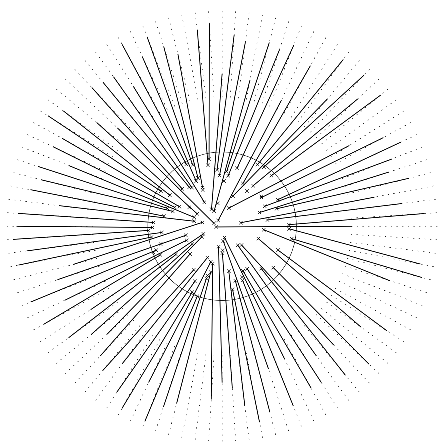
Francis SOURD
DEA d'Algorithmique

Assignation de fibres optiques pour la spectroscopie multi-objets



Stage réalisé entre le 24 mars et le 1er juillet 1997 sous la direction de

Philippe CHRETIENNE
Université Paris 6 - LIP6
4, Place Jussieu
75252 Paris Cedex

Paul FELENBOK
Observatoire de Paris - DAEC
92195 Meudon Cedex

Résumé

Le problème d'assignation de fibres à des objets vient de l'astronomie. Ce problème de géométrie et de combinatoire vise à disposer dans le plan des segments – ou des rectangles – sans intersections. Pour résoudre ce problème, nous avons construit une suite d'algorithmes qui converge vers la solution maximale. En fait, les premiers éléments de cette suite donnent de très bons résultats en un temps bref. De plus, ils peuvent être améliorés par des techniques de post-optimisation basées sur ces mêmes heuristiques. Les bons résultats obtenus ont permis de lancer une recherche de la solution exacte. On montre ainsi que les méthodes heuristiques trouvent généralement la solution optimale en quelques minutes.

Abstract

Fiber-to-Object Assignment Problem comes from Astronomy. This geometrical and combinatorial problem arranges the maximum of segments – or rectangles – in the plane without crossovers. In order to solve this problem, we have recursively built a series of heuristic algorithms that converge on a maximal solution. Anyway the first elements of the series give quickly very good results. They can be improved by post-optimization heuristics based on the same heuristics. The good results given by these algorithms enables the search of the exact solution. Hence it shows that the heuristics are able to reach the best solution within a few minutes.

Table des matières

Table des figures

Liste des tableaux

Résumé des notations

N	Nombre de fibres
n	Nombre d'objets à observer
$\mathcal{F}$	Ensemble des fibres
$\mathcal{O}$	Ensemble des objets à observer
F_i	Fibre i, pour $1 \leq i \leq N$
O_i	Objet i, pour $1 \leq i \leq n$
$\langle F_i, O_j \rangle$	Assignation de la fibre F_i avec l'objet O_j
$\langle F_i, \emptyset \rangle$	La fibre F_i n'est pas assignée
$\langle \emptyset, O_j \rangle$	L'objet O_j n'est pas observé
A	Ensemble des assignations déjà faites
O^i_{acc}	Ensemble des objets accessibles pour la fibre F_i
$O^{(i,A)}_{acc}$	Ensemble des objets accessibles pour la fibre F_i, les assignations de A étant effectuées
l_{tige}	Longueur de la tige contenant la fibre optique

Introduction

Cadre du travail

Le problème qui est l'objet de ce rapport s'est posé en Astronomie. Afin de connaître la vitesse d'objets observables (étoiles, galaxies, amas...), on utilise la spectroscopie. Les raies d'émission et d'absorption de ces objets nous parviennent avec un décalage dû à l'effet Doppler ce qui permet, grâce à la connaissance des raies données par une source de calibration, de retrouver leur vitesse.

Le temps d'enregistrement par le spectrographe d'un objet est de l'ordre d'une heure. De plus, dans le champ d'un télescope, un grand nombre d'objets sont généralement observables. Afin de gagner du temps d'observation, la technique de spectroscopie multi-objets a été mise au point. A son arrivée dans le plan focal, la lumière des différents objets est captée par une fibre optique et conduite jusqu'à l'entrée du spectrographe.

Le travail réalisé pendant la durée de ce stage de DEA s'est fait dans le cadre du projet FUEGOS (Fiber Unit for European General Optical Spectrograph). Cet instrument doit comporter une centaine de fibres, ce qui permet d'analyser simultanément la lumière d'autant d'objets. L'instrument doit être monté sur le VLT (Very Large Telescope) en 2001. Ce télescope est actuellement construit par l'ESO (European Southern Observatory)[1] sur le site du Cerro Paranal au Chili (figure 0.1).

FIG. 0.1: Le site du VLT sur le Cerro Paranal.

[1]Voir le `http ://www.eso.org/vlt/`

Présentation du problème

Le problème à résoudre consiste à trouver comment disposer les fibres optiques de FUEGOS en prenant en compte certaines contraintes essentiellement mécaniques. Ainsi, les fibres optiques doivent être maintenues en position rectiligne. En effet, à chaque réflexion de la lumière conduite au sein de la fibre, une fraction de la lumière est absorbée et donc perdue. Comme il est impossible, pour des raisons mécaniques de croiser des fibres optiques, il est alors clair que dire quelle fibre recevra la lumière de quelle étoile – problème d'*assignation* – n'est pas une chose aisée.

Le problème à résoudre est de trouver non seulement une mise en place des fibres qui satisfasse à toutes les contraintes mécaniques mais surtout une mise en place qui permette de capter la lumière d'un maximum d'objets.

Le chapitre 1 présente plus en détail ce problème en tâchant de le définir mathématiquement.

Méthodes employées

Les problèmes d'optimisation combinatoirement sont souvent très longs à résoudre exactement. Pour cette raison, des méthodes dites *heuristiques*, visant à rechercher de bonnes solutions au problème sans vraiment se soucier de leur optimalité, ont été mises en œuvre. Une fois ce travail effectué, il a en fait été possible d'implanter un algorithme recherchant la solution optimale du problème et la trouvant en un temps "raisonnable".

Après un résumé de différents travaux effectués sur le sujet, ce qui présente différentes approches possibles de résolution du problème, le chapitre 2 se penche plus précisément sur un algorithme heuristique qui sera à la base du travail des chapitres suivants.

Le chapitre 3 présente justement différentes méthodes qui améliorent notablement cet algorithme. Elles sont présentées à la fois dans un cadre théorique, car elles semblent assez facilement généralisables, et dans le cadre du problème, ce qui permet de mettre en évidence quelques astuces spécifiques à l'assignation de fibres optiques.

Finalement, le chapitre 4 présente toutes les propriétés mathématiques qui ont permis d'implanter un algorithme – en fait un traditionnel *branch-and-bound* – prouvant le caractère optimal de la solution trouvée. On découvre ainsi que les heuristiques développées dans le chapitre 3 conduisent rapidement à une solution optimale.

Chapitre 1

Modélisation

Nous étudions dans ce chapitre les propriétés essentiellement mécaniques des différents constituants de l'instrument pour arriver à les modéliser de manière correcte mais pas trop contraignante. On arrive ainsi à reformuler le problème de manière complètement mathématique.

1.1 Vue générale

L'instrument FUEGOS peut être divisé en trois parties :

- le **spectrographe** situé à l'arrière de la focale ;
- les **fibres** chargées de conduire la lumière de la surface focale jusqu'au spectrographe ;
- le **positionneur** qui dispose les fibres et les maintient droites.

C'est bien entendu le positionneur qui nous intéresse particulièrement. Pour le reste, il nous suffit de savoir que chaque fibre optique rejoint le spectrographe en passant par un système rétracteur situé sur un axe radial de la surface focale auquel est attaché la tige chargée de maintenir droite la fibre optique. Ce point d'attache se trouve sur un chariot dont la course est représentée par les segments pointillés dans la figure 1.1. Comme la tige est de longueur constante, le chariot bouge en fonction de l'endroit ou l'on pose l'autre extrémité de la fibre optique, appelée bouton, chargée de capter la lumière.

La figure 1.1 représente les élements pris en compte dans la modélisation du problème.

- Le **disque focal** est représenté par le cercle intérieur. Il contient des croix représentant les points d'impact de la lumière des objets.
- Les **tiges** sont les segments gras. Chacun de ces segments a une extrémité sur un objet et l'autre joint le spectrographe via le rétracteur.
- La **course du chariot** supportant le rétracteur est représentée en pointillés.

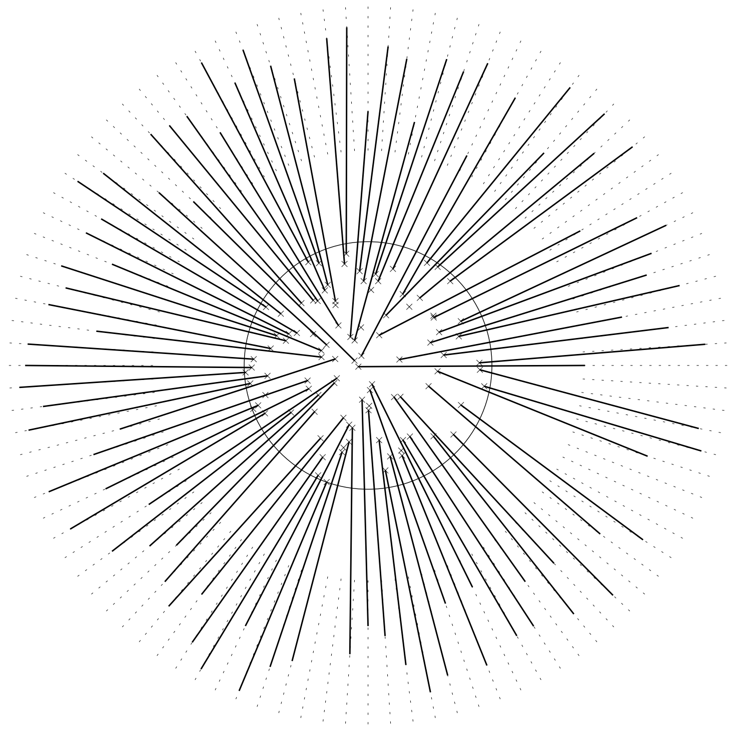


FIG. 1.1: Modélisation du positionneur

Les sections 1.2 et 1.3 justifient et précisent une telle modélisation.

REMARQUE : Il arrive très souvent que certaines tiges ne soient pas utilisées lors d'une observation – en général parce qu'aucun objet ne leur est accessible. Ces tiges ne sont pas représentées sur la figure 1.1. En effet, elles sont ignorées dès qu'on décide de ne pas les utiliser. Mécaniquement, un espace est prévu pour les poser de manière à ce qu'elles ne dérangent pas les fibres en place sur la focale.

1.2 La surface focale

1.2.1 Sphéricité de la surface focale

On a parlé ci-dessus de disque focal, comme si tous les objets arrivaient au point dans un plan contenant ce disque. A la vérité, il ne s'agit pas d'un plan mais d'une sphère, dont le rayon est beaucoup plus grand que le rayon du disque. Cette surface est donc une calotte sphérique de hauteur bien inférieure à son diamètre.

En se plaçant dans une "vue de dessus", la calotte sphérique devient disque mais il y a une modification par rapport au modèle formulé dans la section précédente. En effet, comme les tiges ne se trouvent plus être parfaitement verticales leur projection donne des segments dont la longueur varie en fonction de la position du bouton. Toutefois, on vérifie aisément que la différence de longueur est du second ordre du rapport entre la hauteur et le diamètre de la calotte sphérique. On s'est donc permis de ne pas tenir compte de ce terme dans la suite.

1.2.2 Marches

Les boutons doivent nécessairement être posés sur une surface horizontale. La calotte sphérique a donc été approximée par trois couronnes concentriques horizontales séparées par des marches (voir figure 1.2-a).

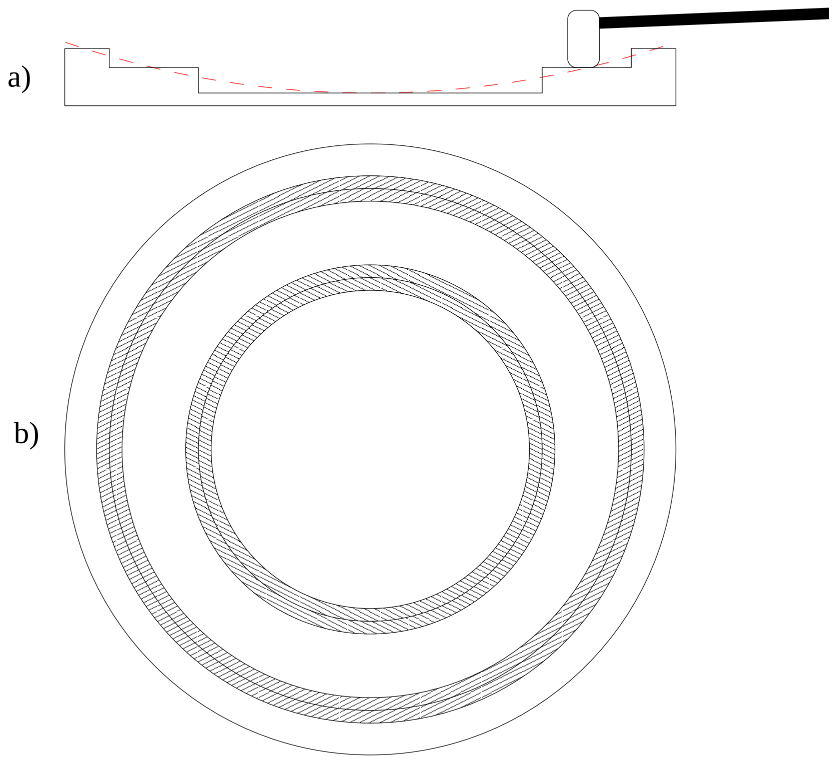


FIG. 1.2: Marches sur la surface focale

Un bouton doit entièrement reposer sur une de ces couronnes. De là, on constate que la zone d'une largeur le rayon du bouton de part et d'autre de chaque marche est inaccessible pour la fibre optique (voir figure1.2-b).

Cette contrainte n'influe guère sur le problème tel qu'il est posé ici. Il suffit de supposer qu'il n'y a aucun objet dans ces zones. Ainsi, la contrainte agit sur les données entrées et non pas sur la manière de résoudre le problème.

REMARQUE : Ce problème des marches implique un autre problème d'optimisation combinatoire couplé à celui de l'assignation. Il s'agit de minimiser le nombre d'étoiles présentes dans ces zones inaccessibles.

1.2.3 Zone d'ombre

En amont de l'instrument, se trouve un bras servant à vérifier et à corriger la visée du télescope. Ce bras a une ombre sur la focale. On a ainsi une autre zone de la focale dans laquelle il ne doit pas y avoir *a priori* d'objets observés.

1.3 Modélisation de la tige

La modélisation de la tige doit faire face à deux exigences. Elle doit être simple pour éviter des temps de calcul trop importants mais suffisamment fine pour ne pas être éloignée de la réalité mécanique. Pour cela, nous avons adopté différents niveaux de modélisation que l'on utilise selon les besoins du moment. Les trois niveaux retenus sont présentés par la figure 1.3.

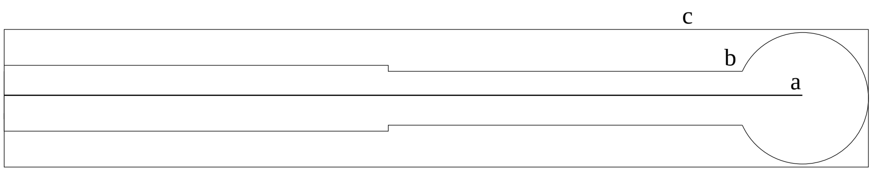


FIG. 1.3: Trois niveaux de modélisation de la tige : (a) le segment entre le centre du bouton et la position de l'axe, (b) son contour, (c) son rectangle d'encombrement.

On a $(a) \subset (b) \subset (c)$. Par conséquent si deux segments représentant deux tiges s'intersectent alors les tiges entrent mécaniquement en collision. Inversement, si deux tiges sont en collision alors leurs rectangles d'encombrement se chevauchent.

Pratiquement, on utilisera donc la modélisation (c) pour s'assurer que deux tiges n'entrent pas en collision et la modélisation (a) pour rechercher des tiges susceptibles de ne pas se gêner.

La modélisation (c) peut être affinée. En effet, on vérifie que deux tiges F_i et F_j n'entrent pas en collision si les quatre propriétés suivantes sont vérifiées :

1. Les deux segments de la modélisation (a) n'ont pas de point d'intersection ;

2. La distance entre la position de la tête de F_i et le segment représentant F_j est supérieure à la somme du rayon du bouton et du diamètre maximal de la tige ;

3. La distance entre la position de la tête de F_j et le segment représentant F_i est supérieure à la somme du rayon du bouton et du diamètre maximal de la tige ;

4. La distance entre les positions des deux têtes est supérieure à deux fois le rayon du bouton.

1.4 Position de la fibre optique

On considère une étoile et la fibre qui lui est associée. Le centre du bouton est posé à la position de l'étoile – dont on connaît les coordonnées. Le but de la section est de calculer la position de l'axe de rotation.

1.4.1 Tige de longueur constante

On considère dans un premier temps que la projection de la tige sur l'horizontale est de longueur constante l_{tige}. L'objet E de coordonnées cartésiennes (x_E, y_E) ou polaires (r_E, θ_E) est associé à la fibre F_i située sur le rayon d'angle $\theta_F = \frac{i}{N} 2\pi$.

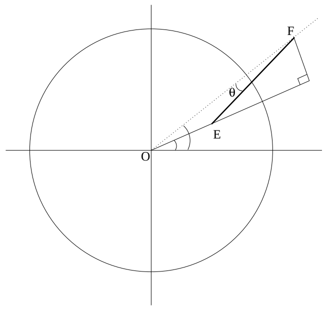


FIG. 1.4: Angles et distances dans une association entre un objet et une fibre

L'angle θ de la fibre par rapport à son axe vérifie l'égalité :

$$l_{tige} \sin \theta = r_E \sin(\theta_E - \theta_F)$$

ce qui permet de le calculer. De là, on calcule les coordonnées de l'axe de rotation de la fibre grâce à l'autre égalité

$$r_F \sin(\theta_E - \theta_F) = l_{tige} \sin(\theta_E - \theta_F + \theta)$$

où r_F représente la distance entre l'axe et le centre du disque focal.

1.5 Définitions et propriétés

Le but de cette section est de définir mathématiquement la modélisation du problème. On s'est attaché à relier des termes usuels à des définitions plus abstraites mais qui suffisent à caractériser les concepts mis en jeu.

Ces définitions et les propriétés – simples – qui en découlent seront essentiellement utilisées dans le chapitre consacré à la recherche d'une solution exacte. Chaque propriété géométrique utilisée dans ce chapitre devra en effet avoir une preuve mathématique pour que la solution trouvée en fin de calcul puisse être déclarée mathématiquement exacte.

1.5.1 Définitions générales

Définition 1 *On appelle* **disque focal** *le disque unité de centre O du plan orienté.*

Définition 2 *Un* **objet** *est un point intérieur au disque focal caractérisé par ses coordonnées cartésiennes* (x, y) *ou polaires* (r, θ).

Définition 3 *Une* **fibre** F *est un heptuplet de réels* $(l, d, b, m, M, \Theta, \alpha)$.

