gW : un environnement hypertexte programmable comme support de cours pour l'optimisation linéaire

Gérald COLLAUD et Jacques PASQUIER

Abstract. For several years, the Institute of Informatics of the University of Fribourg (IIUF) has been developing applications in the hypertext and linear optimization fields. We developed several prototypes, including gLPS for WEBSs (gW), which combines the hypertext capabilities of WEBSs with the graphical modeling aspects of gLPS. This article describes this prototype and, through an imaginary user session, shows how users can interact with the environment and how hypertext and scripting facilities help advanced OR students understand linear optimization.

Keywords: Linear programming, graphical modelisation, hypertext, electronic book.

Mots-clés : Programmation linéaire, modélisation graphique, hypertexte, livre électronique.

1. Contexte

L'institut d'informatique de l'Université de Fribourg poursuit depuis de nombreuses années des recherches dans les domaines de l'optimisation linéaire et des hypertextes. En optimisation linéaire, (Hürlimann, 1993) a développé un langage algébrique de modélisation et de résolution (LPL), alors que (Collaud, 1993) a développé un environnement de modélisation (gLPS) proposant un langage graphique (gLPL) basé sur LPL.

De leur côté, les recherches sur les hypertextes ont conduit à l'aboutissement de deux thèses de doctorat (Savoy, 1987) et (Monnard, 1993).

✉ Gérald Collaud; Faculty of Business Administration; Simon Fraser University; Burnaby, V5A 1S6 (Canada).

✉ Jacques Pasquier; Institut d'informatique; Université de Fribourg – Regina Mundi; 2, rue Faucigny; CH–1700 Fribourg (Suisse).
Fax : +41 37 29 97 26 E-mail : Jacques-Pasquier@UniFr.CH

Ce dernier a étendu un prototype d'hypertexte classique (Pasquier *et al.*, 1988) par l'adjonction de capacités de scriptage (Monnard et Pasquier, 1992) (Pasquier et Monnard, 1995).

De manière générale, ces deux domaines font l'objet de recherches intensives. En programmation linéaire (PL), les environnements existants, tels que GAMS (Brooke *et al.*, 1988), AMPL (Fourer *et al.*, 1987), MIMI (Chesapeake Decision Sciences, 1993) et LPL permettent aux utilisateurs de représenter leurs modèles dans un format déclaratif et exécutable. Les progrès réalisés dans la définition des interfaces utilisateurs ainsi que dans la qualité des écrans ont permis l'émergence de nouveaux systèmes de modélisation graphique tels que les graphes Et-Ou (Raghunathan, 1987), la modélisation structurée (Geoffrion, 1989) (Hamacher *et al.*, 1994), GBMS (Jones, 1990, 1992), NETFORMS (Glover *et al.*, 1990), LPForm (Ma *et al.*, 1989), GIN (Steiger *et al.*, 1991), les diagrammes entité-relation (Choobineh, 1991), ou encore gLPS (Collaud et Pasquier, 1994). (Steiger et Sharda, 1991) ainsi que (Greenberg et Murphy, 1991) proposent une excellente vue d'ensemble des langages algébriques, alors que (Jones, 1994) présente un résumé des langages graphiques.

Dans sa forme la plus simple, un hypertexte améliore la compréhension de textes ou de graphiques en permettant aux utilisateurs de passer d'un point à un autre simplement en cliquant sur un élément particulier (Conklin, 1987). Toutefois, il est souvent désirable d'avoir des interactions plus complexes. Par exemple, de petits programmes, aussi appelés scripts, permettent d'obtenir des comportement plus intéressants. Actuellement, avec l'extraordinaire essor du multimédia et du World Wide Web (WWW), le domaine des hypertextes est l'objet d'une intense recherche, dont il est encore difficile de prévoir toutes les implications.

1.1. Des livres électroniques : pour quelle raison?

Traditionnellement, les étudiants impliqués dans un cours donné *ex cathedra* par un professeur se voient proposer, afin de répéter et d'approfondir la matière enseignée, un support de cours prenant la forme d'un livre classique imprimé sur papier. Un tel livre présente de nombreux avantages. Il est facilement transportable, ne nécessite aucun équipement particulier pour être consulté et, surtout, il propose un modèle d'interaction connu et accepté de tous. De plus, bien que sa structure physique linéaire encourage plutôt une lecture séquentielle parfois mal appropriée à l'apprentissage de

certaines matières, il est en général possible d'accéder à son contenu d'une manière plus directe en suivant des références croisées ou en s'appuyant sur sa table des matières ou son index.

Cependant, malgré son acceptation universelle, lorsque le livre classique doit servir de support à l'enseignement d'une matière scientifique telle que les mathématiques appliquées par exemple, il souffre de nombreuses faiblesses du fait de sa structure figée et de son comportement statique. Les exercices corrigés proposés par un livre classique, par exemple, sont définis une fois pour toutes. Il n'est donc pas possible pour le lecteur de jouer avec les paramètres d'un modèle afin d'en explorer les propriétés. Une réponse partielle à ce problème consiste à joindre au livre une disquette contenant un (ou plusieurs) programme(s) permettant d'effectuer des calculs sur les modèles proposés par l'auteur. Cependant, cette stratégie, comme toute solution hybride de ce genre, implique un effort supplémentaire de la part de l'étudiant (apprentissage d'une application informatique encore inconnue, inconsistances possibles entre la terminologie du livre et celle adoptée par le logiciel, etc.).

