Coloration avec préférences : complexité, inégalités valides et vérification formelle

B. Robillard, S. Blazy et E. Soutif

Laboratoire CEDRIC, 292 rue Saint-Martin, 75 141 Paris cedex 03
robillard@ensiie.fr
blazy@ensiie.fr soutif@cnam.fr

Résumé Nous nous intéressons à un problème de coloration avec préférences minimale CPM dans les graphes triangulés. Cette étude s'inscrit dans le projet CompCert qui a pour objectif la certification, à l'aide de méthodes formelles, d'un compilateur optimisant du langage C. L'une des optimisations du compilateur certifié est l'allocation des registres du processeur. Optimiser cette allocation de registres revient à résoudre le problème CPM auquel nous nous intéressons. Nous montrons un résultat de complexité concernant CPM et proposons l'amélioration d'une méthode de coupes permettant la résolution de ce problème. Ce travail est une jonction entre la recherche opérationnelle et les méthodes formelles, dans la mesure où nous vérifions formellement par ailleurs la résolution du problème en prouvant correct le développement, hormis la recherche effectuée par le solveur dont la vérification consiste à déterminer a posteriori si la solution proposée est bien correcte.

Mots-Clefs. coloration ; multicoupes ; complexité (APX-difficulté) ; méthodes formelles ; allocation de registres ; méthodes de coupes.

1 Introduction

Les résultats présentés dans cet article s'inscrivent dans le projet ANR CompCert qui a pour but le développement d'un compilateur du langage C vers le langage assembleur pour PowerPC (architecture RISC[1]), certifié à l'aide de l'assistant à la preuve Coq. Les méthodes de développement formelles permettent de produire du code source certifié, notamment pour les applications sensibles où des considérations de sécurité nécessitent des garanties fortes sur la validité des programmes écrits. Cependant, une faille du processus de certification demeure au niveau de la compilation de ce code, dans la mesure où un bug du compilateur lui-même peut introduire des erreurs dans le code exécutable généré et donc invalider le code source préalablement certifié. De là est né le besoin de compilateurs certifiés. Le projet Compcert a pour but de développer avec l'assistant

[1] On distingue deux types d'architecture de machines : l'architecture RISC, qui nécessite lors de l'exécution d'une instruction processeur que les opérandes soient en registre et l'architecture CISC qui autorise que des opérandes soient stockés en mémoire mais possède un nombre de registres plus restreint.

à la preuve Coq un compilateur réaliste d'un vaste sous-ensemble du langage C vers le langage assembleur du processeur PowerPC, principalement dévolu au domaine des logiciels embarqués ([Ler06]). La certification du compilateur CompCert consiste à prouver que tout code source possédant une sémantique bien définie est compilé en un code exécutable dont la sémantique est la même si aucune erreur de compilation n'intervient. Le compilateur CompCert accomplit un certain nombre de passes de compilation afin de générer le programme exécutable. Il est dit optimisant dans la mesure où il intègre à différents niveaux des optimisations afin de générer du code efficace. Naturellement, les techniques d'optimisation mises en œuvre doivent elles-même être formellement certifiées. Pour plus de détails concernant le projet CompCert, le lecteur pourra se reporter à la page du projet ([Com]).

L'une des passes de compilation, parmi les plus importantes, est l'allocation de registres. C'est à cette passe que nous nous intéressons ici car son optimisation fait intervenir des techniques de recherche opérationnelle. Les variables manipulées par le programme à compiler sont stockées soit dans la mémoire centrale de la machine, soit dans des registres du processeur. L'accès aux registres est beaucoup plus rapide que l'accès à la mémoire centrale. Malheureusement le nombre de registres est limité et il convient donc d'allouer au mieux ces registres aux différentes variables du programme. Il s'agit d'un problème classique de recherche opérationnelle dans le sens où il est généralement utilisé pour illustrer de façon concrète le problème de coloration dans un graphe d'intervalles, appelé graphe d'interférences : chaque intervalle représente schématiquement la durée de vie d'une variable et deux variables vivantes dans des intervalles qui s'intersectent ne peuvent être placées dans le même registre. Le nombre chromatique du graphe fournit alors le nombre minimal de registres qui seraient nécessaires pour pouvoir placer toutes les variables en registres. En pratique, les graphes d'interférences ne sont pas des graphes d'intervalles (notamment car les variables ont en général des durées de vie qui sont des unions d'intervalles et non des intervalles), mais la plupart sont néanmoins triangulés. En pratique également, ce problème a principalement été abordé sous l'angle heuristique et a connu un regain d'intérêt depuis une dizaine d'années où l'approche par programmation mathématique s'est avérée très intéressante (cf. partie 2).

Le but de cet article est triple. Après avoir décrit plus précisément le problème de l'allocation de registres ainsi que sa modélisation par CPM et un état de l'art des méthodes connues aujourd'hui pour le résoudre (partie 2), nous montrerons un résultat de complexité sur le problème de coloration avec préférences CPM dans les graphes triangulés (partie 3). Nous proposerons dans la partie 4 une amélioration d'une méthode de coupes récente ([GH07]) pour la résolution de CPM. Sans entrer dans les détails de la certification, nous indiquerons dans la partie 5 ce qu'il est possible de certifier et de quelle manière lorsqu'on utilise des techniques de recherche opérationnelle telles que celles que nous proposons. La partie 6 présente des perspectives de recherche.

