RAIRO Operations Research
RAIRO Oper. Res. **37** (2003) 29-66
DOI: 10.1051/ro:2003013

COLORATION DE GRAPHES : FONDEMENTS ET APPLICATIONS

Dominique de Werra[1] et Daniel Kobler[2]

Abstract. The classical colouring models are well known thanks in large part to their applications to scheduling type problems; we describe the basic concepts of colourings together with a number of variations and generalisations arising from scheduling problems such as the creation of school schedules. Some exact and heuristic algorithms will be presented, and we will sketch solution methods based on tabu search to find approximate solutions to large problems. Finally we will also mention the use of colourings for creating schedules in sports leagues and for computer file transfer problems. This paper is an extended version of [37].

Résumé. Les modèles classiques de coloration doivent leur notoriété en grande partie à leurs applications à des problèmes de type emploi du temps ; nous présentons les concepts de base des colorations ainsi qu'une série de variations et de généralisations motivées par divers problèmes d'ordonnancement dont les élaborations d'horaires scolaires. Quelques algorithmes exacts et heuristiques seront présentés et nous esquisserons des méthodes basées sur la recherche Tabou pour trouver des solutions approchées pour des problèmes de grande taille. Enfin nous mentionnons l'application des colorations à la confection de calendriers de ligues de sport et à des problèmes de transferts de fichiers informatiques. Ce texte est une version étendue de [37].

Received December, 2002.

[1] IMA FSB, École Polytechnique Fédérale de Lausanne, 1015 Lausanne, Suisse ; e-mail : dewerra.ima@epfl.ch
[2] Tm Bioscience, 439 University Ave. Ste. 1100, Toronto (Ontario) M5G 1Y8, Canada ; e-mail : dkobler@tmbioscience.com

1. Colorations et emploi du temps

Imaginons que nous ayons une collection $X = \{x_1, ..., x_n\}$ de tâches de même durée à faire exécuter en respectant une série E de contraintes qui spécifie que certaines paires $[x_i, x_j]$ de tâches ne peuvent être simultanément en cours d'exécution.

Si nous admettons que chaque tâche est indivisible et dure exactement un jour, nous pouvons nous demander s'il est possible d'exécuter toutes les tâches dans un intervalle I donné de k jours en respectant les contraintes, ou encore quel est le plus petit intervalle de temps nécessaire à l'exécution de toutes les tâches.

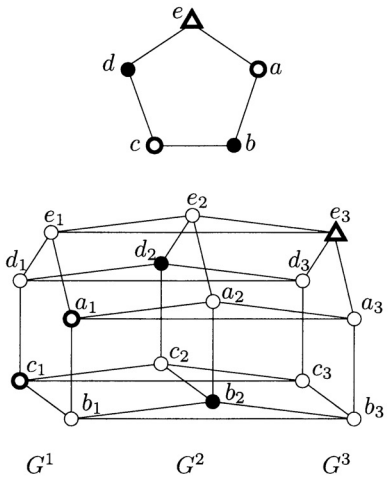
Ce sont là précisément des questions qui se formulent en termes de coloration : associons à chaque tâche x_i de la collection X un sommet x_i d'un graphe simple G et à chaque paire x_i, x_j de tâches liées par une contrainte de non-simultanéité (on parle de contrainte de *disjonction*) nous faisons correspondre une arête $[x_i, x_j]$ de ce graphe qui sera noté $G = (X, E)$. Si l'ensemble des jours de l'intervalle I est constitué des jours $1, 2, ..., k$, alors la première question revient à demander s'il est possible d'attribuer à chaque tâche x_i un jour d'exécution $c(i)$ choisi dans l'ensemble $\{1, 2, ..., k\}$ de manière que pour toute paire de tâches associée à une arête $[x_i, x_j]$ de E on ait $c(i) > c(j)$ ou $c(j) > c(i)$. En d'autres termes $c(i) \neq c(j)$. C'est là précisément une coloration (des sommets) du graphe G. Ainsi, une *coloration* d'un graphe simple $G = (X, E)$ est une affectation à chaque sommet x_i de X d'une couleur $c(i)$ de telle manière que pour toute arête $[x_i, x_j]$ de E on ait $c(i) \neq c(j)$.

On parle de *k-coloration* d'un graphe si l'ensemble I des couleurs utilisables est de cardinalité k. Quant à la seconde question, à savoir quel est le plus petit intervalle I de jours qui permette d'exécuter toutes les tâches, elle revient à chercher le plus petit $k = |I|$ pour lequel le graphe G admet une k-coloration ; c'est le *nombre chromatique* $\chi(G)$ du graphe G.

Par habitude, on parle de coloration d'un graphe, alors que très souvent – et c'est le cas dans l'exemple ci-dessus – on utilise des entiers (non négatifs) $c(i)$ au lieu de couleurs ; on devrait donc parler de "numérotation". Le fait d'utiliser un ensemble $I = \{1, ..., k\}$ d'entiers au lieu de couleurs nous permet d'entrevoir les liens qui existent entre les colorations et les orientations d'un graphe. L'ensemble I est en effet ordonné et l'on peut formuler :

Propriété 1. *Un graphe $G = (X, E)$ admet une k-coloration si et seulement si on peut orienter chaque arête de G de manière à obtenir un graphe orienté H sans circuits et sans chemin rencontrant plus de k sommets.*

Justification. Si G a une k-coloration utilisant les couleurs $1, 2, ..., k$, alors pour toute arête $[x_i, x_j]$ nous choisissons l'orientation $x_i \longrightarrow x_j$ si $c(i) < c(j)$ ou $x_i \longleftarrow x_j$ si $c(i) > c(j)$. Le graphe H obtenu a bien la forme voulue. Inversement, si un graphe H a une telle orientation, nous associons à chaque sommet x_i une valeur $c(i)$ égale au nombre de sommets rencontrés sur le plus long chemin se terminant en x_i ; on vérifie facilement que les $c(i)$ définis sont une k-coloration du graphe G obtenu en supprimant les orientations dans H. □



a) le graphe G avec une 3-coloration utilisant les couleurs (○, ●, ▲); G a 5 sommets.

b) le graphe $\widetilde{G}(3)$ avec un ensemble stable S (sommets en gras) de 5 sommets.

La coloration de G s'obtient à partir de $S = \{a_1, c_1, b_2, d_2, e_3\}$ en donnant à a et c la couleur 1, à b et d la couleur 2 et à e la couleur 3.

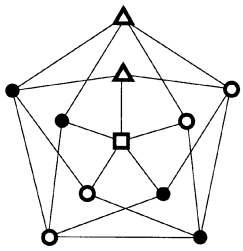
FIGURE 1. Réduction de la coloration à un problème d'ensemble stable.

Il est intéressant de remarquer que la question de l'existence d'une k-coloration d'un graphe simple $G = (X, E)$ peut se ramener à la question de l'existence d'un ensemble stable de $|X|$ sommets dans un graphe auxiliaire $\widetilde{G}(k)$. Cette remarque sera exploitée plus loin lorsque nous aurons à traiter des problèmes de coloration en présence de couleurs interdites pour certains sommets.

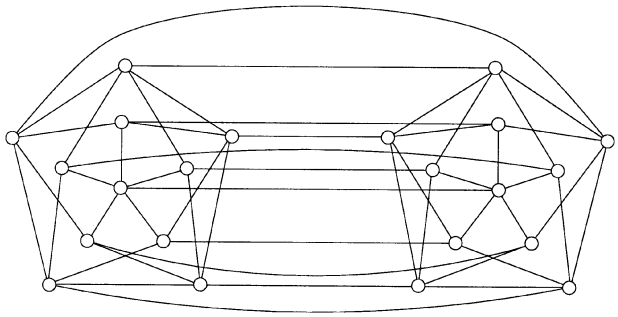
Supposons donc que nous ayons à déterminer si le graphe $G = (X, E)$ admet une k-coloration (k fixé). Nous associons à G un graphe auxiliaire $\widetilde{G}(k)$ obtenu en introduisant k copies $G^1, ..., G^k$ du graphe G : à chaque sommet a de G correspond un sommet a_i de G^i et l'on relie a_i, b_i par une arête $[a_i, b_i]$ si $[a, b]$ est une arête de G. Enfin, pour tout sommet a de G, on relie les sommets correspondants $a_1, ..., a_k$ de manière à former une clique. La construction est illustrée dans la Figure 1.

Il y a correspondance biunivoque entre les k-colorations de G et les ensembles stables de $\widetilde{G}(k)$ avec $|X|$ sommets. En effet, soit S un tel ensemble stable de $\widetilde{G}(k)$, on en déduit une k-coloration par la règle suivante : si $a_i \in S$, alors a a la couleur i dans la coloration. De même à partir d'une k-coloration de G on construit un ensemble stable de $|X|$ sommets dans G : si le sommet a a la couleur i, alors $a_i \in S$. Dans la figure 2, on utilise la construction pour examiner si un graphe a une k-coloration.

Cette transformation du problème de k-coloration en un problème d'ensemble stable est encore utile pour résoudre la question suivante qui a des applications dans certains problèmes d'ordonnancement : quel est le nombre maximum de sommets d'un graphe G donné que l'on peut colorer en utilisant un nombre fixé k de



a) le graphe G avec une 4-coloration utilisant les couleurs (●, ●, ▲, ■); G a 11 sommets.



b) le graphe $\widetilde{G}(2)$ associé au graphe G ci-dessus.

On peut colorer G en 2 couleurs si et seulement si $\widetilde{G}(2)$ admet un ensemble stable de 11 sommets. Puisque dans chaque sous-graphe induit G^1, G^2 de $\widetilde{G}(2)$ on ne peut choisir que 5 sommets pour former un ensemble stable, dans $\widetilde{G}(2)$ tout ensemble stable a au plus 10 sommets; G n'a ainsi pas de 2-coloration.

FIGURE 2. Une tentative de colorer G avec $k = 2$ couleurs.

couleurs ? Ce problème se résout en cherchant un ensemble stable de taille maximum dans $\widetilde{G}(k)$; en termes d'emploi du temps, ceci revient à se demander quel est le nombre maximum de tâches de la collection donnée que l'on parvient à exécuter dans un intervalle de k jours.

Alors que pour le traitement direct en termes de coloration, on ne connaît généralement pas d'algorithme combinatoire, la formulation en termes d'ensemble stable permet de recourir à des algorithmes (exacts ou heuristiques) de construction d'un ensemble stable de cardinalité maximum.

2. Quelques résultats de complexité

Le problème de la détermination du nombre chromatique est difficile. Plus précisément, déterminer si un graphe donné a une k-coloration pour un $k > 2$

fixé est NP-complet [15]. De plus, il est également NP-complet de savoir s'il existe une $(2-\epsilon)\chi(G)$-coloration d'un graphe G pour tout $\epsilon > 0$ [14].

Si $A(G)$ est le nombre de couleurs utilisées par un algorithme polynomial A pour colorer un graphe $G = (X, E)$, il n'est pas possible d'avoir systématiquement

$$A(G) \leq n^{\frac{1}{7}-\epsilon} \cdot \chi(G)$$

pour $\epsilon > 0$ [2]. Par contre, un algorithme polynomial A' bien choisi permet d'avoir

$$A'(G) \leq c \cdot n \cdot \frac{(\log\log n)^2}{(\log n)^3} \cdot \chi(G) \leq c \cdot \frac{n}{\log n} \cdot \chi(G)$$

où c est une constante [19].

Il existe des classes de graphes pour lesquels le nombre chromatique peut mieux être approché, voire trouvé de manière exacte, en temps polynomial (comme les graphes parfaits par exemple, que nous verrons plus loin dans cet article). Dans le cas particulier où G est un graphe planaire (c'est-à-dire un graphe que l'on peut représenter dans le plan sans qu'il n'ait de croisement d'arêtes), on peut montrer que $\chi(G) \leq 5$ [3]. En fait, il est même possible de montrer que $\chi(G) \leq 4$, mais les démonstrations connues sont bien plus ardues.

Afin de déterminer le nombre chromatique d'un graphe, un algorithme exact peut être appliqué si la taille du graphe n'est pas trop grande. Sinon, on se contente d'estimer le nombre chromatique, par exemple en utilisant un algorithme heuristique qui ne donnera qu'une borne supérieure mais qui est bien plus rapide qu'un algorithme exact.

3. Les algorithmes séquentiels de coloration

Si $\omega(G)$ est la taille maximale d'une clique (c'est-à-dire un ensemble de sommets qui sont tous reliés deux à deux) de G, il est facile de se convaincre que $\chi(G) \geq \omega(G)$. Malheureusement, $\omega(G)$ peut être une borne inférieure de mauvaise qualité ; en effet, il est possible de construire des graphes G ne contenant pas de clique de taille 3 (donc $\omega(G) \leq 2$) et ayant une valeur de $\chi(G)$ arbitrairement grande. D'autres bornes inférieures peuvent être trouvées dans [3], comme par exemple

$$\chi(G) \geq \frac{n^2}{n^2 - 2m}$$

pour un graphe G à n sommets et m arêtes.

Un autre moyen d'estimer $\chi(G)$ consiste à trouver des bornes supérieures. Pour cela, il suffit de trouver des colorations. Nous présentons ici quelques méthodes de coloration simples, basées sur l'algorithme séquentiel de coloration suivant :

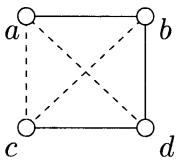


FIGURE 3. Le graphe P_4.

```
Algorithme séquentiel de coloration ;
begin
    déterminer un ordre 𝒪 = (x_1, x_2, ..., x_n) des sommets de G ;
    for i := 1 to n do
         donner à x_i la plus petite couleur possible ;
                --  c'est-à-dire la plus petite couleur qui
                --  n'est pas utilisée par x_j adjacent à x_i
                --  pour tout j < i
    end for ;
end algorithme.
```

Cette méthode très générale est un *algorithme séquentiel basé sur l'ordre* $\mathcal{O}$. L'ordre utilisé a bien évidemment une influence sur le nombre de couleurs utilisées, mais l'on a :

Propriété 2. *Pour tout graphe G, il existe un ordre $\mathcal{O}$ des sommets pour lequel l'algorithme séquentiel basé sur l'ordre $\mathcal{O}$ fournit une coloration en $\chi(G)$ couleurs.*

Justification. Pour se convaincre qu'un tel ordre existe, considérons une k-coloration de G avec $k = \chi(G)$. Il suffit ensuite de créer un ordre dans lequel les sommets de couleur 1 précèdent ceux de couleur 2, eux-mêmes précédant ceux de couleur 3, et ainsi de suite. □

La difficulté du problème de coloration réside dans le fait qu'un tel ordre peut être difficile à trouver. Dans le cas particulier des graphes sans P_4 (c'est-à-dire ne contenant pas la structure de la Fig. 3), n'importe quel ordre des sommets permet à l'algorithme séquentiel d'obtenir une coloration avec le nombre minimal de couleurs.

