_1}, ..., N_{X_k}^{v_k}\}$ associées aux agents participant dans le PA, et finalement un nogood valué N_{CCTX} construit à patir de l'affectation PA.

Contrairement à l'algorithme Adopt-ng [16] où les nogoods valués sont initialisés (re-initialisés) en utilisant les contraintes unuaires, l'algorithme DisDB&B initialise (re-initialisés) sa liste des nogoods valués N_{X_i}, par toutes les contraintes connues à l'agent A_i (voir figure 2). Cette connaissance jouera un rôle très important pour l'ordonnancement des valeurs, et ce, en utilisant l'heuristique Min-Conflicts. Formellement, un nogoods valué est calculé par l'équation suivante :

$$N_{X_i}^v = (X_i^v, h(v), C_v) \tag{4}$$

avec $h(v) = \sum_{X_j \prec X_i} mincost(v,u) \setminus u \in D_j$
et $C_v = \bigcup_{X_j \prec X_i} C_{ij} \setminus C_{ij}(X_i^v, X_j^u) \neq 0$
Les X_j présentent les variables correspondant aux agents les moins prioritaires dans l'ordre.

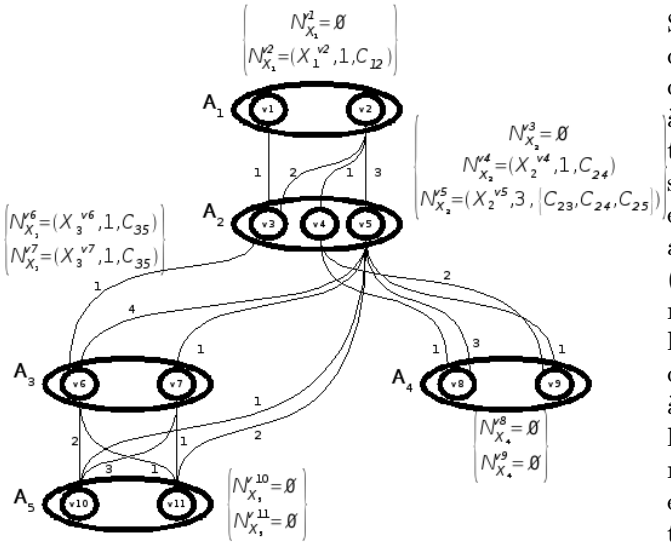

FIG. 2 – Initialisation des nogoods valués.

3.1 Description de l'algorithme DisDB&B

Le protocole de communication de l'algorithme DisDB&B est conçu principalement à base de deux types de messages. Le message **CCTX_MSG** assurant l'avancement de la solution partielle de l'algorithme en transmettant le jeton $CCTX$, et le message **BACKTRACK_MSG** déclenchant le retour arrière (backtrack). Le message **SOLUTION** (resp. **TERMINATE**) permet de diffuser une solution (resp. d'arrêter l'execution). Dans l'algorithme DisDB&B un seul agent est instancié à la fois, lorsqu'une solution locale est trouvée, l'agent courant transmet en avant sa solution et le nogoods valué correspondant ($CCTX$). Pour tester la cohérence de ses valeurs, l'agent recevant le $CCTX$ exploite le nogood valué le plus élevé pour augmenter la valuation de sa borne inférieure. Si aucune valeur n'est trouvée, une opération de min-résolution permettra de générer un nogood valué, qui sera transmis via le message **BACKTRACK_MSG** vers l'agent coupable le plus proche. La reception du message **BACKTRACK_MSG** consiste à une opération d'apprentissage interdisant la solution locale par une valuation plus élevée (opération de réduction partielle) que celle du passé. Puis, une autre invistigation est poursuite afin de trouver une solution locale cohérente ou de déclencer un autre backtrack. L'algorithme DisDB&B est présenté dans *Algorithm 1.* Initialement les agents sont ordonnés totalement. La première étape consiste au lancement de la procédure **Initialize**() (lignes 1-7). Durant cette étape les agents A_i initialisent leurs nogoods valués $N_{X_i} = \{N_{X_i}^{v_1}, ..., N_{X_i}^{v_k}\}$ (équation 4). Seul l'agent le plus prioritaire a le droit de créer le contexte courant $CCTX$ (ligne 6) et d'appeler la procédure **Assign_CCTX**() (lignes 7). Cette procédure à pour tâche de chercher une valeur dont l'estimation du coût est inférieure (lignes 19-25) à la borne supérieur B (initialement infinie). Une fois une valeurs est admise (ligne 22), le contexte courant $CCTX$ est affectée par les informations relatives à cette valeur (X_i^v, $N_{X_i}^v$ et N_{CCTX} qui est mis à jour implicitement). En effet, cette estimation est calculée à travers la fonction **DisDBnB_estimate**() (ligne 43-45), qui consiste à augmenter les nogoods valués correspondant à chaque instanciation des agents passés (ligne 48) par les contraintes du contexte courant N_{CCTX}, si l'estimation est inférieure à B. Alors la valeur est choisie et le message $CCTX\_MSG$ est envoyé à l'agent futur suivant l'ordre. Lorsqu'un agent non encore instancié reçoit le message $CCTX\_MSG$, la fonction **Update_MyNogoods**() est appelée pour mettre à jour les nogoods valués (ligne 9) et réordonner le domaine (ligne 50), puis la procédure **Assign_CCTX**() est lancée à nouveau. Si cet agent est le dernier dans l'ordre et qu'une instanciation cohérente est trouvée (lignes 29-32), alors une solution est atteinte. Cependant, tous les agents sont informés (ligne 30) et la recherche et reprise (ligne 32). Sinon (lignes 33-35), un agent non encore instancié est choisi pour lui transmettre la structure $CCTX$.
L'ensemble des tentatives échouées par l'agent A_i au