2 L'allocation de registres, une phase cruciale dans l'optimisation de la compilation

Au cours de l'exécution d'un programme, le processeur effectue un grand nombre d'accès à la mémoire afin de lire et écrire les valeurs des variables du programme. Ces accès sont naturellement gourmands en temps. Pour accélérer l'exécution des programmes, le processeur est muni d'un petit nombre[2] de zones de stockage à accès beaucoup plus rapide, les registres. Malheureusement, le nombre de registres d'un processeur est presque toujours inférieur au nombre de variables utilisées simultanément dans un programme. Le but de l'allocation de registres est de déterminer où sont stockées les variables d'un programme à tout moment de son exécution : soit en registres si ces derniers sont disponibles, soit en mémoire le cas échéant. La difficulté est de proposer une affectation optimale des registres. Il est par exemple souvent nécessaire de choisir entre conserver une variable v dans un même registre R pendant l'exécution complète d'un programme (ce qui rend R inutilisable pour stocker d'autres variables), et réutiliser R lorsque la valeur de v n'a plus besoin d'être conservée en vue d'utilisations futures (ce qui nécessite de transférer en mémoire la valeur de v à chaque réutilisation de R). L'allocation de registres est la passe de compilation la plus étudiée et la plus difficile à mettre en œuvre dans un compilateur. La qualité du code compilé dépend en effet fortement de la qualité de l'allocation de registres. Les deux tâches principales de l'allocation de registres sont le vidage de registres en mémoire, et la fusion de registres.

Le vidage décide quelles variables seront stockées ultérieurement en registres. Cette étude repose sur une analyse préalable du programme, appelée analyse de vivacité, qui détermine les interférences entre les variables du programme. Deux variables qui interfèrent ne peuvent simultanément utiliser le même registre. Il est alors possible de tracer le graphe d'interférences du programme. Si le nombre de registres était illimité, toute coloration du graphe d'interférences correspondrait à une utilisation possible des registres (une couleur étant alors associée à un registre). Le nombre chromatique $\chi(G)$ du graphe d'interférences indique donc le nombre minimal de registres permettant de résoudre les interférences entre variables. La détermination de $\chi(G)$ est bien sûr un problème NP-difficile dans les graphes quelconques, mais le problème est polynomial dans les graphes triangulés[3]. Or, comme nous le verrons par la suite, la très grande majorité des graphes d'interférences ont la particularité d'être triangulés. Si le nombre de registres présents est insuffisant, il convient alors de stocker (instruction store) certaines variables en mémoire plutôt qu'en registre et de les charger en registre (instruction load) lorsque cela s'avère nécessaire pour résoudre les interférences. Différentes solutions existent alors (cf. parties 2.2 et 2.3).

[2] De l'ordre de quelques unités à une vingtaine de registres selon l'architecture de la machine

[3] Un graphe triangulé est un graphe ne possédant aucun cycle induit sans corde de longueur supérieure ou égale à 4. La coloration gourmande utilisée à partir de l'inverse d'un ordre d'élimination simplicial est un algorithme polynomial qui permet de trouver $\chi(G)$ dans de tels graphes, cf. par exemple [Wes00].

La fusion tient compte le plus possible des préférences entre variables, afin de minimiser les transferts entre registres. Par exemple, une affectation $x = y$ entraîne une préférence entre les variables x et y, correspondant au fait qu'au vu de cette affectation, il serait préférable de stocker x et y dans le même registre. Ces préférences peuvent alors également se modéliser sur le graphe d'interférences par l'ajout d'une deuxième classe d'arêtes, les arêtes de préférence, comme indiqué dans le paragraphe suivant.

2.1 Modélisation de l'allocation de registres par le graphe d'interférences

Le graphe à colorier est un graphe d'interférences, dont les sommets représentent les variables du programme à compiler. Les arêtes sont de deux types. Les arêtes d'interférence relient tous les sommets qui représentent des variables qui ne doivent pas occuper les mêmes registres. Les arêtes de préférence relient tous les sommets qui représentent des variables telles qu'il existe une instruction d'affectation entre ces variables (et il n'existe pas d'arête d'interférence entre ces sommets). Un poids est associé à chaque arête afin de tenir compte de la fréquence d'exécution des instructions, ainsi que de la fréquence d'utilisation des variables.

Dans ce qui suit, nous définissons un graphe G comme étant un triplet (S, I, P), où G est le graphe dont l'ensemble des sommets est S, l'ensemble des arêtes d'interférence est I et l'ensemble des arêtes de préférence est P. De plus, les graphes formés par S et I d'une part et S et P d'autre part sont respectivement appelés interf-graphe et pref-graphe de G. Soient un entier strictement positif k, représentant le nombre de registres de la machine, et un graphe G. Le problème de coloration avec préférences, noté CPM, consiste à trouver une k-coloration partielle[4] de l'interf-graphe de G[5] qui minimise la fonction $f = \sum_{(i,j)\in D} w_{ij} + c|NC|$ où D est l'ensemble des arêtes de préférence dont les extrémités sont de couleurs différentes (ou l'une des extrémités n'est pas colorée), w_{ij} est le poids associé à l'arête (i, j), NC est l'ensemble des sommets non colorés et c est une constante qui modélise le coût d'un accès mémoire.