Face à ce manque de « réactivité » du livre classique, il semble donc que la seule solution véritablement satisfaisante consiste à abandonner le médium papier pour un médium plus riche et surtout plus dynamique, à savoir l'ordinateur.

C'est dans ce contexte que s'inscrit l'émergence du concept de livre électronique (LE). Comme le livre classique, ce dernier représente un ensemble cohérent de connaissances sur un sujet donné. La différence est que son contenu (textes, images, mais aussi modèles mathématiques et programmes divers) est regroupé dans une base de données informatique et qu'il ne peut donc être consulté qu'à l'aide d'un logiciel spécialisé.

Afin de créer et gérer un support de cours portant sur l'optimisation linéaire, nous avons donc tiré partie des recherches existantes à l'IIUF et avons créé un tel LE avec l'aide de notre dernier prototype, gLPS for WEBSs (gW), un environnement logiciel extensible combinant des capacités hypertextes avancées avec la possibilité de définir et de calculer des modèles d'optimisation linéaire (Collaud et Pasquier, 1995).

1.2. Structure de l'article

Le présent article se base sur une expérience concrète faite à l'IIUF dans le cadre d'un cours de deuxième cycle en recherche opérationnelle

afin de démontrer la faisabilité d'un support de cours sous forme d'un livre électronique. Afin d'atteindre au mieux cet objectif, l'article a été décomposé en trois parties. Dans un premier temps, la section 2 décrit gW et ses composants, sans lesquels rien n'aurait été possible. Ensuite, la section 3 offre un aperçu du LE « Optimisation linéaire » sous la forme d'une session imaginaire composée de six étapes. Enfin, la dernière section résume l'expérience et propose quelques enseignements pour le futur.

Il faut noter que vouloir illustrer les avantages d'un livre électronique à l'aide d'explications classiques et statiques sur papier représente un défi auquel il n'est que partiellement possible de répondre (une constatation contraire serait peu encourageante quant à la pertinence de nos recherches sur les livres électroniques). C'est pourquoi nous encourageons le lecteur disposant d'un *Macintosh* à essayer le programme ainsi que le LE « Optimisation linéaire » en les téléchargeant anonymement depuis `ftp-iiuf.unifr.ch`. L'application, accompagnée d'un guide de l'utilisateur ainsi que d'un tutoriel, se trouvent dans le dossier «`pub/soft_eng/gW`», et le manuel dans le dossier «`pub/soft_eng/gW/books`» sous les noms «`optimisation.book`» et «`optimisation`».

2. gLPS for WEBSs

gW combine la modélisation textuelle et graphique de problèmes d'optimisation linéaire ainsi qu'un système de résolution des modèles avec des

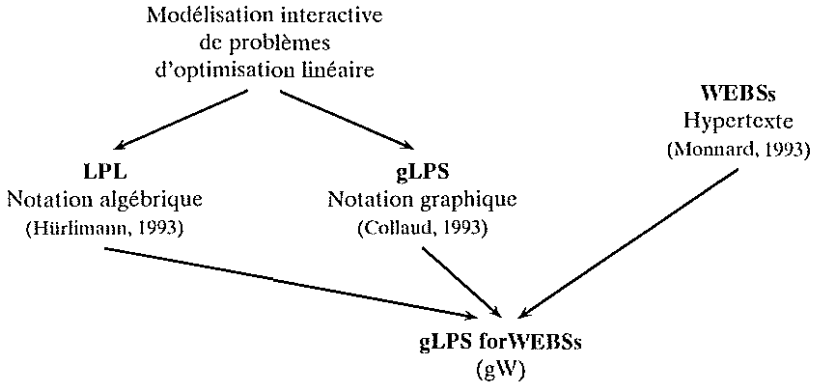

Fig. 1.– Historique de gLPS for WEBSs; gW est le résultat de la mise en commun de WEBSs, gLPS et LPL.

capacités hypertextes (*cf.* Fig. 1). Les capacités de modélisation graphique sont offertes par gLPS, les capacités de modélisation textuelle par LPL et les capacités de scriptage et hypertextes par WEBSs. Les sections suivantes présentent ces différents composants.

2.1. gLPS et LPL

L'optimisation linéaire consiste à essayer de maximiser ou minimiser une fonction d'objectif (profit, coût, ...) en respectant certaines contraintes. L'ensemble — contraintes et objectif — est généralement représenté par une série d'équations linéaires.

