Titre: Title:	Conception d'une interface graphique pour la fabrication d'horaires d'équipages aériens
Auteur: Author:	Souheil Zerbé
Date:	1996
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Zerbé, S. (1996). Conception d'une interface graphique pour la fabrication d'horaires d'équipages aériens [Mémoire de maîtrise, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/6753/

URL de PolyPublie: PolyPublie URL:	https://publications.polymtl.ca/6753/
Directeurs de recherche: Advisors:	François Soumis
Programme: Program:	Non spécifié

UNIVERSITÉ DE MONTRÉAL

CONCEPTION D'UNE INTERFACE GRAPHIQUE POUR LA FABRICATION D'HORAIRES D'ÉQUIPAGES AÉRIENS

SOUHEIL ZERBÉ

DÉPARTEMENT DE MATHÉMATIQUES ET DE GÉNIE INDUSTRIEL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

MÉMOIRE PRÉSENTÉ EN VUE DE L'OBTENTION

DU DIPLÔME DE MAÎTRISE ÈS SCIENCES APPLIQUÉES

(MATHÉMATIQUES APPLIQUÉES)

AOÛT 1996

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Ce mémoire intitulé:

CONCEPTION D'UNE INTERFACE GRAPHIQUE

POUR LA FABRICATION D'HORAIRES D'ÉQUIPAGES AÉRIENS

présenté par: ZERBÉ Souheil

en vue de l'obtention du diplôme de: Maîtrise ès sciences appliquées

a été dûment acceptée par le jury d'examen constitué de:

M. GRANGER Louis, M.Sc., président

M. SOUMIS François, Ph.D., membre et directeur de recherche

M. DUMAS Yvan, Ph.D., membre et codirecteur

M. BOYLE Michael, Ph.D., membre

REMERCIEMENTS

Ce mémoire a bénéficié de l'aide de nombreuses personnes sans lesquelles il n'aurait pu être mené à terme. Je désire remercier sincèrement François Soumis, mon directeur de recherche, dont la disponibilité, les conseils judicieux et le soutien financier m'ont été d'un précieux secours.

J'ai pu progresser dans mon travail grâce à la collaboration de Michael Boyle, qui a bien voulu m'aider à diverses étapes du projet, et Yvan Dumas, qui m'a facilité l'utilisation du logiciel Altitude.

Je remercie également Gregg Labute, François Lacoursière, Nina Visconti et Bernard Telisma pour avoir bien voulu répondre à mes questions concernant des points techniques, ainsi que Line Rondeau pour sa précieuse collaboration dans la correction du mémoire.

Je désire exprimer ici ma gratitude à mes parents pour leur empathie et leurs affectueux encouragements. Finalement, un gros merci à Alison, pour son appui inconditionnel et pour la patience dont elle a fait preuve tout au long de la rédaction de ce mémoire.

RÉSUMÉ

Dans ce mémoire nous présentons une interface graphique utilisateur qui a été conçue afin d'aider les planificateurs à fabriquer des horaires. Nous nous intéressons spécifiquement à la manipulation des horaires et des rotations d'équipages aériens. Le logiciel conçu est une application multi-fenêtrée qui offre des outils de visualisation et d'éditions de données.

La fenêtre de base est une région graphique dans laquelle on affiche des rotations d'équipages, réparties sur plusieurs lignes. Elle est dotée de plusieurs menus déroulants et de boutons poussoirs donnant accès à d'autres panneaux et à d'autres fonctionnalités. Cette fenêtre graphique est le premier contact que l'usager aura avec le système et qui lui permettra par la suite d'interagir avec celui-ci. En effet, l'usager peut contrôler et modifier le contenu de la fenêtre par des opérations graphiques manuelles. Des régions textes sont inclues dans la fenêtre principale qui donnent des informations telles que les statistiques, le protocole de traçage et les messages d'erreurs de la solution affichée à l'écran.

L'interface graphique discutée dans ce mémoire possède un traitement unique de la mémorisation des étapes. Une fonctionnalité très puissante permet de suivre l'évolution d'un scénario et de retourner en arrière si désiré. Toutes les étapes d'un scénario sont accumulées grâce à un mécanisme de traçage (*Audit Trail*) qui offre au planificateur une vue globale de toutes les modifications produites depuis le début du scénario. De plus, le planificateur peut retourner à une étape précédente et créer une autre branche du scénario en modifiant des données ou des paramètres. L'interface permet de naviguer dans l'arbre des étapes contenant les variantes des scénarios qui ont été étudiés.

Ce mémoire inclut la présentation d'un module d'affichage assez complexe qui gère une énorme quantité de données. Il est possible d'afficher, de configurer et de manipuler ces données. On présentera donc l'interrelation des objets graphiques avec

leurs pointeurs aux structures de données en mémoire, ainsi que les puissants mécanismes utilisés pour la gestion et la mise à jour des objets graphiques. Finalement, on décrira les différents outils mis à la disposition de l'usager pour mieux contrôler ces informations.

Dans ce mémoire, nous présentons les techniques les plus importantes du système ainsi qu'une description détaillée de leur implantation. Nous décrivons aussi l'organisation de l'environnement de l'application. Nous terminons en fournissant des exemples réels qui convaincront le lecteur de la nécessité d'un tel outil pour la planification dans le domaine aérien.

ABSTRACT

This thesis introduces a graphical user interface that was created to help planners build schedules. We are particularly interested in the area of scheduling and crew member pairing manipulation in the airline business. This is a multitasking application software that offers tools for editing and viewing data.

The main panel is a graphical window in which pairings are displayed, in several rows. It contains different types of menus and push buttons from which other panels and functionality can be accessed. This window is the user's first mean of interacting with the system. In fact, the user can control as well as modify the content of the graphical window by performing graphical operations manually. The main panel also displays information in text areas that include statistics, audit trails and error messages for the current solution.

The graphical interface discussed in this thesis has a unique process for storing each scenario steps. A powerful functionality allows, through a complex process, to follow the evolution of the scenario from its creation. All the steps taken by the planner are stored using an audit trail mechanism that offers a global view of all the modifications done to the scenario from the moment it was created.

This thesis includes the introduction of a complex display module that administers a large number of data. It is possible to display, configure and manipulate those data. We will, therefore, present the interrelation of the graphical objects with the real data structures pointers in memory, as well as the powerful mechanism used for the administration and update of graphical objects. Finally, we will describe the different tools offered to the user to help him control this information.

In this thesis, we also describe the most important techniques used by the system and the way those techniques were implemented. The organization of the environment for this application is also discussed. The last part of this document provides real examples that will convince the reader that this is an essential scheduling tool for the airline business.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES ANNEXES

INTRODUCTION

Les deux dernières décennies ont été marquées par une augmentation considérable du trafic aérien commercial. La tâche des planificateurs qui doivent bâtir des horaires pour les avions et le personnel de bord est donc devenue de plus en plus complexe. Que ce soit dans le domaine du transport aérien, du transport routier ou tout simplement pour les employés d'une grande entreprise, la création d'horaires optimaux a toujours fait l'objet d'intenses recherches. L'objectif est de produire un horaire valide au coût le plus avantageux possible qui respecte un ensemble de contraintes, de règles et de conventions.

Le sujet qui nous intéresse ici, est celui de la fabrication d'horaires d'équipages des grandes compagnies aériennes. Le problème principal consiste à générer ce qu'on appelle des rotations d'équipages de façon à ce que tous les vols planifiés par une compagnie pour une période donnée soient couverts par un équipage, tout en minimisant les coûts d'exploitation. Plusieurs types d'optimiseurs ont déjà été conçus afin de résoudre ce genre de problème et ainsi générer des solutions optimales. Mais, le problème qui demeure est celui d'arriver à offrir à l'usager une méthode interactive pour la préparation des données, la définition des contraintes, l'optimisation ainsi que la manipulation des solutions en lui offrant une plus grande souplesse et une rapidité d'exécution.

Dans le premier chapitre nous présenterons les différents avantages de la conception d'une interface graphique qui fournira à l'usager une représentation graphique claire de ses horaires d'équipages, tout en lui offrant la possibilité d'y apporter les modifications nécessaires. Nous discuterons aussi du rôle important que joue l'optimiseur dans la manipulation d'une solution.

Dans le deuxième chapitre, nous procéderons à l'analyse et à la mise en place d'un environnement de travail permettant l'exécution d'une l'application modulaire qui

sera robuste. Les données sont d'une grande importance. Nous étudierons donc des techniques pour assurer l'accessibilité, la validation, la mise à jour et le partage des données par tous les usagers. La possibilité de retourner sur ses pas dans les étapes de modifications d'une solution est une option nouvelle et très utile. Elle permet de retourner à des états antérieurs de la solution et offre la possibilité de comparer et de vérifier ces solutions.

Dans le troisième chapitre nous nous concentrerons sur l'aspect technique et graphique de l'interface utilisateur. On y décrit en détail les méthodes utilisées et leurs implantations. Ce sont principalement des opérations manuelles graphiques et des méthodes d'affichage et de traitement des données entre modules.

Le dernier chapitre présentera plusieurs exemples d'utilisation réelle de l'application. Nous y discuterons les outils de sélection (filtrage) et les moyens utilisés pour le traitement des mises en place. On y démontrera aussi l'utilité de l'interface graphique et son importance pour les planificateurs dans le domaine aérien.

CHAPITRE I:

NÉCESSITÉ D'UNE APPLICATION MULTI-FENÊTRÉE POUR LA GESTION D'HORAIRES AÉRIENS

Nous avons constaté que la planification dans le domaine aérien devient une tâche de plus en plus complexe et délicate qui demande beaucoup de précision. D'où l'urgente importance d'offrir aux planificateurs un outil pouvant simplifier leur travail.

Une option qui nous semble intéressante est l'interface graphique qui peut afficher les horaires de vol et les rotations d'équipages de façon claire et dynamique. Cette constatation nous a mené à la conception d'un logiciel qui utilise le graphisme interactif tout en permettant une communication plus facile entre l'usager et l'ordinateur. En effet, l'interface permet de visionner les rotations d'équipages et les résultats d'une solution générée par un optimiseur intégré dans l'application, appelé GENCOL.

Ce chapitre est consacré à la présentation du problème de la planification dans le domaine aérien et du rôle de l'optimiseur dans la résolution de ce type de problème. Nous donnerons aussi une brève description de l'interface graphique et des avantages qu'elle offre à l'utilisateur.

1.1 Le problème de la planification dans le domaine aérien

Dans le contexte des compagnies aériennes, la procédure de planification et de fabrication d'itinéraires se fait en plusieurs étapes plus ou moins indépendantes les unes des autres:

- construction d'horaires des vols;
- routage d'avions;
- construction de rotations d'équipages;

- création des horaires mensuels personnalisés du personnel.

Chaque étape présente plusieurs types de contraintes telles les conventions collectives et les règlements gouvernementaux. Il faut donc faire en sorte que ces contraintes soient respectées et que le coût de la solution reste minimal.

1.1.1 Terminologie

Afin d'éviter tout problème de terminologie, définissons au préalable les termes importants utilisés fréquemment dans le domaine aérien [12]:

- Un **segment de vol** (tronçon de vol) est une portion d'un vol sans escale. Il constitue l'élément de base pour la construction des rotations.
- Une **rotation d'équipage** est une suite alternée de segments de vol, de périodes d'attente et de repos entre un départ et un retour à la base d'assignation du personnel.
- Une **rotation d'avion** désigne une séquence de segments de vol et d'entretiens qui sont assignés à un appareil spécifique.
- Une **station** est un aéroport. Cette station peut aussi être le nom d'une ville où on retrouve plusieurs aéroports.
- Une **base** est une station où sont assignés les membres d'équipage. Une compagnie peut avoir plusieurs bases mais un équipage ne peut être assigné qu'à une seule base, qui correspond généralement au lieu de résidence.
- Un **service de vol** est une séquence de segments séparés par des connexions qui sont effectuées par le même équipage. Cette séquence est considérée comme une journée de travail.
- Une **connexion** est une période entre deux segments consécutifs à l'intérieur d'un même service de vol où l'équipage est au sol. Une connexion peut inclure un déplacement par navette.

- Un **repos** est une période entre deux services de vol durant laquelle l'équipage devra se rendre à un hôtel.

- Une **flotte** est l'ensemble de certains types d'appareils appartenant à une compagnie aérienne. Une compagnie peut posséder plusieurs flottes qui elles, contiennent différents types d'appareils. Par exemple, une flotte peut contenir des 747, 757, 727 et des DC8.

1.1.2 Description des étapes de la planification

La première étape dans la planification d'un horaire de vol consiste à construire un horaire composé de tous les segments de vol qui doivent être effectués dans une période donnée. Des facteurs comme les tranches de temps que la compagnie possède dans différents aéroports, la demande commerciale prévue par le service du marketing ainsi que la compétition, jouent un rôle important dans cette étape.

Lorsqu'elle a déterminé la liste des vols pour une période choisie, la compagnie doit ensuite déterminer le routage de ses avions. Des rotations sont construites en prenant en considération la flotte disponible et la capacité des appareils, ainsi que l'autonomie de vol (rayon d'action) et la vitesse de chaque appareil. Les segments de vols sont regroupés par type de flotte. En d'autres termes, chaque tronçon est assigné à un seul type d'avion. L'objectif est de générer des séquences de vols pour tous les appareils de la flotte, en s'assurant que tous les segments ont été couverts et ceci de façon à obtenir le bénéfice maximum.

Puisque notre intérêt s'applique spécifiquement au domaine des rotations d'équipages, l'interface conçue pour ce mémoire n'inclut pas les manipulations graphiques des routes d'avions. Par contre, comme cette étape est indispensable à la construction des rotations d'équipages, un générateur de rotations d'avions a été intégré au système. Celui-ci permet l'affichage des routes d'avions, mais ne permet pas la manipulation de ces données par l'usager.

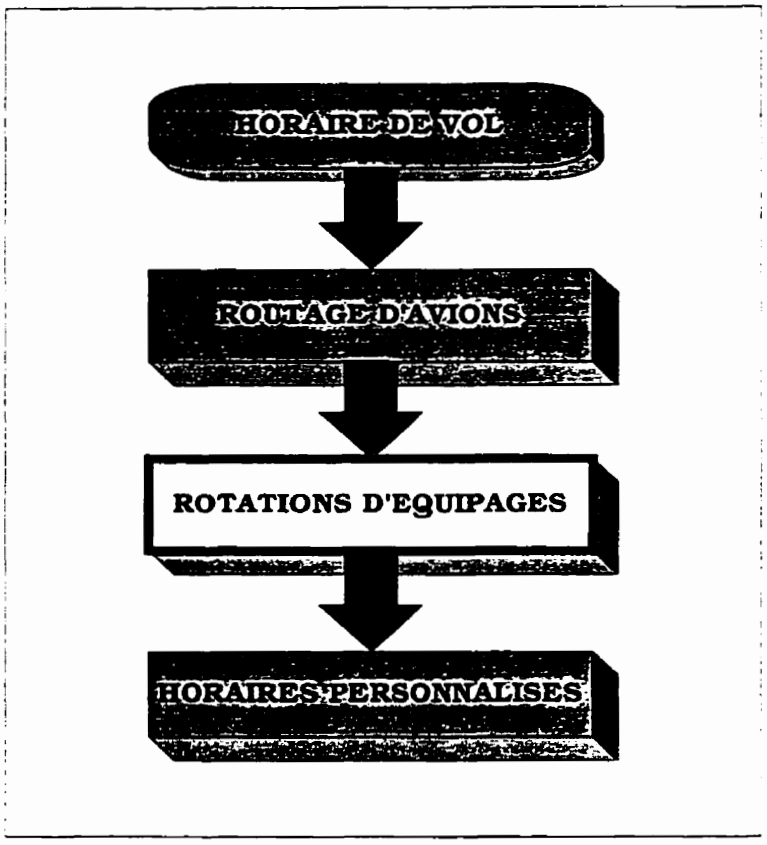


Figure 1.1 Procédure de planification

Une fois le routage des avions déterminé, les rotations d'équipages doivent être générées. Elles sont définies comme suit: le planificateur doit construire, au meilleur coût possible, un ensemble de rotations d'équipages (itinéraires d'équipages) qui couvre tous les tronçons de l'horizon fixé seulement une fois et ce, pour éviter la surcouverture. Dans le cas qui nous concerne, tous les vols appartenant à une flotte donnée sont traités indépendamment des autres flottes. Notons que la compagnie peut disposer de plusieurs flottes dont chacune nécessite un nombre différent de membres d'équipages.

Le planificateur veille ensuite à ce que les rotations soient valides. La difficulté se trouve dans la complexité des règles gouvernementales et des conventions collectives

spécifiques à la catégorie de personnel navigant. Tous ces règlements doivent être respectés au moment où les rotations sont construites. Parmi les différentes contraintes, on retrouve:

- Les conventions collectives: incluant les temps à respecter pour le début et la fin du service de vol (journée de travail) et le temps maximal du service. Notons que ces règles peuvent varier d'une flotte à l'autre.
- Les règlements gouvernementaux: un ensemble de règlements imposés par le gouvernement pour assurer que l'environnement de travail des équipages est sain et sans risque. Nous avons, par exemple, le respect du temps minimal de repos entre les journées de travail.
- Les contraintes d'opération: parmi les facteurs à considérer, on retrouve les temps de connexions imposés par station ou par région entre deux vols consécutifs à l'intérieur d'un service de vol, le temps de transport entre deux aéroports d'une même ville et finalement les bases auxquelles les équipages doivent en principe retourner.

La dernière étape effectuée par la compagnie consiste à assigner à tous ses employés des séquences de rotations alternées par des périodes de vacances et dans certains cas des périodes de stage. Ces horaires mensuels sont bâtis en utilisant l'une des deux méthodes suivantes: par enchères, c'est-à-dire en considérant la préférence du personnel avant de leur assigner les blocs (rotations); ou par séquences de blocs légaux qui sont assignés aux employés en respectant leur séniorité.

Comme le problème de construction des rotations d'équipages est très complexe en lui-même, l'interface développée dans ce mémoire ne traitera pas de l'étape d'assignation des horaires personnalisés. Par contre, ce complément pourrait être accompli en ajoutant une série de fonctionnalités dans l'interface pour qu'il traite les séquences de blocs assignées au personnel.

1.1.3 Information supportée par le système de construction des rotations d'équipage

Après avoir déterminé un horizon pour sa solution (un mois dans la plupart des cas), le planificateur procède à la construction d'un ensemble de services de vols valides ayant le meilleur coût possible. Pour résoudre son problème, le planificateur doit posséder les informations suivantes:

- la durée de la saison ou l'horizon du problème;
- le type de flotte pour laquelle on construit les rotations;
- la liste des vols de la flotte à couvrir ainsi que l'itinéraire des avions sur ces vols;
- la liste des aéroports et des données qui leur sont rattachées tels les coûts d'hôtel et de transport, les temps de connexions des stations, les moments où se produisent les changements de l'heure d'été (avancée) et de l'heure d'hiver (normale), etc.;
- la liste des mises en place internes et externes;
- la liste des contraintes ou des paramètres.

Le planificateur doit ensuite juxtaposer les différents services de vol afin de construire des rotations (fig. 1.2). Ces rotations doivent, elles aussi, respecter les conditions de validité et de coût. Le planificateur doit s'assurer que tous les tronçons de la période choisie sont couverts une seule fois. S'il a besoin de faire voyager un équipage sur un vol déjà affecté à un autre équipage, les membres du second équipage feront donc le tronçon en mise en place interne, c'est-à-dire qu'ils voyageront en tant que passagers.

Le planificateur peut même parfois se trouver devant une situation où l'équipage, après avoir effectué plusieurs vols, doit retourner à la base mais aucun vol n'est disponible pour couvrir ce segment et aucune mise en place interne n'est possible. Dans ce cas, les membres de l'équipage devront prendre place à bord d'un appareil d'une autre

compagnie aérienne (mise en place externe) comme passagers d'un vol commercial. Les coûts de ce voyage seront ajoutés aux coûts de la rotation.

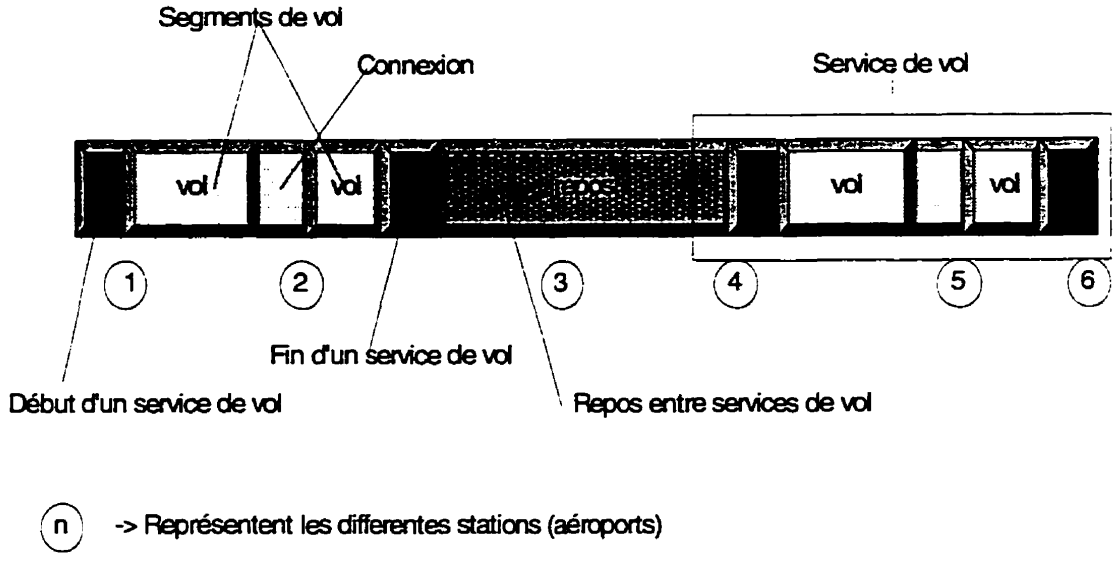


Figure 1.2 Exemple d'une rotation d'équipage

L'interface que nous avons conçu pour ce mémoire devait fournir un outil de travail efficace aidant à générer des rotations d'équipages de façon économique, rapide et productive. Nous avons répondu à ce besoin et donc simplifié la tâche du planificateur qui était devenu de plus en plus complexe.

1.2 Système de prise de décision

La production de solutions de qualité est essentielle dans l'industrie aérienne où la compétition est très serrée et les faibles marges de profit font norme. Par contre, l'automatisation des procédures ne garantit pas toujours des horaires de qualité supérieure. C'est pourquoi les méthodes d'optimisation mathématiques ont été introduites

dans la majorité des systèmes de planification des horaires aériens, afin d'assurer les profits les plus larges possibles.

Le logiciel d'optimisation GENCOL, supporté par l'interface, a permis de faire une percée dans le domaine de fabrication d'horaires d'équipages aériens. La théorie à la base de ce logiciel est présentée par Abbara [1], Desrosiers et al [7, 8]. En effet, c'est le premier logiciel capable de résoudre de façon optimale des problèmes de grande taille. Cette nouveauté entraînera, de la part des compagnies aériennes, une augmentation des exigences en ce qui a trait aux résultats escomptés.

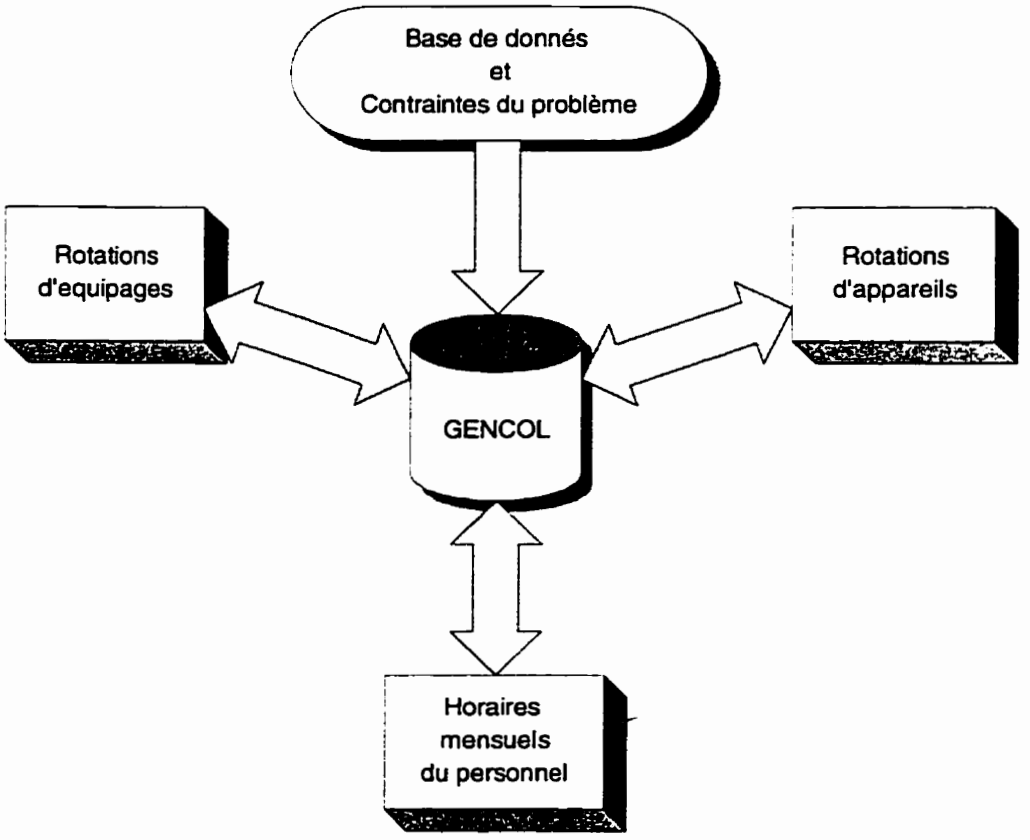


Figure 1.3 Les applications GENCOL

Le nom du logiciel d'optimisation vient de l'expression «GENération de COLonnes» et fait référence à l'approche mathématique dans ce domaine. Pour ce qui est du transport aérien, GENCOL (fig. 1.3) peut servir à la génération d'horaires de travail pour le personnel navigant: rotations d'équipages et horaires de travail personnalisés ou non. Ce logiciel peut aussi servir à l'allocation des avions aux différents segments de vols. Notons que chacune de ces applications comporte certaines adaptations quant à la modélisation du problème. Chaque problème a ses propres données d'entrée et ses propres contraintes qui nécessitent différents ajustements avant l'exécution du logiciel d'optimisation.

1.2.1 Présentation de l'optimiseur et de son comportement externe

Comme nous l'avons déjà mentionné, le coeur de l'interface est composé d'un optimiseur. La première étape consiste à traduire les contraintes en un ensemble de paramètres séparés en différentes sections qui représentent les différents aspects du problème. Parmi ceux-ci, on retrouve les paramètres définissant la longueur de la période sur laquelle s'étend notre solution, ceux qui déterminent les temps de connexions minimaux et maximaux à respecter entre les tronçons de vols, et bien d'autres.

Le temps d'exécution consommé par l'optimiseur pour résoudre un problème dépend généralement de la taille et de la quantité des données d'entrée. Par exemple dans le cas de construction de rotations d'équipages, le nombre de segments de vols à couvrir influencera le temps que l'optimiseur prendra. Il est donc suggéré d'effectuer une certaine sélection sur les tronçons de vols en définissant certains critères pour filtrer seulement les vols qu'on désire couvrir, en opérant une optimisation à la fois. De cette façon, on peut diminuer l'étendu du problème selon les besoins et les préférences de l'usager en plus de réduire le temps de résolution.

Après la sélection des données et la spécification des paramètres à utiliser, l'optimiseur commence par générer les mises en place internes et externes. Il regroupe les

mise en place par séquences en vue de pouvoir les utiliser pour compléter les rotations des équipages qui seront transportés d'une station à l'autre en tant que passagers, afin d'assurer leur retour à leur base. Cette étape est assez importante et doit être exécutée au moins une fois. Par la suite, elle doit être effectuée chaque fois que les données commerciales ou locales changent afin d'assurer la présence et la validité des candidatures de mise en place. Toutes ces mises en place sont gardées de côté et ne sont utilisées par l'optimiseur qu'au besoin et de la façon la plus profitable.

Le recouvrement des segments de vol peut se faire en trois modes: quotidien, hebdomadaire ou en mode fenêtre. En mode quotidien, le système tente de générer les rotations qui peuvent se répéter quotidiennement avec les mêmes séquences de vols. En mode hebdomadaire, l'optimiseur essaie de fabriquer des rotations qui se répéteront à toutes les semaines. Finalement, le mode fenêtre consiste à générer les rotations dans une fenêtre de temps spécifiée par l'usager. Notons que les trois modes de recouvrement peuvent être exécutés séparément ou de façon séquentielle, en commençant par le mode quotidien.

1.2.2 Formulation mathématique

Tous les types de problèmes, qu'ils soient les assignations d'avions, les rotations d'équipages ou les horaires d'équipages personnalisées, consistent à couvrir un ensemble d'objets avec des routes réalisables et à un coût minimal. Ces problèmes possèdent tous une structure mathématique commune pouvant être formulée comme un problème de recouvrement. Ainsi, la formulation mathématique du problème de recouvrement peut être définie comme suit:

$$\text{Minimiser: } \sum_{j=1}^{n} c_j . x_j$$

$$\text{Tel que: } \sum_{j=1}^{n} a_{ij} . x_j \geq 1 \qquad i = 1, \ldots\ldots, m$$

$$x_j \in \{0,1\} \qquad j = 1, \ldots\ldots, n$$

où:

m = nombre d'objets à couvrir,

n = nombre de rotations,

$$a_{ij} = \begin{cases} 1 \text{ si l'objet } i \text{ fait partie de la rotation j,} \\ 0 \text{ autrement,} \end{cases}$$

c_j = coût de la rotation j,

$$x_j = \begin{cases} 1 \text{ si la rotation } j \text{ est selectionnee dans la solution courante,} \\ 0 \text{ autrement.} \end{cases}$$

Dans le problème d'assignation des appareils aux segments de vol, les variables x_j déterminent le meilleur routage des avions. Pour le problème des équipages, la solution offerte par les variables x_j correspondra à l'horaire du personnel qui offrira le coût le moins élevé, etc.

Pour plus de détails sur l'adaptation de cette formulation à différentes applications et les techniques de résolution, le lecteur pourra se référer à [11], [2] et [3].

1.2.3 Quelques exemples réels

La procédure de génération de colonnes décrite précédemment a été implémentée sur de nombreux problèmes réels. Par exemple, le temps d'exécution ou de calcul nécessaire à l'ordinateur pour résoudre un problème de rotation comprenant 1000 segments de vol à couvrir peut varier entre 10 minutes et 1 heure. Dans notre cas les tests ont été exécutés sur une station de travail Unix de marque HP735. Pour illustrer la performance de la procédure par génération de colonnes dans le domaine de construction de rotations d'équipages, nous présentons dans le tableau 1.1, les résultats des tests qui ont été menés chez d'importantes compagnies aériennes.