Une instance I de CPM est décrite par un doublet (k, G)[6] où k désigne le nombre de couleurs maximum à utiliser et G le graphe à colorer. L'allocation de registres est finalement modélisée par le problème de k-coloration avec préférences appliqué au graphe d'interférences. Chaque couleur représente un registre. Les sommets non colorés correspondent aux variables qui devront être stockées en mémoire (il s'agit du vidage des registres). La figure 1 est un exemple d'instance résolue du problème de 3-coloration avec préférences. En trait plein sont représentées les arêtes d'interférence et en pointillé les arêtes de préférence.

[4] Les sommets ne sont pas nécessairement tous colorés.

[5] Pour simplifier la rédaction nous appelons coloration de G toute coloration de l'interf-graphe de G. De même G est dit triangulé si son interf-graphe est triangulé.

[6] On parlera également pour définir une telle instance de k-coloration avec préférences de G.

Dans cet exemple la coloration réalisée est optimale puisqu'elle colore tous les sommets et que les deux arêtes de préférence de plus forts poids ont des couleurs identiques aux deux extrémités. En effet, il est impossible de satisfaire les trois préférences car sinon les sommets A, B, E et F seraient de la même couleur et donc plusieurs contraintes de coloration seraient violées. Étant donnée cette modélisation, la phase de vidage des registres en mémoire consiste à rechercher un ensemble de sommets k-colorable, c'est-à-dire pouvant être colorés avec k couleurs. La phase de fusion consiste à colorer cet ensemble de sommets.

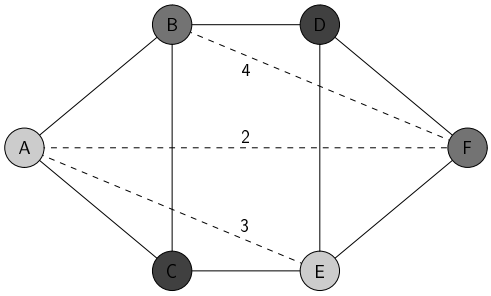

Fig. 1. Un exemple d'instance résolue de CPM.

2.2 Approches heuristiques

CPM étant dans le cas général NP-difficile (puisqu'il généralise le problème de coloration), la quasi-totalité des approches imaginées ont été heuristiques ([Cha82], [BCT94], [GA96], [PP05],...). Ces heuristiques proposent différentes combinaisons des deux phases de vidage et de fusion. Les plus simples effectuent les deux phases de manière séquentielle tandis que d'autres, plus sophistiquées, les réalisent simultanément. Les heuristiques d'allocation de registres évoluent aujourd'hui encore, par exemple afin de tenir compte de la rapidité croissante des processeurs ainsi que du coût croissant des accès à la mémoire. Pour un compilateur d'un langage tel que C, l'heuristique la plus efficace à l'heure actuelle est celle d'Appel et George ([GA96]). C'est d'ailleurs celle qui a été initialement choisie dans le compilateur CompCert. Une autre heuristique récente est celle imaginée par Palsberg et Pereira ([PP05]). Si son efficacité n'est pas suffisante pour remplacer l'heuristique d'Appel et George, ses fondements sont par contre forts intéressants. En effet, suivant la piste ouverte par Andersson ([And03]), ceux-ci ont mesuré qu'une large majorité des graphes d'interférences ont la propriété d'être triangulés. Ils affirment en effet que plus de 95% des graphes d'in-

terférences des méthodes de la bibliothèque Java 1.5 et des 27921 graphes de référence publiés par Appel et George ([AG05]) ont cette propriété. Malheureusement, il a récemment été montré que les problèmes de vidage et de fusion sont également NP-difficiles dans les graphes triangulés ([BDR07b], [BDR07a]).

2.3 Programmation linéaire en nombres entiers appliquée à l'allocation de registres

Les premiers travaux utilisant la programmation linéaire en variables 0,1 pour l'allocation de registres sur des problèmes de petite taille furent ceux de Goodwin et Wilken en 1996 ([GW96]) sur architecture CISC. Les

résultats furent encourageants mais pas suffisamment pour remplacer l'approche traditionnelle. Il s'agissait alors d'une formulation incluant phases de vidage et de fusion. En 2001, Appel et George ([AG01]) ont introduit une formulation de la phase de vidage pour les processeurs à architecture CISC puis ont appliqué une méthode heuristique pour la phase de fusion (leurs tentatives d'utilisation de la programmation mathématique pour la phase de fusion se sont avérées infructueuses, particulièrement en raison du temps de résolution du problème par le solveur). Les résultats furent cependant meilleurs que ceux de Goodwin et Wilken. Cette année, Grund et Hack ([GH07]) ont élaboré un algorithme de coupes (cf. partie 5) pour obtenir un résultat optimal pour la phase de fusion. Leur démarche a permis d'obtenir un résultat optimal pour 471 des 474 graphes de l'Optimal Coalescing Challenge[7] dans des temps très raisonnables pour la plupart des cas (430 cas sont traités en moins de 6 secondes).