Dans les années soixante-dix, ces équations étaient représentées à l'aide de langages informatiques rudimentaires (p. ex. MPS). Les années quatre-vingts ont vu apparaître des langages algébriques de modélisation plus performants (LPL, AMPL, etc.). Ils permettaient de représenter le modèle dans un langage déclaratif proche des équations mathématiques. Enfin, les années quatre-vingt-dix, grâce en particulier à des matériels informatiques toujours plus performants tel que les interfaces à manipulation directe, ont vu l'arrivée de langages graphiques (gLPL, GIN, etc.). La suite de cette section présente brièvement les deux langages développés à l'IIUF à l'aide de la modélisation de l'exemple 1.

> Une entreprise fabrique deux types de micro-ordinateurs : un appareil dit « de table » et un autre dit « portable ». La production des deux sortes d'appareils a lieu en deux étapes qui sont la fabrication des composantes et leur montage. Les machines à disposition pour la fabrication des composantes sont utilisables pendant 200 heures par semaine et il faut une heure pour fabriquer les éléments nécessaires à un appareil de table et deux heures pour un appareil portable. La chaîne de montage est quant à elle disponible pendant 150 heures par semaine et il faut deux heures pour monter un appareil de table et quatre heures pour un appareil portable. On obtient un profit de 80.– par appareil de table et de 100.– par appareil portable. Le département de marketing de l'entreprise exige toutefois une production hebdomadaire minimale de 10 PC de table et 25 PC portables. Déterminez le plan de production qui maximise le profit.
>
> Exemple 1 : « Usine » : production de micro-ordinateurs.

Ce problème nécessite l'utilisation de deux variables, à savoir `X1` représentant le nombre de micro-ordinateurs de table à produire et `X2` le nombre de portables Il est ensuite trivial d'en déduire le modèle ci-dessous, formé des quatre contraintes `fabrication`, `montage`, `marketing1` et `marketing2` et de la fonction d'objectif `Z` :

```
fabrication   :  X1 + 2*X2 ≤ 200
montage       :  2*X1 + 4*X2 ≤ 150
marketing1    :  X1 ≥ 10
marketing2    :  X2 ≥ 25
Z             :  80*X1 + 100*X2 -> à maximiser
```

LPL (*Linear Programming Language*) est un langage algébrique qui tend à représenter un problème dans une forme proche de la notation mathématique ci-dessus. En particulier, LPL divise la représentation en quatre sections. Les trois premières définissent les composants du problème : `SET` pour les indices, `COEF` pour les données associées aux côtés droits et aux coefficients et `VAR` pour la spécification des variables de décision. La dernière section, `MODEL`, contient les équations de contrainte et de fonction d'objectif. La fenêtre droite de la figure 2a présente la version LPL des équations du modèle de l'exemple 1. Notez l'absence des sections `COEF` et `SET` puisqu'aucun indice ni coefficients n'ont été définis.

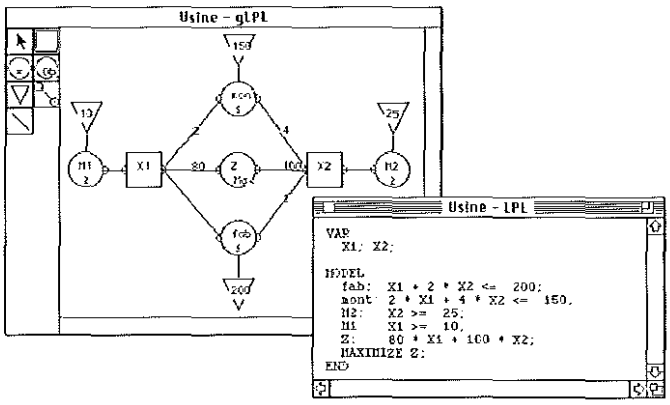

Fig. 2a.– Le modèle « Usine » ; la fenêtre gauche contient le modèle gLPL et la fenêtre droite, le modèle LPL. Avec gLPL, les carrés `X1` et `X2` représentent les micro-ordinateurs à produire (variables), les cercles `M1`, `M2`, ainsi que `mont` et `fab` avec leurs triangles respectifs (côtés droits) correspondent aux contraintes et le cercle `Z` représente la fonction d'objectif.

gLPS permet de modéliser interactivement des problèmes d'optimisation linéaire à l'aide d'un langage graphique (gLPL) basé sur les graphes « activités-contraintes » (Schrage, 1986). Chaque modèle d'optimisation linéaire est constitué au moins d'une variable, d'une contrainte et d'une fonction d'objectif. Le point essentiel à retenir est que gLPL représente ces éléments par un symbole graphique : le carré pour les variables, le cercle pour les contraintes et la fonction d'objectif, et le triangle pour le côté droit. La fenêtre gauche de la figure 2a présente la version gLPL des équations de l'exemple 1.