- Chez AIR FRANCE (AF) les coûts de personnel ont été réduits en moyenne de 6.1% sur une dizaine de problèmes hebdomadaires ayant de 300 à 1000 segments de vol.

- Chez SCANDINAVIAN AIRLINES (SAS) les coûts ont été réduits de 9.1% sur le problème hebdomadaire de 3582 vols, regroupant tous les vols moyen et court courrier.

- Chez ALITATLIA (AI) les coûts ont été réduits de 18.2% relativement à un horaire manuel d'experts d'un problème hebdomadaire de 1035 vols, qui correspond à la plus grande flotte de la compagnie.

Tableau 1.1 Résultats de tests menés auprès de compagnies aériennes

Compagnie et nombre de segments à couvrir	**Réduction des coûts par rapport aux solutions manuelles des planificateurs.**
AF: 300-1000 tronçons de vol	6.1%
SAS: 3582 tronçons de vol	9.1%
AI: 1035 tronçons de vol	18.2%

1.3 L'interface graphique utilisateur

On définit le graphisme par ordinateur comme la création, l'emmagasinage et la manipulation d'un modèle et de son image, le modèle simulant aussi fidèlement que possible une situation réelle ou imaginaire. Le graphisme interactif par ordinateur est un exemple spécifique où l'usager contrôle dynamiquement le contenu et le format de l'image (positionnement des objets) par l'entremise d'outils.

Une interface graphique est un logiciel (programme) utilisant le graphisme interactif pour faciliter la communication entre l'usager et l'ordinateur. La plupart du temps l'application est multi-fenêtrée et composée de plusieurs panneaux dont l'apparence est différente mais dont le comportement interactif est standardisé partout dans l'application. La tâche des différents panneaux est d'accomplir toutes les fonctionnalités utiles à l'usager pour qu'il puisse atteindre son objectif.

La conception de l'interface doit tenir compte de trois notions importantes: l'apparence, le comportement et la fonctionnalité.

1.3.1 L'apparence de l'interface

La première étape lors du développement de l'interface consiste à déterminer son apparence. Doit ensuite suivre une étude approfondie de la présentation graphique du modèle et des objets. Les programmeurs doivent garder en tête que le niveau de complexité de visualisation et d'affichage de leur outil doit rester minimal. Pour ce qui est de l'esthétisme de l'interface, le talent artistique dés programmeurs peut être un atout très utile. (voir [14])

1.3.2 Le comportement de l'interface

Le logiciel doit être conçu de façon à ce qu'un usager éventuel n'ait pas besoin d'une grande expertise informatique pour l'opérer et tirer le maximum de ses

performances. De là vient le terme convivial (*user-friendly*). L'accès aux différents panneaux de l'application se fait par les menus, les boutons poussoirs, les menus déroulants et les listes à sélection. Dans la conception d'interface, le menu est un des critères d'enchaînement et de dynamique de dialogue (pour plus de détails sur les éléments d'une interface graphique, se référer à [15], [4] et [9]. Un menu comprend:

- Un titre appelé rubrique, présentant une tâche, un objet ou une fonctionnalité générique;
- Une liste d'options qui sont en fait toutes les fonctions spécifiques à cette tâche ou variables de l'objet qui peuvent être des sous-tâches de plusieurs catégories possibles. C'est-à-dire:
  - exécution de procédure (préparation, modification, etc.);
  - présentation, mise en forme;
  - recherche d'informations.

La sélection d'une option en utilisant la souris provoque l'ouverture d'un autre menu, d'une fenêtre (panneau), d'une boîte de dialogue ou le déclenchement d'une fonctionnalité.

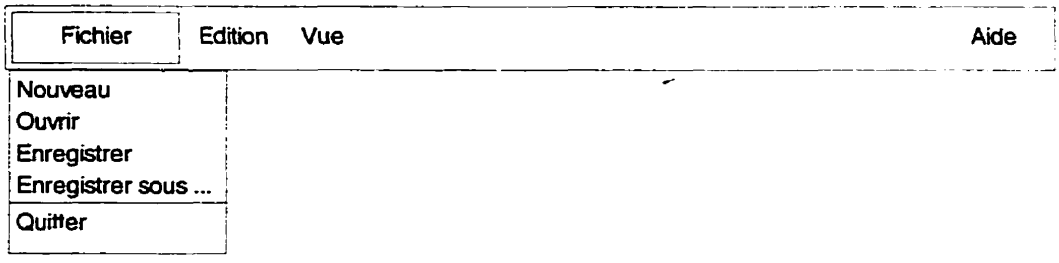


Figure 1.4 Exemple de menu déroulant

Selon Hopgood [10], dans une application multi-fenêtrée on dispose des différents type de menus suivants:

- Le menu déroulant; un menu déroulant est composé d'une rubrique et d'options, accessible à partir d'une barre de menu (fig. 1.4).
- Le menu en cascade; un menu cascade ne s'ouvre qu'à partir d'une des options d'un menu déroulant. Les options d'un menu cascade correspondent à un choix limité de sous-tâches ou d'actions.
- Le menu de fond (*popup*); les options du menu de fond appelé aussi "menu lié au contexte", ne sont pas affichées en permanence sur l'écran. Elles sont appelées à la demande de l'utilisateur pour lancer des actions spécifiques au contexte.

1.3.3 La fonctionnalité de l'interface

Une fonctionnalité est définie comme une action qui agit sur les données. Celles-ci peuvent être des fichiers, des processus ou même des objets graphiques.

Lorsqu'ils sont activés par la souris, tous les éléments de l'interface cités précédemment déclenchent des actions telles la modification de données, le lancement de processus, la modification de fichiers (lecture et écriture) et la manipulation graphique. Dans le cas d'une manipulation graphique, qu'on appelle aussi une action en interaction directe, le positionnement graphique des objets reflète immédiatement les données en mémoire.

L'ensemble de toutes les fonctionnalités offertes par l'interface (ou l'application) doit offrir à l'usager toutes les possibilités de traitement et de gérance de ses données pour qu'il puisse atteindre l'objectif qu'il s'est fixé. Par exemple, un dialogue peut demander à l'usager d'entrer des informations dans des champs de texte spécifiques. Des menus de fond peuvent offrir des détails concernant les objets graphiques. Une liste de

sélection permettra aussi à l'usager de visualiser, de trier, de filtrer et même d'éditer ses données.

1.3.4 Gestion et manipulation d'horaires d'équipages aériens

Notre objectif principal est de procurer à l'usager dans le domaine aérien un logiciel qui pourra l'aider dans ses tâches de planification. En premier lieu, cet outil doit permettre d'afficher les rotations d'équipages d'une façon claire et dynamique. Il doit aussi offrir une série de possibilités de manipulations graphiques qui provoqueront des modifications ou des ajustements à la solution produite par l'optimiseur, lorsque nécessaire. Nous expliquerons plus en détails les manipulations graphiques, un peu plus loin dans ce mémoire.

Par défaut, l'interface affichera dans le panneau graphique principal toute la période correspondant à la durée du scénario choisi. Un mécanisme d'effet de loupe (*zoom*) est disponible afin que l'usager puisse sélectionner une section particulière de cette période.

De plus, la complexité de visualisation et la quantité d'information pouvant être affichée en même temps à l'écran exigent une présentation graphique comportant plusieurs options d'affichage que l'usager pourra sélectionner lui-même.

L'interface dispose aussi de plusieurs panneaux qui servent à modifier les données d'entrée de l'application (application multi-fenêtrée [10]). Chaque panneau d'édition correspond à un type de données spécifiques parmi lesquels on peut citer: les vols, les stations, les paramètres, etc. Ce genre de panneau permet d'éditer, de modifier et de sauvegarder les données dans un fichier d'entrée. Toutes ces données sont automatiquement chargées en mémoire afin de permettre la construction manuelle des rotations d'équipages. Donc, un changement effectué dans un panneau affectera le contenu de la mémoire, du fichier d'entrée et parfois même des objets graphiques. Il est

donc nécessaire de mettre en place un système complexe de mise à jour interne qui s'active après chaque modification (genre de base de données interactive).

Un autre aspect unique du logiciel est l'enregistrement de toutes les modifications effectuées dans l'interface sur tous genres de données. Cette fonctionnalité ajoute, bien sûr à la complexité du logiciel et nécessite donc plus de développement et de ressources. Par contre, elle est pratiquement indispensable. En effet, à chaque procédure de sauvegarde, on enregistre l'état de la solution, ses statistiques et ses coûts. Ces données sont disponibles dans le panneau de gestion des instantanés (*snapshots*) (fig. 1.5).

Cet outil permet au planificateur de comparer les coûts de la solution pour son problème courant, avec une des solutions précédentes du même problème. Si la solution précédente semble plus convenable, le planificateur peut ainsi revenir sur ses pas avec un simple «pointez et cliquez». Une marche arrière dans l'évolution du problème est donc possible. Nous discuterons cette fonctionnalité plus en détails dans le chapitre suivant.

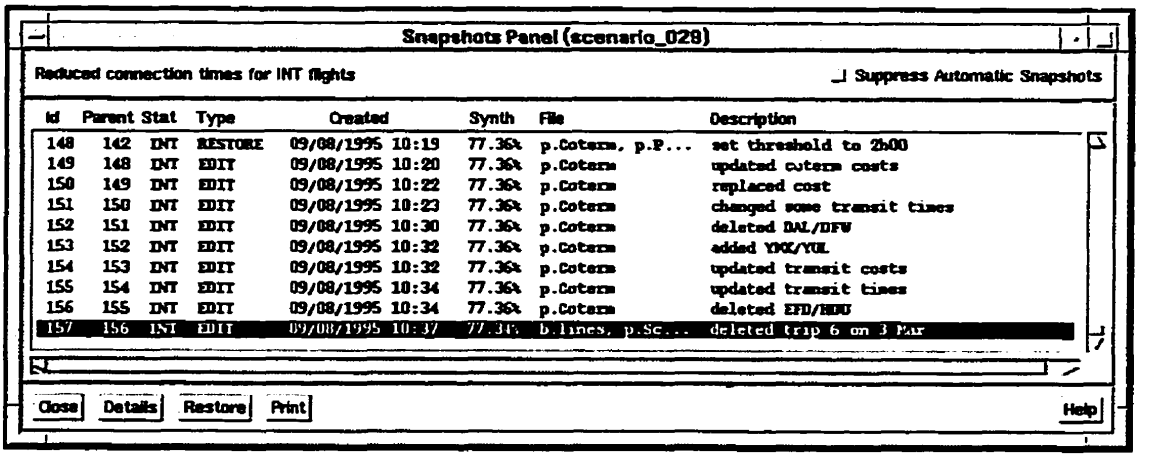


Figure 1.5 Panneau de gestion des instantanés (*Snapshots*)

Afin d'offrir des informations supplémentaires sur chaque modification effectuée lors des procédures de sauvegarde, nous avons mis en place un système qui enregistre toutes les données modifiées dans un fichier et détaille exactement les changements

produits sur les éléments donnés. Le planificateur peut consulter ces données grâce au panneau de gestion *Audit Trail* (fig. 1.6), qui retrace le chemin parcouru tout au cours de l'optimisation du problème.

Notre plus grand défi est d'offrir toutes ces fonctionnalités et ces avantages à l'usager sans toutefois affecter la robustesse et la souplesse du logiciel. La grande quantité d'objets qui sont présentés dans le panneau d'interface graphique demande une recherche constante de techniques optimales de manipulation des données. Dans le cas, par exemple, où une modification générale affecterait un grand nombre d'objets graphiques, nous visons à accélérer le plus possible le temps d'exécution de la tâche.

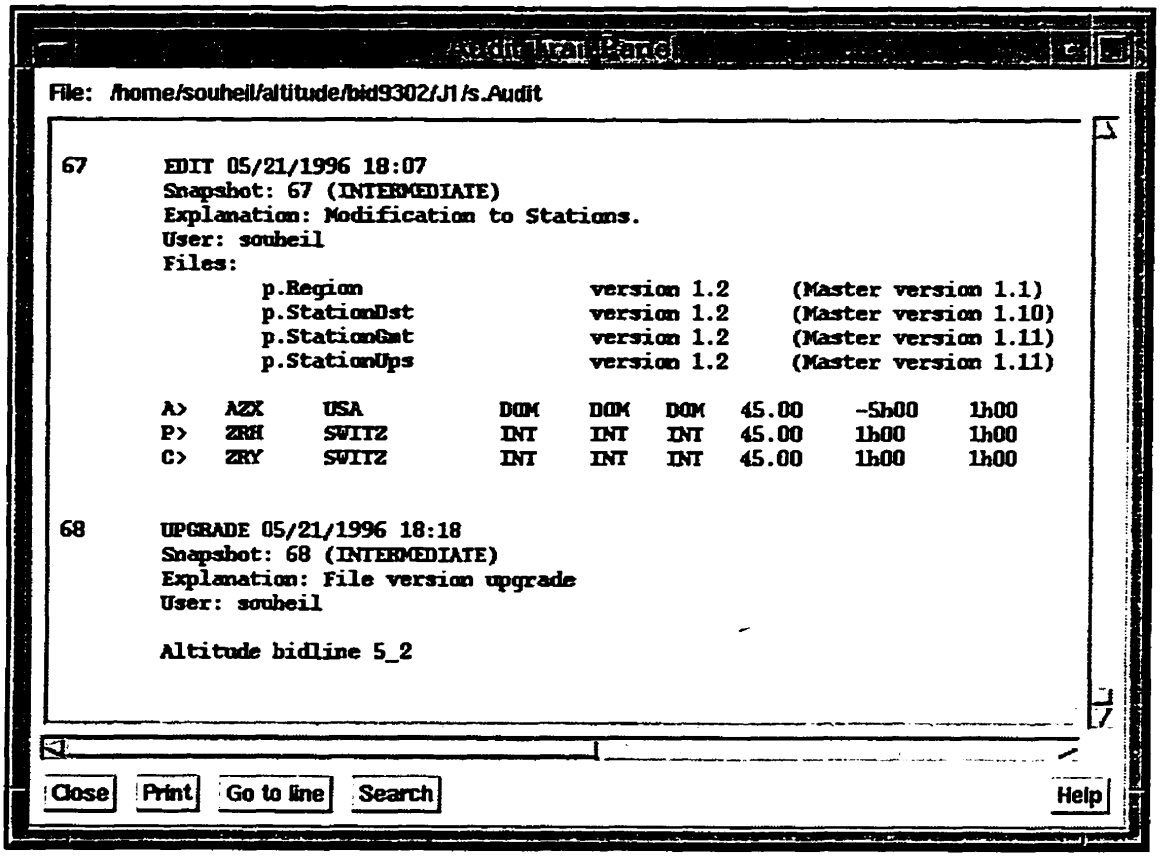


Figure 1.6 Panneau *Audit Trail*

Des techniques optimales sont utilisées pour collecter toutes les informations concernant les modifications. Ces informations sont ensuite gardées en mémoire dans un tampon interne, jusqu'à ce que l'usager décide de sauvegarder sa solution. Un autre point important est de garder la convivialité du logiciel malgré la complexité du problème. En d'autres mots, ce logiciel doit permettre à un usager non spécialisé dans le domaine aérien de manipuler et de maîtriser l'interface facilement pour pouvoir tirer le maximum de l'optimiseur et ainsi garantir une solution optimale du problème.

CHAPITRE II:

DESCRIPTION DE L'ENVIRONNEMENT DE L'INTERFACE

Le travail du planificateur pour produire une solution est divisée en plusieurs étapes. Il doit d'abord créer ou installer un scénario à partir d'un problème pré-défini. Pour ce faire, il spécifie un des contextes parmi ceux qui ont été définis par le superviseur, dans le répertoire maître. Ensuite, les fichiers de données d'entrée sont préparés ou modifiés à l'aide des panneaux d'éditions offerts par l'interface. Une fois la préparation des données terminée, une alternance entre l'exécution de l'optimiseur sera effectuée grâce à un panneau spécialisé de l'interface et des modifications manuelles graphiques faites dans le panneau principal. Les solutions produites dans différents scénarios sont ensuite évaluées selon les coûts correspondants, puis sont rangées par ordre de profit et de coûts, dans le répertoire maître du superviseur.

Le scénario le plus satisfaisant est ensuite sélectionné. Une copie du scénario est faite pour permettre au superviseur de visualiser et consulter la solution, et pour la rendre disponible aux autres planificateurs.

2.1 Organisation des données sur disque

Deux ensembles de fichiers sont utilisés par le système. Le premier ensemble contient les fichiers maîtres qui sont installés et gardés à jour par les superviseurs. Les planificateurs peuvent accéder à ces fichiers en lecture seulement. Ils ne peuvent donc pas les modifier.

Le deuxième ensemble contient les fichiers locaux qui constituent une copie des fichiers maîtres et qui se situent à l'intérieur du scénario créé par le planificateur. Ces fichiers locaux sont fréquemment modifiés dans le processus de résolution du problème.

2.1.1 Le concept du scénario et de la période maître de planification

Un **scénario** est composé de toutes les données associées avec un problème de génération de rotations d'équipages. Les informations sont contenues dans des fichiers, et tous les fichiers correspondant au même scénario sont localisés dans un seul répertoire. Le scénario est identifié par le répertoire, et le nom du scénario est identique au nom du répertoire.

Les fichiers maîtres sont installés dans un répertoire global appelé répertoire maître, accessible à tous les usagers. L'information contenue dans les fichiers maîtres varie selon la période de planification et le calendrier déterminé pour un problème spécifique. C'est pourquoi le répertoire maître comprend plusieurs sous-répertoires définissant plusieurs périodes de planification qu'on appelle *bid period directories* (voir [5]).

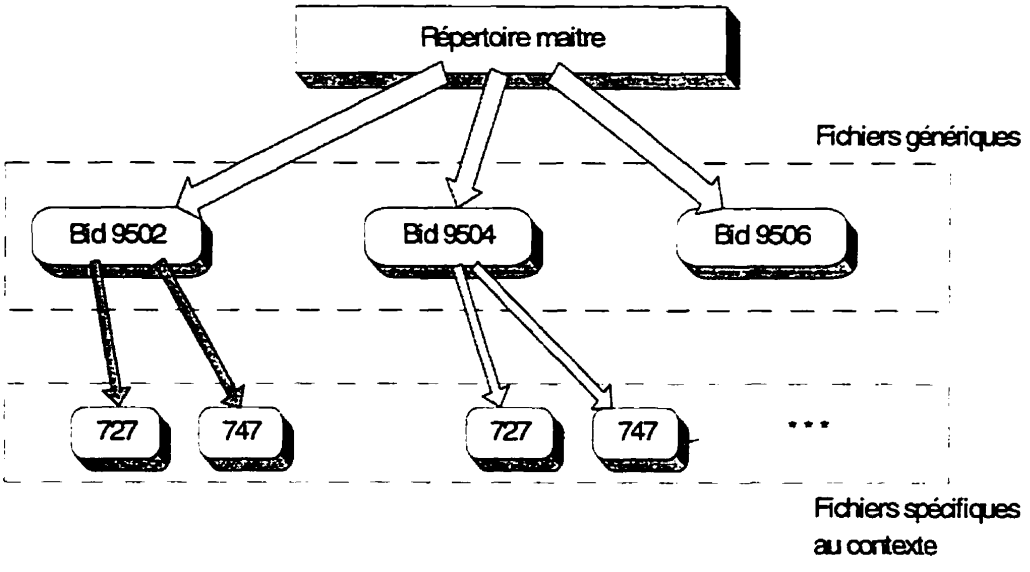


Figure 2.1 Architecture des répertoires maîtres

Généralement, à l'intérieur de chaque répertoire de période de travail on retrouvera certains fichiers dont les données correspondent à un type de contexte déterminé (c.-à.-d.: type d'avions) pour lequel on tente d'optimiser la solution. Ces fichiers se trouvent dans des sous-répertoires représentant tous les contextes correspondant à la période de travail. (fig. 2.1)

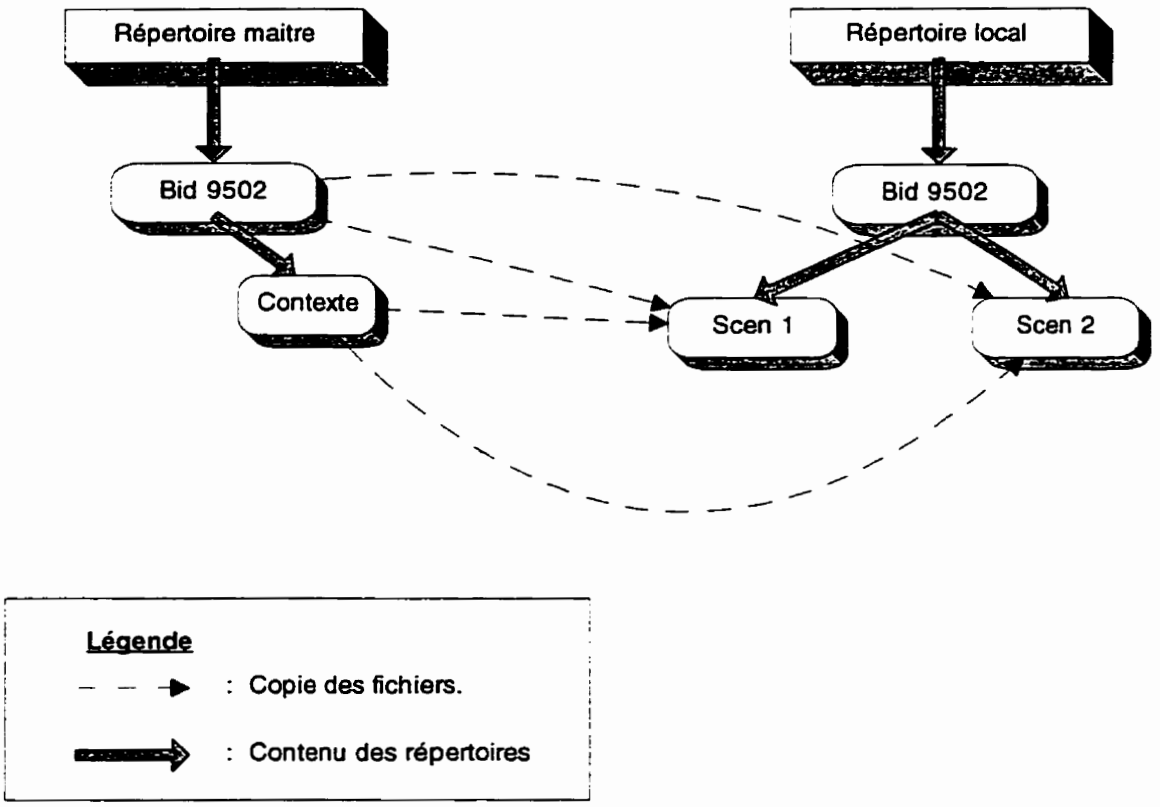


Figure 2.2 Architecture des scénarios locaux

Quand un usager désire travailler avec une période maître, il doit créer un scénario dans un répertoire local de travail. Tous les fichiers maîtres de la période spécifiée sont alors copiés dans le répertoire local. Les fichiers correspondant au contexte spécifié par

l'usager sont également copiés dans ce répertoire local. Il faut noter que l'usager ou le planificateur doit spécifier le contexte pour lequel il désir trouver une solution, lors de création du scénario. (fig. 2.2)

On retrouve donc, à l'intérieur du répertoire local, un scénario nommé par l'usager, qui contient un ensemble de fichiers qui proviennent de deux sources différentes dans le répertoire maître.

2.1.2 Types de fichiers scénario

Plusieurs types de fichiers sont nécessaires à la construction d'un scénario:

- fichiers de données de base requis pour la création du scénario;
- fichiers intermédiaires préparés avec les panneaux d'édition de l'interface;
- fichiers d'entrée et de sortie de l'optimiseur;
- fichiers techniques et de gestion du scénario.

Les fichiers de base requis existent au moment de la création du scénario. Ce sont les fichiers installés par le superviseur dans le répertoire maître, qui sont copiés localement dans le scénario. Ils peuvent être génériques (Générique, tableau 2.1) ou spécifiques au contexte (G/Contexte, tableau 2.1). Les données provenant des fichiers maîtres sont utilisées comme valeurs par défaut. Les données provenant des fichiers spécifiques au contexte, entrées par l'usager lors de la création du scénario, serviront à écraser (remplacer) ces valeurs au besoin.

Une fois le scénario créé, on peut procéder à la modification des fichiers d'entrée et préparer les fichiers intermédiaires. Un outil d'interface (panneau d'édition) est disponible pour chaque type de fichier afin de permettre d'ajouter des données et de modifier les données déjà existantes dans les fichiers.

Le tableau qui suit énumère les fichiers d'entrée provenant du répertoire maître:

Tableau 2.1 Liste des fichiers d'entrée d'un scénario

Type	**Nom du fichier**	**Description**	**Panneau d'interface correspondant.**
Générique	*a.AcTimeConn*	Temps de connexion pour les avions.	Panneau d'avions
Générique	*a.Fleets*	Fichier des contextes.	Panneau des contextes
Générique	p.CmlFare	Coûts des vols commerciaux.	Panneau des coûts
G/Contexte	p.Coterm	Liste des coterminaux et des codomiciles.	Panneau des coterminaux
G/Contexte	p.Param	L'ensemble de tous les paramètres du système.	Panneau des paramètres
Générique	p.Region	Liste des régions regroupant les différentes stations.	Panneau des stations
Générique	p.StationDst p.StationGmt p.StationCie	Informations concernant les stations.	Panneau des stations
G/Contexte	p.TimeConn	Temps de connexions des équipages.	Panneau de temps de connections des équipages
G/Contexte	p.TimeLayover	Temps de repos des équipages.	Panneau de temps de repos des équipages
G/Contexte	p.TmfUser	Gérance des exemplaires de rotations.	Panneau des rotations
Générique	p.SchedCml	Segments des vols commerciaux (mises en place externes).	Panneau des vols commerciaux
Générique	p.SchedLeg	Vols à couvrir de la compagnie en question.	Panneau des segments de vols
Générique	p.SchedJumpseat	Liste des mises en place internes, utilisées.	Panneau des mises en place internes

Certains fichiers sont vides lorsque le scénario est créé. Les données doivent être entrées en utilisant plusieurs panneaux de l'interface. Quelques uns de ces fichiers sont obtenus en filtrant et en sélectionnant des données provenant des fichiers d'entrée du répertoire maître ou en utilisant les modules de génération des rotations des avions (*ARG*) ou de génération des candidatures de mises en place (*DHG*).

Voici un tableau représentant les fichiers intermédiaires (initialement vides) préparés avec quelques panneaux de l'interface:

Tableau 2.2 Liste des fichiers intermédiaires

Nom du fichier	Description
p.LegDomicile	Fichier de contraintes sur les tronçons de vol.
p.SchedLegFiltered	Fichier de sélection des tronçons de vol qui doivent être couverts par l'optimiseur, (Façon de limiter la taille du problème).
p.TmfTripExport	Fichier solution préparé pour être exporté à un autre scénario.

Le tableau suivant (2.3) liste les fichiers de sortie intermédiaires générés par l'optimiseur à partir des fichiers d'entrée et des fichiers préparés par l'usager lors de la sélection de données.

Tableau 2.3 Liste des fichiers générés par l'optimiseur (première partie)

Nom du fichier	Description
a.AcRouting	Fichier des rotations des avions (appareils). Généré par *ARG*, peut servir comme entrée pour les autres modules de l'optimiseur.

Tableau 2.3 Liste des fichiers générés par l'optimiseur (deuxième partie)

Nom du fichier	Description
p.TmfDh	Fichier généré par le module DHG contenant les candidatures de mises en place qui correspondent aux tronçons de vol sélectionnés dans p.SchedLegFiltered.
p.SchedDhFiltered.dhg	Fichier généré par le module DHG contenant les candidatures de mise en place commerciales provenant du fichier p.SchedCml.
p.TmfTrip	Fichier le plus important du scénario. Il représente la solution générée par l'optimiseur couvrant tous les segments de vol formant les rotations d'équipages valides.

2.1.3 Exécution itérative de l'optimiseur avec modifications graphiques manuelles

Les processus d'optimisation sont lancés en arrière plan (*background*) grâce au panneau d'optimisation (*Solve*) (fig. 2.3). Au moment où l'exécution est entamée, un sous-répertoire est créé dans le répertoire scénario. Il est appelé *Solve* et contient tous les fichiers générés par le processus (énumérés dans le Tableau 2.3).

Pendant que l'optimisation s'exécute, une fenêtre déroulante dans le panneau d'optimisation est mise à jour toutes les 5 secondes, donnant à l'usager les dernières statistiques du processus d'optimisation. Les statistiques comprennent le nombre de segments de vol couverts, le coût de la solution atteinte, etc. Lorsque le processus est terminé, l'usager peut décider de sauvegarder sa solution en appuyant sur le bouton *Save* dans le panneau d'optimisation. Les fichiers seront alors transférés du sous-répertoire

solve au répertoire scénario. Au même moment, la solution sera affichée dans le panneau principal.