- l *est la* **longueur de la tige**. *On pose* $longueur(F) = l$.
- d *est le* **diamètre de la tige**. *On pose* $diametre(F) = d$.
- b *est le* **diamètre du bouton**. *On pose* $bouton(F) = b$.
- m *et* M *sont les* **début et fin de course du chariot**. *On pose* $min(F) = m$ *et* $max(F) = M$.
- Θ *est l'***angle de débattement de la fibre**. *On pose* $anglemax(F) = \Theta$.
- α *est la* **position angulaire de la fibre**. *On pose* $angle(F) = \alpha$.

Définition 4 (Problème à N fibres identiques) *Pour* l, d, b, m, M *et* Θ *fixés, on note* F_i $(0 \leq i < N)$ *les* N *fibres* $(l, d, b, m, M, \Theta, \frac{2i\pi}{N})$.

On appelle problème à N *fibre identiques, l'ensemble*

$$\mathcal{F} = \{F_0, F_1, \cdots, F_{N-1}\}$$

Définition 5 *On dit que l'objet de coordonnées polaires* E (r_E, θ_E) *est* **assignable** *à la fibre* F $(l_{tige}, d, b, m, M, \Theta, \theta_F)$ *si les réels* r_F *et* θ *définis par les deux équations*

$$\begin{aligned} l_{tige} \sin\theta &= r_E \sin(\theta_E - \theta_F) \\ r_F \sin(\theta_E - \theta_F) &= l_{tige} \sin(\theta_E - \theta_F + \theta) \end{aligned}$$

vérifient

$$m \leq r_F \leq M$$

et

$$|\theta| \leq \Theta$$

*On parle alors de l'***assignation** $\langle F, E\rangle$. *On note* $A_E(F)$ *le point de coordonnées polaires* (r_F, θ_F). *C'est la position de l'axe vertical autour duquel tourne la fibre.*

Définition 6 *Deux assignations* $\langle F, E\rangle$ *et* $\langle F', E'\rangle$ *sont* **compatibles** *si elles vérifient les 4 conditions suivantes :*

1. *Les deux segments* $[E, A_E(F)]$ *et* $[E', A_{E'}(F')]$ *ont une intersection vide.*
2. *La distance entre* E *et le segment* $[E', A_{E'}(F')]$ *est supérieure à* $diametre(F') + bouton(F)$.

3. *La distance entre E' et le segment $[E, A_E(F)]$ est supérieure à $diametre(F) + bouton(F')$.*

4. *La distance entre E et E' est supérieure à $bouton(F) + bouton(F')$.*

On note $\langle F, E\rangle \triangle \langle F', E'\rangle$. Dans le cas contraire, on utilise la notation $\not\triangle$.

REMARQUE : $\triangle$ est symétrique mais n'est ni réflexive ni transitive.

Définition 7 *Un ensemble d'assignations est dit* **compatible** *si les assignations de cet ensemble sont deux à deux compatibles.*

On dit également que $\langle F, E\rangle$ est compatible avec un ensemble A d'assignations si $A \cup \langle F, E\rangle$ est compatible.

La définition du problème d'optimisation est alors claire. Etant donné un problème à N fibres identiques, on recherche un ensemble d'assignations compatibles de cardinal maximal. On définit maintenant plus précisément une classe de problèmes d'optimisation qui nous intéressera plus particulièrement par la suite.

Définition 8 ($\mathcal{ASS}$) *Soit $\mathcal{O}$ un ensemble d'objets dans le disque focal et $\mathcal{F}$ les fibres définies par le problème dit des $|\mathcal{O}|$ fibres identiques. On considère un ensemble A d'assignations composées de fibres de $\mathcal{F}$ et d'objets de $\mathcal{O}$. On suppose de plus que A est compatible. On note*

$$\mathcal{ASS}_A^{\mathcal{O}} = \max\{\, |B| \;/\; A \subset B \text{ et } B \text{ compatible} \,\}$$

Par abus de langage, on parlera de $\mathcal{ASS}_A^{\mathcal{O}}$ comme d'un problème à résoudre et non pas seulement comme de la solution à ce problème.

Définition 9 *Soit $\mathcal{O}$ un ensemble d'objets, $\mathcal{F}$ un ensemble de fibre et A un ensemble d'assignations compatibles d'objets de $\mathcal{O}$ par des fibres de $\mathcal{F}$.*

On appelle **graphe d'accessibilité** *de $\mathcal{ASS}_A^{\mathcal{O}}$ le graphe biparti $(\mathcal{F}, \mathcal{O}, R)$ où R est une relation entre $\mathcal{F}$ et $\mathcal{O}$ définie par :*

$$(f, o) \in R \Leftrightarrow \langle f, o\rangle \text{ compatible avec } A$$

Définition 10 *On appelle* **enveloppe convexe** *d'un ensemble A d'assignation l'enveloppe convexe des points de*

$$\{E; A_E(F) \;/\; \langle E, F\rangle \in A\}$$

1.5.2 Problème à N fibres identiques

On se place désormais dans le cadre de N fibres identiques de paramètres l, d, b, m, M et Θ.

Définition 11 *On dit que l'assignation $\langle F, E\rangle$* **recouvre** *l'objet E' si la distance entre E' est le segment $[E, A_E(F)]$ est inférieure à $d + b$.*

Proposition 1 *Si $\langle F, E\rangle$ recouvre E', pour toute fibre F' on a*

$$\langle F, E\rangle \not\triangle \langle F', E'\rangle$$

PREUVE : C'est une conséquence directe de la définition 6.

Chapitre 2

Approche du problème

2.1 Etudes préalables

L'assignation des fibres optiques a fait l'objet de quelques études depuis l'apparition de la spectroscopie multi-objets. La géométrie des instruments concernés et en particulier la disposition de leurs fibres n'est pas exactement celle de FUEGOS mais les différences sont minimes et les algorithmes proposés peuvent s'adapter très aisément. Il ne s'agit pas non plus à proprement parler d'études d'optimisation combinatoire mais plutôt de simulations de comportements en fonction des caractéristiques des instruments (disposition des fibres, forme des boutons, possibilité ou non du croisement des fibres...)

Pour la mise en œuvre de MEFOS, un positionneur de 29 fibres, un programme d'assignation a été écrit d'après [Kuhn55] et [Tou]. Son algorithme est en fait basé sur l'optimisation d'un problème linéaire simple pour lequel on a relâché une partie des contraintes mécaniques. Ces contraintes sont ré-introduites après l'optimisation pour obtenir une solution viable. Les utilisateurs de ce programme arrivaient régulièrement à améliorer le résultat donné par l'algorithme. Les premiers tests effectués sur une configuration à 100 fibres ont donné des résultats relativement médiocres.

[Don92] s'intéresse à AMOS, instrument multi-fibres monté sur des télescopes américains. L'algorithme trouve une solution grâce à des opérateurs spécialisés qui font évoluer une solution initialement vide en affectant des objets à des fibres ou des fibres à des objets, en violant éventuellement des contraintes pendant quelques étapes de l'evolution.

[Lew93] présente pour AutoFib-2, instrument équivalent monté entre autres sur le télescope Anglo-Américain, un algorithme que nous allons détailler dans la section 2.3. Les performances de ces deux derniers algorithmes sont équivalentes en qualité, mais celui de [Lew93] semble plus rapide.

2.2 Cliques d'un graphe

On aborde dans cette section le problème en le rattachant à un problème plus général, très classique en optimisation combinatoire.

2.2.1 Énoncé

On définit le graphe non orienté $G = (S, A)$. S est l'ensemble des assignations réalisables – S ne contient pas les affectations nulles de la forme $\langle F_i, \emptyset \rangle$ et $\langle \emptyset, O_j \rangle$. Soit deux assignations x et y, le couple (x, y) appartient à A si et seulement si les assignations x et y ne peuvent pas coexister. On rappelle que deux assignations ne peuvent pas coexister si l'une des trois conditions suivantes est remplie :

1. $x = \langle F_i, \alpha \rangle$ et $y = \langle F_i, \beta \rangle$, c'est-à-dire que x et y concernent la même fibre ;

2. $x = \langle \alpha, O_j \rangle$ et $y = \langle \beta, O_j \rangle$, c'est-à-dire que x et y concernent le même objet ;

3. L'association de x et de y viole les contraintes mécaniques.

Il est alors aisé de voir que l'ensemble des cliques[1] de G est l'ensemble des solutions admissibles du problème. Le problème à résoudre revient donc à chercher une clique maximale de G.

2.2.2 Commentaires

1. La recherche d'une clique maximale dans un graphe est un problème $\mathcal{NP}$ complet.

2. Le cardinal de S est majoré par le produit nN. Il est en général de l'ordre du dixième de cette valeur.

3. Les premiers résultats obtenus en suivant cette piste n'ont pas été convaincants ; la voie a été abandonnée.

2.3 Algorithme géométrique glouton

Cet algorithme est inspiré de celui utilisé dans [Lew93]. Il a été adapté pour pouvoir partir de toute assignation partielle, c'est-à-dire pour pouvoir résoudre $\mathcal{ASS}_A^{\mathcal{O}}$ pour tout ensemble compatible A.

Cet algorithme est dit *glouton* car il sélectionne les fibres non affectées les unes après les autres et leur associe un objet une fois pour toute : ce choix n'est jamais remis en question. Comme nous allons le voir tout de suite, ce choix se fait sur des critères essentiellement géométriques.

2.3.1 Algorithme

1. On part d'une situation à N fibres et n objets dans laquelle n_a ($n_a \leq N$ et $n_a \leq n$) assignations sont fixées.

[1] On rappelle qu'une clique d'un graphe $G = (S, A)$ est un sous-ensemble C de S tels que tout couple (x, y) de points de C n'appartient pas à A.

2. On choisit une fibre déjà affectée. S'il n'y en a pas, on en affecte une quelconque à (par exemple) l'objet le plus proche de son axe radial tel que la fibre, une fois assignée à cet objet, ne recouvre pas un autre objet.

3. On fixe un sens de parcours de fibres, par exemple le sens direct.

4. On sélectionne la fibre suivante.

   (a) Si elle est déjà affectée, on la laisse telle qu'elle est.

   (b) Sinon, on lui affecte l'objet accessible qui, vu de la fibre, est le plus proche de la fibre précédente.

   Tant qu'on n'a pas essayé d'assigner toutes les fibres, on recommence cette étape.

Cet algorithme donne une solution qui vérifie toutes les contraintes puisque celles-ci sont satisfaites par chaque fibre au moment de son assignation.

Les étapes 2 et 3 font appel à un choix *a priori* aléatoire. Pour l'instant, on le biaise en prenant par exemple pour fibre de départ celle – libre – qui a le plus petit numéro et le sens direct comme sens de rotation. Nous verrons plus tard les conséquences d'un tel choix.

L'étape 4 est effectuée n fois. Son temps d'exécution est majoré par la sous-étape (b) que nous allons étudier tout de suite.

2.3.2 Implantation

Comme on vient de le voir, la principale difficulté d'implantation de l'algorithme ci-dessus est la recherche de l'objet qui doit être affecté à la fibre sélectionnée. Supposons que les fibres soient assignées selon le sens direct. On doit donc affecter à cette dernière l'objet accessible qui se trouve le plus à gauche pour un observateur situé sur le bord du champ focal, à la position de la fibre. La figure 2.1 représente la situation.

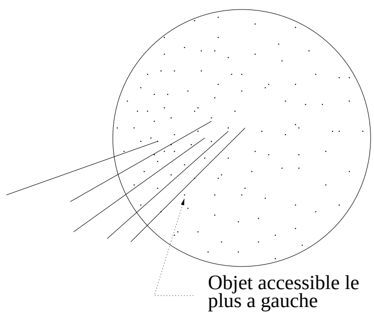


FIG. 2.1: Recherche de l'objet accessible le plus à gauche. Les segments dans la partie inférieure gauche représentent les fibres déjà affectées.

L'idée est de maintenir une liste triée des objets visibles[2] de gauche à droite lorsqu'un observateur parcourt de manière *continue* le bord du disque focal. En effet, lors de ce parcours, il arrive que deux objets s'intervertissent dans cet ordre : le plus éloigné est d'abord à gauche puis les deux points sont confondus lorsque le point d'observation se trouve sur la droite passant par ces deux points et enfin c'est le point le plus proche qui se retrouve à gauche.

Deux faits sont à remarquer :

1. Les positions relatives de deux points s'inversent au moment où l'observateur traverse la droite support des deux points.

2. Juste avant de s'intervertir, les deux points sont consécutifs dans la liste.

De ces deux remarques découle une manière de trouver rapidement, à chaque étape de l'algorithme présenté dans la section précédente, l'objet qui doit être affecté à la fibre sélectionnée. Cette méthode s'inspire d'idées couramment utilisées en géométrie algorithmique et notamment présentées par [Boi95].

Tout au long du déroulement de l'algorithme, on maintient deux listes [3] : celle – ordonnée – des objets accessibles de gauche à droite et celle des événements à venir – classés dans leur ordre d'occurence. On entend par événement :

1. **l'arrivée dans un champ d'un objet** : cela correspond à la position de l'observateur pour laquelle l'objet se trouve pour la première fois dans la zone d'accessibilité[4]. Ce nouvel objet entre dans la liste comme objet le plus à droite. Voir figure 2.2-gauche.

2. **l'inversion de l'ordre de deux objets** : cela correspond à l'alignement de la position de l'observateur et de deux objets. Il faut donc inverser ces deux objets. Voir figure 2.2-droite.

3. **la sortie d'un champ d'un objet** : cela correspond à la position de l'observateur pour laquelle l'objet se trouve pour la dernière fois dans la zone d'accessibilité. L'objet ne doit plus être pris en compte dans la liste.

4. **la rencontre d'une fibre** : lorsque l'observateur passe à la position d'une fibre, on en profite pour assigner cette dernière à l'objet accessible le plus à gauche. Il faut alors remettre à jour la liste des objets, l'accessibilité de certains n'étant plus assurée à cause de cette nouvelle assignation.

Un événement peut être introduit dans la liste soit lors de l'initialisation des données soit lors de la modification de la liste des objets accessibles.

[2] Il existe un parallèle évident entre les objets accessibles pour les fibres et ceux qui seraient visibles pour un observateur dont le champ de vision serait confondu avec le champ d'accessibilité de la fibre. Les fibres mises en place sont alors considérées comme des obstacles. L'algorithme que nous allons présenter s'inspire d'algorithmes utilisés dans la vision, c'est pourquoi nous utiliserons indifféremment les mots *accessible* et *visible*.

[3] Le terme de *liste* est en fait générique, il est par exemple plus habile d'utiliser des arbres rouge-noir ou autre structure plus évoluée.

[4] Cette zone d'accessibilité est en fait virtuelle ; elle correspond à celle d'une fibre dont l'axe passerait par la position de l'observateur.

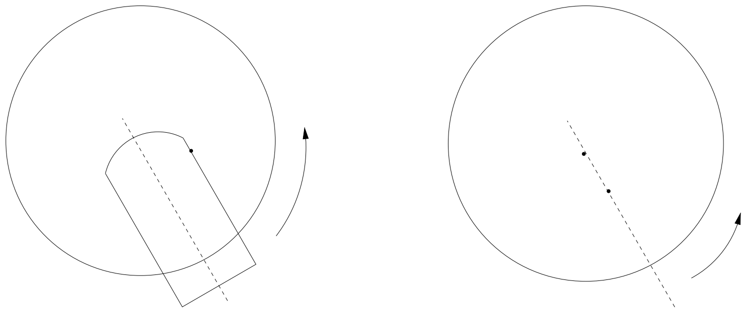


FIG. 2.2: Cette figure illustre les deux événements les plus fréquents. A gauche, l'objet entre dans la zone d'accessibilité de la fibre, représentée par le rectangle terminé par un arc de cercle. A droite l'ordre des objets s'inverse : avant le passage par cette position, l'objet central était à gauche de l'objet le plus extrieur, après il est à droite.

Lors de l'initialisation, on place l'observateur en la position de la fibre définie dans l'étape 2 de l'algorithme. Il est alors possible de classer les objets accessibles de gauche à droite. Pour ces objets, on peut calculer le moment où ils sortiront du champ d'accessibilité (événement de type 3), pour tous les autres objets, on peut calculer le moment où ils entreront dans ce champ (événements de type 1). Pour tout objet accessible, de celui le plus à gauche à celui le plus à droite, on calcule le moment où il inversera sa position avec son voisin de droite (événement de type 2). Les rencontres des fibres (événements de type 4) sont également connues *a priori* lors de l'initialisation.

Lors du calcul, des événements de type 2 sont à introduire lorsqu'un nouvel objet rentre dans le champ ou lorsque deux objets de ce champ sont introduits. La disparition d'un objet du champ d'accessibilité peut rendre certains événements caduques : ils ne produiront alors aucune réaction.

En résumé, une fois les paramètres initialisés l'algorithme consiste à parcourir le bord du disque focal et à réagir aux événements dans l'ordre où ils sont rencontrés. L'avantage de procéder ainsi est que la réaction à un événement donné se fait en temps constant. En revanche, le nombre d'événements générés est assez difficile à évaluer.

2.3.3 Résultats

Cet algorithme a été implanté en C. Il a été testé avec un nombre de fibres N égal à 100. Avec une telle valeur, le temps de calcul est très court, inférieur à la seconde. Il a donc été possible de faire une étude statistique du nombre d'assignations trouvées sur un large échantillon de données.

Dans un premier temps, nous avons regardé son comportement en fonction du nombre d'objets présents dans le champ focal. Les résultats sont représentés par la figure 2.3. Lorsqu'il y a peu d'objets, l'algorithme n'a pas de problème à trouver une fibre pour chacun. Inversement, dès qu'il y a un grand nombre d'objets, il n'a aucun problème à trouver un objet pour chaque fibre.

Le cas intéressant se produit donc quand le nombre de fibres et le nombre d'objets sont du même ordre de grandeur. C'est d'ailleurs un cas fréquent dans la pratique. Dans la suite, on se contentera donc généralement d'échantillons pour lesquels $n = N$. Si un algorithme donne de bons résultats pour un tel cas, il est raisonnable de penser qu'il est aussi efficace pour les autres valeurs de n qui correspondent à des cas *a priori* plus faciles.

