MASTER EN SCIENCES INFORMATIQUES

Établissement d'horaires de cours par coloration de graphes

Henkes, Oswald

Award date:
1982

Awarding institution:
Universite de Namur

FACULTES
UNIVERSITAIRES
N.D. DE LA PAIX

NAMUR

INSTITUT D'INFORMATIQUE

ETABLISSEMENT D'HORAIRES DE COURS PAR COLORATION DE GRAPHES

Oswald Henkes

Mémoire présenté en vue de
l'obtention du grade de
Maître en informatique

Année Académique 1981 - 1982

Je remercie vivement Monsieur Jean FICHEFET d'avoir accepté la direction de ce mémoire.

Je tiens à exprimer toute ma reconnaissance à Jean-Paul LECLERCQ pour l'aide efficace apportée tout au long de ce travail. Il m'a donné entière liberté de faire un travail personnel tout en m'aidant par ses conseils et remarques. Je lui suis surtout très reconnaissant pour l'esprit ouvert et l'ambiance sympathique qui m'ont accompagnés pendant la réalisation de ce mémoire.

Je remercie finalement tous ceux qui de près ou de loin ont contribué à la réalisation de ce travail.

Herzlichen Dank meinen Eltern für die langjährige Unterstützung und liebevolle Aufmerksamkeit, die mir während meines Studiums eine unentbehrliche Hilfe waren.

T A B L E D E S M A T I E R E S

I N T R O D U C T I O N

La conception d'un horaire de cours d'un établissement scolaire est depuis longtemps un casse-tête délicat et compliqué. Chaque année, ce travail très lourd préoccupe les responsables administratifs de nombreuses écoles, mais de bons horaires sont rarement trouvés. L'apparition de l'informatique comme outil de gestion a fait naître le désir d'utiliser l'ordinateur comme soutien lors de l'établissement d'un horaire.

Les horaires conçus manuellement présentent des inconvénients dûs entre autres au fait que l'ampleur du problème combinatoire dépasse les capacités d'appréhension globale d'un individu. Il en résulte des oublis, des incompatibilités, des difficultés de modification, une dépense d'énergie et de temps considérable ainsi que des difficultés liées à un manque d'information (composition des cours, indisponibilités,...).

De nombreuses conversations avec les responsables des horaires ont fourni une vue globale et assez complète du problème. A côté des données de base (classes,professeurs,locaux,cours à donner), on distingue de nombreuses contraintes (ne pas placer deux cours différents du même professeur à une même heure, indisponibilités de locaux, cours à donner consécutivement,...) de nature pratique ou pédagogique.

Le but de ce travail est dès lors d'établir un logiciel résolvant tous les problèmes rencontrés et qui soit utilisable par des non-spécialistes en informatique. Ce programme doit donc gérer les informations nécessaires, effectuer les calculs proprement dits et sortir sous forme compacte les résultats obtenus.

Pour réaliser un outil de gestion répondant aux exigences réelles, il faut surtout trouver un modèle mathématique approprié. Nous travaillons avec un modèle de graphe qu'on va adapter aux contraintes envisagées. La recherche de bonnes méthodes de calcul mènera à diverses algorithmes dans le domaine dit de "coloration de graphes". Le lecteur non-familiarisé en théorie de graphes est référé à l'annexe 1 où il trouvera les notions élémentaires absolument nécessaires à la compréhension de la partie théorique (premier chapitre) de ce travail.

Le premier chapitre décrit le modèle mathématique utilisé et fournit une série d'algorithmes de coloration de graphes ainsi que plusieurs résultats théoriques. Ce modèle sera modifié au deuxième chapitre afin d'inclure les contraintes rencontrées en pratique; nous étudions en plus quelques méthodes de calcul supplémentaires et la structure des données et contraintes relatives à l'horaire. Des résultats numériques de simulation complèteront l'étude théorique des algorithmes de coloration. Le troisième chapitre analysera la structure et les différentes parties du programme (exécution, impression de résultats, gestion de fichiers, documentation). Une série de résultats pratiques seront fournis au quatrième chapitre. Une conclusion finale et quelques annexes clôturent ce travail.

C H A P I T R E I

Chapitre I : L'horaire de cours comme modèle de graphe

Nous décrivons un modèle de graphe ainsi qu'une série d'algorithmes de coloration de graphes. Quelques bornes théoriques du nombre chromatique et des résultats numériques complèteront ce chapitre.

I.1. Définition du problème et mise en modèle

Le problème consiste à calculer l'horaire de cours d'un établissement scolaire quelconque. Etant donné un ensemble de cours, il s'agit de trouver une heure de la semaine pour chaque cours à donner tout en respectant les contraintes évidentes du type "aucune classe ne peut avoir deux cours différents à la même heure". Donc, il faut s'arranger de façon à ce qu'aucune classe, aucun professeur et aucun local ne soit pris par plus d'un cours à la même heure : ce sont les contraintes indirectes du problème. Ce premier chapitre décrit un modèle approprié qui sera modifié au deuxième chapitre en ajoutant d'autres contraintes de nature plutôt pédagogique (dites contraintes directes).

I.1.1. Position du problème

Soient donnés les ensembles suivants :

- les classes $\mathcal{C} = \{c_i \text{ tq } i \in I_1\}$
- les professeurs $\mathcal{P} = \{p_j \text{ tq } j \in I_2\}$
- les locaux $\mathcal{L} = \{l_k \text{ tq } k \in I_3\}$

Un cours à donner se présente comme un sous-ensemble de $\mathcal{C} \cup \mathcal{P} \cup \mathcal{L}$ c-à-d c'est un ensemble de classes, professeurs et locaux qui y participent.

Exemple : Le cours de la classe c_1 avec le professeur p_1 dans le local l_1 se présente sous la forme $\{c_1,p_1,l_1\}$. Si les classes c_1 et c_2 participent au cours donné par p_2 en l_3, alors ce cours se présente comme $\{c_1,c_2,p_2,l_3\}$. Lorsqu'on ne sait pas dans quel local un certain cours doit avoir lieu, on ne le précise pas : c_3 a cours avec p_4, ce qui donnera $\{c_3,p_4\}$.

Soit X l'ensemble des cours à donner et x un élément quelconque de X.
Donc, $X \in \{x \text{ tq } x \text{ est sous-ensemble de } \mathcal{C} \cup \mathcal{P} \cup \mathcal{L}\}$.
Le calcul de l'horaire de cours revient donc à donner une heure à chaque cours x.

I.1.2. Un modèle de graphe

Soit donné un ensemble $E = \mathcal{C} \cup \mathcal{P} \cup \mathcal{L}$

$X \subseteq \{x \text{ tq } x \text{ sous-ensemble de } E\}$ où X est l'ensemble des sommets

$U = \{(x,y) \text{ tq } x,y \in X,\ x \neq y,\ x \cap y \neq \phi\}$ où U est l'ensemble des arcs

$H = \{h \text{ tq } h \in H\}$ l'ensemble des couleurs

A un cours à donner correspond un sommet du graphe. Deux sommets du graphe sont liés par un arc (on dit qu'ils sont adjacents) si les deux cours correspondants possèdent au moins un élément (c-à-d classe, professeur ou local) commun. Dans la suite on raisonnera sur ce graphe $\mathcal{G}(X,U)$. Le calcul d'horaire de cours se ramène ainsi à donner une couleur à chaque sommet du graphe (donner une heure à chaque cours) tout en respectant la contrainte que deux sommets adjacents (deux cours possèdant un élément en commun) ne puissent pas prendre la même couleur. A chaque couleur correspond une heure.

Finalement, on est ramené à un problème de <u>coloration de graphe</u> :

{ Colorer les sommets de façon à ce que deux sommets adjacents ne prennent pas la même couleur.

Le nombre chromatique $\chi(G)$ est par définition le nombre minimal de couleurs nécessaires pour colorer G. Le fait que le sommet x prend la couleur h se note par f(x)=h : l'application "coloration" associe à chaque sommet une couleur; f : X→H : x→f(x)=h.

Remarques :

- Ce modèle possède des avantages importants :
  + il permet de traiter les cours à classes multiples ($\{c_1, c_2, p_2, l_1\}$) ainsi que les cours à option ($\{c_1, p_1, p_2, l_1, l_2\}$, ce cours reprend alors deux cours d'intitulés différents).
  + tous les éléments du cours ne doivent pas être spécifiés s'il y a des incertitudes concernant la configuration du cours.
- Les graphes à colorer sont simples (sans boucles ni arcs multiples).
- Ce modèle de graphe se compliquera encore lors de l'ajout des contraintes dites <u>directes</u> (congés de classe,...). Ces modifications seront étudiées dans le deuxième chapitre.
- Les cours d'un même intitulé sont considérés de façon indépendante, chacun comme un cours différent.

  exemple : Le cours de physique se donne pendant trois heures en semaine et a la forme $\{c_2, p_1, l_4\}$. On le traite alors comme trois cours différents x_1, x_2 et x_3; $x_1 = \{c_2, p_1, l_4\}$; $x_2 = \{c_2, p_1, l_4\}$; $x_3 = \{c_2, p_1, l_4\}$.

I.2. Méthodes de coloration séquentielle

Ce paragraphe traite une série de méthodes de coloration dite séquentielle en étudiant leur principe ainsi qu'en donnant une borne supérieure du nombre chromatique obtenue par ces méthodes.

Pour illustrer un premier principe général, examinons l'exemple suivant.

Soient $E = \{c_1, c_2, c_3, c_4, p_1, p_2, p_3, p_4, p_5, l_1, l_3, l_4\}$

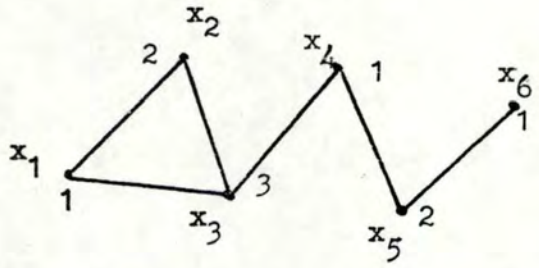
$X = \{x_1, x_2, x_3, x_4, x_5, x_6\}$

où $x_1 = \{c_1, p_1, l_1\}$; $x_2 = \{c_2, p_2, l_1\}$; $x_3 = \{c_1, c_2, p_3, l_3\}$

$x_4 = \{c_3, p_4, l_3\}$; $x_5 = \{c_3, p_4, p_5, l_4\}$; $x_6 = \{c_4, p_5, l_4\}$

$U = \{(x_1, x_2), (x_1, x_3), (x_2, x_3), (x_3, x_4), (x_4, x_5), (x_5, x_6)\}$

$H = \{1, 2, 3, 4, \ldots\}$



Pour colorer ce graphe, traitons les sommets dans l'ordre $x_1, x_2, \ldots, x_6$:

{examiner le sommet x_i revient à lui donner la première couleur possible et interdire cette couleur pour ses adjacents.

Ce raisonnement mène à la coloration suivante :

- donner la couleur 1 à x_1 ; interdire la couleur 1 pour x_2 et x_3.
- " 2 à x_2 ; " 2 pour x_1 et x_3.
- " 3 à x_3 ; " 3 pour x_1, x_2 et x_4.
- " 1 à x_4 ; " 1 pour x_3 et x_5.
- " 2 à x_5 ; " 2 pour x_4 et x_6.
- " 1 à x_6 ; " 1 pour x_5.

Ainsi, on obtient la coloration 1,2,3,1,2,1 c-à-d qu'il faut 3 couleurs pour colorer le graphe. Si on avait considéré l'ordre x_3,x_1,x_2,x_4,x_5,x_6, on aurait trouvé la coloration 2,3,1,2,1,2.
Le résultat de la coloration dépend donc fortement de l'ordre dans lequel les sommets sont examinés. Les méthodes proposées dans la suite diffèrent selon leur critère d'ordonnancement; les sommets sont colorés l'un après l'autre selon un ordre bien déterminé et établi avant la coloration : méthodes de coloration séquentielle.

I.2.1. Deux bornes du nombre chromatique

Le théorème suivant (BROOKS,[1]) constitue un résultat fondamental de limitation du nombre chromatique d'un graphe. Pour une explication eventuelle des concepts utilisés, consultez l'annexe 1.

Théorème 1 : Soit G(X,U) un graphe simple; s'il existe un nombre m tel que $\max_{x_i \in G} d_i \leq m$; $m > 2$, et que G n'a pas de m+1-clique, alors, $\chi(G) \leq m$.

Avant de démontrer ce résultat très important, analysons quelques résultats théoriques.
Sans perte de généralité, G peut être supposé connexe (sinon le théorème peut être démontré pour chaque composante connexe de G). Comme G ne possède pas de m+1-clique, il n'est pas complet. On peut évidemment colorer G avec les couleurs $\emptyset,1,2,\ldots,m$ en donnant à chaque sommet une couleur non prise par un sommet adjacent. Supposons donc G coloré avec m+1 couleurs : on essaye dans la suite d'obtenir une coloration avec m couleurs.
Les opérations suivantes peuvent être faites sans que deux sommets adjacents ne prennent la même couleur :

(1) Un sommet de degré inférieur ou égal à m-1 peut être recoloré avec une couleur différente de $\emptyset$ (on dit qu'on recolore non-$\emptyset$).

En effet, il reste toujours une couleur libre, différente de $\emptyset$ par permutation éventuelle de couleurs. En particulier, un sommet possèdant 2 adjacents de même couleur peut être recoloré non-$\emptyset$.

(2) Si x et y sont adjacents, on peut les recolorer sans changer les couleurs d'autres sommets de façon à ce que x soit coloré non-$\emptyset$. En effet, en négligeant l'arc (x,y), on peut recolorer x non-$\emptyset$ (car degré de $x \leq m-1$ et en appliquant (1)); en fonction de cette coloration de x, y peut être recoloré (peut-être avec $\emptyset$).

(3) Soit x,x',x",...,y une chaîne, alors on peut recolorer x,x',...,y de façon successive, sans changer les couleurs d'autres sommets, de façon à ce qu'au plus y prenne la couleur $\emptyset$. Il suffit pour cela d'appliquer (2) récursivement en commençant en début de chaîne (traiter de gauche à droite chaque paire de sommets et lui appliquer (2) ou ne rien faire).

Dès lors, dégageons les corollaires suivants :

Corollaire 1 : Si G est fini et y un sommet arbitraire de G. Puisqu'il y a une chaîne arbitraire liant y à tout sommet x de G, on peut recolorer G avec au plus y coloré $\emptyset$.

En effet, il suffit d'appliquer (3) à chaque chaîne liant y et tout sommet x de G.

Corollaire 2 : Soit G un graphe infini et G' une partie connexe finie de G et y un sommet non contenu dans G', mais adjacent à un sommet de G'. Alors, on peut recolorer G' et y tels qu'aucun sommet de G' soit coloré $\emptyset$.

En effet, le résultat s'obtient en appliquant (3) à G' et y.

Procèdons maintenant à la démonstration du théorème 1.

Démonstration : Par hypothèse, on dispose d'une m+1-coloration de G. Nous allons établir une m-coloration de G en utilisant les

résultats décrits précédemment ce qui démontre alors la thèse. La couleur qu'on élimine sera $\emptyset$ (toute autre couleur aurait pu être prise). Pour un graphe G quelconque vérifiant les hypothèses du théorème, trois cas peuvent se présenter. Pour chacun d'eux, recolorons G non-$\emptyset$ de façon à obtenir une m-coloration de G.

Cas 1 : Si un sommet quelconque x de G est de degré inférieur ou égal à m-1, on pourra trouver une m-coloration de G. En effet, soit une m+1-coloration de G et x le seul sommet possèdant la couleur $\emptyset$ (possible par le corollaire 1), alors par (1), on peut colorer x non-$\emptyset$ et G sera m-coloré.

Soient deux sommets quelconques x et y du graphe G. Comme G n'est pas complet, on peut trouver les deux sommets x et y non adjacents. Par le corollaire 1, soit G m+1-coloré tel que seul y soit coloré $\emptyset$. Donc, x et ses adjacents sont colorés non-$\emptyset$. Alors, ou bien le degré de x est inférieur ou égal à m-1 (et G peut être m-coloré par le cas 1), ou bien il y a deux sommets adjacents à x de même couleur (car le degré de x vaut m et lui et ses adjacents sont colorés 1,2,...,m, donc au moins deux de ses adjacents prennent la même couleur). Deux cas sont alors possibles :

Si, cas 2, pour tout quadruplet x,y,a,b de sommets différents, il y a une chaîne de x à y ne contenant ni a, ni b. Si cela est vérifié pour tout a et b, il le sera aussi pour les deux adjacents de x de même couleur. Dès lors, par (3), G peut être recoloré sans changer les couleurs de a et b tel qu'au plus x soit coloré $\emptyset$. Comme les adjacents de x, a et b, ont la même couleur, x peut être recoloré non-$\emptyset$ (par (1)). Donc, G est m-coloré.

Sinon, cas 3, il y a quatre sommets différents x,y,a,b tels que toute chaîne de x à y passe par a ou b. Considérons alors les graphes suivants :

$G_1(X_1,U_1)$ où X_1 = x et tout sommet lié à x par une chaîne ne passant ni par a, ni par b comme point intermédiaire.

$$U_1 = U \cap (X_1 \times X_1)$$

$G_2(X_2,U_2)$ où X_2 = a,b et tout sommet de X non contenu dans X1.
$U_2 = U \cap (X_2 \times X_2)$

Par construction, G_1 et G_2 sont connexes, non nuls dont la réunion donne G (il n'y a pas de perte d'arcs, ni de sommets par définition de G_1 et G_2, ainsi que par l'hypothèse du cas 3) et dont l'intersection est au moins a ou b et au plus a et b ainsi que l'arc (a,b).

Soit m_i = nombre d'arcs de G_i contenant a (nombre d'adjacents de a dans G_i)
m_0 = nombre d'arcs (a,b) (qui vaut $\emptyset$ ou 1)

Donc, $m_1 + m_2 \leq m_\emptyset + m$ (*)

En plus, le degré maximal sur G_1 et G_2 ne dépasse pas m.

Trois sous-cas du cas 3 peuvent se présenter :

Cas 3.1 : Supposons que l'intersection de G_1 et G_2 contienne exactement un sommet, soit a. Comme $a,x \in G_1$ et $a,b,y \in G_2$, mais $x \notin G_2$ et $b,y \notin G_1$, on a que toute chaîne de b à x contient a comme point intermédiaire et qu'il existe une chaîne de a à x ne contenant pas b comme point intermédiaire. Dès lors, ou bien $(a,b) \in U$, ou bien $(a,y') \in U$, $y' \in G_2$. Mais en tout cas, $(x',a) \in G_1$, $x' \notin G_2$ et x' fait partie de la chaîne de a à x ne contenant pas b (x' peut être égal à x); et $(a,y') \in G_2$, $y' \notin G_1$ et y' fait partie de la chaîne de x à b contenant a comme point intermédiaire (y' peut être égal à b). Ceci est illustré aux figures 1 et 2.

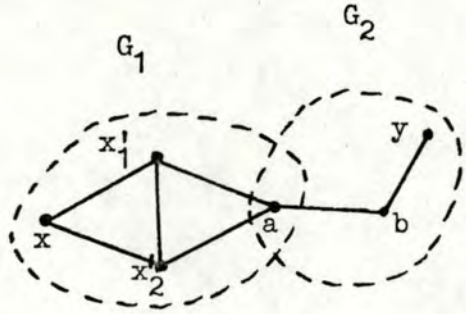


figure 1 : $(a,b) \in U$
$(a,x'_j) \in U$, $x'_j \in G_1$

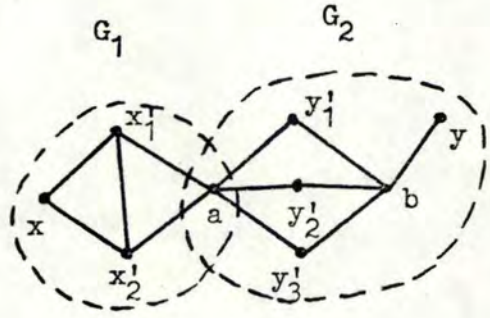


figure 2 : $(a,y_i) \in U$, $y'_i \in G_2$
$(a,x'_j) \in U$, $x'_j \in G_1$

Par conséquent, le degré de a sur G_1 et G_2 ne dépasse pas m-1. Par le cas 1, on peut colorer G_1 et G_2 non-$\emptyset$. En permutant les couleurs de G_2 telles que les couleurs de a dans G_1 et G_2 coïncident, alors G est m-coloré.

Cas 3.2 : Si $G_1 \cap G_2 = \{a,b\}$ et un de G_1, G_2 (soit G_1) est tel que si l'arc (a,b) est ajouté, il devient une m+1-clique (raisonnement analogue pour G_2 par (*)). G_1 peut être m-coloré en donnant des couleurs arbitraires aux m-1 sommets différents et la couleur restante à a et b.
Par (*), $m + m_2 \leqslant 1 + m$, donc $m_2 \leqslant 1$ c-à-d $\deg_{G_2}(a) \leqslant 1$. Le même raisonnement pour b implique que $\deg_{G_2}(b) \leqslant 1$. Comme $m \geqslant 3$, le degré de a et b sur G_2 est inférieur ou égal à m-1. Par le cas 1, on déduit que le graphe résultant c-à-d G_2 peut être m-coloré avec a et b de même couleur. Cette couleur peut être choisie de façon à avoir la même couleur de a et b dans G_1 et G_2. Finalement, G est m-coloré.

Cas 3.3 : Si $G_1 \cap G_2 = \{a,b\}$ et ni G_1, ni G_2 deviennent une m+1-clique par l'ajout de l'arc (a,b) (il se peut que (a,b) existe déjà avant). Supposons qu'ils deviennent G_1' et G_2' par cet ajout. Alors, G_1' et G_2' contiennent moins de sommets que G et par (*) leur degré maximal est inférieur ou égal à m. Si G_1' et G_2' sont m-colorables, alors G l'est aussi. En effet, l'existence de l'arc (a,b) entraîne que a doit avoir une couleur différente de b dans les deux colorations. Par permutation des couleurs de G_1', on trouvera les mêmes couleurs de a et b qu'en G_2' et finalement une m-coloration de G.

Donc, si G est un graphe connexe dont le degré maximal ne dépasse pas m et qui ne possède pas de m+1-clique, alors

soit G est m-colorable,

soit G est m-colorable si deux graphes satisfaisant les mêmes conditions, mais ayant moins de sommets, le sont.

Or, il est évident que le théorème est vrai pour un graphe à moins de quatre sommets. D'où par induction sur le nombre de sommets, G est toujours m-colorable. ∎

Remarquons que les cas 3.2 et 3.3 possèdent une différence fondamentale : dans le cas 3.2, on trouve (par le processus de décomposition de G_1 et G_2 en G_1' et G_2') des graphes ne satisfaisant plus les hypothèses de départ

de G (qui lui ne possèdait pas de m+1-clique) et on ne peut donc pas y continuer le raisonnement commencé sur G.
Dans le cas 3.3, les décompositions de G en G_1' et G_2' qui vérifient les hypothèses du théorème continuent le raisonnement fait sur G. La manière de construire G_1' et G_2' (moins de sommets que G, avec l'arc supplémentaire (a,b)) mènera (en l'appliquant plusieurs fois) soit à des graphes à moins de quatre sommets (théorème évident), soit à des graphes satisfaisant le cas 3.2. Cette stratégie de décomposition pourrait d'ailleurs faire l'objet d'une technique de coloration de graphe qu'on n'étudiera pourtant pas dans ce travail.

Le théorème 1 peut être reformulé de la façon suivante :

$$\chi(G) \leq 1 + \max_{x_i \in G} d_i \qquad \text{(I)}$$

avec égalité si et seulement si G est complet ou G possède une m+1-clique.

Voyons une autre variante de (I) qui est due à WILF [2].