Ces représentations ne sont toutefois pas praticables lorsque les modèles contiennent un nombre plus élevé de variables et/ou de contraintes. Ces deux langages autorisent par conséquent l'indexation des éléments constitutifs d'un modèle. Ainsi, avec deux indices (`i=1,2` et `j=fab,mont`), l'exemple 1 peut être représenté de manière plus concise (*cf.* Fig. 2b). Il faut noter qu'avec gLPL, un élément auquel des indices ont été associés peut être visualisé sous une forme agrégée (`Xi,j`), désagrégée (`X1,1, X1,2, ...`) ou encore partiellement désagrégée (`X1,j, X2,j, ...`). Le lecteur peut se référer à (Collaud, 1993) pour plus de détails sur le langage et l'environnement.

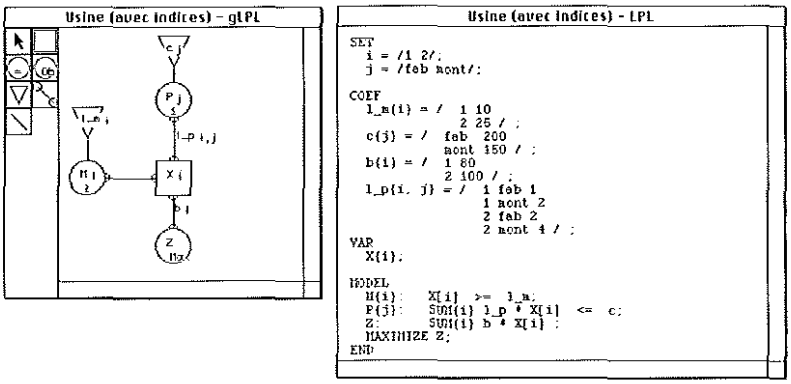

Fig. 2b.– Le modèle « Usine » avec indices; la fenêtre gauche contient le modèle gLPL et la fenêtre droite, la notation LPL.

2.2. WEBSs

WEBSs (*Woven Electronic Book System with scripts*) est un système interactif pour la création et la consultation de LE. Un LE représente un

ensemble organisé de connaissances sur un sujet donné. Il est composé d'une collection de documents de différents types, d'une série de liens entre des portions de documents, et de tables des matières qui structurent l'ensemble (*cf.* Fig. 3).

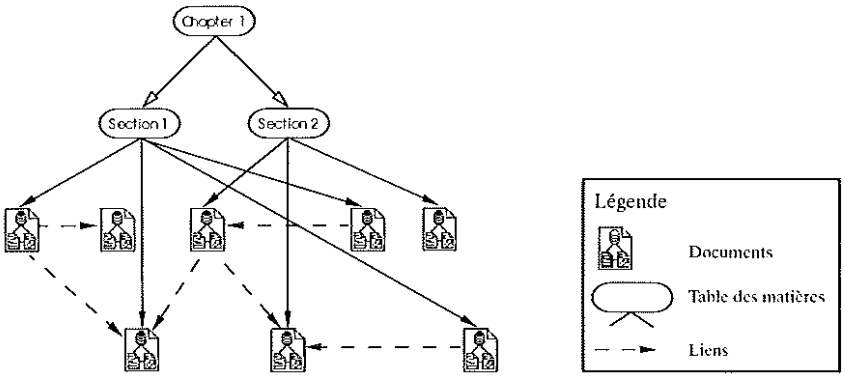

Fig. 3.– Un livre électronique est composé d'une collection de documents de type différent, de table(s) des matières qui structurent le contenu, et d'une série de liens qui ajoutent une dimension hypertexte.

Lors du développement du premier prototype (WEBS), les auteurs se sont rapidement rendus compte qu'il était nécessaire de fournir une aide devant, d'une part, permettre aux utilisateurs de ne pas se perdre dans l'hypertexte, et, d'autre part, faciliter la réalisation de certaines tâches répétitives. Dans la nouvelle version (WEBSs), cette aide a principalement pris la forme de scripts.

Dans sa variante la plus simple, un script est défini par une liste de commandes qui peuvent être mémorisées, puis exécutées automatiquement par le système comme si elles avaient été introduites une à une par l'utilisateur. Dans une variante plus complète, les scripts permettent de définir des variables, de contenir des boucles et des instructions conditionnelles et peuvent faire appel à des fonctions prédéfinies ou à d'autres scripts. En d'autres termes, un script est un programme ayant accès aux mêmes commandes que l'utilisateur final, auxquelles s'ajoutent les capacités offertes par un langage de programmation classique comme Pascal. Un script est déclenché soit manuellement, soit lorsque certaines actions (ouverture ou fermeture d'un certain type de document, suivi d'un lien, etc.) sont effectuées.

Les scripts apportent une dimension supplémentaire en ce sens qu'ils étendent de manière considérable les possibilités d'un LE en le rendant réactif aux actions de ses utilisateurs tout en respectant un cadre bien précis programmé par l'auteur. Ainsi, la fermeture d'un document de type « Tableau de bord » (*cf.* Fig. 4) entraînera la fermeture automatique des documents qui lui étaient associés; une table des matières sera automatiquement et dynamiquement créée afin d'offrir un historique des documents déjà visités par un lecteur; et les divers documents d'un LE occuperont toujours des places appropriées en fonction de la taille de l'écran et de « templates » prédéfinis par l'auteur. De même, l'auteur se servira d'une librairie de scripts définis par des spécialistes afin d'automatiser certaines tâches répétitives (création de tables des matières, génération partielle d'un nouvel exemple sur la base d'un ensemble de documents existants, etc.). La section 3 mettra en lumière l'utilité de quelques scripts, mais le lecteur intéressé est renvoyé à (Monnard, 1993) ou à (Pasquier et Monnard, 1995) pour un aperçu complet du langage de scriptage de WEBSs.