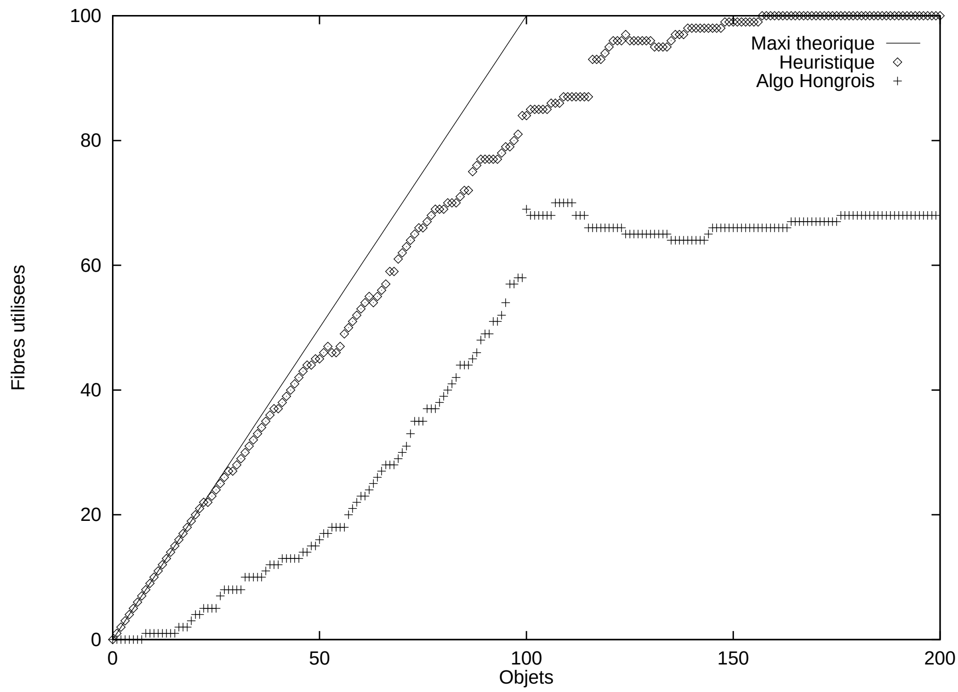


FIG. 2.3: Ce diagramme montre les résultats de tests de l'heuristique. En abscisse est indiqué le nombre d'objets dans le champ focal et en ordonnée le nombre de fibres attribuées. La ligne continue montre la borne supérieure de la solution. Les résultats obtenus sont indiqués par les losanges. A titre indicatif, les résultats obtenus à l'aide de l'algorithme hongrois ont également été portés sur le diagramme.

Ces premiers tests montrent l'inefficacité totale de l'algorithme hongrois pour des problèmes de taille importante. En effet, avec un nombre de fibres plus grand et des fibres plus resserrées, le risque de collisions et d'intersections grandit et le nombre d'assignations éliminées en fin de course devient trop important. Par la suite, nous ne nous intéresserons donc plus à cet algorithme.

La figure 2.4 compare la variation des résultats en fonction de l'échantillon. Tous les échantillons sont des répartitions uniformes de 100 points dans le disque focal. L'écart-type des résultats obtenus est relativement faible puisqu'il est de l'ordre de quelques assignations.

Nous avons voulu également voir la performance de l'algorithme sur des échantillons non-uniformes. Nous en avons créé de 100 objets concentrés au

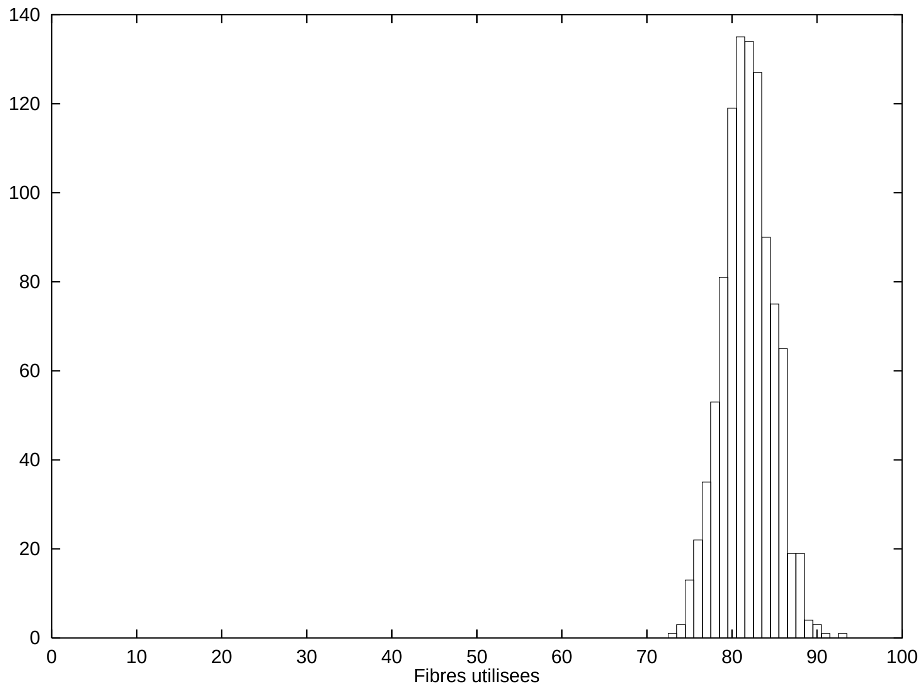


FIG. 2.4: 1000 exécutions ont été effectuées avec à chaque fois 100 objets tirés aléatoirement (pour 100 fibres). Le diagramme comptabilise le nombre de solutions trouvées utilisant x fibres.

centre du disque. Ils sont sensés simuler des amas de galaxies. Les résultats obtenus sont donnés par la figure 2.5. L'écart-type est un peu plus grand et la performance nettement plus faible. C'est principalement dû au fait que les étoiles sont beaucoup plus proches les unes des autres. En raison des contraintes mécaniques, dès qu'une fibre est posée, plusieurs étoiles se retrouvent inaccessibles car situées trop proche de la fibre.

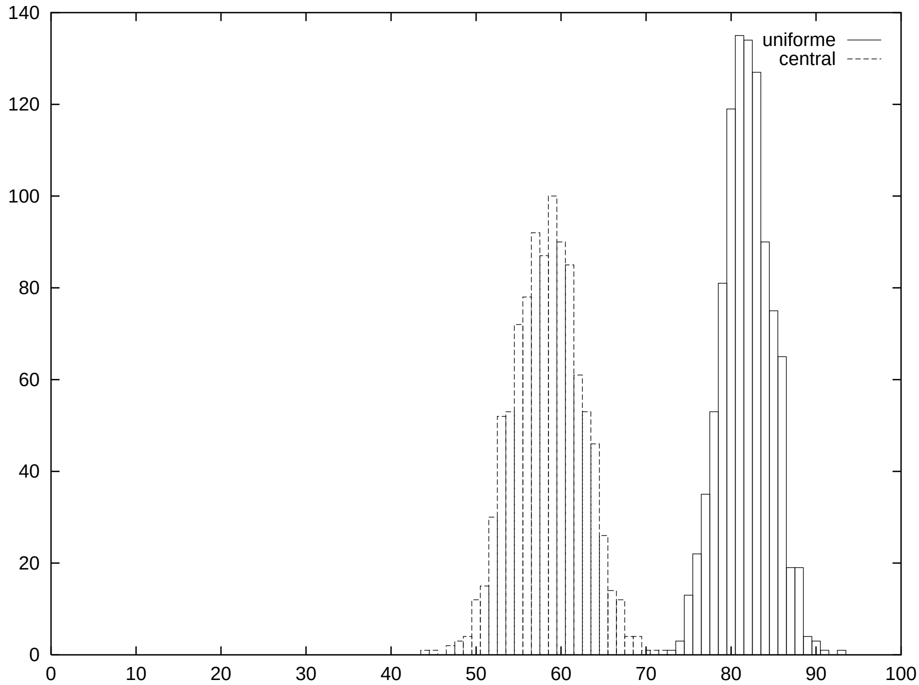


FIG. 2.5: Comparaison de 1000 exécutions de l'algorithme sur (a) une répartition uniforme d'objets, (b) une répartition d'objets centrés au centre de la focale

Chapitre 3

Amélioration des heuristiques

L'algorithme exposé dans la section 2.3 sera désormais appelé *heuristique de niveau 0* et noté H_0. Il sera en effet réutilisé par d'autres algorithmes heuristiques étudiés dans ce chapitre qui seront, eux, notés H_n avec $n \geq 0$.

Nous allons en fait définir par récurrence la suite d'algorithmes H_n. La première section s'attachera à détailler la première étape de la récurrence. Le reste sera fait dans la section 3.2.

Par la suite, les résultats de ces heuristiques seront analysés pour mettre au point un algorithme efficace. Celui-ci se basera essentiellement sur un travail de post-optimisation réutilisant les résultats mis en avant par la section 3.2. La fin du chapitre sera consacrée aux résultats obtenus.

3.1 Heuristique de niveau 1

3.1.1 Description

Rappelons l'idée de base de l'heuristique. A chaque étape, on a un ensemble A d'assignations déjà faites et on veut affecter à la fibre F_i un des objets de $O_{acc}^{(i,A)}$, ensemble des objets qui sont accessibles par F_i sans qu'il y ait de collision avec des assignations de A :

$$O_{acc}^{(i,A)} = \{o \in \mathcal{O} \text{ tel que } A \cup \{\langle F_i, o\rangle\} \text{ soit compatible}\}$$

Parmi ces objets, on sélectionne celui qui minimise l'angle que doit avoir F_i pour l'atteindre.

L'idée est de perfectionner la manière dont on sélectionne l'objet dans $O_{acc}^{(i,A)}$. Pour cela, notons H_A^0 le nombre d'affectations que peut effectuer l'heuristique de niveau 0 lorsque les affectations de A sont déjà faites – c'est en fait la solution de $\mathcal{ASS}_A^{\mathcal{O}}$ donnée par H_0. L'heuristique de niveau 1 H_1 est alors définie ainsi :

1. Sélection de F_i comme fibre courante ($F_i \leftarrow F_{i+1}$ ou F_1 au départ).

2. Calcul de $O_{acc}^{(i,A)}$.

3. Sélection de O dans $O_{acc}^{(i,A)} \cup \{\emptyset\}$ qui maximise $H^0_{A\cup\{\langle F_i,O\rangle\}}$.

4. $A \leftarrow A \cup \{\langle F_i, O\rangle\}$.
5. Tant qu'il reste des fibres non affectées, reprendre à l'étape 1.

3.1.2 Résultats

La figure 3.1 compare les performances des heuristiques de niveau 0 et 1. L'heuristique de niveau 1 donne des résultats supérieurs ou égaux à ceux de l'algorithme de base et le surpasse pour un nombre d'objets compris entre 50 et 150, ce qui correspond aux cas où l'on est le plus loin du maximum théorique.

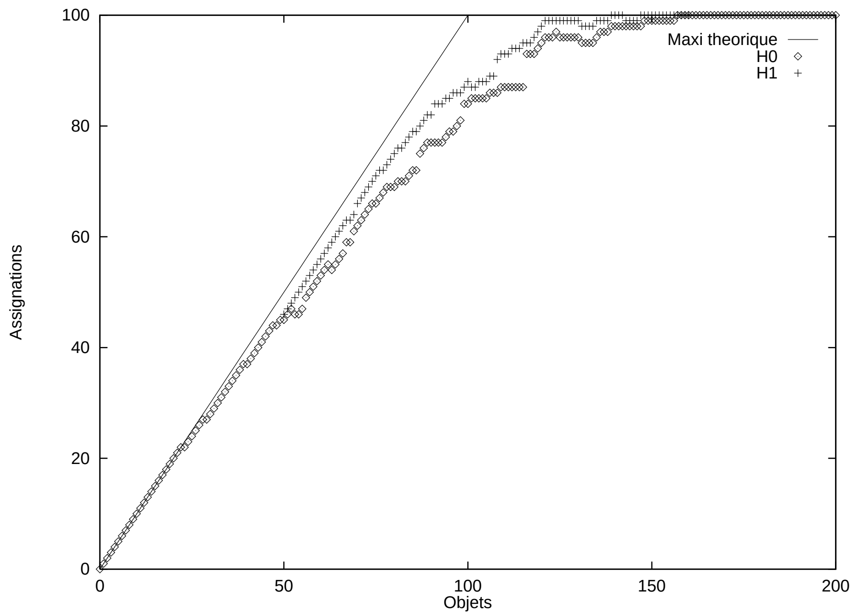


FIG. 3.1: Comparaison des heuristiques de niveau 0 et de niveau 1. En ordonnée est représenté le nombre d'assignations effectuées selon le nombre d'objets en abscisse.

Les temps de calcul sont relativement réduits, de l'ordre de la minute, ce qui convient parfaitement aux besoins des utilisateurs.

3.2 Heuristiques de niveau n

L'amélioration significative des résultats qu'apporte H_1 nous a poussé à continuer sur cette voie et à généraliser le concept.

3.2.1 Description

On considère qu'on a un algorithme H_n qui admet en entrée

- un ensemble $\mathcal{F}$ de fibres ;
- un ensemble $\mathcal{O}$ d'objets ;

– un sous-ensemble $\{F_1 \cdots F_i\}$ de fibres dont les éléments sont assignés à des éléments de $\mathcal{O}$

et qui calcule l'assignation de $F_{i+1} \cdots F_N$. On définit par récurrence l'algorithme H_{n+1} qui a exactement les mêmes caractéristiques d'entrée et de sortie.

Notons comme nous l'avons fait pour l'heuristique de niveau 0 H_A^n le nombre d'affectations que peut effectuer l'heuristique de niveau n lorsque les affectations de A sont déjà faites. L'heuristique de niveau $n+1$ H_{n+1} est alors définie ainsi :

1. Sélection de F_i comme fibre courante ($F_i \leftarrow F_{i+1}$ ou F_1 au départ).
2. Calcul de $O_{acc}^{(i,A)}$.
3. Sélection de O dans $O_{acc}^{(i,A)} \cup \{\emptyset\}$ qui maximise $H^n_{A\cup\{\langle F_i,O\rangle\}}$.
4. $A \leftarrow A \cup \{\langle F_i, O\rangle\}$.
5. Tant qu'il reste des fibres non affectées, reprendre à l'étape 1.

Cet algorithme s'implante facilement en faisant appel à la récursivité.

3.2.2 Résultats théoriques

Performance

Lorsque H_n recherche l'objet qui convient le mieux à F_1, l'algorithme lance pour chaque objet accessible H_{n-1} avec comme paramètre d'entrée supplémentaire F_1 assignée à cet objet. Parmi ces objets, il y a nécessairement celui qu'aurait choisi H_{n-1} pour F_1. On en conclut que toutes les solutions explorées par H_n le sont par H_{n-1} d'où le théorème :

THÉORÈME 1 *La suite des résultats trouvés par $(H_n)_{n\in N}$ est croissante.*

Cette suite est croissante et bornée, elle a donc une limite. De plus, plus n est grand, plus l'ensemble des solutions explorées est grand. On peut donc penser que lorsque n tend vers l'infini H_n trouve la solution optimale. On a même un résultat plus fort :

THÉORÈME 2 *H_N donne le résultat optimal.*

Preuve : H_N est chargé de choisir l'objet assigné à F_1 parmi tous les objets accessibles. Pour cela, l'algorithme fait appel à H_{N-1} pour l'attribution de l'objet assigné à F_2 pour cela H_{N-1} regarde tous les objets accessibles par F_2. Par induction, on en déduit que pour tout $i < N$, H_i explore tous les objets accessibles par F_{1+i}. Par conséquent l'ensemble de toutes les assignations possibles est exploré, la solution optimale est donc trouvée.

Comme nous allons le voir tout de suite, ce résultat n'a pas de conséquence pratique : pour un nombre de fibres usuel, il est totalement inenvisageable de se lancer dans le calcul de H_N...

Temps de calcul

On appellera ici h le niveau de l'heuristique afin de ne pas le confondre avec n, le nombre d'objets présents dans le disque focal. On rappelle en même temps que N désigne le nombre de fibres. On définit également :

- t_h : le temps d'exécution de H_h ;
- $t_{h,i}$: le temps d'exécution du choix de l'affectation de F_i par H_h ;
- $T^{(2)}_{h,i}$: le temps d'exécution de l'étape 2 lors de l'affectation de F_i par H_h.

On a évidemment la relation

$$t_h = \sum_{i=1}^{N} t_{h,i}$$

tandis que l'étude de la relation de récurrence définissant l'algorithme donne immédiatement, puisque $|O^{(i,A)}_{acc}| \leq n$:

$$t_{h+1,i} = \mathcal{O}\left(n t_{h,i+1} + T^{(2)}_{h+1,i}\right)$$

et donc

$$t_{h+1} = \mathcal{O}\left(n t_h + \sum_{i=1}^{N} T^{(2)}_{h+1,i}\right)$$

L'étape 2 doit vérifier pour au plus n affectations possibles la non-intersection avec les $i-1$ fibres déjà affectées. On en déduit que

$$\sum_{i=1}^{N} T^{(2)}_{h+1,i} = \mathcal{O}\left(nN^2\right)$$

ce qui permet de conclure

$$t_{h+1} = \mathcal{O}\left(n(t_h + N^2)\right) = \mathcal{O}\left(n(n^h t_0 + hN^2)\right) = \mathcal{O}\left(n^{h+1} N^2\right)$$

soit encore

$$t_h = \mathcal{O}\left(n^h N^2\right)$$

3.2.3 Diminution du temps de calcul

Comme nous venons de le voir, les temps de calcul augmentent énormément avec le niveau de l'heuristique. Pour pouvoir mettre en œuvre les algorithmes – ne serait-ce que H_2 – il convient d'utiliser toutes les propriétés capables de faire diminuer le temps de calcul.

Restriction du nombre d'objets accessibles

Lorsqu'on regarde le déroulement de H_1 ou de H_2, on remarque que, pour chaque fibre, l'objet sélectionné est généralement celui qu'aurait sélectionné H_0 c'est-à-dire celui qui fait faire l'angle le plus petit à la fibre. Dans les autres cas, c'est pratiquement toujours le second objet le plus à gauche ou l'assignation vide qui sont choisis.

En remplaçant $O_{acc}^{(i,A)}$ par l'ensemble de ses deux éléments les plus à gauche pour la fibre, on ne change en général pas la qualité de la solution trouvée et on diminue grandement le temps de calcul. En effet, en refaisant le raisonnement sur le temps de calcul, on s'aperçoit que, dans les majorations, on peut remplacer n par 3 – nombre d'assignations désormais possibles – ce qui donne une borne asymptotique indépendante du nombre d'objets soit $\mathcal{O}(N^2)$. Pratiquement, on peut se rendre compte que le temps de calcul est diminué par 2.

Majoration par le nombre de fibres disponibles

A chaque étape, et quel que soit le niveau de l'heuristique, la somme du nombre n_a d'assignations déjà faites et du nombre n_f de fibres encore disponibles majore la meilleure solution qu'on pourra obtenir. Ainsi, dès qu'on est sûr de ne pas pouvoir dépasser le meilleur résultat possible, il est souhaitable de faire la coupe de cette branche d'exploration.

Il y a toutefois un piège à éviter : si H_n trouve une solution meilleure au moment où elle se trouve sur la fibre F_i, il faudra que H_{n+1} puisse retrouver cette même solution quand elle passera sur cette fibre F_i. En ne procédant à la coupe que lorsque $n_a + n_f$ est *strictement* inférieur à la valeur de la meilleure solution, on satisfait à cette exigence.

Mémorisation de résultats intermédiaires

On suppose que l'algorithme H_n est en train de traiter la fibre F_i. Il peut lui affecter deux objets a et b ou bien encore la laisser libre. Pour faire son choix, il calcule :

- $H^{n-1}_{A\cup\{\langle F_i,a\rangle\}}$: pour cela, il faut calculer $H^{n-1}_{A\cup\{\langle F_i,a\rangle\}\cup B_j^a}$ pour les différents B_j^a qui sont les sous-ensembles croissants représentant les assignations des fibres F_{i+1} jusqu'à F_N.
- $H^{n-1}_{A\cup\{\langle F_i,b\rangle\}}$ et $H^{n-1}_{A\cup\{\langle F_i,\emptyset\rangle\}}$ dont le calcul induit les mêmes autres calculs intermédiaires que ci-dessus avec les ensembles pareillement définis B_j^b et $B_j^\emptyset$.