Théorème 2 : $\chi(G) \leq 1 + \lambda$ (II)

où $\lambda = \lambda(G)$ est la valeur propre de plus grande valeur de M, matrice d'adjacence de G, c-à-d

$$M(i,j) = \begin{cases} 1 \text{ si } (i,j) \in U \\ \emptyset \text{ si } (i,j) \notin U \end{cases}$$

Démonstration : La matrice d'adjacence du graphe G, M, est symétrique réelle, donc il existe une matrice unitaire U (c-à-d $U' = U^{-1}$ (a)) telle que $U' M U = \text{diag}(\lambda_i)$ où λ_i est une valeur propre de M. On a par conséquent que

$$U' M U = \begin{pmatrix} \lambda_1 & & 0 \\ & \lambda_2 & \\ 0 & & \ddots \lambda_n \end{pmatrix} \qquad \text{(b)}$$

Pour un indice i, $\deg(x_i) = d_i = \sum_{j=1}^{n} M(i,j)$. Soit $\lambda = \max_{1 \leq i \leq n} \lambda_i$, n = nombre de sommets du graphe G.

Montrons que $\max_{1 \leq i \leq n} d_i \leq \lambda$, alors par (I) on conclut la thèse.

Soit $d_k = \max_{1 \leq i \leq n} d_i$, donc

$$(\emptyset \ldots \emptyset 1 \emptyset \ldots \emptyset)\ M \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix} = d_k \quad (c)$$

(1 en k-ième position)

Or, par (b)

$$(\emptyset \ldots \emptyset 1 \emptyset \ldots \emptyset)\ \mathbf{U'}\ M\ U \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix} = (\emptyset \ldots \emptyset 1 \emptyset \ldots \emptyset)\ \mathrm{diag}(\lambda_i) \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix}$$

$$= (\emptyset \ldots \emptyset \lambda_k \emptyset \ldots \emptyset) \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix} = \lambda_k .$$

Comme $\lambda \geq \lambda_k$, il suffit de voir que $\lambda_k = d_k$ (d) pour prouver (II).

Soit $v = \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix}$, alors (d) se ramène à $U'\ M\ U\ v = M\ v$

c-à-d par (a) $U^{-1} M\ U\ v = M\ v$

ou encore $M\ U\ v = U\ M\ v$.

Pour avoir l'égalité de ces deux vecteurs, il suffit de la prouver pour chaque composante :

pour la k-ième composante $\left(M\ U\ v\right)_k = \left(\sum_{j=1}^{n} (M\ U)_j\ v_j\right)_k$

$$= \left(\sum_{j,i=1}^{n} M(j,i)\ U_i\ v_j\right)_k$$

où U_i est la i-ième ligne de U.

De même, $\left(U\ M\ v\right)_k = \left(\sum_{i=1}^{n} U_i\ (M\ v)_i\right)_k$

$$= \left(\sum_{i,j=1}^{n} U_i\ M(i,j)\ v_j\right)_k .$$

Par la symétrie de M, on a que M(i,j) = M(j,i), d'où le résultat. ∎

D'autres bornes du nombre chromatique se trouvent dans [14], [15], [17], [27] et [28]. Nous décrivons dans la suite des méthodes séquentielles dont l'ordonnancement des sommets se base sur des critères différents (degré, vecteur propre de M, nombre d'éléments) tout en dégageant les algorithmes de coloration séquentielle correspondants.

I.2.2. Ordonnancement par degré

Soit le graphe simple G(X,U) à n sommets dont le degré d'un sommet x_i est égal au nombre de ses sommets adjacents (noté d_i).
Nous allons dans la suite proposer quelques méthodes séquentielles utilisant ce concept de degré.

I.2.2.1. Degré largest-first

L'idée de l'algorithme est d'ordonner les sommets selon le critère de degré (priorité aux sommets de degré élevé) et de les colorer successivement dans cet ordre.

Algorithme LF :

1) Ordonner les sommets tels que $d_1 \geqslant d_2 \geqslant \ldots \geqslant d_n$.
2) $\alpha = \emptyset$; $i = 1$
3) Si $i > n$, arrêter.
   Sinon, colorer le sommet i avec la première couleur possible, soit k.
   si $k > \alpha$, $\alpha = k$
4) Interdire la couleur k pour les adjacents de i.
   $i = i + 1$; aller en 3).

A la fin de l'algorithme, α contiendra le nombre de couleurs utilisées lors de la coloration selon la méthode LF (et dès lors une borne supérieure du nombre chromatique).

Analysons de façon plus détaillée la démarche des méthodes séquentielles et dégageons une borne supérieure du nombre chromatique pour la méthode LF.

Soit X_1 le sous-ensemble de sommets de X défini récursivement de la façon suivante : $- x_1 \in X_1$

$- \text{si } \{x_{i_k}\}_{k=1}^{m} \in X_1$, où $i_1 = 1$ et $i_1 < i_2 < \ldots < i_m$
c-à-d $d_{i_1} \geqslant d_{i_2} \geqslant \ldots \geqslant d_{i_m}$ (pour LF),

alors, $x_j \in X_1 \iff \begin{cases} j > i_m \\ x_j \text{ non-adjacent à tout élément de } \{x_{i_k}\}_{k=1}^{m} \end{cases}$.

Par construction, X_1 est fini, non-vide, contenu dans X.
Similairement, définissons X_2 comme sous-ensemble de $X - X_1$ de la façon suivante : - soit i le plus petit entier tel que $x_i \notin X_1$, alors $x_i \in X_2$.
- si $\{x_{j_k}\}_{k=1}^{p} \in X_2$, $\begin{cases} \text{alors } x_l \in X_2 \text{ si et seulement si } l > j_p \text{ et} \\ x_l \text{ non-adjacent à tout élément de } \{x_{j_k}\}_{k=1}^{m} \end{cases}$.

Notons que X_2 peut être vide (si G ne contient que des sommets isolés).
De façon analogue, on définit $X_i \subseteq X - (\bigcup_{j=1}^{i-1} X_j)$ de manière récursive.

On construit ainsi une séquence $\{X_i\}$ de sous-ensembles disjoints de X telle que $X = \bigcup_{i=1}^{\infty} X_i$ et il existe un entier fini $\alpha = \alpha(G)$ tel que $X_r = \phi$ pour $r > \alpha(G)$. Par construction, non deux sommets d'un même ensemble X_j sont adjacents et donc, par assignation d'une couleur j à l'ensemble X_j, on peut colorer le graphe G en $\alpha(G)$ couleurs.

Ce raisonnement mène au théorème suivant (Powell, Welsh [3]) :

Théorème 3 : Par l'algorithme LF, on obtient que

$$\chi(G) \leq \alpha(G) \leq \max_{1 \leq i \leq n} \{\min(d_i + 1, i)\} \qquad \text{(III)}$$

Démonstration : Par construction, chaque sommet dans $X - X_1$ est adjacent à au moins un sommet de X_1. Par induction, un sommet de $X - \bigcup_{j=1}^{m} X_j$ est adjacent à au moins un sommet de chaque X_j, $1 \leq j \leq m$.
Par conséquent, comme $d_k < m \Rightarrow x_k \in \bigcup_{j=1}^{m} X_j$, on a que

$$x_k \notin \bigcup_{j=1}^{m} X_j \Longrightarrow d_k \geq m \quad \text{et donc,} \quad x_i \in \bigcup_{j=1}^{d_i+1} X_j \qquad \text{(a)}.$$

D'autre part, par construction des X_j, $x_i \in \bigcup_{j=1}^{i} X_j$ (b).

En combinant (a) et (b), on obtient que $x_i \in \bigcup_{j=1}^{\Phi(i)} X_j$ où $\Phi(i) = \min(i, d_i+1)$.
Soit $\delta(G) = \max_i \Phi(i)$; donc $X_{\delta(G)+1} = \phi$ ce qui implique $\alpha(G) \leq \delta(G)$ et prouve le résultat (III).

■

La différence $\alpha(G) - \chi(G)$ peut pourtant être assez grande surtout pour des graphes de grande dimension.
L'algorithme LF fournit donc la borne du nombre chromatique :

$$\chi(G) \leq \max_{1\leq i\leq n} \{\min(d_i+1, i)\} \qquad \text{(IV)}$$

Une preuve non-constructive (sans fournir l'algorithme correspondant) de ce résultat a été trouvée par Bondy [4].

I.2.2.2. Degré smallest-last

Les bornes (I) et (IV) peuvent être améliorées considérablement par d'autres méthodes de coloration séquentielle. Les méthodes séquentielles **LF** vérifient le fait suivant :

> soient les sommets $x_1, x_2, \ldots, x_n$ de G, alors l'algorithme de coloration correspondant n'utilise pas plus de couleurs que
>
> $$\alpha = \max_{1\leq i\leq n} \{1 + \deg_{\langle x_1,\ldots,x_i\rangle}(x_i)\} \quad \text{(a) où } \deg_{\langle x_1,\ldots,x_i\rangle} \text{ est le}$$
>
> degré par rapport au sous-graphe engendré par $\{x_1,\ldots,x_i\}$.

Un moyen pour déterminer un ordonnancement minimisant α dans (a) est fourni par la méthode suivante (Matula [5]) :

(*) 1) soit x_n tel que $d_n = \min_{1\leq i\leq n} d_i$
2) pour $i = n-1, n-2, \ldots, 2, 1$ soit x_i le sommet de degré minimal sur $X - \{x_n, x_{n-1}, \ldots, x_{i+1}\}$.

L'ordre ainsi imposé sur les sommets est

$$\deg_{\langle x_1,\ldots,x_i\rangle}(x_i) = \min_{1\leq j\leq i} \{\deg_{\langle x_1,\ldots,x_j\rangle}(x_j)\} \quad 1\leq i\leq n \quad \text{(b)}$$

Cet ordre est appelé ordre smallest-last, d'où

algorithme SL : 1) (*)
2), 3), 4) } Voir algorithme LF

L'algorithme SL recalcule le degré sur les sommets non-ordonnés (ce qu'on aurait d'ailleurs pu faire aussi avec le critère largest-first qui lui calcule le degré sur tous les sommets) et s'avère plus performant que l'algorithme LF (voir I.7.).
Le fait que l'ordre SL minimise α dans (a) parmis les n! ordonnancements possibles a été démontré par Matula [5].
La différence essentielle entre l'ordre LF et l'ordre SL est que le calcul de degré-LF se fait sur le graphe entier tandis que SL n'est appliqué qu'au sous-graphe restant.

Par construction, l'algorithme SL fournit la borne du nombre chromatique suivante :

Théorème 4 : Pour tout graphe G

$$\chi(G) \leqslant 1 + \max_{H}\left[\min_{x \in X(H)}\left\{\deg_H(x) \text{ tq } H \text{ sous-graphe de } G\right\}\right] \quad (V)$$

Pour démontrer ce résultat (SZEKERES, WILF [7]), on se sert d'un lemme.
Soit $\lambda : G \to \mathbb{R}$ tel que $P_1 : G' \subset G \Rightarrow \lambda(G') \leqslant \lambda(G)$
$P_2 : \lambda(G) \geqslant \min_{x \in G} \deg(x)$

Lemme 1 : Soit (G) une fonction à valeurs réelles sur G satisfaisant les propriétés P_1 et P_2. Alors, $\chi(G) \leqslant \lambda(G) + 1$

Rappelons que le graphe G est χ-critique si et seulement si pour tout sommet x de G le sous-graphe engendré par $G - \{x\}$ possède un nombre chromatique strictement inférieur à celui de G.

Preuve du lemme : Soit $\chi(G)$ le nombre chromatique de G. En enlevant un nombre fini de sommets de G, on trouve un sous-graphe critique $G_c \subset G$ tel que $\chi(G_c) = \chi(G)$. Par P_1, $\lambda(G_c) \leqslant \lambda(G)$, mais G_c ne peut pas contenir de sommets x tels que $\deg(x) < \chi(G) - 1$ (car sinon, par théorème 1, cas 1, G_c peut être recoloré avec $\chi(G)-1$ couleurs). Donc, par P_2, $\chi(G) - 1 \leqslant \min_{x \in G_c} \deg(x) \leqslant \lambda(G_c) \leqslant \lambda(G)$ ce qui prouve le lemme. ∎

Preuve du théorème : Soit $\mu(G) = \min_{x \in G} \deg(x)$.

La fonction $\Lambda(G) = \max_{G' \subset G} \mu(G')$ satisfait évidemment les propriétés P_1 et P_2. Donc, le cas le plus favorable du lemme sera $\chi(G) \leqslant \Lambda(G) + 1 = 1 + \max_{G' \subset G} \left\{ \min_{x \in G} \deg(x) \right\}$ ce qui démontre (V). ∎

A partir de (V), d'autres bornes de $\chi(G)$ peuvent être obtenues en ce qui concerne l'algorithme SL. Pour plus de détails, le lecteur est prié de consulter [5], [6], [8] et [9].

I.2.3. Ordonnancement par valeur propre

La méthode séquentielle étudiée dans ce paragraphe ordonne les sommets du graphe selon le critère de valeur propre : considérons la matrice d'adjacence M définie au théorème 2 et calculons le vecteur propre associé à la valeur propre de plus grande valeur; ordonnons ensuite les sommets suivant les composantes de ce vecteur.

On sait que M (qui est symétrique) possède n valeurs propres $\lambda_1, \ldots, \lambda_n$ réelles. Les vecteurs propres $z_1, z_2, \ldots, z_n$ sont linéairement indépendants et tout vecteur $v_\emptyset \in R^n$ peut s'exprimer $v_\emptyset = \sum_{i=1}^{n} \alpha_i z_i$. En multipliant m fois par M, $v_m = M^m v_\emptyset = \sum_{i=1}^{n} \alpha_i \lambda_i^m z_i$. Ordonnons les valeurs propres de M de telle façon à ce que $|\lambda_1| \geqslant |\lambda_2| \geqslant \ldots \geqslant |\lambda_n|$ et supposons λ_1 dominante (c-à-d $|\lambda_1| > |\lambda_2|$).

On peut ainsi mettre v_m sous la forme $v_m = \lambda_1^m \left[\alpha_1 z_1 + \sum_{i=2}^{n} \left(\frac{\lambda_i}{\lambda_1} \right)^m \alpha_i z_i \right]$.

D'où, si $\alpha_1 \neq \emptyset$, $\lim_{m \to \infty} v_m = \lambda_1^m \alpha_1 z_1$

et pour m grand, $v_{m+1} = M v_m = \lambda_1^{m+1} \alpha_1 z_1 \sim \lambda_1 v_m \Longrightarrow \lambda_1 \cong \frac{v_{m+1}}{v_m}$.

Donc, v_m converge vers z_1 (vecteur propre associé à la valeur propre dominante) à un coéfficient près qui est sans importance puisque les vecteurs propres ne sont définis qu'à une constante multiplicative près (normaliser). Cette méthode de recherche de vecteur propre est dite méthode de la puissance.

Algorithme VP :

1) a) $v_{\emptyset} = (1\emptyset\emptyset\ldots\emptyset)$; $m = \emptyset$
   b) $m = m + 1$
   c) $delta = \sum_{i=1}^{n} v_m^2(i)$
   d) $v_{m+1} = M \cdot v_m / \sqrt{delta}$ c-à-d $v_{m+1}(j) = \sum_{i=1}^{n} M(j,i) \cdot v_m(i) / \sqrt{\sum_{i=1}^{n} v_m^2(i)}$
   e) si $|v_{m+1} - v_m| \geqslant \varepsilon$, aller en b)
      sinon, $v = v_{m+1}$ est le vecteur propre associé à la valeur propre de plus grand module de M.
   f) Ordonner les x_i selon les composantes de v.

2) } Voir algorithme LF
3) }
4) }

I.2.4. Ordonnancement par nombre d'éléments

Soit G(X,U), $x \in X$ un sommet du graphe c-à-d $x \subset E$. Chaque sommet étant un sous-ensemble de E, on peut considérer son nombre d'éléments. Le nombre d'éléments du sommet x se note $\#x$ (cardinal de x). Les sommets vont être ordonnés selon cette notion de nombre d'éléments de façon à ce que $\#x_i \geqslant \#x_{i+1}$.

Cette notion est de nature plutôt pratique : chaque sommet représente en réalité un cours à donner; on favorise dès lors les cours contenant le plus d'éléments (cours à classes multiples par exemple) ce qui semble être une bonne stratégie en pratique lors de l'établissement de l'horaire de cours.

Algorithme NELE : 1) Ordonner les sommets selon leur nombre d'éléments

2) }
3) } Voir algorithme LF
4) }

Comparaison des méthodes LF, SL et NELE sur un exemple :

Soit $E = \{c_1, c_2, p_1, p_2, p_3, l_1, l_2\}$

$X = \{x_1, x_2, x_3, x_4, x_5\}$ où $x_1 = \{c_1, p_2\}$; $x_2 = \{c_1, c_2, p_1, l_1\}$

$x_3 = \{c_1, p_3, l_2\}$; $x_4 = \{c_2, p_1, l_2\}$; $x_5 = \{c_2\}$

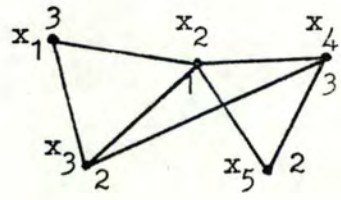
$U = \{(x_1,x_2), (x_1,x_3), (x_2,x_3), (x_2,x_4), (x_2,x_5), (x_3,x_4), (x_4,x_5)\}$

$H = \{1, 2, 3, \ldots\}$

On obtient les colorations suivantes :

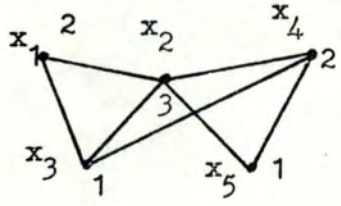
1) LF : ordre : x_2, x_3, x_4, x_1, x_5

coloration : $f(x_1) = 3$
$f(x_2) = 1$
$f(x_3) = 2$
$f(x_4) = 3$
$f(x_5) = 2$



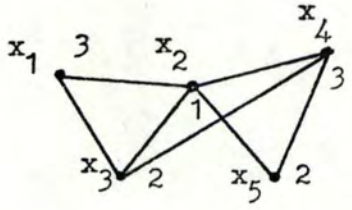
2) SL : ordre : x_5, x_4, x_2, x_3, x_1

coloration : $f(x_1) = 2$
$f(x_2) = 3$
$f(x_3) = 1$
$f(x_4) = 2$
$f(x_5) = 1$



3) NELE : ordre : x_2, x_3, x_4, x_1, x_5

coloration : $f(x_1) = 3$
$f(x_2) = 1$
$f(x_3) = 2$
$f(x_4) = 3$
$f(x_5) = 2$



Comme c'est le cas pour cet exemple, on constate souvent en pratique un comportement analogue des méthodes LF et NELE.

Remarque : Pour les méthodes séquentielles étudiées dans ce chapitre, l'ordonnancement se fait une fois pour toutes avant de colorer le premier sommet; leur complexité est dès lors $O(n)$ (avec des changements suivant le critère utilisé).

Nous avons donc examiné quatre méthodes de coloration séquentielle (LF,SL, VP,NELE) ainsi qu'une série de bornes supérieures du nombre chromatique. Pour plus de résultats théoriques (graphes k-partis, planaires) sur les méthodes séquentielles, le lecteur est referé à [8], [10],[29] et [30]. D'autres méthodes séquentielles seront vues dans le deuxième chapitre.

I.3. Méthodes de saturation

Les méthodes séquentielles ordonnent d'abord les sommets avant de les colorer dans l'ordre établi. Les méthodes de saturation par contre réordonnent les sommets non colorés après coloration d'un sommet quelconque en utilisant les informations relatives aux colorations déjà faites.

Soit le graphe simple G(X,U); définissons le degré de saturation d'un sommet x, noté dsat(x), comme étant le nombre de couleurs adjacentes ç-à-d le nombre de couleurs telles qu'il existe au moins un sommet adjacent à x possèdant cette couleur.

Donc, $dsat(x) = \#\{h \text{ tq } \exists y \in X, (x,y) \in U, f(y) = h\}$

Le principe de l'algorithme dit de saturation est :

> après la coloration d'un sommet quelconque, prendre comme sommet suivant à colorer celui dont le degré de saturation est le plus élevé parmi les sommets non colorés.

Algorithme SAT :

1) Ordonner tous les sommets selon la notion de degré maximal (LF) c-à-d $d_1 \geqslant d_2 \geqslant \ldots \geqslant d_n$.
2) Colorer le sommet x_1 avec la première couleur;
   Interdire cette couleur pour les adjacents de x_1.
3) Prendre le sommet dont le degré de saturation est le plus élevé parmi les sommets non colorés (en cas d'égalité utiliser l'ordre de 1)).
4) Colorer ce sommet avec la première couleur possible et interdire cette couleur à ses adjacents.
5) S'il reste des sommets à colorer, aller en 3).
   Sinon, arrêter.

Etudions un résultat théorique de cet algorithme [11] :

Théorème 5 : L'algorithme SAT donne la valeur exacte de $\chi(G)$ pour les graphes bipartis ($\chi(G) = 2$).

Rappelons qu'un graphe biparti peut être partitionné en deux classes X_1 et X_2 de sorte que deux sommets de la même classe ne soient pas adjacents.

Démonstration : Soit G un graphe biparti connexe avec au moins 3 sommets. Supposons qu'un sommet x a un degré de saturation égal à 2; dans ce cas, x a deux adjacents à couleurs distinctes et on peut construire deux chaînes. Comme G est fini, on peut trouver un sommet commun y des deux chaînes, donc un cycle. Alors, ou bien le cycle est pair (nombre pair d'arcs) et les deux adjacents sont de même couleur ou bien G n'est pas biparti. L'algorithme SAT n'utilise par conséquent pas plus de deux couleurs.

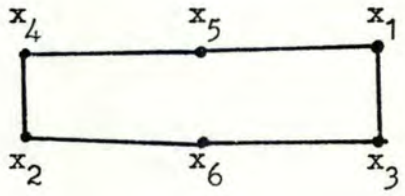
Remarque : Comme on réordonne les sommets après chaque coloration faite, les méthodes de saturation sont caractérisées par une complexité de l'ordre $\mathcal{O}(n^2)$. D'autres méthodes de saturation seront vues au chapitre 2.

I.4. Méthodes de similarité

Les méthodes présentées en I.2 et I.3. colorent chaque sommet de façon individuelle. Nous décrivons dans la suite une méthode dite par similarité de graphe qui colore les sommets par paire en exploitant les informations relatives à la structure du graphe.

I.4.1. Exemple introductif

Soit le graphe suivant :



Comme tous les sommets sont de degré 2, ils vont être ordonnés par LF sous la forme x_1,x_2,x_3,x_4,x_5,x_6; d'où la coloration par LF 1, 1, 2, 2, 3, 3 et donc $\chi(G) \leq 3$.
Pourtant, si on avait rangé les sommets dans l'ordre x_1,x_5,x_4,x_2,x_6,x_3, on aurait trouvé la coloration 1, 2, 1, 2, 1, 2
et par conséquent $\chi(G) \leq 2$.
La notion de similarité permet d'éviter les cas défavorables du premier type.

I.4.2. Similarité de graphe

Soit M la matrice d'adjacence de G.
La matrice de similarité de graphe S est définie par :

$$S = S(i,j)_{1 \leq i,j \leq n} \quad \text{où } S(i,j) = \emptyset \text{ si } M(i,j) = 1 \text{ ou } i = j$$
$$= \sum_{\substack{k=1 \\ \neq i,j}}^{n} M(i,k) \cdot M(j,k) \text{ si } M(i,j) = \emptyset$$

Donc, la similarité de graphe de deux sommets représente le nombre de sommets adjacents aux deux sommets en même temps.
Les sommets sont colorés par paire en favorisant les paires de similarité de graphe élevée ce qui mène à l'algorithme suivant (WOOD [12]) :

Algorithme SIMG :

1) Construire S; soit max la similarité de graphe maximale.
2) Soit $[i,j]$ une paire non traitée telle que $S(i,j) = max$.
   - Si une telle paire existe, aller en 3).
   - Sinon, max = max - 1;
     - si max > Ø, aller en 2)
     - sinon, colorer les sommets restants non colorés par une méthode séquentielle; arrêter
3) Si x_i et x_j colorés, aller en 2).
4) Si un des deux sommets est coloré : soit x_i coloré avec $f(x_i)$ et x_j est non coloré.
   a) Essayer de donner la couleur $f(x_i)$ à x_j.
   b) Si cela réussit, interdire $f(x_i)$ aux adjacents de x_j.
   c) Aller en 2)
5) Si ni x_i, ni x_j est coloré.
   a) Chercher la première couleur possible pour x_i et x_j, soit k.
   b) Interdire k aux adjacents de x_i et x_j.
   c) Aller en 2).