cours de la résolution du problème, déclenche un backtrack dynamique (ligne 27). L'appel de la procédure **Backtrack**() permet d'inférer un nouveau nogood valué Ng à travers une opération de min-résolution (ligne 37). Le choix du point de backtrack est déterminé suivant le principe de Ginsberg [9]. Si le nogood généré contient une affectation vide (pour des raisons de simplification, les opérations de projection des nogoods valués sont ignorés) le processus s'achève (lignes 38-39). Lors de la réception du message $BACKTRACK\_MSG$ envoyé par la procédure **Backtrack**() (ligne 42), l'agent récepteur mis à jour le contexte $CCTX$ à travers **Remove_MySelf**() et **Update_CCTX**() (lignes 12 et 14). Les nogoods relatifs à l'agent sont aussi révisés (ligne 13). Pour assurer un apprentissage des nogoods valués une opération de réduction partielle [5] est effectuée (ligne 15). Cette opération consiste à mettre à jour le nogood relatif à la valeur rejetée. Et ceci, par une valuation plus élevée sans avoir écraser les informations déjà présentes pour cette valeur. Ensuite, **Assign_CCTX**() continue la recherche jusqu'à terminaison (ligne 39).

3.2 La complétude de l'algorithme DisDB&B

Tout d'abord nous rappelons l'opération de réduction partielle comme elle a était introduite par Dago dans [5] :

Définition 2 (Réduction partielle) *Soit AP une affectation partielle, $N_{X_i}^{v\_crt}$ un nogood portant sur $X_i^{v\_crt}$ et N un nogood généralement issu d'une opération d'union (min-resolution). Soit $C_{(AP \cap N_{X_i}^{v\_crt})}$ l'ensemble des contraintes violées par AP et présentes dans la justification de $N_{X_i}^{v\_crt}$, et $C_{(AP - N_{X_i}^{v\_crt})}$ les autres contraintes violées par AP. Nous appelons réduction partielle du nogood N par AP et $N_{X_i}^{v\_crt}$ l'augmenation avec l'ensemble des contraintes présentes dans $C_{(AP \cap N_{X_i}^{v\_crt})}$ suivie par la réduction avec l'ensemble des contraintes présentes dans $C_{(AP - N_{X_i}^{v\_crt})}$.*

Nous notons cette opération de réduction par $\Downarrow_{(AP, N_{X_i}^{v\_crt})}(N)$. Cette opération a pour effet de produire à partir de N un nouveau nogood $N_{X_i}^{v\_crt}$ dont la valuation est plus élevée que la valuation du nogood auparavant enregistrée, ce qui va nous permettre de construire une valeur bornée strictement croissante, assurant la convergence vers une solution optimale.