2.4 Modélisation du problème de fusion par la programmation linéaire 0-1

Si l'on suppose le problème de vidage résolu (soit parce que le nombre k de registres a été montré suffisant, soit parce que l'une des approches proposées en 2.2 ou 2.3 a été appliquées), il est également possible de modéliser la phase de fusion, qui est donc un problème de k-coloration avec préférences sur le graphe $G = (S, I, P)$, à l'aide du programme mathématique défini dans [GH07]. Nous considérons en effet que tous les sommets du graphe doivent être colorés car un sommet qui n'est pas coloré correspond à une variable stockée en mémoire, ce qui est extrêmement plus coûteux qu'une préférence non-satisfaite. Il existe deux types de variables pour modéliser le problème : d'une part, les variables x_{ic} qui valent 1 si et seulement si le sommet i est de couleur c ; d'autre part les variables y_{ij} qui valent 1 si et seulement si (i, j) est une arête et i et j sont de couleurs différentes. Le programme mathématique est défini dans la figure 2. Il comprend trois séries de contraintes :

– à chaque sommet doit être affectée exactement une couleur ($C1$),

[7] Il s'agit d'une bibliothèque de graphes de référence publiée par Appel pour l'optimisation de la phase de fusion.

- chaque arête d'interférence doit avoir des extrémités de couleurs différentes ($C2$),
- $y_{i,j}$ doit valoir 1 si i et j sont de couleurs différentes ($C3$). En effet, si les couleurs sont différentes alors le membre droit de l'inégalité ($C3$) vaut 1 lorsque c est la couleur de i.

Pour optimiser la coloration il suffit de minimiser le poids des arêtes dont les extrémités sont de couleurs différentes, c'est-à-dire à minimiser $f = \sum_{(i,j)\in P} w_{ij}y_{ij}$.

$$(PF)\begin{cases} Min & \displaystyle\sum_{(i,j)\in P} w_{ij}y_{ij} \\ \text{sous les contraintes} & \\ (C_1)\ \forall i \in \{1,\dots,n(G)\}, & \displaystyle\sum_{c=1}^{k} x_{ic} = 1 \\ (C_2)\ \forall (i,j) \in I,\ \forall c \in \{1,\dots,k\}, & x_{ic} + x_{jc} \leq 1 \\ (C_3)\ \forall (i,j) \in P,\ \forall c \in \{1,\dots,k\}, & x_{ic} - x_{jc} \leq y_{ij} \\ (C_4)\ \forall i \in \{1,\dots,n(G)\},\ \forall c \in \{1,\dots,k\}, & x_{ic} \in \{0,1\} \\ (C_5)\ \forall (i,j) \in P, & y_{ij} \in \{0,1\} \end{cases}$$

Fig. 2. Programme mathématique modélisant la fusion de registres.

Notons qu'un modèle analogue peut prendre en compte le vidage si ce dernier est nécessaire en ajoutant une couleur 0 qui représente un sommet vidé en mémoire et sur laquelle pèse des contraintes plus souples que pour les autres couleurs.

3 Complexité du problème de coloration avec préférences dans les graphes triangulés

Nous savons désormais que la grande majorité des graphes d'interférences s'avèrent triangulés ([PP05]). De plus, lorsque le graphe d'interférences est triangulé, il est possible de connaître en temps polynomial le nombre minimal de registres nécessaires afin que toutes les variables puissent être stockées en registres. Dans la majorité des cas également, le nombre de registres du processeur (de l'ordre de la vingtaine pour une architecture RISC actuelle) est suffisant pour éviter la phase de vidage. Il reste alors à traiter le problème de la fusion, qui a récemment été prouvé NP-difficile dans les graphes triangulés ([BDR07a]).

Cette partie précise le statut du problème CPM dans les graphes triangulés, puisque nous montrons que ce problème n'est pas simplement NP-difficile dans ces graphes mais également APX-difficile[8]. La réduction utilisée permet de plus d'établir que le problème CPM généralise le problème de la multicoupe

[8] La classe APX est la classe des problèmes d'optimisation admettant un algorithme α-approché avec $\alpha \geq 1$ fixé. Un problème d'optimisation est APX-difficile s'il existe

minimale, noté MCM, ce qui donne un enjeu supplémentaire à sa résolution, et d'établir un ratio d'approximation dans certains cas.

Rappelons au préalable la définition du problème de multicoupe minimale. Soient G un graphe non orienté, valué, à valuations positives ou nulles, et $T = \{(s_1, p_1), \ldots, (s_q, p_q)\}$ un ensemble de q paires de sommets de G appelés terminaux. Le problème de la multicoupe minimale associé à (G, T) consiste à sélectionner dans G un ensemble d'arêtes de poids minimum (le poids d'une arête (i, j) étant w_{ij}) dont la suppression ne laisse aucune chaîne entre s_i et p_i pour tout i (l'ensemble de ces arêtes définit une multicoupe, les paires de terminaux sont dites déconnectées).

Théorème 1. Le problème CPM est APX-difficile même si l'interf-graphe est un graphe d'intervalles (et donc s'il est triangulé).

Démonstration. Nous allons exhiber une L-réduction ([PY91]) (R, f) à partir du problème de MCM à trois paires de terminaux, qui est un problème APX-difficile ([DJP+94]). Soit R la transformation qui à une instance $I = (G, T = \{(s_1, p_1), (s_2, p_2), (s_3, p_3)\})$ de MCM associe l'instance $R(I) = (t, G')$ de CPM, où t et G' désignent respectivement le nombre de sommets de G qui sont des terminaux et le graphe construit comme suit :

- G' possède les mêmes sommets que G ;
- (i, j) est une arête d'interférence dans G' si et seulement si (i, j) appartient à T ;
- (i, j) est une arête de préférence de poids p dans G' si et seulement si (i, j) est une arête de G de poids p et (i, j) n'appartient pas à T.