Nous représentons par $\Delta(G)$ le degré maximum de G, c'est-à-dire le nombre maximum d'arêtes adjacentes à un même sommet de G. De par le fonctionnement de l'algorithme séquentiel (le sommet à colorer reçoit la plus petite couleur disponible), nous sommes assurés que le nombre total de couleurs utilisées ne dépasse pas $\Delta(G) + 1$, quel que soit l'ordre $\mathcal{O} = (x_1, x_2, \ldots, x_n)$. Ainsi $\chi(G) \leq \Delta(G) + 1$ pour tout graphe G. En suivant le même raisonnement, nous pouvons être plus

précis : si G_i est le sous-graphe de G induit par $x_1, \ldots, x_i$ et $d_H(x)$ le nombre de voisins de x dans H, nous avons :

Propriété 3. $\chi(G) \leq 1 + \max_{1 \leq i \leq n} d_{G_i}(x_i)$.

Dans le but d'obtenir la meilleure borne possible, on peut vouloir chercher un ordre $\mathcal{O} = (x_1, x_2, \ldots, x_n)$ tel que $\max_{1 \leq i \leq n} d_{G_i}(x_i)$ soit aussi petit que possible. Mais les valeurs $d_{G_i}(x_i)$ ne sont pas connues à l'avance. C'est pourquoi l'on préfère la borne

$$\chi(G) \leq 1 + \max_{1 \leq i \leq n} \min(i-1, d_G(x_i))$$

qui découle de la propriété précédente car $d_{G_i}(x_i) \leq \min(i-1, d_G(x_i))$. Cette borne est minimisée en utilisant un ordre $\mathcal{O} = (x_1, x_2, \ldots, x_n)$ tel que $d_G(x_1) \geq d_G(x_2) \geq \ldots \geq d_G(x_n)$ [30].

Différentes autres méthodes pour choisir $\mathcal{O}$ ont été proposées dans la littérature. Les meilleures sont dynamiques, c'est-à-dire qu'elles ne fixent pas cet ordre complètement dès le début. L'une des plus efficaces est RLF (pour "Recursive Largest First"), proposée dans [24], qui fonctionne de la manière suivante. Le premier sommet x_1 est un sommet de degré maximal, et reçoit la couleur 1. Supposons ensuite que i sommets (x_1, ..., x_i) ont reçu la couleur 1. Soit alors N_0 (respectivement N_1) l'ensemble des sommets non colorés adjacents à aucun (respectivement au moins un) sommet coloré ; le sommet x_{i+1} est choisi parmi les sommets x de N_0 maximisant $d_{N_1}(x)$. Cela est répété jusqu'à ce que N_0 soit vide. Le processus est ensuite itéré avec la couleur 2 sur le graphe généré par les sommets non colorés, et ainsi de suite jusqu'à ce que tous les sommets aient reçu une couleur.

Les algorithmes séquentiels ont l'avantage de la rapidité (quelques secondes de calcul sur un ordinateur pour des graphes de plusieurs centaines, voire milliers, de sommets), mais utilisent en général plus de couleurs que nécessaire. À l'opposé, les algorithmes exacts permettent de déterminer $\chi(G)$, mais nécessitent des temps de calculs très grands (si l'on se limite à une journée de calcul sur un ordinateur, il n'est actuellement guère possible de considérer des graphes aléatoires de plus de 85 sommets).

4. Un algorithme exact de coloration

Le seul moyen connu pour déterminer le nombre chromatique d'un graphe G est de faire une énumération (implicite) de toutes les colorations de G. On commence en général par considérer une borne supérieure q de $\chi(G)$; celle-ci peut être obtenue à l'aide d'un algorithme séquentiel ou, mieux, en appliquant la méthode Tabou décrite plus loin.

Nous colorons ensuite G avec l'algorithme séquentiel basé sur un ordre $x_1, \ldots, x_n$; si la couleur q n'a pas été utilisée, nous avons obtenu une coloration en $q_1 < q$ couleurs. Nous remplaçons alors q par q_1 et remontons dans l'algorithme séquentiel (en décolorant les sommets) jusqu'au sommet x_r tel que x_{r+1} ait été

le premier sommet à avoir reçu la couleur q_1. Nous recolorons x_r avec la plus petite couleur possible qui soit plus grande que sa couleur courante, et continuons l'algorithme séquentiel avec $x_{r+1}, \ldots, x_n$. Si à un moment donné la couleur q doit être affectée à un sommet x_s (ce que nous voulons éviter, puisque l'on aimerait utiliser moins de q couleurs), nous remontons dans l'algorithme séquentiel jusqu'à x_{s-1} et faisons comme précédemment (c'est-à-dire recolorer x_{s-1} avec la plus petite couleur possible qui soit plus grande que sa couleur courante et continuer). L'algorithme se termine lorsque l'on est remonté jusqu'à x_1, et $\chi(G)$ est égal à la valeur courante de q.

Cette méthode peut être améliorée en certains points, pour donner l'algorithme appelé EPCOT [11] décrit ci-dessous. Ici les premiers sommets $x_1, \ldots, x_g$ sont choisis de manière à former une clique, et l'ordre des sommets restants est modifé dynamiquement sur la base du degré de saturation des sommets (pour un sommet x et une coloration partielle, le degré de saturation est le nombre de couleurs adjacentes à x). Une troisième adaptation tient compte de l'observation suivante : si c est la plus grande couleur utilisée parmi les sommets $x_1, \ldots, x_p$, il est inutile d'essayer les couleurs plus grandes que $c+1$ pour x_{p+1} (on peut toujours échanger deux couleurs dans une coloration).

L'algorithme EPCOT a comme entrée un graphe $G = (X, E)$ et fournit son nombre chromatique $\chi(G)$. Les variables utilisées sont le nombre de sommets colorés m, le sommet courant s, la cardinalité g de la clique initiale, et les quatre vecteurs `degres` (qui contient le degré de chaque sommet), `color` (la coloration (partielle) courante), `dsat` (qui contient le degré de saturation des sommets induit par la coloration courante) et `ordre` (où `ordre[`j`]` est le sommet qui a été coloré en j-ème position). L'algorithme est le suivant :

```
Algorithme EPCOT ;
begin

        initialiser degres[x] pour tout x ∈ V ;
        poser color[x] := 0 et dsat[x] := 0 pour tout x ∈ V ;
        q := borne supérieure de χ(G) ;
                              −−  obtenue p.ex. par un algorithme séquentiel
        déterminer une clique K maximale au sens de l'inclusion ;
        soit K = {k_1, ..., k_g} ;
        for i := 1 to g do
             color[k_i] := i ; ordre[i] := k_i ;
        end for ;
        mettre à jour dsat ;
        if g ≠ |X| then
             m := g + 1 ;
             chercher_prochain_s := vrai ;
             while m > g do
                  if chercher_prochain_s then
                       trouver un sommet non coloré s maximisant le
                         degré de saturation (en cas d'égalité, choisir
```

```
                    un sommet de plus grand degré (s'il y en a
                    plusieurs, choisir au hasard)) ;
                ordre[m] := s ; chercher_prochain_s := faux ;
            end if ;
            soit c_s la plus petite couleur possible pour s mais
              plus grande que color[s] ;
            if (c_s < q) and (c_s ≤ max_{i=1,...,m-1} color[ordre[i]] +1) then
                color[s] := c_s ; mettre à jour dsat ;
                if m = |X| then
                          -- une meilleure coloration a été obtenue
                    q := max_{s∈X} color[s] ;
                    m := 1 ;
                    while color[ordre[m]] < q do m := m + 1 ;
                       -- on cherche le premier sommet de couleur q
                    for any i ≥ m (i ≤ |X|) do color[ordre[i]] := 0 ;
                      -- on remonte jusqu'à ce sommet en décolorant
                    mettre à jour dsat ; m := m − 1 ; s := ordre[m] ;
                else
                    m := m + 1 ; chercher_prochain_s := vrai ;
                end if
            else
                color[s] := 0 ; m := m + 1 ; s := ordre[m] ;
                                          -- on remonte d'un sommet
            end if ;
        end while ;
    end if ;
    χ(G) := q ;
end algorithme.
```

Par "une clique maximale au sens de l'inclusion", on entend une clique K telle qu'il n'existe pas de clique plus grande K' contenant entièrement K. Une telle clique est facile à trouver puisqu'il suffit de partir d'une clique quelconque, et de rajouter un sommet adjacent à tous les sommets de la clique courante tant qu'un tel sommet existe. L'utilisation d'une méthode plus sophistiquée peut permettre de trouver des cliques plus grandes, et de ce fait de réduire un peu le temps d'exécution d'EPCOT.

5. Colorations par la méthode Tabou

À cause du temps de calcul, il n'est pas possible d'utiliser un algorithme exact pour déterminer le nombre chromatique d'un graphe de taille raisonnable ; il faut se contenter d'en trouver une borne supérieure. Si les algorithmes séquentiels basés sur des ordres ont l'avantage de la vitesse, ils fournissent en général une borne supérieure très large pour les graphes ayant plus de quelques centaines de sommets. C'est pourquoi des algorithmes heuristiques plus efficaces ont été proposés. Celui

que nous décrivons ici est l'adaptation de la méthode Tabou (décrite plus en détails dans [16]) au problème de la détermination d'une k-coloration.

Dans cet algorithme, une solution est simplement une partition $s = (X_1, \ldots, X_k)$ de l'ensemble des sommets X du graphe $G = (X, E)$ à colorer. Une telle partition est une k-coloration si et seulement si aucune arête n'a ses deux extrémités dans un même sous-ensemble X_i. C'est pourquoi nous définissons la fonction-objectif f suivante :

$$f(s) \quad = \quad \text{nombre d'arêtes } [x, y] \text{ telles que } x \text{ et } y \text{ sont dans un même sous-ensemble de la partition } s$$

et cherchons à la minimiser. Nous avons obtenu une k-coloration dès que nous avons trouvé une solution s avec $f(s) = 0$.

Afin d'expliquer le fonctionnement de l'algorithme, nous définissons encore le voisinage $N(s)$ d'une solution s comme étant l'ensemble des solutions s' que l'on peut obtenir à partir de s en déplaçant un sommet x d'un certain sous-ensemble X_i (où x est adjacent à au moins un sommet de X_i) vers un autre sous-ensemble X_j.

La méthode Tabou part d'une certaine solution initiale, obtenue aléatoirement par exemple. À chaque fois que l'on est en une solution s, nous générons aléatoirement un sous-ensemble M de $N(s)$ où $|M|$ est un paramètre de l'algorithme. La prochaine solution courante est la meilleure solution dans M. Comme cette procédure peut cycler, une *liste tabou* T doit être introduite. Le passage d'une solution s vers une solution s' dans $N(s)$ consistant en le déplacement d'un sommet x d'un ensemble X_i vers un ensemble X_j, nous introduisons dans T la paire (x, i), dans le but d'interdire (temporairement) le retour de x dans le i-ème sous-ensemble. Lors de la génération de M, nous ne choisissons donc que des solutions $s' = (X'_1, \ldots, X'_k)$ de $N(s)$ pour lesquelles $x \notin X'_i$ pour toute paire (x, i) dans T. La taille maximale de la liste tabou étant souvent fixée à une certaine valeur, le plus vieil élément de T est supprimé lorsque T devient trop grand.

Cet algorithme ayant la possibilité de détériorer la solution courante, il est nécessaire de stocker la meilleure solution rencontrée dans une mémoire s^* afin de ne pas la perdre. Cette solution permet aussi, lors de la génération de M, d'autoriser une solution s' avec $x \in X'_i$, malgré la présence de la paire (x, i) dans T, sous la condition que $f(s') < f(s^*)$ (pas de risque de cyclage dans ce cas).

Le principal critère d'arrêt de la méthode Tabou est la découverte d'une solution s avec $f(s) = 0$, c'est-à-dire d'une k-coloration. On peut alors diminuer la valeur de k d'une unité et appliquer à nouveau la méthode Tabou, afin de tenter d'obtenir une meilleure borne supérieure sur le nombre chromatique. Mais ce critère d'arrêt ne permet pas d'assurer que l'algorithme s'arrête ; en particulier si $k < \chi(G)$, il n'est pas suffisant. Il faut donc introduire un ou plusieurs autres critères d'arrêt pour interrompre la recherche. Les plus fréquents sont :

- un nombre fixé d'itérations a été effectué ;
- il n'y a plus eu d'amélioration de s^* depuis un nombre fixé d'itérations ;
- le temps de calcul a atteint une limite fixée.

Nous obtenons ainsi la méthode Tabou pour le problème de la k-coloration décrit ci-dessous. Notons qu'afin d'accélérer un peu l'aglorithme, la génération de M peut être interrompue dès que s' avec $f(s') < f(s)$ a été trouvé.

```
Algorithme Tabou ;
begin
    choisir une solution initiale s ;
    s* := s ; T = ∅ ;
    while aucun critère d'arrêt n'est satisfait do
         générer M avec |M| solutions s' = (X'_1, ..., X'_k) ∈ N(s)
           telles que x ∉ X'_i pour toute paire (x, i) dans T
           ou f(s') < f(s*) ;
                --  dès qu'un s' avec f(s') < f(s) est trouvé,
                --  arrêter la génération
         s := meilleure solution dans M ;
         mettre à jour T ;
         if f(s) < f(s*) then s* := s ;
    end while ;
end algorithme.
```

Il est important, pour la rapidité de l'algorithme, de noter que la valeur $f(s')$ d'une solution voisine de s doit être calculée sur la base de la valeur $f(s)$ connue. En effet, considérons que le passage de $s = (X_1, \ldots, X_k)$ à s' consiste à déplacer un sommet x du i-ème au j-ème sous-ensemble. Alors

$$f(s') = f(s) - |N_G(x) \cap X_i| + |N_G(x) \cap X_j|.$$

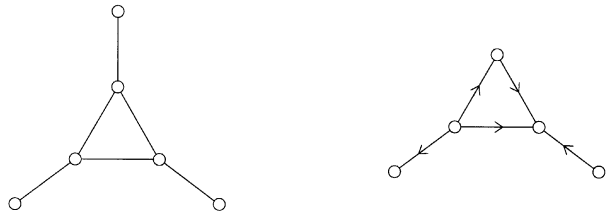
Il n'est donc même pas nécessaire, à ce stade, de générer s' ; x, i et j sont suffisants. On peut se contenter de stocker ces trois valeurs et de les mettre à jour à chaque fois qu'une meilleure solution voisine est trouvée au cours de la génération de M.