2.3. gLPS for WEBSs

Grâce à la mise en commun des capacités de modélisation de gLPS et hypertextes de WEBSs, auteurs et lecteurs de LE ont toute latitude pour créer et structurer, respectivement consulter, calculer et annoter de vastes espaces informatifs dans le domaine de l'optimisation linéaire. Il est par exemple possible de lier une variable d'un modèle avec sa définition dans un document textuel et avec sa valeur dans la solution optimale. Un lecteur peut aussi annoter certaines parties du livre en créant des liens entre ces dernières et un document contenant ses notes.

De même, l'utilisation des scripts permets une grande variété de comportements. Aussi, afin d'illustrer au mieux la nature et la richesse des avantages offerts par un tel système, la section suivante décrit, avec toutes les limitations inhérentes au support écrit, une session imaginaire avec le LE « Optimisation linéaire ».

3. Le livre électronique « Optimisation linéaire »

Le LE « Optimisation linéaire » est un effort entrepris à l'IIUF afin de créer avec gW un support électronique pour la première partie du cours « Introduction à la recherche opérationnelle » enseigné aux étudiants de deuxième cycle en économie. Ce LE est composé de trois parties : la première constitue un rappel des principaux éléments théoriques enseignés dans le cadre du cours; la deuxième présente d'une manière systématique cinq modèles concrets, dont le but est d'introduire le lecteur au processus mental permettant de transformer un problème présenté sous une forme propre aux praticiens en un modèle d'optimisation linéaire calculable; enfin la troisième est une série d'exercices à résoudre. Présentement, seule la deuxième partie est véritablement achevée — c'est-à-dire forme un tout suffisamment cohérent pour être valablement distribué et testé par des étudiants.

Afin d'illustrer notre propos, nous décrivons ci-dessous une interaction possible entre un utilisateur imaginaire (Paul) et le LE « Optimisation linéaire » :

Étape 1 : Activation de l'application

Paul active l'application gW. Le système lui demande alors d'introduire son nom d'utilisateur et de choisir un livre parmi la liste des livres électroniques disponibles. Paul choisit celui ayant pour titre « Optimisation linéaire ». À ce stade, grâce à un script déclenché automatiquement, gW affiche la table des matières générale représentée dans la fenêtre supérieure gauche de la figure 4.

Étape 2 : Lecture d'un exemple

Paul concentre son exploration sur les modèles. Il est particulièrement intéressé par le modèle « Processus de production ». Il décide de l'explorer et clique alors deux fois sur le nœud correspondant de la table des matières. Deux nouvelles fenêtres s'ajoutent alors à l'écran (*cf.* Fig. 4). La fenêtre supérieure droite contient l'énoncé du problème alors que l'autre contient un tableau de bord. Il faut noter que le système utilise un script écrit par l'auteur pour définir la disposition des fenêtres à l'écran. Par conséquent, quel que soit l'exemple choisi et quelle que soit la taille de l'écran, cette disposition reste la même : le modèle gLPL se situe toujours dans la partie

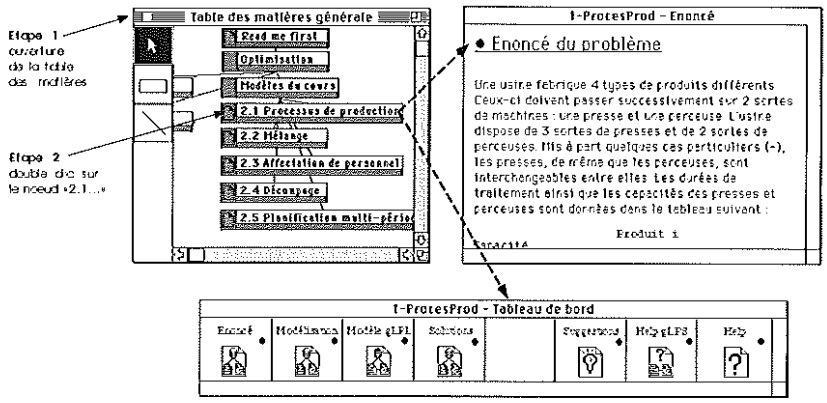

Fig. 4.– À partir de la table des matières (fenêtre supérieure gauche), en cliquant deux fois sur le nœud correspondant, Paul accède à l'exemple « Processus de production ».

supérieure gauche de l'écran, l'énoncé dans la partie supérieure droite, le processus de modélisation dans la partie inférieure droite et les valeurs de la solution dans la partie inférieure gauche (*cf.* Fig. 5). De plus, un fenêtre contenant un tableau de bord est présente pour chaque exemple. Un tableau de bord est un document à partir duquel tous les documents d'un problème peuvent être accédés. Il contient cinq liens vers les documents de l'exemple ainsi que deux liens vers des documents d'aide.