La figure 3 représente un exemple de réduction. Notons que le nombre d'arêtes de l'interf-graphe de G' est 3 puisque I ne possède que 3 paires de terminaux. Or tous les graphes de trois sommets sont des graphes d'intervalles ce qui induit que l'interf-graphe de G' est un graphe d'intervalles (et est donc triangulé puisque les graphes d'intervalles forment une sous-classe des graphes triangulés).

Soit f la fonction qui à toute solution S de $R(I)$ associe l'ensemble des arêtes de préférence dont les extrémités sont de couleurs différentes au vu de la coloration S. Nous allons montrer que $f(S)$ est une solution de I de même valeur que S. Nous considérons, sans entraîner de perte de généralité, qu'aucune arête de G n'appartient à T (ces arêtes font en effet partie de toute multicoupe admissible et peuvent donc être supprimées du graphe).

Soit une arête d'interférence (i, j) du graphe G' associé au problème $R(I)$. S étant une coloration de l'interf-graphe de G', les couleurs associées à i et j sont différentes. Ainsi, toute chaîne d'arêtes de préférence entre i et j possède au moins une arête dont les extrémités sont de couleurs différentes. Puisque les arêtes d'interférence et de préférence de G' correspondent respectivement aux paires de terminaux et aux arêtes de G, il découle que chaque chaîne d'arêtes de G reliant deux terminaux contient au moins une arête qui appartient à $f(S)$. Ainsi, $f(S)$ est une multicoupe de G. De plus, les fonctions économiques de

$\epsilon > 0$ tel que, si P≠NP, il n'existe pas d'algorithme $(1 + \epsilon)$-approché. Enfin un problème est APX-complet s'il appartient à APX et est APX-difficile.

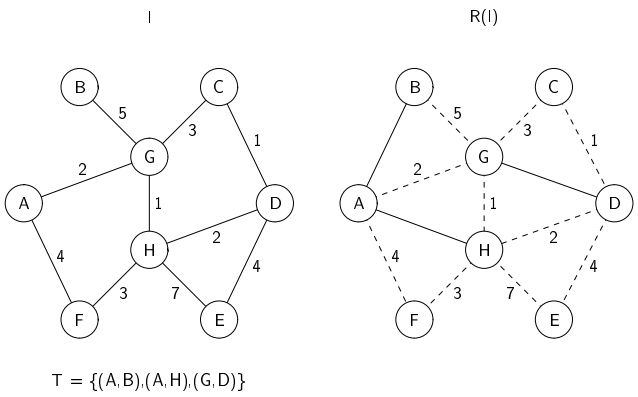

Fig. 3. Une instance I de MCM et $R(I)$.

CPM et MCM sont égales : toutes deux valent $\sum_{(i,j)\in f(S)} w_{ij}$. Donc S et $f(S)$ ont la même valeur. La figure 4 représente une solution S de CPM et $f(S)$ pour les instances présentées précédemment. Trois couleurs suffisent à obtenir une coloration avec préférences optimale (alors qu'il y a 5 sommets dans l'interf-graphe) : une pour A et F , une pour B, C et G et enfin une pour H, D et E. Les arêtes en gras sont celles de la multicoupe $f(S)$.

Pour achever la preuve nous allons montrer que les deux problèmes possèdent le même optimum. Puisque nous avons prouvé qu'à partir d'une solution de CPM nous pouvons construire une solution de valeur égale de MCM, il ne reste qu'à prouver le cheminement inverse.

Nous définissons la fonction g qui à toute solution C de I associe une coloration de G' comme suit :

- supprimer de G' les arêtes de C ;
- ordonner les terminaux de 1 à t et colorer la composante pref-connexe de i avec la plus petite couleur n'ayant jamais été utilisée ;
- colorer tous les sommets non colorés avec la couleur 1 ;
- ajouter les arêtes de C à G'.

Nous allons montrer que si une multicoupe C est une solution de I alors la coloration $g(C)$ est une solution de $R(I)$ de valeur inférieure ou égale.

Si C est une multicoupe alors chaque paire de terminaux (s_i, p_i) est déconnectée dans G. Comme G correspond au pref-graphe de G' et que les arêtes d'interférence sont exactement les paires de T, on peut déduire qu'il n'existe pas d'arête d'interférence interne à une composante pref-connexe de G'. De plus, toutes les composantes pref-connexes qui contiennent au moins un sommet de l'interf-graphe de G' sont colorées par des couleurs distinctes, ce qui utilise au

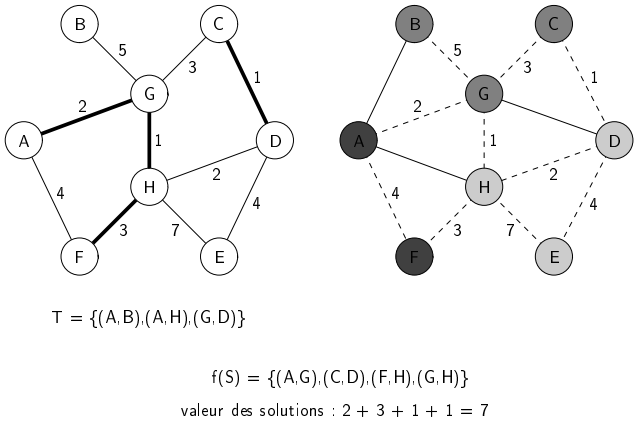

Fig. 4. Une solution S de $R(I)$ et $f(S)$.