Cet algorithme permet obtenir, en quelques minutes de temps de calcul, de meilleurs bornes supérieures que les algorithmes séquentiels. Des expériences numériques sont décrites dans [13]. Pour des graphes aléatoires de 500 sommets et plus (densité $0,5$), il est nécessaire de d'abord réduire le graphe initial à un graphe résiduel de 100 à 200 sommets avant d'appliquer l'algorithme présenté ci-dessus. Cette réduction se fait en trouvant de manière répétée un grand ensemble stable (avec un algorithme Tabou adapté), et en l'enlevant ensuite du graphe (cela correspond à affecter une couleur de manière définitive à cet ensemble stable) [13].

Un autre avantage de l'algorithme Tabou provient du fait qu'il est relativement aisé de l'adapter à divers problèmes de coloration de graphes, tels que ceux que nous verrons dans la suite de cet article.

6. Quelques mots sur les graphes parfaits

Alors que les problèmes de coloration sont généralement difficiles pour des graphes quelconques, la classe des graphes parfaits a la particularité de comprendre



a) un graphe qui n'est pas un graphe de comparabilité.

b) un graphe de comparabilité (avec une orientation transitive de ses arêtes).

FIGURE 4. Un graphe de comparabilité et un graphe qui ne l'est pas.

des graphes pour lesquels le nombre chromatique peut être calculé en temps polynomial [18]. Dans les cas les plus généraux de graphes parfaits, les algorithmes polynomiaux connus sont des algorithmes généraux de programmation linéaire; nous nous limiterons ici à considérer des classes de graphes pour lesquels sont connus des algorithmes de type combinatoire.

Un graphe simple $G = (X, E)$ est *parfait* si pour tout sous-graphe induit H de G on a $\chi(H) = \omega(H)$ c'est-à-dire que le nombre chromatique de H est égal à la cardinalité maximum $\omega(H)$ d'une clique de H.

Puisque le graphe complémentaire $\overline{G}$ de G (c.-à-d. le graphe obtenu à partir de G en éliminant toutes les arêtes de G et en rajoutant toutes celles qui manquaient dans G) est parfait si et seulement si G est parfait (voir [3]), nous en déduisons qu'un graphe G est parfait s'il vérifie $\alpha(H) = \theta(H)$ pour tout sous-graphe induit H de G (c'est-à-dire que le nombre de stabilité $\alpha(H)$ de H est égal au nombre minimum $\theta(H)$ de cliques de H recouvrant l'ensemble des sommets de H).

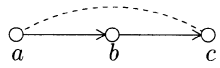
Nous renvoyons le lecteur intéressé aux ouvrages traitant plus spécifiquement les graphes parfaits [3, 5, 17, 18] et nous nous limitons ici à décrire quelques sous-classes de graphes parfaits en mettant en évidence leurs propriétés algorithmiques et chromatiques.

Les graphes *de comparabilité* sont les graphes simples $G = (X, E)$ tels que l'on peut donner une orientation à chaque arête de manière à obtenir le graphe orienté associé à une relation transitive et antisymétrique. Un exemple de graphe de comparabilité et d'un graphe qui ne l'est pas est donné dans la figure 4.

À titre d'exemple les graphes bipartis $G = (X, Y, E)$ sont des graphes de comparabilité : on le voit en orientant toute arête $[x, y]$ avec $x \in X, y \in Y$ de x vers y.

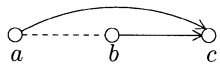
De manière générale, les graphes de comparabilité sont donc des graphes $G = (X, E)$ dans lesquels on peut trouver une orientation ne contenant pas de circuit et pas la configuration (a, b), (b, c) représentée à la figure 5, avec $a < b < c$ et $[a, c] \notin E$.

Pour ces graphes l'algorithme de coloration est particulièrement simple une fois que l'orientation est définie : s'agissant d'un graphe sans circuits, on peut classer ses sommets x dans l'ordre de leur rang $r(x)$ non décroissant (le *rang* $r(x)$



$[a, c]$ n'existe pas dans G

FIGURE 5. Configuration interdite dans une orientation transitive.



$[a, b]$ n'existe pas dans G

FIGURE 6. Configuration interdite dans l'orientation associée à un graphe triangulé.

d'un sommet x est le nombre maximum de sommets rencontrés sur un chemin aboutissant au sommet x).

On peut ensuite colorer les sommets avec un algorithme séquentiel usuel, en donnant la plus petite couleur possible à chaque sommet (en fait, chaque sommet x va recevoir la couleur $r(x)$). Il est facile de vérifier que $\chi(G) = \omega(G)$ puisque le nombre de couleurs utilisées dans cette coloration est égal au nombre de sommets rencontrés sur un plus long chemin et que ce chemin induit une clique dans le graphe par la transitivité de l'orientation. Pour reconnaître si un graphe G est de comparabilité, on dispose d'algorithmes (polynomiaux) d'orientation des arêtes par propagation (en évitant la configuration de la Fig. 5) (voir [17]).

Les graphes *triangulés* sont tels que tout cycle de longueur supérieure ou égale à 4 admet une corde (c'est-à-dire une arête reliant deux sommets non consécutifs du cycle).

Ces graphes sont caractérisés par la propriété suivante :

Propriété 4. *G est triangulé si et seulement si tout sous-graphe induit admet un sommet simplicial (c'est-à-dire tel que tous ses voisins forment une clique).*

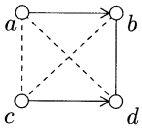
En utilisant cette propriété, nous pouvons construire un ordre $y_1, y_2, ..., y_n$ des n sommets d'un graphe $G = (X, E)$ triangulé : soit $G(i)$ le sous-graphe induit de G engendré par $X - \{y_n, y_{n-1}, ..., y_{i+1}\}$; on a $G(n) = G$;

pour $i = n$ à 1 faire :
choisir pour y_i un sommet x simplicial dans $G(i)$.

Si l'on oriente ensuite toutes les arêtes $[y_i, y_j]$ de y_i vers y_j si $i < j$, alors on obtient une orientation ne contenant pas de circuits et pas non plus la configuration $(a, c), (b, c)$ de la figure 6 (avec $a < b < c$ et $[a, b] \notin E$).

Une coloration du graphe G est obtenue par un algorithme séquentiel (basé sur l'ordre $y_1, ..., y_n$) ; il est facile de voir que cette coloration utilise un nombre minimum de couleurs : si un sommet u a reçu la plus grande couleur k, c'est qu'il a au moins un sommet v de couleur $i < k$ qui le précède dans l'ordre (pour tout $i < k$) ; puisque la configuration de la figure 6 est interdite, tous les prédécesseurs de u sont reliés deux à deux par des arêtes et ils forment donc une clique (de k

(a,b), (c,d), (b,d) ou (d,b) avec $a<b$, $c<d$ et $[a,c]$, $[a,d]$, $[b,c] \notin E$



$[a,c]$, $[a,d]$ et $[b,c]$ n'existent pas

FIGURE 7. Configuration interdite dans l'orientation associée à un graphe parfaitement ordonnable.

sommets si l'on y inclut le sommet u lui-même). Nous avons ainsi obtenu une coloration où le nombre k de couleurs utilisées est égale à la taille k d'une clique de G. Le nombre k de couleurs est ainsi minimum et la taille k de la clique est maximum.

Ces deux classes bien connues de graphes parfaits sont inclues dans la famille des graphes *parfaitement ordonnables* [8] : il s'agit des graphes $G=(X,E)$ pour lesquels on peut trouver un ordre $<$ des sommets (et donc une orientation sans circuits pour l'ensemble E de ses arêtes) de manière que la configuration de la figure 7 soit interdite :
En d'autres termes, le graphe orienté ne peut contenir de chaîne induite sur 4 sommets a,b,c,d (voir Fig. 7) où $a<b$ et $c<d$.

L'orientation définie pour les graphes de comparabilité ne contient pas l'obstruction de la figure 7 (puisque l'obstruction de la Fig. 5 est contenue dans l'obstruction de la Fig. 7) ; les graphes de comparabilité sont donc des graphes parfaitement ordonnables. Le graphe de la Fig. 4a) est parfaitement ordonnable, mais n'est pas un graphe de comparabilité.

De même, l'orientation définie pour les graphes triangulés ne peut contenir l'obstruction de la figure 7 (puisque l'obstruction de la Fig. 6 est contenue dans celle de la Fig. 7), ce qui montre que les graphes triangulés sont parfaitement ordonnables. Le graphe consistant en les arêtes $[a,b]$, $[b,c]$, $[c,d]$, $[a,d]$ est parfaitement ordonnable mais n'est pas triangulé.

Pour les graphes parfaitement ordonnables, tout ordre des sommets tel que l'orientation associée ne comprend pas l'obstruction de la figure 7 peut être utilisé dans l'algorithme séquentiel de coloration. On peut montrer que l'on obtient encore une coloration utilisant k couleurs si k est la taille maximum d'une clique de G [8].

La classe des graphes parfaitement ordonnables est intéressante pour la coloration en raison de la simplicité de l'algorithme exact de coloration. Il est malheureusement difficile de reconnaître si un graphe est parfaitement ordonnable ; le problème est en effet NP-complet [25]. Nous devons ainsi nous limiter à reconnaître en temps polynomial des sous-classes des graphes parfaitement ordonnables. D'autres sous-classes de graphes parfaits sont intéressantes à mentionner en raison de leur champ d'application. Parmi ceux-ci les graphes d'intervalle : on dit qu'un graphe simple $G=(X,E)$ est un graphe *d'intervalle* s'il existe une famille $\mathfrak{X}=\{x_1,...,x_n\}$ d'intervalles sur la droite telle qu'en associant un sommet x_i

de G à chaque intervalle x_i de $\mathfrak{X}$, on ait $[x_i, x_j] \in E$ si et seulement si les intervalles correspondants ont une intersection non vide.

Propriété 5. *(voir [3]). Un graphe G est un graphe d'intervalles si et seulement si G est triangulé et son complémentaire $\overline{G}$ est un graphe de comparabilité.*

Ainsi puisqu'un tel graphe est triangulé, on peut construire un ordre des sommets $y_1, ..., y_n$ de manière à pouvoir utiliser l'algorithme séquentiel classique de coloration. Ici, si l'on note $x_i = [a_i, b_i]$ les intervalles de $\mathfrak{X}$, l'ordre s'obtient en classant les intervalles dans l'ordre des extrémités (limite droite) b_i non croissantes et on les colore dans cet ordre avec la plus petite couleur possible.

À titre d'example, les graphes d'intervalles peuvent servir à modéliser des problèmes d'occupation de mémoires en programmation. Supposons en effet que soient connus dans l'exécution d'un programme les intervalles de temps séparant la première et la dernière utilisation de chacune des variables. Sachant que l'on pourra stocker dans la même mémoire (ou le même registre) des variables dont les intervalles de vie sont disjoints, minimiser le nombre de mémoires (ou registres) nécessaires, c'est chercher une coloration optimale (c'est-à-dire avec un nombre minimum de couleurs) dans le graphe d'intervalles associé aux variables ; les intervalles de couleur i seront associés aux variables qui pourront être stockées dans la mémoire i.

En réalité, ce type de problèmes revêt une grande importance dans les programmes comportant des boucles ; on doit alors utiliser des graphes d'intervalles circulaires qui sont associés à des intervalles situés sur un cercle au lieu d'une droite. Ces graphes ne sont plus parfaits et leur coloration est un problème qui est généralement difficile (voir [17]).

Des applications et des développements pour le problème des boucles dans des programmes informatiques sont mentionnés dans [34]. Les graphes d'intervalles circulaires sont aussi un outil permettant d'aménager le fonctionnement des feux de circulation aux carrefours routiers (voir Chap. 3 dans [27]) : l'optimisation des phases "vertes" pour les divers flux de trafic s'effectue en résolvant un (ou des) problème(s) de programmation linéaire dont la structure est basée sur la construction d'un graphe d'intervalles circulaires.

Enfin, une classe très particulière de graphes parfaits mérite d'être mentionnée : il s'agit des graphes *de permutation* [17].

Soit $P = (p_1, p_2, ..., p_n)$ une permutation des entiers $1, 2, ..., n$; on lui associe un graphe $G(P)$ en introduisant des sommets $1, 2, ..., n$ et en reliant i et j par une arête si $i < j$ et si j précède i dans P. Ceci signifie que, si $p^{-1}(\ell)$ désigne la place occupée par ℓ dans la permutation ($p^{-1}(\ell) = k$ si $p_k = \ell$), alors i et j sont reliés si $i < j$ et $p^{-1}(j) < p^{-1}(i)$. La figure 8 donne une permutation P et le graphe associé $G(P)$.

Un graphe $G = (X, E)$ est appelé graphe *de permutation* s'il existe une permutation P de $1, 2, ..., |X|$ telle que (en rebaptisant $1, 2, ..., n$ les sommets) G est le graphe $G(P)$ associé à P.

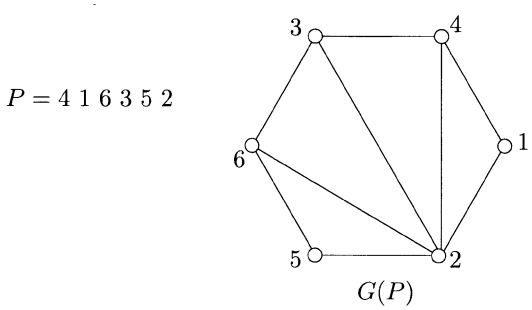


FIGURE 8. Une permutation P et le graphe associé $G(P)$.