Théorème 1 *: L'algorithme Dynamic Branch & Bound Distribué est complet et optimale.*

La complétude totale : Cette opération consiste à démontrer que si notre algorithme se termine, une solution optimale est atteinte. Nous remarquons qu'un backtrack n'est déclenché que si l'augmentation des nogoods valués associés à l'agent courant ont une valuation supérieure ou égale à B (la borne supérieure, lignes 21-22) . En cas d'arrêt, on disposera donc d'un nogood d'affectation vide (car aucun agent coupable n'a pu être choisi) et de valuation supérieure ou égale à B. Ce nogood est la preuve que toute solution possède une valuation supérieure ou égale à B, donc en cas d'arrêt B sera la valuation de la solution optimale du problème.

La complétude partielle : La question ici réside dans la terminaison de cet algorithme. Pour assurer la terminaison nous appliquons la méthode du backtrack dynamique [9] avec l'introduction de la notion de valuation. La preuve de la terminaison s'appuie intuitivement sur le fait que les interdictions vont progressivement rejeter tout l'arbre de recherche en augmentant, non plus vers la racine, mais vers les agents les plus prioritaires selon l'ordre enregistré. Tôt ou tard, mais assurément, on en viendra à interdire l'agent le plus prioritaire avec une justification vide. Ceci est assuré par la construction d'une *valeur bornée strictement croissante* liée à la progression de notre algorithme [3]. La seule opération de stockage d'un nogoods valué (ligne 17) consiste à une réduction partielle [3]. Sachant que le coût d'une contrainte est fini, nous pouvons affirmer que notre algorithme se termine dans un temps fini.

4 Expérimentations

Deux problèmes classiques sont souvent utilisés pour l'évaluation des problèmes d'optimisation de contraintes distribués (DCOPs), les problèmes Max-DisCSPs aléatoires et les problèmes de coloriage de graphe. Pour évaluer notre algorithme (DisDB&B), nous avons utilisé la plate-forme Dischoco [1] dans laquelle les agents sont simulés par des processus multi-tâches, communicant à travers des envois de messages. Lors des expériences, chaque mesure est prise pour une moyenne de 100 problèmes générés aléatoirement. Le générateur des problèmes Max-DisCSPs aléatoires est caractérisé par quatre paramètres principaux ($\#n$, $\#d$, $p1$, $p2$), où $\#n$ est le nombre d'agents/variables d'une instance, $\#d$ la taille initiale des domaines, $p1$ la densité de contraintes et $p2$ la dureté des contraintes (tightness : la proportion de paires interdites dans une contraintes).

La figure 3 présente le nombre moyen des messages échangés entre les agents en fonction du nombre

```
1  Procedure initialize()
2  B ← ∞
3  foreach v ∈ D_i do
4      N^v_{A_i} ← (X^v_i, h(v), C_v)
5  if A_i is the initializer then
6      create_CCTX()
7      Assign_CCTX()

8  when received (CCTX_MSG, CCTX) do
9      Update_MyNogoods()
10     Assign_CCTX()

11 when received (BACKTRACK_MSG, CCTX, Ng) do
12     CCTX ← Remove_MySelf(CCTX)
13     Update_MyNogoods()
14     Update_CCTX()
15     N^{crt_v}_{X_i} ←⇓_{(PA, N^{crt_v}_{X_i})} (Ng)
16     Assign_CCTX()

17 Procedure Assign_CCTX()
18 v ← empty
19 while D_i has not fully explored AND v is empty do
20     v ← Choose_MyValue()
21     tempN^v_{X_i} ← DisDBnB_estimate(CCTX, v)
22     if Valuation(tempN^v_{X_i}) ≺ B then
23         CCTX ← CCTX ∪ ≺ X^v_i, N^v_{X_i} ≻
24     else
25         v ← empty
26 if v is empty then
27     Backtrack()
28 else
29     if CCTX is a full assignment then
30         Broadcast(SOLUTION, CCTX)
31         B ← Valuation(N_CCTX)
32         Assign_CCTX()
33     else
34         A_k ← Choose_NextAgent()
35         Send(CCTX_MSG, CCTX) to A_k

36 Procedure Backtrack()
37 Ng ← min_{u in D_i}_resolution(DisDBnB_estimate(CCTX, u))
38 if Ng is empty then
39     Broadcast(TERMINATE)
40 else
41     choose X_j from Ng such as ∀X_k ∈ N, X_k ≻ X_j
42     Send(BACKTRACK_MSG, CCTX, Ng) to A_j

43 Function DisDBnB_estimate(CCTX, v)
44 tempCCTX ← CCTX ∪ ≺ X^v_i, N^v_{X_i} ≻
   return maximize_{N^u_{X_k} in tempCCTX} (⇑^{N_CCTX} (N^u_{X_k}))
45

46 Function Update_MyNogoods()
47 foreach v ∈ D_i do
48     if N^v_{X_i} is not compatible with PA then
49         N^v_{X_i} ← (X^v_i, h(v), C_v)
50 Reorder domain according to Min-Conflicts

51 Function Update_CCTX()
52 foreach N^v_{X_i} ∈ CCTX do
53     if N^v_{X_i} contains X_i then
54         N^v_{X_i} ← ∅

55 when received (SOLUTION, CCTX) do
56     GA ← PA
57     B ← Valuation(N_CTTX)

58 when received (TERMINATE) do
59     Terminate execution
```