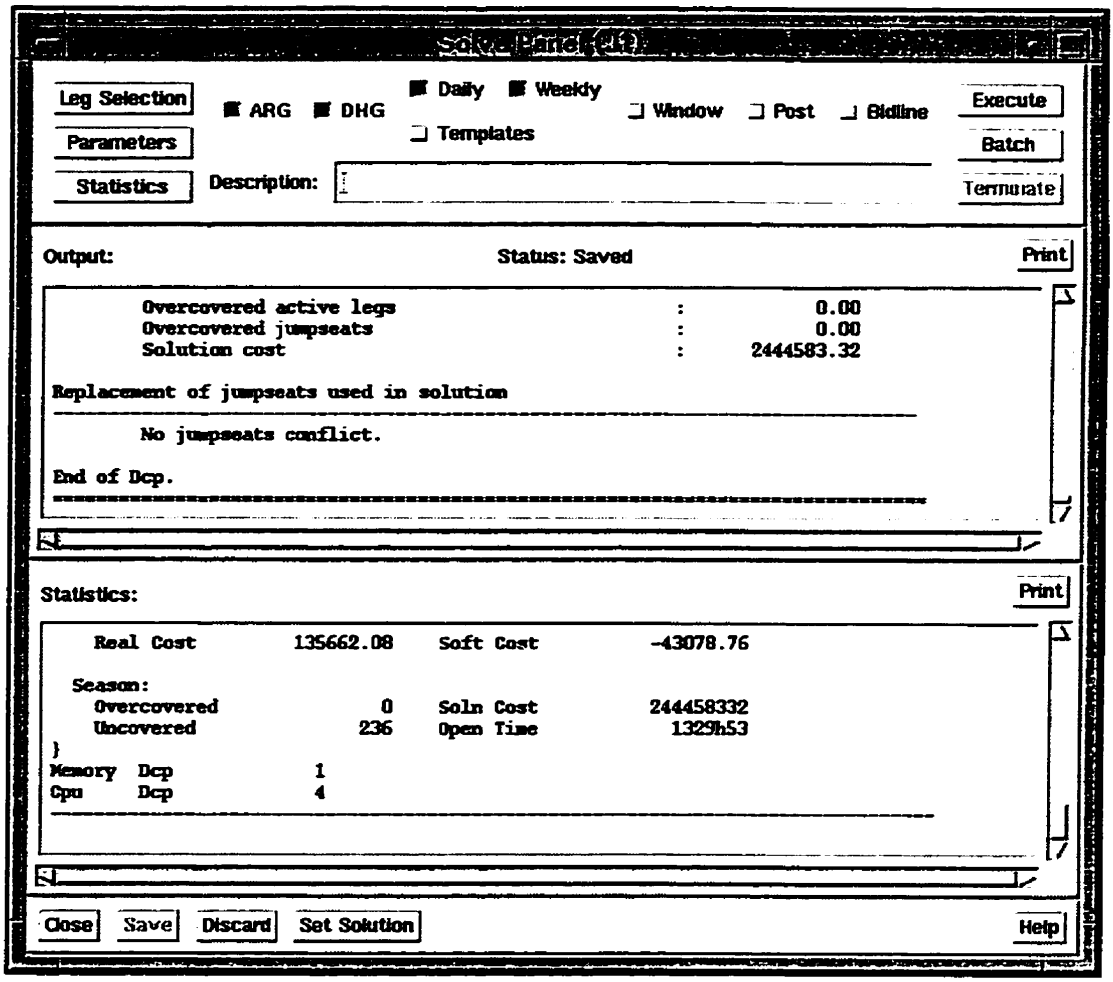


Figure 2.3 Panneau d'optimisation (*Solve*)

Une fois la solution et les rotations affichées, l'usager peut se consacrer à l'édition manuelle et à la manipulation graphique de la solution à l'aide des techniques variées que nous présenterons dans le prochain chapitre de ce mémoire. Si l'usager ne désire pas garder cette solution, il n'a qu'à appuyer sur le bouton *Discard* dans le panneau d'optimisation, ce qui détruira le répertoire *solve* (fig. 2.4).

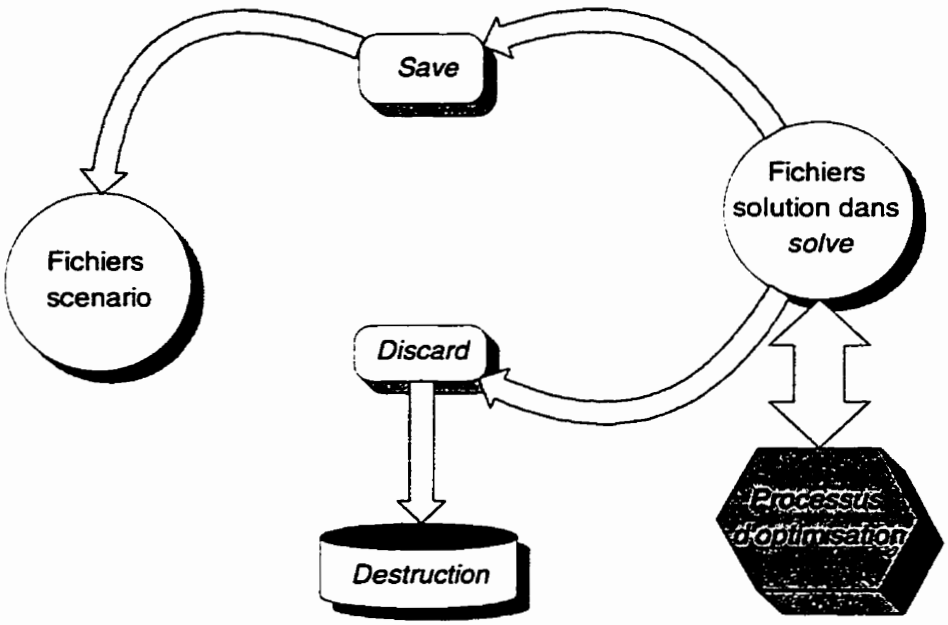


Figure 2.4 Procédure de d'optimisation dans le panneau d'optimisation

2.2 Possibilité de garder une trace des modifications

Après plusieurs itérations, plusieurs essais de modification des données, manipulations graphiques et procédures d'optimisation, l'usager arrivera à une solution satisfaisante pour la période en question. Un instantané du scénario est pris puis transféré au répertoire maître de la période en question.

Un système de contrôle des versions des fichiers est mis en place pour enregistrer toutes les étapes de résolution du problème effectuées avec l'interface. Ce contrôle est réalisé en gardant la trace détaillée de toutes les modifications effectuées par l'usager en utilisant les différents panneaux d'édition de l'interface ou le panneau graphique principal. Le journal des changements est gardée dans des fichiers de configuration spéciaux que nous décrirons en détails dans la dernière section de ce chapitre.

2.2.1 Le système RCS et la numérotation des versions

Le système RCS (*Revision Control System*) est un logiciel de numérotation des versions des fichiers utilisé par l'interface pour gérer les différents fichiers du scénario. Le RCS permet de maintenir une liste complète de tous les changements produits dans un fichier, du moment de la création jusqu'à la version courante. Les numéros de versions sont incrémentés à chaque fois que le fichier est édité par un panneau de l'interface et que l'usager appuie sur un des boutons *Ok* ou *Apply* (enregistrement du fichier). RCS offre également la possibilité de rétablir un fichier d'une version précédente en sélectionnant le numéro de version désiré.

2.2.2 Mécanismes d'enregistrement

Après avoir créé un scénario, on procède ensuite à l'ouverture, via l'interface. En bref, le processus charge toutes les données du scénario en mémoire à l'ouverture de celui-ci, de façon à prendre un instantané du contenu de chaque fichier et de sa structure. A partir de là, ces données serviront à représenter les objets graphiques dans le panneau principal. Toute modification effectuée directement dans les panneaux pouvant affecter ces structures résulteront en un changement graphique des objets sur l'écran.

Comme nous l'avons déjà mentionné, chaque panneau d'édition correspond à un fichier. La technique adoptée pour modifier le fichier sur disque est la suivante. Au moment de l'affichage du panneau, une copie temporaire des données qu'il contient est faite dans des structures spécialement définies pour le panneau d'édition. Un tampon, qu'on appelle *audit trail buffer*, est ensuite créé. Il servira à stocker, sur la copie temporaire des données, toutes les modifications produites dans le panneau, par l'usager. À la suite des modifications, l'usager peut décider de sauvegarder ces changements en appuyant sur *Apply*, ou de les rejeter en appuyant sur *Cancel*.

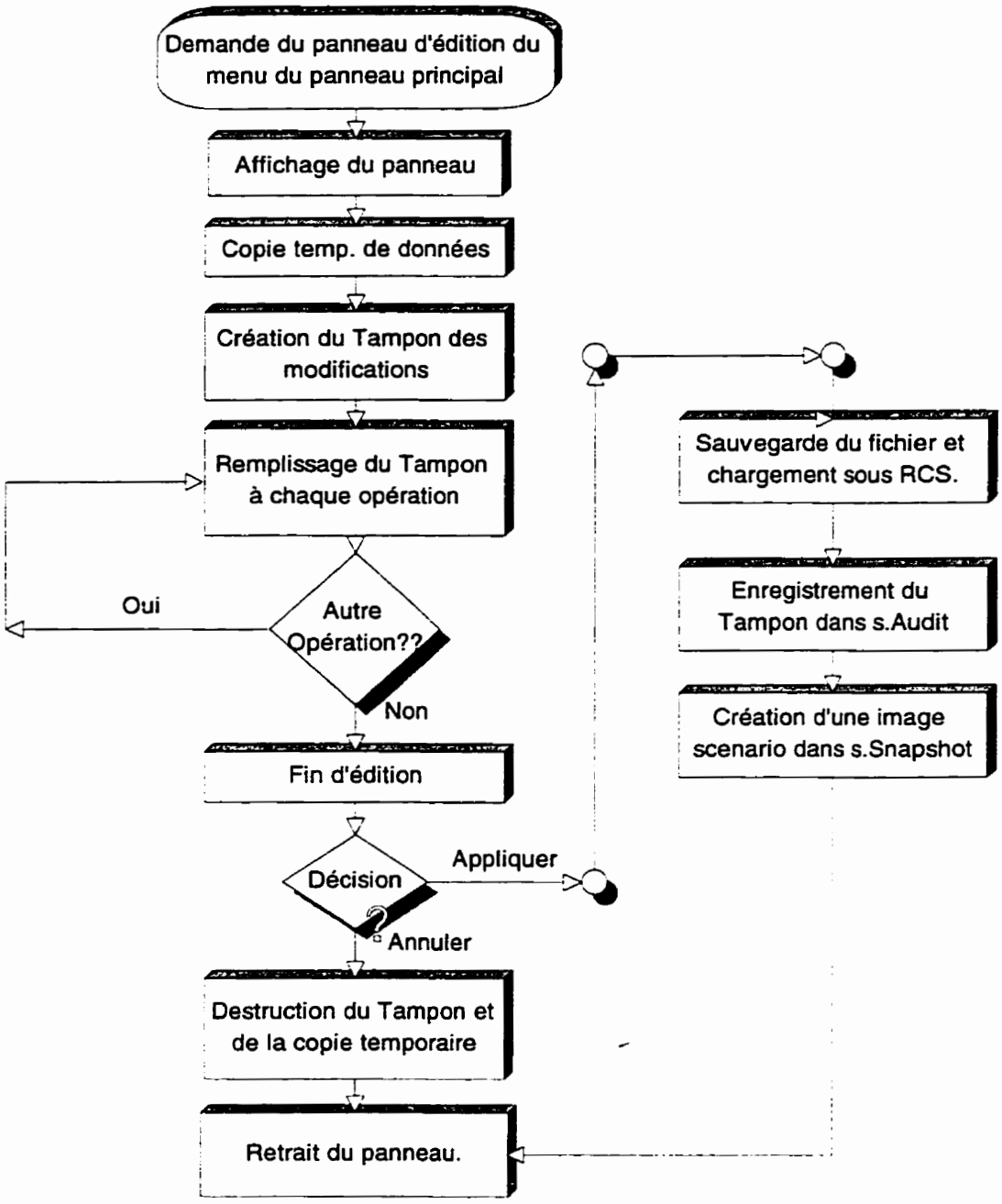


Figure 2.5 Diagramme du mécanisme d'enregistrement des modifications

Dans le cas d'une sauvegarde, la copie temporaire des structures de données servira à mettre à jour la copie originale en mémoire. Les données seront alors enregistrées dans le fichier en appelant une procédure d'écriture correspondante. Le numéro de version RCS sera incrémenté et les détails de modifications gardés dans le tampon du journal (*audit trail*) seront enregistrés dans un fichier spécial de configuration (*s.Audit*) que nous détaillerons plus loin dans ce chapitre.

La dernière étape de cette procédure est de prendre un instantané de l'état présent du scénario et de le garder dans un autre fichier de configuration (*s.Snapshot*) pour consultation future. La figure 2.5 est un diagramme logique qui illustre les opérations décrites ci-haut.

2.2.3 Technique de mémorisation des états intermédiaires du scénario (*Snapshots*)

Au moment de la création du scénario, un répertoire local est créé pour l'usager sur disque où les fichiers y sont copiés. A ce moment-là, l'interface crée un répertoire appelé RCS dans lequel elle placera toutes les versions RCS initiales des fichiers. Un instantané du scénario initial est ensuite créé et placé dans le fichier *s.Snapshots*. Le fichier est principalement composé par des blocs représentant tous les numéros de versions RCS des fichiers constituant le scénario à un état donné. Chaque modification effectuée par l'usager entraîne automatiquement la création d'un nouveau bloc avec tous les détails nécessaires et l'ajout du bloc numéroté à la fin du fichier.

Parmi les informations additionnelles ajoutées au bloc, on peut citer: la date et l'heure auxquelles la modification a été effectuée, le type de modification et les statistiques correspondants à la solution après modification. Les différents types de modification sont énumérés dans le tableau 2.4:

Tableau 2.4 Type de modifications retenues dans le fichier d'instantanés

Type	Définition
PI_L_AUDIT_CREATE	Création du scénario.
PI_L_AUDIT_EDIT	Modification à l'aide d'un panneau d'édition.
PI_L_AUDIT_SOLUTION	Enregistrement de la solution.
PI_L_AUDIT_IMPORT	Importation des données dans le scénario.
PI_L_AUDIT_MASTER	Création du bloc après avoir trouvé une solution, et copie dans le répertoire.
PI_L_AUDIT_RESTORE	Création du bloc lors de la récupération d'un état précédent du scénario.
PI_L_AUDIT_UPDATE	Création du bloc après la mise à jour des fichiers du scénario avec le fichier maître.

Le type de modification *PI_L_AUDIT_SOLUTION* peut être le résultat d'une sauvegarde de solution après exécution de l'optimiseur, ou même une sauvegarde de solution dans le panneau principal après manipulation et construction manuelle graphique des rotations.

Dans les tableaux suivants, le fichier *s.Snapshots* présente trois différents types de blocs comprenant tous les détails expliqués ci-haut. Le tableau 2.5, numéroté ID 1 représente le bloc construit au moment de la création du scénario et il est intitulé *INITIAL VERSION*. Le tableau 2.6 (ID 2) correspond à une modification de données de type

édition effectuée dans un panneau d'édition. Le tableau 2.7 (ID 3) représente une sauvegarde de la solution après modification graphique manuelle dans le panneau principal:

Tableau 2.5 Bloc d'instantanés ID1

```
Snapshots 1.0 Altitude/Pairing Generator
Description:

ID 1
   Tag INITIAL VERSION
   Parent 0
   AuditSerialNumber 1
   File a.AcRouting
   File a.AcTimeConn 1.1 1.11
   File a.Fleets 1.1 1.12
   File p.CmlFares 1.1 1.10
   File p.Coterm 1.1 1.2
   File p.DutyComposition 1.1 1.10
   File p.DutyCriteria
   File p.LegDomicile
   File p.Param 1.1 1.23
   File p.Region 1.1 1.1
   File p.SchedCarryOut
   File p.SchedCml 1.32 1.32
   File p.SchedCmlFiltered
   File p.SchedDhFiltered.dhg
   File p.SchedJumpseat 1.1 1.10
   File p.SchedLeg 1.1 1.10
   File p.SchedLegFiltered
   File p.StationDst 1.1 1.10
   File p.StationGmt 1.1 1.11
   File p.TimeConn 1.1 1.4
   File p.TimeLayover 1.1 1.2
   File p.TmfDh
   File p.TmfDuty
   File p.TmfTrip
   File p.TmfTripExport
   File p.TmfUser 1.1 1.1
   Date 02/15/1996
   Time 14:15
   Type PI_L_AUDIT_CREATE
   Synthetic 0.000000
   End
```

Tableau 2.6 Bloc d'instantanés ID2

```
ID 2
   Parent 1
   AuditSerialNumber 2
   File a.AcRouting
   File a.AcTimeConn 1.1 1.11
   File a.Fleets 1.1 1.12
   File p.CmlFares 1.1 1.10
   File p.Coterm 1.1 1.2
   File p.DutyComposition 1.1 1.10
   File p.DutyCriteria
   File p.LegDomicile
   File p.Param 1.3 1.23
   File p.Region 1.1 1.1
   File p.SchedCarryOut
   File p.SchedCml 1.32 1.32
   File p.SchedCmlFiltered
   File p.SchedDhFiltered.dhg 1.1
   File p.SchedJumpseat 1.3 1.10
   File p.SchedLeg 1.1 1.10
   File p.SchedLegFiltered
   File p.StationDst 1.1 1.10
   File p.StationGmt 1.1 1.11
   File p.TimeConn 1.1 1.4
   File p.TimeLayover 1.1 1.2
   File p.TmfDh
   File p.TmfDuty
   File p.TmfTrip 1.1
   File p.TmfTripExport
   File p.TmfUser 1.1 1.1
   Date 02/15/1996
   Time 14:59
   Type PI_L_AUDIT_EDIT
   Statistics: Soln Blk  = 6h48       Total Blk = 3455h22\Real Cost = 3951.64     Soft Cost =
0.00   \Credit   = 12h59      Synthetic = 90.93%\Duty Days = 2         Open Time = 3448h34
   Synthetic 90.931373
   End
```

Tableau 2.7 Bloc d'instantanés ID3

```
ID 3
   Parent 2
   AuditSerialNumber 3
   File a.AcRouting
   File a.AcTimeConn 1.1 1.11
   File a.Fleets 1.1 1.12
   File p.CmlFares 1.1 1.10
   File p.Coterm 1.1 1.2
   File p.DutyComposition 1.1 1.10
   File p.DutyCriteria
   File p.LegDomicile
   File p.Param 1.3 1.23
   File p.Region 1.1 1.1
   File p.SchedCarryOut
   File p.SchedCml 1.32 1.32
   File p.SchedCmlFiltered
   File p.SchedDhFiltered.dhg 1.1
   File p.SchedJumpseat 1.2 1.10
   File p.SchedLeg 1.1 1.10
   File p.SchedLegFiltered
   File p.StationDst 1.1 1.10
   File p.StationGmt 1.1 1.11
   File p.StationUps 1.1 1.11
   File p.TimeConn 1.1 1.4
   File p.TimeLayover 1.1 1.2
   File p.TmfDh
   File p.TmfDuty
   File p.TmfTrip 1.1
   File p.TmfTripExport
   File p.TmfUser 1.1 1.1
   File t.TmfOptTemplate
   Date 02/15/1996
   Time 14:57
   Type PI_L_AUDIT_SOLUTION
   Statistics: Soln Blk  = 6h48      Total Blk = 3455h22\Real Cost = 3951.64     Soft Cost =
0.00   \Credit   = 12h59     Synthetic = 90.93%\Duty Days = 2        Open Time = 3448h34
   Synthetic 90.931373
   End
```

Si on prend le dernier bloc (Tableau 2.7) comme exemple, on notera qu'en dessous du champs *Type* on trouve le détail des statistiques enregistrées au moment où l'usager a fait une sauvegarde. Parmi ceux-ci, on retrouve le *Synthetic* de la solution

(pourcentage de vols non-couverts), le *Real Cost* (coût réel de la solution courante), le *Open Time* (le nombre d'heures de vols restant à couvrir) etc.

Parallèlement à la construction des instantanés, le fichier de protocole du journal (*Audit Trail*), contient le chemin parcouru et une spécification détaillée des changements qui ont eu lieu à chaque action, soit à chaque prise d'instantané. Voici un exemple du fichier de protocole du journal des changements:

Tableau 2.8 Fichier de protocole de traçage (*Audit Trail*) (première partie)

```
1       CREATE 02/15/1996 14:15
        Snapshot: 1 (INITIAL)
        Explanation: INITIAL VERSION
        User: souheil

        Name: Jump1
        Directory: /home/souheil/altitude/bid9302/Jump1
        Master: /home/pups/master/altitude/bid9302
        Context: _747
        Initial files:
        a.AcTimeConn   /home/pups/master/altitude/bid9302/a.AcTimeConn version 1.11
        b.special    /home/pups/master/altitude/bid9302/_747/b.special version
        b.lines              /home/pups/master/altitude/bid9302/b.lines version
        p.SchedCml      /home/pups/master/altitude/bid9302/p.SchedCml version 1.32
        p.CmlFares    /home/pups/master/altitude/bid9302/p.CmlFares version 1.10
        a.Fleets         /home/pups/master/altitude/bid9302/a.Fleets version 1.12
        p.Coterm         /home/pups/master/altitude/bid9302/_747/p.Coterm version 1.2
        p.TimeConn   /home/pups/master/altitude/bid9302/_747/p.TimeConn version 1.4
        p.DutyComposition/home/pups/master/altitude/bid9302/p.DutyComposition version 1.10
        p.SchedJumpseat  /home/pups/master/altitude/bid9302/p.SchedJumpseat version 1.10
        p.TimeLayover  /home/pups/master/altitude/bid9302/_747/p.TimeLayover version 1.2
        p.Param          /home/pups/master/altitude/bid9302/_747/p.Param version
        p.Region         /home/pups/master/altitude/bid9302/p.Region version 1.1
        p.SchedLeg          /home/pups/master/altitude/bid9302/p.SchedLeg version 1.10
        p.StationDst         /home/pups/master/altitude/bid9302/p.StationDst version 1.10
        p.StationCie         /home/pups/master/altitude/bid9302/p.StationCie version 1.11
        p.StationGmt          /home/pups/master/altitude/bid9302/p.StationGmt version 1.11
        p.TmfTrip            /home/pups/master/altitude/bid9302/p.TmfTrip version
        p.TmfUser        /home/pups/master/altitude/bid9302/_747/p.TmfUser version 1.1
```

Tableau 2.8 Fichier de protocole de traçage (*Audit Trail)*

```
2       EDIT   08/16/94 14:30
        Snapshot : 4 (INTERMEDIATE)
        Modifications to file: p.TimeLayover version 1.2
        Explanation:

        A> S AMS DUTY DOM DUTY INT 10h00


3     SOLUTION 02/15/1996 14:57
      Snapshot: 4 (INTERMEDIATE)
      Explanation:
      User: souheil
      Files:
            p.SchedDhFiltered.dhg    version 1.2    (Master version n/a)
            p.SchedJumpseat          version 1.2    (Master version 1.10)
            p.TmfTrip                version 1.2    (Master version n/a)

      Join
      Create Trip Instance    1 03/03/1993
```

2.2.4 Description du fichier de protocole de traçage (*Audit Trail*)

Des informations détaillées sont enregistrées dans le fichier pour que l'usager (planificateur) puisse retracer toutes les étapes qu'il a exécutées depuis la création du scénario. L'information enregistrée dépend du type d'opération exécutée. Prenons l'exemple ci-dessus. Sous l'entrée numéro 1 de type *CREATE*, on trouve une liste comprenant tous les fichiers copiés du répertoire maître avec les numéros de version RCS et le chemin d'origine complet. Cette entrée correspond à la création du scénario.

L'origine des fichiers se trouve sous le champs *Master*, le nom du scénario sous *Directory* et le nom du contexte sous *Context*. Tous les fichiers copiés sont identifiés comme versions initiales numérotées 1.1. Les numéros de RCS sont les versions maîtres des fichiers que l'on garde pour des raisons de comparaison et de mise à jour que nous

expliquerons plus loin dans ce chapitre. Le champs *Explanation* peut contenir un commentaire que l'usager aura tapé au moment de la sauvegarde.

L'entrée numéro 2 correspond à un type de modification *EDIT* faite à l'aide d'un panneau d'édition des temps minimaux de repos. Le fichier touché est le p.TimeLayover. Puisque c'est la première modification apportée au fichier depuis la création du scénario, la version du fichier est incrementée de 1.1 à 1.2, d'où la ligne «*Modifications to file: p.TimeLayover version 1.2*». Les détails des modifications sont représentés par la ligne «A> S AMS DUTY DOM DUTY INT 10h00» Le champs «A>» au début de la ligne veut dire qu'il s'agit d'un ajout de données au fichier (un retranchement est indiqué par D>). Tout ce qui vient après A> représente une copie des données ajoutées au fichier par l'usager. Il faut interpréter cette ligne comme suit: le temps minimal de repos entre une journée de travail (DUTY) domestique (DOM) et une journée de travail internationale (INT) associé à la station (S) AMS est de 10 heures.

L'entrée numéro 3 correspond à une sauvegarde de solution (*SOLUTION*) suite à des manipulations graphiques effectuées dans le panneau principal. Les fichiers touchés sont: *p.TmfTrip*, *p.SchedJumpseat* et *p.SchedDhFiltered.dhg* qui s'incrémentent tous à la version 1.2. Les modifications se lisent comme suit: *Join* et *Create Trip Instance* 1 03/03/1993 indiquent une opération de type *Join* qui implique la construction graphique manuelle de plusieurs segments de vol pour former une rotation d'équipage (*Trip*). Le chiffre 1 correspond au numéro de la rotation et la date correspond à la journée à laquelle la rotation débute.

Le fichier de protocole de traçage est en fait un moyen pour l'usager ou le planificateur de garder un registre de tous les changements qui ont été effectués au scénario, étape par étape. Ceci lui donne la possibilité de revenir en arrière pour annuler une modification et reprendre le processus afin d'arriver à une solution plus satisfaisante.

Pour se faire, il suffit d'entrer dans le panneau des instantanés et d'utiliser la souris pour sélectionner le numéro identifiant l'instantané avec lequel on désire repartir le processus. La restauration est exécutée instantanément.

2.2.5 Mise à jour avec le répertoire maître

Le superviseur peut, pour différentes raisons, avoir à apporter des modifications à un fichier maître dans le répertoire maître. L'usager planificateur devra donc incorporer ces modifications dans la copie locale de son scénario. La question qui se pose est: qu'arrive-t-il si l'usager a fait des modifications dans sa copie locale du fichier après qu'il ait été copié du répertoire maître? C'est là que les versions maîtres des fichiers gardés dans le fichier s.Snapshots deviennent utiles.

La version maître du fichier correspond à son état dans le répertoire maître, lors de la création du scénario. Elle constitue le point de référence pour identifier les changements faits par le superviseur. Il s'agira donc de fusionner les changements apportés depuis la version maître de référence, dans la version courante du scénario. L'exemple suivant illustre bien la marche à suivre:

On suppose que le fichier *p.TimeLayover* existait dans le répertoire maître à la version 1.5. A la création du scénario de l'usager, la version locale du fichier est initialisée à 1.1. L'usager fait des changements dans son fichier local, à plusieurs reprises, et la version 1.1 devient 1.3. De son côté, le superviseur fait des changements dans le fichier maître qui deviendra la version 1.6.

Le panneau de mise à jour (fig. 2.6) indique à l'usager qu'il doit mettre son fichier à jour afin d'obtenir la nouvelle version du fichier maître contenant des modifications. Cette procédure implique une fusion entre les versions 1.5 et 1.6 du fichier maître ainsi que de la version 1.3 du fichier local de l'usager.

La fusion peut parfois générer des conflits entre les modifications de l'usager et celles du superviseur. Dans ce cas l'usager utilisera le panneau de résolution de conflits (fig. 2.7) afin de choisir l'une ou l'autre des données dans la série de conflits. La dernière étape sauvegarde le fichier dont la version passera à 1.4 automatiquement.

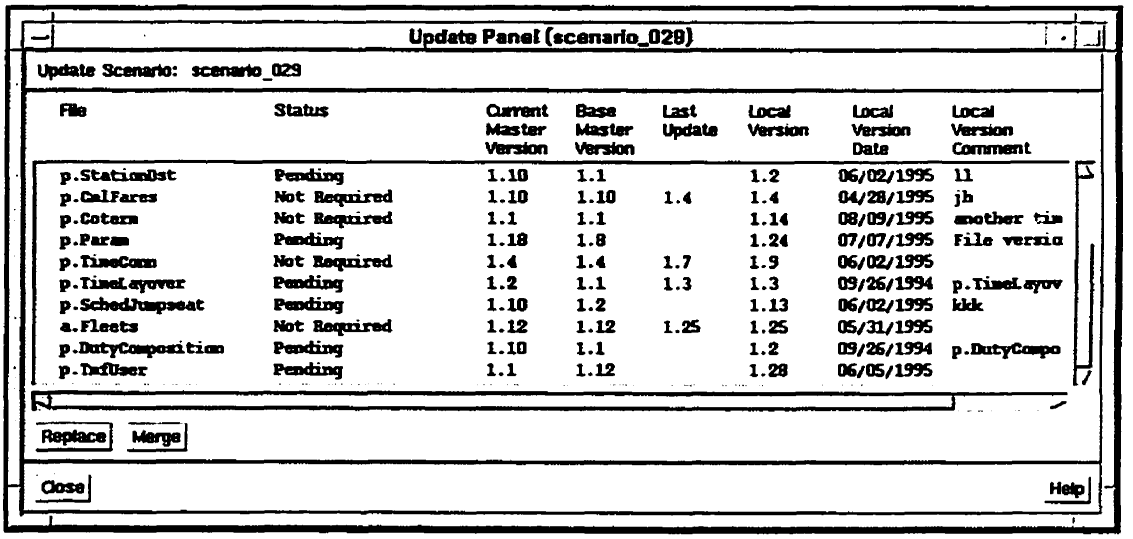


Figure 2.6 Panneau de mise à jour (*Update*)

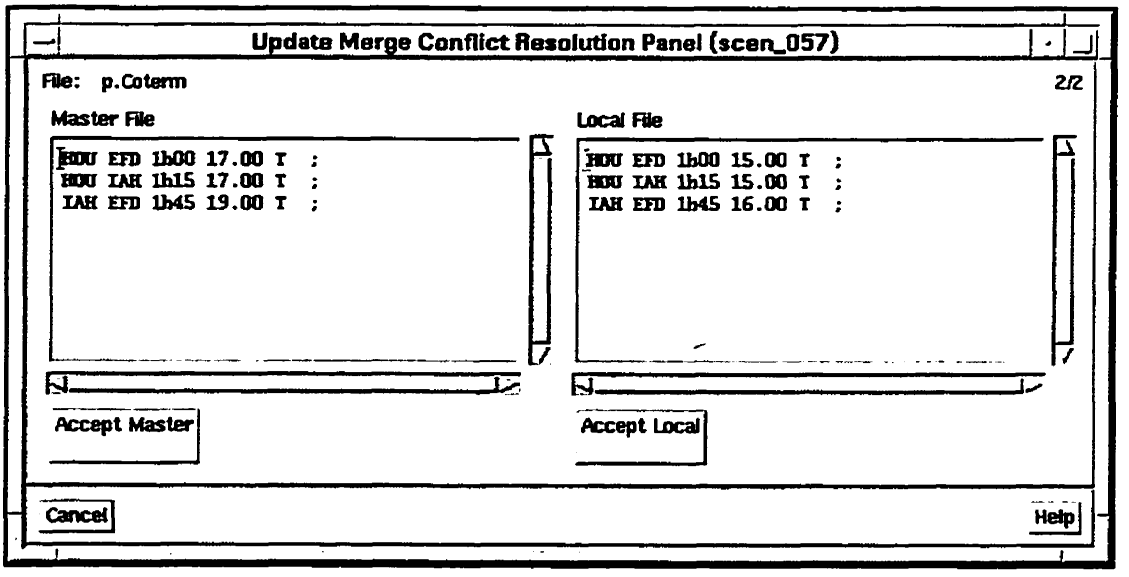


Figure 2.7 Panneau de résolution de conflits (*Update Merge Conflict Resolution*)

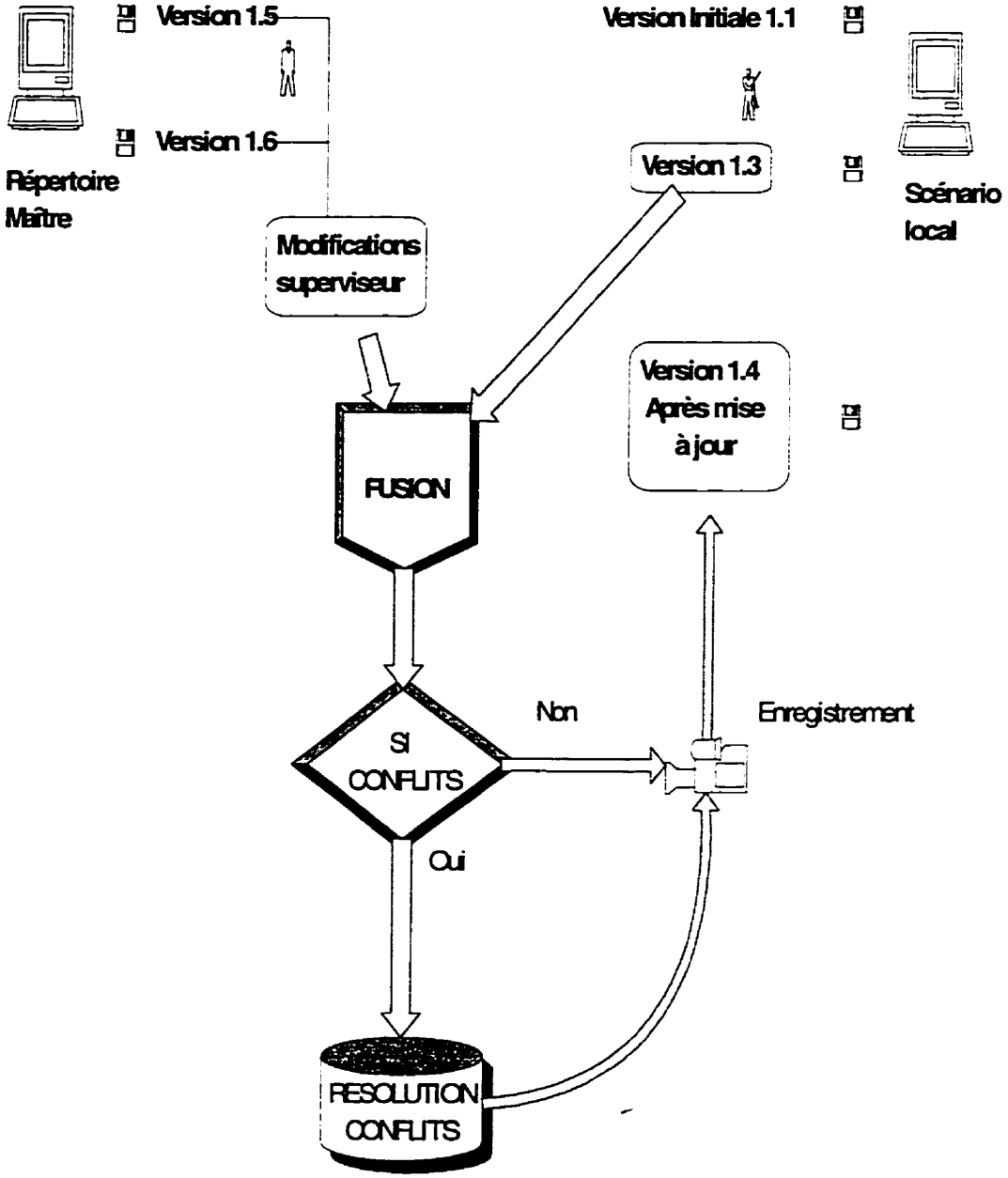


Figure 2.8 Méthode de résolution de conflits

2.2.6 Originalité et avantages des mécanismes

Le système est unique dans son traitement de la mémorisation des étapes. Il possède une fonctionnalité très puissante qui permet à l'usager de revoir l'évolution de son scénario depuis sa création. Le processus de mémorisation est beaucoup plus complexe que la sauvegarde d'une série de modifications dans un seul fichier. En effet, ce processus implique un ensemble de modifications effectuées en différentes étapes, dans plusieurs fichiers.