Soit $o \in \{a, b, \emptyset\}$ le choix de H_n. Quand il passe à la fibre F_{i+1}, H_n a le choix entre plusieurs objets dont nécessairement o' défini ainsi : $\langle F_{i+1}, o'\rangle$ est la première assignation des ensembles croissants B_j^o. Pour tout niveau d'heuristique $n \geq 1$, la mémorisation des résultats intermédiaires qui ont amené au choix de l'assignation de F_i permet de ne pas refaire ces mêmes calculs au moment de l'assignation de F_{i+1}.

Pratiquement, on utilise un tableau à trois dimensions : une pour les fibres, une pour les objets qu'elle peut atteindre et une pour le niveau de l'heuristique.

L'implantation du tableau de mémorisation des résultats intermédiaires réduit le temps de calcul d'environ $\frac{1}{3}$.

Incidence sur la qualité du résultat

La première amélioration à H_n restreint l'ensemble des solutions explorées. Dans certains cas rares, il se peut que l'algorithme modifié trouve une solution moins bonne que H_n. En revanche, les deux autres accélérateurs de calculs ne peuvent mathématiquement pas changer la qualité de la solution.

3.2.4 Résultats calculatoires

La figure 3.2 compare les performances des heuristiques de niveau 1 et 2. H_2 se révèle plus performant dès qu'il y a plus de 90 objets dans le disque focal. Elle permet de gagner jusqu'à 4 assignations. De plus elle permet une utilisation des 100 fibres dès qu'il y a 120 objets contre 140 environ pour H_1.

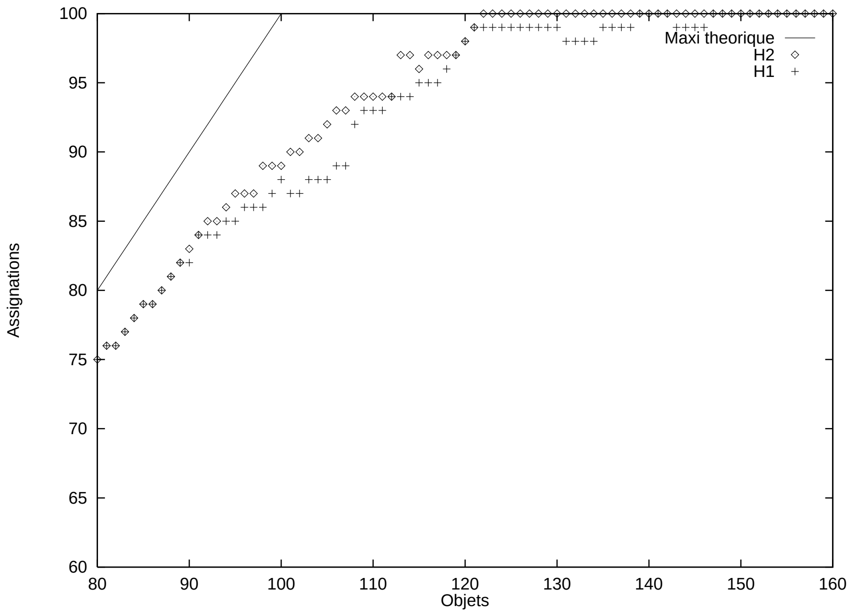


FIG. 3.2: Comparaison des heuristiques de niveau 1 et de niveau 2. En ordonnée est représenté le nombre d'assignations effectuées selon le nombre d'objets en abscisse.

Le temps de calcul est d'environ une quarantaine de minutes sur une machine peu occupée. L'assignation étant faite bien avant l'observation, un tel temps est tolérable pour les utilisateurs. Mais, d'un point de vue pratique, il n'est pas envisageable de faire tourner H_3 pour résoudre quotidiennement le problème de l'assignation. De manière raisonnable, on peut se fixer un temps de calcul inférieur au temps de pose.

3.2.5 Généralisation à d'autres problèmes

Ce type d'heuristique peut être adapté à tous les problèmes d'optimisation combinatoire dont une solution peut être construite par un algorithme *glouton* – qu'on appelera A_0 – qui choisit *au mieux*, à chaque étape de la construction de la solution, un élément dans un ensemble E. Dans le problème d'assignation des fibres optiques, cet ensemble est $\mathcal{O}$ (puisque pour chaque fibre on sélectionne un objet dans $\mathcal{O}$) et l'algorithme qui permet de choisir l'élément au mieux est bien sûr H_0.

Comme cela a été fait dans les deux premières sections de ce chapitre, il est possible de définir une suite d'algorithmes A_n permettant d'obtenir des solutions meilleures que celles construites par la première heuristique. On définit ainsi A_n de manière récursive :

1. On calcule le sous-ensemble F de E des éléments pouvant être sélectionnés *a priori*.

2. On choisit dans F celui qui donne le meilleur résultat quand on termine le calcul avec l'heuristique A_{n-1}.

3. Tant qu'on peut rajouter des éléments de E, on recommence à l'étape 1.

D'un point de vue pratique, il est bien entendu nécessaire que l'heuristique de base soit rapide, *ie* polynomiale de faible degré.

La qualité de A_n dépend bien sûr de la qualité de A_0. Ainsi, si A_0 donne choisit aléatoirement un élément de F, A_n est un exploration aléatoire – plus ou moins complète en fonction de n – de l'arbre qui donne toutes les solutions viables du problème.

Le voyageur de commerce

L'un des problèmes les plus connus d'optimisation combinatoire est celui du voyageur de commerce (*Traveling Salesman Problem* - TSP). Il est donc intéressant de pouvoir tester les performances de ces heuristiques sur quelques instances de ce problème pour lesquelles on connaît une très bonne solution – voire une solution optimale[1].

On s'est intéressé ici au problème dit *euclidien* dont nous rappelons brièvement l'énoncé. Étant donné un ensemble de points dans le plan, il s'agit de trouver un chemin passant une fois et une seule par tous ces points et revenant au point de départ de longueur minimale. Il existe pour ce problème de nombreux algorithmes heuristiques qui permettent de trouver des bonnes solutions au problème.

On a pris pour A_0 l'algorithme d'*insertion du plus proche voisin*. On part d'un point arbitraire, appelé point courant. Ensuite, tant qu'il reste des points qui n'ont pas été sélectionnés, on sélectionne celui qui est le plus proche du point courant. Il devient à son tour le point courant.

---

[1] Disponibles au `http ://www.iwr.uni-heidelberg.de/iwr/comopt/soft/TSPLIB95/tsp/`

L'ensemble E et l'ensemble des points du plan. Le sous-ensemble F est l'ensemble des points qui n'ont pas été sélectionnés. En suivant la méthode générale indiquée ci-dessus, la suite d'algorithme A_n a été implantée. La figure 3.3 montre une solution trouvée par A_4.

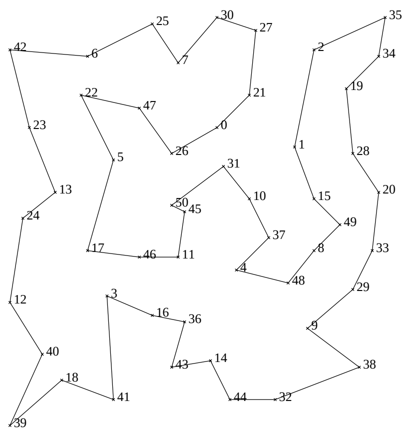


FIG. 3.3: Exemple d'une tournée obtenue à l'aide d'une heuristique de niveau 4 légèrement améliorée. En quelques minutes, on obtient une tournée de longueur $448,7$. La solution optimale pour ce problème de 51 villes est de longueur 426, soit une différence de $4,5\%$.

REMARQUES :

- Pour diminuer le temps de calcul, il est avantageux de limiter F aux 10 ou 15 points les plus proches du point courant.
- Les résultats varient de manière non négligeable en fonction du point de départ. La position de ce point – au centre du nuage ou sur le bord – ne semble pas être significative.
- Les résultats obtenus (figure 3.3) sont encourageants. L'adaptation du travail de post-optimisation tel qu'il sera développé dans la suite de ce chapitre pour le problème d'assignation peut se réveller très fructueuse pour le voyageur de commerce.
- En faisant varier le point de départ, on peu obtenir de cette manière un grand nombre de bonnes solutions différentes. Cela peut se réveller utile pour certaines métaheuristiques (algorithmes génétiques ou *Ant-system*).

Vision duale du problème

Un même problème peut avoir différents algorithmes de construction de bonnes solutions susceptibles d'être améliorés par la méthode exposée ci-dessus. Par exemple, dans le problème qui nous intéresse, on peut imposer de visiter les points dans un certain ordre (par exemple du plus extérieur au plus central) et pour chaque point rencontré on lui assigne la fibre disponible la plus proche

– en veillant bien entendu à ce que les contraintes mécaniques ne soient pas violées.

3.3 Heuristiques dans l'ordre d'assignation des fibres

Le principe de base de l'heuristique développée jusqu'ici est de parcourir l'ensemble des fibres et, pour chacune des fibres rencontrées, de lui affecter l'objet qui semble convenir le mieux. Jusqu'à présent, le parcours de l'ensemble des fibres se faisait en partant d'une fibre quelconque (la fibre 0) puis en prenant celles rencontrées en tournant autour du disque focal dans un sens fixé (le sens direct).

Le but de cette section est d'étudier la façon dont varie la performance de l'heuristique quand on modifie cet ordre de sélection des fibres.

3.3.1 Fibre de départ

Pour mettre en avant l'importance de cette fibre de départ, nous avons comparé, pour un champ d'objets O fixé, la variation du nombre d'assignations obtenues en fonction de la fibre de départ mais aussi du niveau de l'heuristique. Les résultats sont présentés dans la figure 3.4.

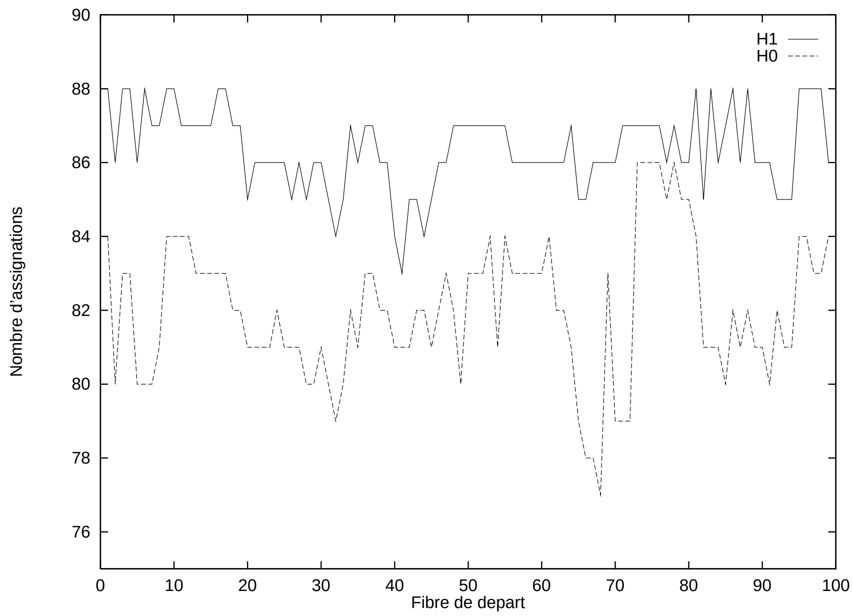


FIG. 3.4: Variation du nombre d'assignations trouvées selon le choix de la fibre de départ. La courbe du dessus correspond à l'heuristique de niveau 1, celle du dessous à l'heuristique de niveau 0.

L'heuristique de niveau 1 est moins sensible au choix de la fibre de départ que celle de niveau 0. L'écart entre le résultat le meilleur et le pire est de 5 contre 8 pour H_0 et la courbe semble moins irrégulière à l'œil.

Toutefois, plusieurs points sont inquiétants :

- Le résultat obtenu, en partant d'une fibre ou bien de sa voisine peut varier de 4 ou 5 affectations, c'est-à-dire de l'ordre de l'amélioration qu'apporte le passage à une heuristique de niveau supérieure.

- Les meilleurs résultats de H_0 sont meilleurs que les plus mauvais de H_1. Ainsi, un algorithme d'une seconde peut donner des meilleurs résultats qu'un algorithme d'une minute...

- Il n'existe pas d'intervalle de 4 ou 5 fibres consécutives qui donnent le même résultat.

- Les courbes correspondant à H_0 et H_1 ne sont pas parallèles, les meilleurs résultats de H_0 ne correspondent pas à ceux de H_1. Il n'est donc pas envisageable d'étudier la courbe correspondant à H_0 avant de lancer H_1 ou H_2.

- Si on compare ces courbes avec la courbe donnant pour chaque fibre le nombre d'objets qui lui sont *a priori* accessibles, on constate qu'il y a plus de chance d'obtenir un bon résultat en partant d'une fibre ayant peu d'objets accessibles mais il est difficile d'en dire plus. En effet, si on part de la fibre ayant le moins d'objets accessibles, le résultat obtenu est médiocre (85 assignations par H_1).

3.3.2 Sens de parcours

Nous avons comparé, pour les deux premiers niveaux d'heuristiques, les variations du résultat obtenu en sélectionnant les fibres selon un parcours dans le sens direct et indirect. Les résultats sont représentés sur les deux diagrammes de la figure 3.5.

On retrouve bien entendu la variation de la performance en fonction de la fibre de départ mais on remarque que globalement un sens de parcours est globalement meilleur que l'autre et ceci aussi bien pour H_0 que pour H_1. Cet écart est important, supérieur en moyenne à 3 assignations pour H_0 et à 2 pour H_1.

Le sens et l'ampleur de cet écart est bien entendu lié à la disposition des objets (et donc à l'énoncé du problème). Le même test a été effectué avec d'autres problèmes. Il est courant de retrouver des écarts aussi importants – surtout pour H_0 – mais il arrive également que cet écart ne soit pas significatif.

En vue d'obtenir un algorithme efficace, on s'est davantage occupé à faire disparaître ce problème plutôt qu'à exploiter la différence de performance qui semble trop peu fiable.

3.3.3 Parcours en *double-sens*

Afin de faire disparaître la différence de qualité de résultat entre le parcours des fibres en sens direct et celui en sens indirect, nous avons fait un mariage de ces deux méthodes. Les fibres affectées forment toujours un intervalle à chaque étape de l'algorithme mais, au lieu de faire croître cet intervalle par une seule

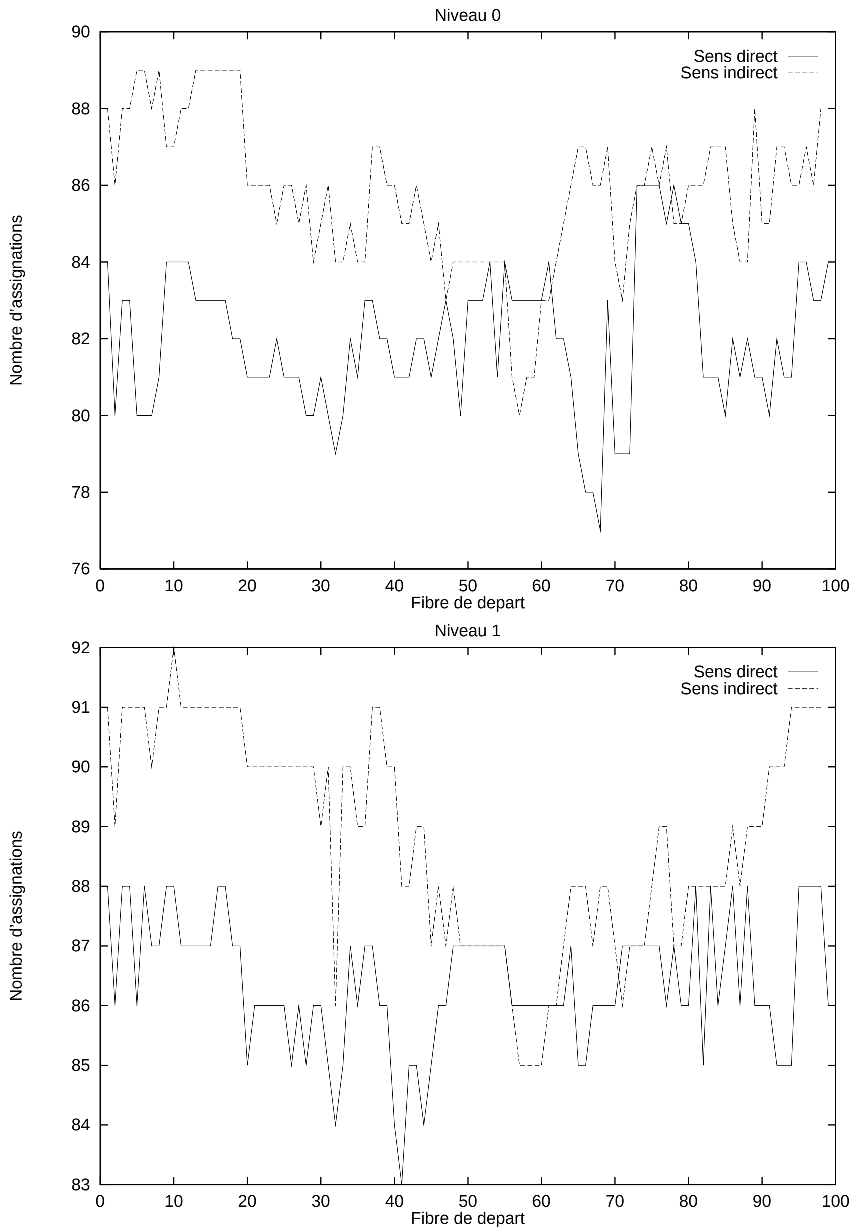


FIG. 3.5: Variation du nombre d'assignations trouvées selon le choix de la fibre de départ et selon l'ordre d'affectation des fibres. Le diagramme du dessus correspond à H_0, celui du dessous à H_1.

extrémité, nous l'avons fait croître par les deux. Ainsi si les fibres affectées formaient l'intervalle $[F_i \cdots F_j]$, on affecte F_{j+1} puis F_{i-1}, et ainsi de suite.

Les résultats obtenus se situent à peu près à mi-chemin entre ceux obtenus par le parcours en sens direct et celui en sens indirect. Cette voie ne semble donc pas améliorer la qualité des résultats, nous ne l'avons pas plus approfondie.

3.4 Changement du niveau d'heuristique

Lorsqu'on regarde la fibre choisie à chaque étape de l'algorithme, on remarque qu'il s'agit dans une grande majorité des cas – supérieure à 90% – de celle qu'aurait choisi l'heuristique de niveau 0. Les heuristiques de niveau supérieur semblent donc essentiellement utiles à résoudre des problèmes locaux, c'est-à-dire à un petit nombre d'objets et de fibres voisins pour lesquels l'algorithme d'assignation de niveau 0 n'est pas efficace.