plus t couleurs. Les sommets n'appartenant pas aux composantes pref-connexes déjà colorées sont des sommets isolés de l'interf-graphe, il est donc possible de les colorer tous avec une seule couleur utilisée précédemment. Ainsi, tous les sommets de G' sont colorés et chaque contrainte d'interférence est respectée, ce qui implique que $g(C)$ est une t-coloration de G'. De plus, toutes les arêtes de C ont des extrémités de couleurs différentes, sauf éventuellement celles qui sont incidentes à des composantes pref-connexes ne contenant pas de sommets de l'interf-graphe, donc la valeur de $g(C)$ est inférieure ou égale à celle de C. Ainsi, S est une solution optimale de $R(I)$ si et seulement si $f(S)$ est une solution optimale de I. En effet, l'optimum de I est inférieur ou égal à celui de $R(I)$ puisque f associe à toute solution de $R(I)$ une solution de I de même valeur. D'autre part, l'optimum de I est supérieur à celui de $R(I)$ puisque g associe à toute solution de I une solution de $R(I)$ de valeur inférieure ou égale.

Cette preuve implique en outre que l'ensemble des arêtes de préférence dont les couleurs sont différentes forment une multicoupe du pref-graphe si le nombre de couleurs utilisables est supérieur ou égal au nombre de sommets de l'interf-graphe. Autrement dit, MCM est un cas particulier de CPM. Résoudre efficacement ce dernier problème pourrait donc améliorer la résolution de MCM. Notons enfin que cette preuve établit également la préservation du ratio d'approximation. Aussi CPM possède un algorithme $O(log|I|)$-approché si le nombre de couleurs est supérieur ou égal au nombre de sommets de l'interf-graphe [GVY93].

4 Algorithme de coupes

Au vu de la complexité de CPM dans les graphes triangulés nous choisissons une résolution par énumération implicite afin d'obtenir un résultat exact. Outre certaines techniques de réduction de taille que nous ne détaillerons pas ici, nous utilisons un algorithme de coupes, initialement présenté par Grund et Hack ([GH07]), auquel nous ajoutons de nouvelles inégalités valides.

L'algorithme de Grund et Hack introduit diverses inégalités valides pour CPM et en particulier les coupes de chaînes définies comme suit. Pour toute arête d'interférence (i, j) et toute chaîne d'arêtes de préférence $(e_1, \ldots, e_l)$ reliant i à j il existe une arête de la chaîne dont les extrémités sont de couleurs différentes ; autrement dit, l'inégalité $\sum_{i=1}^{l} y_{e_i} \geq 1$ est valide.

Ces coupes correspondent simplement aux inégalités classiques de MCM. Nous souhaitons étendre ces inégalités à des chaînes d'arêtes de préférence reliant les deux extrémités d'une arête de préférence, c'est-à-dire aux cycles d'arêtes de préférence. De plus nous voulons également que les coupes agissent si deux sommets quelconques d'un cycle ont des couleurs différentes. En effet, si deux sommets ont des couleurs différentes au sein d'un cycle d'arêtes alors il y a au moins deux arêtes dont les extrémités sont de couleurs différentes dans ce cycle. Nous définissons dans ce but les coupes de cycles.

Théorème 2. Pour tout cycle $(e_1, \ldots, e_l)$, toute paire de sommets (i, j) de ce cycle et toute couleur c de {1,. . .,k} l'inégalité $2(x_{ic} - x_{jc}) \leq \sum_{i=1}^{l} y_{e_i}$ est valide.

Démonstration. On distingue deux cas. Dans le premier cas i et j sont de la même couleur et alors le membre de gauche vaut 0 pour toute couleur c et donc l'inégalité est trivialement valide. Dans le second cas, i et j sont de couleurs différentes et donc le membre gauche de l'inégalité vaut 2 si c est la couleur associée à i, -2 si c est la couleur associée à j et 0 sinon. Si le membre de gauche vaut 2 alors il y a au moins deux arêtes du cycle qui ont des extrémités de couleurs différentes et donc l'inégalité est valide, sinon l'inégalité est trivialement valide.

La figure 5 décrit un cas dans lequel certaines inégalités de cycle sont violées. Les arêtes du cycle ont été orientées de sorte que l'arête (i, j) est orientée de i vers j si y_{ij} apparaît dans (PF) (on remarque en effet que les contraintes C_4 ne sont pas symétriques). A coté de chaque sommet i du cycle apparaît le vecteur x_i (on considère pour cet exemple 3 couleurs). Ainsi nous pouvons calculer les valeurs de chaque y_{ij} : $y_{AB} = 0.2$, $y_{BC} = 0.2$, $y_{CD} = 0.2$ et $y_{DA} = 0.1$. Par conséquent, deux inégalités de cycle sont violées ici, une pour les sommets A et C $(2(x_{A1} - x_{C1}) = 0.8)$ et une pour B et C $(2(x_{B3} - x_{C3}) = 0.8)$.

Notons que ces inégalités sont valides pour tout cycle d'arêtes, indépendamment du nombre d'arêtes de préférence et d'interférence que contient le cycle. Ainsi, si un cycle contient une arête d'interférence alors une des coupes de cycles implique la coupe de chaîne correspondant à l'arête d'interférence du cycle. Par contre, au-delà de deux arêtes d'interférence ces coupes ne suffisent plus car les inégalités sont trivialement vraies. Il pourrait donc être intéressant de généraliser ces inégalités à un nombre quelconque de sommets de couleurs distinctes.