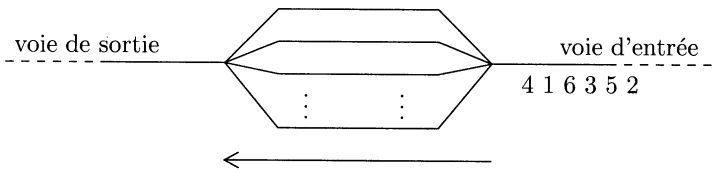


FIGURE 9. Un problème de gare de triage.

Propriété 6. *(voir [26]). Un graphe simple G est de permutation si et seulement si G et $\overline{G}$ sont des graphes de comparabilité.*

Cette propriété est à la base d'algorithmes de reconnaissance des graphes de permutation par construction d'orientations transitives ; on en trouvera dans [17].

Si P est une permutation, les sous-suites décroissantes de P correspondent aux cliques de $G(P)$ et les sous-suites croissantes aux ensembles stables de $G(P)$. Puisque les graphes de permutation sont parfaits, on a alors égalité entre le nombre chromatique $\chi(G(P))$, le nombre minimum de sous-suites croissantes qui recouvrent l'ensemble $\{1, 2, ..., n\}$ et la longueur maximum d'une sous-suite décroissante.

À titre d'application, rappelons le problème de la gare de triage (inspiré de [17]). Dans la figure 9, des wagons sont alignés sur la voie d'entrée (à droite) de la gare de triage ; ils sont désignés par le numéro i $(1 \leq i \leq n)$ de leur destination. Ces destinations sont localisées à gauche de la gare de triage le long de la voie de sortie, les numéros croissants de gauche à droite. Lorsqu'un train arrive dans une gare de destination i, on décroche éventuellement le wagon de queue (s'il porte le numéro i) et le train poursuit son trajet vers la gare $i - 1$.

Il s'agit donc d'envoyer les n wagons de la voie d'entrée sur les voies de triage où l'on va constituer des trains ayant chacun ses wagons numérotés dans l'ordre croissant. On cherche en général à utiliser le nombre minimum de voies de triage, c'est-à-dire à constituer le nombre minimum de trains "croissants".

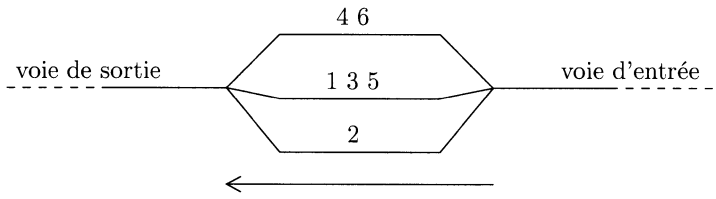


FIGURE 10. Le triage des wagons du problème de la figure 9.

L'algorithme consiste à prendre successivement les n wagons numérotés p_1, p_2, ..., p_n et à envoyer chaque wagon p_i sur la première voie admissible (c'est-à-dire ne contenant pas un dernier wagon portant un numéro $\ell > p_i$).

L'algorithme pourrait se formuler ainsi :

```
Algorithme gare_de_triage ;
begin
    k := 0 ;                          --  compteur de voies utilisées
    for i := 1 to n do
        soit j le numéro de la première voie admissible ;
        placer le wagon p_i sur la voie j ;
                    --  c'est-à-dire colorer le sommet p_i
                    --  de G(P) avec la couleur j
        le dernier wagon de la voie j est p_i ;
        k := max(k, j) ;  --  mise à jour du compteur de voies
    end for ;
end algorithme.
```

La figure 10 donne l'application de cet algorithme à l'exemple de la permutation $P = (416352)$ de la Fig. 9.

On a donc cherché le nombre minimum de sous-séquences croissantes de P qui recouvre l'ensemble $\{1, 2, ..., n\}$; c'est une partition de l'ensemble $\{1, 2, , ..., n\}$ de $G(P)$ en ensembles stables. On peut vérifier que l'algorithme ci-dessus donne une coloration en $\chi(G(P))$ couleurs, c'est-à-dire qu'il met en évidence une sous-suite décroissante de $k = \chi(G(P))$ nombres : on la construit facilement en partant du dernier sommet coloré avec la couleur k.

On peut aussi vérifier que si l'on n'utilise pas à chaque étape la première voie admissible, alors on risque d'utiliser trop de voies : si l'on plaçait par exemple le wagon 6 sur la voie 2 au lieu de la voie 1, on aurait besoin de 4 voies au lieu de 3 !

7. Ordonnancement chromatique et colorations

Formulé en termes de colorations de graphe, le problème d'ordonnancement donné au début de cet article fait partie du domaine de l'*ordonnancement chromatique* (en anglais : chromatic scheduling). Il s'agit là de l'ensemble des problèmes d'ordonnancement ou d'emploi du temps qui sont susceptibles d'être modélisés (et

résolus) à l'aide des concepts de coloration et de leurs extensions. Si l'ordonnancement chromatique se limitait en effet aux seuls problèmes solubles par des concepts classiques de coloration, le domaine serait par trop limité et bien des problèmes d'ordonnancement en seraient exclus.

C'est la raison pour laquelle nous allons présenter ici quelques extensions et variations des concepts de coloration qui ont été introduits précisément dans l'objectif d'appréhender un champ plus large de problèmes d'emploi du temps.

Considérons un problème d'ordonnancement caractérisé par un ensemble X de tâches $x_1, ..., x_n$ indivisibles et de même durée (soit un jour) ; à ceci s'ajoutent des contraintes de *précédence* sur l'exécution des tâches, c'est-à-dire un ensemble U de couples ordonnés (x_i, x_j) de tâches signifiant que pour chaque couple (x_i, x_j) la tâche x_i doit précéder la tâche x_j (c'est-à-dire que x_i doit être attribuée à un jour $c(i)$ tel que $c(i) < c(j)$ si $c(j)$ est le jour où x_j est exécutée). Puis, comme précédemment, nous avons une collection E de paires $[x_i, x_j]$ non ordonnées indiquant que les tâches x_i et x_j ne peuvent être exécutées simultanément (contraintes de disjonction).

Il s'agit alors de déterminer une affectation des tâches à des jours en respectant les contraintes définies par les ensembles U et E. L'existence d'un ordonnancement en k jours ou la recherche d'un ordonnancement de durée minimum sont des questions que l'on peut se poser.

À ce problème d'ordonnancement, nous associons un graphe *mixte*, c'est-à-dire comportant à la fois des arêtes (non orientées) et des arcs (orientés par définition) : chaque tâche x_i correspond à un sommet de G ; chaque contrainte (x_i, x_j) de précédence est associée à un arc (x_i, x_j) et, comme précédemment, chaque contrainte de disjonction est représentée par une arête $[x_i, x_j]$. Un tel graphe sera noté $G_M = (X, U, E)$.

Nous pouvons alors définir une *k-coloration* d'un graphe mixte $G_M = (X, U, E)$ comme une affectation à chaque sommet i d'une couleur $c(i)$ choisie dans l'ensemble (ordonné) $\{1, ..., k\}$ de manière que pour chaque arc (x_i, x_j) de U, on ait $c(j) > c(i)$ et pour chaque arête $[x_i, x_j]$ de E, on ait $c(i) \neq c(j)$. On parle de même de *coloration* (si la valeur de k n'est pas précisée).

Notons que la notion de k-coloration d'un graphe mixte généralise de façon naturelle la notion de k-coloration d'un graphe simple.

Il faut souligner que, contrairement au cas classique, il peut n'exister de k-coloration pour aucune valeur de k. Nous avons en effet la condition suivante :

Propriété 7. *Un graphe $G_M = (X, U, E)$ admet une coloration si et seulement si le graphe partiel orienté $G = (X, U)$ ne contient aucun circuit.*

La justification repose sur le fait qu'on peut toujours orienter les arêtes de E de manière à obtenir avec les arcs de U un graphe orienté ne comportant aucun circuit.

Remarquons encore qu'une contrainte de disjonction (représentée par une arête $[x_i, x_j]$) peut s'interpréter en disant que l'on doit avoir soit $c(i) < c(j)$ (arc (x_i, x_j) dans le graphe) soit $c(j) < c(i)$ (arc (x_j, x_i) dans le graphe). Ceci implique $c(j) \neq c(j)$.

Nous noterons $\chi_M(G_M)$ et appellerons *nombre chromatique mixte* (d'un graphe mixte G_M) le plus petit k tel que $G_M = (X, U, E)$ a une k-coloration.

Il est clair que $\chi_M(G_M)$ est toujours supérieur ou égal au nombre maximum de sommets que l'on rencontre sur un chemin (formé par définition d'arcs (orientés)) dans $G = (X, U)$; en notant $\ell(U)$ ce nombre, nous avons $\chi_M(G_M = (X, U, E)) \geq \ell(U)$.

Diverses bornes de $\chi_M(G_M)$ sont données dans [21]. On montre notamment que pour un graphe $G_M = (X, U, E)$ biparti, on a $\ell(U) \leq \chi_M(G_M) \leq \ell(U) + 1$. Alors que pour un arbre mixte $T_M = (X, U, E)$, il est possible de trouver en temps polynomial si $\chi(T_M) = \ell(U)$, en revanche pour un graphe mixte biparti, le problème est NP-complet (ceci est une conséquence de résultats donnés dans [7]).

Quelques expériences avec un algorithme heuristique sont relatées dans [21] ; mais il y a encore des directions nombreuses à explorer dans le but d'élaborer des algorithmes efficaces pour des problèmes de grande taille.

8. Colorations par intervalles

Jusqu'ici les problèmes d'ordonnancement qui pouvaient être modélisés au moyen des concepts de coloration avec leurs extensions se limitaient à envisager des ensembles de tâches de même durée d'exécution. Si cette restriction est acceptable en particulier dans le cadre de problèmes d'horaires scolaires (où chaque tâche représente une leçon d'une heure), il faut bien admettre que dans la plupart des applications les tâches x_i que l'on manipule ont des durées d_i qui peuvent être différentes les unes des autres mais que l'on supposera être des nombres entiers (de jours ou d'heures par exemple). Nous continuerons à supposer ici que les tâches sont indivisibles, c'est-à-dire qu'une tâche doit toujours être exécutée sans interruptions.

Nous considérerons à ce stade des graphes mixtes $G_M = (X, U, E)$ où à chaque sommet x_i est associé un entier positif d_i qui représente la durée de la tâche x_i correspondant à ce sommet. Un tel graphe sera ainsi noté $G_M^D = (X, U, E, D)$ où D est l'affectation d'entiers d_i aux sommets.

Les contraintes de précédence de U impliquent ainsi que si l'on a un arc (x_i, x_j) la tâche x_i (débutant le jour $c(i)$ et se terminant à la fin du jour $c(i) + d_i - 1$) soit alors achevée avant le jour $c(j)$ où débute la tâche x_j, c'est-à-dire que l'on doit avoir

$$c(j) \geq c(i) + d_i.$$

Quant aux contraintes de disjonction, elles imposent pour chaque arête $[x_i, x_j]$ de E que l'on ait l'une des deux possibilités

$$c(i) + d_i \leq c(j) \quad \text{ou} \quad c(j) + d_j \leq c(i).$$

Une *k-coloration par intervalles* d'un graphe mixte $G_M^D = (X, U, E, D)$ est une affectation à chaque sommet x_i de d_i entiers consécutifs $c(i), c(i)+1, ..., c(i)+d_i-1$ de manière que :

a) $c(j) \geq c(i) + d_i$ pour tout arc (x_i, x_j) de U ;
b) $c(i) + d_i \leq c(j)$ ou $c(j) + d_j \leq c(i)$ pour toute arête $[x_i, x_j]$ de E.

Le *nombre chromatique par intervalle* $\chi_M^D(G_M^D)$ de G_M^D est le plus petit entier k tel que G_M^D a une k-coloration par intervalles. Etant une extension du problème classique de coloration, il est clair que ce problème est en général difficile.

Dans le but de donner des bornes de χ_M^D sous une forme relativement simple, nous supposerons maintenant que nous n'avons pas de contraintes de précédence ($U = \emptyset$). Nous noterons $N_G(x_i)$ l'ensemble des voisins du sommet x_i dans le graphe G. Et $d_G(x_i)$ sera le degré du sommet x_i dans le graphe G (c'est-à-dire le nombre d'arêtes adjacentes à x_i).

Propriété 8. *(voir [36]). Pour tout graphe G_M^D de sommets $x_1, ..., x_n$*

$$\chi_M^D(G_M^D) \leq \max_{1 \leq i \leq n} \left(d_i + \sum_{x_j \in N_{G_M^D}(x_i)} d_j \right).$$

Avec des techniques qui généralisent très directement les méthodes utilisées pour les colorations classiques des graphes, nous pouvons obtenir une seconde forme.

Propriété 9. *(voir [36]).*

$$\chi_M^D(G_M^D) \leq 1 + \max_{H \subseteq G_M^D} \min_{x_i \in H} \left[(d_H(x_i) + 1)(d_i - 1) + \sum_{x_j \in N_H(x_i)} d_j \right].$$

Ici $H \subseteq G_M^D$ signifie que H est un sous-graphe induit de G_M^D et $x_i \in H$ signifie que x_i est un sommet du graphe H.

Un algorithme séquentiel de coloration peut être défini pour colorer un graphe mixte au sens ci-dessus de façon à ce que le nombre de couleurs utilisées ne dépasse pas la valeur donnée dans la propriété 9. Il va consister à construire, en commençant par la dernière position, un ordre $y_1, ..., y_n$ des n sommets $x_1, ..., x_n$ de G_M^D.

```
Algorithme ;
begin
    H = G_M^D ;  i = 1 ;
    while H contient encore des sommets do
         choisir un sommet x_p tel que

              (d_H(x_p) + 1)(d_p − 1) + Σ_{x_j ∈ N_H(x_p)} d_j

         est minimum ;
         placer x_p à la dernière position encore libre ;
                                          −− poser y_{n−i+1} := x_p
         éliminer x_p de H ;        −− c-à-d poser H := H − x_p
         i := i + 1 ;
    end while ;
end algorithme.
```

Si l'on colore ensuite les sommets dans l'ordre $y_1, y_2, ..., y_n$ en utilisant chaque fois les plus petites couleurs consécutives possibles, alors on peut vérifier que l'on obtient une coloration par intervalles satisfaisant la borne de la propriété 9.