Nous avons donc fait la modification suivante à l'algorithme H_{n+1}. Lorsqu'il doit choisir l'objet affecté à F_i , il fait appel à H_n pour les fibres $F_{i+1}, F_{i+2} \cdots F_{i+\eta}$, à H_{n-1} pour les fibres $F_{i+\eta+1}, F_{i+\eta+2} \cdots F_{i+2\eta}$ et ainsi de suite jusqu'à ce qu'on arrive à H_0.

Sur les cas testés, pour des valeurs de η supérieures à la dizaine, les performaces de H_2 n'ont pas été amoindries par cette modification de l'algorithme. Les temps de calculs ont été notablement raccourcis. Cela a ainsi permis de lancer une exécution de H_3 et d'obtenir une assignation de plus que H_2 en partant de la même fibre.

3.5 Post-optimisation

Comme cela a été mis en évidence dans la section 3.3, la manière dont est parcouru l'ensemble des fibres influe grandement sur la qualité du résultat obtenu. La position des objets dans le plan focal ne semble pas permettre de mettre *a priori* en avant une bonne fibre de départ et un bon sens de parcours.

En vue d'amoindrir les effets du choix de la fibre de départ et du sens de parcours, la solution obtenue par H_n est remaniée dans ce qu'il convient d'appeler un travail de post-optimisation. Celui-ci s'est attaché à :

- réutiliser les heuristiques H_n ;
- gommer l'influence du point de départ en ré-attaquant le problème en différents points de départ ;
- gommer l'influence du sens de parcours en changeant successivement le sens de parcours des fibres.

3.5.1 Principe

On commence par affecter toutes les fibres par un algorithme H_n – en pratique n vaut 1 ou même 0, à partir de 2, les temps de calculs sont beaucoup trop long – en partant d'une fibre quelconque et en choisissant aléatoirement le sens de parcours.

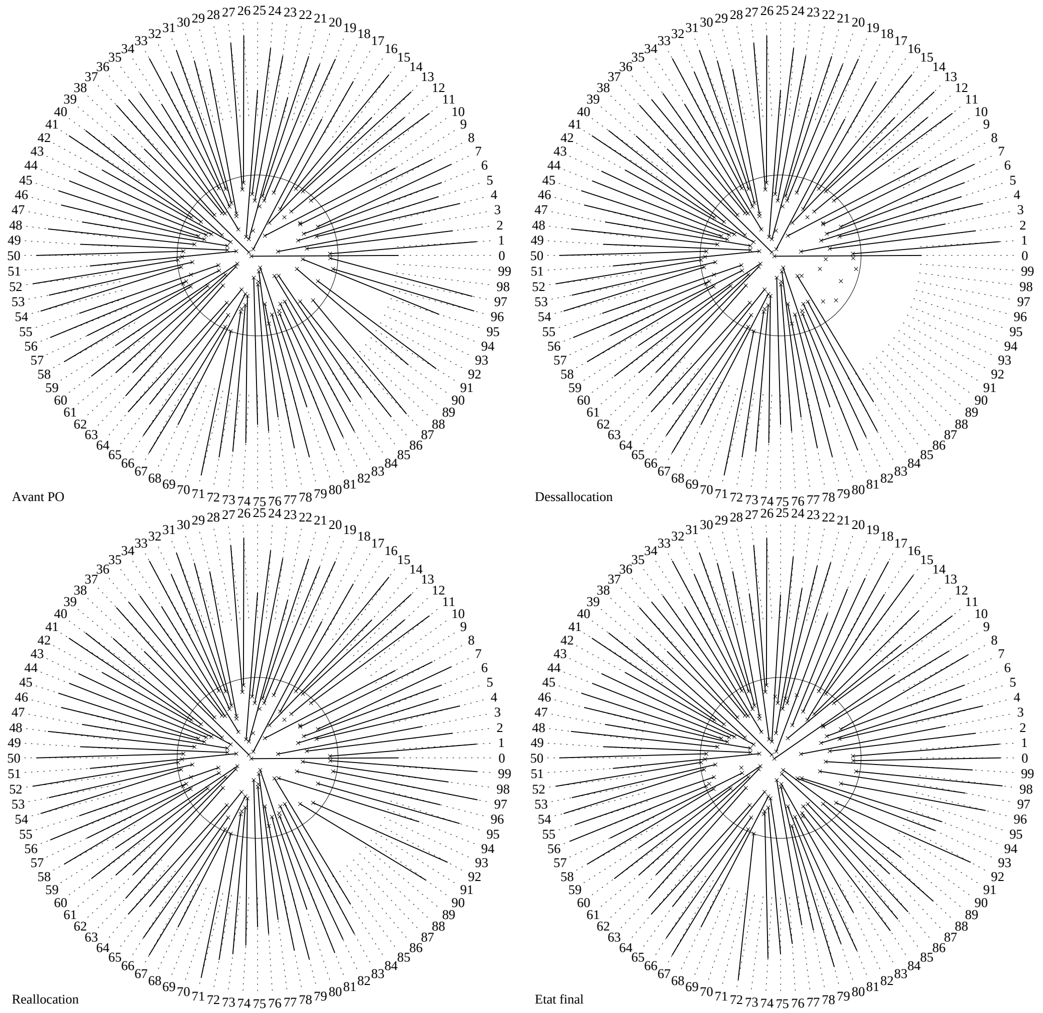


FIG. 3.6: En haut à gauche, 88 fibres ont été allouées par H_1 en partant de la fibre 0 et en tournant dans le sens direct. En repartant de la fibre 0 mais en tournant dans l'autre sens, 15 fibres sont désallouées à partir de la première fibre rencontrée non allouée (la fibre 99). En bas à gauche, les fibres ont été de nouveau allouées, mais par H_2 et en progressant dans le sens indirect. Il y a toujours 88 assignations mais les fibres 89 à 85 ont été libérées et elles pourront être réutilisées par la suite. Lorsqu'on a itéré les opérations de désallocation et réallocation sur tout le tour, on obtient 90 affectations – en bas à droite.

Une phase de post-optimisation, illustrée par la figure 3.6, se déroule alors ainsi :

1. On change de sens de rotation et on choisit un nouveau point de départ
2. On avance dans le sens de rotation tant qu'on ne rencontre pas de fibre non allouée. La phase de post-optimisation s'arrête quand on repasse par le point de départ.
3. Quand on rencontre une fibre non allouée, on applique l'opération suivante :
   - On désalloue les d fibres suivantes ;
   - On les ré-alloue avec H_m $(m > n)$;
   - Si la solution obtenue contient au moins autant d'assignations qu'avant l'opération, on prend cette solution comme solution courante sinon on reprend la précédente.
   - On reprend le parcours des fibres de l'étape 2.

Le parcours en sens inverse et la désallocation à partir d'une fibre non allouée est sensé attribuer des objets aux fibres qui se trouvaient "en manque d'objets". Si on aborde ensuite une région où il y a plus d'objets que de fibres, le fait d'avoir déjà ces objets à des fibres précédemment libres permet de créer un meilleur équilibre entre les nombres d'objets et de fibres ce qui peut permettre de gagner une ou deux affectations.

Il est bien entendu possible d'itérer plusieurs fois cette phase de post-optimisation.

3.5.2 Notation normalisée

On appelle dans la suite

- *PO* les algorithmes pour lesquels $n = 1$ et $m = 2$
- *po* les algorithmes pour lesquels $n = 0$ et $m = 1$

On indique ensuite en indice le nombre d de fibres désallouées lors de chaque ré-allocation et en exposant le nombre de phases de post-optimisation. Ainsi, par exemple, l'algorithme PO^2_{15} alloue dans un premier temps toutes les fibres avec H_1 puis effectue 2 phases de post-optimisation avec H_2 en désallouant 15 fibres à chaque fois.

3.6 Astuces heuristiques

3.6.1 Objet caché par une tige

Lorsqu'on procède à une assignation $\langle F_i, O_j \rangle$, il arrive de temps en temps (voir figure 3.7) qu'un autre objet O_k se trouve *sous* la tige. Cet objet est alors définitivement perdu en ce sens qu'il ne peut participer à aucune assignation

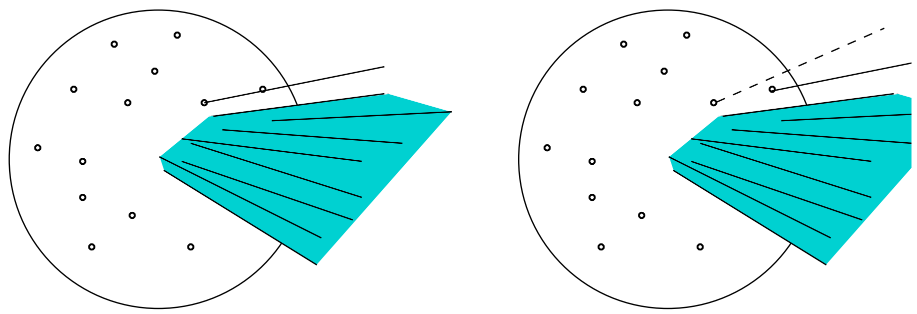


FIG. 3.7: A gauche, un objet qui n'est pas le plus à gauche pour la fibre est caché par celle-ci quand elle est affectée à l'objet plus à gauche mais plus éloigné. La configuration de droite montre qu'il est possible d'observer simultanément les deux objets.

à venir. En revanche, si on affecte la fibre F_i à l'objet O_k, l'objet O_j est bien souvent affecté à une autre fibre.

Lorsque H_1 (ou H_n pour $n \geq 1$) hésite entre O_j et O_k pour l'assignation de F_i, on constate qu'il sélectionne généralement O_k. En forçant la sélection de O_k dans les cas définis dans le paragraphe précédent, on peut gagner pratiquement un niveau d'heuristique sans augmenter significativement le temps de calcul.

3.7 Résultats

Tous les résultats de cette section sont basés sur le problème à 100 fibres identiques.

3.7.1 Présentation des échantillons

Nous avons créé 4 problèmes de 100 objets répartis aléatoirement sur le disque focal et 1 problème avec un nuage de 100 objets au centre du champ focal (`cl100`). Nous avons d'autre part choisi 5 algorithmes. H_1 peut être vu comme algorithme de référence, les quatre autres permettent de comparer les différents paramétrages de la post-optimisation :

- PO^1_{15} ré-affecte à chaque fois un grand nombre de fibres mais sur un seul tour ;
- PO^{10}_{10} fait appel à d'avantage de post-optimisation mais sur des intervalles plus petits ;
- po^{50}_{10} fait appel à des heuristiques de niveau inférieur aux deux précédentes, ce qui permet de créer plus de solutions diversifiées mais *a priori* moins bonnes ;
- po^{50}_{15} fait davantage de travail de post-optimisation que l'algorithme précédent, ce qui demande bien sûr plus de temps de calcul.

En vue de comparer ces différentes stratégies, nous avons lancé chaque algorithme avec chaque problème en partant successivement de chaque fibre. Nous avons ainsi pour chaque sens de parcours (*dir* ou *rev*) un échantillon de 100 résultats. Ces échantillons sont présentés dans le tableau 3.1 qui donne une première idée des performances de chaque algorithme.

		H_1		PO_{15}^{1}		PO_{10}^{10}		po_{10}^{50}		po_{15}^{50}	
		dir	rev	dir	rev	dir	rev	dir	rev	dir	rev
st100	Max	88	92	92	92	92	92	92	92	92	92
	Moy	86.4	88.9	90.3	90.2	90.8	91.0	89.7	90.6	90.1	91.2
	Var	1.08	1.79	0.64	1.13	0.76	0.68	1.22	1.09	1.20	0.86
et100	Max	86	85	87	88	87	88	88	87	88	88
	Moy	85.1	82.7	85.8	86.5	86.7	85.6	86.3	86.0	87.0	87.5
	Var	0.58	1.51	0.72	0.68	0.44	0.97	0.89	0.68	0.30	0.52
ob100	Max	90	90	90	91	90	92	91	92	92	93
	Moy	87.9	86.7	88.8	89.4	89.9	90.8	89.7	90.5	90.0	91.16
	Var	0.89	1.07	0.79	0.79	0.36	0.76	0.52	0.66	0.65	0.82
as100	Max	89	88	91	90	91	91	91	91	91	91
	Moy	87.4	87.5	89.0	88.7	90.0	89.2	90.4	89.5	90.5	90.0
	Var	0.68	0.64	0.63	0.68	0.74	0.66	0.87	0.70	0.64	0.58
cl100	Max	66	65	66	67						
	Moy	60.2	59.7	62.8	63.0						
	Var	2.23	1.50	1.63	2.06						

TAB. 3.1: Comparaison des résultats obtenus par les 5 algorithmes H_1, PO_{15}^{1}, PO_{10}^{10}, po_{10}^{50} et po_{15}^{50} sur 5 problèmes de 100 objets.

On remarque essentiellement l'influence de la post-optimisation en comparant les résultats de PO_{15}^{1}, PO_{10}^{10} et po_{10}^{50} par rapport à ceux de H_1. La diminution de la variance montre que l'influence du choix de la fibre de départ est amoindrie. En ce qui concerne le choix du sens de parcours, on remarque qu'il n'est plus significatif. Dans la suite, on associera donc, pour chaque couple algorithme-problème, les deux échantillons *dir* et *rev* en un seul.

Les notions de moyenne et de variance ne suffisent pas pour comparer les performances des algorithmes. C'est pourquoi la section suivante s'attache à définir le concept d'*espérance de résultat*.

3.7.2 Espérance de résultat

On considère un algorithme heuristique de maximisation $\mathcal{A}$ qui a N initialisations possibles. Lorsqu'on lance $\mathcal{A}$ avec ces N initialisations différentes, l'ensemble des résultats obtenus est supposé composé de K valeurs différentes notées :

$$v_1 > v_2 > \cdots > v_{K-1} > v_K$$

Chacune des valeurs v_i $(1 \leq i \leq K)$ est obtenue n_i fois. On a naturellement

$$N = \sum_{i=1}^{K} n_i$$

On suppose qu'on lance $\mathcal{A}$ l fois avec à chaque fois des initialisations différentes. On note X_l la variable aléatoire correspondant au meilleur résultat obtenu après ces l tirages. On se propose de calculer l'espérance $E(X_l)$ de cette variable aléatoire.

On note $\mathcal{P}(X_l = n)$ la probabilité pour X_l de valoir n. On a alors la formule :

$$E(X_l) = \sum_{i=1}^{K} \mathcal{P}\left(X_l = v_i\right) v_i$$

On a bien évidemment

$$\mathcal{P}\left(X_l = v_i\right) = \mathcal{P}\left(X_l \leq v_i\right) - \mathcal{P}\left(X < v_i\right)$$

Comme de plus

$$\mathcal{P}\left(X_l < v_K\right) = 0$$

et

$$\forall 1 \leq i < K \quad \mathcal{P}\left(X_l < v_i\right) = \mathcal{P}\left(X_l \leq v_{i+1}\right)$$

il suffit de connaître $\mathcal{P}\left(X_l \leq v_i\right)$ pour tout i compris entre 1 et K pour calculer $E(X_l)$. $\mathcal{P}\left(X_l \leq v_i\right)$ est la probabilité que toutes les initialisations donnent un résultat compris entre v_i et v_K. On considère l initialisations – supposées distinctes – correspondant à l exécutions de $\mathcal{A}$. On les appelle une l-initialisation. On peut voir une l-initialisation comme un sous-ensemble de l éléments de l'ensemble des N initialisations possibles. Une l-initialisation conduisant à $X_l \leq v_i$ est alors un sous-ensemble de l'ensemble des initialisations conduisant à une valeur comprise entre v_i et v_K. Cet ensemble a pour cardinal $\sum_{j=i}^{K} n_j$. On a donc :

$$\mathcal{P}\left(X_l \leq v_i\right) = \frac{\begin{pmatrix} l \\ s_i \end{pmatrix}}{\begin{pmatrix} l \\ N \end{pmatrix}} = \frac{s_i!}{N!} \frac{(N-l)!}{(s_i - l)!} = \prod_{j=1}^{l} \frac{s_i - l + j}{N - l + j}$$

avec

$$s_i = \sum_{j=i}^{K} n_j$$

On calcule successivement $\mathcal{P}\left(X_l \leq v_K\right)$, $\mathcal{P}\left(X_l \leq v_{K-1}\right)$ $\cdots$ $\mathcal{P}\left(X_l \leq v_2\right)$ et $\mathcal{P}\left(X_l \leq v_1\right)$ – qui vaut 1 – pour obtenir $E(X_l)$.

Remarque : Lorsque $l = 1$, on retrouve pour $E(X_1)$ la formule de la moyenne. L'espérance de résultat après une exécution est égale à la moyenne de l'échantillon.

3.7.3 Espérance de résultat en un temps fixé

Pour comparer la performance des différents algorithmes, on calcule l'espérance de résultat en un temps CPU donné. Pour une durée T fixée, on calcule le nombre n d'exécutions pouvant être effectuées et on calcule comme il est indiqué dans la section précédente l'espérance de résultat après ces n exécutions. Les résultats obtenus sont résumés dans le tableau 3.2.

		H_1		PO^1_{15}		PO^{10}_{10}		po^{50}_{10}		po^{50}_{15}	
Tps 1 run $_s$		46.0		86.6		229		149		702	
Tps CPU $_{min}$		10	20	10	20	10	20	10	20	10	20
Nbre runs		13	26	6	13	2	5	4	8	1	2
st100	Max	92		92		92		92		92	
	Esp	90.9	91.1	91.4	91.7	91.3	91.7	91.3	91.6	90.7	91.3
et100	Max	86		88		88		88		88	
	Esp	85.8	86.0	87.0	87.2	86.7	87.0	87.0	87.3	87.2	87.5
ob100	Max	90		91		92		92		93	
	Esp	89.4	89.8	90.1	90.4	90.7	91.2	90.8	91.0	90.5	91.1
as100	Max	89		91		91		91		91	
	Esp	88.1	88.1	89.7	90.0	90.1	90.6	90.8	91.0	90.2	90.6
Tps 1 run $_s$		46.0		1060		-		-		-	
Nbre runs		13	26	0	1	-	-	-	-	-	-
cl100	Max	66		67		-		-		-	
	Esp	63.4	64.2	-	62.9		-	-	-	-	-

TAB. 3.2: Espérance de résultat après 10 et 20 minutes de calcul

Champs uniformes

Pour les champs ayant une répartition uniforme d'objets, la post-optimisation améliore nettement la performance des algorithmes H_n en permettant de trouver rapidement de très bonnes solutions. Le réglage des paramètres de cette post-optimisation semble avoir peu de répercutions sur les résultats. Il est par exemple difficile de décider s'il vaut mieux lancer plusieurs calculs courts ou seulement un ou deux longs. Si l'on tient à passer plus de temps en calcul pour obtenir une meilleure solution, on a tout intérêt à en augmenter la durée en prenant des paramètres plus grand. Ainsi, po^{50}_{15} a été la seule à trouver 93 assignations pour le problème ob100.