Une étape est définie comme une modification à un fichier de paramètres ou d'entrée, ou bien, comme une action qui modifie les fichiers de solution (rotations). Elle est représentée par les détails des changements ainsi que la liste des fichiers affectés. Toutes ces étapes sont accumulées grâce à un mécanisme de traçage (*Audit Trail*) qui offre au planificateur une vue globale de toutes les modifications produites depuis le début du scénario. De plus, un bloc de statistiques accompagne chaque étape pour permettre de visualiser l'impact de chaque modification sur l'état de la solution.

Un mécanisme de prise d'instantanés (*Snapshots*) assez complexe agit en parallèle au processus de mémorisation des étapes. Ce processus produit un arbre historique représentant tous les états du scénario après chaque action (modification). Le système est conçu pour permettre à l'usager de se repositionner n'importe où dans l'arbre. Il est important de mentionner que cette opération est beaucoup plus sophistiquée que les mécanismes de ***Undo*** qu'on voit dans les applications commerciales multi-fenêtrée sur le marché. Les outils (panneaux) mis à la disposition de l'usager sont faciles d'utilisation, mais les opérations qui se cachent derrière chaque panneau sont très complexes et ont nécessité beaucoup de recherche et d'analyse.

Chaque panneau est une entité, qui gère ses propres données et leurs traces de modifications. Des méthodes de mémorisation et de mise en tampon (Figure 2.5) assez sophistiquées et uniques, ont été nécessaires pour la gestion des instantanés. Quand l'usager sauvegarde ses changements, le panneau copie la zone tampon et en fait un

enregistrement au niveau du module *Snapshot* présenté dans le panneau correspondant (Figure 1.5).

2.3 Environnement et configuration

La section suivante se concentre sur la configuration qui supporte l'application, que ça soit au niveau du matériel, les variables d'environnement ou les fichiers de configuration. A la fin de la section on discutera les avantages des techniques utilisées et les raisons pour lesquelles le logiciel est facilement transportable sur plusieurs systèmes (compagnies).

2.3.1 Matériel et environnement

La plate-forme utilisée pour le développement comprend des stations de travail *Unix* fonctionnant sur *X Windows*. Les outils de développement sont des librairies de *X Windows*, *Xt* et *Motif* version 1.2, sans oublier le compilateur *gcc*.

L'application nécessite la déclaration de certaines variables d'environnement qui serviront pour les raisons suivantes:

- spécifier l'endroit où le répertoire période maître est situé;
- spécifier l'endroit où les scénario usager doivent être créés;
- spécifier l'endroit des exécutables d'optimisation: *ARG DHG* et *PG*;
- spécifier l'endroit où se trouvent tous les fichiers de configuration.

Au moment où l'interface est lancée une procédure d'initialisation est exécutée. Elle comprend la lecture et le chargement de la majorité des fichiers de configuration, que nous traiterons dans la section suivante. Les variables d'environnement sont des variables qui appartiennent au *shell Unix* et qui sont lues et chargées en mémoire durant cette procédure.

2.3.2 Fichiers de configuration

L'interface possède plusieurs fichiers de configuration qui permettent à l'usager d'ajuster son environnement selon ses besoins, sans avoir à modifier le code. Parmi ces fichiers on cite les catégories suivantes: les fichiers de ressources X , les fichiers de catalogues, le fichier des spécifications d'imprimantes, et les fichiers d'aide.

Les ressources X contrôlent presque tout ce qui est affiché au niveau de l'interface. Par exemple, les polices, la taille des étiquettes (*labels*), les couleurs, le contenu des menus, le mode d'affichage des statistiques et bien d'autres. Un fichier situé dans le répertoire de configuration générique (*/home/projet/gui/config*) sert pour les valeurs de défaut de toutes les ressources. L'usager peut toujours garder, dans un fichier local, ses propres valeurs qui écraseront les valeurs initiales. Le fichier d'écrasement est situé dans le répertoire local de l'usager: */home/user*, sous le nom *.pirc*.

Les fichiers de catalogues contiennent toutes les étiquettes (*labels*) et tous les messages d'erreur de tous les panneaux de l'interface. De cette façon, l'usager pourrait changer la représentation textuelle dans ses panneaux sans avoir à changer le code auquel il n'a d'ailleurs pas accès.

Les spécifications des imprimantes reliées au système, sont indiquées dans le fichier de configuration *printers*. Il s'agit d'un fichier découpé en blocs, un bloc par imprimante. Chaque bloc contient le nom de l'imprimante, sa ligne de commande pour imprimer du texte, sa commande pour imprimer en *postscript* les pages graphiques et finalement sa commande d'annulation du processus.

L'aide contextuelle est séparée en différents fichiers et chaque fichier correspond à un panneau l'interface. Ces fichiers sont situés dans le répertoire de configuration générique au niveau *help* (*/home/projet/gui /config/help*). Un bouton d'aide dans chaque panneau permet d'accéder au contenu de ces fichiers. Des liens hypertextes permettent de naviguer d'un endroit à l'autre dans le texte, ou d'un panneau à l'autre selon les besoins.

Un index est aussi disponible aux usagers pour les sujets ne correspondant pas nécessairement à un panneau de l'interface.

2.3.3 Avantages des techniques utilisées

Beaucoup d'efforts ont été mis sur l'organisation de la gestion de l'environnement pour produire un logiciel flexible et facilement configurable. Tous les éléments mentionnés dans les sections précédentes collaborent à rendre l'environnement assez facile à gérer. L'application conçue est facilement transportable d'un système à l'autre. En effet, le logiciel s'adapte automatiquement à chaque nouvelle installation, parce qu'il a la capacité de mémoriser une multiplicité de configurations qui correspondent aux différents environnements. Par exemple, les fichiers de catalogues peuvent être écrits simultanément dans plusieurs langues et en activant un élément de configuration toutes les étiquettes de l'interface correspondront à la langue désirée.

Un autre aspect important est que la même application peut être adaptée à différents types de clientèle, chez les compagnies aériennes ou autres. Le système est développé de façon à ce que l'interface graphique puisse être configurée selon les différents besoins des clients, par l'entremise des fichiers de ressources.

CHAPITRE III:

ANALYSE OPÉRATIONNELLE DE L'INTERFACE, ORGANISATION GRAPHIQUE ET STRUCTURATION DE LA MÉMOIRE

Le domaine aérien se caractérise par l'immensité du nombre de données à traiter (nombre élevé de vols actifs, vols commerciaux, stations, etc.). Concevoir une interface graphique pour afficher et manipuler ces données a donc nécessité un temps de recherche considérable afin de s'assurer que les techniques implantées sont les plus efficaces et les plus performantes, surtout pour ce qui est de la gestion des structures d'affichage et leurs pointeurs aux structures de données en mémoire. Ce chapitre se concentre sur la description de ces méthodes de gestion.

Nous présenterons aussi l'aspect graphique de l'application. On discutera les possibilités de visualisation des objets graphiques ainsi que les différentes opérations manuelles qui agissent sur ces objets. Une description détaillée des techniques de structuration au niveau des modules ainsi que l'organisation de la mémoire suivront. On finira en présentant les méthodes utilisées pour les opérations graphiques et les techniques de mises à jour.

3.1 Représentation graphique

Le panneau graphique principal est le panneau central de l'application. On y affiche des rotations (*trips*), des journées de travail (d*uties*), des segments de vol (*legs*) et des rotations d'avions. Il nous permet d'éditer plusieurs aspects de la solution et donne aussi accès à tous les autres panneaux de l'interface grâce aux menus, (voir le diagramme des fonctionnalités à l'annexe A). Ce panneau apparaît une fois que la validation du nom et du mot de passe de l'usager est complétée dans le panneau de mot de passe.

Le panneau graphique est divisé en quatre sections. Une barre de menus déroulants se trouve au haut du panneau. Juste en dessous, on trouve un champs de texte contenant une brève description du scénario, suivi d'une barre d'outils qui sont utilisés pour exécuter des opérations manuelles sur les objets graphiques. Les différentes techniques et approches utilisées dans ces opérations sont détaillées plus loin dans ce chapitre. A l'extrémité droite de cette section réside un spécificateur (*Days Slider*) qui indique le nombre de jours affiché dans la section graphique du panneau principal.

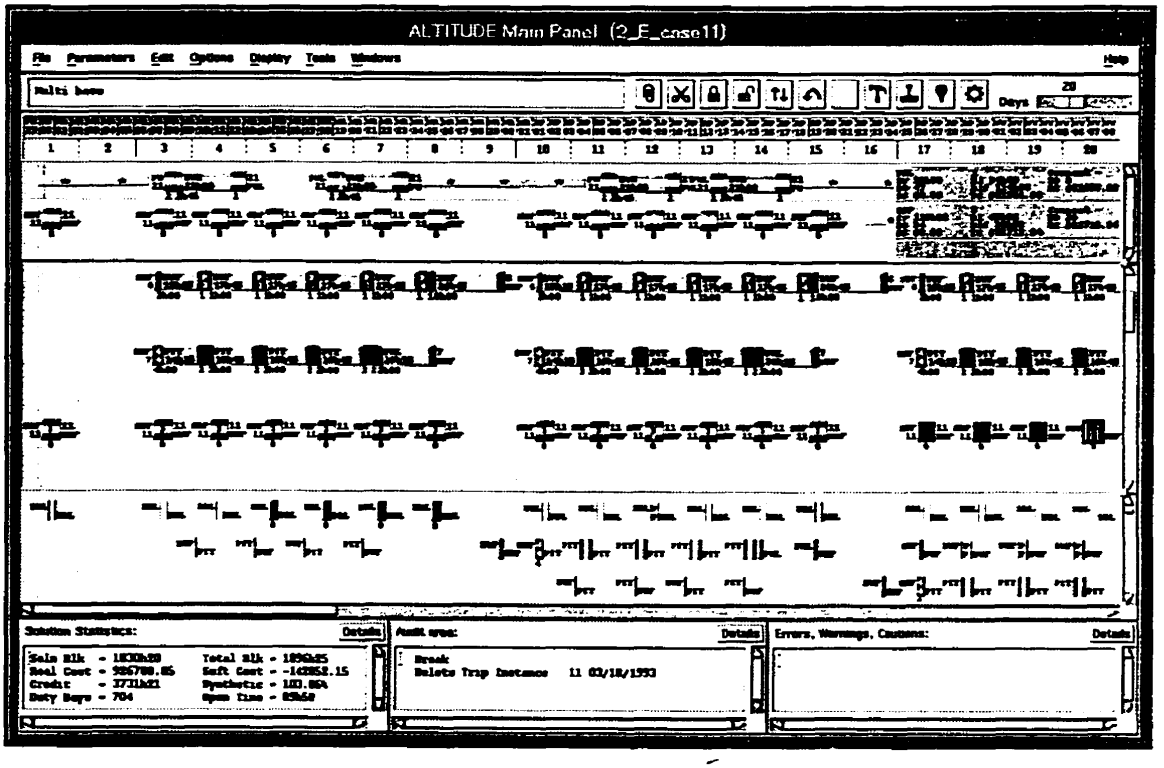


Figure 3.1 Panneau principal

La section la plus importante du panneau est la partie graphique, qui se situe dans la section centrale du panneau et qui offre une représentation graphique des données et des modifications. Cette région graphique affiche les objets, principalement les rotations, regroupés sur des lignes horizontales. La région représente toute la période du scénario

divisée en journées séparées par des lignes pointillées verticales dans toutes les fenêtres d'affichage du panneau. Par défaut les rotations portant le même numéro sont regroupées sur la même ligne horizontale, mais le regroupement ne suit aucune logique. L'usager peut toujours déplacer les rotations et les objets graphiques pour les mettre sur la même ligne tant qu'ils ne se chevauchent pas.

Au bas du panneau, se trouvent trois régions textes qui servent à afficher respectivement, les statistiques courantes de la solution, un résumé des opérations manuelles graphiques et une liste des illégalités qui correspondent aux rotations sélectionnées par l'usager. Le panneau présente aussi des menus de fond (*popup*) qui apparaissent quand l'usager clique sur le bouton droit de la souris en s'assurant que le curseur soit positionné soit dans la fenêtre graphique, soit sur un objet graphique, selon le type de menu désiré.

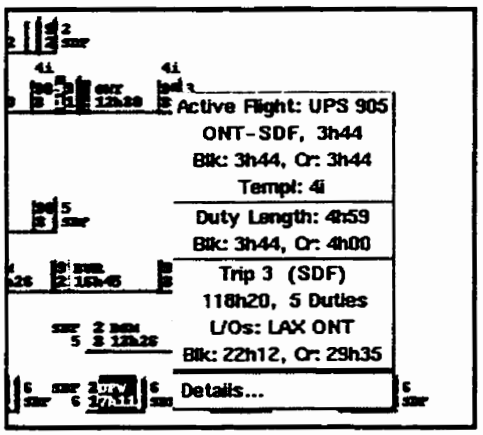


Figure 3.2 Menu de fond

3.1.1 Affichage des objets et sélection:

On reconnaît les différents types d'objets graphiques affichés dans la fenêtre principale grâce aux couleurs distinctes que l'usager aura choisies dans le panneau de couleurs (voir Annexe A). Dans la fenêtre des rotations d'équipage, on dénote généralement trois types d'objets graphiques:

- **Segments de vol**: Ce sont des tronçons de vols individuels. Ils peuvent être des vols actifs (appartenant à la compagnie), des vols passifs (mise en place interne) ou des vols commerciaux (mise en place externe). On différencie les vols couverts et non-couverts par les couleurs que l'usager leur aura attribuées dans le panneau de couleurs.

- **Services de vol (SDV)**: Un SDV est une séquence de tronçons de vols séparés par des temps de repos, temps de connexion ou même de transport d'équipage au sol d'un aéroport à un autre (*ground deadhead*). Les groupes affichés peuvent représenter des journées de travail (*duties*) que l'usager peut utiliser pour construire des rotations.

- **Rotations**: Ce sont des rotations d'équipage. Elles sont différenciées des services par un numéro de rotation qui leur est assigné. La couleur des chiffres qui apparaissent de chaque côté de la rotation indique si celle-ci est générée par l'optimiseur ou créée manuellement par l'usager. L'ensemble de toutes les rotations forme la solution, contribue aux statistiques et représente le contenu du fichier *p.TmfTrip*.

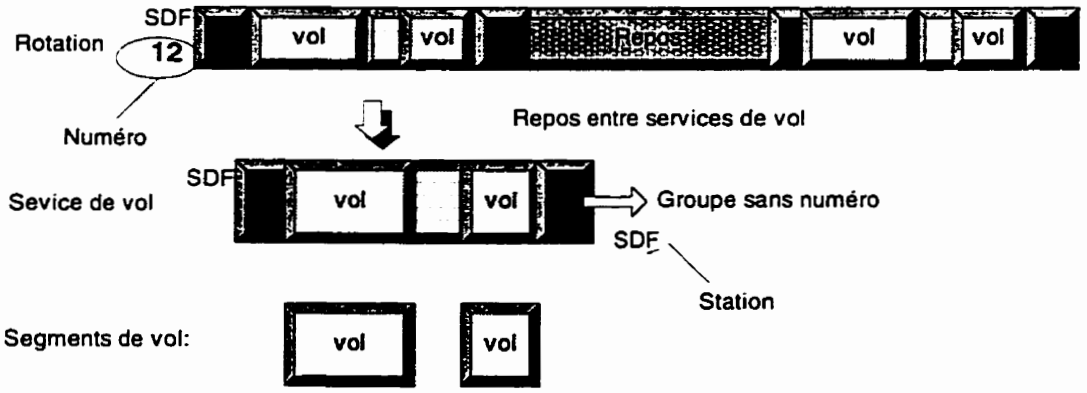


Figure 3.3 Représentation hiérarchique d'une rotation

Les objets graphiques sont représentés par des rectangles de hauteur et de couleur prédéfinies. On utilise le panneau de couleurs pour assigner les couleurs aux objets graphiques. Les extrémités gauche et droite du rectangle indiquent le début et la fin de l'objet respectivement. Plusieurs objets peuvent apparaître sur la même ligne. Les objets sont hiérarchiques; une rotation peut contenir plusieurs services de vol et chaque service de vol peut contenir un ou plusieurs vols séparés par des temps de connexion.

La plupart des opérations offertes par l'interface dépendent de l'ensemble des objets sélectionnés dans la fenêtre. L'état actif des boutons d'opération dépend aussi, dans plusieurs cas, du nombre d'objets sélectionnés. La sélection d'un objet se fait en cliquant le bouton gauche de la souris quand le curseur est sur l'objet. La figure 3.4 démontre la technique de sélection d'un ensemble d'objets, qui se fait en cliquant et déplaçant la souris pour former une région rectangulaire qui englobe tous les objets désirés.

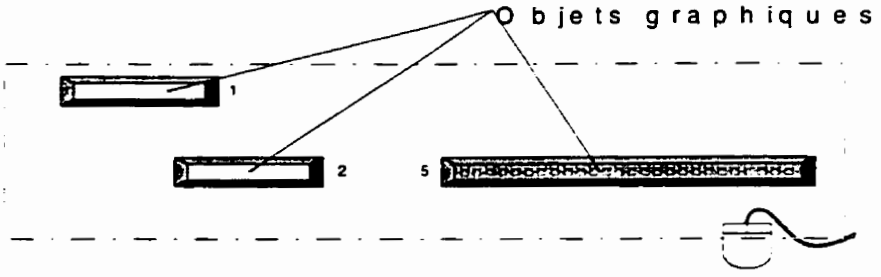


Figure 3.4 Technique de sélection d'un ensemble d'objets

3.1.2 Options et niveaux d'affichage (Zoom)

Au dessus de la fenêtre principale se trouve une barre de période qui affiche l'ensemble des journées contenues dans le scénario. L'usager peut choisir d'afficher seulement qu'une partie de cette période à l'écran. Pour ce faire, il sélectionnera, à l'aide de la souris, la période qu'il veut afficher. Celle-ci sera indiquée par une région

surbrillante (section grise dans la figure 3.5) et la section dessous la barre de période indiquera plus en détails la région affichée.

Les lignes à gauche et à droite de la fenêtre limitent la période affichée. Si l'usager veut appliquer la longueur de la période affichée sur d'autres journées comprises dans la période totale, il peut le faire en glissant la barre de défilement horizontale (*scrollBar*) [13]. Le nombre de journées affichées est incrémenté avec le spécificateur de nombres de jours. Dans le cas où l'affichage des rotations s'étend au delà de la page graphique, l'usager peut défiler la longueur de la page avec la barre de défilement verticale.

La partie centrale du panneau principal peut contenir deux fenêtres d'affichage (*views*) de rotations possédant chacune sa barre de défilement verticale. Les deux fenêtres sont séparées par une ligne mobile équipée d'un bouton (*sash*) qui sert à l'agrandir ou la rapetisser.

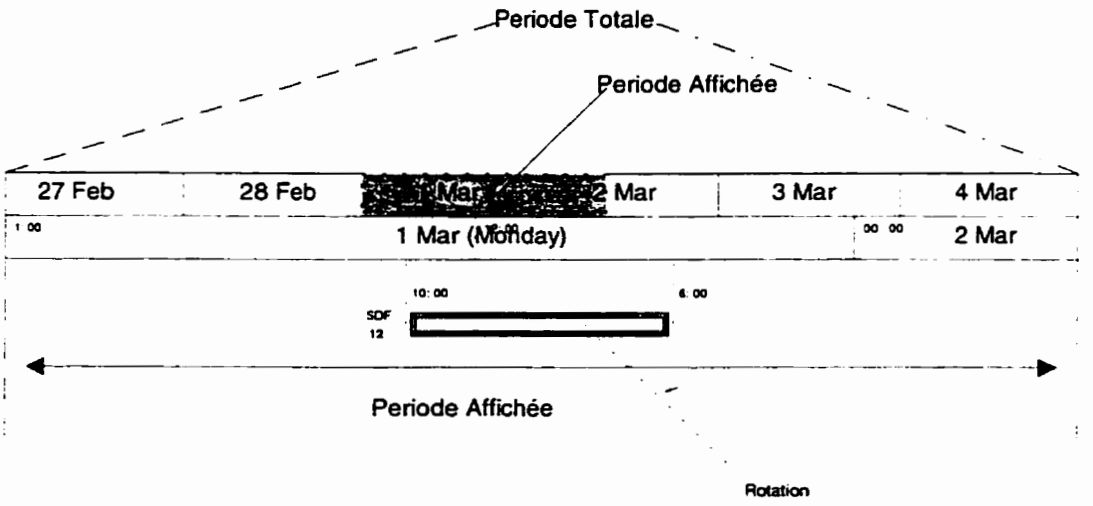


Figure 3.5 Techniques d'affichage

Le détail de chaque rotation est accessible grâce aux menus de fonds qui sont accédés en positionnant le curseur sur l'objet en question, et en choisissant l'option *"Details"* du menu (fig. 3.2). Un exemple de détails d'une rotation est présenté à l'annexe A.

La région texte du panneau, située sous la fenêtre principale est séparée en trois sections. La première section contient les statistiques de la solution qui sont mises à jour après chaque modification manuelle effectuée par l'usager. La section du centre contient un message de traçage (*audit*) correspondant à la dernière opération effectuée par l'usager. Ce message est gardé dans le tampon interne jusqu'à ce que l'usager décide de sauvegarder sa solution. La section de gauche affiche les messages d'erreurs pour avertir l'usager que l'objet sélectionné dans la fenêtre principale est erronné puisqu'il ne respecte pas les règles établies et la convention collective de la compagnie.

3.1.3 Liste des opérations

La plupart des opérations graphiques manuelles peuvent être initiées à partir de la barre d'outils qui se trouve en haut de la fenêtre graphique dans les panneau principal. On peut, entre autre, déplacer, joindre, briser, verrouiller, déverouiller, permuter, annuler, cacher et solutionner des objets dans la fenêtre. La plupart des opérations s'appliquent sur l'ensemble des objets sélectionnés dans la partie graphique du panneau principal.

- **Déplacer** (*Drag and Drop*): Tous les objets graphiques peuvent être déplacés à l'intérieur de la fenêtre graphique ou même d'une fenêtre à l'autre. Il s'agit de cliquer sur l'objet et glisser le curseur, à l'aide de la souris, à l'endroit désiré.

- **Joindre** (*Join*): Permet la formation d'une rotation ou d'un groupe en fusionnant un ensemble d'objets graphiques sélectionnés. Si l'objet sélectionné est déjà un groupe, il deviendra une rotation.

- **Briser** (*Break*): Permet la séparation de l'ensemble en des segments de vols individuels. On peut briser des groupes ou des rotations.

- **Verrouiller** (*Lock*): S'applique uniquement aux rotations. Les rotations verrouillées sont protégées contre tout genre d'opération, incluant le processus d'optimisation.

- **Déverrouiller** (*Unlock*): S'applique uniquement sur les rotations. Les rotations déverrouillées ne sont pas protégées contre ce genre d'opération, incluant le processus d'optimisation.

- **Permuter** (*Swap*): La permutation est effectuée entre les premières stations que les deux groupes ou rotations sélectionnés ont en commun. Si les deux objets sélectionnés n'ont pas de station en commun, le bouton permuter ne sera pas activé. Les deux objets ne sont pas nécessairement dans la même fenêtre.

- **Annuler** (*Undo*)**:** Annule la dernière opération.

- **Cacher** (*Hide*): Supprime tous les objets sélectionnés dans la fenêtre graphique. Les rotations sont cachées mais résident toujours en mémoire et font partie de la solution. On peut afficher ces objets à nouveau dans la fenêtre grâce aux panneaux filtres que nous décrirons au chapitre 4.

- **Résoudre** (*Solve*): Donne accès au panneau d'optimisation. Nous discuterons de ce panneau dans le chapitre 4: Méthode de solution.

3.2 Implémentation: Organisation et structuration de la mémoire

La structure et l'organisation du panneau graphique suit le concept noyau/modules. C'est à dire, qu'il existe un noyau central qui offre des services et des modules externes de plus haut niveau qui utilisent ces services, tout en gardant une certaine indépendance. Le gestionnaire de données est le noyau central qui communique avec, entre autre, le module d'affichage

Dans notre cas, le module d'affichage est le module indépendant qui gère l'affichage et les manipulations strictement graphiques. Ce module possède ses propres structures en mémoire et ses propres copies de données qui servent au détail d'affichage et d'emplacement. Toute opération affectant les données, la solution elle-même ou les statistiques, est effectuée par le module central qu'on appelle aussi le gestionnaire de données. Cette technique est mise au point en se servant des pointeurs de référence aux objets. Nous en discuterons plus loin.

3.2.1 Gestionnaire de données: Générateur de rotations (*Pairing Generator*)

Les fichiers de scénario les plus importants à considérer sont le fichier solution ou de rotations (*p.TmfTrip*), le fichier des vols à couvrir (*p.SchedLeg*) et le fichier des stations (aéroports) (*p.Station*). Lorsqu'on ouvre le scénario avec l'interface, ces fichiers et plusieurs autres sont lus et chargés en mémoire. Pour chaque type de données on assigne un type de structure différent. Les structures sont gardées en mémoire en suivant les méthodes d'arbres et de listes en chaîne.

Les éléments de base principaux entrant dans la composition des rotations sont évidemment les segments de vol (*legs*). Les informations correspondantes sont représentées dans les structures *PgLEG*. Ces éléments sont gardés dans une liste chaînée; toute structure contient un pointeur qui la lie à la structure (*struct*) suivante. La dernière structure pointe automatiquement à *NULL* pour indiquer la fin de la liste.

```
struct PgLEG {
        CH              szFlighAbbrev;      /* Numéro du vol */
                        PgPOINT  pointDep;/* Point de départ du vol */
        PgPOINT         pointArr;           /* Point d'arrivée du vol */
        CH              szAcType;           /* Type d'avion */
        PgJSEAT        *pjseat;             /* Les mises en place */
        PgCENT centFixCost;  /* Coût du vol */
```

```
        PgLEG       *plegNext;          /* Pointeur au leg suivant */
        PgLEG       *plegAggNext; /* Pointeur au leg suivant */
                                  /* dans son groupe d'Aggrégat.*/
};
```

Figure 3.6 Contenu de la structure *PgLEG*

Un autre moyen de garder ces segments est de les regrouper en agrégats (fig. 3.7). Les agrégats sont des vols qui portent le même numéro et qui possèdent les mêmes temps de départ et d'arrivée aux mêmes stations et qui se répètent périodiquement. Un autre pointeur est donc nécessaire pour suivre les instances à l'intérieur du groupe d'agrégats (*plegAggNext*).