Algorithm 1: L'algorithme DisDB&B

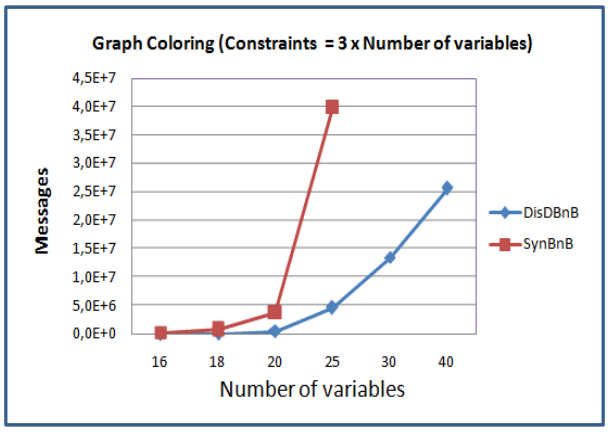

FIG. 3 – : Nombre moyen de messages envoyés par DisDB&B et SynB&B pour la résolution des problèmes de coloriage (3 couleurs) de graphe distribués.

d'agents dans les problèmes de coloriage de graphe (à trois couleur). Nous remarquons que les performances de l'algorithme DisDB&B augmente progressivement par rapport à l'algorithme SynB&B. cette remarque est justifiée par le fait que notre algorithme détecte plus rapidement l'événement du backtrack.

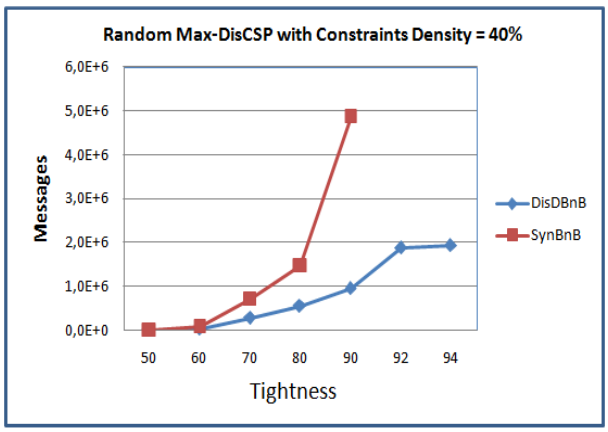

FIG. 4 – Nombre moyen de messages envoyés par DisDB&B et SynB&B pour la résolution des problèmes Max-DisCSP aléatoires $p1 = 0.4$ pour 10 agents.

Dans la figure 4, nous illustrons la performance des deux algorithmes DisDB&B et SynB&B. Pour les problèmes Max-DisCSPs aléatoires avec $p1 = 0.4$ et une dureté variable, nous constatons que dans le cas des problèmes avec des contraintes à dureté faible les deux problèmes se comportent de manière quasi-semblables, mais avec l'augmentation de cette dernière (tightness), nous remarquons une grande baisse en termes de messages échangés dans l'algorithme DisDB&B par rap-

port à l'algorithme SynB&B.