Il convient enfin de noter que si le problème d'ordonnancement formulé au début de ce paragraphe ne comporte au contraire que des contraintes de précédence (et aucune contrainte de disjonction), alors il entre exactement dans le cadre des problèmes classiques d'ordonnancement où les méthodes de chemin critique permettent de trouver (en cherchant un chemin de longueur maximum dans un graphe approprié) un ordonnancement de durée minimum en temps polynomial.

9. T-COLORATIONS

Alors que les applications des méthodes de coloration à des problèmes d'emploi du temps sont les plus connues et les plus anciennes, on recourt depuis moins longtemps à ces mêmes techniques dans le domaine des télécommunications et en particulier pour des questions d'affectation de fréquences à des émetteurs. Nous nous proposons de généraliser encore les notions de coloration étudiées jusqu'ici afin d'être à même de formuler avec des concepts appropriés des problèmes d'affectation de fréquences tels qu'ils surviennent dans des situations où il s'agit d'éviter des interférences parasites.

Dans le problème de la coloration par intervalle, nous pouvons reprendre les deux types de contraintes (de précédence et de disjonction) en cherchant à les reformuler :

a) contrainte de précédence représentée par un arc (x_i, x_j). Ceci implique $c(j) \geq c(i) + d_i$ ou encore $c(j) - c(i) \geq d_i$.
À l'arc (x_i, x_j) nous pouvons associer un ensemble $\overline{T}_{ij}$ de valeurs autorisées pour la différence $c(j) - c(i)$. Ici, nous aurions donc $\overline{T}_{ij} = \{d_i, d_i + 1, ..., \}$. Il est d'usage de considérer plutôt le complément T_{ij} de $\overline{T}_{ij}$ et l'on aurait donc $T_{ij} = \{..., d_i - 2, d_i - 1\}$ avec l'exigence $c(j) - c(i) \notin T_{ij}$;

b) contrainte de disjonction représentée par une arête $[x_i, x_j]$. Par définition, nous devons avoir
   - soit $c(j) \geq c(i) + d_i$ c'est-à-dire $c(j) - c(i) \geq d_i$;
   - soit $c(i) \geq c(j) + d_j$ c'est-à-dire $c(j) - c(i) \leq d_j$.

   En d'autres termes, la valeur de $c(j) - c(i)$ ne doit pas être comprise entre $-d_j + 1$ et $d_i - 1$.
   Contrairement à ce qui a été pratiqué précédemment, nous pouvons associer un arc orienté (x_i, x_j) orienté arbitrairement de x_i à x_j (et non plus une arête $[x_i, x_j]$ non orientée par définition) à une contrainte de disjonction portant sur la paire de tâches x_i, x_j et définir $T_{ij} = \{-d_j + 1, -d_j + 2, ..., 0, ..., d_i - 2, d_i - 1\}$; on devra avoir $c(j) - c(i) \notin T_{ij}$ comme dans le cas de la contrainte de précédence.

Si nous avions choisi l'orientation opposée, c'est-à-dire (x_j, x_i) nous aurions dû définir $T_{ji} = \{-d_i + 1, ..., 0, ..., d_j - 1\}$; notons que $T_{ij} \neq T_{ji}$ si $d_i \neq d_j$.

Ainsi, nous obtenons une formulation semblable pour les deux types de contraintes (dans un cas l'ensemble T_{ij} est fini, dans l'autre, cet ensemble ne l'est pas !). L'ensemble des valeurs possibles pour $c(j) - c(i)$ est un intervalle pour les contraintes de précédence et c'est une collection de 2 intervalles disjoints pour les contraintes de disjonction.

Nous pouvons ainsi définir la notion de T-coloration pour un graphe G^D qui sera maintenant un graphe orienté $G^D = (X, U, \emptyset, D, T)$ où $\emptyset$ est l'ensemble vide qui traduit l'absence d'arêtes dans le graphe et où T représente la donnée pour chaque arc (x_i, x_j) de U d'un ensemble T_{ij} de valeurs interdites pour $c(j) - c(i)$.

Une *T-coloration (en k couleurs)* d'un graphe orienté $G^D = (X, U, \emptyset, D, T)$ est une affectation de d_i entiers consécutifs $c(i), c(i) + 1, ..., c(i) + d_i - 1$ (choisis dans $\{1, ..., k\}$) à chaque sommet x_i telle que pour tout arc (x_i, x_j) on ait $c(j) - c(i) \notin T_{ij}$.

Nous pouvons par analogie avec ce qui a été fait plus haut, définir $\chi_T^D(G^D)$, le *nombre T-chromatique* de G^D, comme le plus petit k pour lequel il existe une T-coloration de G^D en k couleurs. Avant de donner des bornes supérieures sur $\chi_T^D(G^D)$, nous voulons simplement revenir au cas des colorations de sommets les plus classiques. Pour les retrouver, nous devons poser $d_i = 1$ pour chaque sommet x_i (un sommet reçoit en effet *une* couleur) et $T_{ij} = \{0\}$ pour tout arc (x_i, x_j) de G^D, c'est-à-dire que $c(j) - c(i) \notin \{0\}$ ou encore $c(j) \neq c(i)$ pour tout arc (x_i, x_j).

Nous pouvons, à ce stade, considérer qu'un problème classique de coloration tel que formulé au début de cet article peut en fait être associé à un graphe $G^D = G = (X, U)$ orienté avec pour chaque arc (x_i, x_j) un ensemble $T_{ij} = \{0\}$ (en choisissant au lieu de l'arc (x_i, x_j) l'arc opposé (x_j, x_i), on aurait aussi $T_{ji} = \{0\}$, ce qui montre que l'orientation est arbitraire pour les colorations classiques).

Dans les problèmes d'affectation de fréquences, les ensembles T_{ij} de valeurs interdites pour $c(j) - c(i)$ sont symétriques par rapport à 0 ($-r \in T_{ij}$ si et seulement si $r \in T_{ij}$). On a donc $|c(j) - c(i)| \notin T_{ij}$ ce qui permet à nouveau de considérer le problème comme posé dans un graphe non orienté.

Il faut remarquer à nouveau que des T-colorations n'existent pas toujours dans des graphes orientés G^D : considérons par exemple les arcs (x_i, x_j) représentant des contraintes de précédence (T_{ij} est alors un ensemble de la forme $\{..., d_i - 2, d_i - 1\}$) ; il est alors nécessaire que le graphe partiel orienté engendré par les arcs associés à des contraintes de précédence ne comporte aucun circuit. Il est relativement facile de se persuader que cette condition est aussi suffisante pour qu'il existe une T-coloration.

Afin de pouvoir donner une borne supérieure du nombre T-chromatique, introduisons pour chaque arc (i, j) le nombre $g_{ij} = |T_{ij}| - d_i + 1$ et faisons l'hypothèse que chaque T_{ij} est un ensemble d'entiers consécutifs (comprenant la valeur 0) ; lorsque nous parlons d'un arc (x_i, x_j), nous sommes conscients que son orientation est arbitraire et qu'elle pourrait être inversée en remplaçant de plus T_{ij} par T_{ji}.

Lorsque nous considérerons l'ensemble $N_H(x_i)$ des sommets x_j adjacents à un sommet x_i dans un sous-graphe induit H de G, nous pourrons donc supposer que tous les arcs entre x_i et x_j sont orientés de x_i vers l'autre sommet, soit x_j. En suivant [35], nous avons alors la borne suivante pour le nombre T-chromatique.

Propriété 10. *(voir [35]). Pour un graphe $G^D = (X, U, \emptyset, D, T)$ avec des ensembles T_{ij} qui sont des ensembles d'entiers consécutifs comprenant la valeur* 0,

$$\chi_T^D(G^D) \leq 1 + \max_{H \subseteq G^D} \min_{x_i \in H} \left\{ (d_H(x_i)+1)(d_i - 1) + \sum_{x_j \in N_H(x_i)} g_{ij} \right\}.$$

Il est intéressant de remarquer que, même si cette borne paraît exiger que l'on considère tous les sous-graphes induits H de G^D, elle peut être calculée en temps polynomial [35] et qu'un algorithme séquentiel de coloration peut être utilisé pour obtenir une T-coloration satisfaisant la propriété 10.

Définissons la fonction $f(z, H)$ où z est un sommet du graphe H par :

$$f(z, H) = \sum_{y \in N_H(z)} g_{zy} \quad + (d_H(z)+1)(d_z - 1).$$

Nous allons définir un classement $y_1, y_2, ..., y_n$ des $|X| = n$ sommets $x_1, ..., x_n$ de G^D (en commençant par la fin). Notons $G(i)$ le sous-graphe induit de $G^D = (X, U, \emptyset, D, T)$ engendré par l'ensemble $X - \{y_{i+1}, ..., y_n\}$; on a $G(n) = G^D$;

pour $i = n$ à 1 faire :
soit y_i un sommet de $G(i)$ tel que $f(y_i, G(i)) = \min_{z \in G(i)} f(z, G(i))$.

En colorant ensuite les sommets dans l'ordre $y_1, ..., y_n$ (en utilisant chaque fois la plus petite première couleur possible), on obtient une T-coloration vérifiant la propriété 10.

Des expériences numériques avec divers algorithmes séquentiels sont décrites dans [35] ; ces méthodes sont des adaptations directes d'algorithmes connus pour les colorations classiques des graphes.

Dans les applications à des affectations de fréquence, il s'agit généralement d'attribuer à chaque site émetteur x_i (représenté par un sommet x_i du graphe associé) une fréquence d'émission de manière à éviter des interférences avec des émetteurs voisins (représentés par des sommets x_j adjacents au sommet x_i). Ces interférences se produisent si les fréquences $c(i)$ et $c(j)$ attribuées respectivement aux émetteurs voisins x_i et x_j satisfont une relation

$$c(j) - c(i) \in T_{ij}.$$

T_{ij} est donc un ensemble de valeurs interdites pour la différence des fréquences des émetteurs voisins x_i et x_j (comme mentionné plus haut, on a en général $T_{ij} = T_{ji}$ et le problème peut être considéré comme non orienté).

Dans ces types d'application on a $|T_{ij}| < \infty$; l'objectif peut être soit de minimiser le nombre de fréquences différentes utilisées dans le réseau modélisé par le graphe G^D, soit la différence entre la plus grande et la plus petite fréquence utilisée (largeur de bande). Les algorithmes mentionnés ci-dessus sont adaptés au premier objectif. Des méthodes adéquates pourraient être élaborées pour tenter de minimiser la largeur de la bande.

10. Colorations restreintes

Les contraintes disjonctives et les contraintes de précédence apparaissant dans les problèmes d'ordonnancement chromatique peuvent être considérées comme similaires dans le sens où ce sont des contraintes locales, concernant des sous-ensembles spécifiques de sommets (en l'occurrence des paires de sommets). Dans les problèmes de confection d'horaires apparaissent souvent d'autres contraintes locales, chacune d'elles ne concernant qu'un sommet. Ces contraintes restreignent les couleurs qui peuvent être affectées à chaque sommet.

Nous considérons ici un graphe sans contraintes de précédence $G = (X, E)$ où à chaque sommet x est associé un ensemble $\varphi(x) \subseteq \{1, \ldots, k\}$ de couleurs admissibles pour ce sommet (où k est le nombre de couleurs à disposition). Un tel graphe sera noté $G^\varphi = (X, E, \varphi)$. Le problème de *$k$-coloration restreinte* consiste à déterminer s'il exsite une k-coloration c de G telle que chaque sommet x reçoive une couleur admissible $c(x) \in \varphi(x)$. On parle simplement de *coloration restreinte* si le nombre de couleurs k n'est pas spécifié (dans la littérature anglophone, on parle plutôt de "list coloring"). Ce problème peut facilement être étendu aux graphes mixtes, mais cette extension ne sera pas abordée ici.

Dans les problèmes d'horaires scolaires, ce type de contraintes permet par exemple de tenir compte de l'indisponibilité d'enseignants à certaines heures. Si un maître n'est pas disponible pendant l'heure i, cela peut être modélisé en supprimant la couleur i de l'ensemble des couleurs admissibles pour les cours de ce maître. Pour les problèmes d'ordonnancement de tâches, nous pouvons de cette manière prendre en compte l'interdiction d'exécuter certaines tâches un jour donné.

Comme nous pouvons supposer que k est borné polynomialement par le nombre de sommets n (les sommets x pour lesquels $|\varphi(x)| \geq n$ peuvent être supprimés du graphe sans influer sur l'existence de la k-coloration cherchée), le problème de la k-coloration restreinte a une complexité au moins aussi grande que le problème de la k-coloration. En fait, ce problème est même NP-complet pour les graphes bipartis planaires avec $k = 3$.

Il existe également un lien entre le problème de la k-coloration restreinte et un problème d'orientation des arêtes, mais il est moins fort que celui de la propriété 1.

Propriété 11. *(voir [28]). Si l'on peut orienter chaque arête d'un graphe G^φ de manière à obtenir un graphe orienté sans circuits de longueur impaire et avec, pour tout sommet x, au plus $|\varphi(x)| - 1$ arcs quittant x, alors G^φ admet une coloration restreinte.*

Avec une condition supplémentaire, ce résultat peut être étendu en autorisant la présence de circuits de longueur impaire, mais l'existence d'une telle orientation reste simplement une condition suffisante (et non nécessaire) [1].

Une autre condition suffisante est donnée par ce qu'on appelle les conditions de Hall :

Propriété 12. *(voir [33]). Soit un graphe $G^\varphi = (X, E, \varphi)$. Pour tout $A \subseteq X$ et toute partition de X en cliques $K_1, \ldots, K_p$,*

$$\left| \bigcup_{x \in A_i} \varphi(x) \right| \geq |A_i| \quad (i = 1, \ldots, p)$$

est une condition nécessaire à l'existence d'une coloration restreinte (où $A_i = A \cap K_i$).

Ces conditions sont suffisantes dans le cas où le graphe est une collection de cliques disjointes, et l'on peut se restreindre à la partition en cliques qui consiste à prendre les composantes connexes du graphe. Le problème peut alors être résolu par des techniques de flots dans un réseau (méthode de résolution polynomiale).

Si la taille des ensembles de couleurs admissibles est suffisamment grande, l'existance d'une coloration restreinte est assurée. En effet, si chaque sommet a plus de couleurs autorisées que de voisins, il sera toujours possible de le colorer. Ainsi, si $|\varphi(x)| \geq \Delta(G) + 1$ pour tout sommet x, il existe nécessairement une k-coloration restreinte de G.