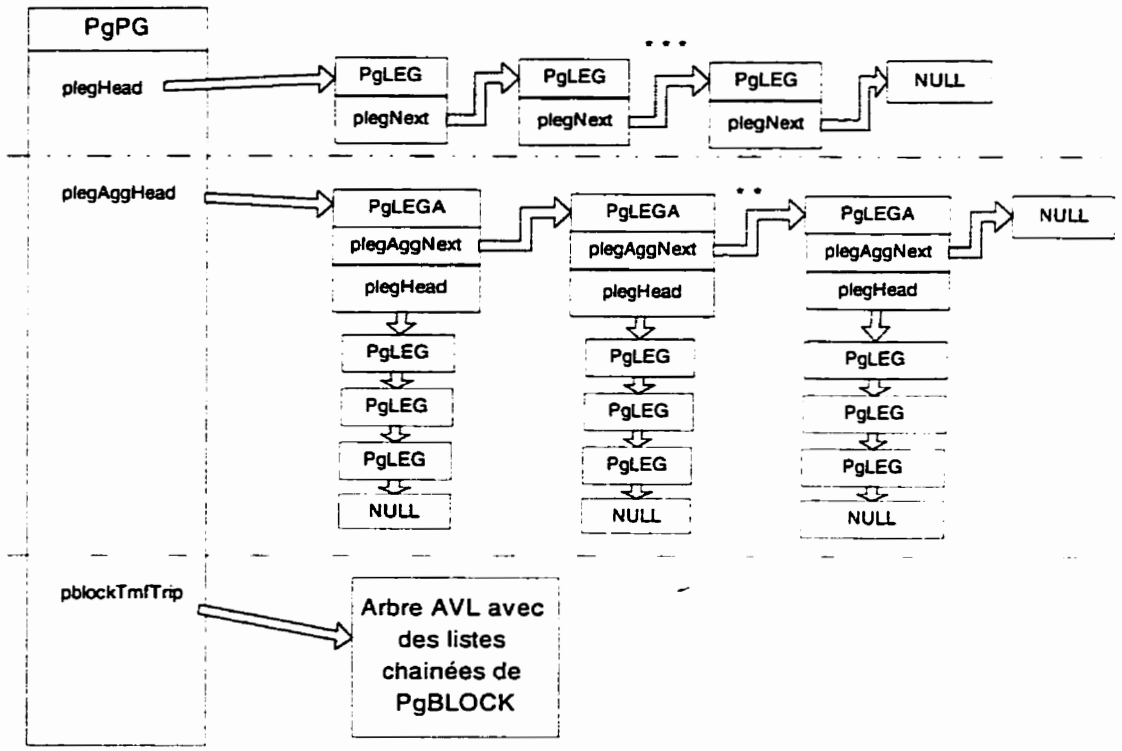


Figure 3.7 Organisation des structures dans le module de gestion (*PG*)

Il existe plusieurs structures de base pouvant entrer dans la construction des rotations. Une cellule peut être constituée d'un repos, d'une connexion, d'un transport au sol (*ground deadhead*), aussi bien que d'un vol (*leg*). Le repos et la connexion sont représentés par la station et le transport au sol nécessite que les deux stations soient des coterminaux. Ces données sont contenues en mémoire dans les structures suivantes: *PgREST* (*PgSTATION*) et *PgCOTERM*.

Afin de pouvoir construire des blocs (rotations) avec une suite d'éléments de mêmes types, les structures de base sont englobées par une structure de niveau supérieur *PgCELL*. La structure contient des pointeurs initialisés au moment de la construction du bloc pour indiquer les cellules qui se suivent et se précèdent dans la rotation (fig. 3.8) Donc, *PgCELL* peut être l'union des cellules suivantes:

PgLEG.

PgCOTERM.

PgREST.

```
struct PgCELL {
PgCELL *pcellNext; /* Pointeur à la cellule suivante de la rotation */
PgCELL *pcellPrev; /* Pointeur à la cellule précédente de la rotation */
PgINFO pinfo; /* Informations relatives à la cellule */
        union {
                PgLEG  pleg;
                PgREST prest;
                PgCOTERM      pcoterm;
        }
        PgGUI         *pGui; /* Pointeur à l'objet graphique affiché */
};
```

Figure 3.8 Structure de la cellule *PgCell*

Les rotations sont chargées dans les structures *PgBLOCK* (fig. 3.9) qui contiennent toutes les informations pertinentes à la rotation. On retrouve le type de bloc, la journée de début du bloc, son numéro, etc., ainsi que deux pointeurs indiquant la première et la dernière cellule du bloc, appelées respectivement *pcellHead* et *pcellTail* (fig. 3.10).

```
struct PgBLOCK {
        PgBLOCK         *pblockLeft,  /* Liens pour l'arbre AVL */
                        *pblockRight,
                        *pblockNext,
                        *pblockPrev;
        PgCELL *pcellHead;   /* Pointeur à la première cellule du bloc */
        PgCELL *pcellTail;          /* Pointeur à la dernière cellule */
        PgID            id;                 /* Journée début Rotation */
        PgBT            st;                 /* Type de bloc */
        L               lNo;                /* Numéro de rotation */
        PgGUI           *pGui;  /* Pointeur à l'objet graphique affiché */
};
```

Figure 3.9 Structure le la cellule *PgBlock*

Les rotations sont organisées dans un arbre *AVL* en regroupement de structures *PgBLOCK* reliées ensemble à l'aide des pointeurs à gauche et à droite (*pblockRight*, *pblockLeft*). Les blocs dans l'arbre sont triés par numéro de rotation (*lNo*). Un arbre AVL est un ensemble de listes chaînées organisé d'une façon équilibrée se servant des pointeurs gauche et droit de chaque élément. Ces pointeurs permettent de passer d'une branche à l'autre de l'arbre, chaque branche étant constituée d'une liste chaînée par elle-même. Chaque liste ou branche possède des propriétés qui facilitent son identification.

L'avantage d'un tel arbre est d'accélérer les algorithmes de recherche (de l'ordre *log N*) comparativement aux algorithmes de liste chaînées (de l'ordre *N*).

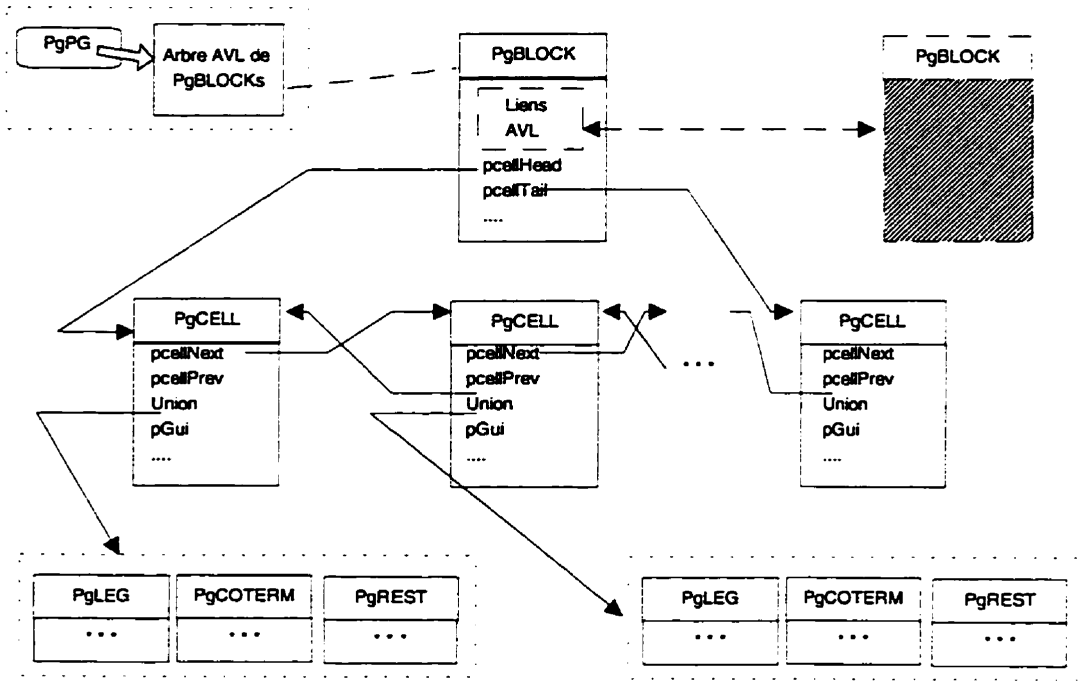


Figure 3.10 Formation des rotations avec les cellules

3.2.2 Techniques d'affichage

La structure principale du panneau graphique central est *PimMAIN* (fig. 3.11) qui est l'objet contexte pour le panneau et tous ses descendants. Théoriquement, comme il est possible d'avoir plus qu'un panneau principal dans l'application, le module interface a été construit en considérant que la structure *PimMAIN* est statique (*static*), mais pas globale. Donc, la plupart des fonctions et des routines découlant du panneau principal portent un pointeur vers cette structure (*void *pClient*). (voir [6])

En d'autres mots, cette structure offre l'accès à tous les différents types de données; le pointeur scénario *psc* contient un pointeur *PgPG* qui est à son tour le

pointeur clé vers le module gestionnaire de données. De son côté, le pointeur *PimVIEW* offre l'accès à tous les objets graphiques affichés dans la fenêtre graphique. À chaque fenêtre graphique dans le panneau principal correspond, évidemment, une structure *PimVIEW* différente. La structure *PgPG* donne accès à toutes les informations contenues dans notre solution: les rotations (blocs), les vols, les stations, les statistiques (coûts), etc.

```
typedef struct PimMAIN {
        MfPopupPaneT  menubar;              // La barre des menus
        MfPopupPaneT  paneToolsMenu;        // La barre d'outils
        PimBUTTON_BAR *pButtonBar;          // La barre des boutons
        PimPERIOD_BAR *pPeriodBar;          // La barre période
        PimDATE_BAR   *pDateBar;            // La barre des dates
        PimPANE       *ppanearea;           // La fenêtre graphique
        PimSCROLL     *pscrollH;            // La barre de défilement
        Widget        wTeStatistics;        // Section des statistiques
        Widget        wTeAudit;             // Section des protocoles de traçage
        Widget        wTeAlerts;            // Section des messages d'erreurs
        MfPanel       *panel;               // Données de gestion du panneau
        PitSCENARIO   *psc;// Données du scenario courant (NULL si pas ouvert)
        PiuERRLOG     errlog;               // Tampon des messages d'erreurs
        PiuAB_HANDLE  abh;                  // Tampon des protocoles de traçage
        PimOPTIONS    options;              // Options d'affichage
        void          *pUndoBuffer;         // Tampon d'annulation
        F             fUpdateTasks:1;       // Indicateur de mise à jour
        F             fRefreshGraphics:1; //Indicateur rafraîchissement
        F             fModified:1;          // Indicateur d'état modifié
        I             iCyclesDisplayed;     // Les périodes cycliques
        SZ            szDayDashes;          // Séparateur de date
        SZ            szHourDashes;         // Séparateur d'heures
    } PimMAIN;
```

Figure 3.11 Détail de la structure *PimMain*

La figure 3.12 décrit de façon schématique comment la structure *PimMAIN* peut être divisée selon les indications présentées plus haut.

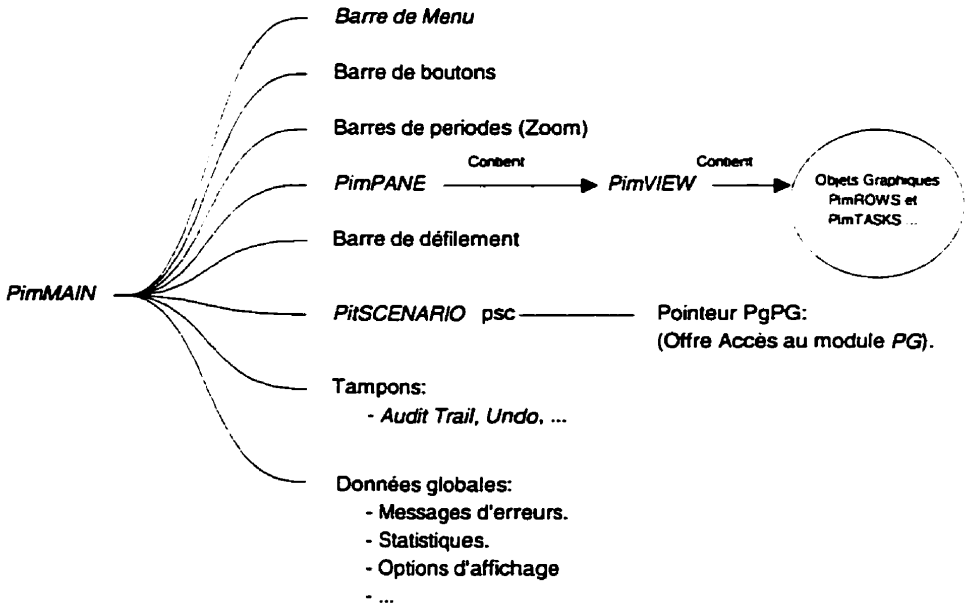


Figure 3.12 Contenu de la structure principale

3.2.2.1 Fenêtres, lignes et objets graphiques

Le panneau central englobe une région d'édition graphique qui peut contenir plusieurs fenêtres. Une fenêtre est une région où les objets sont affichés à l'écran. Chaque fenêtre est divisée en lignes horizontales contenant les objets graphiques. Comme on le sait déjà, un objet et une représentation graphique d'un élément de données, pouvant être une rotation, un vol ou un bloc (journée de travail).

Un objet donné peut avoir plusieurs représentations graphiques simultanées différentes qui se trouvent dans deux fenêtres, où dans plusieurs endroits à l'intérieur de la même fenêtre. De plus, chaque objet peut posséder des sous-objets (comme par exemple un bloc

ou une rotation possède un ensemble de sous-objets représentant les cellules). Les structures de données à ce niveau suivent l'architecture décrite dans la figure 3.13.

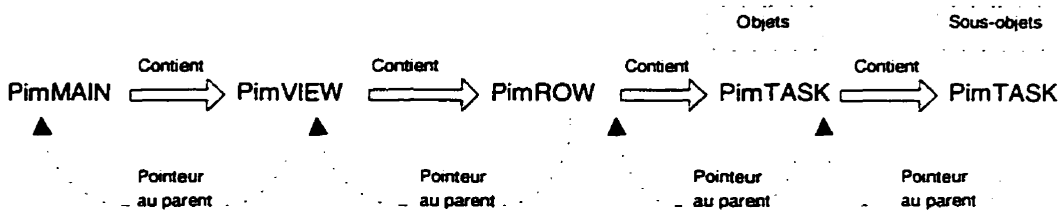


Figure 3.13 Hiérarchie des structures

3.2.2.2 Objets opposés à Données

Nous avons discuté plus tôt, de la distinction entre les données actuelles et leurs représentations graphiques. En effet, le module interface est séparé du module gestionnaire de données. Ceci nécessite donc la présence de structures intermédiaires contenant une copie des informations spécifiques aux éléments et nécessaires pour leur affichage. Comme les informations d'affichage sont semblables pour tous les éléments, nous considérons une seule structure commune pour tous les types d'objets (*PidPgLink*). Cette structure contient toujours un pointeur vers la structure réelle de l'objet, que ce soit un PgBLOCK, un PgLEG, un PgSTATION, ou n'importe quel autre élément. Les informations pertinentes à l'affichage sont gardées dans *PidPgLINK_INFO* à l'intérieur de *PidPgLink*. Par contre, cette même structure possède aussi un tableau de série de pointeurs vers des *PimTASK*, résultant du fait que le même objet peut avoir plus qu'une représentation graphique. À chaque représentation correspond une structure *PimTASK* différente.

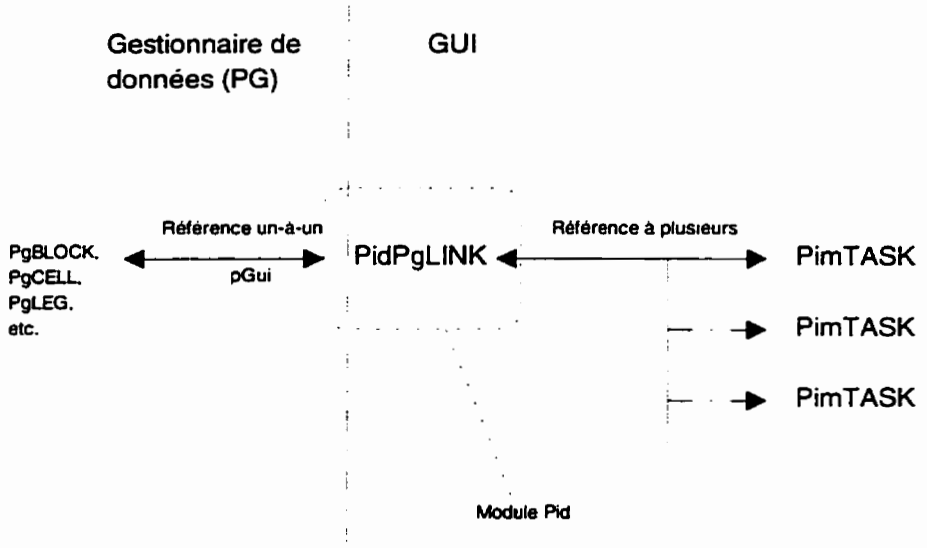


Figure 3.14 Module intermédiaire de l'interface

Le module qui gère ces informations est appelé *Pid* (*Pairing Interface Data*) (fig. 3.14). Le rôle que joue ce niveau, offert par le module *Pid*, est de préserver la symétrie entre les objets graphiques et les structures de données. Il offre une interface constante entre les éléments de données et leurs représentations. Dans le gestionnaire de données (*PG*), tout élément de données possède un pointeur opaque (*void *pGui*) utilisé comme pointeur vers l'arrière à *PidPgLINK* utilisé pour la mise à jour que nous discuterons plus loin dans ce chapitre. Ce pointeur aide le module gestionnaire (*PG*) à opérer indépendamment à partir du *GUI*.

3.2.2.3 Technique des opérations graphiques

Parmi la vaste liste des opérations graphiques, on distingue:

- les opérations utilitaires;
- les opérations de dessin et d'affichage;

- les opérations de déplacement (*Drag and Drop*);
- les opérations manuelles graphiques.

Les opérations utilitaires sont les opérations de base qui servent de support aux autres opérations. Ce sont principalement les fonctions ou les méthodes de libération des fenêtres d'affichage, d'effacement d'objets, etc. Une fonctionnalité qui doit être mentionnée est celle de la recherche d'objet graphique sous le curseur (fig. 3.15).

```
fFindTask(    PimMAIN      *pmain,
              int x, y,    /* Position du curseur */
              PimTASK,      *pTaskTop,    /* Objet sous curseur */
              PimTASK,      *pTaskPrim    /* Sous objet */
              );
```

Figure 3.15 Structure de recherche d'objet graphique sous le curseur

La structure se lit comme suit:

1. déterminer la fenêtre d'affichage parmi les *PimVIEW* dans *PimMAIN*;
2. obtenir l'ordonnée *y* de la position du curseur;
3. soustraire (Première ligne X Hauteur de ligne);
4. diviser par Hauteur de ligne, ce qui donne le numéro (position) de ligne;
5. obtenir l'abscisse *x* de la position du curseur;
6. convertir en minutes en divisant par la largeur de fenêtre, multipliant par (24x60), ce qui donne la position en temps;
7. parcourir tous les objets sur cette ligne, en comparant le temps de début de l'objet et sa fin avec le temps *x*;

a) si trouvée la fonction, retourne la valeur 1, l'objet est représenté par *pTaskTop*;

b) sinon la valeur 0 est retournée, indiquant qu'il n'existe pas d'objet sous le curseur.

Une autre fonction détaillée à la figure 3.16, vérifie si les deux objets se chevauchent ou non et les situent relativement l'un à l'autre:

```
#define STARTS_BEFORE 0x1

#define ENDS_BEFORE 0x2

#define OVERLAPS 0x4


BF      CheckTaskOrder(      PimTASK *pimTask1,

                             PimTASK *pimTask2

                       );
```

Figure 3.16 Structure de vérification de chevauchement

La figure 3.16 indique qu'on doit obtenir les temps de début et de fin des deux objets, des structures *PidPgLINK_INFO* correspondantes, et ensuite on engendre les comparaisons illustrées dans la figure 3.17. Les 3 bits sont livrés par la fonction à l'intérieur de l'octet *BF*.

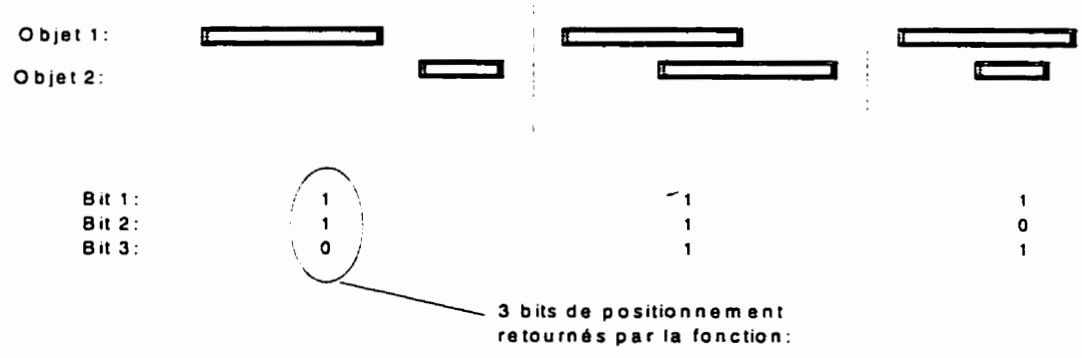


Figure 3.17 Fonction de chevauchement des objets

Le schéma de la figure 3.17 indique les possibilités suivantes:

1. Si le début de l'objet 1 se situe avant le début de l'objet 2, la valeur 1 est assignée au *Bit 1*. Si la fin de l'objet 1 se situe avant la fin de l'objet 2, la valeur 1 est assignée au *Bit 2* (1,1,0).

2. Même chose qu'en 1, sauf qu'il y a chevauchement (1,1,1).

3. Si le début de l'objet 1 se situe avant le début de l'objet 2, la valeur 1 est assignée au *Bit 1*. Dans ce cas, le chevauchement est automatique et la valeur 1 est assignée au *Bit 3* (1,0,1).

4. Si le début de l'objet 1 se situe après le début de l'objet 2, la valeur 0 est assignée au *Bit 1*. Si la fin de l'objet 1 se situe après la fin de l'objet 2, la valeur 0 est assignée au *Bit 2*. S'il y a chevauchement, la valeur 1 est assignée au *Bit 3* (0,0,1).

5. Même chose qu'en 4, excepté qu'il n'y a pas de chevauchement (0,0,0).

6. Si le début de l'objet 1 se situe après le début de l'objet 2, la valeur 0 est assignée au *Bit 1*. Si la fin de l'objet 1 se situe avant la fin de l'objet 2, la valeur 1 est assignée au *Bit 2*. Dans ce cas, le chevauchement est automatique et la valeur 1 est assignée au *Bit 3* (0,1,1).

Les opérations d'affichage sont principalement les fonctions en charge de dessiner n'importe quel type d'objet graphique à l'écran. Elles assignent aussi les couleurs respectives aux objets, ainsi que le texte descriptif qui est affiché à l'écran.

Les opérations de déplacement se servent des fonctions utilitaires pour déplacer les objet graphiques d'une fenêtre à l'autre ou à l'intérieur de la même fenêtre. Le méthode utilisée est la suivante:

1. obtenir l'objet sélectionné. Il est sauvegardé dans une structure temporaire au début de l'opération;

2. déterminer la ligne de destination (indiquée par le curseur de la souris);

3. utiliser la fonction de chevauchement, détaillée dans la première section, pour déterminer s'il y a un chevauchement avec un objet au site de placement;

   si oui, on le place sur une nouvelle ligne:

   a) on alloue une structure de ligne *PimROW*;

   b) on initialise sa position (Index) dans *PimVIEW*;

   c) on initialise sa liste d'objets à vide (nouvelle ligne);

   d) on insère dans la fenêtre.

4. insérer l'objet de la structure temporaire dans la liste des objets de la ligne.

5. enlever l'objet de l'ancienne ligne d'origine. Si l'ancienne ligne est vide, l'effacer et la libérer;

   si une nouvelle ligne est créée:

   a) redessiner la fenêtre de destination avec la nouvelle ligne insérée;

   sinon:

   a) redessiner la ligne de destination avec les nouveaux objets seulement;

   si l'ancienne ligne est effacée:

   a) redessiner la fenêtre d'origine sans la ligne retranchée;

   si non:

   a) redessiner ligne d'origine sans les objets retranchés.

Les opérations graphiques manuelles sont les opérations qui peuvent provoquer des modifications à la solution. Ce sont définitivement les opérations les plus importantes. L'approche générale est présentée dans ce qui suit:

1. identifier les objets impliqués dans l'opération;

2. obtenir les pointeurs vers les structures des éléments de données dans *PidPgLINK*;
3. passer les pointeurs dans les fonctions du gestionnaire de données *PG*. La fonction du gestionnaire *PG* implémente l'opération. Généralement le gestionnaire peut créer des nouveaux éléments;
4. passer les nouveaux éléments, si besoin, dans le filtre du gestionnaire *PG* pour validation et calcul de statistiques;
5. créer les nouvelles structures *PimTASK* correspondantes;
6. enlever les anciens objets (d'entrée) des lignes et des fenêtres d'affichage;
7. ajouter (afficher) les nouveaux objets;
8. détruire les structures des objets d'entrée. Mettre à jour les listes des *PimTASK* dans les lignes *PimROW*;
9. générer les messages de protocole de traçage;
10. mettre à jour les statistiques globales si l'un des objets d'entrée ou de sortie est une rotation;
11. mettre à jour le tampon de d'annulation de la dernière opération (*Undo*);
12. libérer les structures d'entrée *PidPgLINK* non utilisée;
13. redessiner les lignes impliquées.

3.2.2.4 Méthodes de défilement et de zoom

Le système offre les fonctions de défilement et de zoom pour permettre à l'usager de visualiser des régions graphiques spécifiques dans le panneau principal. La barre de défilement est utilisée pour se déplacer de haut en bas et de droite à gauche dans la fenêtre graphique. Le zoom permet d'agrandir une région spécifique de la fenêtre graphique présentant les objets plus en détails.

Une barre de défilement a besoin d'une entité d'attachement. Après avoir créé la barre de défilement, on peut y attacher l'entité qu'on appelle client. L'entité est généralement une fenêtre ou une région d'affichage.

À chaque fois qu'une valeur correspondant à la barre de défilement est modifiée par l'usager, l'entité (région graphique) est avisée grâce à une fonction de mise à jour initialisée au moment de l'attachement. Un indicateur *fDrag* avisera l'entité que la barre est en mode de déroulement (*drag mode*) pour qu'elle puisse faire un rafraîchissement rapide de la région de dessin. Plusieurs entités ou régions de dessin peuvent être attachées à la même barre de défilement. Dans ce cas, toutes les entités sont avisées en même temps.

Les unités de la barre de défilement (fig. 3.18) sont définies par la région de dessin. Ça peut être des pixels, des minutes, des heures, etc. L'idée est que du nombre maximal d'unités affichables (*iDisplayable*), seulement un nombre d'unités définies par *iDisplayed* sont couramment visibles à l'écran. La première unité visible est donnée par *iFirst*. Trois méthodes sont disponibles pour accéder à ces informations:

```
I iDisplayableGet(void); /* Détermine le total des unités affichables*/
I iDisplayedGet(void); /* Détermine le total des unités affichées*/
I iFirstGet(void); /* Détermine l'index de la première unité affichée*/
```

Figure 3.18 Fonction de calcul des unités

Pour que l'entité puisse rafraîchir l'affichage, elle doit aussi obligatoirement considérer ces deux spécifications: la **granularité** et la **ligne**.

La **granularité** est le montant le plus petit incrémentable par la barre de défilement, mesuré en unité. Initialiser la granularité à une certaine valeur entière déplace la barre en multiples de grains. La méthode utilisée pour définir la granularité est

SetGranularity(). Il faut noter que la valeur *iDisplayable* doit être un multiple entier de la granularité.

La **ligne** est le nombre d'unités que la région défile quand l'usager appuie sur les flèches en haut et en bas de la barre de défilement.

La région dessin (*pixmap*) est contenue dans la fenêtre d'affichage (*View*). Quand l'application a fini de dessiner les objets dans la région, ce dernier est copié dans la fenêtre d'un seul coup. Cette méthode de fonctionnement est adoptée pour éviter les vacillements désagréables à l'écran. Il est toujours possible de changer de méthode en initialisant l'indicateur *fXor* à 1 avec la fonction *SetXorMode(F fXor)*. Ceci permet de dessiner directement dans la région, en mode immédiat.

Les fonctions internes de dessin demandent toujours des coordonnées en pixels. Pourtant, la fenêtre graphique sert à représenter des objets en unités de temps-minutes afin d'être plus spécifique. Le déplacement des barres de défilement se rapportent aux unités d'affichage (minutes) et non aux pixels. Pour ce faire, on a besoin de la fonction de conversion (de minutes en pixels) décrit dans la figure suivante:

```
PimXY xyScreen(xy);     // convertir minutes (y) en pixels
PimXY xyWorld(xy);      // convertir pixels (y) en minutes
```

Figure 3.19 Fonctions de conversion de coordonnées

Voici quelques fonctions de dessin:

```
void SetColor(Pixel color);                 // Set the color
void SetLineThickness(I iThickness);        // Set line thickness
void SetLineDashes(SZ szDef);               // Set line dashes
```

```
    void SetLineSolid(void);                    // Clear line dashes
    void SetFont(XFontStruct *pxfs);            // Set the font
    void SetStipple(F f);                       // Turn on/off stipple
    void SetStipple(Pixmap pixmap, PimXY& xyOrg, Pixel pixelOther);
    void Line(PimXY& xyFrom, PimXY& xyTo);      // Draw a line
    void Lines(PimXY *rgxy, CO coSegs);         // Draw connected lines
    void Circle(PimXY& xyCenter,
I iRadius,I iDegStart = 0, I iDegStop = 360);     // Measure 3 o'clock
    void FillRectangle(PimXY xyTL, PimXY xyBR);
I iTextLeft(SZ szText, PimXY& xyBaseLeft, I iWidth = -1);
```

Figure 3.20 Fonctions de dessin

3.3 Mise à jour et réaffichage

Il est difficile de maintenir une certaine constance d'affichage et de mise à jour parce que plusieurs représentations simultanées du même objet graphique peuvent exister dans le panneau principal et dans les autres panneaux de l'application. Il faut donc s'assurer que la représentation graphique de l'objet reflète son état réel en tout temps.

Supposons, par exemple, qu'on a un objet affiché dans deux fenêtres différentes. Si on le change manuellement dans la première fenêtre, on devrait voir les changements se reproduire automatiquement sur le même objet dans la deuxième fenêtre. Le mécanisme implanté est décrit dans ce qui suit.

Quand une modification est faite à un élément de données dans le gestionnaire *PG*, celui-ci informe le module d'interface graphique (*GUI)* en appelant une fonction de mise à jour qui a été enregistré par le module de l'interface au début de l'exécution de

l'application. Cette fonction réside au niveau de la couche *Pid* (figure 3.14). Ses tâches sont les suivantes:

1. obtenir la structure *PidPgLINK* associée à l'élément de donnée dans *PG*;
2. si elle existe, informer tous les structures *PimTASK* qui y sont reliées (figure 3.14) du changement;
3. redessiner les objets graphiques impliqués;
4. générer un message au niveau de tous les panneaux de l'interface pour s'assurer de la mise à jour de la représentation des objets impliqués dans ces panneaux.