I.4.3. Exemple

Reprenons l'exemple I.4.1. On trouve alors :

M	1	2	3	4	5	6
1	Ø	Ø	1	Ø	1	Ø
2	Ø	Ø	Ø	1	Ø	1
3	1	Ø	Ø	Ø	Ø	1
4	Ø	1	Ø	Ø	1	Ø
5	1	Ø	Ø	1	Ø	Ø
6	Ø	1	1	Ø	Ø	Ø

S	1	2	3	4	5	6
1	Ø	Ø	Ø	1	Ø	1
2	Ø	Ø	1	Ø	1	Ø
3	Ø	1	Ø	Ø	1	Ø
4	1	Ø	Ø	Ø	Ø	1
5	Ø	1	1	Ø	Ø	Ø
6	1	Ø	Ø	1	Ø	Ø

L'algorithme SIMG fournit la coloration $f(x_1) = 1$, $f(x_2) = 2$, $f(x_3) = 2$, $f(x_4) = 1$, $f(x_5) = 2$ et $f(x_6) = 1$ et n'utilise donc pas plus de 2 couleurs.

Remarque : Vu le fait que la méthode de similarité se base essentiellement sur la matrice de similarité de graphe, elle se caractérise par une complexité de l'ordre de $\mathcal{O}(n^2)$.

D'autres méthodes de similarité seront vues au deuxième chapitre.

I.5. Méthodes de permutation

Les méthodes I.2, I.3 et I.4 possèdent la particularité que lorsqu'un sommet ne peut être coloré sans utiliser une nouvelle couleur, on ne change pas les colorations déjà faites et on prend la nouvelle couleur nécessaire. Les méthodes de permutation par contre essayent dans ce cas de permuter des colorations faites afin de libérer une couleur pour le sommet à colorer de façon à ne pas utiliser de nouvelles couleurs.

Soit donné un graphe simple G ayant une k-coloration 1,2,...,k et G_i l'ensemble de sommets de G colorés i. Pour $i \neq j$, le sous-graphe bichromatique i,j est le sous-graphe engendré par $G_i \cup G_j$, noté $\langle G_i \cup G_j \rangle$. Une composante connexe de $\langle G_i \cup G_j \rangle$ est dite une i,j composante. Si les couleurs i et j sont permutées sur une i,j composante du graphe k-coloré G, alors une autre k-coloration de G est obtenue. Cette procédure est appelée une $i \leftrightarrow j$ permutation sur le graphe k-coloré G. Une permutation bichromatique sur un graphe k-coloré G est une $i \leftrightarrow j$ permutation pour $i \neq j$.
La recherche de permutations bichromatiques dans le processus de coloration est à la base de l'algorithme suivant :

Algorithme de permutation :

1) Ordonnancement des sommets du graphe par une des quatre méthodes étudiées en I.2. (LF,SL,VP,NELE).

2) Colorer x_1 avec 1 et éliminer l'heure 1 aux adjacents de x_1 ;
$i = 2$; $j = 1$

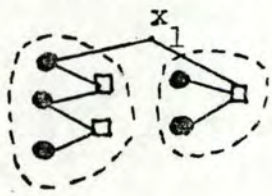
3) - Si $i > n$, arrêter
- Sinon, soit $\langle x_1, x_2, \ldots, x_{i-1} \rangle$ coloré avec $1,2,\ldots,j$
  - si on parvient à colorer x_i avec $1,2,\ldots,j$, colorer x_i et interdire $f(x_i)$ aux adjacents de x_i.
  - sinon, soit $K \subset \{1,2,\ldots,j\}$ l'ensemble des couleurs tel que $\alpha \in K$ entraîne que exactement un sommet de $\langle x_1, \ldots, x_{i-1} \rangle$ adjacent à x_i a la couleur α.
    - Si pour $\alpha, \beta \in K$, $\alpha \neq \beta$, une α,β composante de $\langle x_1, \ldots, x_{i-1} \rangle$ n'a qu'un sommet adjacent à x_i en $\langle x_1, \ldots, x_i \rangle$, alors faire une $\alpha \leftrightarrow \beta$ permutation sur une telle α,β composante de $\langle x_1, \ldots, x_{i-1} \rangle$. Colorer les sommets $x_1, \ldots, x_{i-1}$ de même que dans cette nouvelle coloration et x_i avec la couleur possible, soit α ou β, et une j-coloration de $\langle x_1, \ldots, x_i \rangle$ est obtenue; interdire $f(x_i)$ aux adjacents de x_i.
    - Sinon, une telle permutation n'est pas possible; colorer $x_1, \ldots, x_{i-1}$ de même qu'en $\langle x_1, \ldots, x_{i-1} \rangle$ et colorer x_i avec $j+1$; interdire $j+1$ aux adjacents de x_i. Ainsi, une $j+1$-coloration de $\langle x_1, \ldots, x_i \rangle$ est obtenue; $j = j + 1$

4) $i = i + 1$; aller en 3).

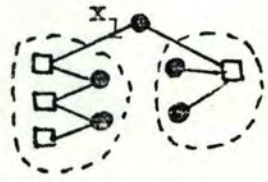
Etudions le principe d'une permutation bichromatique sur un <u>exemple</u> :

Soit x_1 le sommet à colorer ne pouvant ni prendre la couleur i (notée par ●), ni la couleur j (notée par □) et x_i adjacent à exactement <u>un</u> sommet coloré i et <u>un</u> sommet coloré j. Si pour une i,j composante, x_i est adjacent à seulement un sommet (de type ● ou □), une permutation sur cette composante est possible : il suffit d'échanger les couleurs des sommets de cette composante pour que x_1 ne soit adjacent qu'à un type de sommet et puisse donc prendre l'autre couleur.

<u>avant</u> :



<u>après</u> :



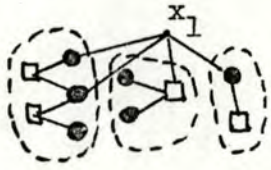
Remarque : Une $i \leftrightarrow j$ permutation peut aussi se réaliser en choisissant les couleurs i et j de la façon suivante :

il faut que toute i,j composante ait des adjacents à x_1 possédant tous la même couleur. Donc, tous les adjacents à x_1 d'une i,j composante quelconque sont soit colorés i, soit colorés j (et ceci pour toute i,j composante).

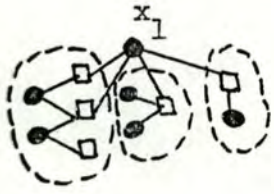
Il suffit dès lors d'échanger les couleurs des sommets d'une série de i,j composantes pour que x_1 ne soit plus adjacent qu'à un type de sommet et puisse donc prendre l'autre couleur. Le principe de choix de i et j décrit dans l'algorithme de permutation en est un cas particulier (exactement un sommet d'une i,j composante est adjacent à x_1).

L'exemple suivant montre une permutation selon ce principe :

avant :



après :



On constate donc un gain de généralité grâce à cette façon de permuter les couleurs. Elle fait d'ailleurs l'objet d'une nouvelle classe de méthodes : méthodes de permutation généralisée. La description exacte de l'algorithme et les performances par rapport aux méthodes de permutation habituelles sont examinées dans l'Annexe 2 de ce travail.

Revenons maintenant à l'algorithme de permutation décrit à la page I.24. Plusieurs méthodes différentes s'obtiennent si dans l'étape 1) de l'algorithme, on applique les quatre méthodes d'ordonnancement proposées dans I.2. :

- méthode LFI (largest-first-interchange)
- méthode SLI (smallest-last-interchange)
- méthode VPI (valeur-propre-interchange)
- méthode NELEI(nombre-éléments-interchange)

Des résultats théoriques concernant les méthodes de permutation sont fournis par MATULA [8] (graphes planaires).

Remarque : Les méthodes de permutation se basent sur les méthodes séquentielles ($\mathcal{O}(n)$), mais elles utilisent le processus de permutation bichromatique et leur complexité varie donc entre $\mathcal{O}(n)$ et $\mathcal{O}(n^2)$.

I.6. Autres méthodes de coloration

Une grande série de méthodes de coloration de graphes sont connues à présent. Nous en mentionnons quelques unes ici; le lecteur est prié de consulter les références citées pour plus d'informations.

- RANDALL - BROWN [13] a développé un algorithme de coloration tout en construisant un arbre de coloration. Cette méthode a été améliorée par SHANNO [11].
- Toutes les méthodes vues jusqu'à présent donnaient une couleur à un sommet donné. TEHRANI [16] par contre donne des sommets à une couleur en question en tenant compte de la structure du graphe (méthode dite "adjacents des adjacents"). Cette même tactique est prise par JOHNSON [24], [25] qui décrit un algorithme dit AMIS (approximately maximum independant set).
- F.T. LEIGTHON développe dans [26] l'algorithme RLF qui est un mélange de LF et AMIS. En plus, il fournit un procédé de construction de graphes à nombre chromatique connu.
- CORNEIL et GRAHAM [18] déterminent le nombre chromatique d'un graphe sur base de réduction de graphes. Les mêmes réflexions sont faites par BERGE ([10], p.316, "principe des reliements-contractions").
- CHRISTOFIDES [19] colore les graphes et détermine leur nombre chromatique par un algorithme basé sur la notion d'ensemble maximal intérieurement stable (voir aussi Berge [10], p.260).
- PLESNIK colore un graphe par la technique de partionnement [31] et DUCHET [32] procède par coloration dite transitive.

I.7. Résultats numériques

On compare les méthodes de I.2. (séquentielles), I.3. (saturation), I.4. (similarité) et I.5. (permutation) en simulant la matrice d'adjacence du graphe. Rappelons que cette matrice est symétrique, à valeurs binaires et à diagonale nulle. Elle sera créee par un générateur de nombres aléatoires (fonction "random") et se caractérise par son taux de remplissage (pourcentage de valeurs "1" de la matrice) ainsi que par sa dimension (nombre de sommets du graphe). Pour les jeux de tests des méthodes en question, on utilise des taux de remplissage de 2Ø % à 8Ø % et des dimensions de 2Ø à 1ØØ.
Pour une dimension et un taux donné, on compare le nombre moyen de couleurs utilisées (obtenu sur 25 essais différents) par les différentes méthodes. Chaque tableau décrit les résultats numériques pour une certaine dimension.

taux de remplissage de la matrice M ←

	2Ø %	3Ø %	4Ø %	5Ø %	6Ø %	7Ø %	8Ø %
VP	3.76	4.91	5.64	6.47	7.54	8.46	1Ø.3Ø
VPI	3.48	4.34	5.37	6.Ø9	7.16	8.25	1Ø.Ø5
LF	3.8Ø	4.72	5.56	6.52	7.52	8.44	9.72
LFI	3.56	4.2Ø	5.Ø8	6.Ø4	6.92	8.Ø4	9.48
SL	3.64	4.36	5.48	6.2Ø	7.28	8.56	1Ø.16
SLI	3.44	4.16	5.Ø4	5.84	6.76	7.96	9.4Ø
SAT	3.52	4.28	5.Ø4	6.Ø4	6.8Ø	8.Ø8	9.52
SIMG	3.76	4.52	5.32	6.36	7.36	8.24	9.68

↑ Méthode utilisée

Tableau 1 : graphe à 2Ø sommets

	2Ø %	3Ø %	4Ø %	5Ø %	6Ø %	7Ø %	8Ø %
VP	6.Ø1	7.12	8.96	1Ø.64	12.Ø7	14.52	17.42
VPI	5.41	6.42	7.94	9.48	11.22	13.49	15.98
LF	5.68	7.12	8.6Ø	1Ø.Ø4	12.2Ø	14.4Ø	16.92
LFI	5.Ø8	6.32	7.72	9.48	11.Ø8	12.96	15.56
SL	5.44	7.Ø4	8.56	1Ø.48	12.2Ø	14.56	17.Ø8
SLI	5.Ø4	6.16	7.72	9.24	1Ø.88	12.72	15.48
SAT	5.Ø4	6.36	7.8Ø	9.56	11.ØØ	13.48	15.92
SIMG	5.8Ø	6.24	8.52	1Ø.ØØ	11.68	13.64	16.56

Tableau 2 : graphe à 4Ø sommets

	2Ø %	3Ø %	4Ø %	5Ø %	6Ø %	7Ø %	8Ø %
VP	7.87	9.51	11.94	14.36	17.Ø5	19.76	23.87
VPI	7.12	8.32	1Ø.65	13.11	15.16	18.53	21.86
LF	6.92	9.12	11.ØØ	13.56	16.24	19.ØØ	22.76
LFI	6.2Ø	8.24	1Ø.16	12.4Ø	14.68	17.56	21.16
SL	7.2Ø	9.24	11.6Ø	14.16	16.48	19.76	23.8Ø
SLI	6.2Ø	8.32	1Ø.28	12.44	14.76	17.2Ø	21.36
SAT	6.Ø8	8.36	1Ø.2Ø	12.48	14.8Ø	17.84	22.2Ø
SIMG	7.4Ø	9.44	11.32	13.52	15.8Ø	18.44	22.12

Tableau 3 : graphe à 6Ø sommets

	2Ø %	3Ø %	4Ø %	5Ø %	6Ø %	7Ø %	8Ø %
VP	9.47	11.62	14.26	17.21	2Ø.89	24.92	3Ø.2Ø
VPI	8.73	1Ø.63	13.19	15.89	19.16	23.Ø1	27.96
LF	8.4Ø	11.Ø8	13.68	16.84	2Ø.24	24.16	28.92
LFI	7.56	1Ø.16	12.64	15.56	18.32	22.44	26.72
SL	8.24	11.4Ø	14.Ø4	17.6Ø	2Ø.44	24.52	29.52
SLI	7.64	9.96	12.68	15.44	18.56	22.12	26.4Ø
SAT	7.52	1Ø.Ø4	12.64	15.6Ø	18.88	22.44	27.28
SIMG	9.Ø8	11.48	13.92	16.76	19.72	23.4Ø	28.Ø8

Tableau 4 : graphe à 8Ø sommets

	2Ø %	3Ø %	4Ø %	5Ø %	6Ø %	7Ø %	8Ø %
VP	11.27	13.45	17.Ø6	2Ø.48	23.8Ø	3Ø.15	36.41
VPI	1Ø.15	12.53	15.82	18.75	22.44	27.5Ø	33.3Ø
LF	9.68	12.88	16.24	19.84	24.Ø4	28.56	34.56
LFI	8.76	11.88	14.84	18.4Ø	21.96	26.28	32.ØØ
SL	9.72	13.28	16.44	2Ø.44	24.Ø4	29.36	35.96
SLI	8.72	11.76	14.88	18.28	22.28	26.24	32.2Ø
SAT	8.76	11.6Ø	14.84	18.36	21.96	26.92	32.92
SIMG	1Ø.56	13.52	16.64	2Ø.16	23.28	27.48	33.28

Tableau 5 : graphe à 1ØØ sommets

Ces cinq tableaux résument les performances des méthodes étudiées sur 875 (= 5 * 7 * 25) cas différents.

De ces tests numériques les conclusions suivantes peuvent être tirées :

1) Les méthodes de permutation (VPI,LFI,SLI) améliorent les méthodes séquentielles de façon considérable (±8%).
2) Parmis les méthodes séquentielles, la méthode par valeur propre est la plus mauvaise; LF et SL sont assez équivalentes (SL semble meilleure pour les petites dimensions et LF pour les grandes dimensions), mais il s'avère que LF est préférable à SL pour des graphes à taux de remplissage élevé (≥ 4Ø %).
3) Parmi les méthodes de permutation LFI est légèrement supérieure à SLI; VPI étant la méthode de permutation la plus faible (mais toujours préférable aux méthodes séquentielles).
4) Remarquons la très bonne performance de la méthode de saturation (SAT) autant pour les grandes que pour les petites dimensions et pour des taux de remplissage bas ou élevés. Elle est d'ailleurs la seule méthode pouvant réellement concurrencer les méthodes de permutation.
5) La méthode par similarité de graphe se situe entre les méthodes séquentielles et de permutation; elle fournit de très bons résultats dans le cas des matrices de graphe à taux de remplissage élevé.

Les résultats de simulation ne reprennent pas la méthode par nombre d'éléments (NELE) dont l'intérêt est plutôt de nature pratique vu le problème de calcul d'horaire de cours sous-jacent au problème de coloration de graphe. Le comportement de la méthode NELE est en général assez analogue à celui de LF.

Remarque : Pour résoudre le problème d'horaire de cours, le choix du modèle est très important. A côté des avantages cités dans la remarque du paragraphe I.1.2., on doit mettre l'importance sur le fait que le modèle de graphe est aussi réaliste et implémentable pour des problèmes de grande dimension et comme nous allons le voir dans le deuxième chapitre, il s'adapte très bien aux modifications dues aux contraintes dites directes.

D'autres modèles sont envisageables pour le calcul d'horaire : programmation linéaire [20], [21], graphes [22], [23], hypergraphes [10], [20], matroïdes [20], flots [10], [20],... Ils sont pourtant moins performants (le modèle d'hypergraphe étant encore le meilleur) et il est très difficile d'y apporter les changements nécessaires afin d'envisager les contraintes directes.

C H A P I T R E I I

Chapitre II : L'horaire de cours complet comme modèle de graphe avec contraintes de temps

Nous adaptons le modèle de graphe aux contraintes pédagogiques et nous décrivons de nouvelles méthodes de coloration ainsi que les fichiers relatifs aux contraintes indirectes et directes.

II.1. Le modèle de graphe et les contraintes directes

Comme on vient de le voir au premier chapitre, le calcul d'horaire de cours revient à donner une heure à chaque cours tout en respectant les contraintes dites indirectes (aucune classe, aucun professeur et aucun local n'est pris deux fois à la même heure). Le modèle de graphe correspondant (I.1.2.) résout parfaitement ce problème et le calcul d'horaire se ramène à la coloration d'un graphe simple.
Ce modèle ne satisfait pourtant pas toutes les exigences réelles : il faut lui ajouter les contraintes pédagogiques dites contraintes directes.

II.1.1. Les contraintes directes

Elles représentent les exigences qui se posent souvent dans le calcul d'horaire d'un établissement scolaire et qui doivent donc être envisagées au niveau du modèle de graphe. Nous distinguons 9 types de contraintes :

1) Congés de classe : Pour une classe donnée, libérer des heures de la semaine en raison de travaux pratiques ou de congés complets.

2) Indisponibilités de professeurs : Un certain professeur désire être libre pendant l'une ou l'autre journée ou heure en raison de cours donnés à l'extérieur, travaux de recherche ou autres.

3) Indisponibilités de locaux : Certains locaux doivent rester libres à des heures déterminées pour le nettoyage ou pour d'autres raisons.

4) Professeurs ne désirant pas donner deux cours par jour :

Un professeur désire un horaire dans lequel il a ses cours bien répartis sur la semaine : il ne désire pas donner deux cours par jour.

5) Professeurs ne désirant pas donner deux cours consécutivement :

Un professeur tient à ce que ses jours ne soient pas trop chargés : il ne désire pas donner deux cours consécutivement.

6) Cours simultanés : Certains cours doivent se donner à la même heure. Ceci est généralement le cas pour les cours à option (ce qui permettra alors d'avoir pour une certaine classe plusieurs cours différents à la même heure).

7) Cours consécutifs : Des cours doivent être donnés de façon consécutive pendant la journée. Cette contrainte pédagogique est parfois formulée par des professeurs donnant le même type de cours dans différentes classes de manière à ce que ces cours se suivent.

8) Horaires calculés : Des horaires déjà calculés par programme peuvent représenter des indisponibilités : il y a parfois des professeurs ou locaux communs qui doivent par conséquent être libérés aux heures auxquelles ils sont déjà pris.

9) Cours fixés d'avance : Lorsqu'un professeur vient de l'extérieur, ses cours doivent parfois être fixés d'avance. Ceci peut être le cas pour beaucoup d'autres raisons (professeurs désirant garder le même horaire que l'année précédente,...).

Les priorités parmi ces contraintes se présentent comme suit :

(1) contraintes de type 9
(2) contraintes de type 1,2,3 et 8
(3) contraintes de type 6
(4) contraintes de type 7
(5) contraintes de type 4 et 5

Toutes ces contraintes doivent être envisagées dans le modèle de graphe qu'on doit par conséquent améliorer en lui ajoutant le concept de fonction de temps (ou matrice de temps). Ces modifications font l'objet du paragraphe suivant.

II.1.2. Graphe avec fonction de temps

Reprenons le modèle I.1.2. :

Soient $E = \mathcal{C} \cup \mathcal{P} \cup \mathcal{L}$

$X \subset \{x \text{ tq } x \text{ est sous-ensemble de } E\}$ où X est l'ensemble des sommets du graphe

$U = \{(x,y) \text{ tq } x,y \in X,\ x \neq y,\ x \cap y \neq \phi\}$ où U est l'ensemble des arcs du graphe

H = ensemble de couleurs possibles (heures en semaine);

ils définissent le graphe G(X,U).

Nous ajoutons à ce graphe une fonction de temps

$$L : X \times H \longrightarrow \{\emptyset, 1\} \text{ telle que}$$

$$L(x,h) = \begin{cases} 1 \text{ si le sommet x peut prendre la couleur h} \\ \emptyset \text{ sinon} \end{cases}$$

Cette fonction de temps contiendra au début de la coloration une partie des contraintes directes (les indisponibilités) et sera modifiée au fur et à mesure que l'algorithme de coloration avance.

La fonction de temps peut être mise sous forme d'une matrice de temps à n * ihse éléments où n = nombre de sommets

ihse = nombre de couleurs possibles (heures)

Donc, $L(i,j) = \begin{cases} 1 \text{ si le i-ième cours peut avoir lieu à l'heure j} \\ \emptyset \text{ sinon} \end{cases}$

Remarquons que les contraintes directes de type 4, 5, 6 et 7 compliquent considérablement le processus de coloration d'un sommet. Nous y reviendrons en détail au paragraphe II.4.

Exemple : Soit $E = \{c_1, c_2, p_1, p_2, p_3, l_1, l_2\}$

$X = \{x_1 = \{c_1, p_1, l_1\}, x_2 = \{c_2, p_3, l_2\}, x_3 = \{c_1, c_2, p_2\}, x_4 = \{p_1, l_2\}\}$

Voyons comment les contraintes directes s'expriment en terme de matrice de temps : Soit ihse = 6 et ihjo = 3 (le nombre d'heures par jour)

- Le professeur p_1 est indisponible à l'heure 2 :
  $L(1,2) = \emptyset$; $L(4,2) = \emptyset$
- Le local l_2 doit rester libre à l'heure 3 :
  $L(2,3) = \emptyset$; $L(4,3) = \emptyset$
- Le cours x_3 est fixé à l'heure 1 :
  $f(3) = 1$ placer le cours x_3 à l'heure 1
  $L(1,1) = \emptyset$; $L(2,1) = \emptyset$ interdire l'heure 1 aux adjacents de x_3
- Le professeur p_1 ne désire pas donner deux cours par jour : cette contrainte est exploitée lors du processus de coloration même (comme les contraintes 5, 6 et 7). Supposons que le processus de coloration place le cours x_1 à l'heure 4, alors les éliminations suivantes sont nécessaires :
  $L(3,4) = \emptyset$; $L(4,4) = \emptyset$ interdire l'heure 4 aux adjacents de x_1
  $L(4,5) = \emptyset$; $L(4,6) = \emptyset$ interdire les heures de la deuxième journée pour tous les cours contenant p_1

On constate que la matrice de temps se creuse au fur et à mesure que l'algorithme de coloration avance (contraintes directes et indirectes) et que le processus d'élimination se complique fortement par rapport au chapitre I (où il ne fallait faire que les éliminations des adjacents).

II.2. Quelques algorithmes de coloration supplémentaires

Au chapitre I, on a décrit une série de méthodes de coloration (paragraphes I.2, I.3, I.4 et I.5) basées sur des informations du graphe à colorer. Ce paragraphe décrira des algorithmes exploitant des informations contenues dans la matrice de temps (c-à-d dans les contraintes directes). Nous définissons également les méthodes de perturbation qui sont d'un intérêt pratique très important. Toutes les méthodes vues au premier chapitre s'adaptent facilement à l'utilisation de la matrice de temps; sauf les méthodes de permutation pour lesquelles les changements sont décrits en II.2.4.