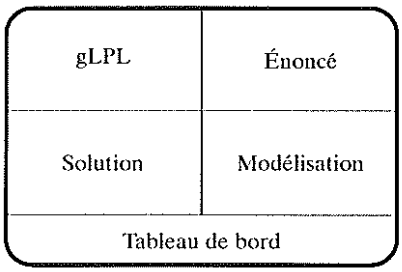

Fig. 5.– Disposition à l'écran des différents documents composant un exemple.

Étape 3 : Ouverture d'un modèle gLPL

Après avoir lu l'énoncé et essayé par lui-même de construire le modèle gLPL, Paul décide de voir le modèle proposé par l'auteur. Il sélectionne

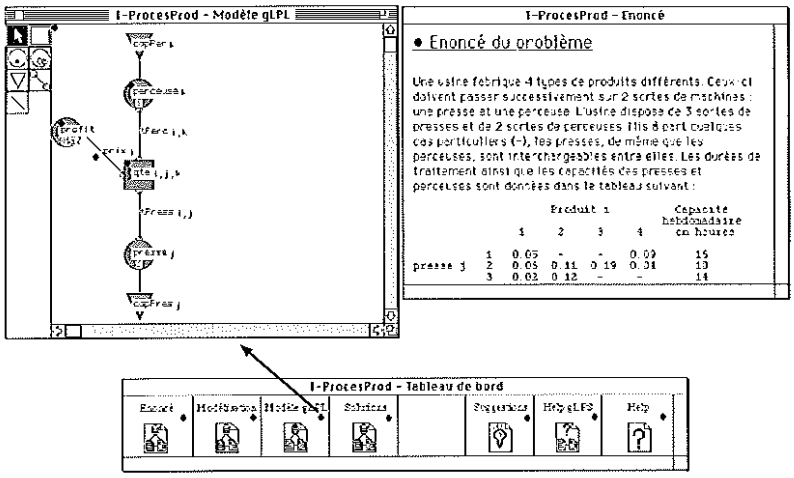

Fig. 6.– Le modèle gLPL; en cliquant deux fois sur le lien « Modèle gLPL » du tableau de bord, la fenêtre contenant le modèle gLPL apparaît à l'écran.

donc le tableau de bord et suit le lien qui lui permet de découvrir le modèle gLPL (*cf.* Fig. 6). Une nouvelle fenêtre contenant le modèle gLPL apparaît à l'écran dans la partie supérieure gauche.

Étape 4 : Exploration d'une variable

Afin de comprendre la signification de la variable `qte`, Paul suit le lien qui lui est attaché et obtient une nouvelle fenêtre contenant l'explication de la modélisation du problème (*cf.* Fig. 7).

Étape 5 : Création d'une annotation

Après avoir essayé diverses valeurs et calculé de nouvelles solutions, Paul comprend mieux la signification de la variable `qte`. Il décide donc d'associer sa propre explication à cette variable. Il sélectionne la variable présente dans le modèle gLPL et exécute le script « *Create annotation* ». gW ouvre alors son document d'annotations personnelles et crée un lien entre la variable sélectionnée (`qte`) et ce document. Paul n'a plus qu'à introduire ses commentaires (*cf.* Fig. 8).

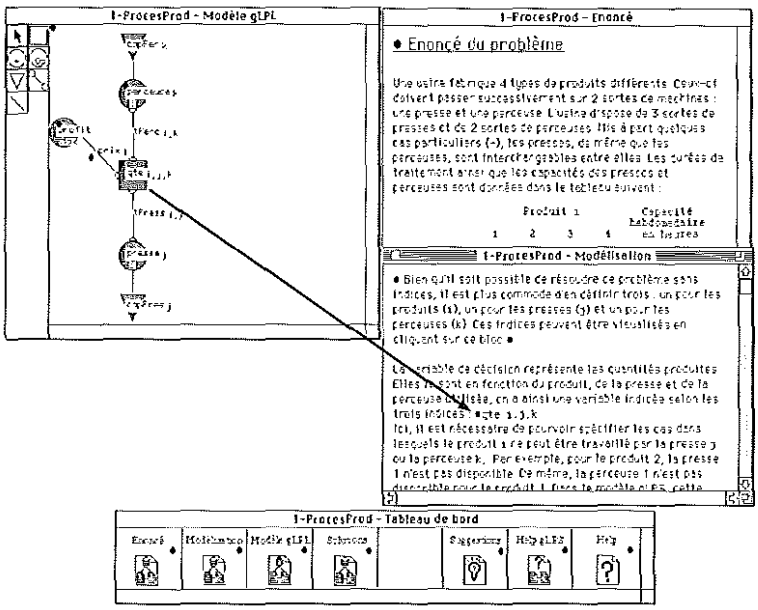

Fig. 7.– Le document de modélisation; l'activation du lien attaché à la variable `qte`, ouvre le document contenant l'explication de la modélisation du problème.