Cette méthode peut paraître assez compliquée à suivre dans le code mais elle présente les avantages suivants:

- chaque panneau de l'interface a une existence indépendante des autres panneaux, ce qui rend le système plus modulaire, plus flexible et plus facilement configurable;
- l'interface répond à des changements produits à l'extérieur de son module par la fonction de mise à jour, comme par exemple, dans le gestionnaire de données.

3.4 Conclusion

Le mécanisme d'affichage et de manipulation graphique est assez complexe et délicat à cause du nombre élevé d'objets graphiques qui peuvent être affichés dans la fenêtre. Les techniques développées sont efficaces et rapides, et permettent de diminuer le temps d'exécution des opérations graphiques. La majorité des techniques développées est concentrée sur la gestion des pointeurs aux structures des données correspondant aux éléments affichés, à cause du nombre assez élevé de ces derniers (dans l'ordre de milliers). Les opérations affectant les données sont effectuées séparément dans un module gestionnaire de données. De plus, entre le module gestionnaire de données et le module

d'affichage réside un mécanisme de mise à jour très efficace et très sophistiqué qui permet de rafraîchir rapidement la représentation des objets graphiques. Ceci permet d'offrir une grande robustesse à l'usager.

CHAPITRE IV:

EXEMPLES RÉELS ET MANIPULATION DES DONNÉES

Dans les chapitres précédents, nous avons discuté l'organisation, l'installation et l'implantation de l'interface graphique. L'objectif visé était de faciliter la tâche du planificateur en lui permettant de visualiser des horaires et en lui offrant la possibilité de les modifier de façon itérative.

Dans ce chapitre, nous discuterons l'utilité de l'interface graphique. Nous nous servirons de quelques exemples réels pour démontrer l'aide que l'interface graphique peut apporter aux planificateurs dans le domaine aérien.

4.1 Filtrage et contrôle du contenu des fenêtres

Le filtrage des objets graphiques dans les fenêtres est effectué à partir des panneaux filtres. Ces outils servent à sélectionner les données à afficher dans les fenêtres graphiques. Tous les objets proviennent originalement du gestionnaire de données PG. Donc, toutes les opérations de filtrage partent des fonctions de requête au niveau de PG. Ces fonctions sont parfois accompagnées d'un ensemble de critères de sélection qui permettent d'afficher une partie des éléments chargés en mémoire. On distingue deux types d'éléments qui peuvent être filtrés: les rotations et les segments de vols. Les autres types d'objets (groupes, service de vols, etc.) sont des éléments intermédiaires générés à partir des deux types d'éléments filtrables qui ne sont pas gardés en mémoire tant qu'ils ne sont pas affichés.

4.1.1 Filtrage des rotations

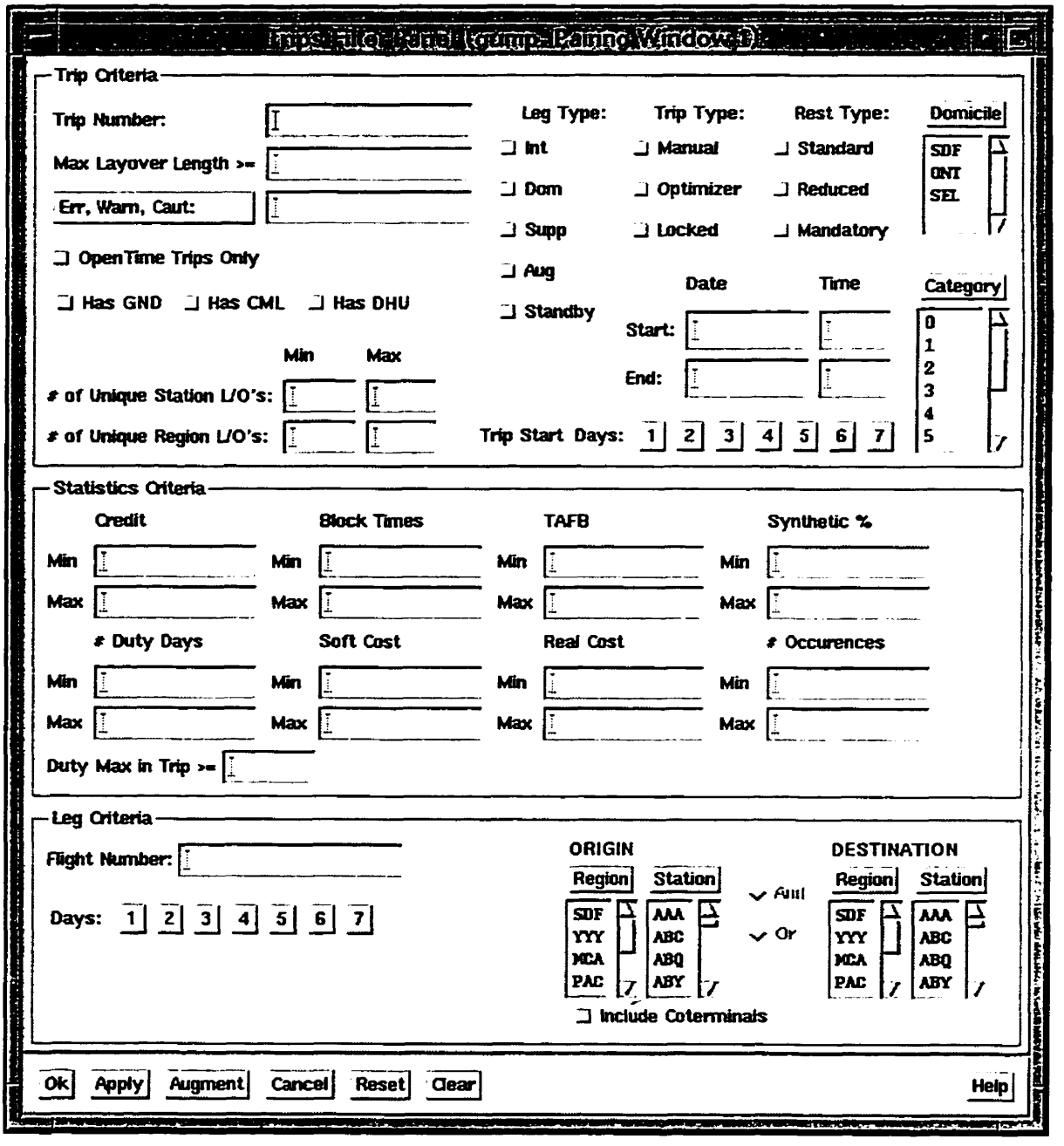


Figure 4.1 Panneau filtre des rotations (*Trips Filter Panel*)

À l'ouverture du scénario, toutes les rotations formant la solution sont affichées par défaut dans la première fenêtre, tandis que la deuxième fenêtre reste vide. Le panneau filtre des rotations(*Trips Filter Panel*) offre des critères de sélection permettant à l'usager de contrôler l'ensemble des rotations contenues dans la fenêtre. Les critères de sélection dans le panneau de la figure 4.1 se divisent en trois catégories: segments de vol, rotations et statistiques. La première catégorie s'applique aux segments de vols à l'intérieur des rotations. Les deux autres s'appliquent aux rotations comme entités

Les critères de sélection spécifiques aux rotations comprennent les numéros de rotations, les types de rotations (manuelle, optimisée ou verrouillée), la base, etc. Tous ces items sont regroupés dans la partie supérieur du panneau de filtre des rotations.

La section centrale du panneau contient les critères de statistiques: coût , durée de vol, crédits, synthétique, etc. Tous ces critères sont dotés de deux champs dont les valeurs déterminent les bornes inférieure et supérieure des statistiques qui correspondent à la rotation.

La section des tronçons de vol, comprend tout ce qui est en rapport avec les vols formant la rotation. Il y a entre autres, les numéros de vol, les stations de départ et d'arrivée et la journée de la semaine à laquelle le vol correspond. Il suffit qu'un seul vol de la rotation réponde aux critères pour que la rotation soit sélectionnées et affichée.

La section au bas du panneau contient des boutons-poussoirs qui servent à déclencher le processus de filtrage.

Une fonctionnalité importante à mentionner est la mémorisation de l'état du panneau, c'est-à-dire, l'ensemble des critères sélectionnés d'une opération à l'autre. La première fois que le panneau apparaît, il est vide. Les critères sélectionnés pendant l'opération sont gardés dans une structure tampon (*PimTRIPS_FILTER_DATA*) qui est liée à la structure du panneau principal *PimMAIN* quand le panneau filtre est inactif. La structure contient plusieurs champs qui correspondent à tous les critères de sélection du

panneau. Elle sert à initialiser l'état du panneau quand l'usager y accède pour la deuxième fois, et ainsi de suite.

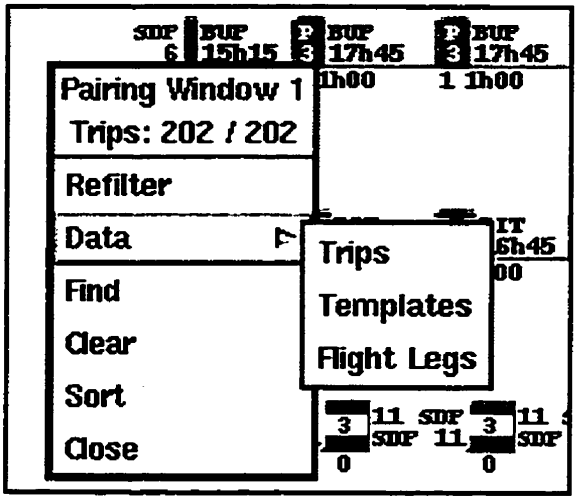


Figure 4.2 Menu de fond pour la fenêtre des rotations

Avant de commencer toute procédure de filtrage d'objets graphiques on doit accéder au panneau. L'accès se fait par un menu de fond qu'on sélectionne dans l'une ou l'autre des fenêtres graphiques (figure 4.2). Après avoir choisi tous les critères désirés, l'usager a le choix d'appliquer le filtre ou de fermer le panneau et d'annuler l'opération. Les boutons poussoirs au bas du panneau déterminent la nature de l'opération filtre. Le bouton *Apply* (appliquer) se charge de vider la fenêtre graphique et de remplacer le contenu par les rotations résultant du filtrage. Le bouton *Augment* (Augmenter) insère les rotations résultant du filtrage à la fin de la liste déjà affichée dans la fenêtre. Le bouton *Reset* servira à remettre le panneau à l'état de la dernière requête ou application. Le bouton *Clear* annule toutes les sélections que l'usager a fait dans le panneau.

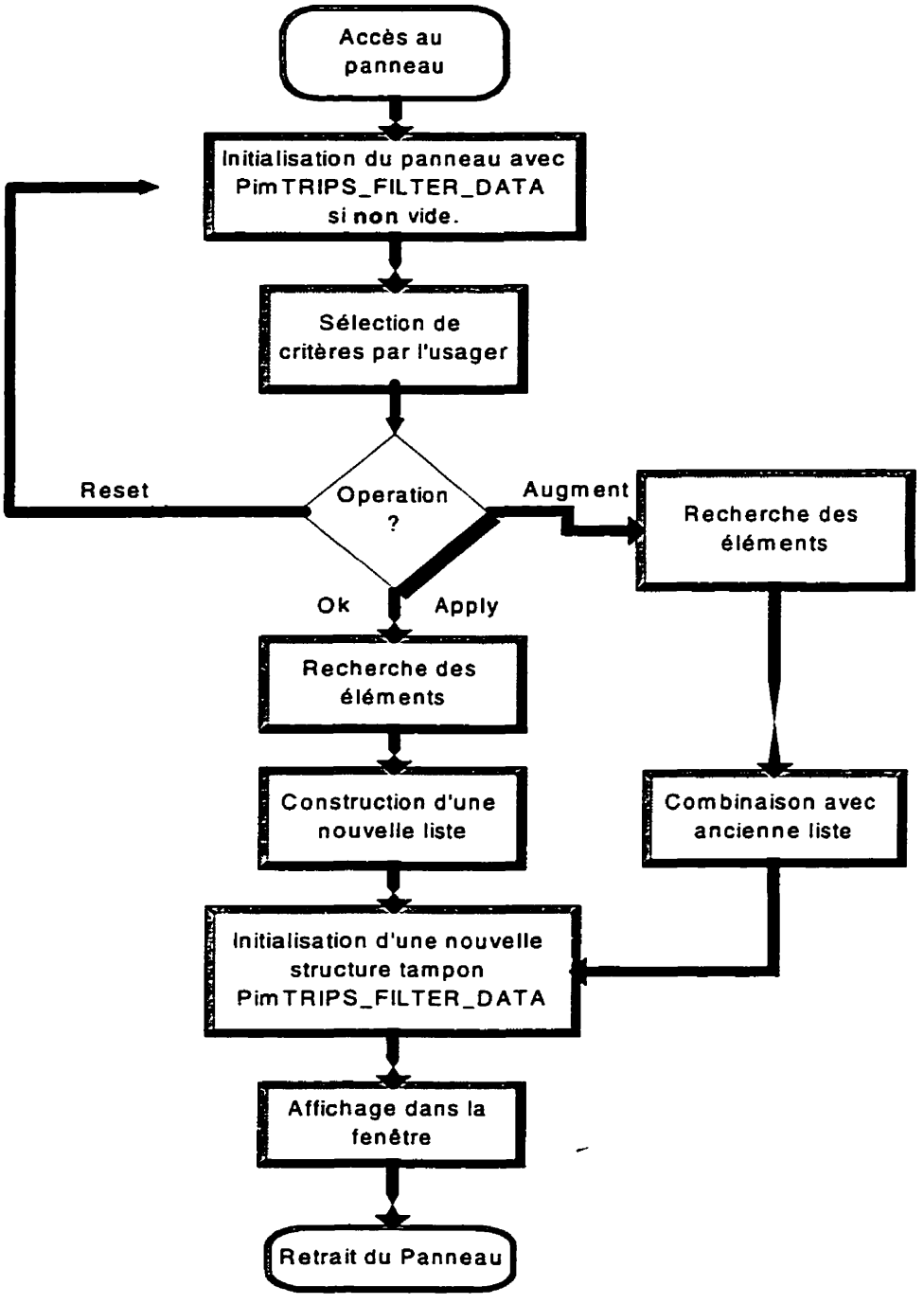


Figure 4.3 Organigramme de l'opération filtre

Pour l'opération de filtrage, la recherche se fait au niveau du module gestionnaire de données. Tous les blocs de rotations (*PgBLOCK*) sont parcourus. La fonction de requête livre les éléments qui répondent aux critères spécifiés par l'usager. Les blocs livrés sont traités au niveau *Pid* de l'interface qui forme les structures *PidPgLINK* servant à l'affichage des objets. Si l'opération est du type *Apply*, les anciennes structures sont détruites et remplacées par les nouvelles. Dans le cas d'une opération de type *Augment*, les nouvelles structures sont combinées avec les anciennes (figure 4.3) pour former une liste étendue des objets. Et dans les deux cas, les objets sont affichés dans la fenêtre graphique où le panneau filtre a été appelé.

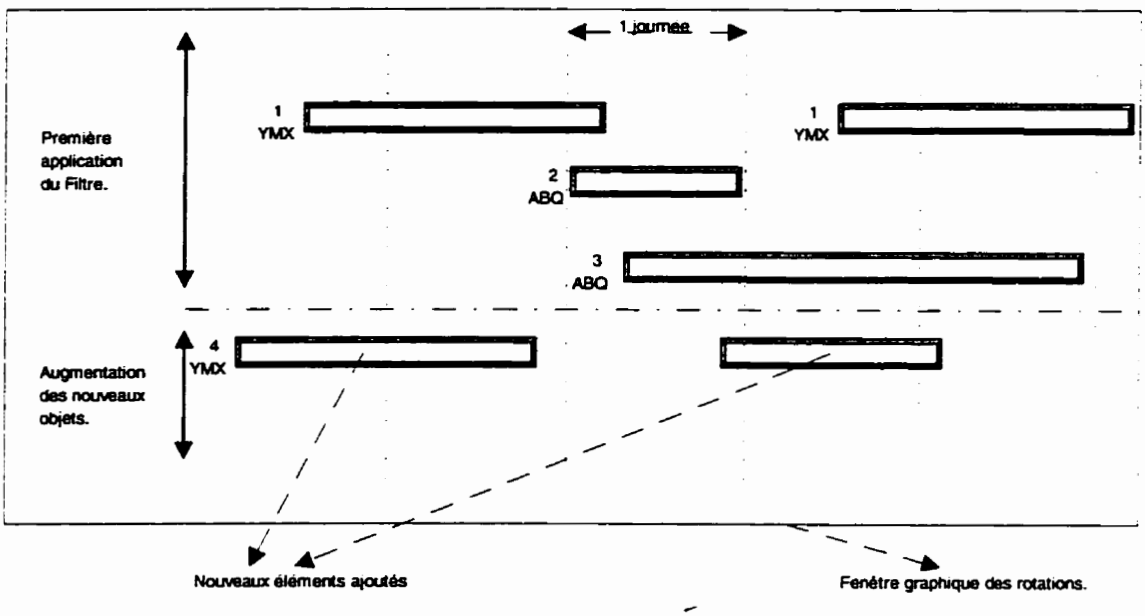


Figure 4.4 Ajout de nouveaux éléments avec le filtre

Supposons, par exemple, que l'usager désire afficher toutes les rotations qui contiennent un vol commercial. Après avoir accédé au panneau, il appuie sur le bouton de sélection *Has CML* dans la section des critères de rotations. Il lance ensuite la requête en appuyant sur *Apply* ou *Ok*. Si par après, l'usager désire ajouter des rotations qui possèdent un coût supérieur à 1500 ($) dans la même fenêtre, il insérera la valeur dans le champ de coût minimal réel (*Min Real Cost*) en laissant le champs de la borne supérieure vide pour indiquer qu'il n'y a pas de limite. En appuyant sur *Augment*, les rotations sélectionnées seront affichées à la fin de la liste déjà existante dans la fenêtre graphique (figure 4.4).

4.1.2 Filtrage de segments de vol

Comme on le sait déjà, les tronçons de vol sont les éléments de base pour la construction des rotations. Comme l'ensemble des données dans un scénario est immense, l'affichage de tous les segments de vol dans la fenêtre graphique n'est pas d'une grande utilité pour l'usager. Le système offre donc le panneau de filtrage des segments de vol (figure 4.5, *Legs Filter panel*), un outil qui permet de sélectionner et de limiter le nombre de vols affichés en même temps dans la fenêtre graphique. L'usager peut donc filtrer et garder seulement les catégories de vols spécifiques.

Un vol peut appartenir à l'une des catégories suivantes:

- un vol actif de la compagnie;
- un vol passif qui sert comme mise en place interne;
- un vol commercial qui sert comme mise en place externe;
- une séquence de vols commerciaux générée par le générateur de mise en place externe.

Un vol actif peut être couvert, non couvert, international, domestique, etc. Il est possible pour l'usager de visualiser les différents types de vols actifs indépendamment. En effet, le panneau filtre contient un bouton de sélection pour chacune des catégories mentionnées ci-dessus. Le sélecteur de stations dans la section en haut, à gauche du

panneau, permet de sélectionner les vols selon les stations de départ et d'arrivée. Les champs correspondant aux dates et aux heures à la droite du panneau permettent de sélectionner les vols pour un certain intervalle de temps. Si cette section est vide, toute la période correspondant au scénario sera considérée. Plus bas, dans la section à droite du panneau, réside le champs de numéro de vol qui permet d'entrer les numéros de vols à afficher. On peut entrer une liste de vols, et cette liste peut également contenir des intervalles de numéros. Par exemple, l'entrée «101, TWA0231, 300-400» affichera le vol 101 de la compagnie aérienne courante, le vol commercial TWA0231 et tous les vols de la compagnie dont le numéro se trouve entre 300 et 400. On peut aussi remplacer les objets affichés ou le remplacer, comme dans le panneau filtre des rotations.

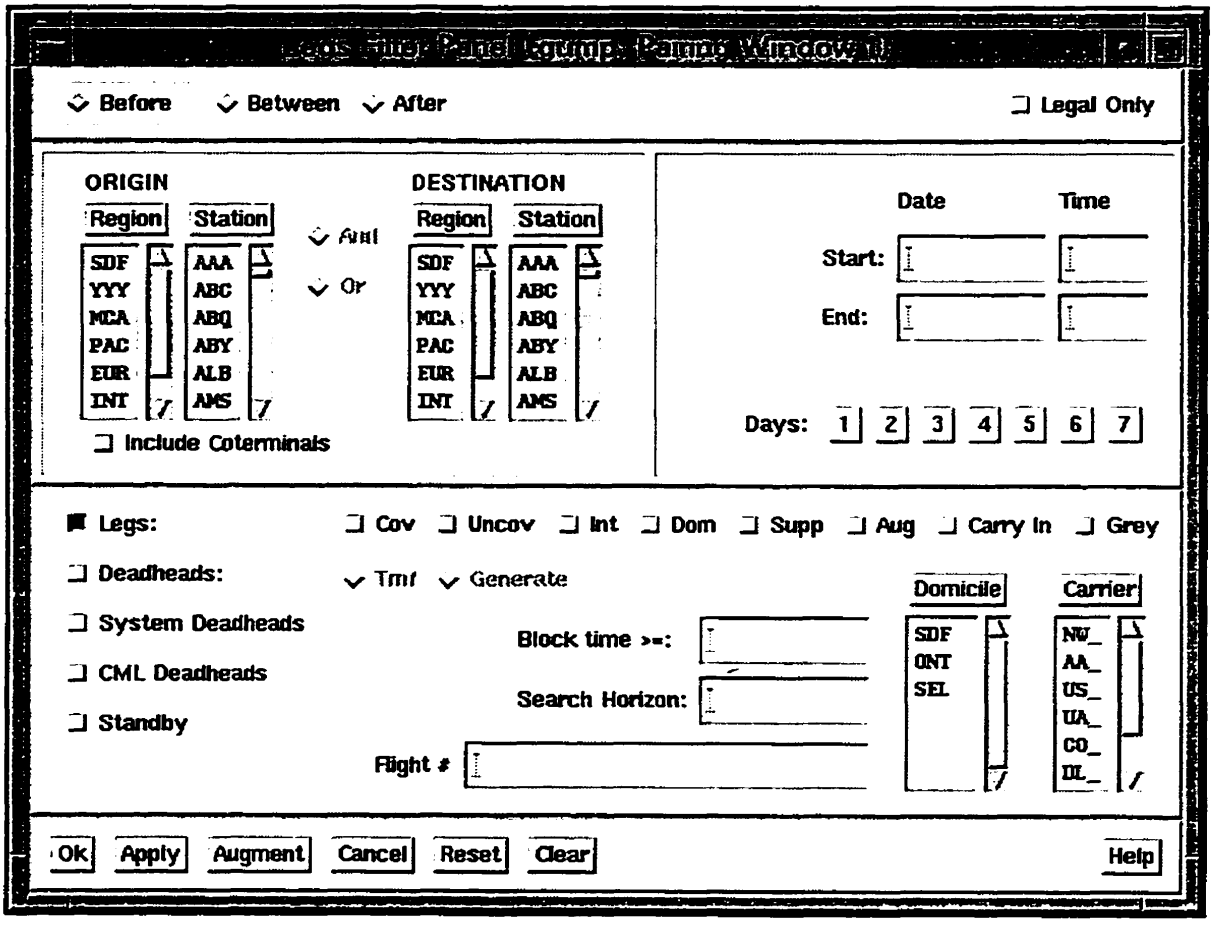


Figure 4.5 Panneau filtre de tronçons de vols (*Legs Filter Panel*)

Le panneau filtre est aussi doté de quelques foncitonnalités spéciales très utiles pour la construction manuelle de rotations. L'usager peut aussi tenter de trouver un segment de vol qui précède celui sélectionné dans le panneau principal, en choisissant le bouton *Before*. Il a aussi la possibilité d'initier une recherche afin d'insérer un segment entre deux vols. Il s'agit tout simplement de sélectionner deux objets dans la fenêtre graphique pour que le bouton *Between* s'active automatiquement dans le panneau filtre (fig. 4.6). Le système prend en considération l'arrivée du premier objet et le départ du second pour initialiser les champs respectifs.

Ces fonctionnalités sont exécutées en sélectionnant les objets qu'on désire compléter dans la fenêtre graphique, et en les faisant interagir avec les boutons identifiés *Before* (avant), *After* (après) et *Between* (entre) qui sont situés dans la partie supérieure du panneau. Ces boutons déterminent dans quelle direction la rotation doit être complétée. L'usager peut exiger que l'opération ne produise que des vols légaux, en sélectionnant le bouton *Legal Only*. Le choix des rotations sera plus limité, mais la légalité de la rotation est garantie.

Le panneau filtre s'initialise automatiquement selon les objets sélectionnés dans le panneau graphique. En d'autres mots, si un ou deux objets sont sélectionnés dans le panneau principal, les champs de stations de départ et d'arrivée, ainsi que les dates de début et de fin de rotation sont automatiquement initialisés lorsqu'on accède au panneau filtre. Prenons, par exemple, un vol partant de la base SDF en direction de la station ABQ, le 3 mars à 10h00. Ce vol est déjà sélectionné dans la fenêtre graphique. Lorsque l'usager accède au panneau filtre, le bouton *After* est activé ainsi que la station ABQ dans le sélecteur de stations de départ pour le vol qui suivra. Les champs de date et d'heure de départ du vol sont initialisés avec la date (3 mars) et l'heure d'arrivée (10h00) du tronçon sélectionné dans le panneau principal.

Afin de mettre au point une telle fonctionnalité, une certaine communication entre les panneaux est nécessaire. La communication entre le panneau filtre et le panneau graphique est faite à l'aide d'une technique de messagerie. Le panneau filtre envoie un

message au panneau graphique en lui demandant de livrer l'ensemble des objets choisis par l'usager et de diriger les pointeurs vers leurs structures correspondantes:

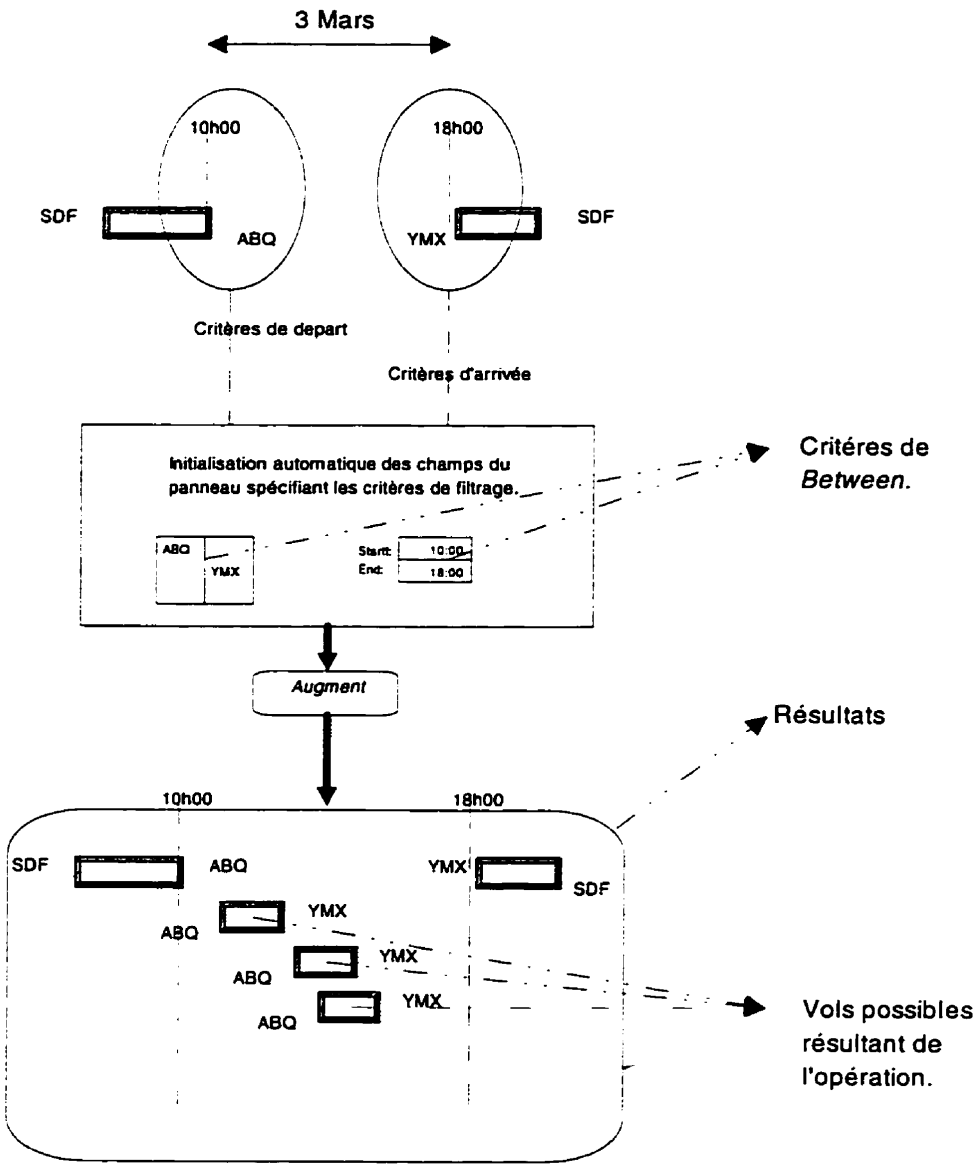


Figure 4.6 Recherche de vols complémentaires

```
PitMessageDeliver(MAIN_PANEL, MsgPIM_GET_SELECTED_TASKS,
                  PimTASK       ***prgpTasks,      // Tableau d'objets.
                  CO            *coTasksSelected   // Nombre sélectionné.);
```

Le champs *coTasksSelected* aide le système à déterminer le nombre d'objets sélectionnés, quel type de bouton de recherche il faut activer (*After* si un objet sélectionné, *Between* si deux, aucun si plus de deux objets sélectionnés). Les structures de données contenues dans *rgpTasks* permettent la validation des blocs.

Le but est de toujours bâtir une rotation légale. L'usager doit activer le bouton *Legal Only* situé au haut du panneau, afin que le système puisse accomplir la tâche de validation des blocs. Dans ce cas, tous les segments de vol retournés par la requête devraient pouvoir former un bloc légal lorsqu'ils sont combinés avec les éléments sélectionnés dans la région graphique. Les autres tronçons sont rejetés par l'opération de filtrage.

Le système suit la procédure suivante:

1. obtenir la liste des objets graphiques sélectionnés dans le panneau principal;
2. lancer la fonction de requête au niveau du gestionnaire de données (*PG*);
3. parcourir la liste des segments de vol retournés:
   a) combiner le vol avec la liste des objets pour former un bloc (*PgBLOCK*);
   b) vérifier la légalité (validation);

      si légal:

         garder le vol dans liste de retour;

      si non légal:

         rejeter et libérer la structure.