II.2.1. Méthodes de coloration séquentielle

Le principe des algorithmes de coloration séquentielle a été étudié en détail au paragraphe I.2. Les méthodes proposées diffèrent selon leur critère d'ordonnancement (LF,SL,VP,NELE). Ces critères se basaient surtout sur des propriétés du graphe et de sa matrice d'adjacence (degré, valeur propre, nombre d'éléments). Nous proposons ici un ordonnancement selon la notion de possibilité de coloration.

Définition : Soit le problème défini en II.1.2., alors la possibilité de coloration du sommet i, notée Poc(i), est définie par

$$\boxed{Poc(i) = \sum_{j=1}^{ihse} L(i,j)}$$

Elle représente donc le nombre de couleurs possibles (parmi les ihse couleurs totales) que peut prendre le sommet i.
Nous allons ordonner les sommets suivant cette notion en favorisant les sommets à possibilité de coloration petite (grande difficulté de coloration). Il en résulte l'algorithme que voici :

Algorithme POCO :

1) Ordonner les sommets tels que $Pos(1) \leqslant Pos(2) \leqslant \ldots \leqslant Pos(n)$.
2) i = 1
3) si i > n, arrêter
   sinon, colorer le sommet i avec la première couleur possible.
4) Faire les éliminations dues aux contraintes directes et indirectes.
   i = i + 1 ; aller en 3).

Les sommets (cours) sont donc ordonnés une fois pour toutes avant la coloration en exploitant surtout les contraintes d'indisponibilité (type 1, 2, 3 et 8). Un critère plus performant sera décrit en II.2.2. : il tient compte des autres contraintes (type 1,2,...,9).

Remarquons une fois de plus que les contraintes directes ajoutées au modèle de graphe compliquent le processus de coloration. Donc, dans tous les algorithmes vus dans ce chapitre, la partie "colorer un sommet et faire les éliminations nécessaires" sera très complexe. On la décrit en II.4. sans insister pour le moment.

II.2.2. Méthodes de saturation

Comme on vient de le voir en I.3., les méthodes de saturation réordonnent les sommets non colorés en utilisant les informations relatives aux colorations déjà faites. Cette caractéristique permet de tenir compte des contraintes intervenant au fur et à mesure que le processus de coloration avance. Ceci est essentiellement le cas pour les contraintes 4 et 5. On réordonne les sommets après la coloration d'un sommet quelconque selon la notion de possibilité de coloration vue précédemment. L'avantage de cette technique par rapport à II.2.1. provient du fait qu'on réordonne les cours en fonction de la matrice de temps qui reflètera, lorsque l'algorithme avance, non pas seulement les contraintes d'indisponibilité,

mais aussi les contraintes "professeurs ne désirant pas donner deux cours par jour",... et permet dès lors de mieux exploiter les contraintes du problème. On dire aussi qu'on ordonne selon la possibilité de coloration effective des sommets.

Algorithme EFF :

1) Ordonner les sommets tels que $Pos(1) \leq Pos(2) \leq \ldots \leq Pos(n)$.
2) Colorer le sommet x_1 avec la première couleur possible et faire les éliminations nécessaires.
3) S'il ne reste plus de sommets à colorer, arrêter.
   Sinon, le sommet non coloré, dont la possibilité de coloration est la plus petite parmi les sommets non colorés, est pris.
4) Colorer ce sommet avec la première couleur possible et faire les éliminations nécessaires.
5) Aller en 3).

II.2.3. Méthodes de similarité

La coloration des sommets par paires a été decrite en I.4. Nous ferons ici un raisonnement analogue, mais au lieu d'exploiter les propriétés du graphe (similarité de graphe), nous allons plutôt nous baser sur les informations relatives à la matrice de temps (similarité de temps).

Définition : Etant donné le graphe G(X,U) à n sommets et M sa matrice d'adjacence ainsi que la matrice de temps L à ihse colonnes (couleurs possibles), définissons la similarité de temps T comme suit :

$$T = T(i,j)_{1 \leq i,j \leq n} \quad \text{où} \quad T(i,j) = \begin{cases} \emptyset \text{ si } i=j \text{ ou } M(i,j) = 1 \\ \sum_{k=1}^{ihse} L(i,k) \cdot L(j,k) \text{ si } M(i,j) = \emptyset \text{ et } i \neq j \end{cases}$$

La similarité de temps de deux sommets est le nombre de couleurs que peuvent prendre les deux sommets à la fois.

L'algorithme décrit ci-après est analogue à celui étudié en I.4. :

Algorithme SIMT :

1) Construire T; soit max la similarité de temps maximale.
2) Soit [i,j] une paire non traitée telle que T(i,j) = max.
3) ...
4) ... } voir SIMG (I.4.2)
5) ...

Les sommets non colorés par similarité sont traités en dernier lieu en les ordonnant selon un des critères séquentiels (LF,SL,VP,NELE,POCO).

II.2.4. Méthodes de permutation

Nous disposons jusqu'à présent de cinq méthodes de coloration séquentielle (VP,LF,SL,NELE,POCO) dont les quatre premières sont reprises du premier chapitre et s'adaptent sans trop de changements au problème de coloration avec matrice de temps. Il en est de même pour les méthodes de saturation (SAT) et de similarité (SIMG). Par contre, les méthodes de permutation doivent être transformées afin de s'adapter à l'emploi de la matrice de temps. Elles sont décrites en I.5. et se présenteront comme suit pour le cas de graphes avec matrice de temps :

{ les changements proviennent du fait que la recherche de permutations bichromatiques doit tenir compte des contraintes de temps.

Soit donné un graphe G et sa matrice de temps L. Supposons G possédant une k-coloration 1,2,...,k. Soit G_i l'ensemble des sommets de G colorés i. Soit L° la matrice de temps initiale avant qu'aucune coloration ne soit faite. Donc, L° vaut L avant le début du processus de coloration (L se transforme durant la coloration).

Pour $i \neq j$, le sous-graphe bichromatique i,j est le sous-graphe engendré par $G_i \cup G_j$, noté $\langle G_i \cup G_j \rangle$. Une composante connexe de $\langle G_i \cup G_j \rangle$ est dite i,j composante. Si les couleurs i et j sont permutées sur une i,j composante du graphe k-coloré G, alors une autre k-coloration de G est obtenue. Cette procédure est appelée i$\leftrightarrow$j permutation sur le graphe k-coloré G avec contraintes de temps.

Soient $x_1, x_2, \ldots, x_k$ les sommets de cette composante connexe de $\langle G_i \cup G_j \rangle$. Alors, une i$\leftrightarrow$j permutation n'est possible que si

$$\boxed{\forall\, x_l \in \{x_1, \ldots, x_k\},\ L^o(l,i) = L^o(l,j) = 1}$$

Autrement dit, les contraintes de temps (avant la coloration) permettent à tous les sommets de la composante connexe en question de prendre aussi bien la couleur i que la couleur j.

Une permutation bichromatique sur un graphe k-coloré G avec contraintes de temps est une i$\leftrightarrow$j permutation pour $i \neq j$.

La recherche de permutations bichromatiques dans le processus de coloration, tout en vérifiant les contraintes de temps, fournit l'algorithme suivant :

Algorithme de permutation-temps :

1) Ordonnancement des sommets par une des méthodes séquentielles classiques (VP,LF,SL,NELE,POCO).
2) $j = 1$
3) si $L(x_1,j) = \emptyset$, $j = j + 1$, aller en 3)
   sinon $f(x_1) = j$; $L(x_l,j) = \emptyset$ pour les adjacents x_l de x_1.
   $i = 2$
4) - Si $i > n$, arrêter
   - Sinon, soit $\langle x_1, x_2, \ldots, x_{i-1} \rangle$ coloré avec $1,2,\ldots,j$
     - si on parvient à colorer x_i avec $l \in \{1,\ldots,j\}$, $f(x_i) = l$ et $L(x_k,l) = \emptyset$ pour les adjacents x_k de i.
     - sinon, soit $K \subset \{1,\ldots,j\}$ l'ensemble des couleurs tel que
       $\alpha \in K \Longleftrightarrow$ 1) $L^o(x_i,\alpha) = 1$
       2) exactement un sommet de $x_1,\ldots,x_{i-1}$ adjacent à x_i a la couleur α.
     - si $\alpha, \beta \in K$, une α,β composante de $\langle x_1,\ldots,x_{i-1} \rangle$ n'a qu'un sommet adjacent à x_i en $\langle x_1,\ldots,x_i \rangle$ et que tout sommet

x_1 de cette composante vérifie $L(x_1,\alpha) = L(x_1,\beta) = 1$, faire une $\alpha \leftrightarrow \beta$ permutation sur une telle α,β composante de $\langle x_1,\ldots,x_{i-1}\rangle$. Colorer les sommets $x_1,\ldots,x_{i-1}$ de même que dans cette nouvelle coloration et x_i avec la couleur possible, soit α ou β, et une j-coloration de $\langle x_1,\ldots,x_i\rangle$ est obtenue; interdire $f(x_i)$ aux adjacents de x_i.

- sinon, une telle permutation n'est pas possible; colorer $x_1,\ldots,x_{i-1}$ de même qu'en $\langle x_1,\ldots,x_{i-1}\rangle$;
  a) j = j + 1
  b) si $L(x_i,j) = 1$, $f(x_i) = j$; interdire $f(x_i)$ aux adjacents de x_i
     sinon, aller en a)

5) i = i + 1 ; aller en 4).

On s'est contenté de n'étudier que le comportement théorique des méthodes de permutation (vu leur complexité et les difficultés d'implémentation des contraintes de type 4, 5, 6 et 7). Elles ne figurent donc pas parmi les méthodes de calcul d'horaire proposées dans le troisième chapitre, mais font partie des résultats numériques présentés ci-après.

II.2.5. Résultats numériques

Nous comparons les méthodes séquentielles (VP,LF,SL,POCO), de permutation (VPI,LFI,SLI,POCOI), de saturation (SAT,EFF) et de similarité (SIMG, SIMT) en simulant la matrice d'adjacence du graphe et la matrice de temps. Le paragraphe I.5. (simulation de la matrice d'adjacence seulement) en est un cas particulier correspondant à un taux de remplissage de 1ØØ % de la

matrice de temps. On s'intéresse donc ici surtout au comportement des méthodes vues selon le taux de remplissage de la matrice de temps au départ de la coloration.
Les paramètres des jeux de test varient entre des dimensions de 2Ø , 6Ø et 1ØØ, entre des taux de remplissage de la matrice d'adjacence de 2Ø % à 8Ø % et entre des taux de remplissage de la matrice de temps de 6Ø % , 75 % et 9Ø %. De même qu'en I.5., la borne supérieure moyenne du nombre chromatique est calculée sur 25 cas différents (pour une dimension et des taux de remplissage donnés) ce qui permet de juger les différentes méthodes sur 9ØØ(= 3 * 4 * 3 * 25) cas différents.

Les tableaux des calculs numériques faits sont repris au pages 12 à 14 de ce chapitre et mettent en évidence les constatations suivantes :

1) Les méthodes basées sur la matrice de temps (POCO,EFF,SIMT) ne sont éfficaces que pour des taux de remplissage de L assez bas. Il s'avère que SIMT est assez faible surtout pour des graphes à taux de remplissage élevé (au-dessus de 4Ø %)
2) Les méthodes de permutation (VPI,LFI,SLI,POCOI) sont très performantes; POCOI (pour des matrices de temps peu remplies) et SLI fournissent des résultats particulièrement bons. Une amélioration de ±8 % par rapport aux méthodes séquentielles souligne l'éfficacité des méthodes de permutation.
3) Parmi les méthodes de saturation (SAT,EFF), SAT est la plus performante, même si EFF donne des résultats remarquables pour une matrice de temps peu remplie.
4) La méthode de similarité de graphe est préférable à la méthode de similarité de temps sans qu'elle puisse réellement concurrencer avec les méthodes de saturation ou de permutation. Les résultats de SIMT ne sont satisfaisants que pour des graphes à taux de remplissage de M et L faible.

De même qu'en I.7., la méthode par nombre d'éléments (NELE) n'a pas été reprise dans cette simulation vu son intérêt plutôt pratique.

Taux de remplissage de M

Taux de remplissage de L

	2Ø %			4Ø %			6Ø %			8Ø %		
	6Ø %	75 %	9Ø %	6Ø %	75 %	9Ø %	6Ø %	75 %	9Ø %	6Ø %	75 %	9Ø %
VP	6.44	5.16	4.16	8.4Ø	6.76	5.96	9.88	8.88	7.68	12.64	11.24	1Ø.44
VPI	6.48	4,96	4.Ø8	8.Ø8	6.52	5.6Ø	9.6Ø	8.32	7.48	12.Ø4	11.Ø4	1Ø.Ø8
LF	6.48	5.Ø4	4.24	8.6Ø	6.32	5.96	1Ø.2Ø	8.76	7.6Ø	12.6Ø	11.ØØ	1Ø.44
LFI	6.44	4.92	3.96	8.48	6.Ø4	5.52	9.68	8.52	7.4Ø	12.ØØ	1Ø.8Ø	1Ø.ØØ
SL	6.52	4.92	4.2Ø	8.ØØ	6.4Ø	5.84	1Ø.28	8.8Ø	7.6Ø	13.16	11.4Ø	1Ø.52
SLI	6.52	4.88	4.16	7.88	6.44	5.64	1Ø.Ø8	8.28	7.12	12.68	1Ø.92	9.76
PCCO	6.52	5.ØØ	4.4Ø	7.88	7.Ø4	6.12	1Ø.Ø4	8.72	8.36	12.72	11.8Ø	1Ø.92
PCCOI	6.12	4.6Ø	3.92	7.64	6.4Ø	5.52	9.Ø4	7.92	7.56	11.56	1Ø.76	1Ø.2Ø
SAT	6.48	5.2Ø	4.16	8.16	6.48	5.68	1Ø.4Ø	8.48	7.44	12.36	1Ø.92	1Ø.24
EFF	6.36	5.ØØ	4.Ø8	7.88	6.72	5.68	9.28	8.48	7.64	12.Ø8	11.2Ø	1Ø.2Ø
SIMG	8.76	5.8Ø	4.4Ø	1Ø.48	6.72	6.ØØ	11.Ø4	8.56	7.6Ø	13.Ø8	1Ø.72	9.96
SIMT	8.24	5.76	4.68	1Ø.68	7.8Ø	6.64	11.64	9.48	8.32	13.Ø4	11.64	10.84

Méthode utilisée

Tableau 1 : graphe à 2Ø sommets

	2Ø %			4Ø %			6Ø %			8Ø %		
	6Ø %	75 %	9Ø %	6Ø %	75 %	9Ø %	6Ø %	75 %	9Ø %	6Ø %	75 %	9Ø %
VP	10.52	8.96	7.48	15.4Ø	13.48	12.28	2Ø.52	18.6Ø	16.92	28.48	25.2Ø	24.24
VPI	9.96	8.32	7.Ø4	13.92	12.56	11.2Ø	19.Ø4	16.68	15.64	25.88	23.4Ø	22.4Ø
LF	1Ø.96	9.Ø4	7.76	15.12	13.24	12.Ø4	2Ø.Ø8	18.32	16.8Ø	27.4Ø	25.24	23.56
LFI	1Ø.12	8.56	6.88	14.4Ø	12.36	11.12	18.64	17.Ø8	15.64	25.64	23.56	22.12
SL	11.2Ø	9.Ø8	7.84	15.68	13.64	12.12	2Ø.ØØ	18.28	17.6Ø	28.2Ø	25.52	24.48
SLI	1Ø.24	8.28	6.88	13.92	12.24	11.Ø8	18.68	16.6Ø	15.44	26.Ø4	23.28	22.Ø8
POCO	1Ø.52	9.36	8.Ø4	14.88	13.96	13.4Ø	2Ø.72	18.96	17.88	28.52	26.Ø8	25.4Ø
POCOI	1Ø.12	8.4Ø	7.4Ø	14.16	12.84	11.68	19.12	17.24	16.52	25.88	24.ØØ	22.88
SAT	1Ø.24	8.6Ø	7.32	14.72	12.84	11.32	19.24	17.48	16.Ø4	26.92	24.8Ø	23.12
EFF	1Ø.6Ø	9.12	7.92	14.8Ø	13.2Ø	12.16	2Ø.ØØ	18.2Ø	17.16	27.16	26.16	24.Ø4
SIMG	13.48	1Ø.52	8.32	16.88	13.76	12.6Ø	21.24	18.36	16.76	27.6Ø	24.16	22.68
SIMT	12.76	1Ø.88	8.92	18.76	15.52	13.88	24.48	2Ø.6Ø	18.88	29.8Ø	27.2Ø	26.ØØ

Tableau 2 : graphe à 6Ø sommets

	2Ø %			4Ø %			6Ø %			8Ø %		
	6Ø %	75 %	9Ø %	6Ø %	75 %	9Ø %	6Ø %	75 %	9Ø %	6Ø %	75 %	9Ø %
VP	13.5Ø	11.92	1Ø.67	21.33	18.42	17.25	29.67	27.25	24.92	41.17	38.17	36.25
VPI	12.58	11.17	9.92	2Ø.ØØ	17.42	15.83	27.33	25.67	23.Ø8	38.25	35.58	33.34
LF	13.58	11.75	1Ø.67	21.33	18.83	17.17	29.42	27.ØØ	24.83	39.92	38.25	35.25
LFI	12.67	11.33	9.5Ø	19.67	17.17	15.75	27.33	24.83	23.25	38.67	35.Ø8	33.Ø8
SL	14.ØØ	12.Ø8	1Ø.92	21.Ø8	19.67	17.Ø8	29.5Ø	27.58	25.17	42.ØØ	37.92	37.Ø8
SLI	12.5Ø	11.25	9.42	19.17	18.Ø8	15.5Ø	27.92	25.ØØ	22.83	38.42	35.ØØ	32.92
POCO	14.Ø8	12.17	11.45	2Ø.75	19.67	18.33	29.83	27.5Ø	25.92	41.67	39.5Ø	38.17
POCOI	12.92	11.33	1Ø.25	19.83	18.ØØ	16.83	27.58	25.33	23.92	38.25	35.83	34.33
SAT	13.75	11.Ø8	9.58	2Ø.42	18.Ø8	16.ØØ	28.75	26.5Ø	23.67	4Ø.Ø8	37.Ø8	33.75
EFF	13.5Ø	11.75	1Ø.5Ø	2Ø.33	18.83	17.33	28.67	27.5Ø	25.33	4Ø.58	38.33	36.67
SIMG	17.33	14.33	11.42	25.33	2Ø.17	18.Ø8	3Ø.17	26.ØØ	25.Ø8	38.75	35.17	33.42
SIMT	19.42	14.17	12.ØØ	25.58	21.92	19.42	34.33	29.75	27.42	44.42	4Ø.42	38.Ø8

Tableau 3 : graphe à 1ØØ sommets

Remarques :

1) Les méthodes de coloration vues colorent le graphe avec le moins de couleurs possibles ce qui signifie en pratique que les horaires de cours vont être calculés avec le minimum d'heures possibles. Ce nombre d'heures doit être inférieur ou égal au nombre d'heures par semaine (ihse) qui représente donc une borne supérieure du nombre chromatique du graphe associé à l'horaire à calculer. Si cela n'est pas le cas, certains cours ne peuvent pas être placés.
2) Le processus de coloration d'un sommet est très compliqué comme on vient de le constater en II.1.2. Il comprend les parties suivantes :
   - coloration proprement dite
   - éliminations dues aux adjacents et contraintes directes (type 4 et 5)
   - tests supplémentaires relatifs aux cours simultanés, aux cours consécutifs,...

   Une étude détaillée sera présentée en II.4.
3) La façon exacte dont interviennent les contraintes directes dans le calcul d'horaire sera décrite en II.3. et II.4. Une description des fichiers correspondants ainsi que les conséquences au niveau du processus de coloration complèteront l'étude des contraintes directes.
4) Les remarques concernant la complexité des méthodes décrites au premier chapitre restent évidemment valables pour les mêmes types de méthodes vues en II.2. (séquentielles, saturation, similarité, permutation).

Nous décrivons dans le paragraphe suivant une dernière classe de méthodes qui sont d'une utilité pratique très importante pour le calcul d'horaire de cours : les méthodes de perturbation.

II.2.6. Méthodes de perturbation

Etant donné qu'un horaire a été calculé par une des méthodes proposées jusqu'à présent, il s'avère qu'en pratique cet horaire représente toujours l'un ou l'autre défaut qu'on sait facilement changer en modifiant légèrement les contraintes directes. Il est dans ce cas très intéressant de disposer de méthodes de calcul d'un nouvel horaire, respectant ces nouvelles contraintes, sans qu'il présente trop de changements par rapport à l'ancien horaire.
Les méthodes dites de perturbation calculent un nouvel horaire basé sur l'ancien horaire déjà connu et respectant les nouvelles contraintes du problème (qui ne doivent pas contenir trop de changements par rapport aux anciennes contraintes pour que la méthode soit réellement éfficace).
Deux types de méthodes de perturbation sont décrits : perturbation globale et perturbation locale.

II.2.6.1. Perturbation globale

Cette méthode essaye de donner à chaque cours du nouvel horaire l'heure qu'il possédait dans l'ancien horaire en ordonnant les cours selon un des critères séquentiels (LF,SL,VP,NELE,POCO) sans accorder une priorité aux cours de l'ancien horaire respectant les contraintes du nouvel horaire.

Algorithme PGLO :

1) Ordonner les cours selon LF, SL, VP, NELE ou POCO.
2) i = 1
3) Si i > n, arrêter
   Sinon, - si f(i) $\neq \emptyset$ (c-à-d le cours i est déjà placé et ceci à cause d'une des contraintes directes), aller en 4).
   - sinon, chercher le cours i dans l'ancien horaire :
     + s'il y existe, essayer de lui donner la même heure que dans l'ancien horaire (tout en respectant les nouvelles

contraintes).

* si cela réussit, faire les éliminations nécessaires ainsi que la coloration.
* sinon, lui donner la première couleur possible et faire les éliminations nécessaires.

+ sinon, lui donner la première couleur possible et faire les éliminations nécessaires.

4) i = i + 1 ; aller en 3).

II.2.6.2. Perturbation minimale

Elle procède de manière analogue à PGLO, mais en donnant cette fois-ci la priorité aux cours de l'ancien horaire satisfaisant les contraintes du nouvel horaire.

Algorithme PMIN :

1) Ordonner les cours selon LF, SL, VP, NELE ou POCO.
2) i = 1
3) Si i < n, colorer les cours restants par une méthode séquentielle.
   Sinon, - si f(i) $\neq \emptyset$, aller en 4).
   - sinon, chercher le cours i dans l'ancien horaire :
     + s'il existe, essayer de lui donner la même heure que dans l'ancien horaire (en respectant les nouvelles contraintes).
       * si cela réussit, faire les éliminations nécessaires ainsi que la coloration.
       * sinon, aller en 4).
     + sinon, aller en 4).
4) i = i + 1 ; aller en 3).

Remarquons qu'on a pris la convention qu'un cours non placé ("non coloré") possède une heure ("couleur") nulle (c-à-d f(i) = $\emptyset$).
La différence entre PGLO et PMIN se reflète bien sur l'exemple de la page suivante :

II.2.6.3. Exemple

Soit l'ancien horaire : cours 1 coloré 1 où les cours 3 et 4 sont
2 1 adjacents (les autres
3 2 contraintes ne nous inté-
4 3 ressent pas)
5 2

et le nouvel horaire à calculer qui reprend les mêms données avec la contrainte supplémentaire $L(3,2)=L(4,2)=\emptyset$ (les cours 3 et 4 ne peuvent pas avoir lieu à la deuxième heure).

Analysons les deux méthodes de perturbation sur cet exemple :

PGLO :

ordre de coloration ↓

cours	coloré	
1	1	} mêmes couleurs que dans l'ancien horaire
2	1	}
3	3	} première couleur possible pour le cours 3
		} et faire $L(4,3) = \emptyset$ (car 3 et 4 adjacents)
4	4	} première couleur possible pour le cours 4
5	2	} même couleur que dans l'ancien horaire

On obtient deux changements par rapport à l'ancien horaire (cours 3 et 4).

PMIN :

ordre de coloration ↓

cours	coloré	
1	1	} mêmes couleurs que dans l'ancien horaire
2	1	}
4	3	}
5	2	}
3	4	} première couleur possible pour le cours 3

On trouve donc un seul changement par rapport à l'ancien horaire (cours 3).