Les bons résultats des algorithmes de type *po* ont pu faire penser qu'il suffisait de remuer localement les solutions obtenues pour avoir de bons résultats. Nous avons alors testé po^{100}_{5} pour faire un grand nombre de modifications locales. Les résultats très moyens (voir tableau 3.3) nous ont convaincus de la qualité d'une approche *à large échelle* du travail d'optimisation et de post-optimisation.

st100	dir	rev
Maximum	92	91
Moyenne	88.5	88.5
Variance	1.44	1.41
Esp (10 min)	89.82	
Esp (20 min)	90.31	

TAB. 3.3: Performances de po_5^{100}

Nuage central

La longueur du travail de post-optimisation dépend grandement du nombre de fibres non affectées. Pour les champ uniformes, il y a à chaque fois une dizaine de fibres libres et on peut considérer que les temps de calcul diffèrent relativement peu. Pour un nuage d'objets très concentré tel que `cl100`, il y a 4 fois plus de fibres non affectées lorsqu'on lance la post-optimisation. Les temps de calculs s'en trouvent grandement rallongés. Par exemple, on constate que PO_{15}^1 est 12 fois plus long. En un temps donné, on fait un nombre de *runs* nettement moindre.

Comme c'est généralement le cas pour les champs uniformes, la post-optimisation permet de trouver un meilleur résultat que celui donné par H_1. En revanche, PO_{15}^1 ne peut-être lancé qu'une seule fois en 20 minutes. L'espérance de résultat se trouve donc être égale à la moyenne de l'échantillon, qui est nettement moins bonne que l'espérance de H_1 après seulement quelques minutes de calcul. Les autres algorithmes faisant appel à la post-optimisation ne pouvaient se terminer dans le temps imparti.

On se trouve donc dans un cas où il est préférable d'alléger le travail de post-optimisation. Par exemple, lorsque plusieurs fibres consécutives ne sont pas affectées, il est certainement aussi efficace de faire le travail d'allocation-désallocation une seule fois, pour la première fibre rencontrée.

Chapitre 4

Solution exacte

Comme nous allons le voir dans ce chapitre, le fait de posséder une solution de bonne qualité[1] à un problème d'optimisation combinatoire est d'une grande aide pour calculer la (une) solution exacte du problème. Nous allons donc utiliser les résultats obtenus par les heuristiques développées jusqu'à présent pour calculer la solution optimale d'un problème.

4.1 Principes généraux

Les problèmes d'optimisation combinatoire consistent à trouver une solution extrémale – maximale dans le cas qui nous intéresse – d'une fonction objectif f satisfaisant un ensemble de contraintes $\mathcal{C}$. On appelle $\mathcal{S}$ l'ensemble des solutions de $\mathcal{C}$. Un élément de $\mathcal{S}$ est appelé *solution valide, viable ou réalisable*. D'un point de vue pratique, une telle solution est réalisable mais pas nécessairement optimale – loin s'en faut souvent. Avec ces notations, le problème peut se formuler ainsi :

$$\max_{s \in \mathcal{S}} f(s)$$

avec

$$\mathcal{S} = \{s \text{ satisfaisant } \mathcal{C}\}$$

Dans de tels problèmes, les difficultés sont de plusieurs ordres' :

- Il peut être difficile de trouver des éléments de $\mathcal{S}$ voire de les caractériser ;

- L'ensemble $\mathcal{S}$, bien que généralement fini, peut avoir un très grand nombre d'éléments parmi lesquels il est difficile de déterminer l'optimal.

Lorsqu'on recherche une solution optimale à un problème d'optimisation combinatoire, on emploie couramment une des deux méthodes suivantes :

- L'**énumération** consiste à évaluer f pour tous les éléments de $\mathcal{S}$. On ne peut utiliser cette méthode que lorsque $|\mathcal{S}|$ est relativement petit. Cela correspond en général à des problèmes où la difficulté est justement de trouver $\mathcal{S}$.

[1] C'est-à-dire satisfaisant toutes les contraintes sans être tout à fait optimale

- Le **branch-and-bound** consiste à partitionner l'ensemble $\mathcal{S}$ en sous-ensembles solution de sous-problèmes. Chaque sous-problème est résolu directement, décomposé en d'autres sous-problèmes ou alors rejeté si on arrive à montrer qu'il ne peut pas contenir de solution optimale du problème initial. C'est évidemment grâce à ce dernier cas qu'on peut obtenir des calculs rapides – ou du moins réalisables par une machine.

Un problème de 100 fibres pour un nombre comparable d'objet peut être considéré comme un problème de combinatoire de grande taille. En effet, on peut estimer qu'alors chaque fibre peut *a priori* accéder à une dizaine d'objets, ce qui fait 100 variables entières pouvant prendre 10 valeurs différentes ce qui fait de l'ordre de 10^{100} configurations à énumérer. Sachant que l'univers n'est agé que d'environ 10^{10} années, il faudrait tester 10^{90} configurations par an soit plus de 10^{82} par seconde pour pouvoir résoudre ce problème dans le temps qui nous sépare de la création de l'univers.

La méthode d'énumération primaire n'est donc pas viable. Il faut tout de même remarquer que parmi toutes les configurations envisagées dans le paragraphe précédent seul un "petit" nombre sont réalisables, c'est-à-dire satisfont $\mathcal{C}$. Il convient donc de mettre en œuvre une méthode d'énumération des seules configurations réalisables qui s'inscrit dans l'algorithme de *branch-and-bound*.

4.2 Mise en place de l'algorithme de *branch-and-bound*

Dans cette section, nous allons d'abord nous intéresser à la partie "*branch*" de l'algorithme. Nous allons en effet présenter une partition du problème en plusieurs niveaux de sous-problèmes permettant à la fois d'énumérer toutes les solutions réalisables du problème et d'implanter dans de bonnes conditions l'algorithme de *branch-and-bound*.

La section 4.2.2 présentera les traits généraux du "*bound*" qui seront détaillés dans les sections suivantes.

4.2.1 Partition du problème en sous-problèmes

Nous allons définir par récurrence les sous-problèmes de $\mathcal{ASS}_{\emptyset}^{\mathcal{O}}$.

$\mathcal{S}$ a été défini plus haut comme l'ensemble des solutions réalisable d'un problème d'optimisation combinatoire. Comme il n'y a aucun risque de confusion entre les deux notions, nous allons désormais confondre le problème et l'ensemble $\mathcal{S}$. En effet, résoudre le problème revient à trouver un maximum de $\mathcal{S}$; on peut donc bien voir l'ensemble $\mathcal{S}$ comme l'énoncé du problème.

Diviser un problème en sous-problèmes revient donc à réaliser une partition de l'ensemble $\mathcal{S}$.

Nous allons considérer le problème $\mathcal{ASS}_A^{\mathcal{O}}$ où A est un ensemble d'assignations compatibles et la fibre F_i dont on suppose qu'elle n'intervient dans aucune affectation de A. On note $O_{acc}^{(i,A)}$ l'ensemble des objets accessibles par F_i :

$$O_{acc}^{(i,A)} = \{o \in \mathcal{O} \text{ tel que } A \cup \{\langle F_i, o\rangle\} \text{ soit compatible}\}$$

$$\{\mathcal{ASS}^{\mathcal{O}}_{A\cup\{\langle F_i,o\rangle\}} \text{ tel que } o \in O^{(i,A)}_{acc} \cup \emptyset\}$$

est une partition en sous-problèmes de $\mathcal{ASS}^{\mathcal{O}}_A$. En effet, deux solutions viables de deux sous-problèmes différents diffèrent au moins par l'affectation de la fibre F_i. Réciproquement, pour toute solution viable s de $\mathcal{ASS}^{\mathcal{O}}_A$, on appelle o l'objet affecté à la fibre F_i – cela peut éventuellement être $\emptyset$. L'assignation $\langle F_i, o\rangle$ est nécessairement compatible avec A donc o appartient à $O^{(i,A)}_{acc}$. Par construction, s est compatible avec $A \cup \{\langle F_i, o\rangle\}$; c'est donc une solution viable d'un des sous-problèmes de $\mathcal{ASS}^{\mathcal{O}}_A$.

Pour toute permutation σ de $[1, N]$, on peut définir par induction un arbre dont la racine est $A = \emptyset$ et dont les nœuds sont des ensembles d'assignations défini selon la règle suivante :

$$A \text{ a pour fils } A \cup \{\langle F_\sigma(i), o\rangle\} \text{ si } i = |A| < N \text{ et } o \in O^{(\sigma(i),A)}_{acc}$$

Il est aisé de mettre en bijection les sous-problèmes avec les sous-arbres. On constate en particulier que les feuilles de l'arbre sont les solutions réalisables de $\mathcal{ASS}^{\mathcal{O}}_A$. Un algorithme d'exploration de l'arbre permet donc d'énumérer toutes les solutions réalisables du problème.

4.2.2 Améliorations nécessaires

L'algorithme tel qu'il est présenté ci-dessus permet une énumération non redondante de l'ensemble des solutions réalisables $\mathcal{S}$. Dans le cas général, le cardinal de $\mathcal{S}$ est trop important pour envisager une recherche par énumération de la solution optimale du problème.

Il est donc nécessaire de mettre en évidence des types de sous-problèmes qui ne peuvent mathématiquement pas conduire à une solution optimale ou, si on se contente de trouver une seule configuration optimale, à une solution meilleure que la meilleure solution connue. Formellement, si le problème P est partitionné en $P_1, P_2 \cdots P_n$, on peut se passer de résoudre P_i si on arrive à démontrer qu'*a priori*

$$\max_{x\in P_i} f(x) \leq \text{ (ou <) } \max_{x\in P} f(x)$$

La fonction objectif f évoquée précédemment dans un cadre général est, dans le problème que nous traitons, le nombre d'affectations réalisées.

On ne peut bien évidemment pas se lancer dans le calcul exact des deux membres de cette inégalité. On est donc amené à trouver un majorant M_{P_i} de $\max_{x\in P_i}$ et un minorant m_P de $\max_{x\in P} f(x)$ en espérant obtenir que

$$M_{P_i} \leq \text{ (ou <) } m_P$$

Le calcul de m_P a fait l'objet des chapitres précédents. En effet, toute solution valide permet de connaître un minorant de $\max_{x\in P} f(x)$. Le fait de posséder des heuristiques de bonne qualité est donc précieux pour la recherche d'une solution exacte d'un problème d'optimisation combinatoire. Cela permet en effet d'avoir une meilleure valeure pour m_P et donc d'éliminer plus de sous-problèmes. Comme le temps de calcul croît exponentiellement avec le nombre de sous-problèmes étudiés, on voit que le gain d'une unité sur m_P est d'une grande importance pour le temps de calcul global. Bien entendu, si, au cours de l'exploration de l'arbre, on découvre un meilleure solution que celle trouvée par les heuristiques, on remplace m_P par cette nouvelle valeur.

Le calcul de M_{P_i} peut se faire de différentes manières. La **relaxation lagrangienne** est une méthode très courante en optimisation combinatoire. En programmation entière (IP), elle consiste à relâcher la condition obligeant une ou plusieurs variables à être entière. Il peut être ainsi possible d'obtenir une solution exacte au nouveau problème qui majorera la solution optimale de P_i. On peut également, de manière générale, supprimer toute contrainte pour avoir un majorant de P_i. Cela n'a d'intérêt que si le nouveau calcul donne rapidement un résultat de qualité – *ie* un M_{P_i} proche de $\max_{x \in P_i} f(x)$. C'est bien là toute la difficulté de la tâche...

Dans le problème qui nous intéresse, ce sont les contraintes de non-collision entre les fibres qui ont le poids le plus important. En supprimer une partie ne semble mener à aucun résultat. Si on en enlève quelques-unes le problème garde une très forte complexité, si on n'en garde que quelques-unes, on aboutit très vite à un résultat trivial : toutes les fibres sont assignables, ce qui n'a aucun intérêt puisque cette valeur est nécessairement supérieure à m_P.

Pour obtenir des majorants de $\max_{x \in P_i} f(x)$, nous nous sommes donc attachés avant tout à la nature géométrique du problème. Dans la section 4.3, nous étudions comment majorer en général la solution d'un sous-problème. Dans la section 4.4, nous nous penchons sur le cas suivant : si $P_1, P_2 \cdots P_i$ est une partition de P, nous mettrons en évidence des cas tels qu'on ait l'assurance que $\max_{x \in P_k} f(x) \leq \max_{x \in P_l} f(x)$, ce qui permet de ne pas étudier P_k si on a précédemment étudié P_l.

4.3 Borne supérieure d'un sous-problème

On essaie de majorer *a priori* le nombre d'assignations possibles d'un sous-problème P. On suppose par conséquent qu'un certain nombre d'affectations sont déjà faites. Dans un premier temps, on ne fait aucune supposition sur les positions des fibres affectées.

4.3.1 Nature géométrique des majorations

On appelle n_a le nombre d'assignations déjà faites. On désire majorer le nombre d'assignations encore possibles en tenant compte des contraintes. Une assignation faisant intervenir un objet et une fibre, nous avons naturellement les deux majorants suivants pour le nombre d'affectations encore possibles :

1. le nombre n_F de fibres pouvant observer au moins un objet,

2. le nombre n_O d'objets pouvant être observés par au moins une fibre.

Le nombre total d'assignations possibles est donc majoré par

$$n_a + \min(n_F, n_O)$$

Il est toutefois possible d'affiner cette majoration dans des cas semblables à celui présenté par la figure 4.1. En effet, les deux fibres déjà affectées ont une implicaton assez forte : les objets à l'intérieur de la portion du disque focal ne peuvent être associés qu'à des fibres situées à l'intérieur de ce même espace angulaire. Ainsi, le graphe biparti entre l'ensemble O des objets et l'ensemble F des fibres représentant les assignations possibles est divisé en deux composantes connexes.

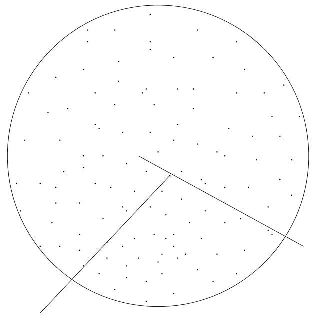


FIG. 4.1: Les deux fibres représentées, affectées à deux objets, divise le graphe biparti des assignations possibles en deux composantes connexes.

Il est ainsi légitime de parler de la partition C de l'ensemble des assignations possibles par l'ensemble des composantes connexes. Pour chacune des classes c ainsi formées, il est possible de majorer le nombre d'assignations possibles comme ci-dessus par :

1. le nombre n_F^c de fibres pouvant observer au moins un objet,

2. le nombre n_O^c d'objets pouvant être observés par au moins une fibre.

On arrive ainsi à la conclusion que le nombre total d'assignations possibles est majoré par

$$n_a + \sum_{c \in C} \min(n_F^c, n_O^c)$$

Comme on a

$$n_F = \sum_{c \in C} n_F^c$$

et

$$n_O = \sum_{c \in C} n_O^c$$

il est aisé de voir que cette seconde majoration est meilleure que la première trouvée.

4.3.2 Calcul des majorations

Lors de l'exploration, on est amené à effectuer l'assignation d'une fibre (lors de la descente dans l'arbre d'exploration) ou à supprimer la dernière assignation réalisée (lors de la remontée). Comme ces deux opérations se produisent un nombre très élevé de fois, il est nécessaire de pouvoir calculer rapidement les majorants mis en évidence précédemment pour garantir l'efficacité de l'algorithme.

On se place désormais dans un cadre moins général. On suppose qu'on sélectionne les fibres pour la partition en sous-problèmes en en prenant une puis une de ses deux voisines puis à chaque fois la voisine de celle sélectionnée précédemment.

Justification d'une majoration plus large

A toute étape de l'exploration, on a un intervalle de fibres affectées ; son complémentaire caractérise les fibres non affectées. Ainsi, la recherche des composantes connexes du graphe d'accessibilité est-elle grandement simplifiée : on se contente de mettre en évidence des composantes réduites à un seul point représentant une fibre laissée de côté ou un objet inacessible.

Pour revenir aux concepts de la section précédente, on s'intéresse donc à toutes les fibres non encore affecées et aux objets qui leur sont accessibles. Leur graphe d'accessibilité est en général connexe et il serait trop coûteux en temps de rechercher les quelques cas où il pourrait ne pas l'être. Le calcul exact de n_F et de n_O est assez long et demande une mémoire importante pour mettre à jour le graphe d'accessibilité lors du parcours de l'arbre d'exploration. On va donc se contenter de calculer et de mettre à jour des majorations de n_F et de n_O qu'on appellera respectivement N_F et N_O.

Les cas où une fibre ne peut accéder à aucun objet semblent rares. En effet, chaque fibre à un champ d'investigation assez vaste ; celles qui en ont un plus réduit sont celles qui se trouvent près des fibres déjà affectées et donc celles qui viendront immédiatement dans l'exploration. Pour toutes ces raisons, on se contentera de prendre pour N_F le nombre de fibres restant à affecter.

La qualité de la majoration provient donc de la qualité du calcul de N_O. Pour cela, il faut rechercher les objets inaccessibles à tout moment du parcours de l'arbre. Dans ce but, il faut mettre en évidence les objets qui deviennent inaccessibles lorsqu'une fibre est assignée – ou en tout cas le maximum d'objets.

Nombre d'objets encore accessibles

Lorsque, dans l'exploration de l'arbre, on est amené à assigner une fibre à un objet, un ou plusieurs autres objets peuvent devenir inaccessibles par les autres fibres. On peut distinguer deux catégories de tels objets :

1. ceux trop proches de la fibre nouvellement affectée,

2. ceux qui se trouvent entre cette fibre et l'ensemble des fibres anciennement affectées qui peuvent être ainsi *cachés* par la nouvelle fibre.

Il n'est pas difficile de voir si un objet appartient à la première catégorie[2]. Il est en revanche plus subtil de voir si un objet appartient effectivement à la deuxième catégorie.

On suppose qu'on a un objet non affecté situé entre les fibres anciennement affectées et la nouvelle – c'est-à-dire que cet objet aurait pu être observé par cette dernière fibre et que pour l'atteindre la fibre aurait eu un écart angulaire inférieur. Une telle situation est représentée par la figure 4.2.

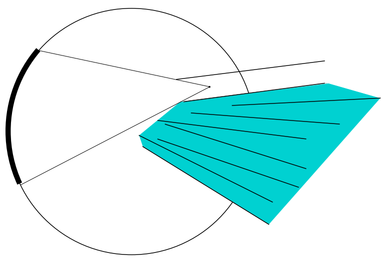


FIG. 4.2: L'objet situé à gauche de la fibre nouvellement assignée est-il encore accessible par des fibres ? Géométriquement ces fibres doivent avoir leur origine dans l'arc (en gras) intercepté par le secteur angulaire d'origine l'objet compris entre la fibre et l'enveloppe convexe des fibres déjà assignées (en gris). On conclut en calculant l'intersection de cet intervalle de fibres et de l'intervalle des fibres pouvant atteindre l'objet *a priori*.

Cette figure illustre également une construction géométrique simple qui permet de connaître l'intervalle des fibres susceptibles d'accéder à l'objet. On recherche en fait la portion du bord du champ focal que pourrait voir un observateur posté à la position de l'objet si on considérait les fibres affectées comme des murs. Elle correspond en fait à un intervalle de fibres, susceptibles d'accéder à l'objet.

L'extrémité supérieure de cet intervalle est calculée au moyen de la tangente à l'enveloppe convexe des fibres déjà affectées passant par l'objet considéré. Il est donc nécessaire de calculer et de mettre à jour cette enveloppe tout au long du déroulement de l'algorithme. Si le point de contact entre la tangente et l'enveloppe convexe se fait en dehors du disque focal, on prendra comme extrémité supérieure la fibre à gauche de la première fibre assignée.