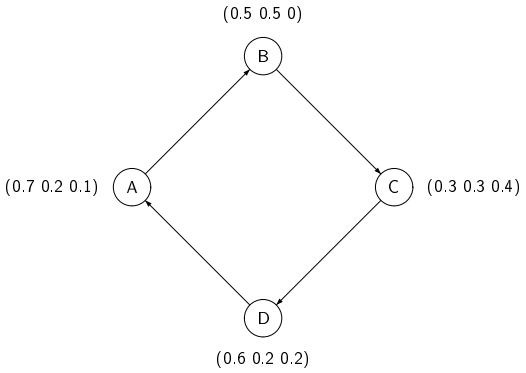

Fig. 5. Un cas de coupes de cycles pour les sommets A et C (première couleur) et les sommets B et C (troisième couleur).

L'utilisation d'inégalités portant sur les cycles du graphe soulève un autre inconvénient : le nombre de cycles d'un graphe est exponentiel. Il est donc nécessaire de limiter le nombre de cycles utilisés pour les coupes. Manifestement les cycles à utiliser contiennent au plus une arête d'interférence. Cependant il est pour lors difficile de conjecturer si les cycles les plus courts, qui permettent de fixer plus précisément les variables, ou les cycles les plus longs, qui produisent plus de coupes, doivent être privilégiés.

5 Vérification formelle de la résolution de CPM

La résolution de CPM que nous réalisons se situe au cœur du développement d'un compilateur certifié formellement. Aussi est-il nécessaire de vérifier formellement le code que nous écrivons. Nous espérons que cette étude permettra d'améliorer la cohabitation entre les deux disciplines de pointe que sont la recherche opérationnelle et les méthodes de développement formel. Quelques utilisations du système Coq, avec lequel est développement CompCert, ont concerné des problèmes d'optimisation, comme par exemple la preuve du théorème des quatre couleurs.

Ainsi nous avons développé dans le cadre de notre étude l'algorithme de coloration gourmande puis l'avons vérifié formellement et prouvé optimal si le graphe que l'on colore est triangulé. En effet, la recherche de la plus grande clique suffit pour conclure à l'existence d'une k-coloration du graphe d'interférence. Mais pour le prouver dans la logique intuitionniste qu'utilise Coq il est nécessaire d'exhiber une k-coloration, par exemple celle construite par l'algorithme de coloration gourmande. Ceci a nécessité de spécifier et prouver la correction d'un algorithme de recherche d'ordre d'élimination simplicial et de l'algorithme de co-

loration gourmande. Ce développement représente environ 10000 lignes de Coq et est décrit dans [BRS08].

Pour ce qui est de la programmation linéaire en variables 0-1 et de l'algorithme de coupes nous utilisons actuellement CPLEX, construisons la coloration associée aux valeurs des variables de la solution renvoyée puis nous vérifions avec Coq qu'il s'agit bien d'une coloration du graphe d'interférences. Ce type de vérification a l'intérêt d'être plus facile à réaliser mais présente deux inconvénients majeurs : elle est réalisée pour chaque exécution du programme et l'optimalité de la solution n'est pas prouvée. Malheureusement, un solveur vérifié formellement n'existe pas et constitue un projet bien trop ambitieux pour lors au vu de l'efficacité des solveurs commerciaux. À terme, nous espérons néanmoins prouver en Coq que toute solution du programme (PF) permet de construire une coloration valide du graphe et développer une bibliothèque Coq plus générale portant sur les programmes linéaires en variables 0-1 afin de faciliter l'accès à cette méthode pour la communauté du développement formel.

6 Perspectives de recherche

Le problème de coloration avec préférences a été relativement peu étudié et les voies de recherche potentielles s'avèrent donc nombreuses. Au premier ordre apparaît la recherche de méthodes de réduction de taille du graphe d'interférences et de préférence sur lesquelles nous travaillons encore actuellement et qui semblent prometteuses. Nous pensons également possible d'améliorer l'algorithme de coupes en déterminant de nouvelles inégalités valides. En effet, les inégalités présentées dans cet article ne concernent que des paires de sommets dont les couleurs sont différentes et nous espérons les étendre à un nombre quelconque de sommets colorés. De plus, il est nécessaire comme nous l'avons évoqué de choisir quels cycles élémentaires permettent en pratique de réaliser le plus de coupes. Enfin nous pourrions ajouter aux inégalités valides de CPM certaines inégalités valides de MCM ou du problème de coloration simple.

Nous aimerions également nous restreindre à des graphes plus particuliers, comme les graphes bipartis dont on sait qu'il sont 2-colorables afin de résoudre MCM. Notons ensuite que CPM permet d'aborder MCM sous un angle totalement différent des approches classiques puisque nous nous intéressons principalement aux caractéristiques du graphe formé par les arêtes d'interférence (qui correspondent aux terminaux de MCM) et non aux arêtes de préférence (qui correspondent aux arêtes du graphe de MCM). Il serait donc également intéressant d'étudier CPM dans un cadre où l'interf-graphe et le pref-graphe sont particuliers. De plus, comme il est possible de le voir sur l'exemple de la figure 4, le nombre de sommets de l'interf-graphe ne correspond pas forcément avec le nombre de couleurs utilisées dans une solution optimale, d'où l'intérêt de se demander si l'on peut déterminer, ou du moins borner, dans certains cas le nombre de couleurs nécessaire pour que la solution de CPM corresponde à une multicoupe optimale.