Donc, le fait que PMIN favorise les cours de l'ancien horaire, dont la coloration satisfait encore les nouvelles contraintes, diminue le nombre de changements entre l'ancienne et la nouvelle coloration. La méthode PMIN se prête dès lors mieux aux petits changements de contraintes (directes ou indirectes) tandis que PGLO s'avère assez efficace lors de grands changements.

Comme les méthodes de perturbation se basent sur l'ordonnancement par un des critères séquentiels connus, leur complexité sera de l'ordre de $\Theta(n)$.

II.3. Les contraintes : structure des fichiers et implémentation

Nous avons décrit le problème d'horaire de cours (I.1 et II.1) en le traduisant sous forme d'un graphe avec fonction de temps. Nous allons analyser dans la suite de façon plus précise les données nécessaires à une formulation et implémentation exacte des algorithmes vus jusqu'à présent en décrivant les fichiers correspondants.

Les informations à traiter se répartissent selon les trois types suivants :

a) Le nombre d'heures par semaine (pendant lesquelles les cours peuvent avoir lieu), le nombre d'heures par jour, le début de chaque heure de la journée, ainsi que les classes, professeurs et locaux de l'horaire en question.
b) Les cours à donner (contraintes indirectes) qui se présentent sous la forme : nom du cours, numéro logique (intitulé correspondant dans le programme des cours), les classes, professeurs et locaux qui participent au cours.
c) Les contraintes directes de type :
   1) indisponibilités de classes, professeurs et locaux
   2) professeurs ne désirant pas donner deux cours par jour
   3) professeurs ne désirant pas donner deux cours consécutivement
   4) cours simultanés
   5) cours consécutifs
   6) horaires déjà calculés
   7) cours fixés

La manière de représenter ces informations sur fichier et la façon dont elles interviennent dans le processus de coloration seront examinées dans ce paragraphe.
Les informations sont passées aux algorithmes de calcul exactement dans l'ordre dont nous les décrivons ci-après.
L'organisation de tous les fichiers est séquentielle.

II.3.1. Les classes, professeurs et locaux

Le fichier correspondant aura la forme suivante :

Fichier de type "classe-prof-local"

1) Le nombre d'heures par semaine (ihse)
2) Le nombre d'heures par jour (ihjo)
3) Le début de chaque heure de la journée (debut(i), $1 \leq i \leq$ ihjo)
4) Le nombre total de classes (nc)
5) Le nombre total de professeurs (np)
6) Le nombre total de locaux (nl)
7) Les classes (val(i), $1 \leq i \leq$ nc)
8) Les professeurs (val(i+nc), $1 \leq i \leq$ np)
9) Les locaux (val(i+nc+np), $1 \leq i \leq$ nl)

Les classes, professeurs et locaux sont identifiés par leur nom.
Chaque classe, professeur et local aura un numéro (attribué lors de la lecture du fichier) :

La classe val(i) est la classe numéro i, $1 \leq i \leq$ nc.
Le professeur val(j+nc) est le professeur numéro j, $1 \leq j \leq$ np.
Le local val(k+nc+np) est le local numéro k, $1 \leq k \leq$ nl.

Ce numéro jouera un rôle important dans la façon d'implémenter les cours à donner (II.3.2).

II.3.2. Les cours à donner

Remarquons encore une fois que le numéro logique d'un cours permet de regrouper les cours faisant partie du même intitulé; cette information sera exploitée par la contrainte "cours simultanés" (II.3.8) et par l'impression des résultats (impressions du type "programme des cours", voir aussi III.2.).

Fichier de type "cours-à-donner"

Il contient les cours à donner dont chacun a la forme que voici :

1) Nom du cours
2) Numéro logique
3) Nombre de classes du cours
4) Les classes
5) Nombre de professeurs du cours
6) Les professeurs
7) Nombre de locaux du cours
8) Les locaux

On se limite aux cours possèdant au plus dix éléments (classes, professeurs et locaux) distincts.
Les cours se trouvent dans un ordre quelconque dans le fichier qui sera lu de façon séquentielle (comme tous les autres fichiers d'ailleurs).
Ainsi, le i-ème cours lu est mémorise par :

- nom du cours : name(i) $1 \leq i \leq n$ où n est le nombre total de cours
- numéro logique : nulo(i) $1 \leq i \leq n$
- ses éléments : a(i,k) = j $1 \leq k \leq 10$, $1 \leq i \leq n$

$\Longleftrightarrow$ 1) si $1 \leq j \leq nc$, le cours contient la classe numéro j

2) si $nc+1 \leq j \leq nc+np$, le cours contient le professeur numéro j-nc

3) si $nc+np+1 \leq j \leq nc+np+nl$, le cours contient le local numéro j-nc-np

a(i,k) = Ø $1 \leq k \leq 10$, $1 \leq i \leq n$

$\Longrightarrow$ le cours contient moins de k éléments

La i-ème ligne de la matrice des cours (a) contient donc (par l'intermédiaire des indices du vecteur val) les éléments du cours numéro i (dans l'ordre : classes, professeurs, locaux).

Exemple : Soit nc = 3, np = 5 et nl = 3.

Le cours numéro 7 contenant
deux classes : "math1", "phys1" où val(1) = "math1"
val(2) = "phys1"

un professeur : "Mersch" où val(7) = "Mersch"
un local : "M1" où val(1Ø) = "M1"
ayant le nom "algèbre" et le numéro logique 13;
alors il sera mémorisé comme
name(7) = "algèbre"
nulo(7) = 13
a(7,1) = 1, a(7,2) = 2, a(7,3) = 7, a(7,4) = 1Ø
a(7,i) = Ø 5 ⩽ i ⩽ 1Ø.

Le nom de chaque cours doit être identifiant (unique parmi tous les cours du fichier). Deux cours (sommets du graphe) ont au moins un élément en commun (sont adjacents) si les deux lignes correspondantes dans la matrice a possèdent au moins un élément non nul en commun.
Lors de la lecture d'un cours du fichier, il se peut qu'un élément (classe, professeur ou local) soit inconnu (ne se trouve pas dans le vecteur val). Dans ce cas, un message d'avertissement est sorti et on continue comme si cet élément n'existait pas.
L'étude de la structure des fichiers des contraintes indirectes se termine ici; passons donc aux fichiers des contraintes directes.

II.3.3. Les congés de classes

Pour tenir compte du fait que certaines classes ont congé à des heures données, il suffit d'interdire ces heures à tous les cours contenant les classes en question.

Fichier de type "congé-de-classe"

Chaque enregistrement est du type :
1) nom de la classe
2) nombre d'heures de congé
3) les heures de congé

L'effet de la lecture d'un enregistrement du fichier est :
soit cette classe la classe numéro i c-à-d val(i) = nom de la classe,
alors chercher tous les cours j tels que a(j,k) = i pour $1 \leq k \leq 10$;
pour ces cours j, L(j,ih) = Ø où ih est une heure de congé.

La matrice de temps se creuse donc déjà avant que le processus de coloration commence.
Lorsque la classe est inconnue (pas dans le fichier de type "classe-prof-local et donc non plus dans le vecteur val), un message d'avertissement est sorti en continuant comme si l'enregistrement n'existait pas.

II.3.4. Les indisponibilités de professeurs

Idem II.3.3. (Fichier de type "prof-indisponible").

II.3.5. Les indisponibilités de locaux

Idem II.3.3. (Fichier de type "local-indisponible").

II.3.6. Professeurs ne désirant pas donner deux cours par jour

Il suffit de connaître les noms des professeurs qui exigent ne pas donner deux cours par jour. Les conséquences exactes de ces contraintes dans le processus de coloration seront étudiées en II.4.

Fichier de type "prof-à-non-2-cours-par-jour"

Chaque enregistrement a la forme :
nom du professeur

Le programme mémorisera ces informations dans un vecteur à np éléments comme suit :

soit j tel que val(j) = nom du professeur,

alors indn2j(j-nc) = 1 , $1 \leq j\text{-}nc \leq np$.

Les professeurs n'exigeant pas cette contrainte ont une valeur nulle dans le vecteur indn2j.

Donc, de façon générale,

le professeur numéro j ne désire pas donner deux cours par jour

indn2j(j) = 1 et indn2j(j) = $\emptyset$ sinon.

II.3.7. Professeurs ne désirant pas donner deux cours consécutivement

Idem II.3.6. (Fichier de type "prof-à-cours-non-consécutifs", vecteur indn2c).

II.3.8. Cours simultanés

Lorsqu'on regroupe les cours à option d'une classe donnée dans un même cours, ils perdent chacun d'eux leur caractère individuel (important pour les impressions du type programme des cours; voir III.2.5.). Ceci est bien illustré par l'exemple suivant :

exemple : La classe c_1 a le choix entre le cours à option avec le professeur p_2 en l_3 et celui avec p_1 en l_4. On considère dans ce cas le cours $\{c_1, p_2, l_3, l_4\}$.

Pour éviter cette perte d'individualisme des cours, on tient compte d'un type de contraintes dites "cours simultanés". Elles obligent certains cours d'avoir lieu à la même heure, et cela même s'ils contiennent des éléments en commun (une classe par exemple) :

exemple : La classe c_1 a deux cours différents à la même heure à savoir $\{c_1, p_2, l_3\}$ et $\{c_1, p_1, l_4\}$.

Comme les cours simultanés peuvent se donner pendant plusieurs heures, il suffit de connaître leurs numéros logiques pour les regrouper aux mêmes heures :

exemple : La classe c_1 a deux cours à option dont chacun se donne pendant deux heures en semaine :

premier cours : numéro logique = 41

$$x_1 = \{c_1, p_2, l_3\},\ x_2 = \{c_1, p_2, l_3\}$$

deuxième cours : numéro logique = 42

$$x_3 = \{c_1, p_1, l_4\},\ x_4 = \{c_1, p_1, l_4\}$$

Il suffit alors de donner les numéros 41 et 42 au programme pour que celui-ci puisse regrouper x_1 et x_3 ainsi que x_2 et x_4 (sans qu'ils perdent **leur caractère individuel**). Donc, pour que le programme fasse ce regroupement, il faut lui spécifier les numéros logiques des cours simultanés en question.

Par convention, on se limite à des regroupements de quatre cours simultanés différents (il y en a au moins deux pour que la notion de simultanéïté ait un sens).

Fichier de type "cours-simultanés"

Chaque enregistrement (de longueur variable) a la forme :

{ numéro logique du premier cours
:
numéro logique du quatrième cours

où le numéro logique du troisième et quatrième cours peut ne pas exister

Ces informations sont transformées par le programme (après lecture du fichier) et placées dans un vecteur appelé "indcsi" telles que :

- indcsi(i) = $\emptyset$ si le cours i a un numéro logique ne faisant partie d'aucun groupement de cours simultanés
- indcsi(i) $\neq \emptyset$ sinon; dans ce cas indcsi(i) reprend les cours qui doivent avoir lieu en même temps que le cours i

Après la lecture du fichier, le programme effectue donc la transformation numéros logiques ⟶ groupement des cours simultanés concernés.
L'impact de cette contrainte au processus de coloration est décrit en II.4.
Lorsqu'au moins un des numéros logiques d'un groupement de cours simultanés est inconnu au programme, un message d'avertissement est sorti et on continue en négligeant la contrainte pour ce numéro logique. Il est évident qu'un numéro logique ne peut appartenir qu'à un seul groupement de cours simultanés.

II.3.9. Cours consécutifs

Il arrive que deux cours doivent se donner de façon consécutive pendant la même journée. On parle dans ce cas d'une paire de cours consécutifs. Pour que le processus de coloration puisse satisfaire cette contrainte de consécutivité, il faut lui passer le nom des deux cours consécutifs de chaque paire.

Fichier de type "cours-consécutifs"

Chaque enregistrement a la forme :
nom du premier cours
nom du deuxième cours

Après lecture du fichier, ces informations sont placées dans un vecteur "indcco" à n éléments de manière à ce que :

- indcco(i) = Ø si le cours i ne fait partie d'aucune paire de cours consécutifs
- indcco(i) = j si les cours i et j forment une paire de cours
  indcco(j) = i consécutifs

L'impact de ces informations au processus de coloration est décrit en II.4. (on ne peut placer un cours d'une paire de cours consécutifs que si l'on peut mettre l'autre cours de la paire à l'heure précédente ou suivante de la même journée).
Lorsqu'au moins un des deux cours d'une paire est inconnu, on sort un message d'avertissement et on continue le traitement en négligeant la contrainte pour la paire en question.
Il est évident qu'un cours ne peut appartenir qu'à une seule paire de cours consécutifs.

II.3.1Ø. Horaires calculés

Un horaire déjà calculé par le programme peut représenter des contraintes d'indisponibilité pour d'autres horaires à calculer après. En effet, il

arrive parfois qu'un professeur du nouvel horaire à calculer intervienne en plus dans un autre horaire (déjà calculé) et sera dès lors indisponible pendant une période qui dépend des heures de début et de fin des cours de chaque horaire en question.

Fichiers de type "horaire-calculé"

Il se présente comme suit :

1) nombre d'heures par semaine
2) nombre d'heures par jour
3) début de chaque heure de la journée
4) les cours placés dont la configuration est :
   a) heure à laquelle le cours est placé
   b) nom du cours
   c) numéro logique
   d) nombre de classes du cours
   e) les classes
   f) nombre de professeurs du cours
   g) les professeurs
   h) nombre de locaux du cours
   i) les locaux

Ces informations sont suffisantes pour gérer les indisponibilités éventuelles pour un nouvel horaire à calculer.

Exemple : Soit le nouvel horaire à calculer où
---

ihse = 12
ihjo = 4
debut(1) = 8.3Ø , debut(2) = 1Ø.3Ø , debut(3) = 14.ØØ
debut(4) = 16.ØØ
le professeur "Callier" intervient dans cet horaire c-à-d

$$val(i+nc) = \text{"Callier"} \qquad 1 \leqslant i \leqslant np$$

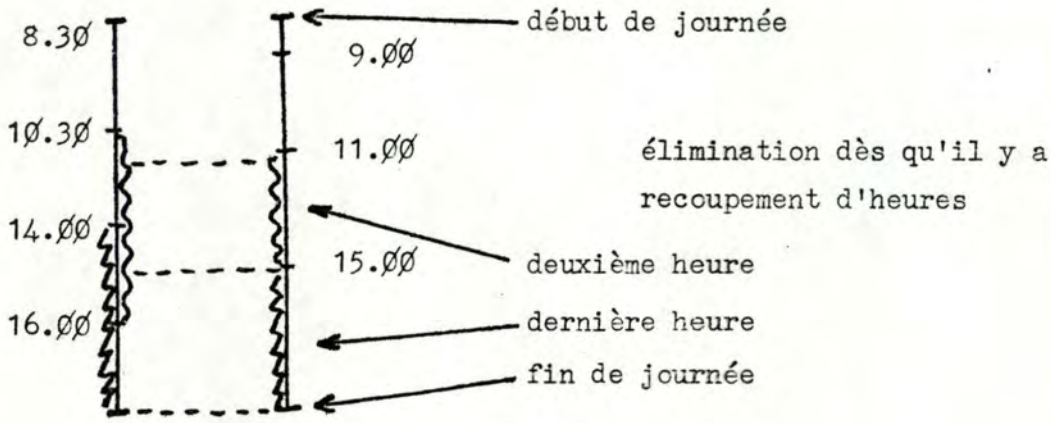
On tient compte de l'horaire déjà calculé pour lequel
nombre d'heures par semaine = 9
nombre d'heures par jour = 3
début des heures de la journée : 9.ØØ , 11.ØØ , 15.ØØ
le professeur "Callier" intervient dans deux cours placés à l'heure 2 et à l'heure 6

Remarquons que l'heure 2 est la deuxième heure du premier jour et l'heure 6 la dernière heure du deuxième jour de l'horaire déjà calculé.
Alors, on gère par programme les indisponibilités suivantes :

pour tout cours i contenant le professeur "Callier"

$L(i,2) = L(i,3) = L(i,7) = L(i,8) = \emptyset$

Ceci provient du fait qu'il faut éliminer toutes les périodes du nouvel horaire qui se recoupent avec celles de l'ancien horaire (auxquelles le professeur "Callier" est pris) :



II.3.11. Cours fixés

Certains cours de l'horaire doivent parfois se donner à des heures précisées d'avance. Après l'introduction des autres contraintes (II.3.3, ...,II.3.1Ø) le programme traite ces cours à fixer d'avance (dans l'ordre du fichier décrit ci-après) et leur donne l'heure en question.

Fichier de type "cours-fixé"

Tout enregistrement a la forme :

1) nom du cours
2) heure à laquelle le cours doit être placé

En parcourant ce fichier, les opérations suivantes sont effectuées :

- chercher le numéro du cours (c-à-d i tel que mame(i) = nom du cours)

- placer le cours i à l'heure proposée (soit ih) c-à-d f(i) = ih
  si L(i,ih) = ∅, afficher un message d'avertissement comme quoi on viole des contraintes moins prioritaires
- faire les éliminations nécessaires (adjacents, contraintes 4 et 5)

Lorsque le cours est inconnu, un message d'avertissement est sorti et le parcours du fichier est continué.
Pour fixer d'avance des cours simultanés, il faut les placer par l'intermédiaire d'un fichier de type "cours-fixé" et en plus les déclarer (au moyen de leurs numéros logiques) comme cours simultanés (pour ne pas avoir des messages d'avertissement comme quoi on viole des contraintes d'indisponibilité).

Remarques :
---

1) L'organisation des fichiers est séquentielle vu que le programme les parcourt séquentiellement avant la coloration proprement dite. Une gestion des fichiers du problème (cfr. III.3.) permet une introduction facile par terminal de toutes les données et contraintes.
2) Les priorités que voici existent parmi les contraintes directes :
   1. Cours fixés (type 9) : les cours sont placés aux heures imposées même si cela viole d'autres contraintes (directes ou indirectes).
   2. Indisponibilités (type 1,2,3 et 8)
   3. Cours simultanés (type 6) : ils sont placés aux mêmes heures en respectant les indisponibilités, mais les contraintes de type 7,4 et 5 ainsi que les contraintes indirectes ne sont pas nécessairement respectées (une classe peut par exemple avoir deux cours à la même heure si ces cours sont déclarés comme cours simultanés).
   4. Cours consécutifs (type 7) : ils sont placés consécutivement en respectant les indisponibilités et les contraintes indirectes.
   5. Professeurs ne désirant pas donner deux cours par jour ou consécutivement (type 4 et 5) : contraintes moins fortes que le type 7 (si un professeur intervient dans une paire de cours consécutifs, ceci violera éventuellement les contraintes de type 4 et 5).

II.4. Coloration d'un cours donné

Etant donné un sommet (c-à-d un cours) choisi dans l'ensemble des sommets du graphe, nous décrivons un procédé de coloration de ce sommet tout en respectant les contraintes imposées (indirectes et directes). Il s'adapte très facilement au cas de coloration par paires de sommets (cfr. méthodes de similarité).

Rappelons d'abord que la lecture préalable des données et contraintes sur fichier fournit les informations suivantes :

ihse : nombre d'heures par semaine
ihjo : nombre d'heures par jour
debut(i), $1 \leq i \leq$ ihjo : début de chaque heure de la journée
nc : nombre de classes
np : nombre de professeurs
nl : nombre de locaux
val(i), $1 \leq i \leq$ nc+np+nl : les classes, professeurs et locaux
n : nombre de cours nulo(i), $1 \leq i \leq n$: les numéros logiques des cours
name(i), $1 \leq i \leq n$: les noms des cours
a(i,k), $1 \leq i \leq n$, $1 \leq k \leq 1Ø$: matrice des cours
f(i), $1 \leq i \leq n$: vecteur des couleurs des sommets (heures des cours)
L(i,ih), $1 \leq i \leq n$, $1 \leq ih \leq$ ihse : matrice de temps
indn2j(i), $1 \leq i \leq$ np : professeurs ne désirant pas deux cours par jour
indn2c(i), $1 \leq i \leq$ np : professeurs ne désirant pas deux cours consécut.
indcsi(i), $1 \leq i \leq n$: cours simultanés
indcco(i), $1 \leq i \leq n$: cours consécutifs

Certains cours sont déjà placés par le traitement des cours fixés (cfr. II.3.11).

II.4.1. Sousroutines de coloration

Considérons un cours (isom) choisi parmi tous les cours; alors, soit il est coloré (c-à-d f(isom) $\neq Ø$ par le traitement des cours fixes ou les tests

supplémentaires (TSTSUP, cfr. plus loin), soit il n'est pas encore coloré. Dans le premier cas, on ne fait rien; dans le deuxième cas, on essaye de le colorer. D'où la sousroutine TSTCOU :

TSTCOU : Si f(isom) = Ø, appeler COLISO ; retour

Pour colorer le sommet isom, essayons de lui donner la première couleur (jcou) possible. Si on ne trouve pas de couleur pour le sommet, afficher un message disant que le cours en question n'a pu être placé.

COLISO : 1) jcou = 1
2) appeler RECHCO
3) si le sommet isom n'est pas coloré
- si jcou < ihse, jcou = jcou + 1
aller en 2)
- sinon , afficher le message que le cours name(isom) n'a pas pu être placé
4) retour (vers la routine appelante)

Essayons de placer le cours isom à l'heure jcou en tenant compte de la matrice de temps :

RECHCO : si L(isom,jcou) = 1 , appeler COUPOS ; retour

Faire une coloration provisoire, calculer les adjacents et faire une élimination provisoire de l'heure jcou pour les adjacents de isom qui ne sont pas des cours simultanés associés à isom. Tester si cette coloration annule la possibilité de coloration de certains sommets non colorés :
- si oui, annuler la coloration ainsi que les éliminations associées
- sinon, faire d'autres tests supplémentaires

COUPOS : 1) f(isom) = jcou
2) calculer les adjacents de isom
3) pour les adjacents j non contenus dans indcsi(isom), L(j,jcou) = Ø
4) Tester si parmi ces adjacents j il y a des sommets tels que f(j) = Ø et Poc(j) = Ø : si oui, f(isom) = Ø et annuler 3)
sinon, appeler TSTSUP
5) retour

Faire les tests supplémentaires suivants :

- cours simultanés
- cours consécutifs
- sommets non colorés dont la possibilité de coloration vaut 1
- contraintes de type 4 et 5

TSTSUP : 1) si indcsi(isom) = $\emptyset$, aller en 2)
---
sinon, essayer de placer les cours simultanés associés à isom
à la même heure : appel à COTST1
si cela réussit, aller en 3)
sinon, annuler 1) et 3) de COUPOS
retour

2) si indcco(isom) = $\emptyset$ aller en 3)
sinon, soit ii = indcco(isom) le cours consécutif associé à isom
essayer de placer ii : appel à COTST2
si cela réussit, aller en 3)
sinon, annuler 1) et 3) de COUPOS
retour

3) Tests des contraintes de type 4 et 5 : TSTELI(isom)

4) Colorer tous les cours i non colorés tels que Poc(i) = 1
et faire les éliminations dues aux adjacents et contraintes de type 4 et 5. Dans le cas où indcsi(i) $\neq \emptyset$ ou indcco(i) $\neq \emptyset$, afficher un message exprimant la priorité des contraintes de disponibilité sur les contraintes de consécutivité et de simultanéité.

5) retour

Lorsque isom fait partie d'un groupement de cours simultanés, il faut essayer de placer ces cours à la même heure que isom (et interdire cette heure aux adjacents et gérer les contraintes de type 4 et 5). Si parmi ces cours simultanés, il y en a qui sont déjà placés (par 4) de TSTSUP ou par le traitement des cours fixés), des messages sont sortis. Il en est de même pour le priorité des cours simultanés sur les cours consécutifs.

COTST1 : 1) Pour chaque cours simultané i non coloré, tester si L(i,jcou) = 1
---
si oui, aller en 2)
sinon, retour

2) si indcco(isom) $\neq \emptyset$, afficher un message de priorité de la contrainte "cours simultanés" sur "cours consécutifs" pour le cours name(isom).