Le plus petit entier t telle qu'il existe une coloration restreinte de G pour toute affectation de couleurs admissibles φ respectant $|\varphi(x)| \geq t$ est représenté par $\chi_\ell(G)$, et peut-être appelé le *nombre liste-chromatique* de G. Nous venons de voir que $\chi_\ell(G) \leq \Delta(G) + 1$, et il est facile de se convaincre que $\chi(G) \leq \chi_\ell(G)$. Dans le cas des graphes planaires, il a été démontré que $\chi_\ell(G) \leq 5$, et $\chi_\ell(G) \leq 3$ si en plus ils sont bipartis.

Comme esquissé dans la première section, le problème de la k-coloration restreinte d'un graphe $G^\varphi = (X, E, \varphi)$ peut se ramener à la question de l'existence d'un ensemble stable de $|X|$ sommets dans un graphe auxiliaire $\widetilde{G^\varphi}(k)$. Ce dernier est construit comme le graphe $\widetilde{G}(k)$ présenté dans la première section, à la différence près que l'on supprime la copie a_i d'un sommet a lorsque i n'est pas une couleur admissible pour a (c'est-à-dire $i \notin \varphi(a)$). De cette manière, on est assuré que la i-ème copie du sommet a ne pourra jamais être dans un stable de $\widetilde{G^\varphi}(k)$, et donc qu'on ne lui assignera pas la couleur i.

Il est ainsi possible, grâce à cette transformation, d'utiliser les méthodes de résolution existant pour le problème de stable maximum pour aborder le problème de la k-coloration restreinte. On peut aussi proposer une adaptation de la méthode Tabou à ce problème. Alors que pour le problème de coloration usuel un mouvement consiste à choisir un sommet x ayant un voisin de la même couleur et à modifier sa couleur, il suffit ici de restreindre le choix de la nouvelle couleur aux couleurs autorisées $\varphi(x)$. Une autre adaptation consisterait à autoriser les partitions dans lesquelles un sommet a une couleur qui lui est interdite, mais en

pénalisant cet état de fait. Pour cela, on peut ajouter une constante donnée à la valeur de la fonction-objectif pour chaque sommet ayant une couleur interdite.

Ces adaptations n'ont pas été testées dans la littérature, en partie à cause du fait qu'il est aussi possible de transformer un problème de k-coloration restreinte sur un graphe $G^\varphi = (X, E, \varphi)$ en un problème de k-coloration usuel sur un graphe $G' = (X \cup X', E \cup E')$ obtenu de la manière suivante. En partant du graphe $G = (X, E)$, une clique $X' = \{x'_1, \ldots, x'_k\}$ de taille k est rajoutée. Chaque sommet x de G est ensuite relié à tous les sommets x'_i représentant des couleurs i interdites pour x (c'est-à-dire n'appartenant pas à l'ensemble $\varphi(x)$). Il y a une correspondance biunivoque entre les k-colorations restreintes de G^φ et les k-colorations de G'. En effet, une k-coloration restreinte de G^φ peut être étendue à une k-coloration de G' simplement en colorant le sommet x'_i avec la couleur i. Réciproquement, dans le cas d'une k-coloration de G', nous pouvons supposer que le sommet x'_i a reçu la couleur i (quitte à renuméroter les couleurs) ; l'arête $[x, x'_i]$ permet alors d'assurer que le sommet x n'a pas reçu la couleur interdite i. Cette transformation augmente un peu la taille du graphe à colorer, mais permet ensuite d'appliquer tels quels les algorithmes développés pour le problème de coloration usuel.

10.1. Cas particulier : les extensions de précolorations

Dans le problème de l'*extension de précoloration*, un sous-ensemble de sommets $P \subset X$ du graphe $G = (X, E)$ est précoloré avec

$$\varphi_P : P \longrightarrow \{1, \ldots, k\}$$

où k est un nombre de couleurs. La question est de déterminer si G admet une k-coloration étendant φ_P, c'est-à-dire s'il est possible d'affecter une couleur dans $\{1, \ldots, k\}$ aux sommets de $X \backslash P$ en tenant compte des couleurs qu'ont déjà les sommets dans P. Dans le cas particulier où $P = \emptyset$, nous retrouvons le problème de la k-coloration.

L'étude de ce problème a été motivée notamment par l'observation que, sur les graphes d'intervalles, il permet de modéliser un problème de gestion d'avions [6]. Ce dernier consiste à affecter des vols à un certain nombre d'avions en fonction d'un horaire, avec la condition que le programme de révision (fixé pour chaque avion) ne soit pas modifié.

Le problème de coloration restreinte (ou liste-coloration) a été introduit plus tard (et pour d'autres raisons), mais est une généralisation du problème de l'extension de précoloration. En effet, il suffit de reformuler ce dernier en associant à chaque sommet une liste de couleurs admissibles :

$$\varphi(x) = \begin{cases} \{\varphi_P(x)\} & \text{si } x \in P \\ \{1, \ldots, k\} & \text{si } x \in X \backslash P. \end{cases}$$

Comme le problème de la coloration restreinte, le problème de l'extension de coloration est NP-complet pour les graphes bipartis planaires avec $k = 3$. Si l'on

représente par P_i l'ensemble des sommets précolorés avec la couleur i, nous pouvons supposer, sans perte de généralité, que $|P_1| \geq |P_2| \geq \ldots \geq |P_k|$. Le problème est polynomial pour les graphes parfaits si $P_3 = \emptyset$ et $|P_2| \leq 1$, ainsi que pour les graphes d'intervalles lorsque $|P_1| = 1$. D'autres résultats sur les extensions de coloration et la coloration restreinte peuvent être trouvés dans [28].

11. Colorations avec contraintes de cardinalité

Parfois les contraintes locales, qui ne concernent que quelques sommets (deux ou un dans les problèmes présentés ci-dessus), ne suffisent pas à modéliser d'autres formes de contraintes au demeurant très naturelles. Il peut être nécessaire d'introduire des contraintes de nature globale qui concernent le graphe dans son ensemble. Un tel exemple est donné par les contraintes de cardinalité, qui limitent le nombre de sommets de chaque couleur. Nous considérons un graphe $G = (X, E)$ à colorer avec un nombre k de couleurs, avec la condition supplémentaire qu'il est interdit d'avoir plus de h_i sommets de la couleur i où les h_i sont des bornes données. Le problème de la *k-coloration avec contraintes de cardinalité* consiste à déterminer si une telle k-coloration existe. À nouveau, le problème pourrait facilement s'étendre aux graphes mixtes.

Ce type de contraintes est par exemple utilisé dans la modélisation de problèmes d'ordonnancement pour tenir compte de la disponibilité de certaines ressources, telles que le personnel ou les machines. En effet, puisque dans ce type de problèmes une couleur correspond à un jour, la contrainte de cardinalité h_i peut être fixée au nombre de machines à disposition au cours du jour i. De cette manière il est possible d'assurer que chaque tâche aura une machine à sa disposition. Similairement, dans les problèmes d'horaires scolaires ces contraintes permettent de prendre en compte le nombre de salles à disposition à chaque heure.

En posant h_i égal au nombre de sommets du graphe considéré, on se rend compte que le problème de la k-coloration est un cas particulier de ce problème. Ce dernier est donc difficile. En fait, il est NP-complet pour les graphes d'intervalles (même si tous les h_i valent 4) et les graphes bipartis. Les seules classes connues pour lesquelles le problème se résout en temps polynomial sont les cliques disjointes et les complémentaires des graphes sans triangle. Des conditions nécessaires sont données par :

Propriété 13. *(voir [33]). Soit un graphe $G = (X, E)$ avec des contraintes de cardinalité h_i, $i = 1, \ldots, k$. Pour tout $A \subseteq X$, tout $C^* \subseteq \{1, \ldots, k\}$ et toute partition de X en cliques $K_1, \ldots, K_p$,*

$$\kappa_A(k - |C^*|) + \sum_{i \in C^*} h_i \geq |A|$$

est une condition nécessaire à l'existence d'une k-coloration avec contraintes de cardinalité (où κ_A est le nombre de cliques K_j intersectant A).

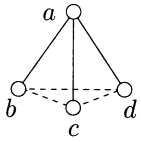


FIGURE 11. La griffe.

Comme dans le cas de la propriété 12, ces conditions sont suffisantes pour les graphes qui sont des cliques disjointes.

Un cas particulier du problème de k-coloration avec contraintes de cardinalité qui est mieux étudié dans la littérature est celui où toutes les bornes h_i sont égales à une certaine valeur h. On définit alors le *nombre chromatique borné* $\chi_h(G)$ comme étant le plus petit entier k pour lequel il existe une k-coloration de G avec au plus h sommets de chaque couleur.

Il est facile de voir que $\chi_h(G) = \chi(G)$ lorsque $h \geq \alpha(G)$. En effet, dans toute k-coloration de G, chaque ensemble de sommets ayant une même couleur forme un stable, qui est nécessairement de taille au plus $\alpha(G)$ (par définition de ce dernier). De même, nous avons clairement que $\chi_1(G)$ est égal au nombre n de sommets dans G. Une dernière valeur que l'on peut obtenir en temps polynomial quel que soit G est $\chi_2(G)$. En effet, si E' est un couplage maximum dans le graphe complémentaire $\overline{G}$ (un couplage est un ensemble d'arêtes non adjacentes deux à deux), nous avons $\chi_2(G) = n - |E'|$. Or, il se trouve qu'un couplage maximum peut être trouvé en temps polynomial [23].

Pour les autres valeurs de h, il est possible de borner la valeur de $\chi_h(G)$ par

$$\max\left(\left\lceil \frac{n}{h} \right\rceil, \chi(G)\right) \leq \chi_h(G) \leq h + \left\lfloor \frac{n - \chi(G)}{h} \right\rfloor$$

où n est le nombre de sommets dans G [20]. La borne $\lceil \frac{n}{h} \rceil$ provient directement du fait que l'on cherche à recouvrir n sommets avec des ensembles de taille au plus h. Dans le cas des graphes sans griffe (c'est-à-dire ne contenant pas la structure de la Fig. 11), la borne inférieure est même égale à $\chi_h(G)$.

La borne supérieure dépend du nombre chromatique $\chi(G)$ qui est en général lui-même difficile à déterminer. C'est pourquoi il peut être utile d'avoir une autre borne supérieure, plus simple à calculer. Si l'on met de côté les cliques (pour lesquelles $\chi_h(G) = \Delta(G) + 1$) et les cycles impairs lorsque $h > \frac{n-1}{2}$ (pour lesquels $\chi_h(G) = 3$), nous avons

$$\chi_h(G) \leq \Delta(G) + \left\lfloor \frac{n - \Delta(G)}{h} \right\rfloor.$$

Pour les graphes bipartis, il même possible de donner une borne supérieure qui diffère au plus de 1 de la borne inférieure. En effet, considérons un graphe biparti ayant comme ensemble de sommets $X = X_1 \cup X_2$ et des arêtes uniquement entre X_1 et X_2. L'absence d'arêtes reliant deux sommets de X_1 nous permet de colorer

X_1 avec au plus $\left\lceil \frac{|X_1|}{h} \right\rceil$ couleurs. Le même raisonnement s'applique à X_2, et donc

$$\left\lceil \frac{|X|}{h} \right\rceil \leq \chi_h(G) \leq \left\lceil \frac{|X_1|}{h} \right\rceil + \left\lceil \frac{|X_2|}{h} \right\rceil \leq \left\lceil \frac{|X|}{h} \right\rceil + 1.$$

Mais comme déjà mentionné, le problème de la détermination de la bonne valeur est NP-complet. Dans le cas particulier des arbres, le problème devient polynomial.

Comme pour les autres problèmes, la façon la plus naturelle de résoudre une instance de ce problème consiste à utiliser une adaptation des méthodes connues pour le problème de k-coloration. Dans le cas de la méthode Tabou, on peut garder la même notion de solution (une partition de l'ensemble de sommets en k sous-ensembles) et ajouter à la fonction-objectif une valeur proportionnelle à

$$\text{nombre de sommets de couleur } i \ - h_i$$

pour toute couleur i violant sa contrainte de cardinalité. Dans ce cas, un mouvement consisterait à modifier la couleur soit d'un sommet ayant un voisin de la même couleur, soit d'un sommet ayant une couleur utilisée trop fréquemment (par rapport à la contrainte de cardinalité).

11.1. Cas particulier : les colorations restreintes avec contraintes de cardinalité

Nous avons vu dans les sections précédentes que les contraintes de cardinalité ainsi que les ensembles de couleurs admissibles pour les sommets apparaissent de manière naturelle dans les problèmes d'horaires scolaires par exemple. Dans le cas où le graphe G considéré est une simple chaîne, le problème de coloration restreinte avec contraintes de cardinalité correspond également à la situation suivante.

Des travaux, au nombre de n et chacun ayant une durée d'exécution d'une période, sont placés sur une ligne à intervalles réguliers. Un ensemble $\varphi(v)$ de périodes auxquelles v peut être exécuté est donné pour chaque travail v. Ces travaux sont traités par des robots dont h_i sont à disposition au cours de la période i. À cause de la place nécessaire à ces robots, deux travaux qui sont situés à des places adjacentes sur la ligne (correspondant à deux sommets adjacents dans G) ne peuvent pas être exécutés en même temps. Déterminer s'il existe un horaire admissible de k périodes pour réaliser ces travaux est équivalent au problème de l'existence d'une k-coloration restreinte de G^φ avec contraintes de cardinalités h_i.

Malheureusement, ce problème est NP-complet, même si chaque travail a au plus deux périodes au cours desquelles il peut être exécuté. La seule classe de graphes pour laquelle un algorithme polynomial est connu est celle des cliques disjointes. Dans ce cas, un algorithme de flots permet de résoudre le problème [33].

Les propriétés 12 et 13 présentées dans les paragraphes précédents sont en fait des cas particuliers de la propriété suivante :

Propriété 14. *(voir [33]). Soit un graphe $G^\varphi = (X, E, \varphi)$ avec des contraintes de cardinalité h_i, $i = 1, \ldots, k$. Pour tout $A \subseteq X$, tout $C^* \subseteq \{1, \ldots, k\}$ et toute*

partition de X en cliques $K_1, \ldots, K_p$,

$$\sum_{i=1}^{p} \left| \left[\bigcup_{x \in A_i} \varphi(x) \right] \cap (\{1, \ldots, k\} - C^*) \right| + \sum_{i \in C^*} h_i \geq |A|$$

est une condition nécessaire à l'existence d'une k-coloration restreinte avec contraintes de cardinalité (où $A_i = A \cap K_i$).