5 conclusion

Dans cet article, nous proposons un nouvel algorithme d'optimisation de contraintes distribués, appelé DisDB&B pour Distributed Dynamic Branch & Bound. nous montrons que notre algorithme est complet et qu'il nécessite seulement une complexité spéciale polynomiale ($O(nd)$ avec n le nombre d'agents et d la taille maximale des domaines) pour le stockage des nogoods valués. La combinaison des bornes inférieures induites à partir des nogoods valués contribue assurément à l'accélération de la recherche et à l'élimination des sous problèmes irréalisables. L'évaluation expérimentale montre que notre algorithme est beaucoup plus performant par rapport à SynB&B. Pour nos travaux futurs, une orientation vers l'introduction d'un comportement asynchrone est sera mise en place.

Références

[1] C. Bessière, R. Ezzahir, M. Belaissaoui, and E. H. Bouyakhf. Dischoco : A platform for distributed constraint programming. In *Proceeding of Workshop DCR, IJCAI Hyderabad, India*, 2007.

[2] C. Bessière, A. Maestre and P. Meseguer. Dynamic backtracking distribué. In *JNPC'01*, pages 61–72, Toulouse, France, 2001.

[3] P. Dago. *Extension d'algorithmes dans le cadre des Problémes de satisfaction de contraintes valués : application à l'ordonnancement de systemes satellitaires.* thèse, 1995.

[4] P. Dago. Nogood recording for valued constraint satisfaction problems. In *ICTAI*, pages 132–139, 1996.

[5] P. Dago. Backtrack dynamique valué. In *JFPLC*, pages 133–148, 1997.

[6] R. Ezzahir, C. Bessière, E. H. Bouyakhf, I. Benelallam, and M. Belaissaoui. Asynchronous breadth-first search discop algorithm. In *EUMAS-07 proceeding*, 2007.

[7] A. Gershman, Amnon Meisels, and Roie Zivan. Asynchronous forward-bounding for distributed constraints optimization. In *IJCAI*, pages 103–108. Riva Del Garda, Italy, August 2006.

[8] A. Gershman, Amnon Meisels, and Roie Zivan. Asynchronous forward- bounding with backjumping. In *IJCAI*, pages 28–39, 2007.

[9] M. Ginsberg. Dynamic backtracking. *Journal of Artificial Intelligence Research*, 1 :25–46, 1993.

[10] K. Hirayama and M. Yokoo. *Distributed partial constraint satisfaction problem.* G. Smolka editor, Principales and practice of constraint Programming, 1997.

[11] K. Hirayama and M. Yokoo. An approach to over-constrained distributed constraint satisfaction problems : Distributed hierarchical constraint satisfaction. In *Proceeding of International Conference on Multiagent System*, 2000.

[12] M. Lemaître and G. Verfaillie. An incomplete methode for solving distributed valued constraint satisfaction problems. In *AAAI-97, Workshop on Constraints and Agents.* Providence, RI, USA, 1997.

[13] P. J. Modi, W.-M. Shen, M. Tambe, and M. Yokoo. Adopt : Asynchronous distributed constraint optimisation with quality guarantees. In *AIJ*, pages 161 :149–180, 2005.

[14] J. Scerri, W. Modi, M. Shen, and M. Tambe. Are multiagent algorithms relevant for real hardware : a case study of distributed constraint algorithms. In *ACM Symposium on Applied Computing (SAC'03)*, 2003.

[15] T. Schiex and G. Verfaillie. Nogood recording for static and dynamic csp. In *the 5th IEEE International Conference on Tools with Artificial Intelligence*, pages 48–55. Boston, Ma 1993.

[16] M. C. Silaghi. Dynamic branch-and-bound, an optimization counterpart for dynamic backtracking. In *Report*, 2006.

[17] M. C. Silaghi and M. Yokoo. Nogood-based asynchronous distributed optimization (adopt-ng). In *AAMAS*, 2005.

[18] M. Yokoo. Asynchronous weak-commitment search for solving distributed constraint satisfaction problems. In *Proceding CP-95*, pages 88–102. Cassis, France, 1995.

[19] M. Yokoo, E. H. Durfee, T. Ishida, and K. Kuwabara. The distributed constraint satisfaction problem : Formalization and algorithms. *IEEE Transactions on Knowledge and Data Engineering*, 10(5) :673–685, 1998.

[20] W. Zhang and L. Wittenburg. Distributed breakout revised. In *Proceeding of the Eighteenth National Conference on Artificial Intelligence*, 2002.