3) pour tout cours simultané i associé à isom :
   si f(i) $\neq \emptyset$, afficher un message comme quoi on ne sait pas réaliser la simultanéïté pour le cours name(i)
   si indcco(i) $\neq \emptyset$, afficher un message de priorité de la contrainte "cours simultanés" sur "cours consécutifs" pour le cours name(i).
   si f(i) $\neq \emptyset$, f(i) = jcou
   interdire jcou aux adjacents de i
   effectuer les tests de type 4 et 5 : appel à TSTELI(i)

4) retour

Pour satisfaire la contrainte des cours consécutifs, voyons si on peut placer les cours ii et isom consécutivement : si oui, essayer les différentes possibilités; sinon, retourner. Si le cours ii fait partie d'un regroupement de cours simultanés, retourner; s'il est déjà placé, afficher un message.

COTST2 : 1) si indcsi(ii) $\neq \emptyset$, retourner (comme si ii était placé)
---

2) si f(ii) $\neq \emptyset$, afficher un message disant que la consécutivité n'est pas nécessairement réalisable pour le cours name(ii); retourner

3) si la condition de consécutivité n'est pas réalisable, retourner
   c-à-d L(ii,jcou+1) = L(ii,jcou-1) = $\emptyset$
   ou L(ii,jcou+1) = $\emptyset$ et jcou en début de journée
   ou L(ii,jcou-1) = $\emptyset$ et jcou en fin de journée

4) si début de journée et L(ii,jcou+1) = 1, appeler ESSAI(jcou+1)
   sinon, si fin de journée et L(ii,jcou-1) = 1, app. ESSAI(jcou-1)
   sinon, si L(ii,jcou+1) = 1, appeler ESSAI(jcou+1)
   si L(ii,jcou-1) = 1 et f(ii) = $\emptyset$, app. ESSAI(jcou-1)

Essayons de placer le cours numéro ii à l'heure proposée (jcou-1 ou jcou+1) sans annuler la possibilité de coloration de l'un ou l'autre de ses adjacents non colorés. En plus, sortir des messages d'avertissement de priorité des contraintes de consécutivité sur les contraintes de type 4 et 5.

ESSAI(jcou) : 1) calculer les adjacents de ii
2) pour ces adjacents j, L(j,jcou) = $\emptyset$
3) tester si parmi ces sommets j il y en a tels que
f(j) = $\emptyset$ et Poc(j) = $\emptyset$
si oui, f(ii) = $\emptyset$ et annuler 2)
sinon, sortir des messages de priorités éventuels (si dans les cours isom et ii on trouve des professeurs k tels que indn2j(k)=1 ou indn2c(k)=1)
f(ii) = jcou (définitivement)
appeler TSTELI(ii)
4) retourner

Eliminations dues aux contraintes de type 4 et 5 : professeurs ne désirant pas donner deux cours par jour (resp. consécutivement). Soit donc le cours isom placé à l'heure jcou. TSTELI effectue alors les éliminations relatives aux contraintes 4 et 5 :

TSTELI(isom) : Pour tous les professeurs j contenus dans le cours isom :
si indn2j(j) = 1; L(i,ih) = $\emptyset$ pour tous les cours i contenant le professeur j et les heures ih de la même journée que l'heure jcou
si indn2c(j) = 1, L(i,ih) = $\emptyset$ pour tous les cours i contenant le professeur j et les heures ih avant et après jcou durant la même journée

On s'aperçoit donc que la coloration d'un sommet choisi constitue la partie la plus importante de tous les algorithmes de coloration. Les sousroutines s'appliquent sans aucun changement lorsque le principe de coloration traite les sommets individuellement (coloration séquentielle, saturation et perturbation). La coloration des sommets par paires par contre exige quelques légères modifications.

II.4.2. Coloration par paires

Les méthodes de similarité (SIMG,SIMT) colorent les sommets par paires. Soit [isom1,isom2] une telle paire, alors SIMG et SIMT deviennent :

- Si un des deux sommets est coloré (soit isom1), appeler RECHCO avec le sommet isom2 et la couleur f(isom1)
- Si aucun des deux sommets est coloré :
  1) jcou = 1
  2) • si jcou < ihse et ni isom1, ni isom2 sont colorés
     - si L(isom1,jcou) = Ø ou L(isom2,jcou) = Ø, aller en 3)
     - sinon { appeler COUPOS(isom1,jcou)
       si L(isom2,jcou) = 1 et f(isom2) = Ø, app. COUPOS(isom2,jcou)
     • sinon, retourner
  3) jcou = jcou + 1 ; aller en 2)

Remarque concernant 2) :

Lorsque le premier sommet de la paire est coloré, quelques tests supplémentaires sont effectués :

- tester si L(isom2,jcou) = 1 puisqu'il pouvait être nul suite aux éliminations dans TSTSUP
- tester si f(isom2) = Ø puisque isom2 pouvait être coloré lors de la coloration de isom1 par l'intermédiaire de TSTSUP (cours simultanés, cours consécutifs, possibilité de coloration égale à 1).

En résumant, nous disposons d'une série de méthodes pour colorer le graphe avec contraintes directes (horaire de cours complet) :

1) méthodes séquentielles : LF,SL,VP,NELE,POCO
2) méthodes de saturation : SAT,EFF
3) méthodes de similarité : SIMG,SIMT
4) méthodes de perturbation : PGLO,PMIN

Comme on vient de le remarquer en II.2.4., les méthodes de permutation n'ont pas été adaptées aux contraintes directes ce qui pouvait d'ailleurs faire l'objet d'une étude dépassant le cadre de ce mémoire.

II.5. L'implémentation

Tous les algorithmes ont été implémentés en FORTRAN sur l'ordinateur Dec-2Ø6Ø et les conventions suivantes concernant les données du problème ont été prises :

- La dimension maximale du problème est limitée à 5ØØ c-à-d qu'un horaire à calculer peut contenir au plus 5ØØ cours. Cela implique les matrices et variables maximales suivantes :
  - a(5ØØ,1Ø) matrice des cours (entière)
  - f(5ØØ) vecteur des heures (entier)
  - name(5ØØ) vecteur des noms des cours (double précision)
  - indcsi(5ØØ) vecteur des cours simultanés (entier)
  - indcco(5ØØ) vecteur des cours consécutifs (entier)
  - n ⩽ 5ØØ nombre de cours

- Le nombre total de classes, professeurs et locaux est supposé inférieur ou égal à 3ØØ et le nombre de professeurs doit être inférieur ou égal à 2ØØ :
  - val(3ØØ) vecteur des noms des classes, professeurs et locaux (double précision)
  - nc + np + nl ⩽ 3ØØ nombre de classes, professeurs et locaux
  - np ⩽ 2ØØ nombre de professeurs
  - indn2j(2ØØ) vecteur des contraintes de type 4 (entier)
  - indn2c(2ØØ) vecteur des contraintes de type 5 (entier)

- Le nombre d'heures par semaine est supposé inférieur à 4Ø et le nombre d'heures par jour inférieur à 8 :
  - ihse ⩽ 4Ø
  - ihjo ⩽ 8
  - debut(8) vecteur de début de chaque heure de la journée (réel) dont chaque composante est de la forme F5.2 c-à-d --.-- : 8.3Ø , 1Ø.35 , 15.ØØ , ...

- Les noms des cours, classes, professeurs et locaux sont des variables alphabétiques qui s'implémentent en double précision ce qui limite leur taille à 1Ø caractères (le Dec-2Ø6Ø ayant des mots de 36 bits).

- La matrice de temps devrait normalement se présenter sous la forme L(5ØØ,4Ø) à éléments binaires. On gagne de la place mémoire très importante par la transformation du code binaire en code décimal. Vu la longueur des mots sur le Dec-2Ø6Ø (36 bits), on parvient à réduire la matrice de temps en une matrice L(5ØØ,2) à élements entiers (à valeur inférieure à $2^{2Ø}$).

- Les matrices de similarité de graphe (S) et de temps (T) devraient avoir la forme S(5ØØ,5ØØ) et T(5ØØ,5ØØ). Comme elles sont symétriques et très creuses (à éléments non binaires), on peut les memoriser par S(8ØØØ,3) et T(8ØØØ,3) où S(k,1) = i, S(k,2) = j, i < j
  S(k,3) = similarité de graphe de i et j si la similarité du couple [i,j] est non nulle (idem pour T)

  On ne reprend donc pas les paires à similarité nulle et les éléments symétriques de la matrice. La borne 8ØØØ s'avère suffisante pour des problèmes à grande taille (près de 5ØØ) et peut être augmentée en cas de dépassement.

- Les numéros logiques doivent être entiers, positifs et inférieurs ou égaux à 999 :

  nulo(5ØØ) vecteur des numéros logiques (entier à valeurs positives, inférieures à 1ØØØ)

C H A P I T R E I I I

Chapitre III : Le programme d'horaire de cours

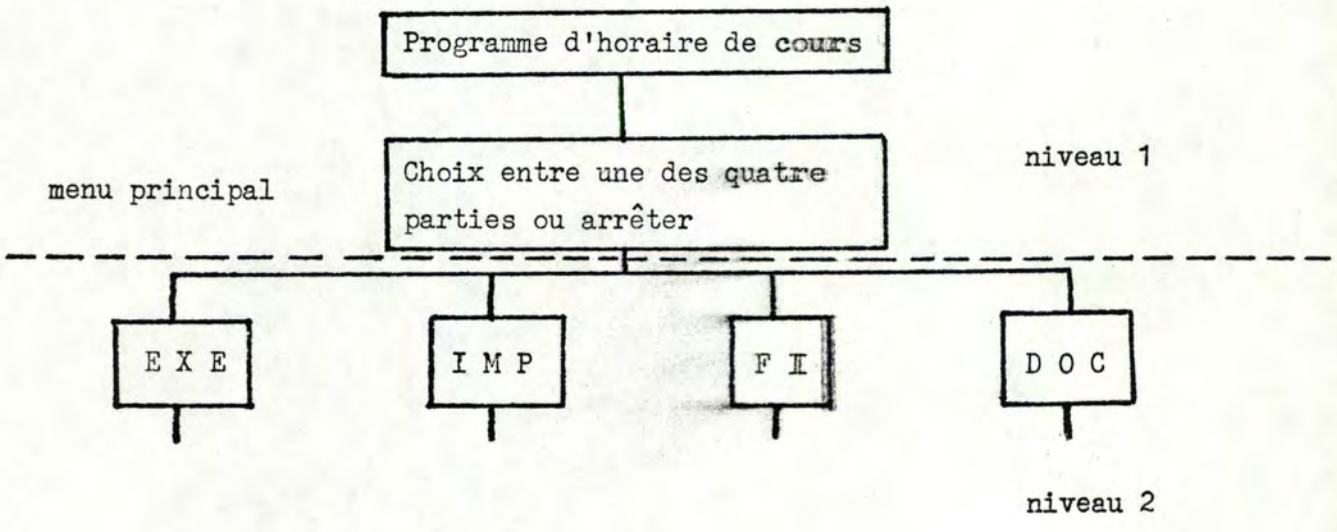
Un logiciel complet a été conçu de manière à ce qu'une personne non-expérimentée en informatique puisse s'en servir facilement. La structure arborescente du programme comprend quatre parties :

exécution (calcul d'horaire proprement dit : EXE)
impressions de résultats (IMP)
gestion de fichiers (FI)
documentation (DOC)

Elles fournissent tous les outils pour résoudre le problème d'horaire de cours et constituent un "software package" complet.

Nous examinons dans la suite les différentes parties du programme et nous décrivons leur fonctionnement pratique. La structure arborescente du programme se base essentiellement sur une décomposition en niveaux sous forme de menus ce qui facilite considérablement le travail de l'utilisateur.

Le menu principal permet les choix entre EXE, IMP, FI, DOC et ARR (où ARR signifie l'arrêt du programme). Suivant l'option prise par l'utilisateur, on branchera vers une des possibilités suggérées.



III.1. Exécution

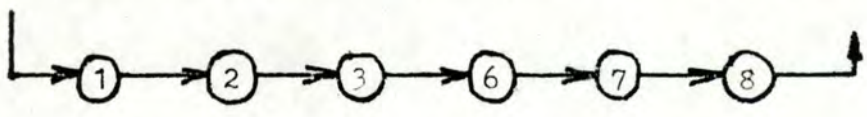
Dans le menu d'exécution, l'utilisateur a le choix entre une série de méthodes de calcul et le retour au menu principal. Ces méthodes sont notamment :

- degré-largest-first (LF)
- degré-smallest-last (SL)
- valeur propre (VP)
- degré de saturation (SAT)
- possibilité de coloration (POCO)
- possibilité de coloration effective (EFF)
- nombre d'éléments (NELE)
- similarité de graphe (SIMG)
- similarité de temps (SIMT)
- perturbation (PER)

L'introduction des données ainsi que la plupart des autres traitements à effectuer sont assez similaires pour les différentes méthodes.
On distingue les traitements

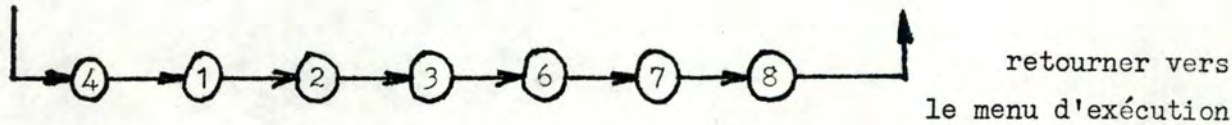
1. Lecture du fichier de type "classe-prof-local"
2. Lecture du fichier de type "cours-à-donner"
3. Lecture des contraintes directes
4. Choix d'une méthode d'ordonnancement secondaire (VP,LF,SL,NELE,POCO)
5. Lecture d'un horaire existant qui servira comme base pour le nouvel horaire à calculer (voir méthodes de perturbation)
6. Calcul d'horaire proprement dit (selon la méthode choisie)
7. Affichage provisoire de l'horaire calculé
8. Mise sur fichier de l'horaire calculé (fichier de type "horaire-calculé")
9. Choix entre le calcul d'un horaire sur fichier ou dernièrement calculé
10. Choix entre perturbation globale (PGLO) et minimale (PMIN)

Pour les méthodes LF,SL,VP,SAT,POCO,EFF et NELE les traitements sont :

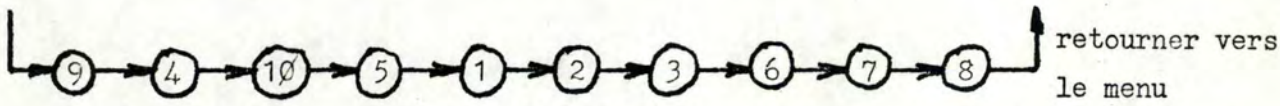


retourner vers le menu d'exécution

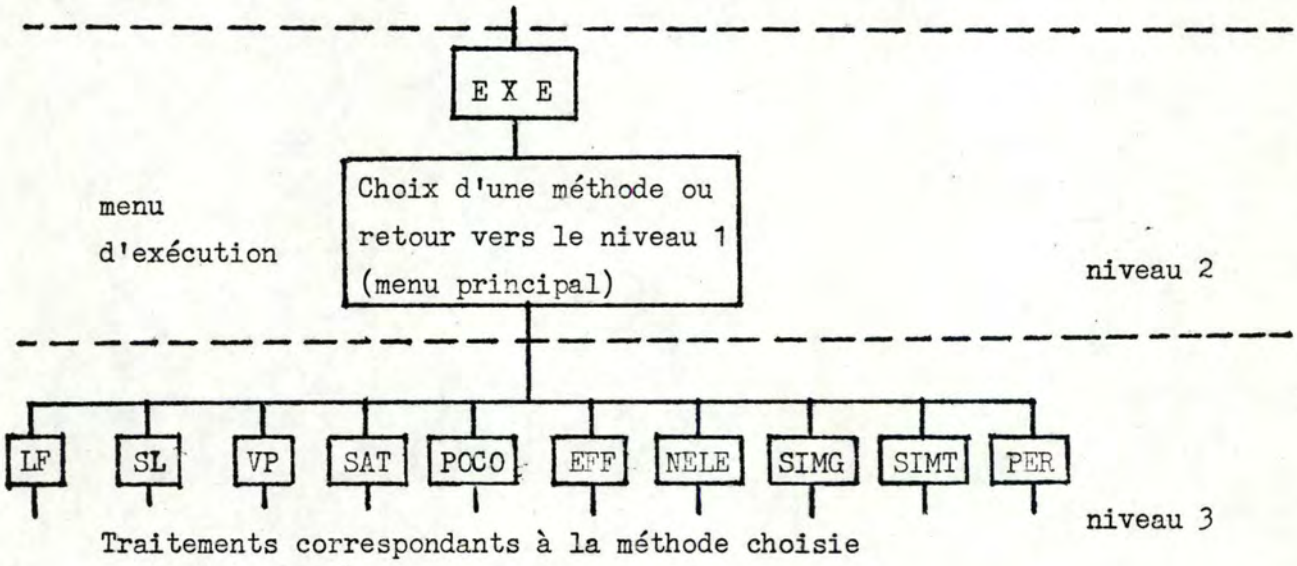
Les méthodes SIMG et SIMT utilisent les traitements que voici :



Les méthodes de perturbation procèdent comme suit :



Tous ces traitements se situent au niveau 3 du programme et se déroulent en séquence comme on l'a indiqué précédemment. Résumons la structure de la partie exécution (EXE) :



III.2. Impression de résultats

Cette partie permet une meilleure visualisation des horaires calculés au moyen d'impressions de tableaux.

On envisage cinq types d'informations :

- horaire total (TOT)
- horaire par classe (CL)
- horaire par professeur (PR)
- horaire par local (LOC)
- horaire de type programme des cours (PRG) c-à-d l'horaire par classe où les cours de même numéro logique (même intitulé) sont regroupés

III.2.1. Horaire total

Imprimer tous les cours de l'horaire groupés par jour et par heure. Ce tableau fournit une vue globale de l'horaire calculé.

HORAIRE TOTAL	début de la prem. heure	début de la 2-ème heure	. . .	début de la dern. heure
nom du premier jour	liste des cours de la prem. heure du prem. jour	liste des cours de la 2-ème heure du prem. jour	. . .	liste des cours de la dernière heure du prem. jour
nom du deuxième jour	liste des cours de la prem. heure du 2-ème jour	liste des cours de la 2-ème heure du 2-ème jour	. . .	liste des cours de la dernière heure du 2-ème jour
:	:	:	:	:
nom du dernier jour	liste des cours de la prem. heure du dern. jour	liste des cours de la 2-ème heure du dern. jour	. . .	liste des cours de la dernière heure du dern. jour

Selon les données du problème (nombre d'heures par semaine et par jour, cours à donner) ce tableau possède une configuration variable en longueur et en largeur. Chaque cours se présente sous la forme : nom du cours, classes, professeurs et locaux. Dans le cas des heures sans cours, des cases vides sont imprimées. Le nombre maximal de cours différents pour une heure donnée doit être inférieur ou égal à 2Ø (limite qu'on s'est imposée et qui semble raisonnable).

III.2.2. Horaire par classe

Pour une certaine classe, on imprime tous les cours dans lesquels cette classe intervient (regroupés par jour et par heure). On obtient ainsi une bonne vue sur les cours d'une classe donnée.

nom de la classe	début de la prem. heure	début de la 2-ème heure	• • •	début de la dern. heure
nom du premier jour	cours à la première heure du prem. jour	cours à la deuxième heure du prem. jour	• • •	cours à la dernière heure du prem. jour
nom du deuxième jour	cours à la première heure du 2-ème jour	cours à la deuxième heure du 2-ème jour	• • •	cours à la dernière heure du 2-ème jour
• • •	• • •	• • •	• • •	• • •
nom du dernier jour	cours à la première heure du dern. jour	cours à la deuxième heure du dern. jour	• • •	cours à la dernière heure du dern. jour

Ce tableau est variable selon les données du problème et chaque cours se présente sous la forme : nom du cours, professeurs et locaux. Remarquons qu'il peut y avoir plusieurs cours différents à la même heure pour une classe donnée (cfr. cours simultanés) c-à-d des cours à option.

III.2.3. Horaire par professeur

Pour un certain professeur, on imprime tous les cours dans lesquels il intervient. Le tableau résultant est analogue à celui décrit en III.2.2. avec la différence que les cours ont la configuration suivante : nom du cours, classes et locaux.

III.2.4. Horaire par local

Etant donné un local, on sort un tableau reprenant tous les cours qui ont lieu dans ce local. Le tableau résultant est similaire à celui de III.2.2. avec des cours sous la forme : nom du cours, classes et professeurs.

III.2.5. Horaire de type programme des cours

Il s'agit de sortir des tableaux qui donnent par classe l'horaire calculé en ne considérant plus les cours par heure à donner (comme c'était le cas pour III.2.1. à III.2.4.), mais en regroupant les cours par intitulé (c-à-d par numéro logique) tels que plusieurs cours à même numéro logique forment un intitulé dont les heures sont déterminées par les cours ayant le numéro logique en question. C'est donc une formulation plus compacte du tableau III.2.2.

Ces types d'impressions sont très utiles pour la mise au point des horaires et la forme du tableau décrit ci-après est adaptée aux conventions prises par les Facultés de Namur.

Soient n1,n2,...,nk les numéros logiques (croissants) des cours d'une classe donnée. On obtient alors le tableau suivant :

nom de la classe	LOCAL	nom du prem. jour	nom du 2-ème jour	. . .	nom du dern. jour
cours n1 nom des professeurs	nom des locaux	heures au prem. jour	heures au 2-ème jour	. . .	heures au dern. jour
cours n2 nom des professeurs	nom des locaux	heures au prem. jour	heures au 2-ème jour	. . .	heures au dern. jour
⋮	⋮	⋮	⋮	⋮ ⋱	⋮
cours nk nom des professeurs	nom des locaux	heures au prem. jour	heures au 2-ème jour	. . .	heures au dern. jour

Lorsque pour un intitulé donné, il y a des différences de locaux pour l'un ou l'autre de ses cours, ces locaux seront imprimés après l'heure à laquelle ils se donnent.

Exemple : soit l'intitulé numéro 13 composé de quatre cours de la forme

nom du cours	numéro logique	classes	prof.	locaux	jour	heure
Mat-de-1	13	Chim1	Delande	CH3	lundi	9.3Ø
Mat-de-2	13	Chim1	Delande	CH3	mardi	1Ø.3Ø
Mat-de-3	13	Chim1	Delande	CH2	jeudi	8.3Ø
Mat-de-4	13	Chim1	Delande	CH3	mardi	11.4Ø

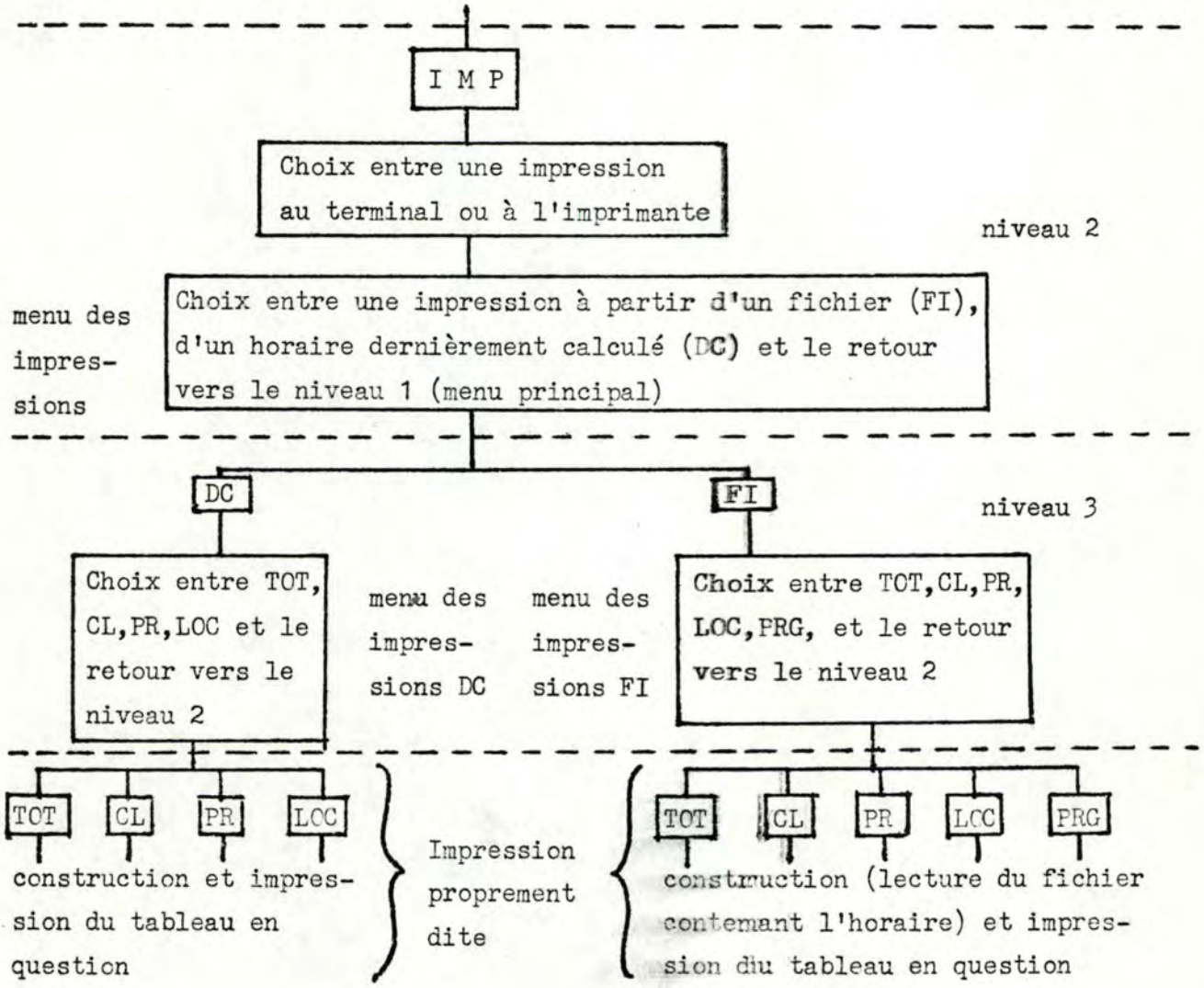
alors le tableau contiendra :

Chim1	LOCAL	lundi	mardi	mercredi	jeudi	. . .
⋮	⋮	⋮	⋮	⋮	⋮	⋮ ⋱
cours 13 Delande	CH3	9.3Ø	1Ø.3Ø 11.4Ø		8.3Ø CH2	
⋮	⋮	⋮	⋮	⋮	⋮	⋮

Pour plus de détails sur les impressions, voyez les exemples de IV.2.