4. sélectionner le vol suivant dans la liste, et ainsi jusqu'à la fin de la liste;

5. afficher les segments de vol légaux dans le panneau graphique.

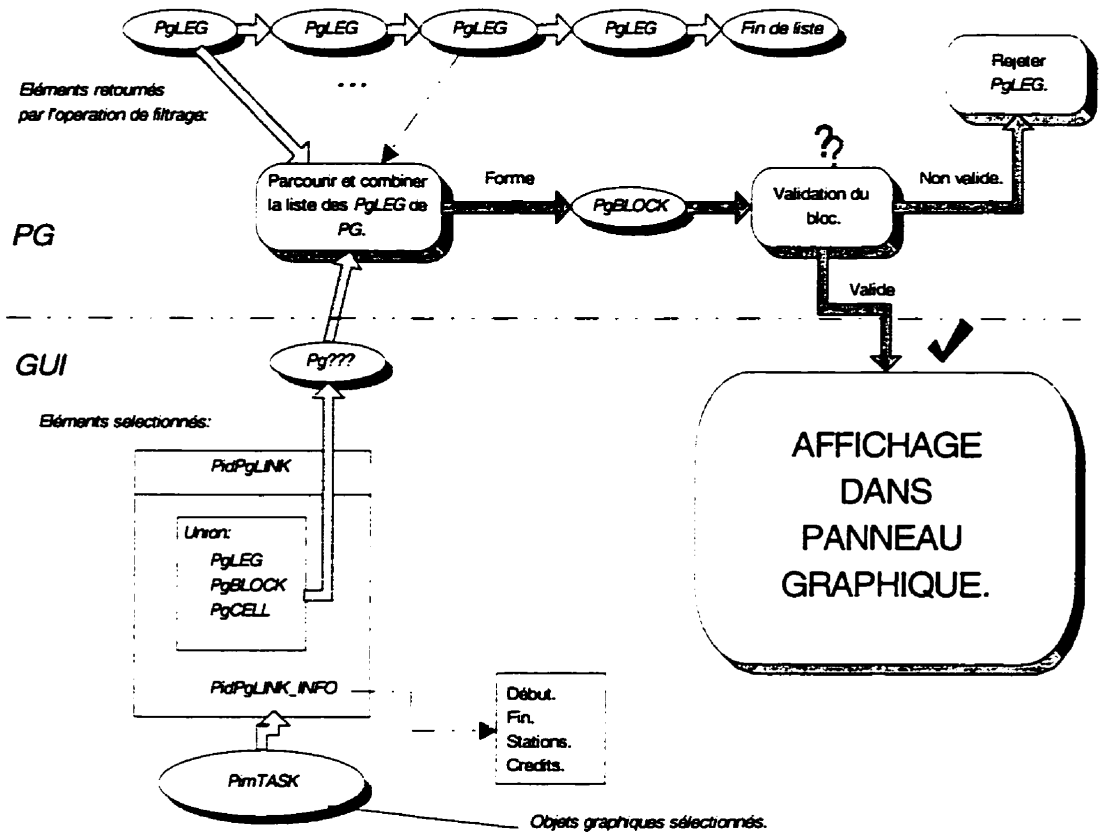


Figure 4.7 Filtrage des vols avec validation.

Un bloc (*PgBLOCK*) est, comme nous l'avons déjà vu, une séquence de vols (*legs*). Pour former une rotation, il faut donc accéder à une série de structures *PgLEG* pour les combiner la séquence de vol produite par l'opération de filtrage. Les structures *PimTASK*, que nous avons expliquées dans le chapitre 3, donnent accès aux structures au niveau *PG* grâce à des pointeurs (figure 4.7). La fonction de validation est codée au

niveau de PG, par contre, le module de l'interface est responsable d'accumuler les structures *PgLEG* en suivant les pointeurs dans les *PimTASK*, et de former la rotation (ou le bloc) pour ensuite utiliser le gestionnaire pour la valider à nouveau.

4.1.3 Possibilité de trier (*Sort*)

Un autre outil de contrôle des objets graphiques dans le panneau principal est le panneau de tri. C'est un panneau qui contient une liste prédéfinie de critères de tri définissant dans quel ordre les objets doivent être affichés dans la fenêtre. Les éléments peuvent être triés par numéro de rotation, numéro de vol, temps de bloc, temps de crédit etc. L'usager définit les différentes priorités en établissant l'ordre des critères dans la liste. Le réorganisation des objets est effectuée quand l'usager appuie sur le bouton *Apply* ou *Ok*.

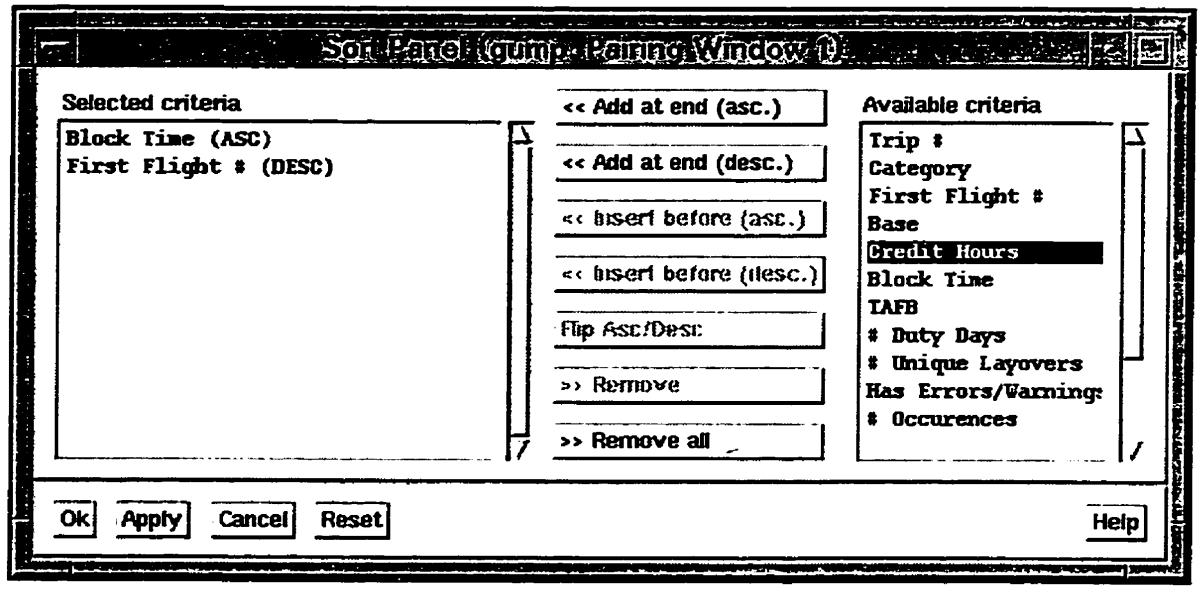


Figure 4.8 Panneau de tri (*Sort Panel*)

4.2 Résolution des conflits des mises en place internes

Les mises en place internes représentent l'usage des vols internes par les membres d'un équipage qui voyagent en tant que passagers. Dans le panneau graphique, on les représentent comme les segments de vols réguliers mais d'une couleur différente (figure 4.9), que l'usager aura spécifié dans le panneau des couleurs (voir Annexe A).

Tous les vols de la compagnie, même ceux d'une autre flotte, peuvent être utilisés pour la mise en place interne. Le nombre de sièges occupés est déterminé par le nombre des membres de l'équipage. Quand le nombre de siège utilisé est supérieur au nombre disponible sur l'avion, il en résulte un conflit de mise en place. Dans un cas de conflit, il faut rediriger certains membres d'équipage sur un autre vol pour résoudre le conflit. Le panneau de mise en place sert donc à lister l'utilisation de tous les vols internes pour la mise en place et ainsi indiquer les conflits. Dans la fenêtre graphique, les rotations qui sont en conflit sont identifiés par un cercle d'illégalité affiché à leur centre (figure 4.9).

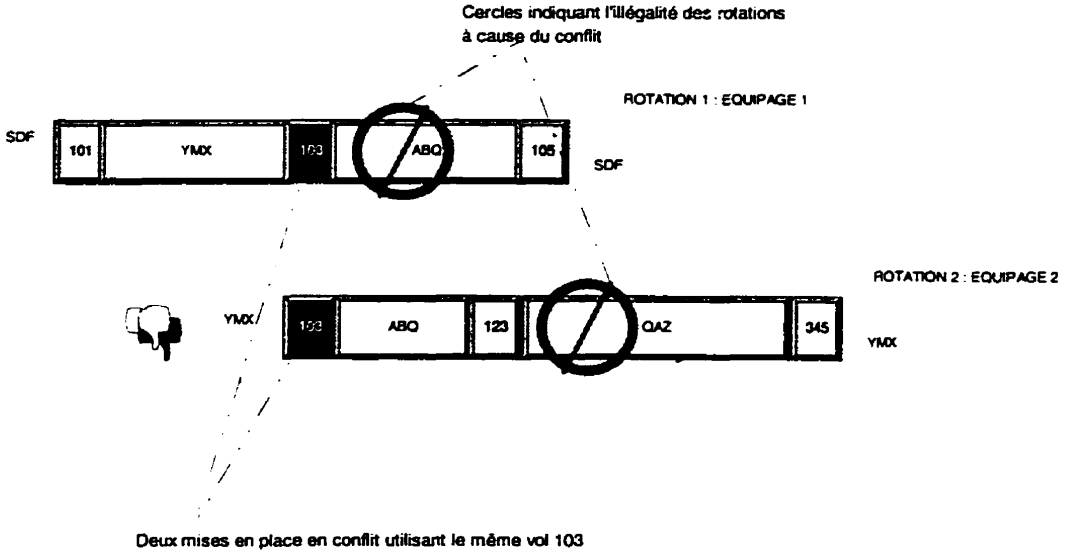


Figure 4.9 Exemple de conflit de mise en place dans le panneau graphique

4.2.1 Structures de données

Les structures de mise en place sont contenues dans le gestionnaire de données (PG) sous le pointeur *PgJSEAT *pjseatHead*; (fig. 4.10)

Ce pointeur représente l'entête de la liste en chaîne contenant toutes les mises en place de la solution. Toute structure *PgJSEAT* contient un regroupement de mises en place utilisant le même vol pointé par *PgLEG*. Le nombre de sièges disponibles sur ce vol se trouve dans le champs identifié par *coSeatsAvailable* dans *PgLEG*.

```
struct PgJSEAT {
  PgJSEAT    *pjseatNext;  /* Liste chaînée */
  PgLEG      *pleg;        /* Vol utilisé */
  PgJSOL *pjsolHead;       /* mise en place utilisant le vol */
  CO         coSeatNb;     /* Nombre total de sièges utilisés */
};
```

Figure 4.10 Contenu de la structure *PgJSEAT*

A chaque fois qu'un nouvel équipage (rotation) utilise un vol déjà assigné, une nouvelle structure *PgJSOL* est ajoutée à la liste dans *PgJSEAT*:

```
struct PgJSOL {
  PgJSOL     *pjsolNext;          /* Liste  chaînée */
  PgCONTEXT     *pcontext;        /* Flotte */
  PgCENT     centCostRepl; /* Coût de remplacement */
  CO            coSeatUsed;      /* Nombre de sièges utilisés */
  CH   rgszFlight[lPgJSEAT_REPLACE_NB][lPgFLIGHT_NUM_LEN];
                                 /* Vols de remplacement */
};
```

Figure 4.11 Contenu de la structure *PgJSOL*

4.2.2 Utilisation du panneau de mise en place

Le panneau de mise en place (figure 4.12) fonctionne en mode immédiat. Il permet à l'usager de manipuler les données et de refléter les changements produits immédiatement dans la fenêtre graphique (voir [6]). Le contenu de la fenêtre peut aussi affecter le contenu du panneau de mise en place, si l'usager modifie des rotations manuellement.

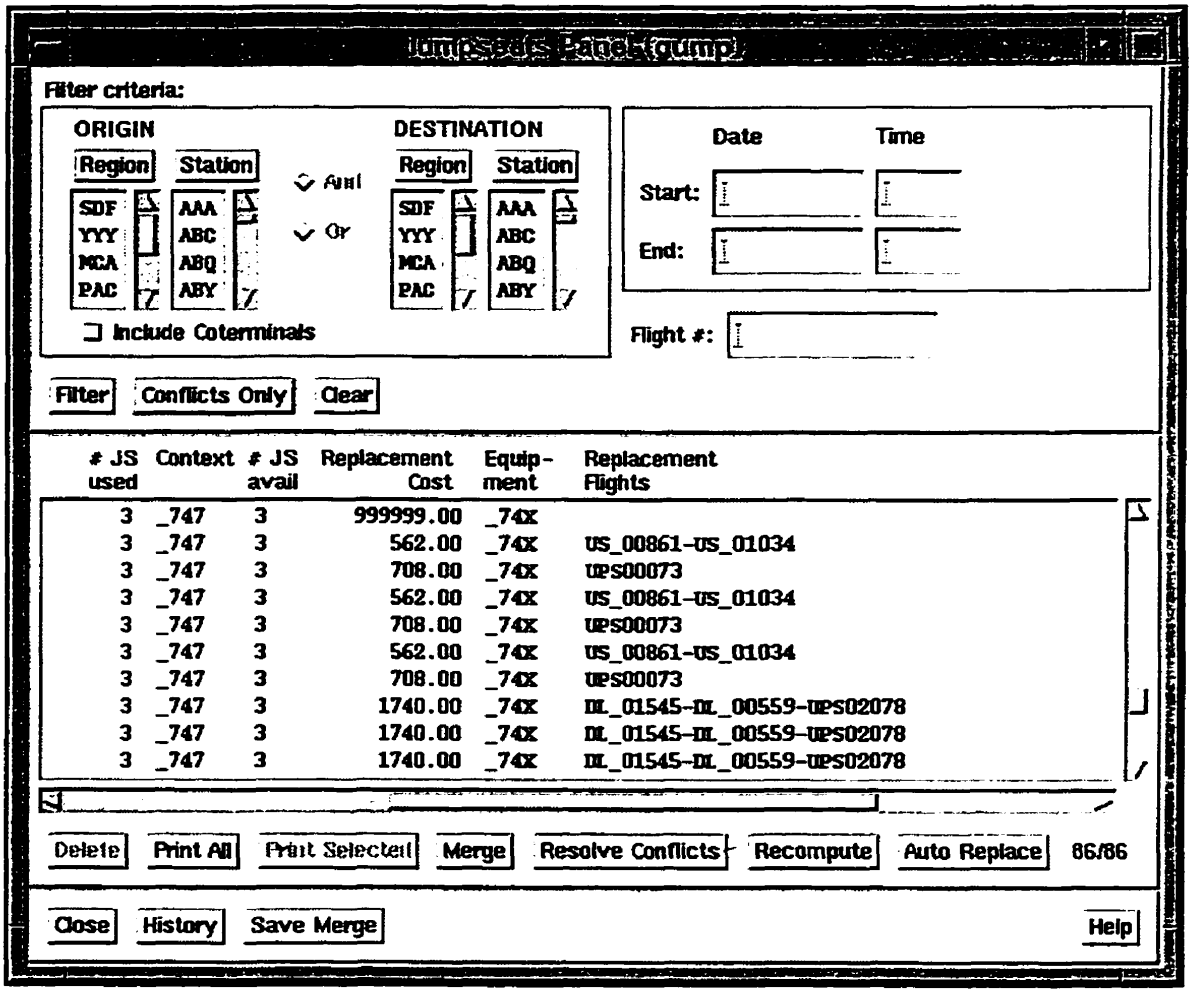


Figure 4.12 Panneau de mise en place (*Jumpseats Panel*)

Le panneau est principalement composé d'une liste à défilement au centre, d'une section de filtrage au haut et de boutons de contrôle au bas. Les opérations effectuées par ces boutons sont les suivantes: annulation (*Delete*), fusion (*Merge*), résolution de conflits (*Resolve Conflicts*), calcul des coûts de remplacement (*Recompute*) et remplacement automatique (*Auto Replace*) des mises en place.

4.2.2.1 Fusionnement des mises en place

Cette fonctionnalité permet à l'usager de consulter les mises en place utilisées dans d'autres solutions (scénarios) qui sont produites par d'autres usagers. Elle permet aussi de résoudre les conflits localement à l'aide du panneau, en activant les boutons correspondants. Les mises en place provenant des autres scénarios sont identifiées par le type *Merge* pour que l'usager puisse toujours les différencier. Lorsque le bouton *Merge* est sélectionné, une liste des scénarios externes est affichée à l'écran pour permettre à l'usager de sélectionner le scénario qu'il veut. La liste dans le panneau de résolution de conflits affiche les nouvelles mises en place après les avoir combinées à celles du scénario courant.

4.2.2.2 Interaction avec le panneau graphique

Nous avons déjà mentionné que toutes les mises en place internes utilisées par les rotations de la solution courante interagissent avec la liste dans le panneau de mise en place interne. Comme les objets peuvent être édités à deux endroits différents, le système a été conçu pour que toute opération effectuée dans l'un des panneaux soit automatiquement reflétée dans l'autre, afin d'éviter les problèmes de synchronisation.

L'annulation d'une mise en place dans le panneau de mise en place doit entraîner la disparition de la rotation correspondante dans la fenêtre graphique. Par ailleurs, dans la fenêtre graphique, la construction d'un nouvelle rotation incluant l'usage d'une nouvelle mise en place doit entraîner l'ajout automatique de cette dernière à la liste du panneau.

Dans le cas de l'annulation d'une mise en place dans le panneau de mise en place, l'entrée annulée dans la liste correspond à une structure *PgJSOL*, qui à son tour offre l'accès au vol (*PgLEG*). A partir du *PgLEG*, on vérifie si le vol est utilisé dans une rotation (*PgBLOCK*). Si c'est le cas, une fonction du gestionnaire est appelée pour détruire cette rotation (figure 4.13).

```
fPgBlockDelTrip(PgPG       *ppg,
                PgBLOCK  *pblock
               );
```

Figure 4.13 Fonction de destruction d'un bloc

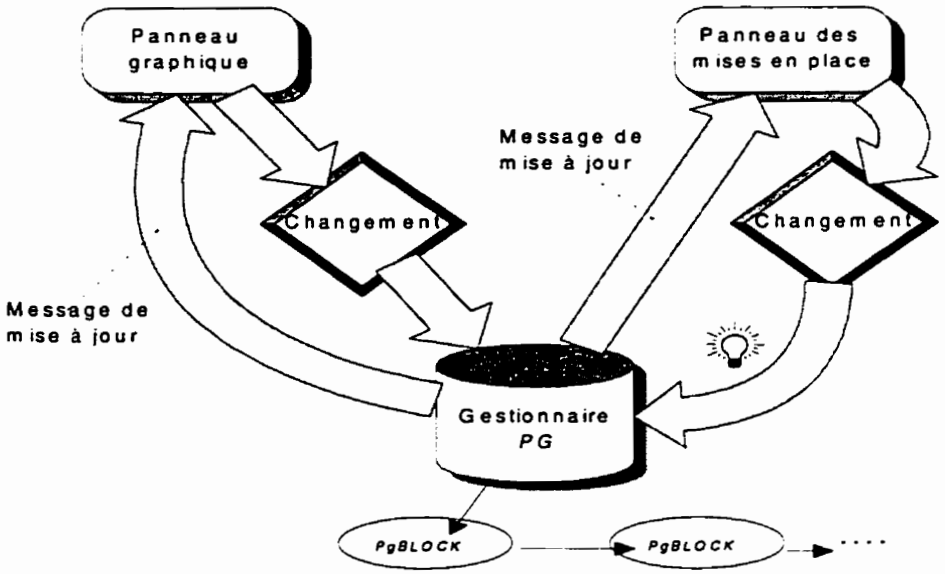


Figure 4.14 Méthode d'interaction avec le panneau graphique

Le gestionnaire utilise la fonction de mise à jour enregistrée par le module *GUI* au début de l'application (discuté au chapitre 3) pour transmettre l'information au panneau principal. La fenêtre graphique reçoit le message qui indique le numéro de la rotation à annuler et rafraîchit l'affichage.

Lorsqu'une rotation est construite dans la fenêtre graphique, c'est celle-ci qui modifie la structure *PgBLOCK*. Le gestionnaire de données transmet un message de mise à jour au panneau pour rafraîchir la liste de mise en place (figure 4.14).

4.2.2.3 Calcul des coûts de remplacement

Toute mise en place interne utilisée par une rotation peut être remplacée par un vol commercial. Cette étape devient parfois nécessaire pour résoudre le conflit. Quand deux rotations utilisent le même vol pour la mise en place de plus qu'un équipage, une des rotations doit remplacer ce vol interne par un vol commercial ou même par un autre vol interne disponible. Un coût de remplacement sera attribué pour que le système remplace le vol interne par un autre type de vol. Le calcul de ce coût est la différence entre la rotation incluant le remplacement et celle-ci sans ce dernier. C'est-à-dire, le surplus qu'il faut payer pour que le vol interne soit remplacer par un vol externe. Donc c'est assez logique que l'usager veuille remplacer les mises en place en conflit qui ont le coût de remplacement le plus bas (section suivante).

Les coûts de remplacement sont affichés dans la liste du panneau de résolution de conflits. Comme l'opération de calcul prend un temps d'exécution considérable, un bouton poussoir a été ajouté au panneau pour permettre à l'usager de la lancer au moment où il juge nécessaire. Pour cette même raison, cette opération n'est pas appelée lors de l'accès au panneau. Si aucun calcul n'a été sauvegardé auparavant, la valeur *999999* est initialement assignée au coût par défaut.

Pendant que les coûts sont calculés, le système sélectionne et suggère les vols de remplacement. Il est parfois nécessaire d'utiliser plus d'un vol commercial pour remplacer la mise en place interne qui est actuellement utilisée. La liste des vols de remplacement est aussi affichée dans la liste, à l'extrémité droite parce que le nombre de vols de remplacement est variable. Les mises en place pour lesquelles on ne trouve pas de remplacement gardent toujours un coût très élevé, qui est la valeur de défaut 999999.

Après avoir activé le bouton de résolution, la procédure de calcul est appelée au niveau du module *Pid*. La fonction *fPidJumpseatsRecomputeCost()* appelle à son tour la fonction *pdhm->qfRecomputeCostJseat(ppg)* au niveau du module *PG*. La procédure initialise et assigne les valeurs et les vols trouvés dans la structure *PgJSOL* dans les champs *PgCENT centCostRepl* et *CH rgszFlights* respectivement. A la fin de l'opération, le panneau rafraîchit le contenu de la liste (Figure 4.15).

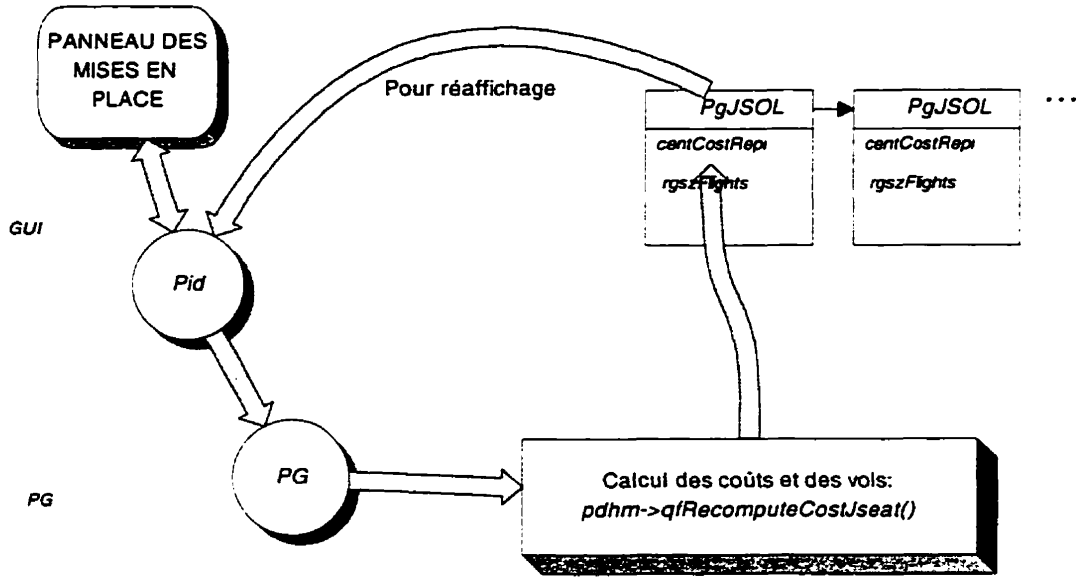


Figure 4.15 Méthode de calcul et de réaffichage

4.2.2.4 Résolution automatique des conflits

La procédure de résolution de conflits dépend énormément des coûts de remplacement et des vols suggérés par le système. Après avoir calculé les coûts de remplacement, l'usager peut résoudre les conflits de deux façons. La première méthode consiste à appuyer sur le bouton *Resolve Conflicts*. Une procédure implantée dans la fonction *ResConfCB()*, regroupe les mises en place utilisant le même vol et qui sont en conflit. Elle élimine ensuite les vols qui coutent le moins cher à remplacer. Une fonction

de mise à jour des conflits (*fPidJumpseatsUpdateConflicts()*) est ensuite appelée pour vérifier si le conflit est résolu pour ce groupe. Si c'est le cas, on recommence avec un nouveau groupe de conflits. Si ce n'est pas le cas, on continue l'élimination des mises en place avec les coûts les plus bas jusqu'à ce que le conflit soit résolu. Le résultat de cette procédure est que les rotations correspondantes sont éliminées de la fenêtre graphique et c'est à l'usager d'aller compléter manuellement ces dernières avec les vols suggérés. Un panneau listant les rotations affectées apparaît ensuite pour informer l'usager des changements effectués dans la région graphique.

La deuxième méthode est le complément de la précédente. Le système poursuit avec le remplacement des vols automatiquement. En d'autre mots, il détruit les anciennes rotations, y insère les nouveaux vols de remplacement et les affichent automatiquement dans la fenêtre (Figure 4.16).

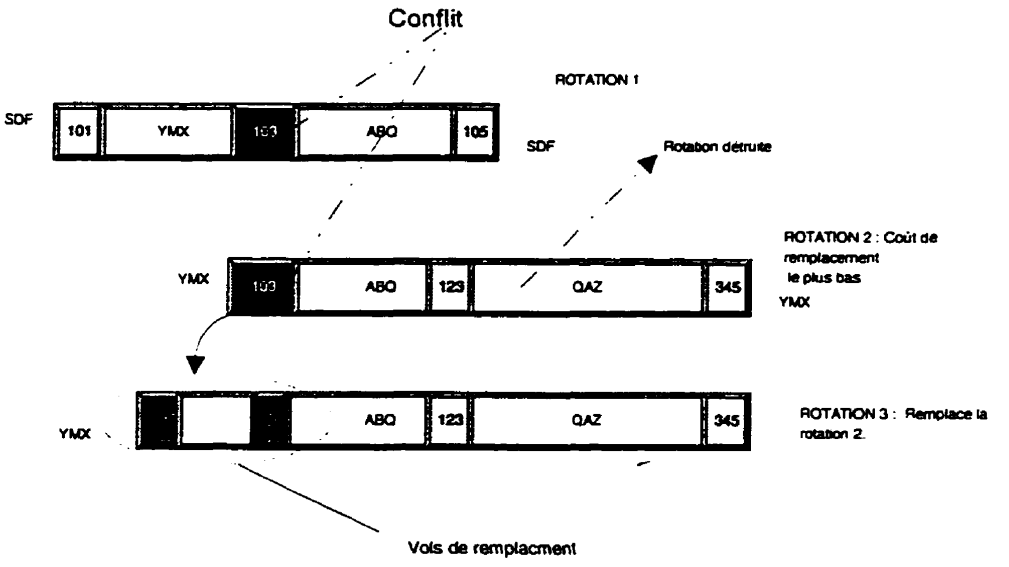


Figure 4.16 Remplacement automatique des rotations

CONCLUSION

Le nombre de rotations d'équipages qu'un planificateur aura à manipuler dans une certaine période de temps peut être gigantesque. L'interface graphique utilisateur traitée dans ce mémoire est un outil conçu pour aider le planificateur à construire des rotations d'équipages légales et optimales, et lui permettre de manipuler et d'ajuster ces rotations, ainsi que les données d'entrée. L'interface est facile d'emploi, s'adapte à tout genre d'usager et possède un système d'aide contextuelle permettant une utilisation plus efficace donc, une productivité accrue. Elle offre des mécanismes de création, de copiage, de destruction et de déplacement de scénarios, dans une application multi-fenêtrée.

La partie la plus importante de l'interface est sûrement le panneau principal qui affiche et permet de modifier toutes les rotations générées automatiquement par un système d'optimisation complexe, sur une période de temps prédéfinie. Le contenu de la région graphique est contrôlable par l'usager. Des tronçons de vol, des groupes (cellules) et des rotations peuvent y être affichés. Différents outils de filtrage et de sélection d'objets et de triage sont utilisés comme moyen de spécifier les critères qui définissent l'ensemble des objets affichés.

Notre démarche consistait à donner une description détaillée de l'organisation des données maîtres pour qu'elles soient accessibles par l'interface, à tous les usagers. Puis, nous avons analysé l'environnement global des données usager. Nous avons mis au point une méthode unique de mémorisation des démarches effectuées par l'usager. Nous avons aussi développé des outils de récupération de données et la possibilité de comparer plusieurs étapes de l'optimisation de la solution et une méthode de mise à jour avec les données maîtres.

L'interface graphique discutée dans ce mémoire est unique dans son traitement de la mémorisation des étapes. Un fonctionnalité très puissante permet, grâce à un processus complexe, de suivre l'évolution d'un scénario, dès sa création. Toutes les étapes d'un scénario sont accumulées grâce à un mécanisme de journal de changements (*Audit Trail*)

qui offre au planificateur une vue globale de toutes les modifications produites depuis le début du scénario. Chaque modification est représentée en détails, suivies de la liste des fichiers affectés et d'un bloc de statistiques permettant de visualiser l'impact de chaque modification sur l'état de la solution.

Un mécanisme de prise d'instantanés (*Snapshots*) aussi complexe que le processus de mémorisation des étapes agit en parallèle. Il produit un arbre historique détaillant chaque étape du scénario. Ce système est beaucoup plus sophistiqué que les mécanismes de *Undo* qu'on retrouve sur le marché dans les applications commerciales multi-fenêtrées, puisqu'il permet à l'usager de se repositionner n'importe où dans l'arbre historique.

Les outils (panneaux) mis à la disposition de l'usager sont faciles d'utilisation, mais les opérations qui se cachent derrière chaque panneau sont très complexes et ont nécessités beaucoup de recherche et d'analyse. En effet, beaucoup d'efforts ont été mis sur l'organisation de la gestion de l'environnement afin de produire un logiciel flexible et facilement configurable.