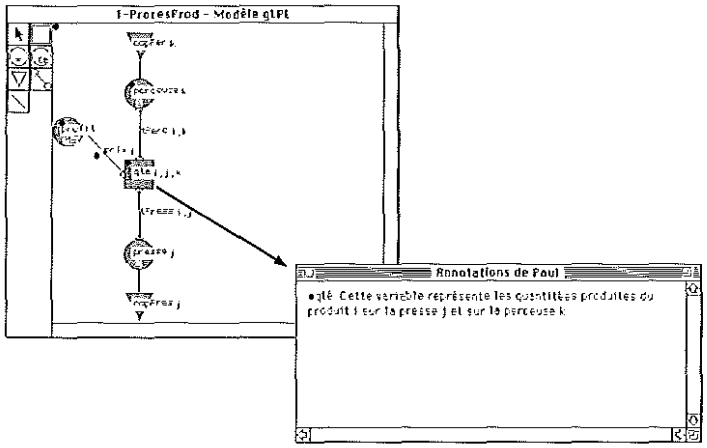

Fig. 8.– Annotations; l'activation du script «*Create annotation*» permet à Paul d'associer un commentaire à l'élément sélectionné dans son document d'annotations personnelles.

Étape 6 : Fin de l'exploration

Après avoir modifié les paramètres du modèle et calculé diverses solutions, Paul termine son exploration. gW lui demande s'il désire conserver les modifications et mémorise alors les nouveaux documents et liens.

4. Enseignements pour le futur

Le LE « Optimisation linéaire » a été utilisé avec les étudiants de deux manières complémentaires :

— d'une part, dans le cadre du cours *ex cathedra*, le professeur a enrichi sa présentation « classique » des cinq modèles mentionnés dans la table des matières de la figure 1 par des démonstrations effectuées à l'aide d'un système de projection vidéo, pour lesquelles les étudiants disposaient de copies d'écran sur papier afin de prendre des notes. En principe, pour chaque modèle, la présentation classique correspondait à environ une heure de cours et la démonstration à une vingtaine de minutes. Dans ce contexte, l'utilisation du LE s'est donc limitée à un peu moins de deux heures réparties sur quatre cours de deux heures. À une exception près, la dizaine d'étudiant(e)s interrogé(e)s dans le cadre d'un petit sondage informel a trouvé très positive cette manière de procéder;

— d'autre part, dans le cadre des séances d'exercices accompagnés, chaque étudiant a reçu un dossier comprenant (1) l'application gW, (2) une fiche de manipulation pour le familiariser avec les fonctions hypertextes de gW accompagnée d'un LE intitulé « Tutorial » spécialement conçu à cet effet, (3) une fiche de manipulation pour l'introduire à l'éditeur de modèles, (4) le LE « Optimisation linéaire » présenté ci-dessus, et enfin (5) deux énoncés d'exercices de modélisation à effectuer à l'aide de gW. Une nouvelle fois, la réaction des étudiants a été très positive quant à l'apport de gW pour définir et calculer leurs propres modèles. Les réactions, par contre, bien que dans l'ensemble favorables, furent plus mitigées concernant l'apport des fonctions hypertextes intégrées dans gW. Il est cependant intéressant de constater que plus le nombre d'heures passées avec le système est grand (en moyenne environ six heures), plus la réaction de l'étudiant aux capacités hypertextes de gW est positive.

En conclusion, notre expérience avec le LE « Optimisation linéaire » est globalement encourageante et nous comptons la renouveler avec les modifications suivantes :

— la partie théorique du LE sera complétée de façon à offrir aux étudiants une révision complète sous forme électronique de la partie du cours consacrée à l'optimisation linéaire;
— les heures de cours *ex cathedra* consacrées à l'étude des cinq modèles mentionnés dans la table des matières de la figure 1 seront remplacées par des heures de laboratoire où les étudiants devront d'abord explorer individuellement chaque modèle avant qu'ils ne soient révisés avec l'aide des assistants et du professeur. Cette manière de procéder devrait encourager les étudiants à véritablement investir du temps dans l'utilisation du système sans pour autant les surcharger exagérément;
— enfin, bien que l'application gW ne se prête pas à gérer un LE qui serait partagé simultanément par plusieurs étudiants susceptibles chacun d'y apporter sa propre contribution, nous comptons encourager ce genre d'approche d'une manière qui reste à définir (par exemple, création d'un nouveau prototype).

Remerciements

Cette recherche a été partiellement financée par le Fonds National Suisse de la Recherche Scientifique (bourse 11-37219.93).

Bibliographie

BROOKE (A.), KENDRICK (D.), MEERAUS (A.) : 1988, *GAMS a User's Guide*, (Palo Alto, CA : Scientific Press).