Le calcul de l'extrémité inférieure se fait de la même manière en considérant la droite passant par l'objet considéré et par l'objet observé par la fibre nouvellement affectée.

La construction géométrique de cet intervalle est toujours possible, même dans le cas où l'objet ne *voit* aucune fibre. Mais, dans ce cas, les deux droites sont inversées, c'est-à-dire que, pour aller d'une extrémité à l'autre de l'arc dans le sens direct, on est obligé de passer par la zone des fibres déjà affectées.

[2]Même s'il est rapide de vérifier qu'un objet est – ou non – trop proche de la fibre, il faut quand même éviter de faire cette vérification pour tous les objets. On verra dans la section consacrée à l'implantation de l'algorithme comment limiter cette vérification aux seuls objets concernés.

En particulier, on remarque qu'il est suffisant de montrer que la dernière fibre affectée appartient à cet arc pour dire que l'objet ne *voit* en fait aucune fibre.

Dans le cas où l'ensemble des fibres visibles depuis l'objet n'est pas vide, il faut vérifier que les fibres de cet intervalle peuvent mécaniquement atteindre l'objet avant de conclure à son accessibilité.

4.4 Coupe suivant

Le but de cette section est de mettre en évidence des cas où, dans une partition $P_1, P_2 \cdots P_i$ d'un sous-problème P, l'existence de couples (α, β) dans $[1, i]^2$ tels que

$$\max_{x \in P_\alpha} f(x) \geq \max_{x \in P_\beta} f(x)$$

ce qui permet de ne pas étudier P_β si on a précédemment étudié P_α.

On reste dans le cadre de la section précédente dans lequel les fibres sont affectées de manière consécutive.

On suppose que les fibres F_1 jusqu'à F_{l-1} sont affectées, ce qui définit un ensemble A d'assignations compatibles. On en est à l'affectation de F_l. On suppose que les deux objets $O(P_\alpha)$ et $O(P_\beta)$, correspondant respectivement aux sous-problèmes P_α et P_β, sont accessibles par F_l et que les deux ensembles $A \cup \{\langle F_l, O(P_\alpha)\rangle\}$ et $A \cup \{\langle F_l, O(P_\beta)\rangle\}$ sont compatibles.

On recherche des conditions nécessaires sur $O(P_\alpha)$ et $O(P_\beta)$ pour que le nombre maximal d'assignations pouvant être effectuées une fois les assignations de $A \cup \{\langle F_l, O(P_\alpha)\rangle\}$ imposées soit supérieur au nombre maximal d'assignations pouvant être effectuées une fois les assignations de $A \cup \{\langle F_l, O(P_\beta)\rangle\}$ imposées, soit en d'autres termes

$$\mathcal{ASS}^O_{A \cup \{\langle F_l, O(P_\beta)\rangle\}} \leq \mathcal{ASS}^O_{A \cup \{\langle F_l, O(P_\alpha)\rangle\}}$$

Pour cela, il suffit que le graphe d'accessibilité de $\mathcal{ASS}^O_{A \cup \{\langle F_l, O(P_\beta)\rangle\}}$ soit inclus dans celui de $\mathcal{ASS}^O_{A \cup \{\langle F_l, O(P_\alpha)\rangle\}}$. Afin d'écrire cette condition d'inclusion, on considère une fibre $f \in \{F_{l+1} \cdots F_N\}$ quelconque et un objet o qui lui est accessible. Cela donne

$$\forall a \in A \cup \{\langle F_l, O(P_\beta)\rangle\}\ \langle f, o\rangle \bigtriangleup a \Longrightarrow \forall b \in A \cup \{\langle F_l, O(P_\alpha)\rangle\}\ \langle f, o\rangle \bigtriangleup b$$

Ce qui se simplifie en :

$$\langle f, o\rangle \bigtriangleup \langle F_l, O(P_\beta)\rangle \Longrightarrow \langle f, o\rangle \bigtriangleup \langle F_l, O(P\alpha)\rangle$$

soit, dit autrement,

$$\langle f, o\rangle \not\bigtriangleup \langle F_l, O(P_\alpha)\rangle \Longrightarrow \langle f, o\rangle \not\bigtriangleup \langle F_l, O(P_\beta)\rangle$$

Ces deux dernières implications peuvent être vues de la manière suivante : pour toute solution réalisable de P_β, il est possible de fabriquer une solution réalisable de P_α de même cardinal en remplaçant l'objet $O(P_\beta)$ affecté à F_l par l'objet $O(P_\alpha)$, qui est par construction libre et accessible. Cela montre d'une autre manière qu'il est vain d'étudier P_β.

La vérification d'une telle propriété sur l'ensemble des objets accessibles par F_l n'est guère coûteuse en temps de calcul puisqu'elle est quadratique en le nombre d'objets accessibles. En remarquant qu'un couple $(O(P_\alpha), O(P_\beta))$ vérifie la condition ci-dessus en général quand $O(P_\alpha)$ se trouve "coincé" (voir figure 4.3) entre les fibres déjà assignées et F_l, on peut rendre la vérification linéaire sans risquer de perdre un trop grand nombre de couples $(O(P_\alpha), O(P_\beta))$ satisfaisant la contrainte.

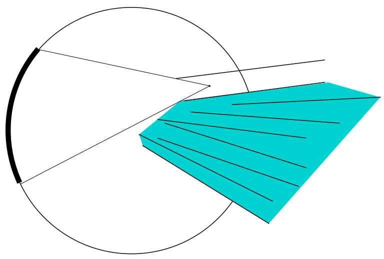


FIG. 4.3: L'intervalle des fibres à regarder est l'arc intercepté par le secteur angulaire d'origine $O(P_\alpha)$ et délimité par $O(P_\beta)$ et l'enveloppe convexe de A.

A chaque étape de l'exploration, au moment d'affecter une fibre F_l, on recherchera donc des couples de tels objets $O(P_\alpha)$ et $O(P_\beta)$ ce qui permettra de ne pas lancer d'exploration pour l'affectation $\langle F_l, O(P_\beta)\rangle$.

4.5 Départ et sens de parcours

Comme on l'a vu ci-dessus, le fait de sélectionner les fibres d'une manière consécutive met en évidence quelques propriétés simplificatrices. On reste dans cette optique dans cette section qui étudie le choix des deux premières fibres sélectionnées – ce qui implique d'ailleurs l'ordre du choix de toutes les autres fibres. On recherche dans ce choix un moyen de diminuer *a priori* le temps de calcul global. Les idées qui suivent restent à un niveau heuristiques mais les quelques tests effectués semblent prouver leur bien fondé.

Une idée majeure pour diminuer le nombre de solutions explorées est de commencer par le plus dur à faire, c'est-à-dire d'assigner en premier les zones qui ont le plus de chances d'avoir des fibres non affectées. En effet, lorsqu'on procède à l'exploration, il est préférable d'avoir le plus tôt possible des fibres non affectées car cela permet de faire des coupes plus haut dans l'exploration.

Intuitivement, il est donc préférable de commencer par affecter une zone peu dense où, localement, le nombre de fibres est supérieur au nombre d'objets accessibles. Pour déterminer une telle région et pour choisir une bonne fibre de départ associée à un bon sens de parcours, nous utilisons la méthode suivante :

- On suppose que la meilleure solution connue – heuristiquement – comprend n fibres non affectées.

– A partir de chaque fibre $f \in [0, N-1]$ et pour les deux sens $s \in \{-1, 1\}$ de parcours, on lance H_0 et on l'arrête dès qu'il a été obligé de laisser n fibres de côté. On mémorise le nombre F(f,s) de fibres déjà affectées à ce moment-là.

– On choisit comme fibre de départ et sens de parcours le couple (f, s) qui minimise $F(f, s)$.

La figure 4.4 indique les variations de $F(f, s)$.

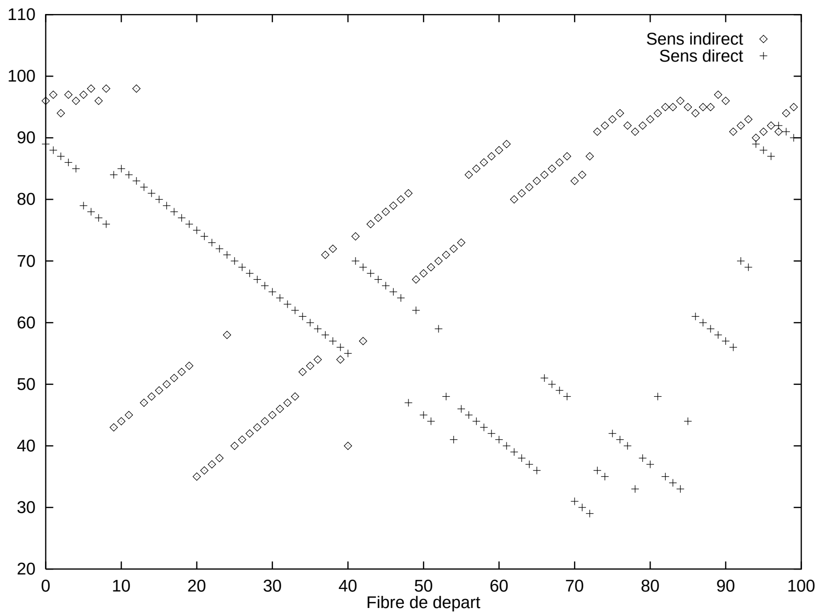


FIG. 4.4: La meilleure solution connue pour ce problème comportait 8 fibres non affectées. Ce diagramme présente, en fonction de la fibre de départ et du sens de parcours, la profondeur d'exploration après laquelle H_0 arrive à 8 fibres non affectées. En partant de la fibre 72 et en tournant dans le sens direct, H_0 arrive à 8 fibres non affectées après avoir tenté d'assigner seulement 29 fibres.

Dans le cas où l'on observe par exemple deux zones peu denses séparées par deux zones denses, on peut se dire qu'il convient d'affecter la première zone peu dense puis la seconde avant de terminer par les zones denses. Quand on passe à une fibre dont la voisine n'est pas affectée, on agrandit *a priori* le nombre de possibilités d'affectations et donc de branches dans l'arbre d'exploration. Il ne faut donc procéder ainsi que dans des cas bien spécifiques. De toute manière, trouver l'ordre optimal de visite des fibres apparaît comme un problème de combinatoire difficile...

4.6 Implantation de l'algorithme exact

Nous notons ici quelques remarques pratiques relatives à l'implantation efficace de l'algorithme de *branch-and-bound*.

La sélection des fibres se fait de manière consécutive dans l'ordre indiqué par l'heuristique de la section précédente.

4.6.1 Calculs avant exploration

Toutes les opérations faites à chaque noeud de l'opération sont effectuées un grand nombre de fois. Il faut dont en minimiser le nombre et la complexité. En particulier, il est intéressant de stocker les résultats d'un maximum de calculs qu'on évitera ainsi de refaire à chaque étape.

Avant de partir dans l'exploration de l'arbre des solutions, on calcule donc :

1. Le **tableau d'accessibilité** : la case (i, j) signale si la fibre F_i peut mécaniquement atteindre l'objet O_j, c'est-à-dire si O_j se trouve dans le champ de F_i. Ce tableau a pour taille $\mathcal{O}(N \times n)$.

2. Le **champ de chaque fibre** : pour chaque fibre, les objets accessibles sont classés de gauche à droite, ce qui correspond à l'ordre – heuristique – des branchements. Pour chacun de ces objets, on a mémorisé la position de l'axe de rotation de la fibre lorsqu'elle lui est affectée. La taille de cette structure est également en $\mathcal{O}(N \times n)$.

NOTE : On aurait pu également calculer tous les risques de collisions pour chaque paire d'assignations possibles mais on aurait dû supporter une structure de taille $\mathcal{O}(N^2 \times n^2)$.

4.6.2 Résultats et temps de calcul

En un peu plus d'une journée de calcul, on a ainsi montré que le problème `st100` avait un maximum de 92 assignations. Cet optimum est trouvé très rapidement par les algorithmes heuristiques, ce qui prouve leur qualité. De plus, la recherche des solutions comportant plus de 91 assignations aurait demandé un temps de calcul plusieurs fois plus long. Le gain de temps apporté par les heuristiques pour le calcul exact est clairement mis en évidence sur cet exemple.

Conclusion

L'étude du problème de l'assignation de fibres optiques pour la spectroscopie multi-objets a permis de mettre en avant des résultats intéressants.

Sur le plan pratique, les résultats donnés par les algorithmes heuristiques permettent d'obtenir dans une durée convenant parfaitement aux astronomes des résultats très bons. Ils sont en effet de l'ordre de 10% supérieurs à ceux obtenus jusqu'alors et frôlent couramment le maximum absolu.

Sur le plan théorique, la suite d'algorithmes H_n est originale et a des propriétés intéressantes de croissance et de convergence. Elle est de plus généralisable à d'autres problèmes. La nature géométrique de l'énoncé a été exploité pour mettre en évidence des propriétés permettant de trouver la solution exacte du problème par un simple calcul de *branch-and-bound*.

Elargissement des recherches

Le problème de l'assignation de fibres optiques peut être généralisé de manière à compliquer la combinatoire. On peut ainsi affecter des coefficients de priorité aux objets, mettant ainsi en évidence ceux qui sont plus intéressants que d'autres, pour obtenir un problème pondéré. Le champ d'objets à observer dépasse souvent le champ du télescope. Le problème de le partitionner en différentes pauses, qui se recouvrent partiellement, en tenant compte des temps d'observation de chaque objet est un problème d'optimisation combinatoire très intéressant en lui-même et difficile, surtout s'il est couplé avec le problème d'assignation.

La section 3.2.5 a montré que la suite d'heuristiques H_n pouvait se généraliser de manière assez efficace à d'autres problèmes d'optimisation combinatoire. La post-optimisation à l'aide des H_n pourrait être généralisé de la même manière. Des résultats théoriques plus poussés sur l'efficacité *a priori* de H_n peuvent sans doute également être trouvés.

Remerciements

Je tiens pour terminer à remercier Philippe Chrétienne et Paul Felenbok, mes deux directeurs de stages, pour leurs conseils et leurs encouragements, Robert Cori pour l'intérêt qu'il a porté au sujet et les idées qu'il a eues, Jean Czarny, Jean Guérin et Jean-Pierre Lemonnier pour leurs explications sur le fonctionnement général de FUEGOS et sur de nombreux aspects de l'astronomie

et, de manière générale tous les membres de l'équipe FUEGOS qui m'ont permis de travailler dans d'excellentes conditions d'amitié malgré les difficultés qu'a pu rencontrer le projet.

Annexe A

Tests

On archive ici les données numériques utilisées pour tester les algorithmes.

A.1 L'instrument

Ces données ne correspondent pas exactement aux dimensions de FUEGOS car elles n'étaient pas toutes connues au moment de la réalisation des tests. Les ordres de grandeur sont néanmoins respectés.

Dans la section 1.5, on prend comme unité de longueur le rayon du disque focal qui est de 452 mm. Comme il a été dit dans le chapitre concernant la modélisation, les marches n'ont pas été introduites.

Tous les tests faisaient intervenir $N = 100$ fibres dont les caractéristiques suivent :

Longueur de la tige	l	827 mm
Diamètre de la tige	d	9 mm
Diamètre du bouton	b	9 mm
Début de course du chariot	m	777 mm
Fin de course du chariot	M	1327 mm
Angle de débattement	Θ	$5°$

A.2 Les fichiers d'objets

Quatre fichiers de 100 objets ont été créés spécialement pour tester et comparer les algorithmes. Les 100 points correspondant sont répartis uniformément dans un disque de rayon 1.

A.2.1 st100

0	0.527590	0.401575	1	-0.175916	0.902781	2	0.015474	-0.199101	3	-0.482114	0.834847	4	-0.828692	-0.239179
5	0.337866	-0.574149	6	0.418052	0.546029	7	0.274722	-0.727885	8	0.201072	0.780521	9	-0.035602	0.681615
10	-0.824036	0.472101	11	0.003706	-0.361685	12	-0.152378	-0.478585	13	-0.349260	0.606685	14	0.145566	0.405759
15	-0.263476	0.520213	16	-0.366166	-0.741349	17	0.689592	-0.557935	18	0.280054	0.576247	19	-0.197788	-0.664243
20	-0.537854	0.503827	21	0.253262	0.047983	22	0.333180	0.474245	23	-0.078246	-0.009845	24	-0.108184	0.202753
25	0.573862	0.269159	26	-0.153324	-0.617539	27	-0.251904	-0.103239	28	0.267518	-0.687521	29	-0.271956	-0.141707
30	-0.478312	-0.188431	31	-0.067882	0.761943	32	0.261552	-0.253015	33	-0.663076	0.199653	34	0.903098	-0.034965
35	-0.375608	0.067425	36	-0.887162	-0.322361	37	-0.053742	0.077795	38	-0.396704	0.826649	39	0.009182	-0.326209
40	-0.431604	-0.373931	41	0.209912	-0.255535	42	-0.048220	-0.276723	43	0.937922	-0.167981	44	0.528400	-0.572151
45	0.140046	-0.847825	46	0.032060	-0.151483	47	-0.262502	0.488267	48	0.488550	-0.167385	49	-0.574508	0.262781
50	0.081650	0.681027	51	0.748992	0.357497	52	-0.402882	-0.887649	53	-0.056980	0.307509	54	-0.442664	0.525873
55	-0.333802	-0.942057	56	-0.950878	-0.145357	57	-0.922896	0.050921	58	-0.325266	0.651663	59	0.475898	0.828971
60	-0.406520	0.521121	61	-0.239290	0.323719	62	-0.109260	0.039005	63	-0.831918	-0.386781	64	-0.621792	-0.379943
65	-0.190050	0.820095	66	-0.265140	0.052245	67	-0.209992	-0.700655	68	-0.198812	-0.422963	69	0.663298	0.681363
70	-0.704176	0.417993	71	0.503374	0.184815	72	0.732668	0.235013	73	0.561242	-0.047093	74	0.282088	-0.602751
75	0.090150	0.244455	76	-0.142700	0.230461	77	-0.488014	-0.122109	78	-0.376704	0.119865	79	-0.988930	-0.056225
80	0.024364	0.610165	81	-0.811510	-0.083845	82	-0.926696	-0.351375	83	-0.383530	-0.588073	84	0.750178	-0.319629
85	0.061104	0.738345	86	0.899036	0.021477	87	-0.785990	0.136939	88	0.178504	-0.734751	89	-0.639884	0.255389
90	-0.737262	0.240611	91	-0.444064	0.253593	92	-0.335010	0.166719	93	0.089484	-0.603179	94	-0.939016	-0.016815
95	0.610742	0.084151	96	-0.400988	-0.238963	97	0.562236	0.796357	98	0.530330	0.387275	99	-0.126552	-0.503487