III.2.6. Structure

Comme en III.1., une structuration arborescente guidera l'utilisateur et facilite une exploitation efficace des possibilités d'impression proposées. Selon les besoins de l'utilisateur, les impressions peuvent se faire soit au terminal (intéressant dans le cas d'un terminal papier), soit sur imprimante rapide de l'ordinateur. Les impressions reprennent un horaire dernièrement calculé (en revenant d'une exécution par l'intermédiaire du menu principal) ou un horaire sur fichier. Le choix d'une impression de type programme des cours n'est possible qu'à partir d'un horaire sur fichier.



La structure arborescente mène systématiquement aux impressions, outils indispensables lors du traitement de problèmes d'horaire de grande taille.

III.3 Gestion de fichiers

Cette partie est chargée de créer et de mettre à jour les fichiers dont ont besoin les parties "exécution" et "impression de résultats" afin qu'une exploitation efficace de tous les outils soit garantie. Comme la structure des fichiers est parfois très complexe, il est indispensable de fournir une gestion de fichiers pour un utilisateur non expérimenté en informatique. Nous décrivons les primitves de mise à jour et d'accès, leurs fonctions ainsi que les paramètres correspondants. En plus, une description de la structure de la gestion de fichiers (arborescente) est fournie.

III.3.1. Opérations sur les fichiers

Une étude complète des fichiers et de leur structure a été faite en III.3. Or, la description d'un type de données n'est complète que si à côté de leur structure on spécifie les opérations permises sur les données de ce type. Nous appelerons primitives les opérations de base mises à la disposition du programmeur et de l'utilisateur.
On distingue généralement les primitives de mise à jour (création, modification, suppression) dont la fonction est de modifier les données et les primitives d'accès dont la fonction est de rendre disponible les données. Une primitive est, pour l'utilisateur qui en demande l'exécution, une opération atomique; quelle qu'en soit la complexité, la durée et la portée dans l'espace des données de l'exécution, elle sera toujours soit non exécutée, soit exécutée complètement tout en gardant la structure cohérente du fichier. Chaque primitive peut être caractérisée par ses arguments, sa fonction et ses messages d'erreurs éventuels (dans le cas de non-exécution). Les primitives proposées varient selon le type de fichier et répondent aux besoins réels lors de la mise en place des fichiers du problème. Comme l'organisation des fichiers est séquentielle, les modifications se font en copiant sur un fichier intermédiaire détruit après l'exécution complète de la primitive de modification en question.

III.3.1.1. Fichier de type "classe-prof-local" (CPL)

Comme on vient de le voir en III.3.1., ces fichiers ont la particularité de contenir des enregistrements de type différent :

- nombre d'heures par semaine, par jour, début des heures
- nombre de classes, professeurs et locaux
- classes, professeurs et locaux

Avant de faire n'importe quel traitement sur le fichier, il faut l'initialiser ce qui revient à donner le nombre d'heures par semaine, par jour et le début des heures de la journée (le nombre de classes, professeurs et locaux est mis à zéro par programme).

Les primitives associées sont :

nom de la primitive	arguments	fonction	erreurs
CPLIN	nom du fichier ihse, ihjo, debut	initialiser le fichier	aucune
CPLAC	nom du fichier nom de la classe	ajouter une classe nc = nc + 1	classe existe déjà
CPLAP	nom du fichier nom du professeur	ajouter un prof. np = np + 1	professeur existe déjà
CPLAL	nom du fichier nom du local	ajouter un local nl = nl + 1	local existe déjà
CPLSC	nom du fichier nom de la classe	supprimer une cla. nc = nc − 1	classe n'existe pas
CPLSP	nom du fichier nom du professeur	supprimer un prof. np = np − 1	professeur n'existe pas
CPLSL	nom du fichier nom du local	supprimer un local nl = nl − 1	local n'existe pas
CPLRC	nom du fichier n-c-anc, n-c-nouv	remplacer n-c-anc par n-c-mouv	n-c-anc n'existe pas, n-c-nouv existe déjà

CPLRP	nom du fichier n-p-anc, n-p-nouv	remplacer n-p-anc par n-p-nouv	n-p-anc n'existe pas, n-p-nouv existe déjà
CPLRL	nom du fichier n-l-anc, n-l-nouv	remplacer n-l-anc par n-l-nouv	n-l-anc n'existe pas, n-l-nouv existe déjà

III.3.1.2. Fichier de type "cours-à-donner" (CD)

Ce type de fichier possède aussi bien des primitives de mise à jour que des primitives d'accès.

nom de la primitive	arguments	fonction	erreurs
CDAFN	nom du fichier nom du cours	afficher le contenu du cours	cours n'existe pas
CDANL	nom du fichier numéro logique	afficher les cours ayant ce num. log.	ces cours n'existent pas
CDAFC	nom du fichier nom de la classe	afficher les cours contenant cette classe	ces cours n'existent pas
CDAFP	nom du fichier nom du professeur	afficher les cours contenant ce prof.	ces cours n'existent pas
CDAFL	nom du fichier nom du local	afficher les cours contenant ce local	ces cours n'existent pas
CDAC	nom du fichier nom du cours et son contenu	ajouter le cours	cours existe déjà
CDSC	nom du fichier nom du cours	supprimer le cours	cours n'existe pas

CDMC	nom du fichier nom du cours et son nouveau contenu	modifier le contenu du cours	cours n'existe pas

III.3.1.3. Fichier de type "congé-de-classe" (CI)

nom de la primitive	arguments	fonction	erreurs
CIAFH	nom du fichier nom de la classe	afficher les heures de congé de la cla.	classe n'existe pas
CIAFC	nom du fichier heure	afficher les classes ayant congé à l'heure donnée	classes n'existent pas
CIAJ	nom du fichier nom de la classe heures de congé	ajouter la classe avec ses congés	classe existe déjà
CISU	nom du fichier nom de la classe	supprimer la classe et ses congés	classe n'existe pas
CIMO	nom du fichier nom de la classe heures de congé	modifier les congés de la classe	classe n'existe pas

III.3.1.4. Fichier de type "prof-indisponible" (PI)

Les primitives sont analogues à celles de III.3.1.3. :
PIAFH,PIAFP,PIAJ,PISU,PIMO

III.3.1.5. Fichier de type "local-indisponible" (LI)

Les primitives sont analogues à celles de III.3.1.3. :
LIAFH,LIAFL,LIAJ,LISU,LIMO

III.3.1.6. Fichier de type "prof-à-non-2-cours-par-jour" (CNJ)

nom de la primitive	arguments	fonction	erreurs
CNJAJ	nom du fichier nom du prof.	ajouter le professeur	professeur existe déjà
CNJSU	nom du fichier nom du prof.	supprimer le professeur	professeur n'existe pas
CNJRE	nom du fichier nom-prof-anc nom-prof-nouv	remplacer nom-prof-anc par nom-prof-nouv	nom-prof-anc n'existe pas; nom-prof-nouv existe déjà

III.3.1.7. Fichier de type "prof-à-cours-non-consécutifs" (CNC)

Les primitives sont analogues à celles de III.3.1.6. : CNCAJ,CNCSU,CNCRE

III.3.1.8. Fichier de type "cours-simultanés" (CS)

nom de la primitive	arguments	fonction	erreurs
CSAF	nom du fichier numéro logique	afficher les num. log. des cours sim. associés	numéro logique n'existe pas

CSAJ	nom du fichier les numéros logiques des cours simultanés	ajouter les cours simultanés	numéro logique existe déjà
CSSU	nom du fichier numéro logique	supprimer les cours simultanés de cet enregistr.	numéro logique n'existe pas
CSRE	nom du fichier num-log-anc et les nouveaux num. log.	remplacer les num. log. contenant num-log-anc par les nouveaux num. log.	num-log-anc n'existe pas; un des nouveaux num. log. existe déjà

III.3.1.9. Fichier de type "cours-consécutifs" (CC)

nom de la primitive	arguments	fonction	erreurs
CCAF	nom du fichier nom du cours	afficher le cours consécutif associé au cours donné	cours n'existe pas
CCAJ	nom du fichier nom-cc-1,nom-cc-2	ajouter la paire de cours consécut.	nom-cc-1,nom-cc-2 existe déjà
CCSU	nom du fichier nom du cours	supprimer la paire de cours consécut. contenant le cours	cours n'existe pas
CCRE	nom du fichier nom-cc-anc nom-cc-nouv-1 nom-cc-nouv-2	remplacer la paire contenant nom-cc-anc par nom-cc-nouv-1 et nom-cc-nouv-2	nom-cc-anc n'existe pas; nom-cc-nouv-1, nom-cc-nouv-2 existe déjà

III.3.1.1Ø Fichier de type "horaire-calculé" (HC)

Ce fichier construit par programme (dans "mise sur fichier" de la partie EXE) n'est normalement pas directement accessibile à l'utilisateur puisqu'il ne contient que des horaires calculés. Or, il peut être intéressant de faire des changements manuels de l'horaire sur fichier par les primitives proposées ci-après (mais attentiton au fait que ces modifications peuvent rendre l'horaire incohérent). Ces changements se font donc sous l'entière responsabilité de l'utilisateur et ne portent que sur les heures des cours (pour changer la configuration des cours, utilisez les changements proposés en III.3.1.2. avant une exécution).

nom de la primitive	arguments	fonction	erreurs
HCAFH	nom du fichier nom du cours	afficher l'heure du cours	cours n'existe pas
HCAFC	nom du fichier heure	afficher les cours placés à l'heure donnée	cours n'existent pas
HCMH	nom du fichier nom du cours nouvelle heure	placer le cours à la nouvelle heure	cours n'existe pas

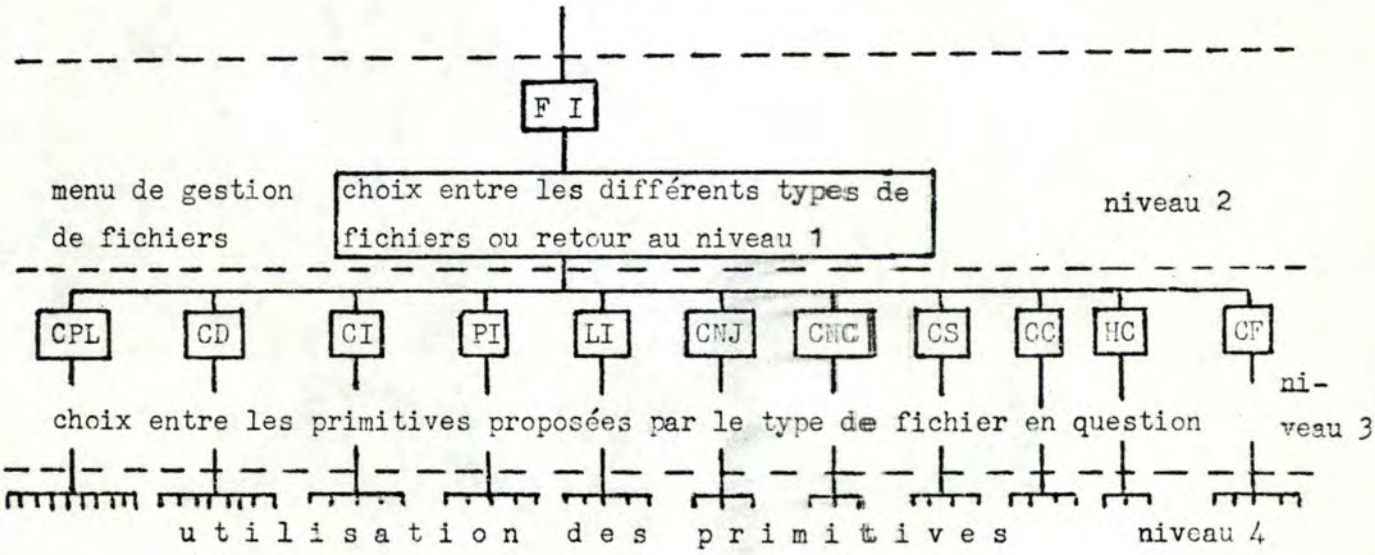
III.3.1.11. Fichier de type "cours-fixé" (CF)

nom de la primitive	arguments	fonction	erreurs
CFAFH	nom du fichier nom du cours	afficher l'heure du cours	cours n'existe pas

CFAFC	nom du fichier heure	afficher les cours fixés à cette heure	cours n'existent pas
CFAJ	nom du fichier nom du cours heure	ajouter le cours fixé	cours existe déjà
CFSU	nom du fichier nom du cours	supprimer le cours	cours n'existe pas
CFRE	nom du fichier nom du cours nouvelle heure	placer le cours à la nouvelle heure	cours n'existe pas

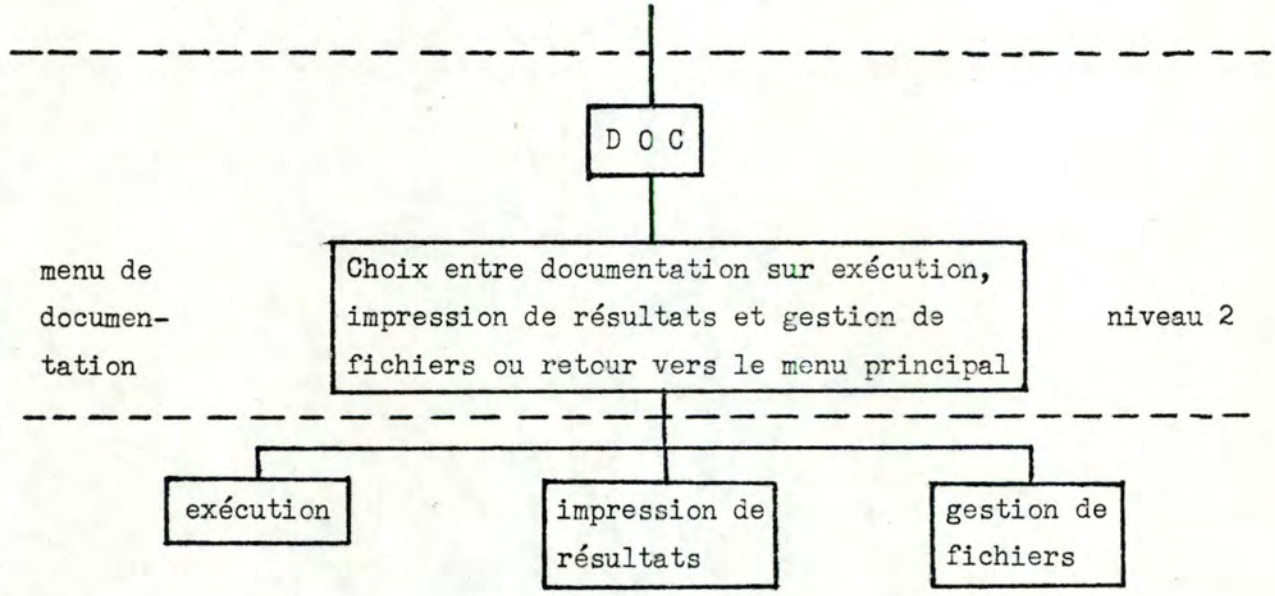
III.3.2. Structure de la gestion de fichiers

Dans le menu de gestion de fichiers (niveau 2), le choix porte sur un type de fichier ou le retour vers le menu principal (niveau 1). Après avoir choisi un type de fichier, le menu associé (niveau 3) permet d'utiliser les primitives proposées (niveau 4) pour ce type ou de retourner vers le menu de gestion de fichiers. Pour quitter le niveau 4, il suffit de répondre aux questions de continuation correspondantes ou d'utiliser l'option "END" (dans le cas des primitives d'ajout).



III.4. Documentation

Elle fournit à l'utilisateur des informations sur la structure et le fonctionnement du programme et décrit donc les trois parties étudiées précédemment : exécution, impression de résultats et gestion de fichiers. Cette documentation donne un résumé des possibilités offertes par le programme. Pour plus de détails, l'utilisateur est référé au mode d'emploi du programme d'horaire de cours sorti à cet effet.



Le passage du niveau 3 au niveau 2 se fait automatiquement lorsque les informations de la partie choisie ont été affichées à l'écran.

C H A P I T R E I V

Chapitre IV : Expériences pratiques

Le programme d'horaire de cours dont la structure est décrite au troisième chapitre, a été implémenté et mis sous forme de "software package". Les tests effectués dans la faculté des sciences et dans l'institut d'informatique, les conclusions pratiques correspondantes ainsi qu'une série d'impressions de tableaux sont décrits dans ce chapitre.

IV.1. Tests effectués

Les tests ont été effectués dans la faculté des sciences (première et deuxième candidature) et à l'institut d'informatique (licences et maîtrises). Le travail d'établissement d'un horaire comprend trois grandes parties :

1) Saisie des données de l'horaire

- Etablir une liste des classes, professeurs, locaux et cours à donner de l'horaire en question. Un travail important consiste donc à fixer la composition des cours à donner et il est conseillé d'y spécifier le plus d'éléments possible (aussi les locaux) par la raison qu'on doit essayer d'obtenir un modèle complet et proche de la réalité.
- Spécifier toutes les contraintes directes : des DESIDERATA (l'annexe 3 décrit un exemple de désiderata) distribués aux professeurs permettent de rassembler les contraintes du type "indisponibilités de professeurs", "professeurs ne désirant pas donner deux cours par jour (resp. consécutivement)","cours consécutifs" et "cours fixés" (pour des professeurs venant de l'extérieur). Les "cours simultanés" sont déterminés en fonction des cours à option. Les "congés de classe" doivent être basés sur les heures des travaux pratiques (si ceux-ci ne sont pas repris dans les cours à donner) et éventuellement sur des raisons pédagogiques (bonne répartition des cours). Les "indisponibilités de locaux" proviennent des heures de nettoyage des locaux ou d'autres raisons. Si des horaires

d'autres facultés ont déjà été calculés, ils représenteront des contraintes supplémentaires pour le nouvel horaire et seront repris sous forme de "horaires calculés".
Ce travail de secrétariat est fondamental pour que l'horaire réponde bien aux exigences réelles. Il est très important d'entrer toutes les informations disponibles pour que l'ordinateur puisse au mieux tenir compte de la situation présente vu "qu'il ne faut jamais supposer que l'ordinateur suppose quelquechose".

2) Encodage des données

Les données préparées en 1) sont entrées par terminal ce qui se fait facilement grâce à la gestion de fichiers.

3) Calcul de l'horaire

Une fois que toutes les données ont été mises sur fichier, on procède au calcul d'horaire proprement dit. Par une des méthodes proposées, on établit un horaire provisoire. Cet horaire peut alors être amélioré en changeant légèrement des contraintes et par l'utilisation des méthodes de perturbation. On obtient ainsi après quelques exécutions un horaire satisfaisant qu'on peut d'ailleurs toujours facilement changer lorsqu'il faut y apporter des modifications (oublis de contraintes ou autres raisons). Les tests effectués mettent en évidence l'importance pratique des méthodes de perturbation.

L'horaire de la faculté des sciences (candidature) pour le deuxième semestre comprend 17Ø cours différents, 2Ø classes, 8Ø professeurs et 3Ø locaux. On a mis 2-3 jours pour l'établir (y compris saisie et encodage des données). Remarquons cependant que les données saisies et encodées (qui demandent le plus de travail) ne changent qu'assez peu d'une année à l'autre et pourront d'ailleurs être reprises l'année prochaine (gain de temps important). L'horaire de cours de l'institut d'informatique a été calculé en fonction de la prochaine réforme de cours. Les tests furent très satisfaisants et mettent en évidence la réalisabilité du projet de réforme prévu.

Faisons quelques constatations pratiques sur le programme d'horaire de cours :

- Avant d'utiliser le programme pour un calcul d'horaire complet, il est conseillé de se familiariser avec son emploi.
- On a intérêt de charger le problème avec le plus possible de contraintes selon les exigences pédagogiques et les desideratas des professeurs. Ces contraintes peuvent être changées d'après les horaires provisoires obtenus.
- Comme la tendance des algorithmes de calcul est de placer les cours en début de semaine, l'utilisateur peut facilement obtenir (par changement des contraintes et par application des méthodes de perturbation) un horaire bien réparti sur toute la semaine. Après quelques calculs successifs, un horaire satisfaisant est établi.
- La formulation des contraintes laisse assez bien de choix à l'utilisateur. Les cours à option par exemple sont introduits soit sous forme de cours à professeurs et locaux multiples, soit sous forme de cours simultanés à professeurs et locaux uniques. Il en est de même pour des cours qui doivent avoir lieu consécutivement : ils sont soit fixés d'avance, soit déclarés comme cours consécutifs. L'utilisateur doit dès lors bien savoir de quelle façon il désire réaliser les contraintes réelles.
- Même si l'ensemble du software package facilite énormement le travail total d'établissement d'horaire, il restera toujours du travail de dactylograhie à faire : remettre tous les tableaux produits sous forme d'un livre ("programme des cours") qui sera distribué aux étudiants, ajouter des explications concernant la répartition des cours sur les deux semestres, donner les intitulés exacts aux numéros logiques,...

IV.2. Tableaux de résultats

Les impressions reproduites ci-après décrivent l'horaire de la faculté des sciences. Les cours se donnent sur des périodes de 65 minutes; chaque journée peut contenir jusqu'à sept cours différents et il peut y avoir des cours le samedi.

Le tableau repris ci-dessous décrit l'horaire du professeur "Mersch" :

HORAIRE MERSCH	8.25	9.30	10.35	11.40	14.00	15.05	16.10
LUNDI					ALG-MER-1 MATH1 M1	ALG-MER-2 MATH1 M1	
MARDI							
MERCREDI					ALG-MER-3 MATH2 M2		
JEUDI		ALG-MER-4 MATH2 M2		GEOM-MER MATH2 M2			
VENDREDI							
SAMEDI							

Le tableau suivant donne un exemple d'un horaire par local : il reproduit l'horaire pour le local "S2".