Ces conditions sont suffisantes lorsque $G = (X, E)$ est une collection de cliques disjointes, et ces graphes sont les seuls pour lesquelles ces conditions sont suffisantes quel que soit le choix de φ et des h_i.

12. D'autres extensions

En dépit de la complexité des formules donnant des bornes du nombre T-chromatique, les modèles de graphes décrits jusqu'ici ne sont pas encore susceptibles d'appréhender la totalité des types de contraintes que l'on rencontre dans les problèmes réels.

Dans les modèles décrits plus haut (excepté dans le cas des contraintes de cardinalité), nous nous sommes limités à envisager des contraintes portant sur des tâches isolées ou des paires de tâches ; qu'il s'agisse de précédence ou de disjonction, deux tâches intervenaient en effet dans la formulation de la contrainte. Quant aux contraintes de couleurs interdites, elles étaient liées à une tâche.

Nous pouvons cependant concevoir une description plus générale des contraintes de disjonction, qui se formuleraient en donnant un ensemble R de tâches et un entier r avec l'exigence que parmi les tâches de R il y en a au plus r qui peuvent être simultanément en cours d'exécution. (Dans le cas des contraintes de disjonction, nous avions $|R| = 2$ et $r = 1$ pour chaque contrainte). Ce type de contraintes apparaît notamment lorsqu'une ressource (qualifiée de "renouvelable" par les spécialistes de l'ordonnancement car elle est utilisée par des tâches comme le serait une machine et non consommée définitivement, comme le serait un carburant) est disponible en quantité r ; les tâches de R utilisent chacune une unité de cette ressource pendant leur exécution.

Le modèle que l'on associe à un tel problème est alors un *hypergraphe* (construit comme précédemment sur des sommets associés aux tâches) dont les ensembles $R\,(|R| \geq 2)$ associés aux contraintes sont les arêtes.

Ces modèles sont plus complexes que ceux basés sur les graphes et, sauf dans des cas très particuliers, ne donnent pas lieu à des algorithmes exacts qui sont polynomiaux (voir [4]). Des méthodes heuristiques sont à développer pour ces cas plus généraux ; la dérivation de bornes sur le nombre chromatique de ces hypergraphes peut dans certains cas s'appuyer sur les techniques utilisées pour les graphes en les généralisant.

13. Les colorations d'arêtes

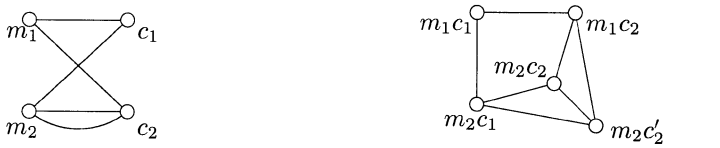
Dans des cas très spécifiques, nous pouvons formuler un problème d'emploi du temps en termes de coloration d'arêtes. Etant donné un multigraphe $G = (X, E)$ (celui-ci peut avoir des arêtes multiples, mais pas de boucles), une *k-coloration des arêtes de G* est une affectation à chaque arête e d'une couleur $c(e)$ choisie dans l'ensemble $\{1, ..., k\}$ de manière que deux arêtes adjacentes soient toujours de couleurs différentes. Le plus petit nombre k de couleurs tel que G a une k-coloration d'arêtes est l'*indice chromatique* de G, noté parfois $\chi'(G)$. Clairement, nous avons $\Delta(G) \leq \chi'(G)$; Vizing [29] a démontré que pour les graphes simples nous avons aussi $\chi'(G) \leq \Delta(G) + 1$. Déterminer si $\chi'(G) = \Delta(G)$ est un problème NP-complet.

Le modèle de coloration d'arêtes est généralement utilisé dans les cas les plus simples de problème d'horaire scolaire où l'on a un ensemble M de maîtres et de maîtresses m_i, un ensemble C de classes (groupes d'élèves) c_j et une collection E de leçons (durant chacune une heure) caractérisées par le maître (ou la maîtresse) m_i qui la donne et la classe c_j qui la suit ; pour associer un graphe au problème de la construction d'un horaire en k heures, nous introduisons un sommet pour chaque m_i, un sommet pour chaque c_j et chaque leçon impliquant m_i et c_j est représentée par une arête $[m_i, c_j]$; le graphe $G = (M, C, E)$ obtenu est biparti. Il est connu que $\chi'(G) = \Delta(G)$ pour les multigraphes bipartis (voir [3]). On vérifie aisément qu'il y a correspondance entre les horaires en k heures et les k-colorations d'arêtes de G, puisque chaque maître (ou maîtresse) ne peut donner deux leçons simultanément et que chaque classe ne peut suivre plus d'une leçon à la fois. On trouvera dans [32] des extensions et variations de ce modèle.

Les modèles de coloration d'arêtes sont en fait des cas particuliers des colorations (de sommets) définies précédemment. Il suffit d'observer que l'on peut toujours transformer le problème de la coloration des arêtes d'un multigraphe G avec k couleurs en un problème de k-coloration des sommets d'un graphe simple $L(G)$ associé à G (et appelé graphe *aux arêtes* de G). $L(G)$ est obtenu en introduisant un sommet $\overline{e}$ pour chaque arête e de G et en reliant dans $L(G)$ les sommets $\overline{e}$ et $\overline{f}$ s'ils correspondent à des arêtes e, f adjacentes de G. On constate alors qu'il y a correspondance entre les k-colorations d'arêtes de G et les k-colorations (de sommets) de $L(G)$. La figure 12 donne un exemple de problème d'horaires avec la transformation décrite ci-dessus. Par contre, on ne peut pas généralement transformer un problème de k-coloration (de sommets) en un problème de k-coloration d'arêtes.

Dans le cas des multigraphes bipartis, les colorations d'arêtes sont obtenues par des méthodes polynomiales classiques (voir [32]), mais pour des multigraphes quelconques, le problème de la k-coloration d'arêtes est NP-complet.

Si le problème de k-coloration des arêtes d'un multigraphe G peut toujours se ramener à un problème de k-coloration des sommets d'un graphe associé $L(G)$, il existe cependant des variations du modèle de coloration des arêtes survenant dans les problèmes d'ordonnancement qui ne peuvent pas se ramener de manière



a) le multigraphe G biparti associé à un problème d'horaire scolaire (2 maîtresses, 2 classes, 5 leçons).

b) le graphe aux arêtes $L(G)$ du multigraphe G de a).

FIGURE 12. Transformation d'un problème de coloration d'arêtes en un problème de coloration (de sommets).

naturelle à un modèle de coloration des sommets. Nous allons faire un survol de quelques-unes de ces variations.

13.1. Applications sportives

Les problèmes de coloration d'arêtes se présentent aussi dans le cadre de la construction de calendriers de ligues de sport. Plus précisément, il s'agit alors de colorer les arêtes d'un graphe et en plus de leur donner une orientation adéquate ; on parle alors de trouver une *coloration orientée* d'un graphe.

Considérons donc une ligue de $2n$ équipes $1, 2, \ldots, 2n$. Au cours d'un tournoi, chaque équipe doit rencontrer une fois exactement chacune des autres équipes. La construction d'un calendrier minimisant le nombre de journées de jeu est un problème de coloration d'arêtes d'un graphe complet K_{2n} (graphe à $2n$ sommets représentant les équipes avec les $2n \cdot (2n-1)$ arêtes représentant les matches à jouer) : on doit donc colorer les arêtes de K_{2n} avec un nombre minimum de couleurs ; chaque couleur représente une journée et l'on sait que $\chi'(K_{2n}) = 2n-1$. Il faut donc $2n-1$ journées pour que tous les matches prévus aient lieu (à raison de n matches par journée). La construction d'une telle coloration $(F_1, F_2, \ldots, F_{2n-1})$ où F_i représente les matches à jouer le jour i (couplage des n arêtes de couleur i de K_{2n}) peut se faire très simplement en posant :

$$F_i = \{[2n, i]\} \cup \{[i+k, i-k] \ : \ k = 1, 2, \ldots, n-1\}$$

où les nombres $i+k$ et $i-k$ sont choisis modulo $2n-1$ entre 1 et $2n-1$ (voir [3, Ch. 9]). Graphiquement, on place les équipes $1, 2, \ldots, 2n-1$ sur un cercle, l'équipe $2n$ étant au centre du cercle ; on construit $F_1 = \{[2n, 1], [2, 2n-1], [3, 2n-2], \ldots, [n, n+1]\}$; les classes successives F_2, F_3, ... s'obtiennent sur le dessin en faisant tourner la configuration d'arêtes de F_1 (voir Fig. 13 pour $2n = 6$).

En pratique, un match entre l'équipe i et l'équipe j est joué soit sur le terrain de i soit sur le terrain de j. Dans le premier cas, c'est un jeu "à la maison" (désigné par H pour "home") pour i et "à l'extérieur" (désigné par A pour "away") pour j. On souhaite en général que pour chacune des $2n$ équipes, l'alternance des matches H et A soit aussi régulière que possible. Il n'est évidemment pas possible d'avoir

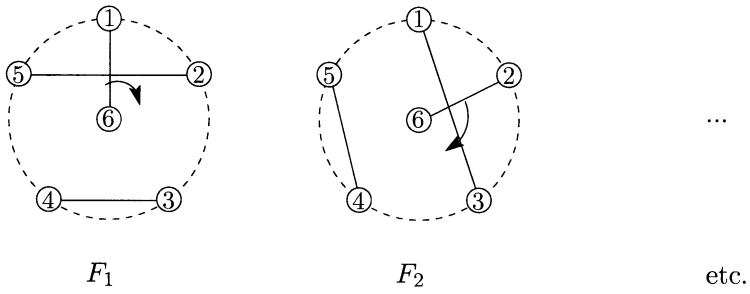


FIGURE 13. Construction du calendrier d'un tournoi à $2n = 6$ équipes.

jour 1	$\overleftarrow{61}$	$\overrightarrow{25}$	$\overleftarrow{34}$
jour 2	$\overrightarrow{62}$	$\overrightarrow{31}$	$\overleftarrow{45}$
jour 3	$\overleftarrow{63}$	$\overrightarrow{42}$	$\overleftarrow{51}$
jour 4	$\overrightarrow{64}$	$\overrightarrow{53}$	$\overleftarrow{12}$
jour 5	$\overleftarrow{65}$	$\overrightarrow{14}$	$\overleftarrow{23}$

FIGURE 14. Calendrier pour $2n = 6$ équipes.

une alternance parfaite (pour une ligue de plus de 2 équipes !) ; on cherche donc à minimiser le nombre de fois où une équipe a deux matches A consécutifs (ou deux matches H consécutifs) ; on peut montrer qu'on a toujours au moins $2n - 2$ telles ruptures d'alternance (voir [31]).

La construction d'un calendrier peut être représentée par la coloration des arêtes déjà mentionnée à laquelle on rajoute une orientation des arêtes pour indiquer où les matches entre i et j sont joués : une arête $[i, j]$ sera orientée de i vers j si le match est joué sur le terrain de j ($[i, j]$ devient l'arc (i, j)) et de j vers i dans le cas contraire. On peut vérifier que la construction ci-dessous donne bien une coloration orientée de K_{2n} (avec les ensembles F_i définis précédemment) avec un nombre minimum de ruptures d'alternances [31] :

(1) pour tout i, l'arête $[2n, i]$ devient l'arc $(i, 2n)$ pour i impair ou l'arc $(2n, i)$ pour i pair ;
(2) pour tout i, l'arête $[i + k, i - k]$ devient l'arc $(i + k, i - k)$ pour k impair ou l'arc $(i - k, i + k)$ pour k pair.

Pour le cas de $2n = 6$, le calendrier est donné à la figure 14.

Ce modèle de base doit encore être adapté aux situations diverses qui se présentent, notamment lorsque l'on doit tenir compte de contraintes de disponibilité des terrains de sport. De plus, il arrive aussi qu'une équipe soit amenée à jouer successivement des matches à l'extérieur afin de réduire la distance à parcourir pendant la saison. Des méthodes plus générales doivent alors être utilisées pour traiter le problème de la construction des horaires, mais celles-ci restent basés sur

les colorations d'arêtes (voir [12] par exemple) qui fournissent un outil fondamental de formulation.

13.2. Les f-colorations d'arêtes

D'autres modèles de coloration des arêtes d'un multigraphe surviennent dans le domaine technologique. Considérons en effet le problème de transfert de fichiers dans un réseau d'ordinateurs. Soient un ensemble X d'ordinateurs, chacun capable de communiquer directement avec chaque autre, et un ensemble E de grands fichiers devant être transférés entre divers ordinateurs. Chaque ordinateur $v \in X$ dispose de $f(v)$ ports identiques lui permettant de gérer jusqu'à $f(v)$ transferts à tout instant. Nous supposons ici que chaque fichier est transféré directement de son origine à sa destination, qu'une fois un transfert a commencé il s'effectue sans interruption, et que tous les transferts nécessitent le même temps. Le but est de trouver un ordonnancement des transferts de manière à minimiser le temps total requis pour effectuer tous les transferts.

Ce problème peut être modélisé à l'aide d'un multigraphe $G^f = (X, E, f)$ dont les sommets représentent les ordinateurs et où chaque fichier est représenté par une arête liant les deux ordinateurs concernés par le fichier. Trouver un ordonnancement minimisant le temps total revient alors à trouver une affectation d'un nombre minimal $\chi'_f(G^f)$ de couleurs aux arêtes de G^f de manière que, pour chaque sommet v et chaque couleur i, on ait $|C_i(v)| \leq f(v)$, où $C_i(v)$ est l'ensemble des arêtes de couleur i incidentes à v. Ce problème est connu sous le nom de *f-coloration des arêtes*.

Puisque le cas particulier où $f(v) = 1$ pour tout ordinateur v revient à déterminer l'indice chromatique de G^f, le problème de f-coloration des arêtes est difficile en général. Mais dans la situation où chaque ordinateur n'a que des fichiers à envoyer ou que des fichiers à recevoir (*i.e.* le multigraphe G^f associé est biparti), le problème peut être résolu en temps polynomial [9].