CHESAPEAKE DECISION SCIENCES : 1993, *MIMI: Manager for Interactive Modeling Interfaces: User's Manual*, (New Providence, NJ : Chesapeake Decision Sciences).

CHOOBINEH (J.) : 1991, «A Diagramming Technique for Representation of Linear Programming Models», *Omega*, 18 :1, pp. 43–51.

COLLAUD (G.) : 1993, *Modélisation et optimation linéaire, un système graphique de création et de gestion des modèles (gLPS)*, (Fribourg (Suisse) : Institut d'Informatique, Université de Fribourg, Thèse de doctorat).

COLLAUD (G.), PASQUIER-BOLTUCK (J.) : 1994, «gLPS: a Graphical Tool for the Definition and Manipulation of Linear Problems», *European Journal of Operational Research*, 72 :2, pp. 277–284.

COLLAUD (G.), PASQUIER (J.) : 1995, «gLPS for WEBSs: A Scriptable Object Oriented Hypertext System for Learning Linear Optimisation»,

LEARNTEC'94 (Europäischer Kongress für Bildungstechnologie und betriebliche Bildung), Tagungsband, Uwe Beck & Winfried Sommer (Hrsg.), Springer-Verlag, Berlin, pp. 309–319.

CONKLIN (J.) : 1987, «Hypertext: An Introduction and Survey», *IEEE Computer*, 2 :9, pp. 17–41.

FOURER (R.), GAY (D.M.), KERNIGHAN (B.W.) : 1987, *AMPL: A Mathematical Programming Language*, (Murray Hill, NJ : AT & T Bell Laboratories).

GEOFFRION (A.M.) : 1989, «Computer-based Modeling Environments», *European Journal of Operational Research*, 41, pp. 33–43.

GLOVER (F.), KLINGNAN (D.), PHILLIPS (N.) : 1990, «Netform Modeling and Applications», *Interfaces*, 20 :4, pp. 7–27.

GREENBERG (H.J.), MURPHY (F.H.) : 1991, *A Comparison of Mathematical Programming Modeling Systems*, (University of Colorado at Denver : Internal publication).

HAMACHER (S.), DEJAX (P.), LUSTOSA (L.), HAMACHER (P.) : 1994, *A diagram Representation for Conceptual Models of Operations Research Problems*, (Paris : École Centrale, Laboratoire Productique Logistique, Cahiers d'Études et de Recherche N° 93–04A).

HÜRLIMANN (T.) : 1993, «LPL: A Mathematical Programming Language», *OR Spectrum*, 15, pp. 43–55.

JONES (C.) : 1990, «An Introduction to Graph-Based Modelling Systems, Part I: Overview», *ORSA Journal on Computing*, 2 :2, pp. 136–151.

JONES (C.) : 1992, «Attributed Graphs, Graph-Grammars, and Structured Modeling», *Annals of Operations Research*, **38**, pp. 281–324.

JONES (C.) : 1994, «Visualization and Optimization», *ORSA Journal on Computing*, 6 :3, pp. 221–257.

MA (P.C.), MURPHY (F.H.), SROHR (E.A.) : 1989, «A Graphics Interface for Linear Programming», *Communications of the ACM*, 32 :8, pp. 996–1012.

MONNARD (J.), PASQUIER (J.) : 1992, «An Object-Oriented Scripting Environment for the WEBSs Electronic Book System», *Proceedings of the ACM Conference on Hypertext ECHT'92*, ACM Press, pp. 81–90.

MONNARD (J.) : 1993, *WEBSs, un environnement de scriptage pour hypertextes*, (Fribourg (Suisse), Institut d'informatique, Université de Fribourg, Thèse de doctorat).

PASQUIER (J.), GROSSMAN (E.), COLLAUD (G.) : 1988, «Prototyping an Interactive Electronic Book System Using an Object-Oriented Approach», *Lectures Notes in Computer Science*, 322, Springer-Verlag, Berlin, pp. 177–190.

PASQUIER (J.), MONNARD (J.) : 1995, *Livres électroniques, De l'utopie à la réalisation*, (Lausanne : Presses Polytechniques et Universitaires Romandes).

RAGHUNATHAN (S.) : 1987, *An Intelligent DSS for Model Formulation*, (University of Pittsburgh, KS : Working Paper).

SAVOY (J.) : 1987, *Le Livre électronique Ebook3*, (Bern : Peter Lang, Thèse de doctorat).

SCHRAGE (L.) : 1986, *Linear, Integer, and Quadratic Programming with LINDO*, (Palo Alto : Scientific Press, CA).

STEIGER (D.), SHARDA (R.) : 1991, *LP Modeling Languages for Personal Computers: A Comparison*, (Oklahoma : College of Business Administration, Oklahoma State University, Working Paper).

STEIGER (D.), SHARDA (R.), LECLAIRE (B.) : 1991, *Functional Description of a Graph Based Interface for Network Modeling (GIN)*, (Oklahoma : College of Business Administration, Oklahoma State University, Working Paper).