Ensuite, cet article établit un ratio d'approximation pour certains cas de CPM. Savoir s'il existe un algorithme approché pour CPM est donc un problème qui reste ouvert.

Enfin, puisque notre étude concilie optimisation et méthodes formelles nous souhaitons améliorer l'accès aux méthodes d'optimisation, particulièrement celles utilisées pour notre problème, afin d'inclure des méthodes efficaces d'optimisation à des développements vérifiés formellement.

7 Conclusion

Cet article a présenté le problème CPM qui permet de modéliser, entre autres, l'allocation de registres. Nous avons montré que ce problème est APX-difficile dans les graphes triangulés, qui regroupent la majeure partie des graphes d'interférences, et généralise deux problèmes classiques d'optimisation combinatoire : le problème de coloration et le problème de multicoupe minimale. Nous avons en outre décrit de nouvelles inégalités valides pour CPM et présenté succinctement le processus de vérification formelle associé à la démarche de résolution de ce problème.

De plus, de multiples voies de recherche sont ouvertes car les aspects théoriques de ce problème ont relativement peu été étudiés. Nous espérons donc que des recherches approfondies permettront d'améliorer grandement la résolution de CPM. Les applications de ce problème sont extrêmement nombreuses, puisqu'il permet non seulement d'affiner le modèle de coloration classique, mais aussi de résoudre le problème de multicoupe minimale.

Références

[AG01] Andrew W. Appel and Lal George. Optimal spilling for CISC machines with few registers. In *SIGPLAN Conference on Programming Language Design and Implementation*, pages 243–253, 2001.

[AG05] Andrew W. Appel and Lal George. 27,921 actual register-interference graphs generated by standard ML of New Jersey, version 1.09 – `http ://www.cs.princeton.edu/~appel/graphdata/`, 2005.

[And03] Christian Andersson. Register allocation by optimal graph coloring. In *Compiler Construction (CC)*, pages 33–45, 2003.

[BCT94] Preston Briggs, Keith D. Cooper, and Linda Torczon. Improvements to graph coloring register allocation. *Transactions on Programming Languages and Systems (TOPLAS)*, 16(3) :428 – 455, 1994.

[BDR07a] Florent Bouchez, Alain Darte, and Fabrice Rastello. On the complexity of register coalescing. In *International symposium on code generation and optimization (cgo'07)*, San Jose, USA, mar 2007. IEEE Computer Society Press. Best paper award.

[BDR07b] Florent Bouchez, Alain Darte, and Fabrice Rastello. On the complexity of spill everywhere under ssa form. In *Acm sigplan/sigbed conference on languages, compilers, and tools for embedded systems (lctes'07)*, San Diego, USA, jun 2007.

[BRS08] Sandrine Blazy, Benoît Robillard, and Eric Soutif. Vérification formelle d'un algorithme d'allocation de registres par coloration de graphe. *Actes des Journées Francophones des Langages Applicatifs 2008, 26-29 Janvier 2008, Étretat, France*, 2008.

[Cha82] G J Chaitin. Register allocation and spilling via graph coloring. *Symposium on Compiler Construction*, 17(6) :98 – 105, 1982.

[Com] Projet CompCert. http ://pauillac.inria.fr/ xleroy/compcert/.

[DJP+94] E. Dahlhaus, D. S. Johnson, C. H. Papadimitriou, P. D. Seymour, and M. Yannakakis. The complexity of multiterminal cuts. *SIAM J. Comput.*, 23(4) :864–894, 1994.

[GA96] Lal George and Andrew W. Appel. Iterated register coalescing. *ACM Trans. Program. Lang. Syst.*, 18(3) :300–324, 1996.

[GH07] Daniel Grund and Sebastian Hack. A fast cutting-plane algorithm for optimal coalescing. volume 4420 of *Lecture Notes in Computer Science*, pages 111–125. Springer, 2007.

[GVY93] Naveen Garg, Vijay V. Vazirani, and Mihalis Yannakakis. Approximate max-flow min-(multi)cut theorems and their applications. In *ACM Symposium on Theory of Computing*, pages 698–707, 1993.

[GW96] David W. Goodwin and Kent D. Wilken. Optimal and near-optimal global register allocations using 0-1 integer programming. *Softw. Pract. Exper.*, 26(8) :929–965, 1996.

[Ler06] Xavier Leroy. Formal certification of a compiler back-end or : Programming a compiler with a proof assistant. *33rd symposium Principles of Programming Languages*, pages 42–54, 2006.

[PP05] Fernando Magno Quintão Pereira and Jens Palsberg. Register allocation via coloring of chordal graphs. *Programming Languages and Systems, 3rd Asian Symp., APLAS 2005, Japan, November, 2005, Proc.*, 3780 :315–329, 2005.

[PY91] C.H. Papadimitriou and M. Yannakakis. Optimization, approximation and complexity classes. *J. Comput. System Sci.*, 43 :425 – 440, 1991.

[Wes00] Douglas B. West. *Introduction to Graph Theory (2nd Edition)*. Prentice Hall, August 2000.