Pour un graphe $G^f = (X, E, f)$, soient $d(v)$ le degré de v dans G^f et

$$\Delta_f(G^f) = \max_{v \in X} \left\lceil \frac{d(v)}{f(v)} \right\rceil .$$

Nous avons alors clairement $\Delta_f(G^f) \leq \chi'_f(G^f)$. En considérant le problème de l'indice chromatique sur un graphe secondaire construit à partir de G^f, on peut montrer que $\chi'_f(G^f) \leq \Delta_f(G^f) + 1$ lorsque G^f est un graphe simple [38]. Pour les multigraphes bipartis et pour les graphes simples planaires tels que $\Delta_f(G^f) \geq 8$, on a $\Delta_f(G^f) = \chi'_f(G^f)$.

De nombreuses variations et généralisations du problème de transferts de fichiers, incluant par exemple des temps de transferts dépendant de la taille des fichiers ou considérant l'absence d'un contrôleur central construisant l'ordonnancement, sont étudiés dans [9].

13.3. Les $[g, f]$-colorations d'arêtes

Dans le problème de transferts de fichiers, la fonction f nous permet de limiter le nombre de fichiers qu'un ordinateur peut recevoir et envoyer à tout instant. Si l'on désire équilibrer le processus de transfert, on peut exiger qu'à chaque instant au moins $g(v)$ ports de l'ordinateur v soient utilisés pour un transfert. Ce nouveau problème peut être modélisé à l'aide du problème de $[g, f]$-*coloration d'arêtes* (en k couleurs) sur un multigraphe $G^{g,f} = (X, E, g, f)$. Celui-ci consiste à affecter une couleur (parmi les k disponibles) à chaque arête de $G^{g,f}$ de manière que $g(v) \leq |C_i(v)| \leq f(v)$ pour tout sommet $v \in X$ et toute couleur i. Le problème de $[g, f]$-coloration d'arêtes est aussi connu sous le nom de problème de $[g, f]$-factorisation.

Pour toute $[g, f]$-coloration des arêtes d'un multigraphe $G^{g,f}$ en k couleurs, nous avons

$$\Delta_f(G^{g,f}) \leq k \leq \min_{v \in X : g(v) > 0} \left\lfloor \frac{d(v)}{g(v)} \right\rfloor .$$

De manière équivalente, un multigraphe $G^{g,f}$ n'a pas de $[g, f]$-coloration des arêtes en k couleurs s'il existe un sommet v tel que

$$g(v) > \left\lfloor \frac{d(v)}{k} \right\rfloor \quad \text{ou} \quad f(v) < \left\lceil \frac{d(v)}{k} \right\rceil .$$

Certaines conditions permettent d'assurer l'existence d'une $[g, f]$-coloration des arêtes en k couleurs d'un graphe.

Propriété 15. *(voir [22]). Soit un graphe simple $G^{g,f} = (X, E, g, f)$ tel que les sommets v pour lesquels $g(v) = f(v)$ sont deux à deux non-adjacents et tel que $k \cdot g(w) + 1 \leq d(w) \leq k \cdot f(w) - 1$ pour les autres sommets w (avec $k \geq 2$). Alors $G^{g,f}$ a une $[g, f]$-coloration des arêtes en k couleurs.*

Une coloration en k couleurs des arêtes d'un graphe G est dite *équitable* si l'on a $||C_i(v)| - |C_j(v)|| \leq 1$ pour chaque sommet v de G et $1 \leq i < j \leq k$. Dans le cas particulier où

$$g(v) = \left\lfloor \frac{d(v)}{k} \right\rfloor \quad \text{et} \quad f(v) = \left\lceil \frac{d(v)}{k} \right\rceil$$

pour tout sommet v, une $[g, f]$-coloration de $G^{g,f}$ est une coloration équitable et la propriété précédente devient :

Corollaire 1. *(voir [22]). Soit un graphe simple $G = (X, E)$ tel que les sommets v pour lesquels k divise $d(v)$ sont deux à deux non-adjacents. Alors G a une coloration équitable des arêtes en k couleurs.*

Notons que, même si un graphe n'a pas de coloration équitable des arêtes en k couleurs, il a malgré tout une coloration "quasiment équitable" des arêtes en k couleurs, où $||C_p(v)| - |C_q(v)|| = 2$ pour au plus une paire de couleurs $\{p, q\}$ et $||C_i(v)| - |C_j(v)|| \leq 1$ pour toutes les autres paires de couleurs $\{i, j\} \neq \{p, q\}$ [22].

Des algorithmes (séquentiels et parallèles) polynomiaux pour le problème de $[g, f]$-coloration d'arêtes sur diverses classes de graphes sont proposés dans [38].

14. Perspectives d'avenir

Les problèmes réels se formulant en termes de coloration de graphes impliquent souvent des graphes de grande taille. Certains travaux semblent montrer que de nombreux graphes provenants de problèmes réels sont 1-parfaits (un graphe G est *1-parfait* si $\chi(G) = \omega(G)$), ce qui mène à des algorithmes de coloration de sommets extrêmement efficaces permettant de trouver une coloration optimale en quelques secondes pour des graphes avec plusieurs milliers de sommets [10]. Mais pour des graphes aléatoires, ou pour s'attaquer à diverses variations du problème de coloration, il est nécessaire de trouver des algorithmes plus efficaces que ceux existants pour ces grands graphes. En effet, la méthode Tabou nécessite un temps de calcul prohibitif pour ceux-ci. Dans ces cas, les algorithmes évolutifs peuvent se révéler utiles. Ces algorithmes connaissent un grand succès en optimisation combinatoire ces dernières années, mais leurs performances sont très variables. Les algorithmes évolutifs les plus connus sont les algorithmes génétiques, dont nous décrivons le principe général.

Contrairement à la méthode Tabou dans laquelle il n'y a qu'une solution courante qui est modifiée au fil des itérations, les algorithmes génétiques gèrent un ensemble (appelé *population*) de solutions (appelées *individus*). A chaque itération ("génération"), des paires d'individus sont sélectionnées (en préférant des solutions de bonne qualité par rapport à la fonction-objectif), et les deux individus d'une paire sont utilisés pour créer un ou plusieurs nouveaux individus. Ces derniers subissent en général encore une modification aléatoire (l'analogue d'une *mutation*) ou, mieux, sont améliorés à l'aide d'un algorithme itératif, du type Tabou par exemple. La nouvelle population courante est composée d'individus choisis parmi la population précédente et ces nouveaux individus.

Pour le problème de coloration usuel, un individu est une partition de l'ensemble des sommets et la fonction-objectif reste celle utilisée dans l'algorithme Tabou. On cherche donc à nouveau une partition s telle que $f(s) = 0$. Sur la base de deux partitions $s_1 = (X_1^1, \ldots, X_k^1)$ et $s_2 = (X_1^2, \ldots, X_k^2)$, une nouvelle partition $s' = (X_1', \ldots, X_k')$ peut être créée de la manière suivante. Considérons un sommet v se trouvant dans l'ensemble X_i^1 de la partition s_1 et dans l'ensemble X_j^2 de la partition s_2 : dans s', v sera placé dans l'ensemble X_i' ou X_j' en fonction du nombre de voisins qu'il a dans X_i^1 et X_j^2. Plus de détails sont donnés dans [13], où est proposé l'algorithme évolutif suivant :

```
Algorithme évolutif ;
begin
    choisir une population initiale P ;
    s* := meilleur individu de P ;
    i := 0                -- compteur du nombre d'itérations ;
    while aucun critère d'arrêt n'est satisfait do
        i := i + 1 ;
        sélectionner quatre individus s1 à s4 dans P ;
```

```
            créer s'_1 sur la base de s_1 et s_2 ;
            créer s'_2 sur la base de s_3 et s_4 ;
            appliquer l'algorithme Tabou à s'_1 et s'_2 ;
            ajouter s'_1 et s'_2 à P ;
            enlever les deux plus mauvais individus de P ;
            if f(meilleur individu dans P) < f(s*) then
                s* := meilleur individu de P ;
            end if ;
        end while ;
    end algorithme.
```

Si cet algorithme n'est pas concurrentiel par rapport à une méthode Tabou, il est cependant utile pour les graphes aléatoires de 500 sommets et plus (densité $0,5$). En effet, cet algorithme est plus robuste que la méthode Tabou sur les graphes résiduels mentionnés dans la section 5 (voir [13]).

Par contre, de meilleures manières de créer un individu sur la base de deux autres individus (voire plus) pourraient permettre d'avoir des algorithmes plus efficaces. La recherche d'un tel opérateur, ainsi que le développement de bons algorithmes heuristiques pour diverses généralisations et variations du problème de coloration, constituent un axe important de la recherche dans le domaine de la coloration de graphes.

Références

[1] N. Alon et M. Tarsi, Colorings and orientations of graphs. *Combinatorica* **12** (1992) 125-134.

[2] M. Bellare, O. Goldreich et M. Sudan, Free bits, PCPs and non-approximability – towards tight results. *SIAM J. Comput.* **27** (1998) 804-915.

[3] C. Berge, *Graphes*. Gauthier-Villars, Paris (1983).

[4] C. Berge, *Hypergraphes*. Gauthier-Villars, Paris (1987).

[5] C. Berge et V. Chvátal, Topics on Perfect Graphs. *Ann. Discrete Math.* **21** (1984).

[6] M. Biró, M. Hujter et Zs. Tuza, Precoloring extension. I. Interval graphs. *Discrete Math.* **100** (1992) 267-279.

[7] H.L. Bodlaender, K. Jansen et G. Woeginger, Scheduling with incompatible jobs. *Discrete Appl. Math.* **55** (1994) 219-232.

[8] V. Chvátal, Perfectly ordered graphs, in *Topics on Perfect Graphs. North Holland Math. Stud.* **88**, *Annals Discrete Math.* **21** (1984) 63-65.

[9] E.G. Coffman Jr., M.G. Garey, D.S. Johnson et A.S. Lapaugh, Scheduling file transfers. *SIAM J. Comput.* **14** (1985) 744-780.

[10] O. Coudert, Exact Coloring of Real-Life Graphs is Easy, in *Proc. of 34th ACM/IEEE Design Automation Conf.* ACM Press, New York (1997) 121-126.

[11] N. Dubois et D. de Werra, EPCOT : An Efficient Procedure for Coloring Optimally with Tabu Search. *Comput. Math. Appl.* **25** (1993) 35-45.

[12] K. Easton, G. Nemhauser et M. Trick, *The traveling tournament problem : description and benchmarks*. GSIA, Carnegie Mellon University (2002).

[13] C. Fleurent et J.A. Ferland, *Genetic and Hybrid Algorithms for Graph Coloring*, édité par G. Laporte et I.H. Osman (éds). Metaheuristics in Combinatorial Optimization, *Ann. Oper. Res.* **63** (1996) 437-461.

[14] M.G. Garey et D.S. Johnson, The complexity of near-optimal graph coloring. *J. ACM* **23** (1976) 43-49.
[15] M.G. Garey, D.S. Johnson et L. Stockmeyer, Some simplified NP-complete graph problems. *Theoret. Comput. Sci.* **1** (1976) 237-267.
[16] F. Glover et M. Laguna, *Tabu Search*. Kluwer Academic Publ. (1997).
[17] M.C. Golumbic, *Algorithmic Graph Theory and Perfect Graphs*. Academic Press, New York (1984).
[18] M. Grötschel, L. Lovasz et A. Schrijver, *Geometric Algorithms and Combinatorial Optimization*. Springer-Verlag, Berlin (1988).
[19] M.M. Halldórsson, *A still better performance guarantee for approximate graph coloring. Inform. Process. Lett.* **45** (1993) 19-23.
[20] P. Hansen, A. Hertz et J. Kuplinsky, Bounded Vertex Colorings of Graphs. *Discrete Math.* **111** (1993) 305-312.
[21] P. Hansen, J. Kuplinsky et D. de Werra, Mixed Graph Coloring. *Math. Meth. Oper. Res.* **45** (1997) 145-160.
[22] A.J.W. Hilton et D. de Werra, A sufficient condition for equitable edge-colourings of simple graphs. *Discrete Math.* **128** (1994) 179-201.
[23] E.L. Lawler, *Combinatorial Optimization : Networks and Matroids*. Holt, Rinehart and Winston, New York (1976).
[24] F. Leighton, A Graph Coloring Algorithm for Large Scheduling Problems. *J. Res. National Bureau Standards* **84** (1979) 742-774.
[25] M. Middendorf et F. Pfeiffer, *On the complexity of recognizing perfectly orderable graphs*, Discrete Mathematics 80 (1990) 327-333.
[26] A. Pnueli, A. Lempel et S. Even, Transitive orientation of graphs and identification of permutation graphs. *Canadian J. Math.* **23** (1971) 160-175.
[27] F.S. Roberts, *Discrete Mathematical Models*. Prentice-Hall, Englewood Cliffs (1976).
[28] Zs. Tuza, *Graph colorings with local constraints – a survey*, Discussiones Mathematicae – Graph Theory 17 (1997) 161-228.
[29] V.G. Vizing, *On an estimate of the chromatic class of a p-graph* (en russe), Metody Discret Analiz. 3 (1964) 25-30.
[30] D.J.A. Welsh et M.B. Powell, *An upper bound on the chromatic number of a graph and its application to timetabling problems*, Computer J. 10 (1967) 85-87.
[31] D. de Werra, *Some models of graphs for scheduling sports competitions*, Discrete Applied Mathematics 21 (1988) 47-65.
[32] D. de Werra, *The combinatorics of timetabling*, European Journal of Operational Research 96 (1997) 504-513.
[33] D. de Werra, *On a multiconstrained model for chromatic scheduling*, Discrete Applied Mathematics 94 (1999) 171-180.
[34] D. de Werra, Ch. Eisenbeis, S. Lelait et B. Marmol, *On a graph-theoretical model for cyclic register allocation*, Discrete Applied Mathematics 93 (1999) 191-203.
[35] D. de Werra et Y. Gay, *Chromatic scheduling and frequency assignment*, Discrete Applied Mathematics 49 (1994) 165-174.
[36] D. de Werra et A. Hertz, *Consecutive colorings of graphs*, Zeischrift für Operations Research 32 (1988) 1-8.
[37] D. de Werra et D. Kobler, *Coloration et ordonnencement chromatique*, ORWP 00/04, Ecole Polytechnique Fédérale de Lausanne, 2000.
[38] X. Zhou et T. Nishizeki, *Graph Coloring Algorithms*, IEICE Trans. on Information and Systems E83-D (2000) 407-417.