Un autre aspect important de l'application est qu'elle a été conçue de façon à être facilement transportable d'un système à l'autre. En effet, le logiciel a la capacité de mémoriser une multiplicité de configurations qui correspondent aux différents environnements ce qui fait qu'il s'adapte automatiquement à chaque nouvelle installation. De plus, la même application peut être adaptée à différents types de clientèle, chez les compagnies aériennes ou autres. Le système est développé de façon à ce que l'interface graphique, par l'entremise des fichiers de ressources, puisse être configurée selon les différents besoins des clients.

Le panneau principal de l'interface offre l'affichage des rotations sur plusieurs niveaux configurables, ainsi que des possibilités de manipulations graphiques et de constructions de nouvelles rotations. Plusieurs opérations manuelles ont été étudiées ainsi que la hiérarchie des modules et la communication entre ces derniers. Nous avons clarifié

le rôle du module gestionnaire de données et celui de l'interface graphique, tout en clarifiant son fonctionnement.

Le nombre élevé d'objets graphiques qui peuvent être affichés dans la fenêtre rend le mécanisme d'affichage et de manipulation graphique assez complexe et délicat. La rapidité et l'efficacité des techniques developpées permettent de diminuer le temps d'exécution des opérations graphiques. Les opérations affectant les données sont effectuées séparément dans un module gestionnaire de données. De plus, un mécanisme de mise à jour très efficace et très sophistiqué qui permet de rafraîchir rapidement la représentation des objets graphiques réside entre le module gestionnaire de données et le module d'affichage. Ceci permet d'offrir une grande robustesse à l'usager.

Le traitement des mises en place est une autre fonctionnalité unique à l'interface conçue pour ce mémoire. Elle consiste à résoudre des conflits d'utilisation de vols par les rotations. Le travail se fait à l'aide d'un panneau spécial qui exécute plusieurs algorithmes de recherche et de calcul, tout en offrant à l'usager la possibilité de visualiser les coûts de mise en place, et d'interagir là où il le juge nécessaire.

L'interface proposée ici a été facilement acceptée par les planificateurs des compagnies aériennes où le système est implanté. La flexibilité et la convivialité de l'interface permet à l'usager de manipuler ses données facilement. Une telle interface offre aux planificateurs un accès beaucoup plus facile aux méthodes informatiques de gestion de données et à l'utilisateur des outils d'optimisation. Rappelons que les planificateurs sont des experts en transport aérien mais pas en informatique. Des experts de plusieurs compagnies aériennes ont déclaré, après avoir expérimenté l'interface graphique, qu'ils considéraient maintenant possible l'automatisation du processus de planification.

La suite logique de la recherche dans ce domaine consisterait sûrement à adapter le module d'interface pour qu'il affiche et permette la manipulation de d'autres éléments, dans des applications autres que l'aérien.

BIBLIOGRAPHIE

[1] ABBARA, J. (1989), "Applying Integer Linear Programming to the Fleet Assignment Problem", Interfaces, No. 19.

[2] AGGARD, J., ARABEYRE, J.P., et VAUTIER, J. (1967), "Génération automatique de rotations d'équipages", RAIRO, No. 6

[3] BARUTT, J., HULL, T. (1990), "Airline Crew Scheduling: Supercomputers and Algorithms", SIAM News, No. 23

[4] BERDYCH, J., "Adopting Motif: Issues Encountered and Lessons Learned", Ericsson Communications Inc.

[5] BOYLE, M. (1994), "Altitude/Pairing GUI Design Specifications".

[6] BOYLE, M., TELISMA, B. et Zerbé, S. (1995) "Altitude Programmer's Guide".

[7] DESROSIERS, J., DUMAS Y., SOLOMON M.M. et SOUMIS F. (1992), "Time Constrained Routing and Scheduling", Cahiers du GERAD G-92-42, École des Hautes Études Commerciales, Montréal, Canada H3T 1V6.

[8] DESROSIERS, J., DUMAS, Y., DESROCHERS, M., SOUMIS, F., SANSO, B.et TRUDEAU, P.(1991), "A Breakthrough in Airline Crew Scheduling", Cahiers du GERAD G-91-11, École des Hautes Études Commerciales, Montréal, Canada H3T 1V6.

[9] DUGAL, Y. (1983) "Interface usager - ordinateur pour tracer les efforts internes d'un ensemble de poutres", École Polytechnique de Montréal.

[10] HOPGOOD, F.R.A. (1986) "Methodology of Window Management" Springer-Verlag.

[11] LAVOIE, S., DESROSIERS, J., DUMAS, Y., SOLOMON, M. et SOUMIS, F. (1994), "An Optimizer for Real World Aircraft and Crew Scheduling", Cahiers du GERAD G-94-23, École des Hautes Études Commerciales, Montréal, Canada H3T 1V6.

[12] LAVOIE, S. et RIOUX, B. (1993), "Lexique Crew", Cahiers du GERAD G-93-17, École des Hautes Études Commerciales, Montréal, Canada H3T 1V6.

[13] RONDEAU, L. (1996), "Altitude Pairing and Bidline User's Guide".

[14] SCHENEIDERMAN, B. (1987), "Desiging The User Interface". Addison Wesley.

[15] VOISIN, G., (EDF), "Les techniques du dialogue: Les menus, direction des études et recherches", Électricité de France.

[16] YOUNG, D.A. (1991), "Programmation avec Xt Intrinsics", Masson: Paris.

ANNEXE A: Panneaux de l'interface

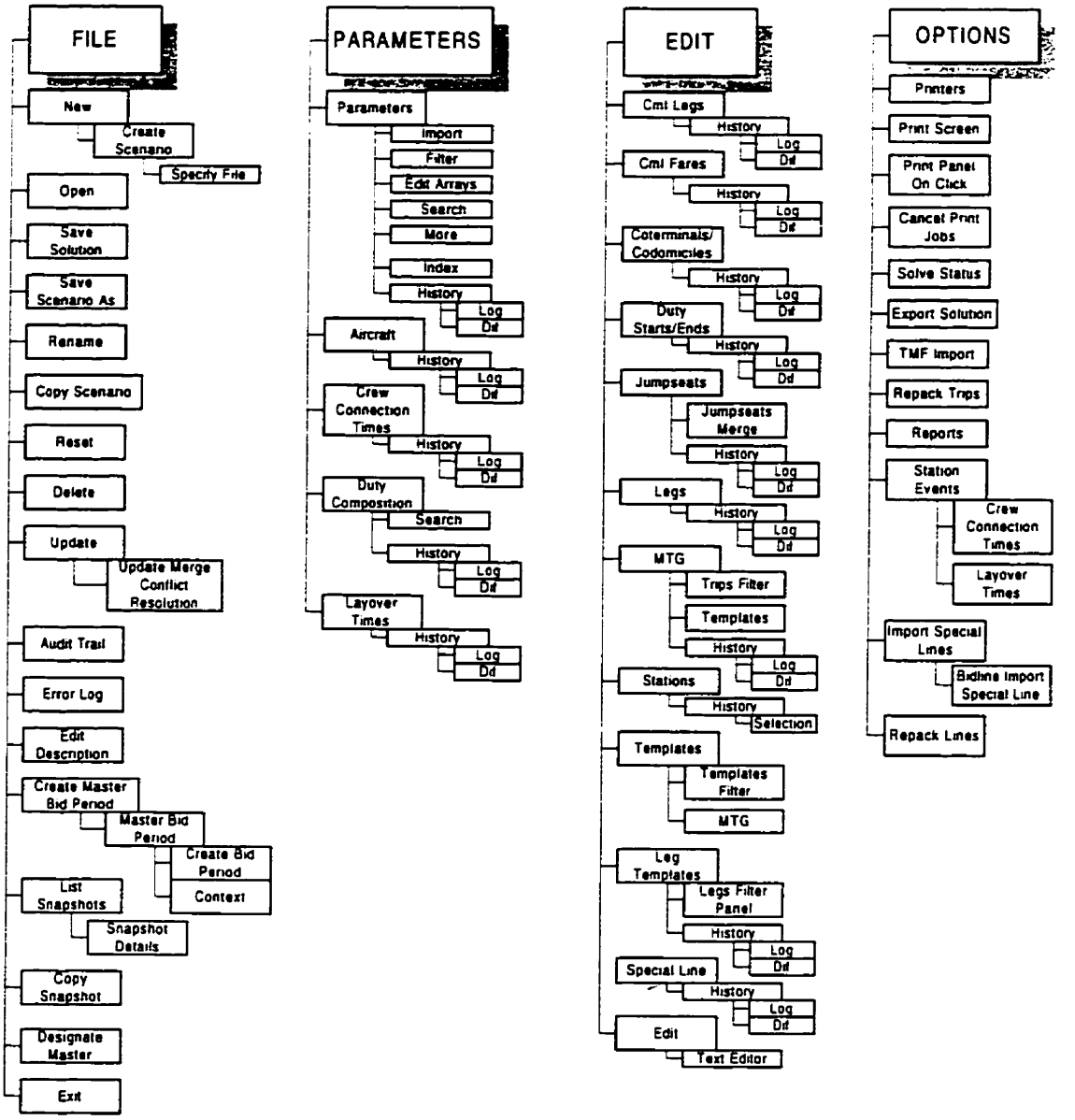
Dans cet annexe on représente le diagramme de toutes les fonctionnalités accessibles par le panneau principal. On inclut aussi quelques panneaux qu'on a mentionné dans le mémoire, avec à la fin un exemplaire de détails d'une rotation.

Le diagramme explique comment accéder à tous les panneaux et tous les outils de l'interface graphique en se servant de la barre de menus. On peut citer les menus suivants: *File*, *Parameters*, *Edit*, *Options*, *Display*, *Tools* , *Windows* et *Help*.

Le panneau de couleurs permet à l'usager d'assigner des différentes couleurs à tous les objets graphiques, pour les différencier à l'écran.

Le panneau de détails d'un objet graphique (*Task details panel*) contient tous les détails nécessaires d'un bloc, d'une rotation ou d'un autre objet affiché graphiquement à l'écran. On joint aussi un exemple de rotation détaillée à la fin de l'annexe.

MAIN


FILE
New
Create Scenario
Specify File
Open
Save Solution
Save Scenario As
Rename
Copy Scenario
Reset
Delete
Update
Update Merge Conflict Resolution
Audit Trail
Error Log
Edit Description
Create Master Bid Period
Master Bid Period
Create Bid Period
Context
List Snapshots
Snapshot Details
Copy Snapshot
Designate Master
Exit
PARAMETERS
Parameters
Import
Filter
Edit Arrays
Search
More
Index
History
Log
Dif
Aircraft
History
Log
Dif
Crew Connection Times
History
Log
Dif
Duty Composition
Search
History
Log
Dif
Layover Times
History
Log
Dif
EDIT
Cmi Legs
History
Log
Dif
Cmi Fares
History
Log
Dif
Coterminals/ Codomiciles
History
Log
Dif
Duty Starts/Ends
History
Log
Dif
Jumpseats
Jumpseats Merge
History
Log
Dif
Legs
History
Log
Dif
MTG
Trips Filter
Templates
History
Log
Dif
Stations
History
Selection
Templates
Templates Filter
MTG
Leg Templates
Legs Filter Panel
History
Log
Dif
Special Line
History
Log
Dif
Edit
Text Editor
OPTIONS
Printers
Print Screen
Print Panel On Click
Cancel Print Jobs
Solve Status
Export Solution
TMF Import
Repack Trips
Reports
Station Events
Crew Connection Times
Layover Times
Import Special Lines
Bidline Import Special Line
Repack Lines

PANEL

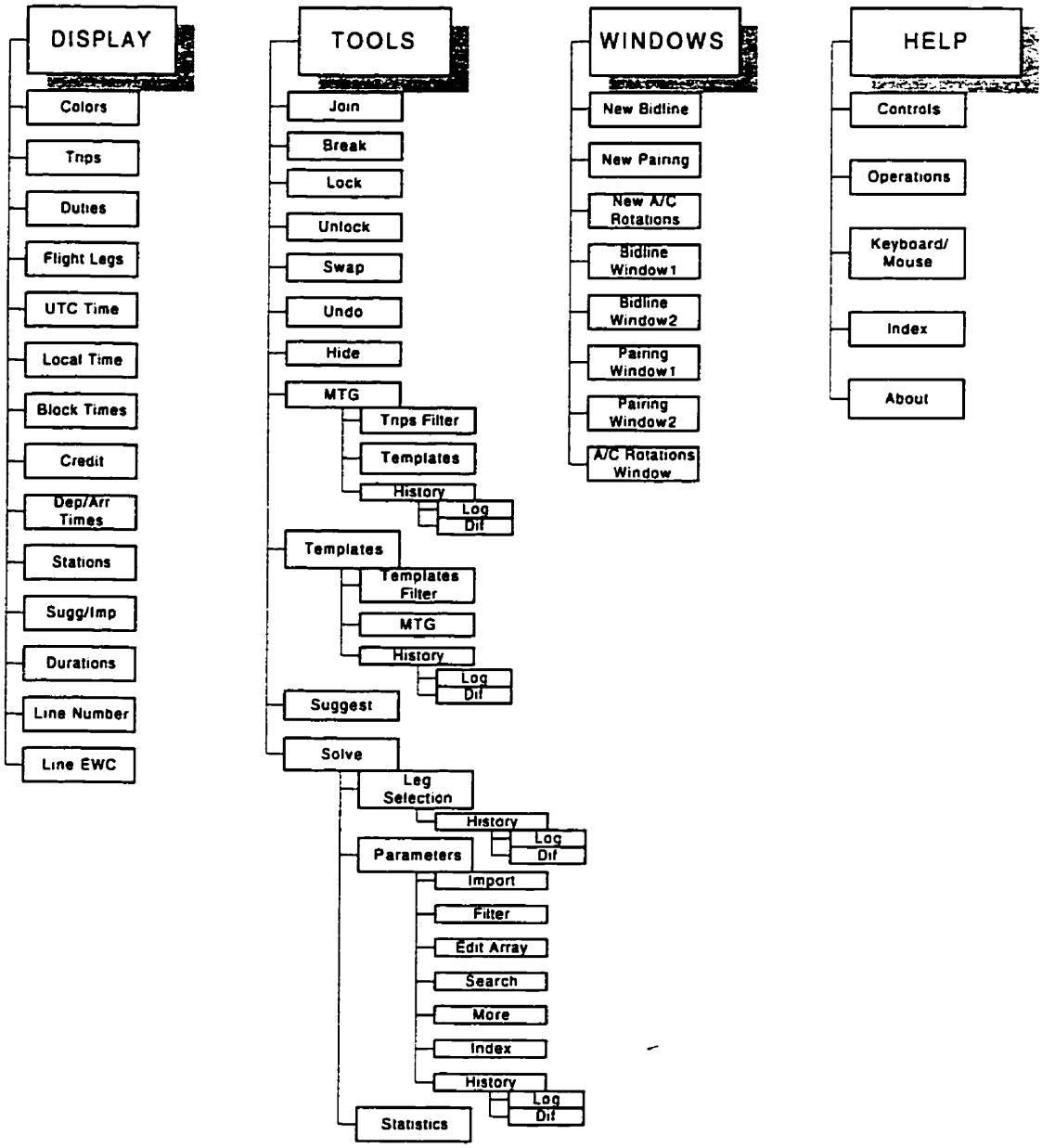
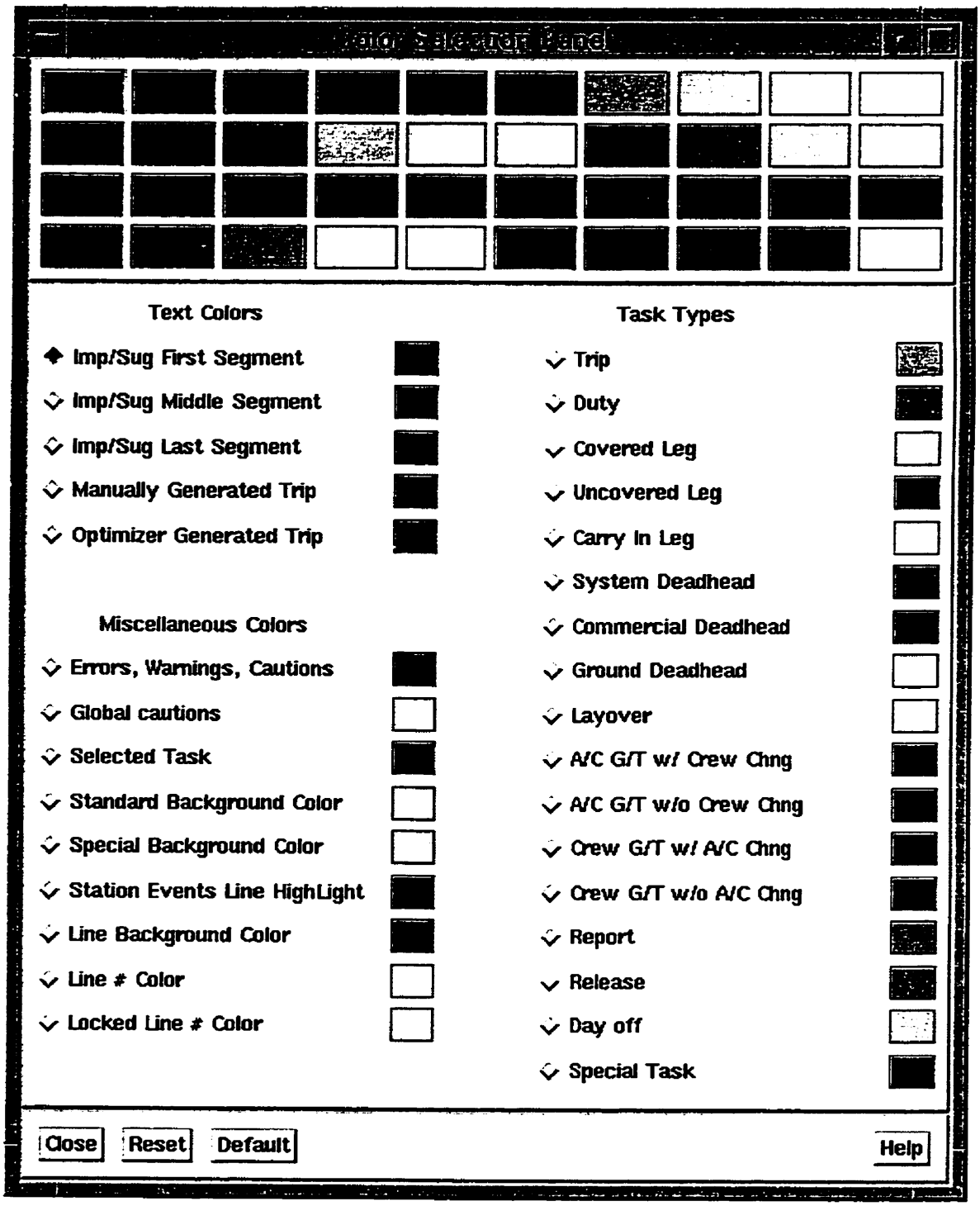
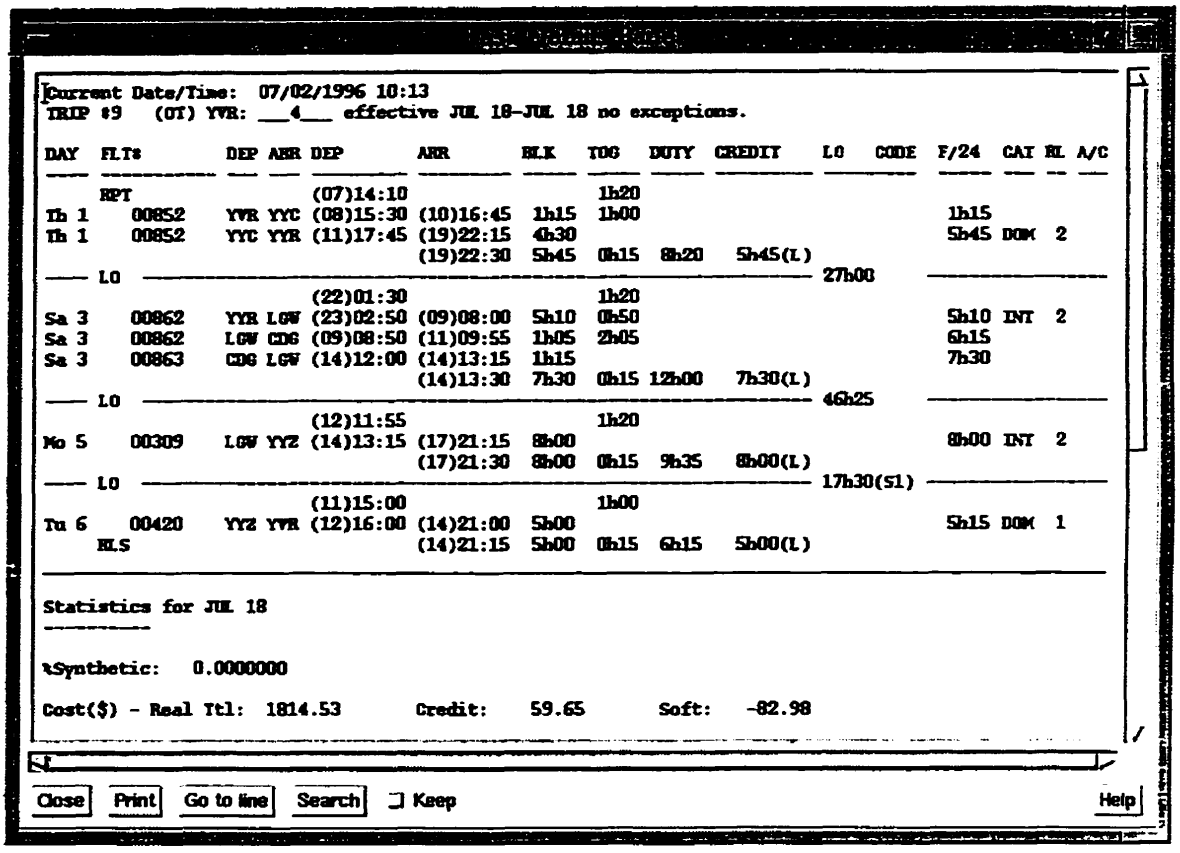


Diagramme de fonctionnalités



Panneau des couleurs



Current Date/Time: 07/02/1996 10:13
TRIP #9 (OT) YVR: ___4___ effective JUL 18-JUL 18 no exceptions.

DAY	FLT#	DEP	ARR	DEP	ARR	BLK	TOG	DUTY	CREDIT	LO	CODE	F/24	CAT	RL	A/C
	RPT			(07)14:10			1h20								
Th 1	00852	YVR	YYC	(08)15:30	(10)16:45	1h15	1h00					1h15			
Th 1	00852	YYC	YVR	(11)17:45	(19)22:15	4h30						5h45	DOM	2	
					(19)22:30	5h45	0h15	8h20	5h45(L)						
	LO									27h00					
				(22)01:30			1h20								
Sa 3	00862	YVR	LGW	(23)02:50	(09)08:00	5h10	0h50					5h10	INT	2	
Sa 3	00862	LGW	CDG	(09)08:50	(11)09:55	1h05	2h05					6h15			
Sa 3	00863	CDG	LGW	(14)12:00	(14)13:15	1h15						7h30			
					(14)13:30	7h30	0h15	12h00	7h30(L)						
	LO									46h25					
				(12)11:55			1h20								
Mo 5	00309	LGW	YYZ	(14)13:15	(17)21:15	8h00						8h00	INT	2	
					(17)21:30	8h00	0h15	9h35	8h00(L)						
	LO									17h30(S1)					
				(11)15:00			1h00								
Tu 6	00420	YYZ	YVR	(12)16:00	(14)21:00	5h00						5h15	DOM	1	
	RLS				(14)21:15	5h00	0h15	6h15	5h00(L)						

Statistics for JUL 18

%Synthetic: 0.0000000

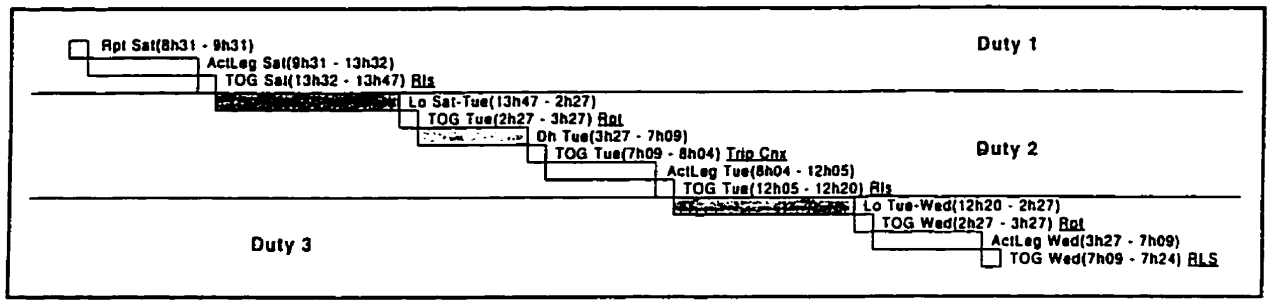
Cost($) - Real Ttl: 1814.53 Credit: 59.65 Soft: -82.98

Close | Print | Go to line | Search | Keep | Help

Panneau de détails d'un objet graphique.

Itinéraire trip #1

Bid Period: BID9302
Fleet: 757
Solution no: 0
Domicile Gateway (Dom_Gtw): SDF

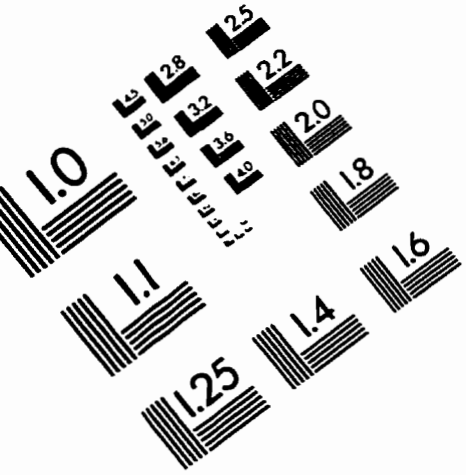


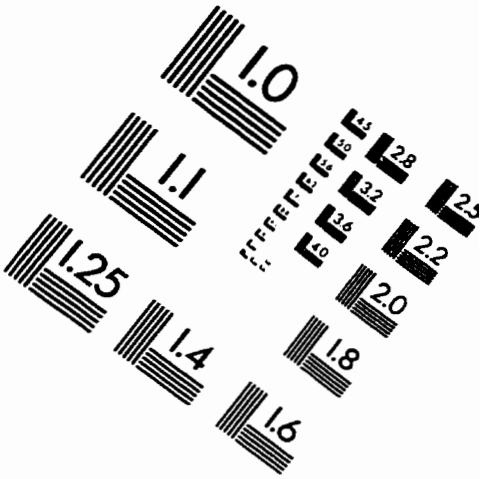
TRIP #1 (OT) SDF: _____6_ effective FEB 27-FEB 27 no exceptions.

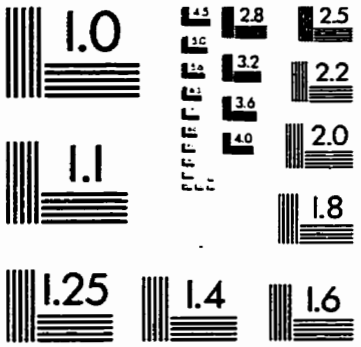
	DAY	FLT#	DEP	ARR	DEP	ARR	BLK	TOG	DUTY	CREDIT	LO	CODE	F/24	CAT
RPT					(03) 08:31			1h00						
	1	00904	SDF	LAX	(04) 09:31	(05) 13:32	4h01						4h01	NO
						(05) 13:47	4h01	0h15	5h16	4h01(L)				NO
LO											60h40	(S1)		
					(18) 02:27			1h00						
	4	UPS00903	LAX	SDF	(19) 03:27	(02) 07:09	3h42	0h55						NO
	4	00904	SDF	LAX	(03) 08:04	(04) 12:05	4h01						4h01	NO
						(04) 12:20	4h01	0h15	9h53	7h43(L)				YES
LO											14h07	(S1)		
					(18) 02:27			1h00						
	5	00903	LAX	SDF	(19) 03:27	(02) 07:09	3h42						7h43	NO
RLS						(02) 07:24	3h42	0h15	4h57	4h00(M)				NO

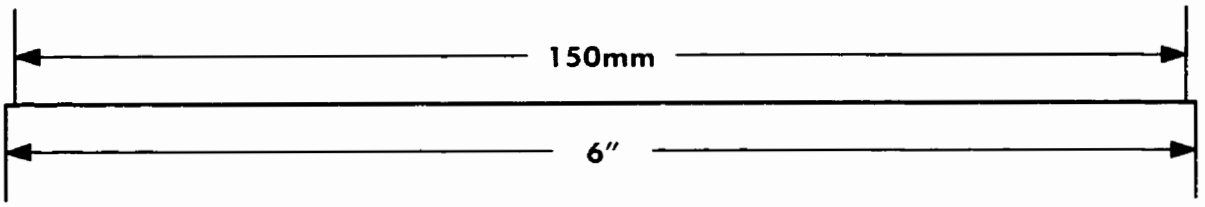
Rotation détaillée

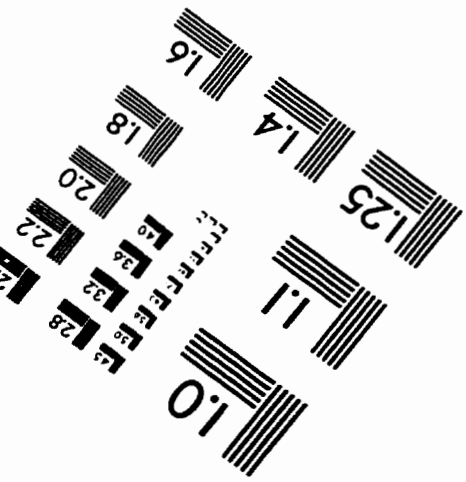
IMAGE EVALUATION
TEST TARGET (QA-3)

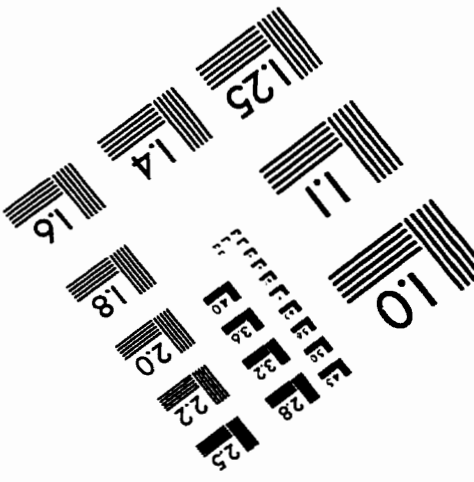












APPLIED IMAGE, Inc
1653 East Main Street
Rochester, NY 14609 USA
Phone: 716/482-0300
Fax: 716/288-5989