A.2.2 et100

0	0.000523	0.775656	1	0.080001	-0.814810	2	-0.134553	-0.736748	3	-0.419395	0.427698	4	0.480771	-0.331904
5	0.068319	0.059180	6	-0.245387	-0.703222	7	0.642929	-0.388010	8	0.110167	0.355908	9	-0.305363	0.378850
10	-0.860247	0.021550	11	0.321359	-0.532772	12	0.130789	0.624826	13	0.623595	0.496072	14	0.152673	-0.315130
15	0.789575	-0.461068	16	0.805917	0.327826	17	0.200803	-0.373536	18	0.358463	-0.905844	19	-0.620843	0.172394
20	0.087515	-0.626952	21	-0.251951	-0.749002	22	0.223543	0.917796	23	0.221069	0.041026	24	-0.202487	-0.802546
25	-0.521019	-0.291174	26	-0.740277	-0.195288	27	-0.375999	-0.707482	28	-0.226777	-0.449708	29	-0.096899	0.262770
30	0.847811	0.460352	31	0.810873	-0.001602	32	-0.377291	0.892746	33	0.831547	0.182232	34	0.954647	-0.279804
35	0.552301	0.021666	36	-0.709325	0.328816	37	0.678543	0.125596	38	-0.735835	0.127482	39	-0.806485	-0.252024
40	-0.029433	0.243636	41	-0.378275	0.002386	42	-0.772189	-0.330848	43	0.611033	0.103582	44	-0.653313	0.136652
45	-0.316907	-0.714326	46	0.450843	-0.163528	47	0.947149	-0.213374	48	-0.375095	-0.715186	49	0.238063	-0.410116
50	0.516997	0.630106	51	-0.012021	-0.172056	52	0.252929	-0.537434	53	0.556007	-0.204524	54	-0.204739	0.708530
55	-0.607485	0.175232	56	-0.479047	0.794878	57	0.479839	-0.312404	58	0.098555	0.840216	59	0.074353	0.123990
60	0.879149	0.128354	61	-0.548951	-0.464210	62	-0.154545	-0.746916	63	0.384741	-0.256454	64	-0.358303	0.544134
65	0.207815	-0.452876	66	-0.844317	0.389344	67	0.018201	-0.156834	68	-0.131777	-0.479348	69	0.694101	-0.515478
70	0.755291	0.604536	71	0.539345	-0.411978	72	-0.303561	0.172068	73	0.389389	0.649538	74	-0.499885	0.279824
75	-0.191607	-0.691058	76	-0.208593	0.023740	77	0.647621	-0.613350	78	0.188491	-0.668120	79	0.375105	0.288358
80	-0.357821	0.145216	81	-0.528519	-0.428610	82	0.263583	-0.615060	83	0.237877	0.546378	84	0.131003	-0.938408
85	0.861105	0.119190	86	0.743575	-0.474364	87	0.769385	0.203246	88	0.837775	-0.410340	89	0.047147	-0.477944
90	-0.170719	0.367686	91	-0.991609	0.068020	92	0.365021	-0.379694	93	-0.326493	-0.718688	94	0.140633	0.919966
95	-0.529409	0.325196	96	0.268437	0.825386	97	-0.302309	-0.949832	98	-0.171375	0.924150	99	0.637261	-0.660606

A.2.3 ob100

0	-0.109428 0.840277	1	0.351210 -0.022181	2	-0.667612 0.255757	3	0.123408 0.530313	4	-0.374002 0.598319
5	0.461372 -0.056507	6	-0.080486 -0.415413	7	0.476518 0.619687	8	-0.245420 0.194813	9	-0.837698 -0.230625
10	-0.374420 0.639669	11	0.018634 -0.658245	12	-0.611368 -0.178511	13	0.050966 -0.807529	14	0.436228 -0.830867
15	0.266018 -0.922189	16	-0.245648 0.538857	17	0.095854 0.757007	18	0.912924 -0.159323	19	0.839546 -0.273877
20	0.795528 -0.539231	21	0.541010 -0.696797	22	-0.517088 -0.311591	23	-0.668130 -0.695169	24	0.071372 0.218901
25	0.624298 0.478235	26	0.197816 -0.183535	27	0.595574 -0.794889	28	0.318820 0.250701	29	0.118274 0.918547
30	-0.344752 -0.911159	31	0.456026 0.617867	32	0.231528 -0.913791	33	-0.430042 -0.200473	34	-0.157548 -0.205251
35	0.016512 0.076217	36	0.178302 -0.728609	37	-0.724308 -0.155787	38	0.532618 0.524923	39	-0.213352 0.797425
40	-0.599996 -0.139603	41	-0.045470 0.994931	42	0.186288 0.389545	43	-0.119250 -0.429489	44	0.116956 0.162917
45	-0.461510 -0.496021	46	-0.170168 -0.628127	47	0.158342 -0.135993	48	-0.225420 -0.563043	49	0.488978 -0.527389
50	0.166110 -0.647617	51	-0.114454 -0.134821	52	0.776184 -0.309807	53	0.089014 -0.535625	54	-0.670172 0.023821
55	-0.445246 -0.013741	56	0.163344 -0.458615	57	-0.058308 0.902725	58	0.634010 0.423371	59	-0.302362 0.266407
60	-0.101932 -0.157827	61	-0.665364 -0.471755	62	0.667978 0.644411	63	0.988548 -0.149523	64	0.541858 0.268467
65	-0.174736 0.535913	66	0.614684 -0.512987	67	0.648954 0.144299	68	-0.569976 0.300577	69	0.037574 0.160903
70	-0.324428 -0.240035	71	0.692818 -0.290781	72	-0.543584 -0.515367	73	0.635166 0.692095	74	-0.145972 0.316949
75	0.707754 -0.197221	76	0.301752 -0.048111	77	-0.611484 -0.335411	78	-0.584368 0.772169	79	0.197454 0.530799
80	-0.834180 -0.086267	81	-0.183718 0.746123	82	0.496744 -0.091135	83	0.678054 -0.310809	84	-0.265452 -0.459843
85	-0.733774 -0.265469	86	-0.324096 -0.012615	87	0.420222 0.902239	88	-0.505812 -0.355595	89	0.126602 -0.990213
90	-0.306090 -0.528169	91	-0.195004 0.874285	92	-0.584222 -0.718861	93	0.601008 0.023977	94	0.463662 0.354383
95	0.371548 0.362853	96	-0.328954 0.562247	97	0.449012 0.681885	98	-0.669678 -0.105117	99	0.317280 -0.773223

A.2.4 as100

0	0.110474 0.194683	1	-0.707560 0.392561	2	0.308420 -0.107219	3	-0.116772 -0.004763	4	-0.420550 0.260459
5	-0.133754 0.245575	6	-0.374254 -0.057885	7	-0.353568 0.702617	8	0.252126 0.373695	9	-0.384372 -0.519723
10	-0.378390 0.352475	11	0.286712 0.544337	12	0.190518 -0.743625	13	-0.550974 0.083795	14	0.351120 -0.425975
15	0.962574 0.165807	16	0.662460 -0.412219	17	-0.348442 0.374823	18	-0.643628 -0.340611	19	-0.075586 -0.488161
20	-0.373908 0.112821	21	0.448074 -0.609477	22	0.107352 0.106801	23	-0.651370 -0.715881	24	-0.623198 0.670003
25	-0.496528 -0.405399	26	0.270190 -0.136305	27	-0.044644 -0.888155	28	-0.644998 -0.505429	29	-0.492664 -0.086623
30	0.194118 0.354183	31	-0.659426 0.174975	32	0.670838 0.206199	33	-0.454556 0.675149	34	-0.009266 -0.632593
35	-0.514564 -0.482811	36	0.268378 0.026379	37	-0.828890 -0.206489	38	-0.264044 -0.290755	39	0.142386 -0.827901
40	0.760832 0.580665	41	-0.619650 0.026207	42	-0.615764 -0.608523	43	0.103960 -0.423695	44	0.335830 -0.099241
45	-0.498492 -0.563667	46	0.938850 0.326643	47	-0.512848 -0.396503	48	0.796334 0.350159	49	-0.826276 -0.129819
50	0.255418 0.261355	51	-0.671866 -0.193465	52	0.060660 -0.526051	53	0.684434 -0.035741	54	-0.735904 0.099097
55	-0.524450 -0.734785	56	-0.509684 0.752085	57	0.085354 0.090971	58	0.099064 0.960209	59	-0.860234 -0.250185
60	0.728868 0.080525	61	-0.160830 0.411859	62	-0.605170 -0.010961	63	-0.344900 -0.064443	64	0.895898 -0.277941
65	0.429736 -0.451903	66	-0.037146 0.043559	67	0.224852 -0.861827	68	-0.675342 0.283331	69	0.660076 0.081205
70	-0.960694 -0.169541	71	0.558808 0.773937	72	0.430870 0.639255	73	0.042372 -0.733715	74	0.044912 -0.902167
75	-0.372114 -0.002289	76	0.278300 -0.921691	77	0.579834 0.209451	78	-0.094072 0.975905	79	-0.308812 -0.613923
80	0.272544 0.671961	81	-0.516706 -0.808065	82	0.566476 -0.390635	83	0.600106 -0.600293	84	-0.468040 -0.274415
85	-0.457982 -0.828013	86	0.349136 0.558793	87	0.073212 -0.703611	88	-0.038296 0.810753	89	0.361638 0.793831
90	-0.396524 0.116541	91	-0.563534 -0.544381	92	-0.800512 -0.508231	93	0.183294 -0.652577	94	-0.077814 0.468219
95	0.490264 -0.069263	96	-0.208682 -0.816041	97	0.086498 0.466675	98	0.590128 0.090921	99	-0.940754 0.080207

A.2.5 cl100

0	0.004157 0.011475	1	-0.066764 0.053446	2	-0.000847 0.002737	3	0.074342 -0.059344	4	0.011134 0.006160
5	0.046598 -0.144472	6	0.222259 0.107493	7	0.779742 -0.023342	8	-0.022049 0.335038	9	0.517124 0.504548
10	0.020860 0.750092	11	-0.471030 -0.687186	12	0.365945 -0.139292	13	-0.122578 -0.399492	14	-0.044579 0.045316
15	-0.376091 0.046847	16	-0.231953 0.309034	17	-0.079035 -0.022605	18	-0.001454 0.036268	19	-0.001929 0.000454
20	0.174319 0.101318	21	-0.110189 -0.055296	22	0.467602 0.799901	23	-0.373779 -0.229961	24	0.056922 -0.038465
25	-0.007839 0.028850	26	-0.299852 0.300520	27	-0.068462 -0.106968	28	0.164637 -0.687759	29	0.002447 0.001590
30	-0.852178 -0.063508	31	0.297558 0.255179	32	-0.155116 0.127467	33	0.009026 0.007166	34	-0.025857 0.118761
35	-0.018997 0.040091	36	-0.068397 -0.124419	37	-0.210008 -0.141953	38	0.047827 -0.055154	39	0.696460 0.589893
40	-0.125953 0.004298	41	0.220199 -0.133274	42	-0.022942 0.029893	43	0.000021 -0.000334	44	0.355577 -0.414305
45	-0.749150 -0.239304	46	0.052675 0.103340	47	0.121868 -0.012623	48	0.008175 0.009318	49	0.135799 -0.951559
50	0.122293 0.433088	51	-0.134818 -0.265209	52	0.282365 -0.069565	53	-0.023872 -0.067477	54	-0.660174 0.726374
55	-0.234275 -0.025401	56	0.071040 0.011883	57	0.267727 0.174588	58	0.007905 -0.028191	59	-0.137185 -0.523970
60	0.009532 0.017893	61	-0.506094 -0.280988	62	0.142709 0.023714	63	-0.114625 0.093048	64	0.049808 0.888068
65	-0.009589 -0.004985	66	0.091558 -0.065856	67	-0.231653 0.097950	68	-0.408251 0.781195	69	-0.188883 -0.143676
70	-0.373255 0.512664	71	0.020685 -0.021977	72	-0.004651 0.065072	73	-0.539125 0.838294	74	0.037448 -0.100950
75	-0.003630 0.000600	76	0.456687 -0.104317	77	0.056658 0.040675	78	-0.021587 0.225115	79	0.003149 -0.240389
80	0.006127 0.003941	81	0.009509 -0.002428	82	-0.000023 -0.019535	83	0.464690 0.257465	84	-0.598948 0.370362
85	0.199222 -0.141869	86	0.155521 -0.010728	87	0.394454 -0.169059	88	-0.140186 -0.790296	89	-0.176304 -0.054963
90	0.892582 0.250578	91	0.032672 0.038583	92	-0.021913 0.103100	93	0.162823 -0.105256	94	0.363024 0.197201
95	0.380207 0.212785	96	0.000748 -0.001352	97	0.046788 -0.308741	98	-0.119005 -0.142489	99	-0.000183 0.001786

Annexe B

Les programmes

B.1 Présentation

Plusieurs programmes ont été mis au point pour tester, comparer et faire progresser les algorithmes heuristiques et exacts. Ils se lancent selon la syntaxe suivante :

```
vlt [-f #fibres] nomfic
```

- `nomfic` est le nom du fichier contenant la position des objets sur le disque focal. Le format de ce fichier est très simple. Chaque ligne est consacrée à un objet et est composée de deux nombres réels, abscisse et ordonnée de l'objet. Pour normaliser ces coordonnées, le disque focal est ramené au disque unité.

- Le paramètre facultatif `-f #fibres` permet de préciser le nombre de fibres du positionneur. La valeur par défaut est de 100.

Quelle que soit la version du programme, le résultat trouvé est mis dans un fichier appelé `perm`. Ce fichier est mis à jour pendant que le programme tourne, il contient à tout moment la meilleure solution trouvée depuis le début de l'exécution. Tout fichier `perm` contient autant de lignes qu'il y a détoiles. Chaque ligne contient deux entiers. Le premier rappelle le numéro de l'étoile et le second le numéro de la fibre qui lui est associé. S'il n'y en a pas, ce second chiffre est remplacé par la valeur -1.

B.2 Les paramètres

Lors de la description des algorithmes, un grand nombre de paramètres ont été introduits. Pour plus de clarté ces paramètres ont fait l'objet de constantes C. Nous donnons ici la correspondance entre ces constantes et leur rôle dans l'algoritme.

- `NIVEAU_MAX` est le niveau de l'heuristique utilisée.

- `NIVEAU_POST` est le niveau de l'heuristique utilisée pendant la post-optimisation.

- `SENS` indique le sens de départ lors de l'affectation des fibre. Cette constante vaut 1 ou -1.

- `MAX_OBJETS` indique le nombre d'objets accessibles pris en compte dans $\mathcal{O}_{acc}^{(i,a)}$. Voir section 3.2.3.

- `DEC_EACH` est le paramètre η de la section 3.4.

- `OPTLOC` est le nombre de fibres désallouées pendant la post-optimisation. c'est le paramètre d de la section 3.5.

- `NBPOST` est le nombre de phases de post-optimisation (section 3.5).

- `NBTOURS` est la durée en nombre de tours de chaque phase de post-optimisation. Dans la section 3.5, on a considéré que cette valeur était de 1.

- `TESTSUIV` vaut 1 ou 0 selon que l'astuce de la section 3.6.1 est implantée ou non.

Un autre groupe de constantes concerne les paramètres mécaniques de l'instrument. Leur signification est écrite en commentaire dans le source du programme. Pour plus d'information, on peut aussi se rapporter à l'annexe A.

Enfin, deux options de compilation concernent plus particulièrement le programme :

- `verbose` permet de décider si les informations concernant l'avancée du calcul doivent être affichées.

- `test` permet de créer un fichier de sortie dans lequel sont consignés les résultats de calcul en fonction de la fibre de départ.

B.3 L'environnement graphique

Afin de faciliter la visualisation des assignations proposées par le calcul, un interface graphique a été programmée en TCL et TK. La figure B.1 montre la fenêtre principale de cet environnement. Au lancement, on sélectionne le fichier indiquant la position des étoiles en cliquant sur le bouton **File**, ce qui ouvre une boîte de dialogue permettant de spécifier le nom du fichier.

Pour lancer le calcul d'optimisation, il suffit de cliquer sur le bouton **Calculate**. Il est alors automatiquement fait appel au programme de calcul écrit en C. Le fichier de sortie est ensuite récupéré pour afficher le résultat.

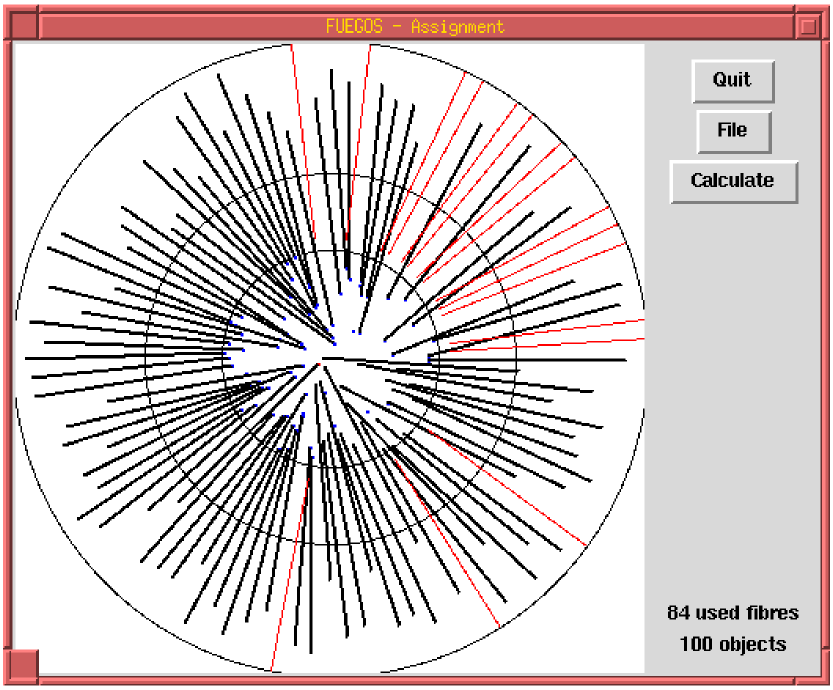


Fig. B.1: Interface graphique

Bibliographie

[Boi95] JEAN-DANIEL BOISSONNAT, MARIETTE YVINEC, *Géométrie Algorithmique*, Ediscience, 1995.

[Chr79] NICOS CHRISTOPHIDES (ed.), *Combinatorial Optimization*, John Wiley & Sons Ltd, 1979.

[Don92] R. HANK DONNELY, JEAN P. BRODIE, S. L. ALLEN *Multiobject Spectroscopy : Engineering Design Constraints from Fiber Asignment Optimization Studies*, Publicatoins of the Astronomical Society of the Pacific, **104 :** 752-759, Sept. 1992.

[Kuhn55] H. W. KUHN, *The Hungarian method for assignment problems*, Naval Res. Logist. Quat. 3, p.253-258, 1955.

[Lew93] IAN J. LEWIS, IAN R. PARRY, RAY M. SHARPLES, KEITH TAYLOR *Fibre-to-Object Allocation Algorithmes for AutoFib type fibre Positioners*, Fiber Optics in Astronomy II, ASP Conference Series, Vol. 37, p.249-269, 1993, Peter M. Gray (ed.)

[Reev93] COLIN R. REEVES (ed.), *Modern heuristic techniques for combinatorial problems*, Blackwell Scientific Publications, 1993.

[Rein77] EDWARD M. REINGOLD, *Combinatorial Algorithms*, Prentice Hall Inc., 1977.

[Tou] PIERRE TOURNASSOUD, RÉGIS VAILLANT *Des robots et des étoiles*, Note, INRIA, Domaine de Voluceau, 78153 le Chesnay CEDEX.