HORAIRE S2	8.25	9.30	10.35	11.40	14.00	15.05	16.10
LUNDI		CHGE-VE-2 CHIM2 GEOL2 BIOL2 PHARMA2 VERBIST	SCRE-FAV-1 MATH1 PHYS1 CHIM1 GEOL1 FAVRAUX	CHI-DER-1 VT1 DEROUANE			
MARDI	CHI-PA-B1 MEDB PANIER	PHIL-DU-1 MEDA DUCHENE	PHIL-DU-2 MEDB DUCHENE	CHI-PA-B2 MEDB PANIER		CHI-DER-2 VT1 DEROUANE	CHI-VER-1 VT1 VERBIST
MERCREDI	CHGE-VE-1 CHIM2 GEOL2 BIOL2 PHARMA2 VERBIST			CHI-DER-3 VT1 DEROUANE	CHI-VER-2 VT1 VERBIST		
JEUDI				CHI-PA-B3 MEDB PANIER	CHI-VER-3 VT1 VERBIST		
VENDREDI	CHI-PA-B4 MEDB PANIER	SCRE-DU-1 MEDA DUCHENE	SCRE-DU-2 MEDB DUCHENE				
SAMEDI							

Le tableau ci-dessous décrit l'horaire de la classe "Meda" (première candidature medecine groupe a). Les heures libres sont réservées aux travaux pratiques (par la contrainte "congés de classe") qui n'ont pas été repris dans la liste des cours à donner.

HORAIRE MEDA	8.25	9.30	10.35	11.40	14.00	15.05	16.10
LUNDI		HISTO-LE-1 LELOUP A4	CHI-HEV-1 HEVESI CH3	PHY-CAU-A1 CAUDANO S3			
MARDI	BIO-GI-A2 GILLET A3	PHIL-DU-1 DUCHENE S2	PHY-CAU-A2 CAUDANO S3	CHI-GRI-1 GRIFFE CH3			
MERCREDI					CHI-GRI-2 GRIFFE CH3	PHY-CO-A1 COURTOY S3	BIO-DE-A1 DEVOS A4
JEUDI							
VENDREDI	PHY-CO-A2 COURTOY S3	SCRE-DU-1 DUCHENE S2	BIO-GI-A1 GILLET A3	CHI-HEV-2 HEVESI CH3	BIO-DE-A3 DEVOS A4		
SAMEDI	PHY-CO-A3 COURTOY S3	BIO-DE-A2 DEVOS A4					

Le tableau suivant reprend l'horaire de la classe "Meda" de la page précédente, mais sous la forme compacte dite de type "programme des cours" :

MEDA	LOCAL	LUNDI	MARDI	MERCREDI	JEUDI	VENDREDI	SAMEDI
COURS 3 DUCHENE	S2					9.30	
COURS 5 DUCHENE	S2		9.30				
COURS 38 COURTOY	S3			15.05		8.25	8.25
COURS 40 CAUDANO	S3	11.40	10.35				
COURS 42 GRIFFE	CH3		11.40	14.00			
COURS 43 NEVESI	CH3	10.35				11.40	
COURS 44 GILLET	A3		8.25			10.35	
COURS 46 DEVOS	A4			16.10		14.00	9.30
COURS 98 LELOUP	A4	9.30					

C O N C L U S I O N

Le but essentiel de ce travail était de mettre au point un outil efficace de gestion d'horaire de cours. Le "software package" ainsi développé répond aussi bien aux exigences scientifiques qu'aux exigences de gestion.

Comme les tests effectués aux facultés ont été faits (ou du moins surveillés) surtout par des personnes assez qualifiées dans le domaine de l'informatique, il reste à étudier le comportement du programme auprès du personnel administratif. Il est cependant très probable qu'après quelques heures de familiarisation avec le programme, celui-ci devrait être facilement utilisable de façon efficace par une personne non expérimentée en informatique.

Le grand nombre de méthodes de calcul, le caractère complet de la gestion de fichiers et les divers moyens d'impression de résultats soulignent l'énorme utilité de l'ensemble du "software package" pour résoudre le calcul d'horaire de cours.

La recherche de bonnes méthodes de calcul d'horaire a conduit vers un grand nombre d'algorithmes de coloration de graphes. L'introduction du concept de temps lors de la coloration du graphe ouvre un champs énorme de nouveaux algorithmes dont on ne possède pas encore assez de résultats théoriques pour l'instant. Ils vont d'ailleurs faire l'objet d'une étude personnelle que je compte poursuivre et qui portera essentiellement sur la complexité des algorithmes, l'influence de la matrice de temps et sur le choix de l'ordre de traitement des sommets lors du processus de coloration.

A N N E X E S

Annexe 1 : Quelques concepts de la théorie de graphe

D'une façon intuitive, un graphe est un schéma constitué par un ensemble de points $x_1, x_2, \ldots, x_n$, et par un ensemble de flèches reliant chacune deux de ceux-ci, et dénotées $u_1, u_2, \ldots, u_m$. Les point sont appelés sommets du graphe et les flèches les arcs du graphe (cfr. figure 1).

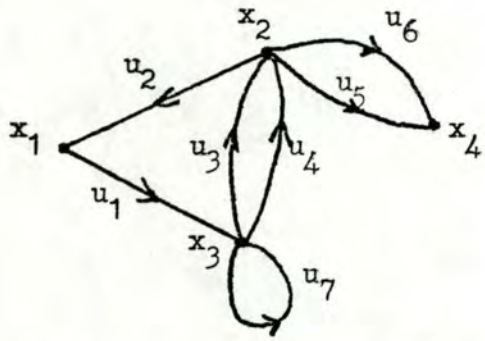


figure 1

L'ensemble des sommets du graphe G se désigne généralement par X, l'ensemble des arcs étant lui-même désigné par U.
L'arc u_1 allant de x_1 à x_3 se note aussi $u_1 = (x_1, x_3)$.

D'une façon plus formalisée, un graphe G = (X,U) est le couple constitué :

1) par un ensemble $X = \{x_1, x_2, \ldots, x_n\}$
2) par une famille $U = \{u_1, u_2, \ldots, u_m\}$ d'éléments du produit cartésien $X \times X = \{(x,y) \text{ tq } x \in X, y \in X\}$;

Un élément (x,y) de $X \times X$ peut apparaître plusieurs fois dans la famille U. Si le nombre d'arcs allant d'un sommet x_i à un sommet x_j ne peut pas excéder un entier p, on dira qu'on a un p-graphe. Le nombre de sommets du graphe G est appelé l'ordre de G. Un arc de la forme (x,x) est appelé une boucle.

On s'intéresse dans la suite (et tout au long de ce travail) aux concepts non-orientés d'un graphe G, c-à-d que la direction des flèches n'importe pas; seul importe le fait de savoir les paires de points reliées, et combien de fois elles sont reliées.

De ce fait, on considère $u_i = (x,y) = (y,x)$ la i-ième arête (abusivement, on utili--se le terme "arc") du graphe. Les sommets x et y sont dits extrémités de l'arc u_i. Au lieu de parler du graphe G(X,U) sans orientation, on parlera aussi du multigraphe G(X,U).

Un multigraphe G(X,U) sera appelé graphe simple (cfr. figure 2) si l'on a les deux conditions :

1) il n'a pas de boucles
2) entre deux sommets, il n'y a jamais plus d'une arête (d'un arc) pour les relier.

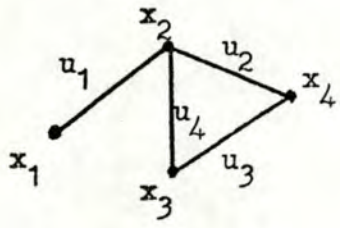


figure 2

Vu l'importance pour le modèle d'horaire de cours (voir I.1), on ne s'intéresse dans la suite qu'aux graphes simples. Les notions définies ci-près vous seront très utiles dans la compréhension de ce travail. Les graphe G est donc supposé d'être un graphe simple.

1) Sommets adjacents : Etant donné $x \in X$, alors $y \in X$ est adjacent à x si l'arc $(x,y) \in U$.

De façon plus générale, deux sommets sont adjacents, s'ils sont liés par un arc du graphe. Le degré d'un sommet est le nombre de ses adjacents. Notons par $d_i = \deg(x_i)$ le degré du sommet x_i.

2) Graphe complet : G(X,U) est complet si pour tout $x,y \in X$, $(x,y) \in U$. Donc, un graphe est complet si chaque paire de sommets est liée par un arc. Un graphe complet de n sommets s'appelle une n-clique.

3) Graphe biparti : Un graphe est biparti si l'ensemble de ses sommets peut être partionné en deux classes X_1 et X_2 de sorte que deux sommets de la même classe ne soient jamais adjacents. Il se note parfois $G = (X_1, X_2, U)$. Un graphe k-parti est la généralisation à k classes.

4) Graphe planaire : Un graphe est dit planaire s'il est possible de le représenter sur un plan de sorte que les sommets soient des points distincts, les arêtes des courbes simples et que deux arêtes ne se rencontrent pas en dehors de leurs extrémités.

5) Sous-graphe : Le sous-graphe de G engendré par $A \subset X$ est le graphe G_A dont les sommets sont les points de A, et dont les arcs sont les arcs de G ayant leurs extrémités dans A.
Donc $G_A = (A, U_A)$ où $U_A = \{(x,y) \in U \text{ tq } x,y \in A\}$.

6) Graphe partiel : Le graphe partiel de G engendré par $V \subset U$ est le graphe (X,V) dont les sommets sont les points de X, et dont les arcs sont ceux de V. Autrement dit, on élimine de G les arcs de U - V.

7) Chaîne : Une chaîne de longueur $q > \emptyset$ est une séquence $\mu = (u_1, u_2, \ldots, u_q)$ d'arcs de G telle que chaque arc de la séquence ait une extrémité en commun avec l'arc précédent, et l'autre extrémité en commun avec l'arc suivant. Le nombre d'arcs de la séquence est la longueur de la chaîne μ. Une chaîne qui ne rencontre pas deux fois le même sommet est dite élémentaire; une chaîne qui n'utilise pas deux fois le même arc est dite simple.

8) Cycle : C'est une chaîne $\mu = (u_1, u_2, \ldots, u_q)$ telle que :
1) le même arc ne figure pas deux fois dans la séquence.
2) les deux sommets aux extrémités de la chaîne coïncident.

9) Connexité : Un graphe est connexe si pour toute paire de sommets distincts, il y a une chaîne qui les relie.
La relation "x = y, ou $x \neq y$ et il existe une chaîne dans G reliant x et y" est une relation d'équivalence (réflexive, symétrique et transitive). Les classes de cette équivalence constituent une partition de X en sous-graphes connexes de G, appelés composantes connexes de G. Le graphe de la figure 3 admet deux composantes connexes.

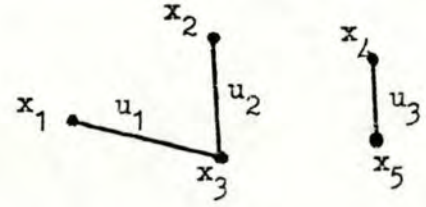


figure 3

1Ø) Nombre chromatique : On appelle nombre chromatique d'un graphe G le plus petit nombre de couleurs nécessaire pour colorier les sommets, de sorte que deux sommets adjacents distincts ne soient pas de même couleur. On le désigne par $\chi(G)$.
Un graphe G avec $\chi(G) \leq k$ peut être coloré avec k couleurs. Une telle coloration, dite k-coloration, partitionne l'ensemble des sommets en k ensembles $S_1, S_2, \ldots, S_k$ de façon à ce que tous les sommets d'un ensemble donné prennent la même couleur.

11) Graphe critique : Un graphe simple G est dit χ-critique si pour tout sommet x, le sous-graphe engendré par $X - \{x\}$ a un nombre chromatique strictement inférieur à celui de G.

Annexe 2 : Méthodes de permutation généralisée

Rappelons que les méthodes de permutation décrites en I.5. se basent sur la recherche de permutations bichromatiques en choisissant les couleurs i et j de la façon suivante :

1) $i,j \in K$ où $k \in K \iff$ exactement un sommet de $\langle x_1,x_2,\ldots,x_{l-1}\rangle$ adjacent à x_l a la couleur k; x_l étant le sommet à colorer.

2) une i,j composante de $\langle x_1,\ldots,x_{l-1}\rangle$ n'a qu'un sommet adjacent à x_l en $\langle x_1,\ldots,x_l\rangle$.

Si ces deux conditions sont satisfaites, une $i \leftrightarrow j$ permutation est possible sur une telle i,j composante de $\langle x_1,\ldots,x_{l-1}\rangle$. L'effet de cette permutation est de libérer une des couleurs i ou j pour le sommet x_l (voir aussi l'exemple de I.5).

Nous proposons ici un autre critère de sélection des couleurs i,j afin d'effectuer une permutation bichromatique (cfr. remarque de I.5) :

(*) toute i,j composante admet des adjacents à x_l en $\langle x_1,\ldots,x_l\rangle$ possèdant tous la même couleur c-à-d que tous les adjacents à x_l sur $\langle x_1,\ldots,x_l\rangle$ d'une i,j composante quelconque sont soit colorés i, soit colorés j.

La condition (*) contient les conditions 1) et 2) comme un cas particulier et en représente une généralisation importante.
L'algorithme de permutation décrit en I.5. (basé sur les conditions 1) et 2)) est à la base de l'algorithme suivant (utilisant la condition (*) et qui sera dit algorithme de permutation généralisée); on ne fait que remplacer le critère de choix de la permutation bichromatique.

Algorithme de permutation généralisée :

1) Ordonnancement des sommets du graphe par une des quatre méthodes étudiées en I.2. (LF,SL,VP,NELE).

2) Colorer x_1 avec 1 et éliminer l'heure 1 aux adjacents de x_1;
$i = 2$; $j = 1$

3) - Si $i > n$, arrêter
- Sinon, soit $\langle x_1, x_2, \ldots, x_{i-1} \rangle$ coloré avec $1, 2, \ldots, j$
  - si on parvient à colorer x_i avec $1, \ldots, j$, colorer x_i et interdire $f(x_i)$ aux adjacents de x_i.
  - sinon, - si pour $\alpha, \beta \in \{1, \ldots, j\}$ toute α, β composante admet des adjacents à x_i sur $\langle x_1, \ldots, x_i \rangle$ possédant tous la même couleur, alors faire une $\alpha \leftrightarrow \beta$ permutation soit sur toutes les α, β composantes ayant des adjacents à x_i colorés α, soit sur toutes les α, β composantes ayant des adjacents à x_i colorés β. Colorer les sommets $x_1, \ldots, x_{i-1}$ de même que dans cette nouvelle coloration de $\langle x_1, \ldots, x_{i-1} \rangle$ et x_i avec la nouvelle couleur possible, soit α ou β, et une j-coloration de $\langle x_1, \ldots, x_i \rangle$ est obtenue; interdire $f(x_i)$ aux adjacents de x_i.
    - sinon, une telle permutation n'est pas possible; colorer $x_1, \ldots, x_{i-1}$ de même qu'en $\langle x_1, \ldots, x_{i-1} \rangle$ et colorer x_i avec $j+1$; interdire $j+1$ aux adjacents de x_i. Ainsi, une $j+1$-coloration de $\langle x_1, \ldots, x_i \rangle$ est obtenue; $j = j + 1$.

4) $i = i + 1$; aller en 3)

Comme on vient de le voir au paragraphe I.7., les méthodes de permutation permettent de diminuer la borne supérieure moyenne du nombre chromatique de 8 % par rapport aux méthodes séquentielles. Les résultats numériques suivants étudient l'amélioration des méthodes de permutation généralisée par rapport aux méthodes de permutation et aux méthodes séquentielles. Pour obtenir ces résultats, on colore les sommets de façon séquentielle sans les ordonner par une des méthodes vues. Ceci permet de bien voir la différence entre le processus de coloration séquentielle et de coloration par permutation.
Les trois types de méthodes (séquentielles, permutation, permutation généralisée) ont été testés sur des problèmes de dimension 2Ø, 6Ø et 1ØØ et

des matrices d'adjacence du graphe à taux de remplissage de 2Ø %, 4Ø %, 6Ø % et 8Ø %. Les bornes supérieures moyennes du nombre chromatique s'obtiennent sur 25 cas différents ce qui fournit un jeu de test total de 3ØØ (= 3 * 4 * 25) cas distincts.

	2Ø %	4Ø %	6Ø %	8Ø %
séquentiel	4.2Ø	6.Ø8	8.24	1Ø.52
permutation	3.72	5.44	7.24	9.76
permut. génér.	3.64	5.12	7.16	9.72

← taux de remplissage de la matrice M

méthode utilisée ↑

Tableau 1 : graphe à 2Ø sommets

	2Ø %	4Ø %	6Ø %	8Ø %
séquentiel	7.96	12.4Ø	17.92	24.6Ø
permutation	7.12	11.44	16.ØØ	22.52
permut. génér.	6.52	1Ø.44	15.Ø8	21.52

Tableau 2 : graphe à 6Ø sommets

	2Ø %	4Ø %	6Ø %	8Ø %
séquentiel	1Ø.8Ø	17.56	25.12	36.84
permutation	9.8Ø	16.28	23.6Ø	33.64
permut. génér.	9.Ø4	15.16	22.Ø8	32.44

Tableau 3 : graphe à 1ØØ sommets

Les performances de la méthode de permutation généralisée sont très remarquables : elle améliore la méthode de permutation de $\pm$ 5 %; le gain par rapport à la méthode séquentielle est d'ailleurs de $\pm$ 13 % (13 = 5 + 8).

La classe des méthodes de perturbation généralisée fournit les meilleurs résultats parmis toutes les méthodes étudiées dans ce travail (vu le fait que les méthodes de saturation et de permutation étaient les plus performantes jusqu'à présent).

De même qu'en II.2.4 pour les méthodes de permutation, on peut adapter les méthodes de permutation généralisée aux contraintes de temps ce qui devrait également souligner leur qualité.

Annexe 3 : Exemple de Desiderata

Le Desiderata repris à la page A.1Ø permet d'établir les contraintes et données nécessaires à un calcul d'horaire complet. Ces contraintes sont souvent de nature pédagogique ou portent sur des indisponibilités.

Le Desiderata en question contient :

- une liste des cours du professeur en question
- les indisponibilités
- cours fixés
- contraintes du type "non-deux cours-par-jour"
- regrouper les cours sur certains jour revient à créer des indisponibilités

Les professeurs formulent des désirs supplémentaires (cours consécutifs,...) sur le verso de la feuille.

1. NOM DE L'ENSEIGNANT : .....

2. LISTE DES COURS A L'INSTITUT (2ème SEMESTRE)

1. .....
2. .....
3. .....
4. .....
5. .....
6. .....

3. HEURES INDISPONIBLES (ou souhaitées libres)

	mardi	mercredi	jeudi	vendredi
8.30				
10.40				
14.00				

Les croix déjà présentes correspondent aux cours que vous donnez en Sciences Economiques ou en Mathématique.
En cas d'omission, veuillez compléter s.v.p.

4. HEURES OBLIGATOIRES (par ex. pour un cours commun avec la candi ou la licence en Sciences Economiques, ou n'importe quelle autre raison)

	mardi	mercredi	jeudi	vendredi
8.30				
10.40				
12.40				

Veuillez indiquer le numéro du cours de la liste 2 dans la case correspondante. Nous avons déjà noté ceux que nous connaissons; en cas d'omission, veuillez compléter s.v.p.

5. REPARTITION : 1 Souhaitez-vous grouper vos cours sur certains jours de la semaine, si oui, lesquels ?

2 Inversément, souhaitez-vous ne pas avoir deux cours le même jour
Répondez par oui ou par non.

B I B L I O G R A P H I E

[1] R.L. BROOKS, "On colouring the nodes of a network", Proceedings Cambridge Phil. Society 37, 194-197 (1941)

[2] H.S. WILF, "The Eigenvalues of a Graph and Its Chromatic Number", J. London Math. Society, à paraître

[3] D.J.A. WELSH, M.B. POWELL, "An upper bound for the chromatic number of a graph and its application to timetabling problems", Computing Journal, 1Ø, 85-86 (1967)

[4] J.A. BONDY, "Bounds for the Chromatic Number of a Graph", Journal of Combinatoriel Theory 7, 96-98 (1969)

[5] D.W. MATULA, "A min-max theorem for graphs with application to graph coloring", SIAM Revue 1Ø, 481-482 (1968)

[6] D.W. MATULA, "k-components, clusters and sliclings in graphs", SIAM Journal Apll. Math. 22, n° 3, 459-48Ø (1972)

[7] SZEKERES G., WILF H.S., "An inequality for the chromatic number of a graph", Journal of Combinatorial Theory 4, 1-3 (1968)

[8] D.W. MATULA, G. MARBLE, J.D. ISAACSON, "Graph coloring algorithms" Graph Theory and Computing, Academic Press, New York, 1Ø9-122 (1972)

[9] ORE O. "The Four Color Problem", Academic Press, New York, 1972

[1Ø] BERGE C., "Graphes et Hypergraphes", Dunod Université, 197Ø

[11] SHANNO D.F., "New Methods to Color the Vertices of a Graph", Communications of the ACM 22, n° 4, 251-256 (1979)

[12] WOOD D.C., "A technique for coloring a graph applicable to large scale timetabling problems", Computer Journal 12, 317-319 (1969)

[13] RANDALL-BROWN J., "Chromatic Schedulding and the Chromatic Number Problems", Management Science 19, n° 4, part 1, 456-463 (1972)

[14] J.A. FORMBY, "A computer Procedure for bounding the Chromatic Number of a Graph", University of Birmingham (England), Combinatorial Mathematics and its applications, Welsh, Academic Press (1971)

[15] M. BEHAZED, "The Total Chromatic Number of a Graph : a Survey", National University of Iran, Combinatorial Mathematics and its applications, Welsh, Academic Press (1971)

[16] A. TEHRANI, "Un algorithme de coloration", Cahiers du Centre d'Etudes de recherche opérationnelle, Bruxelles, vol 17, 395-398 (1975)

[17] K. MAGHOUT, "Applications de l'algèbre de Boole à la théorie de graphes et aux programmes linéaires et quadratiques", Cahiers du Centre d'Etudes de recherche opérationnelle, Bruxelles, vol 5, 193... (1963)

[18] CORNEIL D.G., GRAHAM B., "An algorithm for determining the chromatic number of a graph", SIAM Journal of Computing, vol 2, n° 4, 311-318 (1973)

[19] N. CHRISTOFIDES, "An algorithm for the chromatic number of a graph", The Computer Journal 14, n° 1, 38-39 (1970)

[20] A. DEFRENNE, "The timetabling problem : a survey", Cahiers du Centre d'Etudes de recherche opérationnelle, Bruxelles, 163-169 (1978)

[21] D.F. SHANNO, "On Maintenance of the Opportuny List for Class-Teacher Timetable Problems", Communications of the ACM 18, n° 4, 203-208 (1975)

[22] D.F. SHANNO, Graph Coloring Conditions for the Existence of Solutions to the Timetable Problem", Communications of the ACM 17, n° 8, 450-453 (1974)

[23] C.J.H. Mc DIARMID, "The solution of the Timetabling Problem", J. Inst. Maths Applications 9, 23-34 (1972)

[24] D.S. JOHNSON, "Approximation Algorithms for the Combinatorial Problem", Journal of Computer and System Sciences 9 , 256-278 (1974)

[25] D.S. JOHNSON,"Worst Case Behavior of Graph Coloring Algorithms", Proceedings of the 5th Southeast Conference on Combinatorics, Graph Theory, and Computing, 513-527 (1974)

[26] F.TH. LEIGTHON, "A graph coloring algorithm for large schedulding problems", Journal of Research of the National Bureau of Standards, n° 6, 489-506 (1979)

[27] P.A. CATLIN, "A bound on a chromatic number of a graph", Disrete Math.,1978,22,n° 1, 81-83

[28] S. WAGON, "A bound on the chromatic number of a graph without certain induced subgraphs", Journal of combinatorial theory, B, 1980, 29, n° 3, 345-346

[29] N. CHIBA, "A linear algorithm for five-coloring a planar graph", Lecture of notes comput. sciences, 1981, 1Ø8, 1-8

[3Ø] V.A. AKSIONOV, "On uniquely 3-colorable planar graphs", Discrete Math., 1977, 2Ø, n° 3, 2Ø9-216

[31] J. PLESNIK, "Coloring of graphs by partionning", Math. Slov., 198Ø, 3Ø, n° 3, 121-127

[32] P. DUCHET, "Colorations transitives des graphes", Cahiers du Centre d'Etude et de Recherche opérat., 1978, 2Ø, n° 3-4, 363-371